Software Risk Assessment for Windows Operating Systems with respect to CVSS

HyunChul Joh

Abstract—CVSS is recognized as a de facto standard for categorizing and measuring software vulnerabilities in both how easy for exploitation for the given security bug and how much impact on a system having the vulnerability in a sense of the three security factors. Meanwhile, since the early 2000s, quantitative risk assessments of software systems had been able to be examined thanks to the accumulated enough datasets for a scientific investigation. However, there are still a lot of research attempts not to be taken in a quantitative examination of software risk assessments. In this paper, we are quantitatively analyzing CVSS scores in vulnerabilities from the three most recent Windows products, namely, Windows 7, Windows 8.1 and Windows 10. The result shows that AML vulnerability discovery model represents Windows vulnerability discovery trend reasonably. Furthermore, we found explicitly that, most of the time, security bugs are compromised with no authentication required systems. This result is corresponding with the output from the previous research based on Web browsers.

Index Terms—Software Security Vulnerability, Windows OSes, CVSS, Quantitative Software Risk Assessment.

I. INTRODUCTION

In spite of the fact that operating systems in personal computer is one of the most important software system, still a lot of security bugs are found in all of the major operating systems such as Windows, OSX, Linux, Android, iOS etc. Most security bugs are dangerous because they provide malicious users some kind of privileges which can be used for blocking security policies. Meanwhile, operating systems are managing critique resources such as allocating CPU time slot and memory distribution to processes, and scheduling of threads when multiple tasks are required. As a result, operating systems are the most important factors in all computer systems, and because of that, some researchers say that computers can be divided into hardware, operating system, application program, and user [1].

In personal computer environment, usually, a general-purpose computer operating system such as Microsoft Windows are used. Market share, in the desktop computer environment, Windows occupies majority of the shares as shown in Table I. Windows family products are used due to the use of easiness, open policy of IBM PC architecture, compatibility among the different Windows versions and market preoccupation so that there are a lot of applications to be used.

Since operating systems act as a manager role of personal computers, it is extremely important to keep operating systems free of security flaws. And security flaws in software system, we can call them as vulnerabilities. A software vulnerability could be defined as a weakness in the security system which might be exploited by malicious users causing loss or harm [2]. Known vulnerabilities which have been discovered but not have been patched represents a security risk. As a result, newly found vulnerabilities put billions of the Net connected users at risk. Hence, significant concerns related to the possible cyber-attacks by using security vulnerabilities are very subject to increasing researchers’ attention. Not to be a victim of the cyber-attack, overall security risk stays within a certain acceptable digress. To do that managers need to check risks in their organization constantly since if you cannot measure it somehow, you are not able to improve the security environment around the organizations.

Especially, since operating systems are critical part of computing system they need more care than other types of software applications. Usually, major vendors, such as Microsoft for Windows or Apple for MAC OSX, provide automatic updates, but for Linux systems are not as simple to patch due to the nature of Linux software and its many various distributions. However, some major Linux distros can be also guided by distributors, such as Red Hat Enterprise Linux or Ubuntu, etc.

Although there have been various discussions for software of vulnerabilities from the early day of the software development, majority of the studies are about detecting and preventing of individual vulnerabilities. Moreover, in many cases, the methods have been taking a qualitative approach, some of them are quantitative manners though [3]. In this paper, we are trying to examine the CVSS score values quantitatively from the three Windows operating systems, namely, Windows 7, Windows 8.1 and Windows 10. The reason why we are selecting the three Windows OSes is that they are the only personal Windows operating systems which still are currently supported by Microsoft. In Table I, the three shaded cells represent the future time point with respect to the current time of October 2019. Also they are occupying the top three market shares. Besides, Windows are the most popular operating systems in personal computers as shown in Table II. Datasets used in this paper were mined from cvdetai.com on October 2019.

The rest of the paper is organized as follows. Section 2 presents some of the related works and section 3 reviews CVSS which requires readers to understand later section. Section 4 investigates vulnerability discovery process in the three Windows operating systems, and Section 5 analyzes each element from the CVSS base metric. Section 6 concludes this work.

Published on November 27, 2019.

HyunChul Joh is with the Department of Computer Engineering, Kyungil University, Gyeongsan City, Gyeongbuk 38428, South Korea (e-mail: joh@kiu.kr).

