Vulnerability Evaluating Based on Attack Graph

Chunlu Wang[1,2], Yu Bao[1,2], Xuesen Liang[1,2], and Tianle Zhang[1,2]

[1] School of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing, China
[2] Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing, China
wangcl@bupt.edu.cn

Abstract. Networked hosts are facing more and more threats due to software vulnerabilities. Every year, there are an increasing number of security vulnerabilities discovered in software. It is impractical that we patch all the vulnerabilities because of the high cost of patching procedure. In this paper, we propose a user environments based scoring method. We analyze vulnerability impact from three aspects: confidentiality, integrity and availability. The score is customized to reflect the vulnerability's risk under certain security request by assigning the weight on the three aspects according to the host's function in an organization. We use attack graph to analyze the relationships among vulnerabilities in a host, and calculate on the context to get each vulnerability's threat. The experimental results indicate that our scoring method can better reflect the real situation.

Keywords: vulnerability, network security, risk assessment.

1 Introduction

Over the past decade, we have seen an ever-increasing number of security incidents reported as well as the vulnerabilities discovered in software. As described by CVE [1] official web site, a vulnerability is "a mistake in software that can be directly used by a hacker to gain access to a system". According to the statistics published by CERT [2], the number of vulnerabilities has grown in the last five years as shown in Fig. 1.

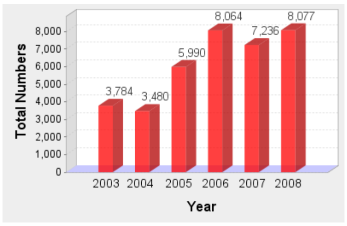

Fig. 1. Reported vulnerabilities from 2003 to 2008

One may consider the system is secure from attacks after all vulnerabilities are patched, but for many organizations, keeping each vulnerability patched is unrealistic and sometimes undesirable. Firstly, there are so many new vulnerabilities discovered each year. Existing scanners can find the latest vulnerability by keeping them update, but for some vulnerabilities, they only provide the characteristics but no solutions. System managers are also incompetent to solve these problems. Secondly, unconfirmed patches may bring the system into instability and introduce more bugs. Thirdly, patching on OS kernel level often needs to be rebooted, and some organizations are intolerant of availability being affected.

In order to keep the organizations safe, the security managers must have a clear image of which hosts are most critical and execute system security checks when new vulnerabilities are published or when new hosts are installed in strict rotation. Security manager has to ensure that any un-patched vulnerabilities will not be exploited, or would not cause much cost even exploited. Since it's not practical that we patch all discovered vulnerabilities, we have to face the following problem: Which vulnerability needs to be patched first? To answer this question, the managers need to understand the risk and potential damage of each vulnerability to the hosts. Such an understanding is hard to achieve only by reading daily vulnerability reports from various sources even from the automated security tools. Modern sophisticated intrusions usually consist of multi-stage attacks which combine multiple vulnerabilities, while most security tools typically focus on identifying individual vulnerabilities, and have no clue about which and how vulnerabilities can be combined for an attack.

This paper introduces a way to analyze vulnerabilities based on the host's customizable security request of Confidentiality (C), Integrity (I) and Availability (A) and the combined effect of vulnerabilities. We calculate the threat of individual vulnerabilities to certain host by using the context provided by the attack graph, and then prioritize them and give advice to the manager. The rest of the paper is organized as follows. In section 2, related work of this paper is discussed. In Section 3, we present how we evaluate the vulnerabilities and explain the related concepts. The calculations of analysis using attack graphs are given in section 4. Section 5 shows the detail of the method a through an experiment. Conclude in Section 6.

2 Related Work

When analyzing the system risk, we can get the information about vulnerabilities from various vulnerability databases, for example OSVDB [5], Security Focus's vulnerability database [9], and Public Cooperative Vulnerability Database [6]. But until the creation of CVE which is a common identifier, it is hard to share data across separate databases. Now CVE Identifiers are frequently used and are easily cross-linking with other repositories that also use CVE identifiers. In our work, attack graph will be talked about. Phillips and Swiler first proposed the method that uses attack graphs for analyzing network security [8]. In the graph, nodes represent network states, and its edges represent the application of an exploit. The path is a series of exploits leading to the goal of an attacker. Since this method can not automatically generate attack graph, it can not be used in large-scale network security analysis. Ritchey and Ammann [9] first used model checker to analyze network vulnerabilities. The advantage of using model-checking approach is that we can use existing model checkers rather than write an analysis engine. A model checker can check the model against a security formula, and then a counterexample shows the attack path that leads to the violation of the security property. In Ritchey's method it can only give one counterexample. Sheyner [10] improves the model checker which can give all the counterexamples. However, in model checking, most state transition sequences are unnecessary for network security analysis and lead to combinatorial explosion. In order to cut down the space and time complexity, the monotonicity is proposed. It states that gaining more privileges can only help the attacker in further compromising the system and there is no need for backtracking. Based on the monotonicity, Ammann, et al. proposed an approach where dependencies among exploits are modeled in a graph structure [11]. The method described in this paper assumes the same monotonicity property, and is compatible with other attack graph generation method.

