Poster: Exploitation of vulnerabilities - if, when and how often

Kartik Nayak*, Daniel Marino†, Petros Efstathopoulos†, Tudor Dumitras‡,
kartik@cs.umd.edu, Daniel_Marino@symantec.com, Petros_Efstathopoulos@symantec.com, tdumitra@umiacs.umd.edu
*Student, University Of Maryland, College Park
†Symantec Research Labs
‡Assistant Professor, University Of Maryland, College Park

I. Introduction

In order to improve the security of our software systems, we need to be able to measure how they are impacted by the various defensive techniques introduced to the system. Measuring security, however, is challenging. Many security metrics have been proposed, including the total count of vulnerabilities in source code, the severity of these vulnerabilities, the size of the attack surface and the time window between the vulnerability disclosure and the release of a patch. System administrators and security analysts often rely on these metrics to assess risk and to prioritize some patches over others, while developers use them as guidelines for improving software security. Practical experience, however, suggests that the existing security metrics exhibit a low level of correlation with vulnerabilities and attacks, and they do not provide an adequate assessment of security [1], [2].

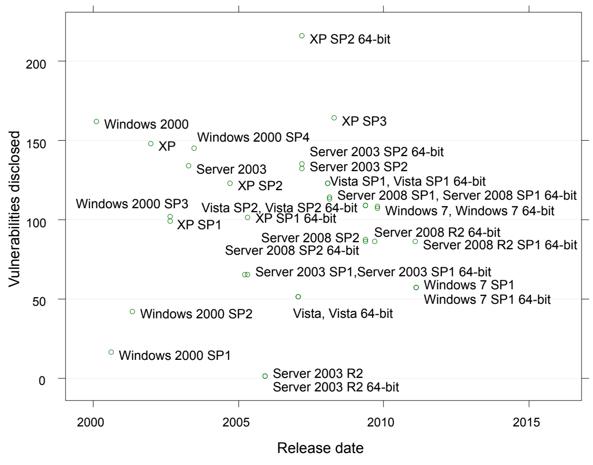
The *total number of vulnerabilities* discovered in source code is commonly used as a measure of the system's security [1], [2]. However, this metric does not account for the fact that cyber attackers never make use of some of the discovered vulnerabilities, which may be hard to successfully exploit in the presence of security technologies such as data execution prevention (DEP) and address space layout randomization (ASLR). For example, CVE-2007-1748 was exploited in the wild, but there is no similar evidence for CVE-2007-1749.

Another popular metric is based on the observation that attacks can succeed only if the vulnerable software accepts input from potential attackers. For this reason, system administrators have long advocated turning off unnecessary system services to avoid exposure to exploits of unpatched or unknown vulnerabilities. For example, network-based attacks exploiting CVE-2007-1748 are unsuccessful—even if the vulnerability was not yet patched—if the DNS server is not running. This idea is formalized in the concept of *attack surface* [3], [4], which proposes quantifying the amount and severity of potential attack vectors that a system exposes using a formula that takes into account the open sockets and RPC endpoints, the running services and the privilege at which they are running, the active Web handlers, the accounts enabled, etc. Reducing the attack surface, however, does not always improve security; for example, including security mechanisms in the OS may increase the attack surface, but render the system more secure. Furthermore, the attack surface of software products changes after they are deployed in the field, as users install new applications and modify system configuration. To the best of our knowledge, the size and variability of attack surfaces has not been evaluated empirically in the field. It is, therefore, difficult to determine the effectiveness of this metric in capturing real-world conditions.

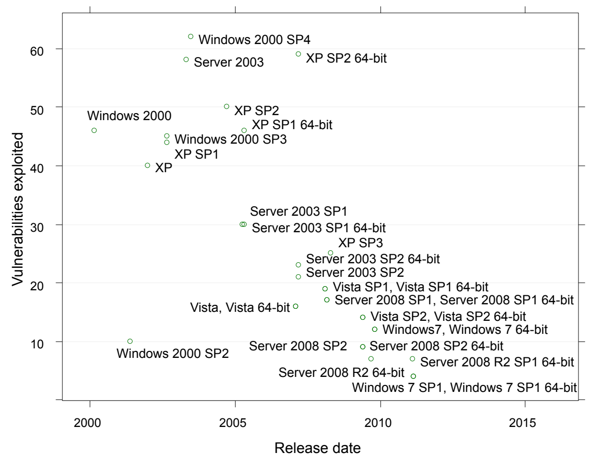
These examples illustrate that our ability to assess the security of systems that are deployed and actively utilized is currently limited by the metrics being used. In particular, the developers and the users may employ different security metrics. For example, one way of estimating the vulnerability density and the attack surface is to use existing tools that measure these properties by directly analyzing the code and the configuration of the system in question [5], [6]. However, these measurements are conducted in lab conditions, and do not reflect the real-world security of systems that are deployed and actively used in the field.

