Abstract: - System management, now mainly a manual process based on individual knowledge and experience of administrators, is now very time-consuming, inefficient, and error-prone. This paper proposes an open framework to design and implement self-managing systems that dramatically alleviates the burden of system administrators and improves the security and stability of systems. To be machine-processable, system information is described using XML in the format of system information description language (SIDL), and security advisory or update notification for specific software issued by vendors (third-parties) is also described using XML in the format of advisory and notification description language (ANDL). The agent system manager (SM) maintains inventory of system information in SIDL (including the hardware type, OS version, services running, third-party applications etc.). SM also listens for messages, which are described in ANDL, from a support service relevant to software registered with SM. Based on the rules set by system administrators, SM determines how to handle the situation automatically. We have implemented a prototype of the framework that showed the effectiveness and efficiency of the solution.

Key-Words: - XML, system management, self-managing system, security advisory, update notification

1 Introduction
Incredible progress has been achieved in almost every aspect of computing: microprocessor power up by a factor of 10,000, storage capacity by a factor of 45,000, communication speeds by a factor of 1,000,000. But along with that growth has come increasingly sophisticated architectures governed by software whose complexity now routinely demands tens of millions of lines of code. Some operating environments weigh in at over 30 million lines of code created by over 4,000 programmers! With the evolution of computers from single machines to modular systems to personal computers networked with larger machines, complexity is becoming the next grand challenge in the information technology area. Summarily computer systems have become increasingly difficult to manage and ultimately to use.

In fact, the growing complexity of the IT infrastructure threatens to undermine the very benefits information technology aims to provide. Up until now, we've relied mainly on human intervention and administration to manage this complexity. Therefore system management of computers is a most daunting task for administrators. Anyone can learn how to manage a system properly by following the vast amount of information available (such as security advisories, product updates etc.), but depending on the number of systems to be managed, you might have to look for too much information and the task is very time-consuming and error-prone.

This paper proposes a new framework to design and implement self-managing systems that dramatically alleviates the burden of system administrators and improves the security and stability of systems. To support automatic process, system information is described using XML in the format of system information description language (SIDL), and security advisory or update notification for specific software issued by vendors or third-parties is also described using XML in the format of advisory and notification description language (ANDL). The agent system manager (SM) maintains inventory of system information in SIDL (such as the hardware type, OS version, services running, third-party applications etc.). SM also listens for messages, which are described in ANDL, from a support service relevant to software registered with SM. Based on the rules set by system administrators, SM determines how to handle the situation automatically.

The rest of this paper is organized as follows: first we give the current status of system management and related solutions in section 2, then the framework of self-managing systems is elaborated in section 3, we have implemented a prototype of the framework and present the observation in section 4. Finally discussions and future work is presented.
2 Problems and Related Work

System management is an important task that we all should do—from end-users to system administrators. The main idea behind system management is to keep your system stable and secure. It sounds simple, but it has become one of the most daunting tasks for administrators.

2.1 Current System Management Process

Generally system management is a circular process as depicted in Fig.1.

![System Management Process Diagram](image)

We should start with knowing what systems are to be managed in the management domain (MD) which may be 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 [1].

Secondly it is needed to monitor the system in real time to see if there is any anomaly in the functionalities (for example no response of a web server to the request) or in the performance (for example very low response of a web server to the request). We should also need to monitor real time attacks from hackers by intrusion detection systems (IDS) deployed.

Getting the reliable, entire information relevant to system management (such as security advisories and the software updating notifications, bug-fixes etc.), which we call system management messages here, from various sources timely is prerequisite for the assessment phase. Typically vendors publish notifications of security alerts or performance improvement of their products. There are many sources of security advisories such as Bugtraq[2],CERT[4], and ICAT Metabase[5] etc. There are now no automated tools for administrators acquire customized system management messages because the advisories or notifications from different sources have different formats, terminologies. Moreover, most of them are ambiguous text-based description and not machine-readable.