3 Evaluating Aindividle Vulnerability

There are many vulnerability "scoring" systems, for example, CERT/CC [2], SANS [12], Microsoft's proprietary scoring system [13] and CVSS [4], each of which has its metrics and vulnerability database. Organizations use different labels to index the vulnerabilities. In this paper we choose CVSS as our vulnerability scoring system, which is designed to provide an open and standardized method for rating IT vulnerabilities. The National Vulnerability Database [3] (NVD) provides CVSS metrics for almost all known vulnerabilities, e.g. Fig. 2.

Vulnerability: CVE-2007-3168
Access Vector: Network exploitable
Access Complexity: Low
Authentication: Not required to exploit

Fig. 2. A vulnerability example

An attack scenario is a series of vulnerability exploitation with the attacker's privilege escalated. Detailed privilege classification method was determined by operating systems. In order to eliminate the diversity of privilege levels, most vulnerability databases only provide three levels: admin, user and other. The risk levels are in a decreasing order of admin > user > other. We think that the actual situation could not be properly described only by privilege and it may lead to an underestimation of potential risk. Instead of privilege, our method divides vulnerability impact into three aspects: C, I and A, and uses a three-dimensional vector(x_1, x_2, x_3) to characterize vulnerability's degree of loss on C, I and A. Each component has three levels of degree: None (N), Partial (P), and Complete (C). The corresponding values are listed in Table 1.

$$v = (x_1, x_2, x_3);\ x_{1,2,3} \in \{N, P, C\} \tag{1}$$

Table 1. Impact degrees and values

Level	Value
N	0
P	0.275
C	0.660

When a vulnerability cause complete loss of confidentiality, integrity, and availability, it equals to providing a root privilege, while vulnerabilities that give user privilege can be represented with only partial loss of C, I and A. Let Impact (*v*) denotes the impact value of *v*, and the formula is:

$$\text{Impact}(v) = 1 - (1 - x_1) \times (1 - x_2) \times (1 - x_3) \tag{2}$$

The above formula came from CVSS equations, and we made little change. A ratio that makes a score rang from 1 to 10 was removed from the old equations. Because we think it is unnecessary in our model and doesn't affect the relationship of vulnerabilities.

Modern enterprise network consists of many computers and other equipments, which take different responsibility. For example, some are running a HTTP server, and some have databases installed for confidential information storage. For the web server, availability is more important than confidentiality; while for a database server, quite contrary. Thus availability volatized vulnerability will not have the same effect on these hosts, but get a same score. More importantly, sometimes sensitive information can be obtained by an attacker without privilege escalation. So to make a rational evaluation, both the vulnerabilities and host's security requirement should be taken into consideration. A Weight Group (*W*) can be assigned to customize the security requirement on CIA of a certain host by the system administrator.

$$W = (\alpha, \beta, \gamma), 0 \le \alpha, \beta, \gamma \le 1; \alpha + \beta + \gamma = 1 \tag{3}$$

α, β and γ are preference weights for C, I and A respectively. These weights enable the manager to customize the way we evaluate vulnerabilities depending on the function of the host to a user's organization. That is, if a host is used to store confidential document for which confidentiality is most important, the manager should assign a higher weights to α, relative to β and γ.

Given a host with specific security requirement, we assign the *W*. Risk is the weighted impact of a vulnerability.

$$\text{Risk}(W, v) = 1 - (1 - \alpha x_1) \times (1 - \beta x_2) \times (1 - \gamma x_3) \tag{4}$$

4 Calculation on Attack Graphs

When analyzing sophisticated intrusions, we find that multiple vulnerabilities can be combined together for reaching a goal. During the attack, vulnerability may be a step stone of others and can still keep its effect after the exploitation. In this paper, we use attack graph to help analyze the threat of a host instead of a large network. After reducing the scale of problem the attack graph analysis can be done in desirable time. By using the context provided by the attack graph, we calculate each vulnerability's contribution to the system's compromise and get the ranked list of all vulnerabilities to help the administrator make priority remediation. Another weakness of many previous approaches is that information used to build the graph is usually freely formatted, which requires extensive manual analysis of vulnerabilities and attacks. Our approach extracts most information from NVD, where well-formed data can be easily accessed. We try to make our method more compatible with different design and easier to implement. Because the attack in a host is monotonic, the attack graph is a Directed Acyclic Graph (DAG) or an attack tree. Each node is a vulnerability and the edges mean the exploitations. A path from root to leaf indicates a successful attack.

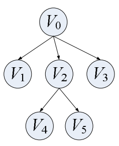

Fig. 3. Attack graph

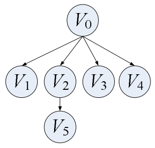

Fig. 4. Attack tree

The root of the tree v_0 represents a start point which contains no vulnerability information.

A. Attack Complexity(AC)

AC measures the complexity of the vulnerability required to be exploited once the conditions are complied. It can be regard as the likelihood of a successful attack. In CVSS [4], AC is a variable that has three values: 0.31(H), 0.61(M) and 0.71(L). Let **AC**(v) be the function to get the AC of v.

B. Base Score(BS)

BS is an overall score of a vulnerability ranging from 0 to 10. A vulnerability scored 10 is one of the most critical vulnerabilities. We define a function **BS**(v) to achieve the base score of v from the databases. Once we give a list of vulnerabilities to **BS**(*vulnerability list*) it returns the sum of each vulnerability's base score.

Next, we define two functions to manipulate the attack tree:

a) **Father**(v_i) returns the father of the node v_i on the tree. Take the example of Fig.3, **Father**(v_2)= v_0, Father(v_0)= Ø.

b) **Children**(v_i) is a node list of v_i's direct children, for example, in Fig.3. **Children**(v_0)=***V***=(v_1, v_2, v_3), **Children**(v_1)= Ø

C. Attack Factor(AF)

AF describes how likely an attacker is going to exploit each vulnerability under a certain condition. When building the attack tree the vulnerabilities which have been exploited together with the ones providing less privilege than current one will not be the next target. Attack factor of v depends on the proportion of **BS**(v) among all reachable vulnerabilities.

$$AF(v) = \frac{BS(v)}{BS(Children(Father(v)))} \tag{5}$$

D. **Success Probability(Prob)**

Success probability measures the likelihood of a vulnerability to be successfully exploited. The Prob of v to be exploited equals is the product of the probability of its conditions to be met and the probability to be chosen multiplying the attack complexity. So the formula goes like:

$$\text{Prob}(v) = \text{Prob}(Father(v)) * AF(v) * AC(v) \tag{6}$$

and we set **Prob**(v_0) equals to one.

E. **Threat**

Threat is an overall score of vulnerability in its host. It integrates the possible harm and successful probability.

$$\text{Threat}(v) = \text{Risk}(W, v) * \text{Prob}(v) \tag{7}$$

5 The Experiment

In this section, we perform an experiment based on real situation. A server runs Serv-U under Windows XP to provide a FTP service. In order to make our demonstration short and clear, all the operating system vulnerabilities have been patched. After a full vulnerability scan is done, five Serv-U vulnerabilities are found: CVE-1999-0219, CVE-2000-1033, CVE-2001-0054, CVE-2005-3467 and CVE-2004-2111. There are many vulnerability scanners available such as Nessus and OVAL Scanner, the usage will not be elaborated here. Here we use v_1 to v_5 to represent these vulnerabilities. Some characteristic are listed in the Table 2.

Table 2. Sample vulnerabilities' characteristic

Node	CVE-ID	Privilege	Impact on C, I, A	Description
v_1	1999-0219	/	(N,N,C)	Buffer overflow caused dos
v_2	2000-1033	user	(P,P,P)	Unrestricted brut forcing of user accounts.
v_3	2001-0054	/	(P,N,N)	Directory traversal
v_4	2005-3467	/	(N,N,P)	Denial of service
v_5	2004-2111	admin	(C,C,C)	Stack-based buffer overflow

v_1 is a buffer overflow vulnerability that could allow a remote attacker to create a denial of service, causing a complete loss of availability on the host. v_2 allows remote attackers to guess the passwords of other users. v_3 can cheat the server into allowing a remote attacker access to any directory on the FTP server's disk partition by a command containing specially crafted hexadecimal encoding, causing a partial loss of confidentiality. v_4 is an unspecified Denial of Service vulnerability. v_5 could be exploited by a remote authorized attacker to ultimately execute instructions with the privileges of the Serv-U server process, typical administrator or system. It could possibly be exploited by the use of the booty of v_2.

A simple attack graph is generated, as shown in Fig. 4.

Here we consider a normal attacker who has neither privilege nor any authorized account. And there are no firewalls between the FTP server and the attacker. This server is used to store and share public business data. If this server can not provide stable service, it will affect daily work. So the administrator set the security request a higher weight on availability rather than on confidentiality and integrity, the value of w is (0.2, 0.2, 0.6). First of all, we calculate each vulnerability's impact score and risk score based on (2) and (4), the base score of vulnerability (defined by CVSS) is also listed in Table 3.

