Security-related vulnerability life cycle analysis

Géraldine Vache Marconato, Vincent Nicomette, Mohamed Kaâniche

Security-related vulnerability life cycle analysis

Geraldine Vache Marconato*†, Vincent Nicomette*†, Mohamed Kaâniche*‡
*CNRS; LAAS; 7 avenue du colonel Roche, F-31077 Toulouse, France
†Université de Toulouse ; INSA ; LAAS ; F-31077-Toulouse, France
‡Université de Toulouse ; LAAS ; F-31077-Toulouse, France
Email: gvache, nicomett, kaaniche@laas.fr

***Abstract*—This paper deals with the characterization of security-related vulnerabilities based on public data reported in the Open Source Vulnerability Database. We focus on the analysis of vulnerability life cycle events corresponding to the vulnerability discovery, the vulnerability disclosure, the patch release, and the exploit availability. We study the distribution of the time between these events considering different operating systems (Windows, Unix, Mobile OS), and different attributes such as the vulnerability impact on confidentiality, integrity or availability, the access vector reflecting how the vulnerability is exploited, and the complexity of the exploit. The results obtained highlight some interesting trends and behaviours, concerning, e.g. the time between the disclosure of a vulnerability and the availability of a patch or of the exploit, that are sometimes specific to the considered operating system or the vulnerability attributes. The results are also aimed at providing useful inputs to security risk assessment and modelling studies.**

I. Introduction

In the last decade, malicious activities have proliferated on the Internet and exhibited a dramatic increase in volume and diversity. In the same period, the number of detected vulnerabilities in computer software has never decreased. According to the Open Source Vulnerability Database (OSVDB) [1], the number of vulnerabilities per year since 2005 was always higher than about 7400.

In order to build efficient protection mechanisms, the security systems developers need to take into account the vulnerabilities that might affect their system and the real threats observed in an operational context. Thus, it is important to collect and monitor real world data related to vulnerabilities and attacks, and analyse them to better understand and assess the actual risks encountered by a particular computer system when being targeted by these threats. This paper focuses in particular on the analysis of security-related vulnerabilities reported in public databases in order to extract relevant information that can be used e.g., to support quantitative security modelling and risk assessment studies.

Security-related vulnerabilities are regularly tracked by public and industrial companies and are recorded in several well-known databases such as the previously mentioned OSVBD, the National Vulnerability Database [2] managed by the NIST, or the Security Focus database [3] managed by Symantec Corporation. These databases currently contain a large amount of data that can be used to identify significant trends about vulnerability attributes and patterns. We argue that there is a lack of deep analysis of these vulnerabilities, in order to better characterize them. Indeed, the vulnerabilities and their characteristics may be very different from several points of view: e.g. the operating system they target, the damage they can provoke and especially their life cycle. The life cycle is defined as a set of events that occur during the life of a vulnerability such as the vulnerability disclosure, the exploit disclosure, the patch release, etc. To date, only a few studies have been published on the characterization of vulnerabilities and to the best of our knowledge, past research did not analyse the link between the vulnerability life cycle events and other vulnerability attributes, such as the type of the operating system, the severity of the vulnerability, etc.

The results presented in this paper are aimed at addressing this objective. In particular, we focus on the statistical analysis of the empirical distribution associated to the time intervals between the occurrence of vulnerability life cycle events. Combined analyses are carried out taking into account different types of operating systems (Windows, Unix, Mobile OS) and different vulnerability attributes such as the vulnerability impact on confidentiality, integrity or availability, the access vector reflecting how the vulnerability is exploited, or the complexity of the exploit. This paper tries to answer questions such as: i) what is the order of magnitude of the mean times between the considered vulnerability life cycle events? ii) do Unix, Windows and Mobile OS vulnerabilities exhibit similar vulnerability life cycles ? iii) Do the observed times between life cycle events distributions exhibit different patterns depending on the vulnerability characteristics (impact, complexity, etc.)? The results of this paper are based on the data recorded in the Open Source Vulnerability Database that contains more than 69000 vulnerabilities recorded since 1980. The methodology used to select the data needed for the different analyses and the main lessons learned are detailed in the paper.

We believe that the knowledge derived from the combined analysis of life cycle events with other vulnerability characteristics provides useful insights to the security administrators and designers. Indeed, by providing specific and real data related to their system, they can have a better idea about the actual risks faced by their system. This information is also useful to the research community dealing with the development of quantitative security assessment models (see e.g., [4]). Indeed, the statistical distributions associated to the vulnerability life cyle events presented in this paper can be used to estimate some of the parameters used in these models.

This paper is structured as follows. Section II is dedicated to the vulnerability life cycle presentation. Section III presents

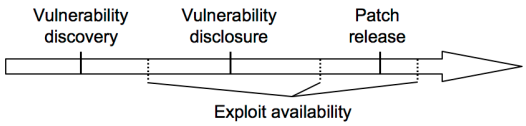

Fig. 1. The vulnerability life cycle events

the related work about vulnerability life cycle study. Section IV presents an analysis of different vulnerability databases, explains how we selected one of them for our experiments and then focuses on the methodology we adopted and the characteristics we chose to study. Section V presents the obtained results. Finally, Section VI concludes this paper.