Research on the results of the monitoring phase and system management messages relevant to the components of the inventory will determine what vulnerabilities or defects exist in the system presently and what to do to improve the security, stability and performance of the system. A number of free and commercial vulnerability assessment tools (such as Nessus[8], ISS[9] etc.) exist. But these tools can at most be used separately, partially for assistance in the procedure and currently individual knowledge and experience of system administrators play a key role in the result of assessment.

Finally Based on the results from the assessment phase, you might need to update or configure the system to make it stable and secure. It is the system administrator's responsibility to locate the necessary files or steps needed to update the system. 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. After updating the system, system inventory should be updated and a new cycle begins.

2.2 Problems

From above analysis, we can see that although there are some aided tools in some phases of system management now, but they can only be used separately, partially to help the administrators to synthesize all kinds of reports and take actions accordingly. System management is now generally a manual, time-consuming job. As it is mainly based on the individual knowledge and experience of administrators, system management is also error-prone.

Even worse point is that system management has to be a ‘non-end’ circular process to keep the system secure and stable, which means the above four steps should be often repeated time and time whenever necessary (such as receiving new messages, installation of new software etc.). The ‘process nature’ of system management 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 have to be managed, (imagining an administrator in charge of 100 hosts), which makes things worst. 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. 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 system management. That is difficult to achieve in current solution.

In fact, system administrators in general are swamped by the flood of security advisories, related patches and updating notifications 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 of system management is seriously time-consuming, error-prone and should be improved.

2.3 Related Work
There are many system management tools out there to ease the task of updating a system. Some examples are Microsoft's Windows Update[10], Red Hat's up2date[11], and other advanced tools for managing entire networks. Instead of performing all the steps manually, a tool can create the inventory, research for the latest updates, download the necessary files, update the system, and then possibly test the system. But tools like Windows Update and up2date are for managing specific pieces of software that are in the tool maker's database. This becomes a problem when you want to manage a piece of software that your management tool doesn't support. Besides Microsoft and Red Hat, there are many software vendors that offer management solutions. These enterprise management solutions also maintain private databases that keep track of service packs, updates, hotfixes, and other updates.

IBM has a good overview on their website about “autonomic computing” (also referred to as self managing systems) in which they suggests that there are eight defining characteristics of an autonomic system[12]. In another IBM’s paper “Autonomic Computing Concepts”[13], they detail self-management into self-configuring, self-healing, self-optimizing and self-protecting. IBM’s autonomic computing intends to embed the complexity in the system infrastructure itself, both in the hardware and software, and then automates system management. Autonomic computing proposed by IBM is splendid and may be the right direction of future development of the whole IT industry, but the design itself is too complex, and there are so many problems to resolve to implement it.

CIM stands for Common Information Model[14]. It is a conceptual information model for describing management that is not bound to a particular implementation. This allows for the interchange of management information between management systems and applications. This can be either "agent to manager" and "manager to manager" communications that provides for Distributed System Management. But to be managed using CIM successfully, applications and devices will have to be CIM-enabled or CIM-compliant. This leaves a big question over applications and devices that do not support CIM and whether or not vendors will attempt to incorporate CIM in future and existing products.

Desktop Management Interface (DMI)[15] generates a standard framework for managing and tracking components in a desktop pc, notebook or server. DMI was the first desktop management standard and a core technology of DMI is the Management Information Format (MIF). A MIF file describes manageable characteristics for both hardware and software components. Generally DMI has the same problem of compatibility as CIM, and is also too complex to be widely acceptable.

Summarily, there has been some valuable research and solution to simplify the process of system management, including the brilliant autonomic computing presented by IBM. But they do not meet the requirements of either being open, simple, backwards-compatible, or efficient.

3 An Open Framework for Self-managing Systems
Management is a very complex field and we think that the lack of open standards is one of the foremost reasons why it remains so complex. Our solution is the creation of open standards to cover the problems of software, system, and advisory description. More importantly, making these standards simple enough for administrators to use and yet robust enough to solve these problems.

