Distribution Fitting for Zero Day Vulnerability Life Spans: A Quantitative assessment with reference to Weibull Distribution

C. Shabana Janani
Park Global School of Business Excellence,
Coimbatore

Abstract. Vulnerability in software is one of the key metrics in defining software reliability. As this vulnerability can be discovered by any user of a software, it becomes critical to identify who has discovered the vulnerability and how quick it has to be patched so as to minimize the risk. The risk is heavier when the vulnerability is a Zero Day Vulnerability. Many probabilistic models have been developed to study the attackers' behavior in exploiting vulnerability and the duration taken by vendors in patching vulnerability. As such these models assume the basic underlying data distribution in varied category. This research study aims to fit in one such distribution to the life spans of Zero Day Vulnerabilities prevalent in web browsers. Since the life span characteristics matched the characteristics of Weibull distribution, an attempt is made to check if the life span data distribution follows a Weibull distribution. Data were collected from the Zero Day Initiative of HP. An analysis verifying the properties of Weibull was run for the data collected. Finally it was found that the life span data of Zero Day vulnerabilities followed a Weibull Distribution.

Keywords: Software vulnerability, Zero Day vulnerability, Weibull distribution

Nobody intentionally develops software with a bug. However, bug free software is beyond the scope of any software developer. And be it a thousand dollar Hackathon or a multi-million dollar (black) market for an exploit code, everyone is basically interested in one thing, "The Vulnerability" in the software. These software vulnerabilities can cause serious economic damages (Kannan & Telang, 2005). End users lose money and valuable information. Software vendors lose their market share as software system reliability is defined as the number of failures (vulnerabilities) during a given time period (Rescorla,2005). Vulnerabilities (software flaws/bugs) are typically created by accident as a result of coding mistake, often involving mismanagement of memory (Frei et al., 2009). However this is just a comprehensive view in vulnerability creation. This flaw as we call it as software vulnerability poses little threat as long as it is discovered by a safer hand. But the probability of the discovery by someone who genuinely tries to fix it is much less. Most of the time it is exploited by hackers, whose full time job may be is to locate a security hole in software and inject malicious contents or hack the end users' system. These exploit codes sold in black market can fetch them a hefty pay than being sold to the vendor of the software. And the ultimate risk bearer is the end user. In one of the Google run Hackathon, a team of hackers from security firm Vupen were able to discover security bugs in the famous Google Chrome. Though each hacker was paid $60,000 to share every details of the exploit to fix the vulnerability discovered, Vupen's CEO openly acknowledged that Vupen never had the intention of sharing it with Google and had better markets for the vulnerability (Greenberg, 2012). In spite of all these, however, if a vulnerability happens to fall in the safer hands (hopefully without any rediscovery by hackers), vendors take their own time fixing up these vulnerabilities. These fixes, called patches, are additional pieces of code that are developed to fix the vulnerability in the software (Mell et al. 2005). Though the cost of developing the patches is less compared to the cost of developing the software, a number of vulnerabilities considerably reduce the reliability of the software. So as long as the vulnerability is not published or disclosed in public forum, the time to patch the vulnerability by a vendor is very long.

McQueen et al. (2011) in a study of the life cycle of vulnerabilities between 2009 and 2011, specified that on an average, vulnerability is kept unpublished (since they are not patched) for about 197 days. There were times when attacks came 6 months after the vulnerability was published (with patch), e.g., Nimba and Code Red (Applewhite, 2004). But unfortunately technology has pushed us into a world where we can anticipate an attack right immediately after the software is commercialized, e.g., iOS7 Control Center Vulnerability (Greenberg, 2013). So a study of vulnerability is always an ongoing and special process for various security researchers.