II. The vulnerability life cycle

The vulnerability life cycle is defined as the set of events that occur during the life of the vulnerability. Several definitions exist, taking into account more or less events (see e.g., [5], [6], [7], [8], [9]). In this section, we review some of these definitions. To our knowledge, [5] is among the first studies that defined the vulnerability life cycle as a set of events. The authors included the birth, the discovery (merged with the birth event if the vulnerability is an intentional fault), the vulnerability disclosure (defined as the information dissemination only for security insiders), the patch release, the full disclosure of the vulnerability (more global than in the case of the disclosure event), the exploit availability, and the death of the vulnerability. The vulnerability full disclosure and death events are not considered in [7]. Moreover, it is assumed that the exploit availability could occur only after the vulnerability disclosure and before the patch release. In [8], the author considers in addition the publicity event, defined as the moment when the already disclosed vulnerability is known by a large population. This event is equivalent to the full disclosure event. Additional events are defined in [6], including in particular the vulnerability rediscovery and the information propagation. More details are also considered in the definition of the exploit event (considering the exploit concept discovery, the successful attack event and the exploit automation) and the patch release event (distinguishing the patch application and the complete patch release).

Some of the events considered in these different life cycle definitions are actually difficult to characterize and to date. In our study, we decided to take into account only the main events that are recorded in public databases (cf. Figure 1):

- the *vulnerability discovery*: once this event has occurred, the discoverer knows about the vulnerability existence and can use this knowledge for a malicious or non malicious purpose. The corresponding date is denoted t_0;
- the *vulnerability disclosure*: this is the first time the information about the vulnerability is freely available on an important source. This vulnerability has generally been studied by experts for risk evaluation. The corresponding date is denoted t_d;
- the *patch release*: once this event has occurred, it is possible to protect the system by removing the vulnerability. The corresponding date is denoted t_p;
- the *exploit availability*: this event enables the attacker population to exploit the vulnerability and may occur at anytime after the discovery. The exploit may be elaborated by the attackers or may correspond to the reuse of the proof of concept disclosed at the same time as the vulnerability. The corresponding date is denoted t_e.

The next section discusses related work dealing with the characterization of these events.

III. Related work

A few studies have addressed the characterization of some of the vulnerability life cycle events discussed in the previous section. The study reported in [10] focused on the impact of the vulnerability disclosure on the attack process, considering a set of 308 vulnerabilities. The statistics obtained are used to parameterize an economical model assessing the evolution of the number of attacks per host and per day. In [11], the authors studied a set of 240 vulnerabilities for which the life cycle is complete. Each vulnerability life cycle is classified into the *Zero-day attack* category, or the *Potential for attack* category which groups the vulnerabilities for which an exploit exists and there is no patch. Then, a metric is evaluated for each of these categories considering eight attributes such as the age of the vulnerability, the risk, or the exploit existence. The results highlighted, for example, the lack of patch for Microsoft vulnerabilities. A larger set of data including 14326 vulnerabilities from several databases was investigated in [12]. This work focused on the characterization of the probability distribution of the time interval between the vulnerability patch release and the exploit availability events, relatively to the vulnerability disclosure event. This study did not consider the specific characteristics and attributes of the vulnerabilities, as carried out in this paper. The authors continue this work by defining the 0-day patch metric, which measures the proportion of vulnerability for which the patch and the vulnerability are disclosed at the same time [13]. Some organisms study the security and vulnerability life cycle impacts. For instance, the french Clusif association publishes a cybercriminality panorama every year [14] and the Symantec company publishes a Security threat report twice a year, which discloses duration between vulnerability life cycle events.

More recently, in [15], the authors analysed security incidents recorded from real computing systems. These incidents are grouped in nine categories and are linked to vulnerability classes. However, these analyses aimed at evaluating the detection effectiveness of the tools used in their experiments and are not correlated to the life cycle of the vulnerability. Experimental studies dealing with the analysis of the vulnerability life cycle events and the correlation between the vulnerability life cycle and the characteristics of the vulnerabiliy are seldom reported in the literature. We can mention in this context our previous work reported in [16]. This work aimed at characterizing the probability distributions associated to the occurrence of the vulnerability life cycle events, that are then used to parameterize a stochastic model allowing the quantitative assessment of security measures. The life cycle events are characterized taking into account a global set of

Total	68195
Discovery date (t_0)	4690
Disclosure date (t_d)	68191
Patch disclosure date (t_p)	7764
Exploit availability date (t_e)	22823

TABLE I
SUMMARY OF THE OSVDB DATABASE

vulnerabilities, without distinguishing their specific attributes such as the type of operating system, the severity of the vulnerability, etc.

The results presented in this paper aim at taking into account these specific attributes in the analysis of the time intervals between the vulnerability life cycle events.

IV. DATASET AND ANALYSIS APPROACH DESCRIPTION

This section is structured as follows. Subsection IV-A presents different freely available vulnerability databases and the motivations that led us to select one of these databases for our analyses. Subsection IV-B outlines the vulnerability characteristics investigated in our study. Finally, Subsection IV-C presents the analysis approach that we followed.

A. The vulnerability database

We studied different databases that are freely available in order to choose the most suitable one for our work. These included the National Vulnerability Database [2] managed by the NIST, the Security Focus database [3] managed by Symantec Corporation, the Open Source Vulnerability Database (OSVDB) [1] created by the Black Hat conference participants, and the Secunia database managed by the Secunia company. Among these, OSVDB is the one that contains the most comprehensive and complete information needed in our study to carry out the combined analysis of the vulnerability life cycle events taking into account some characteristics of the vulnerabilities. This database contains more than 69000 vulnerabilities, recorded since 1980. The information recorded includes the disclosure date, the discovery date, the patch date, the exploit availability date, the list of the vulnerable components, and the CVSS (Common Vulnerability Scoring System) score.