3.1 High-level View
Our solution is simple and does not introduce new technologies that will leave out any piece of software. It leverages existing technologies to solve the problem. The architecture is very basic as depicted in Fig.2:

![Self-Managing System](image)

- A inventory of all the current system information including hardware and software is collected and maintained by the agent host system manager (HSM) in the format of system information description language (SIDL).
- When installed on the system or updated, a piece of software described in XML is written to the inventory and registered to the HSM.
- HSM listens for messages, which are described in advisory and notification description language, from a support service relevant to software registered with the HSM.
- The system administrator sets rules on how the HSM should handle the common scenarios described in messages (e.g. if vulnerability rating is "Critical", automatically update).
- Based on the rules set, the HSM determines how to handle the situation (e.g. automatically update the software).

To manage a local network, we can also introduce a domain manager (DM), which collect and maintain network-related information in the control domain (for example connectivity, topology of network device such as firewall, router etc.). DM is mainly to improve security and performance of the whole network. DM can receives reports from HSMs under its control and synthesize them to take corresponding action on the policy set by domain administrators.

3.2 Maintain System Information in SIDL
To know just what system exists, HSM maintain system inventory with the machine-readable format SIDL to describe an entire host system.

Elements in SIDL 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, SIDL also contains the version, patch number, the address of the supportive update service (when possible), and the rules indicating how the HSM should react when new security advisory or notification is received from the support service or other sources. It is emphasized that adoption of XML grammar makes system inventory machine-processable.

Ideally, we hope all software could register itself and the support service in system inventory when installed or updated. But it would be impossible to get all software vendors to deploy such a feature so we will actually focus on getting the HSM to register the software and collect all the system information like[7]. Administrators can also manually complete the system inventory. System inventory must have a standard location on the system to allow for software to register and to allow for manual operation. When the HSM detects new software in system inventory, it will subscribe to the support service indicated in the relevant entry of the software, and ask the administrator to set rules for managing the software. The rules are stored in system inventory too.

DM also maintains the domain inventory containing structural information about the hardware and software configurations of all the hosts in the control domain. Domain inventory is similar to system inventory except for some additional information from the view of the whole network such as network topology and host connectivity. Domain inventory is produced through the reports of HSMs and some additional assessment of DM. It also can be completed by manual approach by domain administrators.

3.3 Listen for Messages in ANDL
A system manager must have the ability to query the support service for the latest status of the registered software. It will send requests with entries in system inventory specific to certain software (such as the platform, name, version, patch number etc.) in the payload and the support service will respond with relevant ANDL in its payload. In a complex scenario,
a support service could broadcast messages to all subscribed system managers. The system management messages includes advisories and notifications. The word ‘advisory’ means a message conveying a warning message such as a vulnerability advisory. A ‘notification’ is an informational message such as a feature enhancement. As mentioned above, the terms and formats currently used in the messages from various sources tend to be unique to different providers, moreover most of this information is informal, text-based description and not machine-readable. So it is necessary to provide a standard, machine-readable format of message, we have designed the XML-based Advisory and Notification Description Language (ANDL). Due to limit of space, we will only give a simple introduction here instead of the detailed schema.

There are 3 main parts in ANDL:

The element ‘affected products’ indicates what kinds of systems are affected and should be improved according to the message. It consists of product name, version, build, architecture, and the affected files or configurations. For example the ‘affected products’ of a security advisory will tell in what systems vulnerabilities exist.

The element ‘evaluation’ evaluates the Severity/Impact of the vulnerability correction or the feature enhancement. The direct result caused by the vulnerability can be root access, system down, disclosure of sensitive data etc. The impact of a feature enhancement can be expressed in the manner of quality or quantity. Vulnerabilities with more severe direct impact or feature enhancement with high improvement should be pay special attention to. HSM and DM will evaluate the system management message according to this.

The element ‘solution’ contains information about how to improve the system. It consists of two parts: configurations or downloads. Configurations provide operations taken to alter the system configurations to avoid the vulnerability or improve the performance (such as disable vulnerable service, restrict buffer size with a system management tool, make specific configuration of the software etc). Downloads is a sequence of files to download and installed on the target system to improve the system. It gives a short note and the URL pointing to a download site or a web service to install the file.