LITERATURE REVIEW

Vulnerability in software is "an instance of a mistake in the specification, development or configuration of software such that its execution can violate the explicit or implicit security policy" (Anderson & Moore, 2006). A defect is that which enables an attacker to bypass security measures (Schultz et al. 1990). Shepherd (2003) defines vulnerability as a flaw in the logical operation of a product that may leave the product in an undesirable state, especially allowing unauthorized access, elevating privileges or denying the services of the software. The RFC 4949 Internet Security Glossary explains vulnerability as "a flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy". Microsoft defines security vulnerability as "a flaw in a product that makes it infeasible—even when using the product properly—to prevent an attacker from usurping privileges on the user's system, regulating its operation, compromising data on it, or assuming ungranted trust"

Vulnerability Discovery

The motivation behind the discovering a vulnerability can be personal or competitive (Cecini et al. 2005). These vulnerabilities are identified either by someone, who reports it directly to the vendor or in public forums like Secunia, Bugtraq or by any malicious hacker who tries to exploit the vulnerability. These two potential scenarios are described as White Hat Discovery (WHD) and Black Hat Discovery (BHD). Vulnerability discovered with an intention of no exploits is WHD, where the discoverer is usually a freelance security professional, or a security researcher working for the vendor. The vulnerability is then notified to the vendor with an intention of preventing further exploits. The vendor in turn releases a patch along with the vulnerability details (Rescorla, 2005). In BHD, the vulnerability is discovered by a hacker or a script kiddie (a non-technical hacker) with an intention to exploit it. Usually in this scenario, the end users of the software as well as the vendors are unaware of the vulnerability, while a limited pool of in-the-know attackers keep exploiting the vulnerability. As any disclosure of information about vulnerability may lead to further exploits until it is patched, the question of how the knowledge about the vulnerability has to be disclosed to the public has always been a debated issue in Information Security (Secure Business Quarterly, 2002). As the number of vulnerabilities in software is a key metric for software reliability and finding them is competitive, a tricky ethical framework of how vulnerability has to be disclosed is required (Cecini et al., 2005).

Vulnerability Disclosure Policies

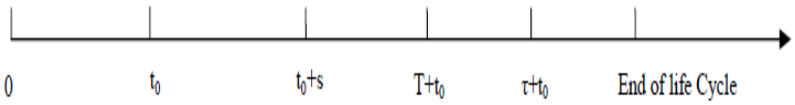
Disclosure debates range from immediate public disclosure of the vulnerability by the discoverer to full vendor disclosure/responsible disclosure until a patch is available. Proponents, who favor immediate public disclosure, claim that such disclosures force the vendors to release the patch quickly (Leyden, 2002). In addition, it cautions users to protect themselves by disabling the affected software (or related functionality) before an exploit is issued (Cencini et al., 2005). The more the end user knows about the problem the better they can defend against it (Ragan, 2010). However, the immediate public disclosure policy fails to consider the impact of the duration gap in which the vulnerability is publicly available without a patch. Studies have also shown that there is an upswing in intrusions using a given security weakness once it has been publicly disclosed (Reid, 2003). It is widely believed that the cost of developing or acquiring exploit tools and implicitly the frequency of attacks on hosts depends largely on how much information about the vulnerability is publicly known (Seltzer, 2004). On the other hand, proponents favoring full vendor disclosure, hope that it may not aggravate exploits, since limited information about the vulnerability will be available at any given time. However, the duration required to release a patch after discovery is a business decision and is directly under the control of vendors (McQueen et al., 2011). As the vendor cannot be forced to develop a fix, some reported vulnerabilities have gone unfixed or have been fixed after a long delay (Schneier, 2001). So as long as the vendor is not committed enough to develop a patch at the earliest, full vendor disclosure still may put the public at risk (Cavusoglu et al., 2007). In any case, there is a time duration in which the vulnerability can be rediscovered either by a WHD or BHD.

The concept of Full Disclosure and Non-Disclosure was a heavily debated issue until a policy of Responsible Disclosure (Partial Disclosure) was introduced. A responsible disclosure is a policy in which information about software vulnerabilities are disclosed in a limited manner, so that it puts users at the least risk (Cecini et al. 2005). Responsible Vulnerability Disclosure stresses on how the knowledge about the vulnerability has to be shared in appropriate times and through appropriate channels (Cavusoglu et al. 2007). When a vulnerability is discovered, the discoverer (WHD) informs the software vendor and if the vendor is not responsive (usually for a given grace period of time), the discoverer may then go to a community/public forums and proceed with full disclosure of the vulnerability (Shepherd, 2003). This Grace Period is the amount of time the discoverer/security researcher allots to the vendor for providing a fix/patch, after which the researcher may independently announce the vulnerability (McQueen et al. 2011). This grace period can be best understood with the help of vulnerability life cycle events.