For these reasons, users are ultimately interested in metrics that help them assess the effectiveness of these techniques in the field. Figure 1 illustrates this problem. The total number of vulnerabilities discovered in the code-base of Windows OSes does not follow a clear trend with respect to the OS release date. Basing the evaluation of system security on this metric would lead to the conclusion that security has not improved considerably over the past decade. However, the number of vulnerability exploits exhibits a decreasing trend over time,which suggests the existence of deployment-specific factors, yet to be characterized systematically, that influence the security of systems in active use.

Our *goal* in this work is to propose new metrics that better reflect security in the real world and to employ these metrics for evaluating the security of popular software. Rather than measuring security in lab conditions, we derive metrics from field-gathered data and we study the trends for vulnerabilities and attack surfaces exercised in attacks observed in the real world. These metrics try to capture the notion of *whether* disclosed vulnerabilities are exploited and *how often* they are exploited. While the vulnerability count and the attack surface are metrics that capture the *opportunities* available to attackers, we instead focus on attempted, though not necessarily successful, attacks in the field. This new understanding will allow developers, system administrators and policy makers to improve security by focusing on the attacks and vulnerabilities that matter most in practice.



(a) All vulnerabilities disclosed publicly.



(b) Vulnerabilities exploited in the wild.

Fig. 1. Number of vulnerabilities disclosed and exploited for Microsoft Windows over 11 years of releases, estimated using the National Vulnerability Database and WINE. No clear trend exists for total number of vulnerabilities disclosed but number of vulnerabilities actually exploited in the field decreases with newer OSes.

II. Proposed metrics

We propose several metrics, derived from field-gathered data, that capture the state of system security as experienced by the users.

The following metrics capture the notion of whether disclosed vulnerabilities get exploited.

1) *Count of vulnerabilities exploited in the wild.*
2) *Exploitation ratio.* The exploitation ratio is the proportion of disclosed vulnerabilities for product p that have been exploited up until time t. It captures the likelihood that a vulnerability will be exploited.

We also propose the following metrics that capture how often vulnerabilities are exploited on hosts in the wild.

1) *Attack Volume.* The attack volume is a measure that captures how frequently a product p is attacked.
2) *Exercised Attack Surface.* The exercised attack surface captures the portion of the theoretical attack surface of a host that is targeted in a particular month.

III. Experimental methods

We use the *National Vulnerability Database* (NVD) [7], *Symantec attack signatures* [8], [9] and Symantec's *Worldwide Intelligence Network Environment* (WINE) [10] for our study. The NVD is a widely accepted database for research related to software vulnerabilities. Symantec security products include an extensive database of attack signatures obtained from Symantec's intrusion-protection systems as well as descriptions of anti-virus signatures that are used to scan files for known threats. WINE contains records of vulnerabilities exploited in the field (IPS telemetry dataset) as well as information of binaries (binary reputation dataset) on end-user hosts running Symantec products. We combine these three databases to identify attacks on end hosts running the following products - *Windows Operating systems, Microsoft Office, Internet Explorer and Adobe Reader*.

Using NVD and Symantec signatures, we identify the vulnerabilities in the products that are exploited as well as the signatures used to exploit the vulnerabilities. This information is used to compute the exploitation metrics. Using the binary reputation dataset, we find the products installed on the hosts. The IPS telemetry dataset identifies an attack on a host. Knowing the products installed on the host as well as the signatures corresponding to vulnerabilities, relevant attacks on the hosts can be detected. This information can be used to compute the attack volume and attack surface metrics.

IV. Conclusion

Our ability to improve system security rests on our understanding of how to measure and assess security under real-world conditions. We believe that our metrics would better reflect security in the real world and help evaluate the security of softwares. This can help the design of future security technologies.

References

[1] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne, "Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities," *IEEE Trans. Software Eng.*, vol. 37, no. 6, pp. 772–787, 2011.

[2] T. Zimmermann, N. Nagappan, and L. A. Williams, "Searching for a needle in a haystack: Predicting security vulnerabilities for windows vista," in *ICST*, 2010, pp. 421–428.

[3] M. Howard, J. Pincus, and J. M. Wing, "Measuring relative attack surfaces," in *Workshop on Advanced Developments in Software and Systems Security*, Taipei, Taiwan, Dec 2003.

[4] P. K. Manadhata and J. M. Wing, "An attack surface metric," *IEEE Trans. Software Eng.*, vol. 37, no. 3, pp. 371–386, 2011.

[5] Microsoft Corp., "Microsoft Attack Surface Analyzer - Beta," http://bit.ly/A04NNO.

[6] Coverity, "Coverity scan: 2011 open source integrity report," 2011.

[7] National Vulnerability Database, http://nvd.nist.gov/.

[8] "Symantec Attack Signatures," http://bit.ly/1hCw1TL.

[9] Symantec Corporation, "A-Z listing of threats and risks," http://bit.ly/11G7JE5.

[10] T. Dumitras and D. Shou, "Toward a standard benchmark for computer security research: The worldwide intelligence network environment (wine)," in *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, ser. BADGERS '11. New York, NY, USA: ACM, 2011, pp. 89–96. [Online]. Available: http://doi.acm.org/10.1145/1978672.1978683