Automated Vulnerability Management of Computer Systems

H.T. Tian  L.S. Huang  Z. Zhou  J.L. Shan
Department of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui
CHINA

Abstract: - With the continuous flood of vulnerabilities of computer systems, vulnerability management is a very important task for administrators to keep systems as secure as possible. However current manual vulnerability management by administrators is very time-consuming and error-prone. This paper proposes an open framework of automated vulnerability management that dramatically alleviates the burden of administrators and improves the security of systems. In this framework, all of the vulnerability information and system information is expressed using XML to support automatic processing. A Host Vulnerability Manager (HVM) running on the target host maintains the crucial system information, decide what vulnerabilities exist according to the system information and vulnerability information from various sources and try to fix them automatically. The domain vulnerability manager (DVM) is responsible for the vulnerability management of the local network, DVM correlates reports from HVMs and scan for network-based vulnerabilities in this domain. We have implemented a prototype of the framework that shows the effectiveness and efficiency of the solution.

Key-Words: - XML, system security, vulnerability management, vulnerability assessment, automated patching

1 Introduction
With networks increasingly proliferating into virtually every aspect of our daily life, security has gained more and more importance from the perspective of today’s network’s increasing role in modern automation and control. Unfortunately the speed at which information technology is advancing guarantees that there will always be a sequence of new, exploitable security vulnerabilities, which means weaknesses in a system allowing unauthorized action [10].

According to the latest Symantec Internet Threat Report [1], there was an 81.5% increase in computer vulnerabilities during 2002. Overall, some 450 new viruses and 250 new vulnerabilities are discovered globally each month, and these require system updates and patches. It also shows that 85% of active attacks were classified as ‘reconnaissance’ – the cyber equivalent of a burglar checking doors and windows to see if they are locked. Only 15% of attacks were actual exploitation attempts – the burglar entering the building. Most attackers are looking for commonly known vulnerabilities in a network. If they fail to find them, they are unlikely to pursue their attack; instead they will seek out an easier target. Symantec Internet Threat Report shows that 76% of attacks over the last six-month period were opportunistic and 24% were targeted.

We can see from above data most attackers aim at systems with commonly known vulnerabilities and consequently most security incidents of computer systems arise from one or more security vulnerabilities in target systems that are not properly fixed or patched by administrators. Therefore vulnerability management of networked computer systems is a very important task in the security area for system administrators to keep system as resistant as possible to existing and newly discovered attacks.

This paper proposes an integral automated framework of vulnerability management of computer systems that dramatically alleviates the burden of administrators and improves the security of systems. The rest of this paper is organized as follows: first we give a systematical description of vulnerability management and current situation in section 2, then the design of the automated vulnerability management framework is elaborated in section 3, we have implemented a prototype of the framework and introduce it in section 4. Finally conclusions and future work is presented.

2 Vulnerability Managements and Current Circumstance
In theory, vulnerability management (VM) of computer systems is simple and the philosophy is: start with systems that have no known vulnerabilities and when security vulnerabilities that affect those systems are announced, quickly apply patches to keep the systems invulnerable. But in practice, VM is
far more complex and daunting that includes four main steps as depicted in Fig.1.

![Vulnerability Management Process](image)

2.1 Get and Maintain System Information
Effective VM starts with knowing what systems are to be managed in the management domain (MD) which maybe a host or a local network. In other words, we should set up and maintain an accurate inventory of system information (such as the hardware type, OS version, services running, third-party applications etc).

There are two basic approaches now: manual and automated. The manual approach depends on capable people accessing each system, gathering the information and entering it into a database or spreadsheet. The automated approach involves purchasing and installing software agents that gather this information. One well-known vendor product is Tivoli Inventory [7].

2.2 Get Vulnerability Information
Getting the reliable, entire vulnerability information of target domain timely is prerequisite for VM. There are many free sources of vulnerability information such as Bugtraq[2] , CERT[4], and ICAT Metabase[5] etc. There are also some sources of purchasable vulnerability information like Security Intelligence Alert from SecurityFocus[2] , SecurityTracker[6] etc.

There are no automated tools for administrators acquire customized vulnerability information because the vulnerability advisories from different sources have different formats and terminologies and are mainly ambiguous text-based description. According to a survey by SecurityFocus[2] system security administrators now spend an average of 2.1 hours/day hunting for security information relevant in all kinds of security bulletins and mailing lists.