Table 3. Sample vulnerabilities' scores

Node	Base Score	Impact score	Risk score
v_1	7.8	0.6600	0.3960
v_2	7.5	0.6189	0.2543
v_3	5.0	0.2750	0.0550
v_4	5.0	0.2750	0.1650
v_5	8.5	0.9607	0.5449

The outcome clearly shows the improvement after we introduced the weight group based on the user environment. v_4's risk is three times higher than v_3's risk. Thus vulnerability's risk is no longer constant, but changeable under different security requests. This enables the analysis closer to our actual situation. Then the correlations of the vulnerabilities are calculated by formulas (5), (6), (7).

Table 4. Sample vulnerabilities' prob

Node	Attack complexity	Attack factor	Prob(Fther(V_i))	Prob
v_1	Low (0.71)	0.3084	1.0000	0.2190
v_2	Low (0.71)	0.2964	1.0000	0.2104
v_3	Low (0.71)	0.1976	1.0000	0.1403
v_4	Low (0.71)	0.1976	1.0000	0.1403
v_5	Medium (0.61)	1.0000	0.2104	0.1283

Table 5. Threat of each vulnerability

Node	Threat
v_1	0.0867
v_2	0.0535
v_3	0.0077
v_4	0.0232
v_5	0.0699

Finally we calculate threats from v_1 to v_5, ranked as $v_1 > v_5 > v_2 > v_4 > v_3$.

6 Conclusion

In this paper, a new methodology for vulnerability analysis has been presented. This methodology correlates the vulnerabilities and the possibility of successful attacks and the security requests of certain asset. In order to make our method widely applicable, we describe our work at an abstract level. The risk of a particular vulnerability was analyzed based on user environment, and the threat was calculated according to the context of attack graph. Other methods can be compatible too if essential information can be provided. Vulnerability databases can be changed dynamically. In our method, the thread of a vulnerability is getting high either its risk is high or the total vulnerability number is small. Our task is to provide a priority remediation list, only used for host patch up.

Acknowledgements. The research is supported by China Natural Science Foundation (60973009) and China Postdoctoral Science Foundation(20100470256).

References

1. CVE, `http://cve.mitre.org/`
2. CERT/CC,CERT/CC Statistics (2004-2008), `http://www.cert.org/stats/cert_stats.html/`
3. NVD, `http://nvd.nist.gov/`
4. CVSS, `http://www.first.org/cvss/`
5. Open Source Vulnerability Database (OSVDB), `http://osvdb.org/`
6. Public Cooperative Vulnerability Database, `https://cirdb.cerias.purdue.edu/coopvdb/public/`
7. Security Focus Vulnerability Database, `http://www.securityfocus.com/vulnerabilities`
8. Phillips, C., Swiler, L.: A graph-based system for network-vulnerability analysis. In: Proceedings of the New Security Paradigms Workshop, NSPW 1998 (1998)
9. Ritchey, R.W., Ammann, P.: Using model checking to analyze network vulnerabilities. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 156–165 (2001)

10. Sheyner, Haines, J., Jha, S., Lippmann, R.: Automated generation and analysis of attack graphs. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 254–265 (2002)
11. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA, pp. 217–224 (2002)
12. SANS Institute. SANS Critical Vulnerability Analysis Archive. Undated (cited March 16, 2007)
13. Microsoft Corporation. Microsoft Security Response Center Security Bulletin Severity Rating System (November 2002) (cited March 16, 2007)
14. Sheyner, O., Wing, J.: Tools for Generating and Analyzing Attack Graphs. In: Proc. of Workshop on Formal Methods for Comp. and Objects, pp. 344–371 (2004)
15. Jha, S., Sheyner, O., Wing, J.: Two formal analyses of attack graphs. In: Proceedings of the 15th IEEE Computer Security Foundations Workshop, pp. 49–63 (2002)
16. Ingols, K., Lippmann, R., Piwowarski, K.: Practical Attack Graph Generation for Network Defense. In: Proc.of Comp. Sec. App. Conf., pp. 121–130 (2006)
17. Noel, S., Jacobs, M., Kalapa, P.: Multiple Coordinated Views for Network Attack Graphs. In: Workshop on Visualization for Computer Security, Minneapolis, MN, USA, October 26, pp. 99–106 (2005)
18. Dawkins, J., Hale, J.: A Systematic Approach to Multi-Stage Network Attack Analysis. In: Proceedings of the Second IEEE International Information Assurance Workshop (IWIA 2004) (2004)
19. Jajodia, S., Noel, S., O'Berry, B.: Topological analysis of network attack vulnerability. In: Kumar, V., Srivastava, J., Lazarevic, A. (eds.) Managing Cyber Threats: Issues, Approaches and Challenges. Kluwer Academic Publishers, Dordrecht (2003)
20. Wang, L., Noel, S., Jajodia, S.: Minimum-cost network hardening using attack graphs. Computer Communications 29(18), 3812–3824 (2006)