DOI: http://dx.doi.org/10.24018/ejers.2019.4.11.1610
TABLE I: WINDOWS OPERATING SYSTEMS

<table>
<thead>
<tr>
<th>OS</th>
<th>Release Date*</th>
<th>Support Until*</th>
<th>Market Share**</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 10</td>
<td>2015-07-29</td>
<td>2029-01-09</td>
<td>60.65%</td>
</tr>
<tr>
<td>Windows 8.1</td>
<td>2013-10-17</td>
<td>2023-01-10</td>
<td>5.22%</td>
</tr>
<tr>
<td>Windows 8</td>
<td>2012-10-26</td>
<td>2016-01-12</td>
<td>2.93%</td>
</tr>
<tr>
<td>Windows 7</td>
<td>2009-10-22</td>
<td>2010-01-14</td>
<td>29.39%</td>
</tr>
<tr>
<td>Windows Vista</td>
<td>2006-11-08</td>
<td>2017-04-11</td>
<td>0.31%</td>
</tr>
<tr>
<td>Windows XP</td>
<td>2001-10-25</td>
<td>2014-04-08</td>
<td>1.44%</td>
</tr>
<tr>
<td>Windows 2000</td>
<td>2000-02-17</td>
<td>2010-07-13</td>
<td></td>
</tr>
<tr>
<td>Windows ME</td>
<td>2000-02-14</td>
<td>2006-07-11</td>
<td></td>
</tr>
<tr>
<td>Windows NT 4.0</td>
<td>1996-08-24</td>
<td>2004-06-30</td>
<td></td>
</tr>
<tr>
<td>Windows 95</td>
<td>1995-08-24</td>
<td>2001-12-31</td>
<td></td>
</tr>
<tr>
<td>Windows NT 5.1</td>
<td>1993-07-27</td>
<td>2000-12-31</td>
<td></td>
</tr>
</tbody>
</table>

**https://gs.statcounter.com on September 2019

TABLE II: MARKET SHARE IN MAJOR OPERATING SYSTEMS

<table>
<thead>
<tr>
<th>Windows</th>
<th>OS</th>
<th>Unknown</th>
<th>Linux</th>
<th>Chrome OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>79.1%</td>
<td>14.37%</td>
<td>3.31%</td>
<td>1.74%</td>
<td>1.48%</td>
</tr>
</tbody>
</table>

* https://gs.statcounter.com on September 2019  

II. RELATED WORKS

Vulnerability discovery models describe the discovery of vulnerabilities with the passage of time. A few vulnerability discovery models have recently been proposed. One of the most well-known models is the Alhazmi-Malaiya Logistic (AML) model [3], and it was originally proposed and validated for operating systems. Joh and Malaiya [4] compares AML with other types of S-shaped vulnerability discovery Models based on the skewness in target datasets. Authors found that AML and Gamma distribution based model perform better than other S-shaped vulnerability discovery models with skewed left and right datasets respectively.

Compared with other software systems such as operating systems and office software products, newer versions of the Web browsers tend to be released faster. A new version of a software system adds new functions and implements some defect fixes. However, a new version does not necessarily imply a reduced number of vulnerabilities since the new codes can inject new vulnerabilities. Frei et al. [5] have found that a substantial number of the Internet users are exposed at risk since many portion of the users tend not to apply or tend delay patching the Web browsers and related software systems even if new patches are available. The authors quantified the dangers created by delayed patching.

Further, Duebendorfer and Frei [6] have examined the Web browser security updates for several popular Web browsers and made a conclusion that silent updates are very actual appliances. Acer and Jackson [7] raised a question that Web browsers with infrequent security patches are safer. They suggest methods for evaluating browser security which take into account new industry practices such as silent patch utilization.

Grosskurth and Godfrey [8] have used a semi-automated analysis method to investigate the architecture and evolution of web browsers. The authors have inspected different methods for code recycle, emergent domain boundaries, convergent evolutions, and debate among the open and closed source development approaches. Schryen [9] has empirically scrutinized the software vulnerability discovery processes in several systems and found that a lot of the systems have a significant linear or piecewise linear relationship between the size and the cumulative number of published vulnerabilities. However, the author did not examine further about the underlying causes of the linear growth pattern.

Meanwhile, there are several factors that impact the vulnerability discovery rate. Software code size, software age, popularity and software evolution are the most significant factors among others. Several papers [10-13] have shown the relationship between the size of software code and the number of software defects. The studies suggest that the number of defects increases with code. The first order approximation assumes a linear relationship between the code size and the number of defects, which allows definition of the concept of defect density. Since the vulnerabilities are a class of defects, a similar measure called vulnerability density [14] can be defined also.

III. COMMON VULNERABILITY SCORING SYSTEM

In July 2003, National Infrastructure Advisory Council (NIAC) commissioned a project to address the problem of multiple and incompatible IT related vulnerability scoring systems. As a result, the Common Vulnerability Scoring System (CVSS) has been adopted by many vendors since its first launch in 2005 such as application vendors, vulnerability scanning and compliance tools, risk assessment products, security bulletins, and academics [16, 17, 18]. Then, significant issues (https://www.first.org/cvss/v2/history) with the first draft had been found, which led the project to the second version CVSSv2, released in 2007. After that, in June 2015, CVSSv3.0 has been released with reflecting further considerations.

The CVSS is composed of three metric groups: base, temporal and environmental as shown in Fig. 1. The final produced score ranges from 0.0 to 10.0; scores close to 0.0 indicate more stable whereas scores close to 10.0 mean more vulnerable to exploitation and cause more serious outcome. For the final CVSS score, the base metric is must be calculated while the other two are optional.

The base metric group, ranges of [0.0, 10.0], represents the intrinsic and fundamental characteristics of a vulnerability, so the score is not changed over time. The base metric has two sub-scores of exploitability and impact sub-scores. The two sub-scores are also ranges of [0.0, 10.0]. The exploitability sub-score captures how a vulnerability is accessed and whether or not extra conditions are required to exploit it while the impact sub-score measures how a vulnerability will directly affect an IT asset as the degree of losses in confidentiality, integrity, and availability.
The exploitability sub-score is composed of four elements: access vector (AV), access complexity (AC), privileges required (PR), and user interaction (UI). AV reflects how the vulnerability is exploited in terms of Network (N), Adjacent (A), Local (L), or Physical (P). AC measures the complexity of the attack, required to exploit the vulnerability, once an attacker has gained access to the target system in terms of High (H) or Low (L). PR describes the level of privileges an attacker must process for a successful exploitation for a target vulnerability, and this metric is measured in terms of None (N), Low (L), or High (H). And, UI determines whether the vulnerability can be exploited solely at the will of the attacker, or whether a separate user must participate in some manner, and the metric has two possible values of None (N) or Required (R). Although PR and UI are introduced in CVSS version 3, frequently, many public vulnerability database does not provide the two. Instead, in many times, they give an authentication (Au) information which counts the number of times an attacker must authenticate to a target in order to exploit a vulnerability in terms of Multiple (M), Single (S), or None (N). In this paper, for the exploitability sub-score, we are analyzing AV, AC and Au.

Meanwhile, the impact sub-score is composed of the three key aspects in information security components: confidentiality (C), integrity (I) and availability (A). The impact attributes are all assessed in terms of High (H), Low (L), or None (N). A new metric has been added into the base metric CVSSv3.0, called scope (S) which has an ability capturing a fact whether a vulnerability in one software component to impact resources beyond its means, or privileges. This metric has two possible values of Unchanged (U) or Changed (C).

Before CVSS scores are entered into NVD, security experts analyze the vulnerabilities and assign one of the qualitative letter grades mentioned above on the vulnerabilities [18]. Since the central goal of CVSS is producing comparable vulnerability scores, analyzers are allowed to rate the vulnerabilities only with those letters. Finally, scoring is in the process of combining all the metric values according to the specific formulas.

The temporal metric group, ranges of [0.0, 10.0], is measured dynamically in terms of Exploit Code Maturity (E), Remediation Level (RL), and Report Confidence (RC). E measures the current state of exploit techniques or code availability, and is evaluated in terms of Not Defined (X), High (H), Functional (F), Proof-of-Concept (P) or Unproven (U). RL refers to the type of remediation available in terms of Official Fix (O), Temporary Fix (T), Workaround (W), Unavailable (U), or Not Defined (X). RC attribute refers to the confidence in the existence of the vulnerability and the credibility of the known technical details, and is evaluated in terms of Not Defined (X), Confirmed (C), Reasonable (R), or Unknown (U).

The environmental metric group, ranges of [0.0, 10.0], measures the characteristics of a vulnerability that are associated with a user’s IT environment: all related to the system environment and the stakeholders’ values. It is measured in terms of, first, Security Requirements of Confidentiality (CR), Integrity (IR), and Availability (AR), and second, modified based metrics which enable analysts to adjust the base metrics according to the modifications that exist within analysts’ environments. The security requirements are all measured in Not Defined (X), High (H), Medium (M) or Low(L), and the possible values for the Modified base metric are the same values as the corresponding Base Metric, as well as Not Defined as default value. The attributes of security requirements enable analysts to customize the score depending on the importance of the affected IT asset to a user’s organization.