2.3 Assessments and Evaluation
This is to determine what vulnerabilities exist in the MD presently and evaluate them for risk assessment, which are the most difficult and the core of VM.

A number of free and commercial security vulnerability assessment tools (such as Nessus[8], ISS[9] etc.) exist. Most tools are network-based, meaning that they scan the system under test from another location on the network. Some tools also have a host-based component that is installed on the actual test system and reports back to a management station. But unfortunately because of different private vulnerability databases and assessment mechanism they have, the results of these tools are not precise enough and not consistent at all [11][12]. So these tools can at most be used separately for assistance in the procedure and currently individual knowledge and experience of system administrators play a key role in the judgment of existence and seriousness of vulnerabilities.

2.4 Fixing or Patching
Once a patch is available, some testing needs to be done before patching production systems to ensure the effectiveness and reliability of the patch. Without an available patch, some workarounds should be taken to fix the vulnerability.

Currently, it is the system administrator's responsibility to locate the necessary files or steps needed to update the system from the results of assessment and evaluation. Once finished, the administrator can optionally test the system for possible side effects from the update. If the administrator notices an instance of system instability, they can rollback to an earlier system state.

2.5 Current Problems
From above analysis, we can see that although there are some aided tools in some phases of VM, but they can only be used separately in each step to help the administrators. Current vulnerability management is generally a manual, time-consuming job. As it is mainly based on the individual knowledge and experience of administrators, VM is also error-prone and leads to certain exposure of systems to attackers.

Even worse point is that VM has to be a ‘non-end’ circular process to keep the system secure, which
means the above four steps should be often repeated time and time whenever necessary (such as issue of new vulnerabilities, installation of new software, configuration change of systems, after a suitable maintenance interval etc). The ‘process nature’ of VM makes it unbelievably time consuming.

Moreover, as more people and businesses turn to the web, there will be a continuous increase in the number of systems to be managed, (imaging an administrator in charge of 100 hosts) which makes things worse. Administrators often view their systems within the context of their private networks, but we must begin view the Internet as a distributed network in which a problem in one node affects all. Poor VM by some administrators can later affect other networks. Your network might be secure, but others that aren't can be used to send spam, attempt a DoS attack, and cause problems that have potential to affect us all. So every system in the Internet should have a good VM. That is difficult to achieve in current solution.

In fact, system administrators in general are swamped by the flood of vulnerabilities and related patches being released. Various research initiatives including the recent survey by the Department of Trade and Industry of UK[3] have revealed that most breaches occur through known vulnerabilities that are not properly fixed by administrators, which indicates that current manual solution VM is seriously time-consuming, error-prone and should be improved.

There is some research relevant to vulnerability management such as CVE naming mechanism [13], managing vulnerabilities using CVE [16] and Unix vulnerability management [17] etc. Unfortunately we have not seen any research on automated vulnerability management that we think is the way forward.

3 An Open Framework for Automated Vulnerability Management
An open framework we proposed here will provide an integral, automated vulnerability management solution without the intervention of administrators.

3.1 High-level View
As depicted in Fig.2, there are two basic components in our automated VM framework: Host vulnerability manager (HVM) responsible for VM of individual host, and domain vulnerability management (DVM) for management network-based vulnerabilities on a local network. HVM stores and maintain system information of the host, receive vulnerability advisories directly from vulnerability provider (VP) or from the buffer of DVM, then assess, evaluate vulnerabilities and take some action according to the policy set by host administrators. HVM also report the system information and assess result to the DVM. Based on the reports from the HVMs under its control, DVM stores and maintain system information of the local network, scan for the network-based vulnerabilities among the hosts in the local network and take some measure on the policy set by domain administrators.

![Automated Vulnerability Management](image-url)

**3.2 Get and Maintain System Information**
To know just what system exists, HVM store and maintain a local XML file we called system information indicator (SII). As the format of SII, we present the system information markup language (SIML) to describe an entire system using XML.

Elements in SIML includes mainly hardware manufacturer [e.g., Sun, HP, IBM, Intel], hardware configuration (processor[s], network card[s], etc.), OS type and version (e.g., Solaris 8, Red Hat Linux 7.2, Windows2000), services running (e.g., DNS, SSH, NFS), third-party applications (e.g., Oracle database, Apache Web Server), and primary system function (e.g., Internet Web server, intranet Web server, mail server, back-end database server, individual workstation, etc.). What’s more, for specific software SIML also contains the elements of patches (and workarounds), the address of the supportive update service (when possible), and the rules indicating how the HVM will react when new vulnerability information is received from the support service or other sources. It is emphasized that adoption of XML grammar makes SII machine-processable.

