A Framework for Measuring the Vulnerability of Hosts[1]

Karen Scarfone
National Institute of Standards and Technology (NIST)
karen.scarfone@nist.gov

Tim Grance
National Institute of Standards and Technology (NIST)
grance@nist.gov

Abstract

This paper proposes a framework for measuring the vulnerability of individual hosts based on current and historical operational data for vulnerabilities and attacks. Previous approaches have not been scalable because they relied on complex manually constructed models, and most approaches have examined software flaws only, not other vulnerabilities such as software misconfiguration and software feature misuse. The framework uses a highly automatable metrics-based approach, producing rapid and consistent measurements for quantitative risk assessment and for attack and vulnerability modeling. In this paper, we propose the framework and its components and describe the work needed to implement them.

1. Introduction

Vulnerabilities are often assumed to be software flaws, but also include software misconfigurations and software feature misuse. Vulnerabilities can be mitigated through many means, but not all vulnerabilities can be completely offset. For example, no patch might be available, or reconfiguration might disable needed functionality. Attacks can also be mitigated in many ways, such as using firewalls or antivirus software. However, attack mitigation cannot be fully effective, so some attacks will still succeed.

Decisions involving security policy and host security configuration are usually based on conventional wisdom for best practices, not quantitative assessments of host security. Best practices are often dated and do not take into account current threats. Without quantitative security measures, an organization cannot easily determine a host's security posture. Hosts could be better secured if quantitative measures were used to answer security questions. For example, which types of attacks are most likely to seriously impact a host? Which types of vulnerabilities on a host are most likely to be exploited? How can a host be better secured to reduce the impact of attacks?

This paper proposes a metrics-based framework for quantifying host security that analyzes the characteristics of vulnerabilities in the context of observed attacks and security controls to generate host security metrics. Framework data would be collected and analyzed primarily through automated means, providing a flexible, scalable method for measuring technical host security.

The framework can be used for many purposes. A host's technical security posture can be compared to a baseline, such as the organization's security policy or a vendor-recommended configuration, and the postures of hosts that use the same baseline can be compared. Another use for the framework is planning security policies and controls, such as quantifying the strength of a policy, determining the effect of a policy change, comparing security configurations, and providing data for attack models. The framework can also be used for risk assessment, such as determining how much risk remains from unmitigated vulnerabilities, identifying weaknesses in security controls, selecting security controls within resource limitations, and estimating the mean time to exploitation for a host.

The rest of this paper explains the framework. Section 2 discusses related work and explains the need for the framework. Section 3 provides an overview of the framework, and Section 4 discusses the framework components. Section 5 presents conclusions.

2. Related work

Most previous work has emphasized attack modeling; attack graphs and trees, which show how an attacker could achieve certain goals, have been studied

[1] Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States.

extensively [5]. The approach in [4] involves determining the effort different types of attackers would expend to compromise a system. Some work has also been done on vulnerability modeling; for example, [1] proposed a method for measuring a system's vulnerabilities based on the vulnerabilities of its network services. Another approach [2] is an automated risk analysis method based on analyzing vulnerabilities and information on the system's components, policies, configuration, and service interactions and dependencies. Unfortunately, these approaches are not well-suited to our goals. Approaches that rely on complex threat models are generally not scalable [5] and are not particularly helpful in determining what specific actions should be taken to improve security [3, 4]. Also, most approaches do not address a wide range of vulnerabilities [1, 2].

Another approach to host security measurement is to apply dependability modeling principles. For example, [9] proposes monitoring host-related activity such as possible signs of attacks and compromises, analyzing those actions, and applying the results to stochastic models to estimate the current security state of the system and the likelihood that it will be compromised within a certain period. This approach does not analyze vulnerabilities and attacks—it associates observed events with likely security states. Although our framework's goals differ from those of [9], both frameworks analyze operational attack data to make determinations about host security.

3. Framework overview

The security controls for a host and its environment are intended to mitigate attacks and vulnerabilities. Examples of attack mitigation are antimalware utilities and firewalls; examples of vulnerability mitigation are patching and host hardening. If these mitigations are not sufficient for an attack, it will succeed. For this framework, we consider only technical vulnerabilities, which we define as software flaws, misconfigurations, and feature misuse.

We propose that quantitative measures of technical vulnerabilities, attacks, and security controls be collected and analyzed together. The effectiveness of attack mitigation can be determined by analyzing successful and failed attacks, and for vulnerability mitigation by analyzing the unmitigated vulnerabilities and how successful compromises used them. Most measures can be collected through automated means, allowing metrics to be regenerated to reflect the current security posture.

Our framework asserts that vulnerabilities should be weighted according to several characteristics. For example, the level of access gained varies based on the nature of the vulnerability and the environment in which the vulnerable component (e.g., service, application) runs, such as user-level or administrator-level rights. For some hosts, certain impact types are more important—confidentiality might be valued over availability for a host storing personal information. Also, some vulnerabilities are easier to exploit than others (e.g., require less skill, are accessible remotely). The ease of exploitation and potential impact of exploitation are two major factors in determining how likely it is that a vulnerability will be exploited; other relevant factors include the popularity of the vulnerable component and the perceived value of the host.

The framework takes into account how often different host components and vulnerabilities are exploited. Quantitative data on detected attacks against an organization's hosts and the vulnerabilities the attacks targeted is available from security controls, such as antivirus software and intrusion detection systems, as well as from incident reports. This data can be used to determine which host components and types of vulnerabilities are at greatest risk and which attack vectors are most likely to be used. Such analysis allows organizations to assess the relative strength of host security controls so that they can prioritize resources accordingly for improving security. There are also interdependencies involving vulnerabilities and other host characteristics that the framework will take into account. For example, if a service is disabled, then vulnerabilities in that service are not exploitable.

The intent of the framework is to analyze a host's technical security in just enough detail to identify important weaknesses. Vulnerability and attack measurement is not exact because of the ever-changing nature of vulnerabilities and attacks. We strive to define metrics that can be gathered quickly and consistently and that are reasonably accurate. The framework avoids the level of detail of existing vulnerability and attack modeling paradigms, such as identifying all possible attack paths for a network.

Figure 1 shows major elements of the framework as could be used for comparing a host's current security state to a security baseline. First, a host profile is created by documenting the host's security baseline, component definitions, and interdependencies. Next, data is collected to determine weightings, which are applied to the host profile, along with data for the host's current security state. This generates host security measures that indicate the host's security posture relative to the baseline.

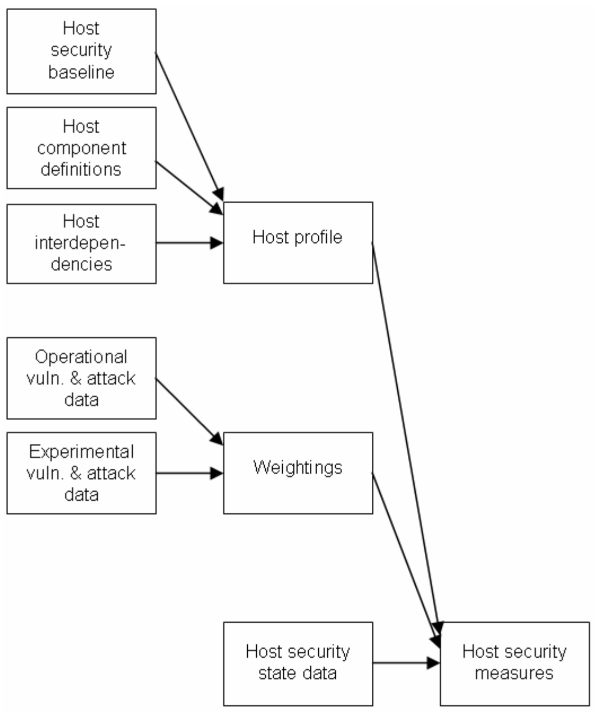

Figure 1. Framework used for baseline comparison

4. Framework elements

Before the proposed framework can be implemented and tested, all the framework's elements must be established. Below we describe each element, its current state, and the work needed to complete it. The standards referenced are from the Security Content Automation Protocol (SCAP) [8], a suite of open standards for expressing host security information.

4.1. Vulnerability characterization

We propose using the Common Vulnerability Scoring System (CVSS) for documenting vulnerability characteristics. CVSS includes several measures of the difficulty and potential impact of exploitation. Additional research will determine which of these measures are needed for the framework and which other measures not in CVSS may also be needed, such as the number of days since a vulnerability was announced or a patch or exploit code was publicly available. CVSS was originally defined for use with software flaws only [7], but we are currently finalizing a definition for misconfigurations and will also propose definitions for other types of vulnerabilities.

The framework will need sources of the three types of CVSS data: static (also known as base), temporal, and environmental. CVSS data for static vulnerability characteristics is already available for flaws identified in the Common Vulnerabilities and Exposures (CVE) dictionary. CVSS data for static characteristics would need to be generated for the other types of vulnerabilities, and this data could be shared with all organizations. Other CVSS data is called temporal because it changes over time, such as the availability of exploit code for a vulnerability, so an up-to-date source of temporal data would be needed. (Some security vendors are currently maintaining temporal data for some software flaws.) Other CVSS data is called environmental because it is organization-specific, so an organization would have to calculate it periodically.

The framework also needs documentation of the security settings for each piece of software of interest and the settings' interdependencies. For most software, such information is not currently available in a standard format, so this would need to be done.

4.2. Attack characterization

Ideally, an organization could identify all attacks and determine the vulnerabilities each attack targeted and the success or failure of each attack. However, this is not feasible because of the inaccuracy inherent in attack detection. False negatives in attack mitigation controls are unavoidable if the false positive rate is to be manageable. The sheer number of attempted attacks makes verification and analysis of each infeasible. Also, some types of threats, such as insiders inappropriately releasing data, may be hard to identify. However, as long as attack identification and characterization is done consistently, the results should be sufficiently accurate for the framework.

Research will be needed to determine which attack characteristics should be included. Likely sources of data are actual vulnerability, attack, and incident data, as well as data from honeypots, penetration testing, and other forms of security testing and experimentation. For example, network activity could be duplicated onto a test network and then evaluated using a variety of security controls to quantify the effectiveness of each control. Testing would also be helpful for determining how long it takes attackers to exploit certain types or combinations of vulnerabilities.

4.3. Scoring

Much research needs to be done on how scores should be calculated. CVSS can be used to calculate scores for individual vulnerabilities, but these scores do not take into account the relative likelihood of vulnerability exploitation or the interdependencies between vulnerabilities. Research into scoring must explore how vulnerabilities should be grouped (such as by host component [6]), what scoring scales should be used, and how vulnerabilities should be weighted. Research may show that vulnerability scores provided by CVSS are not as important for the framework as knowing the major characteristics of each vulnerability.

4.4. Automation mechanisms

The framework needs a mechanism for automatically collecting host configuration and security data, as well as performing scoring. We propose using the Extensible Configuration Checklist Description Format (XCCDF) for this. XCCDF can define the security baseline to be measured against and the relationships between vulnerabilities, as well as calculating various scores for a host. For the framework, an XCCDF document will be needed for each OS and application. In terms of content development, there are a few publicly available XCCDF documents for widely used operating systems and more XCCDF documents are in development.

We also propose using the Open Vulnerability and Assessment Language (OVAL). OVAL provides an automated way to gather information on host configurations, such as applications installed and services running, and to check hosts for software flaws and misconfigurations. An XCCDF document can call OVAL definitions as needed and use the returned data in its analysis and reporting. At this time, OVAL definitions are available for several widely used operating systems, and additional OVAL definition development is ongoing.

Automation mechanisms would also be helpful in extracting vulnerability and attack mitigation information from security controls such as antivirus software, vulnerability management utilities, and firewalls. Much of this information can be extracted from product management consoles and host logs, but efficiency and consistency would be improved if open formats such as XCCDF and OVAL were used.

5. Conclusions

The proposed framework will provide consistent, largely automated measures of host security that are based on operational and experimental host security data. The framework can be used for quantitative risk assessment, attack and vulnerability modeling, and other purposes. Much work remains to be done on completing the framework components and then testing the entire framework.

6. References

[1] M. Abedin, S. Nessa, E. Al-Shaer, and L. Khan, "Vulnerability Analysis for Evaluating Quality of Protection of Security Policies", *Proceedings of the 2006 ACM Workshop on Quality of Protection*, ACM, Alexandria, Virginia, October 2006, pp. 49-51.

[2] M. D. Aime, A. Atzeni, and P. C. Pomi, "AMBRA – Automated Model-Based Risk Analysis", *Proceedings of the 2007 ACM Workshop on Quality of Protection*, ACM, Alexandria, Virginia, October 2007, pp. 43-48.

[3] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal Security Hardening Using Multi-Objective Optimization on Attack Tree Models of Networks", *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ACM, Alexandria, Virginia, October 2007, pp. 204-213.

[4] D. J. Leversage and E. J. Byres, "Estimating a System's Mean Time-to-Compromise", *IEEE Security & Privacy*, IEEE Computer Society, January/February 2008, pp. 52-60.

[5] R. P. Lippmann and K. W. Ingols, "An Annotated Review of Past Papers on Attack Graphs", Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts, March 2005, http://www.ll.mit.edu/IST/pubs/0502_Lippmann.pdf (current 03/2008).

[6] A. Mayer, "Operational Security Risk Metrics: Definitions, Calculations, Visualizations", Second Workshop on Security Metrics, Boston, Massachusetts, August 2007, https://securitymetrics.org/content/attach/Metricon2.0/Mayer_Metricon-Final.ppt (current 03/2008).

[7] P. Mell, K. Scarfone, and S. Romanosky, "A Complete Guide to the Common Vulnerability Scoring System Version 2.0", Forum of Incident Response and Security Teams, June 2007, http://www.first.org/cvss/cvss-guide.html (current 03/2008).

[8] National Institute of Standards and Technology, "The Information Security Automation Program and The Security Content Automation Protocol", http://nvd.nist.gov/scap.cfm (current 03/2008).

[9] K. Sallhammar, B. E. Helvik, and S. J. Knapskog, "A Framework for Predicting Security and Dependability Measures in Real-Time", *International Journal of Computer Science and Network Security*, Seoul, Korea, March 2007, pp. 169-183.