Arbaugh et al., Vulnerability Life Cycle Model (2000)

A vendor releases a product at time '0'. At period 't0' a vulnerability is discovered and notified to vendor. This discovery is usually done by WHD, who informs to public forum like CERT/CC (Computer Emergency Readiness Team -- Coordination Center). CERT then takes the responsibility of contacting the vendor and giving a grace period of (T+t0) to fix the vulnerability (usually 45 days from the date of informing the vendor in case of CERT). And say the vendor releases the patch at (τ+t0). The period (t0+s) is the time period during which the vulnerability can be rediscovered by attackers to exploit it.

Figure 1– Life cycle of software vulnerability – Arbaugh Model



McQueen et al., Vulnerability Life Cycle Events (2011)

This model explains the cost of vulnerability as the sum of, the cost for the vendor to create a patch, the cost to mitigate the vulnerability by the end user, and the total losses from its exploitation through various phases of discovery. Similar to Arbaugh et al., this model identifies the phases as discovery, exploitation, vendor notification, disclosure and release of patch. Since the initial discovery cannot be firmly ascertained (as it may have been rediscovered by exploiters and kept secret), this model assumes three basic events in a vulnerability lifecycle for which the reported period can be dependably verified: Vendor notification period, disclosure, and release of patch. The grace period in this model is the period between vendor notification and disclosure. Consequently, a number of rediscoveries (the same vulnerability being discovered by different people within a period time) of the vulnerability is possible during this period.

This grace period is more crucial as well as critical since the software remains unpatched, exposing itself to the windows of exploitation. The vulnerability during this period is called a "Zero Day Vulnerability" and the duration during which the vulnerability remains discovered but yet to be published is the life span of a Zero Day Vulnerability. Though there is no formal definition for a Zero Day vulnerability, McQueen et al. (2009) defined it as the vulnerability deployed in software, that has been discovered at least by one person but has not yet been publicly announced or patched. A zero day attack is more vulnerable as it cannot be detected by an antivirus product through signature based screening (Bilge & Dumitras, 2012), making it further easy for BHD.

Need For the Study

Analyzing Zero Day attacks is usually complicated as the data about the attacks are generally not available unless an attack is discovered. Many studies have focused on the rate of change of the number of zero day vulnerabilities over a period (Shahzad et al., 2000; McQueen et al., 2011; Arbaugh et al., 2000) and life span of zero day vulnerabilities (Bilge & Dumitras, 2012 & MCQueen et al., 2009). High-end researches have focused on intrusion detections on various software categories (Qualys Inc, 2009; Symantec Corp, 2012). Studies also focused on comparing the vulnerabilities of different software vendors (Frei et al., 2009; Frei, 2011). Each study has contributed a model in defining its own objective. Although any modeling starts with the basic assumption that vulnerability discovery is a stochastic process, a further analysis of the distribution of time/duration of the discovery or the patch for vulnerability becomes crucial to build a model. Because, when the assumed data distribution does not hold correct, the developed probabilistic model may not be reliable. Most of the vulnerability discovery models and Zero day life span models assume the basic time to discover or patch the vulnerability in terms of wide varieties of distributions. McQueen et al. (2009) modeled the life span of zero day vulnerability with a Log-Normal Distribution. Rescorla (2005) assumes a Poisson distribution for the time to discover the vulnerability. Still many studies follow the basic assumption of Gaussian distribution for modeling life spans of vulnerabilities. In spite of these assumed distributions, other distribution models like Exponential and Weibull are also of significant interest in studying the time distribution. This paper attempts to fit a Weibull distribution model for the life span of Zero Day Vulnerabilities of various web browsers.

DATA & DISTRIBUTION FITTING OF ZERO DAY VULNERABILITY LIFE SPAN

Studies on Zero day attacks and vulnerabilities are usually carried out as a post-mortem analysis as vulnerabilities cannot be observed in lab experiments (Bilge & Dumitras, 2012). The Common Vulnerabilities & Exposure Consortium maintains an extensive database of all discovered vulnerabilities so far, with unique CVE id (Common Vulnerability Exposure) and a CVSS (Common Vulnerability Scoring System) score. This is the most commonly used database for many academic and institutional researches. While the CVE database sometimes indicates when vulnerabilities were reported to vendors, it does not give information about when vulnerability was actually discovered by a WHD or BHD (Bilge & Dumitras, 2012). Hence it is highly unlikely that an exact date of discovery would be available in any public database. A study on Zero Day vulnerability by McQueen et al. (2009) uses the data from "Zero Day Initiative" (of HP Tipping Point, a security research program acting as a broker between discoverer and vendor). Although it is a conservative estimate, many studies consider the data to be the closest possible estimate for discovery dates. ZDI gives a grace period of 6 months.

To characterize the Weibull distribution, I took into analysis the life spans of 175 Zero Day vulnerabilities, with special reference to various Web Browsers from 2011 to September 2013, from HP's Zero Day Initiative database. The data available in ZDI consists of the date when the vulnerability was reported to the vendor and the date when it was published. Assuming the date of reporting as the closest possible date of discovery, I calculated the Life Span of Zero Day vulnerabilities of web browser.

A Weibull distribution is a best fit when the data is a life data especially with reference to failure time. Also, as the scope of the Weibull covers non-zero time origin and unknown ages of units of interest, I find it best describes the characteristics of vulnerability discovery process. Though discovery of vulnerability is not a complete failure of software, its reliability is drastically reduced whenever vulnerability is discovered. And because of the inability to find the exact date of software vulnerability discovery and due to further re-discoveries, a Weibull may describe the life span distribution of vulnerabilities better than any other distributions. Also, unlike normal curve which tries to fit data over a bell shaped curve, Weibull fits or models a distribution according to the data. For this study, I consider a 2-parameter Weibull distribution, which is characterized by slope and characteristic life (mean time to fail) and is best suitable for small sample sizes. The advantage of Weibull distribution is that it adequately defines extreme values that deviate from the median of the distribution.

Weibull distribution function is given as

$F(x) = 1 - \exp\{-(x/\alpha)\beta\}$

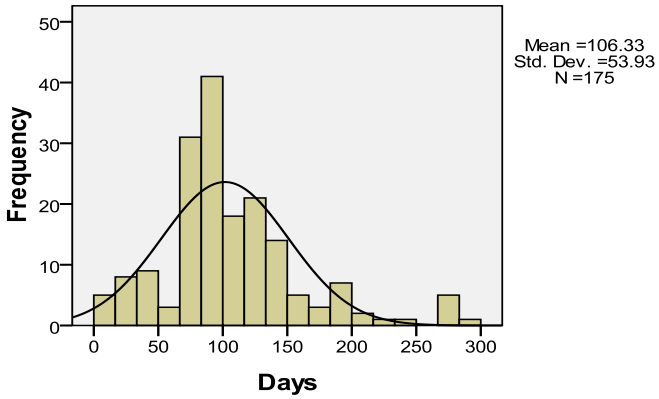
Where α is the Scale parameter that characterizes the life of the distribution and β is the Shape parameter which is equivalent to the slope parameter of a linear trend.

In order to get a brief understanding of the underlying data, I used the descriptive statistics of the life spans of Zero Day vulnerability data set (Table 1). It is clear that the data did not follow a normal distribution. The data is right skewed with skewness 1.171. This is further illustrated with a histogram in Figure 2.

Table 1 - Descriptives of the Life Spans of Zero Day Vulnerabilities

		Statistics	Std Error
Days	Mean	106.33	4.07
	Median	95	
	Std Dev	53.93	
	Skewness	1.17	.18
	Kurtosis	2.09	.36

Figure 2– Histogram of Zero Day Life Spans



A Weibull is characterized by the following properties.

1. The Scale parameter of the life span distribution should lie approximately at the 63rd percentile of the population/sample.
2. The median rank of the distribution is given by Px= (Rank(x)-0.3)/(n+0.4). If the distribution is Weibull then ln(ln(1/(1-Px))) against ln(x) should give a linear trend.
3. And the slope of the linear trend in (2) should be approximately equal to the raw Shape parameter β of Weibull.
4. The linear trend equation of (2) produces α and β closer to the raw values of scale and shape parameter obtained from (1) and (3)
5. Weibull becomes exponential when shape parameter β=1. Therefore a new transformation $Y=X^{\beta}$, will be an exponential distribution with mean = αβ.
6. The Weibull probability versus the life span data should be linear.
7. And the slope of the linear trend in (6) should be unity.

Applying the Weibull properties discussed above to the life span of Zero Day Web Browser vulnerabilities, we can interpret the following:

I. After arranging the data in ascending order and ranking them, the 63rd percentile life spans' rank is calculated as 0.63 *n= 0.63 * 175 = 110.25
The life span value corresponding to the 63rd percentile at 110.25th rank is approximately 108 days. This gives us the rough estimate of the Weibull's characteristic life or scale parameter α.

II. Calculating the median rank Px for the data points, a regression is run with the log transformed values for analyzing the linear trend. The regression fit (Figure 3) shows an almost linear trend with a high coefficient of determination 95% (Table 2). The linear trend is also significant with slope =2.13 and intercept = -10.22 (Table 3).

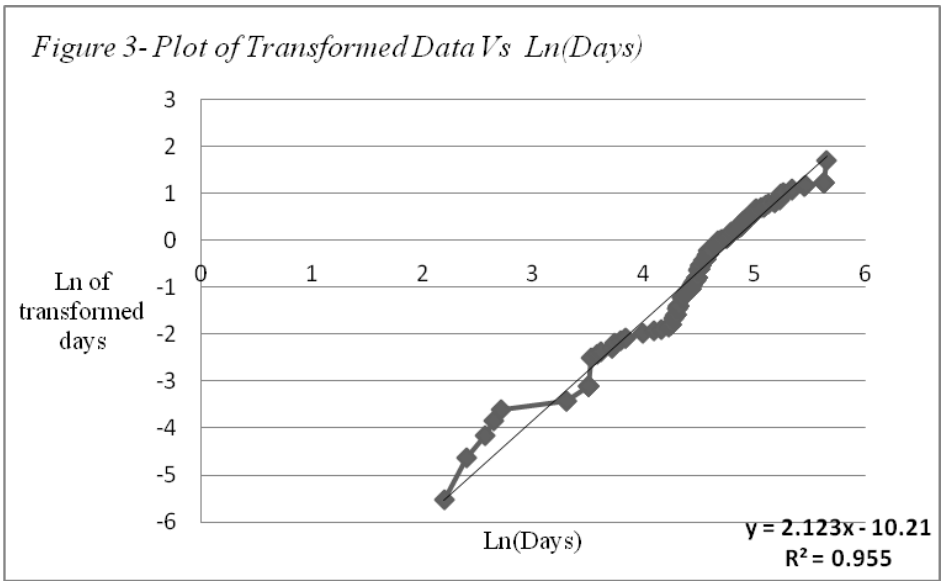


Table 2 - Regression Statistics of the log transformed values

Multiple R	0.98
R Square	0.95
Adjusted R Square	0.95
Standard Error	0.27

Table 3- Regression Output

Predictors	Coefficients	Standard Error	t Stat	P- Value
Intercept	-10.22	0.17	60.96	0.00
2.20	2.13	0.04	57.94	0.00

III. The slope of log transformed linear trend, 2.12 will be the shape parameter β of the Weibull distribution of the life span data.

IV. The scale parameter α is given as $\alpha = \exp\{-(\text{Intercept}/\text{Slope})\}$
$= \exp\{-(-10.22/2.12)\} = 124.05$

The raw value of the slope is calculated using the general slope formula and the value is 2.12. As suggested in (1) and (3), we find that the rough estimates of (α,β) i.e., (108,2.12) is closer to the formal estimates (124, 2.13) calculated using the trend equation. The variation in the scale parameter is due to the fact that means are affected by the extreme values in the data set. A few Zero day vulnerabilities had lifetime that extended even to 285 days. However, removing the extremities made 'α' move closer to the raw characteristic life value at 114 days. The extremities were removed under the assumption that the software might have been in its obsolete phase, during which the vendors' preference is usually towards developing a new product version of the earlier software.

V. Also by assessing the exponentiality of Y, as Y=X2.1, for all the ZD vulnerability life spans, we find that the mean of the exponential distribution (calculated using arithmetic mean) 26342.8 is nearer to the exponential mean $\alpha\beta = (124)2.12 = 27419.15$, estimated using the regression line.

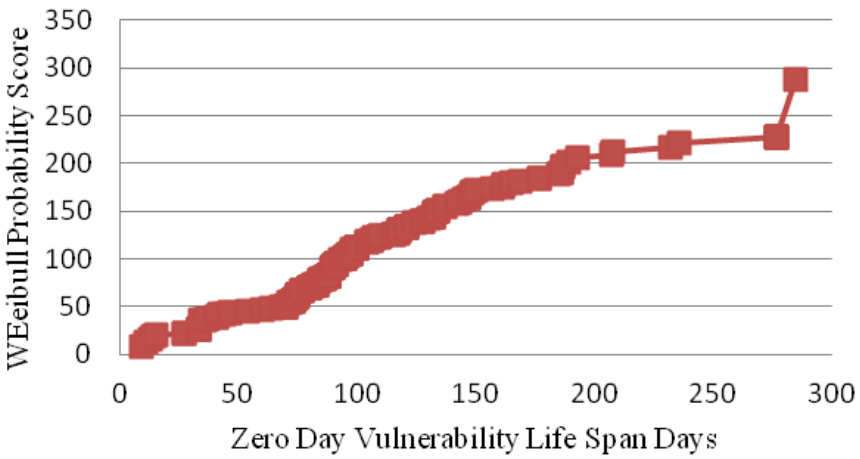
VI. Comparison of estimated Weibull probability with actual life span probability: The Weibull probability xi is then calculated from the Weibull distribution function

$F(x) = Px = 1 - \exp\{-(xi/\alpha)\beta\}$

Therefore $xi = \{-\alpha\beta \ln(1-Px)\} 1/\beta$

Regression is run for the calculated Weibull probability against the corresponding life span data. The regression line (Figure 4) shows an upward linear trend thus satisfying (6).

Figure 4- Weibull Probability Plot



VII. Finally running a linear trend line for Weibull probability versus actual life span data gives us the following regression line.

Weibull probability = 1.093 + (.990) Weibull Sample (Table 5)

It is clear that the slope of the regression line covers unity as required by (7) and a high index of fit 93.7% (Table 4) shows that Zero Day Vulnerability Life span follows a Weibull distribution.

Table 4- Regression Statistics

Multiple R	0.97
R Square	0.94
Adjusted R Square	0.94
Standard Error	14.06

Table 5 – Regession Output for Weibull Probability

	Coefficients	*Standard Error*	*t Stat*	*P-value*
Intercept	1.09	2.38	0.45	0.64
9	0.99	0.02	49.95	0.00

CONCLUSION & FUTURE WORK

Thus, as the life span data confirms to the properties of Weibull distribution, it can be reasonably inferred that the life span of Zero Day vulnerabilities of web browsers closely follows a Weibull distribution with shape parameter 2.12 and characteristic life 124 days. However, while working with the log transformed data of life span, a QQ plot of the same showed that the distribution fit Poisson distribution much more adequately. Hence, further research can be done to check if the data might follow a Poisson distribution. Also, as it's concluded that the distribution is Weibull, it may be further developed to fit a distribution model appropriately verifying the shape and scale parameter. One of the limitations of the study is that, the data for Zero Day vulnerability was available only from 2011 from Zero Day Initiative. So, the sample may not adequately represent the population and the distribution may vary for a different time period or a different category of software vulnerability.

Acknowledgement

This research would not have been possible without the published advisories collected from www.zerodayinitiative.com. I would thank HP's Zero Day Initiative and DVLabs for the data available for public access about vulnerability disclosures.

REFERENCES

Alhazmi,H., Whan Woo, S. & Malaiya, Y.K. (2008). Security Vulnerability Categories in major Software Systems. http://www.cs.colostate.edu/~malaiya/pub/CNIS-547-097.pdf

Anderson,R. & Moore, T. (2006). The Economics of Information Security. In Science, Vol 314. No. 5799

Applewhite, A. (2004). Whose Bug is it Anyway? The Battle over handling software Flaws. IEEE Software, 21(2).

Arbaugh, W.A., Fithen, W.L. & McHugh,J. (2000). Windows of Vulnerability: A Case Study Analysis. IEEE Computer. 33. (52-59)

Arora,A., Nandkumar,A. & Telang, R. (2006). Does Information Security attack frequency increase with Vulnerability Disclosure? An Empirical Analysis. Springer Science.

Bilge, L., & Dumitras, T. (2012). Before We Knew it- An Empirical Study of Zero Day Attacks in the Real World. 2012 ACM Conference on Computer & Communication Security

Cavusoglu, H., Cavusoglu, H. & Raghunathan, S. (2007). Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge. IEEE Transactions on Software Engineering.

Cencini,A., Yu,K. & Chan,T. (2005). Software Vulnerabilities: Full-, Responsible- and Non-Disclosure. http://courses.cs.washington.edu/courses/csep590/05au/whitepaper_turnin/software_vulnerabil ities_by_cencini_yu_chan.pdf

Culp, S. (2000). Definition of a Security Vulnerability. Microsoft TechNet.

Frei, S. (2011). End-Point Security Failures, Insight Gained from Secunia PSI Scans. Predic Workshop

Frei, S., Schatzmann, D., Plattner, B., & Trammel, B. (2009). Modelling the security ecosystem- The Dynamics of (in) security. http://www.techzoom.net/papers/weis_security_ecosystem_2009.pdf/

Greenberg, Andy. (2012). Shopping for Zero-Days: A price list for the hackers' secret software exploits. www.forbes.com.

Greenberg, Andy. (2013). Another iOS7 bug lets anyone make calls from locke iphonesand this one has no quick fix. www.forbes.com.

HP Enterprise Security. (2012). HP Cyber Risk Report, 2012.

Internet Engineering TaskForce. RFC 4949 Internet Security Glossary.

Kannan, Karthik & Telang, Rahul (2005, May). Market for software vulnerabilties? Think Again. Management Science, ABI/INFORM complete (pp 726-740)

Leyden, J. (2002). Show us the Bugs - Users want Full Disclosure. The Register.

McQueen, M. A., McQueen, T.A., Boyer, W.F. & Chaffin, M.R. (2009). Empirical Estimates and Observations of 0 Day Vulnerabilities. Proceedings of 42nd Hawaii International Conference on System Sciences.

McQueen, M., Wright, J.L., & Wellman,L. (2011). Are Vulnerability Dusclosure Deadlines Justified? Third International Workshop on Security Measurements and Metrics, 2011, The IEEE Computer Society

Mell, P., Bergeron, T., & Henning, D. (2005). Creating a patch and Vulnerability Management Program. NIST Special publication, SP 800-40V2.

Qualys Inc. (2009). The Laws of Vulnerability 2.0.

Rangan, S. (2010). The New Era of Vulnerability Disclosure- A Brief Chat with HD Moore. The Tech Herald.

Rescorla, E. (2005). Is finding security holes a good idea? Economics of Information Security, The IEEE Computer Society.

Romeu, J.L. Empirical Assessment of Weibull Distribution. Selected Topics in Assurance Related Technologies, 10(3).

Schneir, B. (2001). Bug Secrecy Vs Full Disclosure. ZDNet Tech Update.

Schultz, E.E., Brown, D.S. & Longstaff, L.T.A. (1990). Responding to Computer Security Incidents. Lawrence Livemore National Laboratory Technical Report, NTIS Issue No. 9102.
Secure Business Quarterly. (2002). Special Issue- Vulnerability Disclosure, 2.
Seltzer, L. (2004). How should researchers handle exploit code? E- Weel, April, 2004.
Shahzad, M., Shafiq, M.Z. & Liu, A.X. (2012). A Large Scale Exploratory Analysis of Software Vulnerability Life Cycles. Proceedings of 2012 International Conference on Software Engineering
Shepherd, S. (2003). Vulnerability Disclosure: How do we define Responsible Disclosure?. Part of SANS Institute InfoSec Reading Room.
Skibell, R. (2003). The Phenomenon of Insecure Software in a Security Focused World. Journal of Technology, Law & Policy.
Symantec Corporation. (2012). Symantec Internet Security Threat Report.
Zheng, C., Zhang, Y., Sun, Y. & Liu, Q. (2011). IVDA: International Vulnerability Database Alliance. 2011 Second Worldwide Cyber Security Summit.