The results presented in this paper are based on the data available in the OSVDB database up to 16/11/2010. This database contains 68195 vulnerabilities. Unfortunately, as summarized in Table I, all the data we are interested in are not recorded for every vulnerability. As an example, there are 68191 vulnerabilities for which the disclosure date is recorded, and only 4690 vulnerabilities with the discovery date.

B. Vulnerability categories

The vulnerability characteristics included in our study concern the operating systems affected by the vulnerability, the software company which developed the component and some attributes of the CVSS score covering the vulnerability impact, the access vector, and the access complexity. More details are provided hereafter.

1) The operating system: We analysed in particular the following operating systems:

- Windows: the vulnerability is in the Windows OS or a Windows application;
- Unix: the vulnerability is in a Unix family OS or a Unix application;
- Mobile OS: the vulnerability is in a mobile device OS (IOS, Android, Palm OS, Blackberry OS, Windows Mobile) or in a mobile device application.

2) Software vendors: We analysed also the vulnerability life cycle according to two important software vendors : Adobe and Microsoft software companies. The purpose of this analysis is to identify strategies of the software vendors, regarding the disclosure of the vulnerability for instance.

3) The CVSS attributes: This scoring system includes several metrics [17], covering the intrinsic attributes of a vulnerability, other characteristics related to time and to the environment. In our study, we selected the following metrics:

- The *Access Vector* reflects how the vulnerability is exploited. The possible values for this metric are: *Local, Adjacent Network* and *Remote*.
- The *Access Complexity* measures the complexity of the attack required to exploit the vulnerability. The possible values for this metric are: *High, Medium* and *Low*.
- The *Confidentiality (resp. Integrity, Availability) Impact* measures the impact on confidentiality (resp. integrity, availability) of a successfully exploited vulnerability, with three possible values: *None, Partial, Complete*.

C. Methodology

For each category, we selected vulnerabilities from the database for which information is available about the life cycle events occurrence dates allowing the evaluation of the time intervals t_d-t_0, t_p-t_d, t_e-t_p, t_e-t_d and t_e-t_0. Even if the date of an event is available, the time interval obtained can be invalid. In particular, the time intervals t_p-t_d, t_d-t_0 and t_e-t_0 from the automatic processing of the database must be positive: as we explained in Section II, the patch release cannot occur before the vulnerability disclosure, and the vulnerability disclosure and the exploit availability must occur after the vulnerability discovery.

V. RESULTS

For each set of vulnerabilities (classified according to OS or CVSS attributes), we analysed the time intervals presented in Section IV-C in order to find interesting correlations or common trends. These analyses for operating system category and CVSS attributes are respectively presented in Sections V-A and V-B. Then, the main conclusions of these analyses are summarised in Section V-C.

A. Analysis according to the operating system category

The Windows, Unix and Mobile OS datasets contain respectively 1527, 1809, and 239 vulnerabilities. The small number of mobile OS vulnerabilities is related to the fact that these OS are quite recent, and as a consequence, less exploited than

Time interval	before 2001	2001-05	2006-10	total
t_p-t_d	1	12	124	137
t_d-t_0	7	71	29	107
t_e-t_p	1	23	32	56
t_e-t_d	25	165	126	316
t_e-t_0	3	41	9	53

TABLE II
VULNERABILTY SETS - WINDOWS CATEGORIES

Time interval	before 2001	2001-05	2006-10	total
t_p-t_d	1	6	72	79
t_d-t_0	8	99	9	116
t_e-t_p	1	4	6	11
t_e-t_d	64	156	65	285
t_e-t_0	3	28	4	35

TABLE III
VULNERABILTY SETS - UNIX CATEGORIES

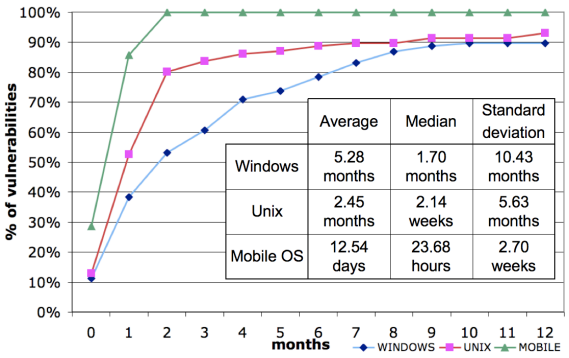

Fig. 2. t_d-t_0 - OS categories

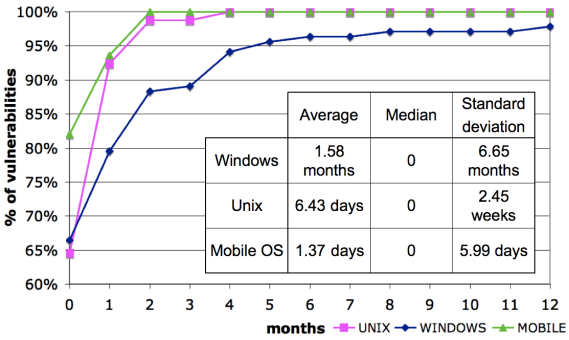

Fig. 3. t_p-t_d - OS categories

Windows and Unix. Tables II, III and IV present the number of Windows, Unix and mobile OS vulnerabilities for which information is available to compute the interval t_p-t_d, t_d-t_0, t_e-t_p, t_e-t_d or t_d-t_0. These numbers are given for the whole data set, and

Time interval	before 2001	2001-05	2006-10	total
t_p-t_d	0	0	40	40
t_d-t_0	0	3	4	7
t_e-t_p	0	0	9	9
t_e-t_d	2	9	34	45
t_e-t_0	0	0	0	0

TABLE IV
VULNERABILTY SETS - MOBILE OS CATEGORIES

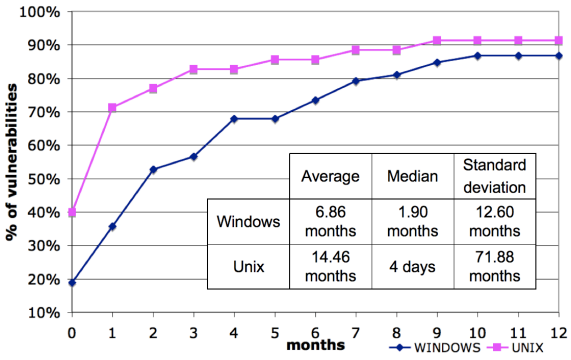

Fig. 4. t_e-t_0 - OS categories

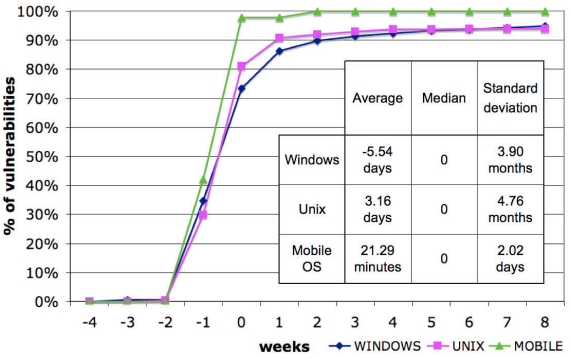

Fig. 5. t_e-t_d - OS categories

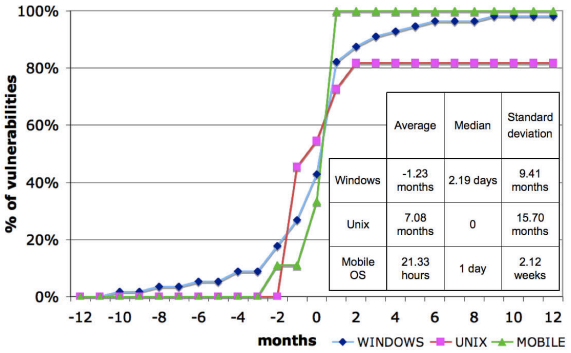

Fig. 6. t_e-t_p - OS categories

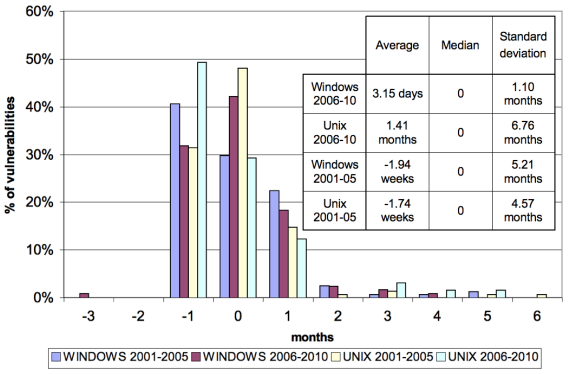

Fig. 7. t_e-t_d - OS categories

also for different periods of time, corresponding, respectively,

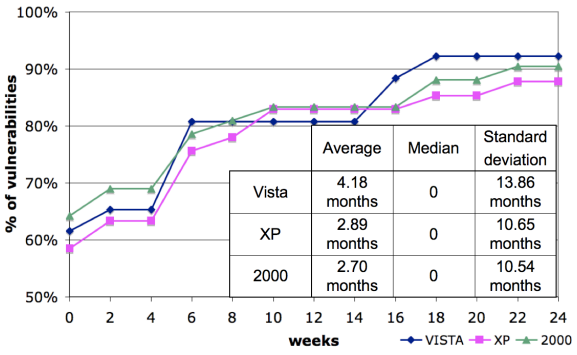

Fig. 8. t_p-t_d - Windows versions

to the vulnerabilities recorded before 2001, between 2001 and 2005 and since 2006. For instance, Table II shows that the OSVDB database includes 137 Windows vulnerabilities for which t_p and t_d (and consequently t_p-t_d) are available. 124 out of these 137 vulnerabilities are recent (from 2006 to 2010). Figure 2 represents the cumulative distribution (CDF) of t_d-t_0 for each OS category. The cumulative distribution is not entirely depicted on the figure to highlight the most relevant results. Detailed statistics about the median, the average and the standard deviation are also provided. Considering the median values, Figure 2 shows that the mobile OS vulnerabilities are disclosed faster (23.68 hours) than the Windows (1.7 month) and Unix (2.18 weeks) vulnerabilities. According to the time interval t_p-t_d (see Figure 3), it appears that the mobile OS vulnerabilities patchs are published as soon as vulnerabilities are disclosed (the same day for 80% of the vulnerabilities). These two results mean that there is a high reactivity of the security community for this category of vulnerabilities. On the other hand, for Windows category, even if the patch is mainly released as soon as the vulnerability is disclosed (66.4%), a higher delay is generally observed between the discovery and the disclosure of the vulnerabilities. This seems to indicate that the disclosure of Windows vulnerabilities is delayed to hide the existence of the vulnerabilities to attackers until the corresponding patch is ready. Nevertheless, the patch for Unix vulnerabilities is generally released faster than for Windows (6.43 days compared to 1.58 months in average).

Figures 4 to 6 are related to the exploit availability. Figure 4 represents t_e-t_0 for Windows and Unix (this information is not available for mobile OS vulnerabilites). Considering the median values, surprisingly, this figure shows that Unix vulnerabilities are, for most of them, exploited sooner and faster than the Windows vulnerabilities (4 days compared 1.9 months). It is noteworthy that the time intervals recorded for Unix exhibit a high variability illustrated e.g. by an average value around 14.46 months compared to 6.86 months for Windows. Figure 5, depicting t_e-t_d, shows that, for the three OS categories, many vulnerabilities are exploited the same day when they are disclosed (38.6% for the Windows, 51.2% for Unix and 55.6% for mobile OS). This result shows the reactivity of the attackers community. Figure 6 representing t_e-t_p shows different trends for the three OS categories. The mobile OS vulnerabilities (for which we noticed that the patch

Time interval	Windows				Unix		
	Seven	Vista	XP	2000	Linux	Open BSD	Free BSD
t_p-t_d	1	26	41	42	10	2	0
t_d-t_0	0	1	36	34	28	21	9
t_e-t_p	1	7	26	22	2	0	1
t_e-t_d	1	9	93	83	82	26	63
t_e-t_0	0	0	21	21	15	4	9

TABLE V
VULNERABILTY SETS - WINDOWS AND UNIX VERSIONS

is released very quicky) are, for most of them, exploited after the patch release (21 hours in average). However, we have to carefully consider the results related to Mobile OS because the vulnerability set contains only a few data. For Unix vulnerabilities, 45.5% are exploited before the patch release (the month preceding the patch release for most of them). A different behavior is observed for Windows for which only 26.7% of the vulnerabilities are exploited before the patch release, and in some cases the patch is released more than ten months after the exploit.

1) Time evolution: Let us focus now on the time between the exploit availability and the vulnerability disclosure date, for Windows and Unix, considering in particular the data recorded for the two periods: from 2001 to 2005 and since 2006 (see Tables II, III and IV). The set of vulnerabilities disclosed before 2001 is not large enough to obtain representative results. Comparing the mean time between the discovery and the disclosure of vulnerabilities obtained for periods (2001-2005) and (2006-now), we observe the same decreasing trend for the two OS categories: from 6.33 to 4.62 months for Windows and from 10.6 to 7.4 months for Unix.

A different behavior is observed when considering the mean time between the disclosure of a vulnerability and the availability of the exploit. Figure 7 shows that t_e-t_d time interval is negative for many Windows vulnerabilities for the 2001-2005 period. The trend is reversed during the period after 2006 for which the majority of Windows vulnerabilities are disclosed before the first exploit is available. In this scenario, the security administrators can be alerted on time and they have the possibility to take security measures to protect their system before the exploit is available. The mean time between the disclosure of a vulnerabilty and the availability of the exploit is higher for Unix (1.41 months) than for Windows (3.15 days). Thus, the Windows vulnerabilities are in average exploited much faster than the Unix vulnerabilities. This can be explained by the fact that Windows systems are far more widespread in the world and are generally more often targeted by the attackers compared to Unix systems.

2) Windows and Unix versions analysis: Table V summarizes the vulnerability life cycle data for the different Windows versions including Windows 7, Vista, XP and 2000. The number of vulnerabilities for Windows 7 is not significant. For the other versions, we have observed that the distributions of the time intervals between life cycle events generally exhibit similar trends, with some small variations. This could be due to the fact that several vulnerabilities are present in some applications that can run on several Windows versions. As an example, Figure 8 plots the distribution of the time

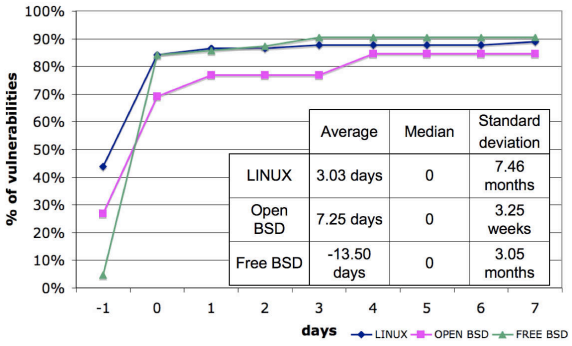

Fig. 9. t_e-t_d - Unix versions

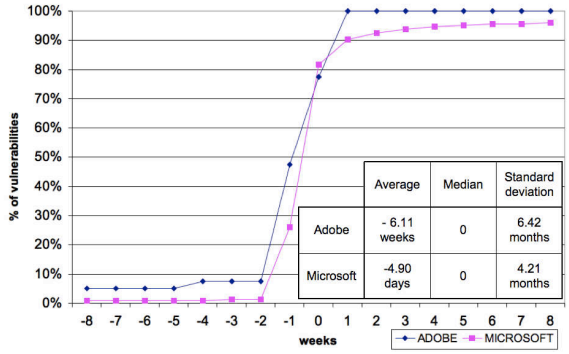

Fig. 10. t_e-t_d - Adobe and Microsoft products

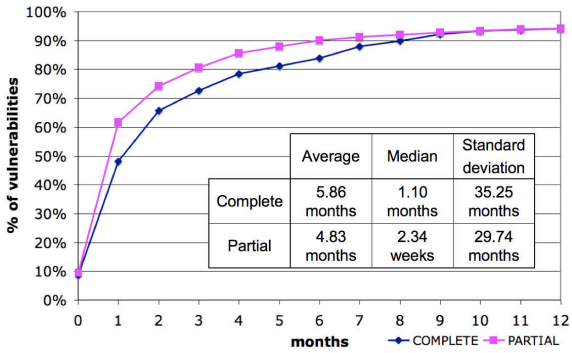

Fig. 11. t_d-t_0 - CVSS impact level

between vulnerability disclosure and patch release events. Another example concerns the time between the vulnerability disclosure and the exploit availability. We have observed that the percentage of t_e-t_d intervals which are lower than one month is 66.7% for Vista, 55.8% for XP and 51.1% for 2000. This means that Vista is the Windows version for which the exploits are available the fastest. It is noteworthy that when a vulnerability affects several versions, it is counted for all these versions.

As regards Unix, we analysed Linux, FreeBSD and OpenBSD vulnerabilities. The vulnerability sets are presented is Table V. The results highlight different scenarios. Let us consider for example the time interval t_e-t_d. Figure 9 shows that for about 80% of the vulnerabilities, the OpenBSD developers are very reactive: the t_e-t_d interval mean is 1.04 weeks, with a null median and the t_d-t_0 interval is the smallest of the three categories: 3.89 days mean and 10.29 hours

Time interval	Adobe	Microsoft
t_p-t_d	10	104
t_d-t_0	29	179
t_e-t_p	4	57
t_e-t_d	40	510
t_e-t_0	18	104

TABLE VI
VULNERABILTY SETS - ADOBE AND MICROSOFT VENDORS

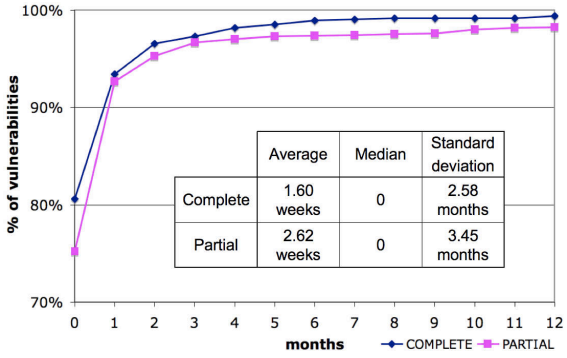

Fig. 12. t_p-t_d - CVSS impact level

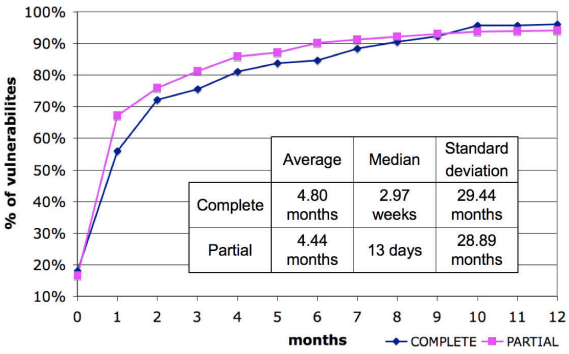

Fig. 13. t_e-t_0 - CVSS impact level

median. The Linux vulnerabilities have time intervals with a higher standard variation compared to the other.

3) Adobe and Microsoft vendors analysis: In this section, we analyse two sets of vulnerabilities included in software from two different companies: Microsoft and Adobe. These sets are summarised in Table VI. Figure 10 depicts the different values of the t_e-t_d time interval for these two sets. This figure shows that the mean of this time interval is negative for both software companies. This means that, in average, for both of them, the exploit is available before the vulnerability is disclosed. For vulnerabilities in Adobe software, this time interval may be very important (several months). This analysis seems to emphasize a common strategy of these two vendors: keep the vulnerability secret as long as the corresponding patch is not available so that they can publish at the same time the vulnerability and the patch. This hypothesis seems to be confirmed by the analysis of the t_p-t_d time interval (not depicted in the paper). Indeed, this analysis shows that 60% of the Microsoft vulnerabilities and 40% of the Adobe vulnerabilities are disclosed the same day as the patch is released.

Time interval	Maximal impact			Network access			Complexity		
	Complete	Partial	None	Remote	Adjacent	Local	High	Medium	Low
t_p-t_d	1314	2412	3	3353	16	360	139	1862	1728
t_d-t_0	532	3192	1	3335	2	388	224	1019	2482
t_e-t_p	174	733	0	863	0	44	31	432	444
t_e-t_d	1896	16973	4	17887	10	976	1050	5738	12085
t_e-t_0	234	1699	0	1770	1	162	97	592	1244

TABLE VII
VULNERABILTY SETS - CVSS MAXIMAL IMPACT, ACCESS AND COMPLEXITY LEVELS

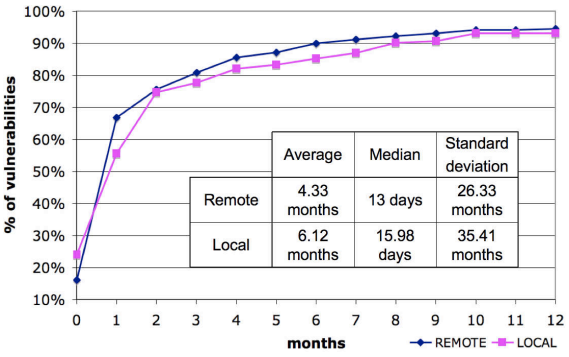

Fig. 14. t_e-t_0 - CVSS access level

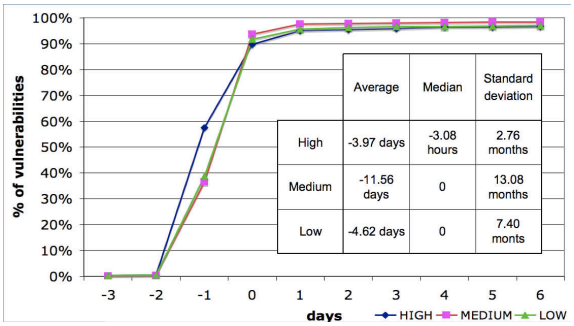

Fig. 15. t_e-t_d - CVSS Complexity level

B. Analyses based on the CVSS attributes

We classified the vulnerabilities according to the following CVSS metrics: the *Access Vector*, the *Access Complexity*, the *Confidentiality Impact*, the *Integrity Impact* and the *Availability Impact* metrics.

1) Maximal impact: We classified the vulnerabilities according to the maximum value of these three last metrics, which we denote as *maximal impact* (e.g., a vulnerability is classified into the *Complete* maximal impact category if at least one of its three impact metrics has the *Complete* value). These should correspond to the most severe vulnerabilities. The corresponding vulnerability sets are represented in Table VII. The set including vulnerabilities with *None* maximal impact does not contain enough vulnerabilities to obtain significant results. For the other sets (*Complete* or *Partial* maximal impact), t_d-t_0 is depicted in Figure 11 and shows that the majority of vulnerabilities of both categories are disclosed after two weeks. However, the vulnerabilities of the *Partial* maximal impact category are disclosed faster than the vulnerabilities of the *Complete* maximal impact category. Figure 12 shows that more vulnerabilities from *Complete* maximal impact category are patched the same days when the vulnerability is disclosed. Nevertheless, the overall trends displayed in Figures 11, 12 and 13 for the *Complete* and *Partial* impact categories look similar.

The vulnerabilities with the complete impact have the same kind of life cycle as the Windows vulnerabilities we analysed in this section: the disclosure seems to be delayed as much as possible so that attackers community cannot learn the vulnerability existence before the corresponding patch is ready. It is not the case for the vulnerabilities that have a *Partial* impact, as they are less critical. However, the ratio of vulnerabilities of *Partial* maximal impact category disclosed and patched the same day is high as well. The t_e-t_d analysis shows that, for more than 40% of *Partial* maximal impact vulnerabilities, the exploit is available before the vulnerability disclosure (vs. 28% for the high impact vulnerabilities).

2) Access level: The *Access Level* metric describes whether the vulnerability can be exploited from a *Remote*, *Adjacent* or *Local* network. Table VII represents the vulnerabilities available for these three categories. It appears first that most of the recorded vulnerabilities can be remotely exploited. Moreover, Figure 14 shows that these vulnerabilities are exploited faster after their discovery than the vulnerabilities of the *Local Access* category. It appears that the attackers community exploiting remote vulnerabilities are more reactive than the attackers exploiting the local vulnerabilities. That could be explained by the fact that there are more attackers looking for exploiting remote vulnerabilities than local ones. The same trend is observed for the other time intervals. Nevertheless the difference is not very significant as the median values are generally of the same order of magnitude.

3) Complexity level: The *Complexity Level* metric measures the complexity of the attack required to exploit the vulnerability. The possible measures are *High*, *Medium* or *Low*. Table VII represents the available vulnerabilities for each category. An interesting result of this experiment is the trend concerning t_e-t_d, depicted in Figure 15. It shows that, for the majority of the vulnerabilities with a *High Complexity Level* metric, the exploit is available before the vulnerability is disclosed (57.7%). For the majority of the vulnerabilities with a Medium Complexity Level, the exploit is available the same day as the disclosure of the vuln (57.1%). This is surprising because this means that complex vulnerabilities are exploited faster than simple vulnerabilities. Furthermore, this means that a lot of *Zerodays* exploits exist for Medium and complex vulnerabilities. These exploits are particularly interesting for attackers because they consist in exploiting vulnerabilities either unknown by the developers of the software, or for which the corresponding patch does not exist yet. Considering the whole database, we have identified 822 vulnerabilities corresponding to *Zeroday* exploits. For these vulnerabilities, we analysed the *maximal impact* and *Access Complexity* metrics. The analysis shows

that: 1) all these vulnerabilities have at least *Partial impact* on the system (107 have a *Complete impact* and 712 have a *Partial impact*), and 2) most of these vulnerabilities are easy to exploit (511 have a *low Complexity level*, 285 a *medium Complexity Level* and 26 a *high Complexity Level*). The latter result is surprising because *Zeroday* exploits usually include complex malware dedicated to security experts.

C. Summary

These analyses highlighted a few interesting results:

- The mobile OS vulnerabilities have a short life cycle (the average time interval between the discovery and the patch release is around 14 days): this reflects the high reactivity, from both attacker community and developers.
- The Windows vulnerability life cycle exhibit different characteristics when compared to Unix vulnerability life cycle as it seems that the disclosure of the vulnerability is delayed as much as possible to be close to the patch release time. This may be explained by the widespread of computers running Windows combined with the fact that: 1) many users are not aware about security risks and will not be careful even when the vulnerability is disclosed and 2) many attackers mainly target Windows vulnerabilities.
- The time between vulnerability disclosure and patch release decreases with years for both Unix and Windows. This result is confirmed by the fact that more recent Windows version vulnerabilities are patched faster.
- The high impact vulnerabilities and the Windows vulnerabilities have similar life cycles. It is likely that the developers try to delay the vulnerability disclosure and to disclose the vulnerability and the patch at the same time, as these vulnerabilities are particularly critical.
- The remote access vulnerabilities are exploited faster than the local vulnerabilities.
- The complexity has a surprising effect on the exploit availability: the more complex is the vulnerability exploit, the faster this exploit is available.
- The *Zeroday* exploits have a high impact on the system but the corresponding vulnerabilities are not necessarily complex to exploit.

VI. Conclusion

In this paper, we presented experimental results from our analysis of the Open Source Vulnerability Database. The aim of our analysis is to highlight correlations between some vulnerability characteristics and the vulnerability life cycle. We selected the vulnerabilities from the database considering several characteristics: the operating system and the CVSS base metric group attributes. We analysed each set of vulnerabilities obtained and highlighted the differences of vulnerability life cycles. These results could be useful for administrators to analyse the security risks for their system.

The analyses presented in this paper provide some initial interesting trends but they need to be refined. For instance, we considered globally the vulnerabilities targeting a particulier operating system (Unix or Windows). It would be interesting to distinguish vulnerabilites targeting the kernel of the OS and the vulnerabilites targeting the applications. It would also probably be interesting to distinguish vulnerabilites targeting software from commercial or non commercial developers. We can imagine that the patch release strategy may be different for commercial and non commercial software. We are currently investigating these directions.

This work presents several perspectives. The first one is to characterize these new sets of vulnerabilities by probabilistic distributions. The second perspective is to highlight new characteristics influencing the vulnerability life cycle by analysing sets of vulnerabilities corresponding to a same vulnerability life cycle. The third perspective is to analyse these sets of vulnerabilties considering a multidimensional analysis including several vulnerability characteristics. These analyses may explain the differences between the life cycles we observed: for example, taking into account not only the impact level but also the complexity of the attack could explain why the partial impact level vulnerabilities have an exploit available before the high impact level vulnerabilities. Finally, one of our perspectives is to provide a vulnerability database as complete as possible, from several public vulnerability databases, such as NVD. The dataset provided by a such database will be more complete and enable us to proceed to in-depth analyses. This is actually an on going work.

References

[1] O. S. Foundation, "Open source vulnerability database," http://osvdb.org.

[2] N. I. fo Standards and Technology, "National vulnerability database," http://nvd.nist.gov.

[3] S. Focus, "The bugtraq database," http://www.securityfocus.org.

[4] G. Vache, "Environment characterization and system modeling approach for the quantitative evaluation of security," in *Proceedings of the 28th International Conference on Computer Safety, Reliability and Security*, 2009, pp. 89–102.

[5] W. Arbaugh, W. Fithen, and J. McHugh, "Windows of vulnerability: a case study analysis," *Computer*, vol. 33, no. 12, pp. 52–59, Dec 2000.

[6] N. Fischbach, "Le cycle de vie d'une vulnrabilit," http://www.securite.org, 2003.

[7] E. Rescorla, "Is finding security holes a good idea?" *Security & Privacy, IEEE*, vol. 3, no. 1, pp. 14–19, Jan.-Feb. 2005.

[8] J. Jones, "Estimating software vulnerabilities," *Security & Privacy, IEEE*, vol. 5, no. 4, pp. 28–32, July-Aug. 2007.

[9] S. Frei, "Security econometrics - the dynamics of (in)security," ETH Zurich, Dissertation 18197, ETH Zurich, 2009.

[10] A. Arora, R. Krishnan, R. Telang, and Y. Yang, "Impact of vulnerability disclosure and patch availability - an empirical analysis," in *In Third Workshop on the Economics of Information Security*, 2004.

[11] A. Jumratjaroenvanit and Y. Teng-amnuay, "Probability of attack based on system vulnerability life cycle," in *ISECS '08: Proceedings of the 2008 International Symposium on Electronic Commerce and Security*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 531–535.

[12] S. Frei, M. May, U. Fiedler, and B. Plattner, "Large-scale vulnerability analysis," in *LSAD '06: Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, 2006, pp. 131–138.

[13] S. Frei, B. Tellenbach, and B. Plattner, "0-day patch - exposing vendors (in)security performance," http://www.blackhat.com, 2008.

[14] "Clusif," http://www.clusif.asso.fr.

[15] A. Sharma, Z. Kalbarczyk, J. Barlow, and R. Iyer, "Analysis of security data from a large computing organization," Tech. Rep., 2010.

[16] G. Vache, "Vulnerability analysis for a quantitative security evaluation," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, vol. 0. IEEE Computer Society, 2009, pp. 526–534.

[17] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," http://www.first.org/cvss/cvss-guide.html.