3.4 Assessment and Improvement

After achieving the latest system information and messages, HVMs and the DVM will search for vulnerabilities or feature to enhance on the host or in the network.

HSM compares the ‘Affected Products’ of a message with with entries in System Inventory to match out what vulnerabilities exist in the host, what feature to enhance or if newer version of software is available. It goes through the messages received 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 match.

After knowing the vulnerabilities or the feature to be enhanced, HSM will evaluate the message with the ‘evaluation’ element in ANDL. Then based on the management rules set by the system administrators for relevant software, HSM take some action according to the ‘solution’ element.

For example, if HVM finds that a vulnerability exists in the system and the ‘evaluation’ element of it is ‘very serious’ or some figure with the same meaning, it will read the policy in system inventory 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 to in the ‘solution’ element of message for the patch file and hotfix the sytem.

What we want to emphasize again here is the time and frequency for HVM to performe the assessment. When some new messages have been received, HSM will immediately check for the applicability of the message to the target system. When some system configuration and deployment have been altered (such as installation of new software), HSM will ask for all the messages relevant to the change from the issuer and perform the assessment. For example, when Office2000 is installed, HSM will ask for all messages related to Office2000 and check if some is applicable to the system.

4 Implementation and Results

We have implemented a prototype of our open framework of self-managing systems and it works well as we expected.

First we have built a database as system management message provider. This database has lots of the Microsoft related messages, totally 1000 entries of advisories or notifications in ANDL, which comes mainly from the website of Microsoft.

On a local network with five PCs in our lab, we have deployed a HSM in each host and a DM in a specific PC. Three hosts are installed with WindowsXpPro,
Win2000Pro, and Win98 respectively, 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, HSMs automatically listen for messages from the provider, assess and improve the managed hosts for relevant message very well and made acceptable mistakes. System administrators only need to set security policy for software and pay attention to the alerts. But DM sometimes made unacceptable false positives and negatives, which show DM’s performance should be improved.

5 Conclusion and Future work

System management, now mainly a manual process based on individual knowledge and experience of administrators, is now very time-consuming, inefficient, and error-prone. This paper proposes an open framework to design and implement self-managing systems that dramatically alleviates the burden of system administrators and improves the security and stability of systems. To be machine-processable, system information is described using XML in the format of system information description language (SIDL), and security advisories or update notification for specific software issued by vendors (third-parties) is also described using XML in the format of advisory and notification description language (ANDL). The agent system manager (SM) maintains inventory of system information in SIDL (such as the hardware type, OS version, services running, third-party applications etc.). SM also listens for messages, which are described in ANDL, from a support service relevant to software registered with SM. Based on the rules set by system administrators, SM determines how to handle the situation automatically. We have implemented a prototype of the framework that showed the effectiveness and efficiency of the solution.

But there are also many open questions to be discussed in our solution. First, our solution mainly deals with self-managing from the perspective of automatically receiving advisories or notifications from trustworthy parties and taking corresponding action to improve the security or performance. It is toward the right direction but far from the ultimate self-managing systems in which self-configuring, self-healing, self-optimizing and self-protecting are finished real time without any human intervention. Human can’t feel the existence of the management at all, such as autonomic computing proposed by IBM, to be realized in the next generation IT infrastructure.

Even in this specific perspective, there are many problems unresolved. For instance, it is important and difficult to check the effectiveness and reliability of the correction proposed by the message. We should be very cautious in any updating of the system to keep it stable. And how to describe the system management policy or rules set by administrators in a more precise and comprehensive manner? It is necessary to get a suitable control granularity of the automated management. How to improve the performance of the Domain Manager? It is necessary but difficult to map the network topology precisely in a standard format. It is also a challenge to check out multi-host or network-based vulnerabilities more accurately while current result is not satisfying. In the future, we also want to integrate the real time monitoring of the system for instance intrusion detection into our solution.

References: