available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Measuring, analyzing and predicting security vulnerabilities in software systems

O.H. Alhazmi, Y.K. Malaiya, I. Ray[*]

Department of Computer Science, Colorado State University, Fort Collins, CO 80523, United States

ARTICLE INFO

Article history:
Received 7 November 2005
Accepted 4 October 2006

Keywords:
Vulnerabilities
Security holes
Risk evaluation
Quantitative security modeling
Defect density

ABSTRACT

In this work we examine the feasibility of quantitatively characterizing some aspects of security. In particular, we investigate if it is possible to predict the number of vulnerabilities that can potentially be present in a software system but may not have been found yet. We use several major operating systems as representatives of complex software systems. The data on vulnerabilities discovered in these systems are analyzed. We examine the results to determine if the density of vulnerabilities in a program is a useful measure. We also address the question about what fraction of software defects are security related, i.e., are vulnerabilities. We examine the dynamics of vulnerability discovery hypothesizing that it may lead us to an estimate of the magnitude of the undiscovered vulnerabilities still present in the system. We consider the vulnerability discovery rate to see if models can be developed to project future trends. Finally, we use the data for both commercial and open-source systems to determine whether the key observations are generally applicable. Our results indicate that the values of vulnerability densities fall within a range of values, just like the commonly used measure of defect density for general defects. Our examination also reveals that it is possible to model the vulnerability discovery using a logistic model that can sometimes be approximated by a linear model.

1. Introduction

The security of software systems has been under considerable scrutiny in recent years. However, much of the work on security has been qualitative, focused on detection and prevention of vulnerabilities in these systems. There is a need to develop a perspective on the problem so that methods can be developed to allow security related risks to be evaluated quantitatively.

Quantitative methods can permit resource allocation for achieving a desired security level, as is done for software or system reliability. They can be used to evaluate metrics that can guide the allocation of resources for security testing, development of security patches and scheduling their releases. They can also be used by end-users to assess risks and estimate the needed redundancy in resources and procedures for handling potential security breaches. Unfortunately, only limited attention has been paid so far to the quantitative aspects of security. To develop quantitative methods for characterizing and managing security, we need to identify metrics that can be evaluated in practice and have a clearly defined interpretation. In this work, we examine the data on vulnerabilities identified in several popular operating systems to determine if concepts such as “vulnerability density” and “vulnerability discovery rate” can be characterized as useful metrics for software security. *Vulnerability* has been defined

[*] *Corresponding author*. Tel.: +1 970 4917097; fax: +1 970 4912466.
E-mail addresses: omar@cs.colostate.edu (O.H. Alhazmi), malaiya@cs.colostate.edu (Y.K. Malaiya), indrajit@cs.colostate.edu (I. Ray).

0167-4048/$ – see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2006.10.002

as ''a defect, which enables an attacker to bypass security measures'' (Schultz et al., 1990). Pfleeger (1997) defines vulnerability as ''a weakness in the security system that might be exploited to cause loss or harm''. Quantitative methods for general defects are now widely used to evaluate and manage overall software reliability. Since software system vulnerabilities—the faults associated with maintaining security requirements—can be considered a special case of software defects, similar measures for estimating security vulnerabilities appear useful.

It is not possible to guarantee absence of defects in non-trivial sized programs like operating systems. While extensive testing can isolate a large fraction of the defects, it is impossible to eliminate them. This is because the effort needed to discover residual defects increases exponentially (Lyu, 1995). Nonetheless, examination of defect densities (that is the number of defects identified in the unit size of the software code) is useful. It can lead to the identification of fault-prone modules that need special attention. Researchers have evaluated the ranges of defect densities typically encountered during different phases of the software life cycle using data from available sources (Musa et al., 1987). This has led to industry wide standards for software defect densities. The information can be used for comparison with the defect density measured in a project at a specific phase. The result can identify if there is a need for further testing or process improvement. Similar methods for managing the security aspects of systems by considering their vulnerabilities, can potentially reduce the risk of adopting new software systems.

Researchers in software reliability engineering have analyzed software defect finding rates. Software reliability growth models relate the number of defects found to the testing time (Lyu, 1995; Musa et al., 1987; Malaiya and Denton, 1997). Methods have been developed to project the mean time to failure (MTTF) or the failure rate that will occur after a specific period of testing. Software defect density (Malaiya and Denton, 2000; Mohagheghi et al., 2004; Mockus et al., 2002) has been a widely used metric to measure the quality of a program and is often used as a release criterion for a software project. Not much quantitative work, however, has been done to characterize security vulnerabilities along the same lines.

Security can be characterized by several possible metrics. Depending on how we view the systems, the applicable metrics can vary. Littlewood et al. (1993) and Brocklehurst et al. (1994) discuss some possible metrics to measure security based on dependability and reliability perspectives. They propose using effort rather than time to characterize the accumulation of vulnerabilities; however, they do not specify how to assess effort. Alves-Foss and Barbosa (1995) have proposed a metric termed Software Vulnerability Index (SVI). SVI is calculated using some predefined rules and can take values between 0 and 1.

An analysis of exploits for some specific vulnerabilities has been considered by Arbaugh and Fithen (2000) and Browne et al. (2001). The security intrusion process has also been examined by Jonsson and Olovsson (1997) and Madan et al. (2002). Other researchers have focused on modeling and designing tools that make some degree of security assessment possible (Schultz et al., 1990). Only a few studies have examined the number of vulnerabilities and their discovery rates. Rescorla (2005) has examined vulnerability discovery rates to determine the impact of vulnerability disclosures. Anderson (2002) has proposed a model for a vulnerability finding rate using a thermodynamics analogy. Alhazmi and co-workers (January 2005, August 2005, November 2005) have presented two models for the process of vulnerabilities discovery using data for Windows 98 and NT 4.0. In this work we focus on the density of defects in software that constitute vulnerabilities, using data from five versions of Windows and two versions of Red Hat Linux.

Vulnerability density is analogous to defect density. Vulnerability density may enable us to compare the maturity of the software and understand risks associated with its residual undiscovered vulnerabilities. We can presume that for systems that have been in deployment for a sufficient time, the vulnerabilities that have been discovered represent a major fraction of all vulnerabilities initially present. For relatively new systems, we would like to estimate the number of remaining vulnerabilities. This requires development and validation of appropriate vulnerability discovery models. Ounce Labs (2004) uses a metric termed V-density which appears to be somewhat related. However, their definition and evaluation approach is proprietary and is thus not very useful to the general community.

Unlike the data for general defects in a commercial operating system, which are usually hard to obtain, the actual data about known vulnerabilities found in major operating systems are available for analysis. We analyze this data to address a major question: do we observe any similarity in behavior for vulnerability discovery rates for various systems so that we can develop suitable models? We examine several software systems, which we grouped into related families after plotting the cumulative number of their vulnerabilities. One of our objectives is to identify possible reasons for the changes in the vulnerability detection trends.

One major difference makes interpreting the vulnerability discovery rate more difficult than the discovery rate of general defects in programs during testing. Throughout its lifetime after its release, an application program encounters changes in its usage environment. When a new version of a software system is released, its installed base starts to grow. As the newer version of the software grows, the number of installations of the older version starts to decline. The extent of vulnerability finding effort by both ''white hat'' and ''black hat'' individuals is influenced by the number of installations; this is because the larger the installed base the more is the reward for the effort. Thus, the rates at which the vulnerabilities are discovered are influenced by this variation in usage.

The rest of the paper is organized as follows. In Section 2 we introduce vulnerability density as a metric and examine its value for several major operating systems. We analyze, in Section 3, the vulnerability discovery rates. We consider two models for the vulnerability discovery process. We then examine in Section 4 the applicability of these models to several version of Windows as well as to Linux, an open-source operating system. Conclusions are presented in Section 5 and future research that is needed is identified.