Ideally, we hope all software could register itself and the support service in SII when installation. But it would be impossible to get software vendors to deploy such a feature so we will actually focus on
getting the HVM to register the software and collect all the system information like [7]. Administrators can also manually complete the SII. SII must have a standard location on the system to allow for software to register during installation and to allow for manual operation. When the HVM manager detects new software, it will subscribe to the support service given in the SII and ask the administrator to set rules for managing the software which is stored in SII too.

DVM also maintain a XML file called domain information indicator (DII) containing structural information about the hardware and software configurations of all the hosts in the network. DII is similar to SII except for some additional information from the view of the whole network such as network topology and host connectivity. We have designed network system description language using XML for the format of DII. DII is produced through the reports of SII of HVMs and some additional assessment of DVM. It also can be completed by manual approach by domain administrators.

3.3 Get Vulnerability Information
As mentioned in section 2-2, there are many free or commercial sources of vulnerability information called vulnerability provider (VP). But the terms and formats currently used in the field of computer vulnerability tend to be unique to different VPs, moreover most of this information is informal, text-based description and not machine-readable. So it can not currently be directly used in automatic process. It is necessary to provide a standard, machine-readable format of the vulnerability information, we have designed the XML-based common vulnerability markup language (CVML). Due to limit of space, we will only give a simple introduction here instead of the detailed schema.

There are mainly five parts in CVML:

- General information is contained now in most of vulnerability databases, such as id, name, CVE name (optional), references, related dates, revision history, summary etc.
- The element ‘check-existence’ provides related information about how to check for the presence of vulnerability on a computer system and multi-host vulnerabilities. Checking single-host vulnerabilities requires all of the platforms in which the vulnerability exists, the affected software consisting of name and version, affected files (optional) including the path and the name, the related configuration (optional) and the patch names (optional) with which the vulnerability will not exist. OVAL queries [14] can also be included for compatibility. How to check the existence of multi-host vulnerability precisely is still under research.
- The element ‘evaluation’ evaluates the vulnerability with three attributes: direct impact, total cost of exploit and rating. The direct impact tells the direct result caused by the vulnerability such as root access, system down, disclosure of sensitive data etc. Total Cost of Exploit means the total resources and skills (such as hardware and software resources and knowledge of specific system) needed to exploit the vulnerability. Rating gives the severity of the vulnerability in the manner of quality or quantity. Vulnerabilities with more severe direct impact and less total cost of exploit are more serious. HVM and DVM will evaluate vulnerabilities according to this.
- The element ‘solution’ contains information about how to correct the vulnerability. It consists of two parts: workarounds and patches. Workarounds provides operations taken in the absence of patching to avoid the vulnerability (such as disable vulnerable service, restrict buffer size with a system management tool, make specific configuration of the software etc). Patches is a sequence of Patch giving a short note and the URL pointing to a download site or a web service to fix the vulnerability, which can enable HVM (DVM) patch the vulnerability.
- The element ‘exploit’ is from the view of attackers and designed to support the automatic attack imitation and generation for HVM (DVM) to test the existence of the vulnerability. The subelements ‘precondition’ and ‘post-condition’ is for imitation and ‘procedure’ with exploit methodology for implement the test.

In our framework, HVMs can directly require and receive vulnerability information in the format of CVML from VP periodically. VP can also actively send information of newly discovered vulnerabilities to HVMs based on subscription service. We can also set up a vulnerability buffer in DVM and use DVM as proxy of all HVMs (such as in an organization) under its control for getting vulnerability information, which decreases interconnection with VP and improve the efficiency.

3.4 Vulnerability Assessment
After achieving the latest system information and vulnerability information, HVMs and DVMs will search for vulnerabilities on the host or in the network.

A HVM compares the ‘check-existence’ elements of all available vulnerabilities with entries in System Information Indicator (SII) to match out what vulnerabilities exist in the host. It goes through the vulnerabilities received one by one and check
whether the affected platforms consistant with the system, whether there is software with same name and version of affected software and whether there is a same file in the system with the affected file to prove the existence of a vulnerability. If there is a vulnerability, HVM will record it in a report to take some action.

Network-based vulnerabilities here means those caused by interconnection or interoperation between hosts in a management domain. Such as the incorrect position of a modem behind the firewall, an unencrypted password transmission path etc. A DVM mainly search for network-based vulnerabilities according to the information in Domain Information Indicator (DII) and some rules set up for judgement. However building a perfect set of rules is so difficult and complex that we are working for improvement.

Due to the different security requirements and policies, a sequence of ‘normal’ operations on different hosts however ultimately leads to a vulnerability exposed to attackers. To identify this kind of network-based vulnerability, DVM also uses a backward goal-based search method like[15] to give all the paths and give the alerts to domain administrators.

What we want to emphasize again here is the time and frequency for HVM or DVM to perform the assessment. Generally after a entire process of four steps of VM, a system will have no commonly known vulnerabilities for the time.

When some new vulnerabilities has been received from the Vulnerability Provider (VP) or DVM, HVM will immediately check for the existence of these vulnerabilities only. When some system configuration and deployment have been altered (such as installation of new software, disableing service running etc), HVM will ask for all the vulnerabilities relevant to the change from the VP or DVM and perform the assessment. For example, when Office2000 is installed, HVM will ask VP or DVM for all vulnerabilities related to Office2000 and check if some exist on the system.

3.5 Evaluating and Fixing
After knowing what vulnerabilities exist in the host, HVM will evaluate the vulnerabilities based on the ‘evaluation’ element in the vulnerability information with the format of CVML, and take some efficient action according to the ‘solution’ element and the security policy set by the system administrators for relevant software.

Evaluation of vulnerabilities, which should be performed by the VP, we think is based on the following factors. The ‘direct impact’ means the direct result caused by exploiting the vulnerability (such as root access, system down, Denial of Service etc). ‘Total Cost of Exploit’ means the total resources and skills (such as hardware and software resources and knowledge of specific system) needed to exploit the vulnerability. ‘Popularity’ means in what degree exploiting the vulnerability prevails on the whol Internet. We are trying to give a quantitative evaluation scheme to mark the severity of vulnerabilities now.

For example, if HVM finds that a vulnerability exists in the system and the ‘evaluation’ element of it gives ‘very serious’ or some figure with the same meaning, it will read the policy in SII set for the software that the vulnerability lies in. Assuming the policy is ‘patch when serious, alert when normal’, HVM will connect to the corresponding web site or web service referred in the ‘solution’ element of vulnerability for the patch file and hotfix the system.

4 Implementation and Results
We have implemented a prototype of our automated vulnerability framework and it works well as we expected.

First we have built a vulnerability database as Vulnerability Provider. This database has now most of the Microsoft related vulnerabilities, totally 1000 entries of vulnerabilities in the format of Common Vulnerability Markup Language, which comes mainly from CERT.

On a local network with eight PCs in our lab, we have deployed a HVM in each host and a DVM in a specific PC. Seven hosts are installed with WindowsXpPro, Win2000Pro, or Win98, and the other web-server is running Win2000 Server and IIS5.0. There are also all kinds of application software from Microsoft and third-party vendors.

In the prototype, HVMs automatically assess and patch the managed hosts for relevant vulnerabilities very well and made few mistakes. System administrators only need to set security policy for software and pay attention to the alerts. But DVM sometimes made unacceptable false positives and negatives, which show DVM’s performance should be improved.

5 Conclusion and Future work
With the continuous flood of vulnerabilities of computer systems, vulnerability management is a very important task for system administrators to keep systems as resistant as possible to existing and newly
discovered attacks. However current manual vulnerability management by administrators is generally very time-consuming and error-prone. This paper proposes an open framework of automated vulnerability management that dramatically alleviates the burden of administrators and improves the security of systems.

In this framework, all of the vulnerability information and system information is expressed using XML to support automatic processing. A Host Vulnerability Manager (HVM) running on the target host maintains the crucial system information, decide what vulnerabilities exist according to the system information and vulnerability information from various sources and try to fix them automatically. The domain vulnerability manager (DVM) is responsible for the vulnerability management of the local network. DVM correlates reports from HVMs and scan for network-based and multi-host vulnerabilities in this domain. We have implemented a prototype of the framework that shows the effectiveness and efficiency of the solution.

But there are also many open questions in this framework that we are working on. For instance how to check multi-host or network-based vulnerabilities more accurately while current result is not satisfying, how to evaluate the vulnerabilities quantitatively? How to check the effectiveness and reliability of the patch?

References: