Information Assurance and Security Letters.
ISSN 2150-7996 Volume 2 (2011) pp. 008-012
© MIR Labs, www.mirlabs.net/iasl/index.html

A Proposed Model for Vulnerability Analysis Using Critical Paths

Michael Reeves
mike.reev0@yahoo.com

Abstract: **Determining the means by which an attacker may compromise a given system is the main purpose of vulnerability assessment. As such, there are several models currently in place to track vulnerabilities. Some models focus on the susceptibility of a computer system as a whole where other models track attack paths through a network. This paper proposes a model which accomplishes both. The primary focus of the model is to identify synergistic attacks--consisting of multiple exploits used in tandem; thereby, resulting in a greater threat than the individual exploits alone. By using this data, a critical path can be identified; thus, revealing the exploit combination posing the greatest risk. Applying the critical path in conjunction with attack origins expands the diagram to depict attack vectors. Exploding the diagram by applying the model to all systems on the network with attack vectors depicts the entire network as a whole. Information from the model can then be used to harden both systems and the network, maximizing the benefits of the added security measures**

Keywords: Vulnerability Assessment, Critical Path, Attack Vectors, Management and Documentation, Standardization.

I. Introduction

As computers are relied upon more frequently for everyday business transactions they are equally an asset as wells as a liability. Computer crime, such as data theft or destruction, has been exponentially on the rise in recent years. Thanks to the diversity of software typically on computer systems today, an attacker may utilize any number of means to infiltrate a system or network. For this reason vulnerability assessment exists: to identify vulnerabilities, mitigate risks, and if possible eliminate threats. Unfortunately, vulnerability assessment methods used today do not approach analysis in the same manner as an attacker strategizes.

Computer criminals have the distinct advantage of being able to choose their targets and plan their attacks with a specific target in mind. Conversely, computer security personnel are tasked with protecting a network from all possible venues of attacks. This paper outlines a vulnerability assessment model that illustrates susceptibility to synergistic attacks, similar to the process in which an adversary would plan and deploy attacks. This approach is designed to identify the most effective mitigation for preventing system compromise.

II. Vulnerability Assessment

Every program has the potential of containing vulnerabilities, which degrade or circumvent security mechanisms in place. Vulnerabilities exist due to limitations of protocols in use, human error in writing program code (both logical and typographical errors), and improper configuration. Standards in vulnerability reporting dictates the following information be tracked for each vulnerability: origin, impact, software affected, date of discovery, and available vender patch(s) [1][2][3][4][5]. Additionally tracked information may include the existence of hot fixes and workarounds, known exploits, technical details of known exploitation methods, and a list of files that are created or altered as a result of exploiting the vulnerability [2]. Several online databases exists that track vulnerabilities in accordance with and addition to industry standards (e.g. Secunia, Symantec's Security Focus, and McAfee's Advisories).

A. Data Fields

1) Origin

The origin of a vulnerability identifies where the attacker may initiate their attack [2][3][4]. The three possible origins are 'local,' 'remote,' and 'both.' 'Local' origin attacks can only be initiated from the vulnerable system itself. 'Remote' origin attacks are initiated from a different system via a network connection. A vulnerability which can be exploited either through the vulnerable system or from a remote system has an attack origin of 'both.'

2) Impact

The impact of a vulnerability defines potential results of exploitation [2][3][4]. There are five different impact types: 'denial of service,' 'information release,' 'privilege escalation,' 'code execution,' and 'system compromise.' 'Denial of service' deprives legitimate users access to resources and services. 'Information release' discloses information that would not otherwise be available. 'Privilege escalation' elevates a user's rights and access to the system beyond the previously existing confines. 'Code execution' allows rouge code to be run. 'System compromise' is the outright unauthorized access to the computer. As defined here, 'system compromise' does not necessarily entail administrator-level compromise. A vulnerability is not limited to a single impact and may have multiple (or possibly all) impact types.

3) Software affected

The software affected defines the specific product and version [2][3][4]. Affected software is identified by vender, product name, and version. Some vulnerabilities may affect multiple products from numerous venders. In addition, some versions of a product may be unaffected while other versions are susceptible.

4) Date of discovery

The date which the vulnerability was discovered is generally tracked for statistical reasons [2][3][4]. The longer a vulnerability is unaddressed the more likely an exploit will be created before a resolution is available [6].

5) Vender Patches

Vender patches identify all available software updates that are designed to resolve specific vulnerabilities [2][3]. Once a patch has been applied, the software is no longer considered susceptible to the vulnerability.

6) Hot fixes and Workarounds

Hot fixes and workarounds resolve vulnerability susceptibility through means other than a vender patch [2][3]. These may include the installation of additional software, the removal of software, or configuration changes which once implemented negates the vulnerability. Hot fixes and workarounds effectively serve the same role as vender-provided patches but may not always be an acceptable solution due to operational requirements.

7) Exploits

Exploits are known or proven methods to utilizing a vulnerability [2]. They may be automated or implemented manually. In the case of vulnerability assessment reports, the exploitation tools (and source code if available) or manual procedures are identified and documented.

8) Exploitation Technical Details

The details of an exploit identify information above and beyond the specific code or procedure used in the exploit [2][3][4]. This information identifies the technical interworking of the known exploit and how the system is affected.

9) Altered Files

Where the exploitation details identify how the exploit works and the specific effects on the system, all residual changes may be used to identify that an exploit was in fact used [2][3]. Information regarding the created and altered files can assist in identifying successful vulnerability exploitation attacks; simultaneously identifying which specific exploit was employed and how to revert the system to a previous state without the need of full disaster recovery. This information is most useful above and beyond the realm of vulnerability assessment, such as during incident response.

B. Using Assessments

Traditionally, vulnerability assessment reports are used to create an evaluation of a program, system, or network as a whole [7]. Software is evaluated based on the associated vulnerabilities where as individual system assessments are based on the sum of the vulnerabilities for all applications installed. Individual system analysis is commonly automated using vulnerability assessment software such as Nessus, Retina, and OpenVAS [6].

Network vulnerability assessment consists of dissecting the network into protected security zones [7]. These zones are assessed as a whole through the sum of the vulnerabilities of the systems within the zone. The connections between security zones are analyzed, totaling all vulnerabilities which can permit an attacker to traverse from one zone to another. With network-based vulnerability assessment, the overall depiction of the zones and their interconnections acts as the overall assessment of the network.

III. Critical Path Analysis

Critical path analysis is used in project management to identify the tasks that if delayed would in turn elongate the entire project [8][9]. Critical path analysis is possible through identification of task requirements and creating a chronological sequence of tasks. As a result, tasks which can be performed (or in-progress) concurrently are also identified. The order-of-operations information is then diagramed, depicting the time and resource requirements of each task. The connecting lines between sequential tasks show the paths to project completion.

The critical path is the longest path from project start to completion, because each task must be accomplished in the diagrammed sequence [8][9]. By identifying the critical path, project managers are able to readjust resources, therefore shortening the time required to complete tasks on the critical path; thus, reducing the time needed to complete the project. Once resources are reallocated, in-turn changing the length of the path, the analysis must be re-accomplished as another path may become the new critical path. Re-analyzing the critical path stops when resources are optimally distributed, minimizing the overall project time length.

IV. Proposed Model

This paper proposes a model which views vulnerability exploitation as tasks of an attacker's project--outright compromise of the targeted system. This approach differs from actual project management in that not all vulnerabilities must be exploited in order to compromise a system. Where in project management the critical path is the sequence of tasks that require the longest time to complete before the project can be finished, this model's critical path is the fewest tasks (which in both cases reveal the minimum time needed for project completion).

In order to model a system's critical path toward complete compromise, all known vulnerabilities of all software on the system being analyzed must be identified. At a minimum, the origin, impact, and criticality of each vulnerability must be tracked. However, this model is designed to identify synergistic attacks, which is only possible with extensive information about the known exploits and their possible use to satisfy requirements of other vulnerabilities which would otherwise be unavailable.

As an option, this model can be used to either identify each individual vulnerability or group automated exploits that are known to employ vulnerabilities in tandem as a single task. By using individual vulnerabilities as critical path tasks, more granularity of the critical path is shown. Clustering vulnerabilities that are automated reveals the true complexity of exploitation, possibly presenting a more accurate picture of the time between initial attack and system compromise--thanks to the benefits of automation.

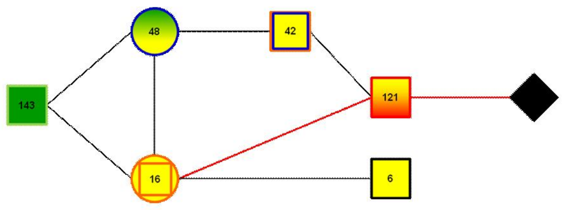
A. Diagramming Standard

For standardization purposes in this paper, visualization of each diagramed vulnerability represents origin, impact, and criticality. Additionally, each vulnerability is given a unique number for identification purposes. These diagramming standards are not based on any existing diagramming protocol, and can be adjusted as needed (for use and readability). Attack origin is identified by shapes: a square (local), a circle (remote), and a square inside a circle (both). The specific impact(s) of a vulnerability is depicted by the color outlining each shape: black (denial of service), light green (information gathering), orange (privilege escalation), blue (code execution), and red (system compromise). Multiple impacts are depicted by multiple layers of outlines.

Lastly, the fill color of the shape represents criticality ranging from green for low, yellow for medium, and red for high. Exploitation Progression flows from left to right. The far right of the diagram contains a black diamond representing potentially successful administrator-level system compromise. Paths between vulnerabilities and attack vectors are displayed using a solid black line where the critical path is emphasized with a bold red line.

B. Model Diagram Creation

Each vulnerability must be reviewed to determine if it can compromise the system. If no singular vulnerability can compromise the system outright, any additional requirements beyond the capabilities of exploiting the vulnerability (in order to compromise the system) must be identified. These requirements are compared against the impacts of the exploitable vulnerabilities on the system. Attacks that can be used to facilitate the requirements for other exploits are then connected and arranged in sequence (based on order-of-operation requirements) and connected to the system compromise diamond if appropriate. An example diagram is depicted in Figure 1.



Legend

Origin	Impact	Criticality
□ Local	Denial of Service	🟩 Low
○ Remote	Information Gathering	Low-Medium
Both	Privilege Escalation	🟨 Medium
	Code Execution	Medium-High
	System Compromise	🟥 High

Figure 1. Critical Path Analysis Model for one system.

Each path consisting of more than one vulnerability illustrates a synergistic attack sequence. The shortest path, defined previously as the critical path, is the greatest threat to the systems because it requires the least amount of work for the attacker.

Hardening the system against exploits of vulnerabilities along the critical path greatly decreases the risk of system compromise. Similar to resource reallocation used in critical path analysis for project management, system hardening may change the critical path; thus, requiring the analysis to be re-accomplished (taking into account the new mitigations in place). This process of analysis, hardening, and reanalysis continues until no further hardening can be implemented, either due to technical limitations or operational needs.

C. Expanded Model

Because the proposed model identifies the possible origins of vulnerability exploits, the model can be expanded to identify systems with the connectivity necessary to attempt exploitation of the system being modeled. Of course, any remote system that cannot connect to the system does not need to be documented in the expanded model. By identifying which systems may potentially be used to exploit vulnerabilities of remote origin, an attack vector is revealed. The attack vector is used to visualize where an attacker must either come from or go through (including local access to the system itself). An example of the expanded model containing attack vectors for a single system is depicted in Figure 2.

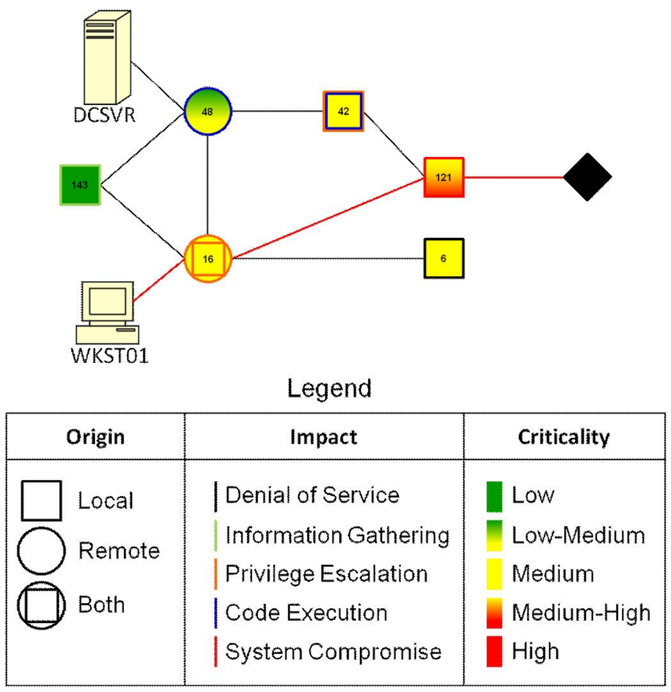


Figure 2. Expanded Critical Path Analysis Model for one system including attack vectors.

Aside from the internal threat, attacks generally originate from the Internet. Through continuous expansion, this model can trace attack vectors through the network with the intention of identifying attack vectors whose ultimate source is the Internet. In order to identify attack vectors of Internet origin, each attack vector (and its corresponding remote system) resulting in system compromise must also be diagramed. Once diagramed, the process is repeated on the remote system that can potentially compromise the targeted system using the attack vector. This outward spiral of system analysis would continue until no additional remote systems exist. If one of the systems is directly exploitable from the Internet, the Internet-initiated threat vector is identified.

Through the expanded diagram, a network view is illustrated. Additionally, because critical path analysis includes the attack vectors, it demonstrates the fewest systems necessary to be used as a pivoting point by an external attacker in order to eventually compromise the targeted system (the system in which the model diagramming began).

D. Exploded Model

By applying this expanded model to every system on the network (or at least every critical system), the diagram can be exploded, assessing the entire network. By mitigating key vulnerabilities, not only can critical paths be elongated (requiring more work for the attacker and potentially preventing system compromise outright), but total attack vectors can be elongated or eliminated.

Where the normal and expanded models focus on a single targeted system, the exploded model focuses on the network as a whole (or a critical network segment). Like the expanded model, each system's critical path and attack vector are identified. However, due to the web-like nature of networks and the number of systems being diagramed the exploded model is exponentially more difficult to analyze than the simpler expanded model.

E. Maintenance and Upkeep

Though initial implementation is quite time consuming, as vulnerabilities are mitigated the overall diagram naturally becomes more simple and manageable. Like all other vulnerability assessment models, the diagram instantly becomes outdated as soon as a new vulnerability which is applicable to the system (or "a system" in the case of the expanded or exploded model) is discovered. Therefore, constant maintenance is needed to keep the model up-to-date and provide an accurate assessment.

V. Model Shortcomings

While this model identifies realistic attacks which can propagate through a system or network, it is extremely taxing on the part of the modeler. Assessing a single system may take days or weeks, depending on the amount of software installed on the system and the number of vulnerabilities for the corresponding software. An expanded model would multiply the time required by the number of remote systems in the identified attack vectors. Similarly an exploded model would multiply the time needed by the total number of systems on the network (or at least by the number of critical systems, if that is the scope in which the exploded model is applied).

The time required is not simply for diagramming the model but also for analysis of synergistic attacks. Additionally, accurate analysis can only be accomplished by knowledgeable analysts who are familiar with the vulnerability implications and corresponding exploitation means. There is no pure automation capability for this proposed model at this time. Even if this model were able to be automated with existing tools, the readability of the model would be decreased in relation to its scope.

VI. Future Work

Currently no software is available which inherently tracks the data and metadata necessary to automate diagramming this model; however, the process can be assisted through a robust database. Capturing metadata regarding the technical effects of the vulnerabilities and correlating these results with other vulnerability prerequisites is the first step in automating this model. In addition, the database must be able to identify the security mechanisms in place and how potential hardening relates to the vulnerability requirements.

In addition to the database back-end, a front-end interface is needed to present the model. To enhance comprehension, the model could be broken down into different views of the data. A drill-down view of the diagram improves visualizing attack vectors by encapsulating individual system software vulnerabilities. The drill-down capability also allows for system-specific vulnerability information to still be accessible. By contrast, a fully diagrammed view of all vulnerabilities and corresponding attack vectors is the only way to identify every vulnerability along the critical path. Of course, filtering the full diagram to a specific network segment refines the scope (like described earlier with the exploded model).

Artificial Intelligence technologies have been proven to enhance the analytical capabilities of intrusion detection systems [10][11]. A similar implementation may possibly aid in the discovery and correlation of synergistic attacks. Given the correct metadata, logical neural networks can make correlations that might otherwise become overlooked. Fuzzy logic may improve the quality of the metadata and subsequent calculations required.

In the future, a system designed around this model (able to fully take advantage of the model) could identify critical paths and attack vectors based on existing or previous knowledge of vulnerabilities on the system(s) being analyzed. Such a system may reduce the modeling time exponentially and identify paths missed by a human administrator (who would have limited knowledge of all vulnerabilities and exploits discovered). Increasing the readability of the model and its context is possible by refining views of the data, such as a drill-down view and network segment filtering.

References

[1] D. Waltermire, S. Quinn, K. Scarfone, A. Halbardier. "The Technical Specification for the Security Content Automateion Protocol (SCAP): SCAP Version 1.2". *Special Publication 800-126 Revision 2*, National Institue of Standards and Technology, Gaithersburg MD, 2011.

[2] M. Schiffman. "The Common Vulnerability Reporting Framework". *An Internet Consortium For Advancement of Security in the Internet (ICASI) Whitepaper*, Internet Consortium for the Advancement of Security in the Internet, pp. 3-5, 2011.

[3] "Guidelines for Security Vulnerability Reporting and Response". *Version 2.0*, Organization for Internet Safety, pp. 18-20, 2004.

[4] P. Mell, K. Scarfone. "A Complete Guide to the Common Vulnerability Scoring System Version 2.0". *Common Vulnerability Scoring System (v2)*, National Istitute of Standards and Technology, Gaithersburg, MD, 2007.

[5] S. Harris. *All In One CISSP Exam Guide Fifth Edition*, McGraw Hill, New York, pp. 1133-1138, 2010.

[6] M. Gregg, B. Haines. *CASP CompTIA Advanced Security Practitioner Study Guide*, John Wiley & Sons, Inc., Indianapolis, pp. 193-194, 269, 2012.

[7] R. Trost. *Practical Intrusion Analysis*, Learning Solutions, New York, pp. 119-138, 2010.

[8] J. Marchewka. *Information Technology Project Management*, John Wiley & Sons, Inc., New Jersey, pp.185-186, 2010.

[9] "A Guide to the Project Management Body of Knowledge Third Edition". *ANSI/PMI99-001-2004*, Project Management Institute, Newtown Square, PA, pp. 145-148, 2004.

[10] V. Das, V. Pathack, S. Sharma, Sreevathsan, M. Shrikanth, G Kumar. "Network Intrusion Detection System Based on Machine Learning Algorithms", *International Journal of Computer Science & Information Technology*, II (6), pp. 138-151, 2010.

[11] L. Anyanwu, J. Keengwe, G. Arome. "Dynamically Self-adapting and Growing Intrusion Detection System", *International Journal of Multimedia and Ubiquitous Engineering*, V (3), pp. 15-22, 2010.

Author Biography

Michael Reeves Born in Greenbay, Wisconsin, Michael has worked professionally in the Computer Security field for over 10 years. In 2011, he earned a Masters of Science degree in Information Technology (with a specialization in Information Assurance) from the University of Maryland University College, located at Adelphi Maryland. Previous, in 2008 Michael received a Bachelors of Science in Management of Computer Information Systems from Park University, located at Parkville Missouri. Additionally, he holds CISSP, A+, Network+, Security+, and Linux+ certifications.