The temporal metrics measure impact of developments such as release of patches or code for exploitation. The environmental metrics allow assessment of impact by taking into account the potential loss based on the expectations for the target system. The optional temporal and environmental metrics can add additional information to the base metric used for estimating the overall software risk more accurately if the required information is available. In this paper, we only utilize the base metric.

IV. VULNERABILITY DISCOVERY TRENDS

In Fig. 2., the solid S-shaped line shows the shape of the Alhazmi-Malaiya Logistic (AML) vulnerability discovery model [3]. The model is originally based on the observation that the attention given to an operating system increases as it gains market share, it peaks at some time and then drops when a newer competing version is introduced. It is assumed that the cumulative number of vulnerabilities is governed by two factors in Equation (1). The first factor increases with the time because of the rising share of the installed base. The second factors declines as the number of remaining undetected vulnerabilities declines. The saturation effect is modeled by the second factor. Assuming that the vulnerability discovery rate is given by Equation (1), Equation (2) can be obtained by solving the differential equation which gives the cumulative number of vulnerabilities \( \Omega \). Parameters \( A \) and \( B \) are empirical constants determined from the recorded data. \( C \) is a constant introduced while solving Equation (1). The model is defined for time values \( t \) from the negative infinity to the positive infinity.

\[
\omega(t) = A \Omega(B - \Omega) 
\]

(1)

\[
\Omega(t) = \frac{B}{BC^{-ABt} + 1} 
\]

(2)

During software release period, vulnerability discovery rate gradually increases due to the gaining market share.
This phase is called learning phase, as shown in Fig. 1. In the linear phase, the discovery rate reaches the maximum due to the popularity, and finally, in the saturation phase, vulnerability discovery rate slows down. The two transition points and a midpoint are mathematically defined in [19]. Even though the AML model was originally designed based on the behavior of vulnerability discovery in operating systems, with other types of software systems, the model performs well too [20, 21].

Fig. 2, Fig. 3 and Fig. 4 show the AML model fitting on the on the vulnerability discovery processes for the three Windows, and Table III contains the AML model fitting parameters from the figures with Transition point 1 (T1), Transition point 2 (T2), and Mid-Point (MP) from Fig. 2. The vulnerability discovery trends in the Windows are well represented by AML model based on the R2 value.

Meanwhile, the transition points in the table signify that Windows 7 and 10 are currently in the saturation phase as of October 2019. In other words, the two operating systems are in stable state, at least in terms of vulnerability discovery rate point of view. Whereas Windows 8.1 is still in the linear phase. However, please notice that the AML model assumes that a significant chunk of codes is not going to be introduced into a system anymore, which is not true in this case. Windows share a lot of codes between the versions.

Table IV, V and VI (Fig. 6, 7, and 8) show the number of each value in exploitability and impact sub-score groups from CVSS. For AV, all three operating systems show that access from Local have the most count. For AC, there are few High factors. Majority of them are medium or low factors. It implies that more complex systems have a lot less chances to be compromised. For Au, no operating system has multiple authentications. And many of them have zero authentications in the system. This indicates that if we have at least one authentication process in our systems, it is a lot safer than systems having no authentication. For the Impact sub-score, C (complete) takes place the highest numbers for the all three categories (Confidentiality, Integrity and Availability).
V. CONCLUSION

This paper applies the AML vulnerability discovery model on the three Windows operating systems and analyzes CVSS base scores quantitatively. The results indicate that the AML model which originally proposed and validated for operating systems are still applicable to the Windows operating systems.

Also, the results show that, almost all the time, vulnerabilities are compromises with no authentication required systems. This suggests to add authentication process in the systems. The result also reveals that exploitation aftermath is getting worse. An analogous study had been conducted by Scarfone and Mell [22] in 2009 based on 11,012 CVEs. They examined CVSS version 2 scoring system in depth without software categorizations. Also, Joh [21] conducted analogous study too with datasets from four major Web browsers.

REFERENCES


Hyunchul Joh is an assistant professor at the Department of Computer Engineering in Kyungil University, Korea, since March 2014. He is also serving as an executive director at computing information center in the same university. From 2012 to 2014, he was a GIST college laboratory instructor in division of liberal arts and sciences at Gwangju Institute of Science and Technology (GIST) in Korea. His research focuses on modeling the discovery process for security vulnerabilities and risk metrics. Recently he had started research on wearable computing. He received his Ph.D. and M.S. in computer science from Colorado State University, CO USA, in 2011 and 2007 respectively. He also received a B.E. in information and communications engineering from Hankuk University of Foreign Studies in Korea, 2005.

DOI: http://dx.doi.org/10.24018/ejers.2019.4.11.1610