Architectural specifications and design for an automated vulnerability resolver

A.Al-Ayed, S.M.Furnell, D. Zhao, I.Barlow and M.Tomlinson

School of Computing, Communication and Electronics, University of Plymouth, Plymouth, United Kingdom.

Email : info@network-research-group.org

Abstract

Vulnerability management represents an essential task for the IT administrators, in order to safeguard systems against exploitation by attackers and malicious software. However, the management task is non-trivial, as a result of an increasing number of vulnerabilities and the workload implications associated with reading the incoming advisories and acting upon the resulting information. As a step towards addressing the problem, this paper presents the architectural design of an automated vulnerability resolver, which is designed to provide a vendor-independent means of vulnerability notification and rectification for system administrators. The architecture enables incoming advisory messages, from multiple sources, to be filtered and prioritised according the specific requirements of the target environment, and then provides an automated facility by which any associated patches can be obtained and deployed to affected systems. The paper describes the key elements of the architecture, and illustrates the viability by means of a prototype system.

Keywords

Vulnerability management, Security.

1. Introduction

In recent years, published statistics have continually illustrated the significant scale of the problem posed by vulnerabilities in system and application software. For example, Symantec's widely-cited Internet Security Threat Report shows that 1,432 new vulnerabilities were identified during the first half of 2003 – representing a 12% increase over the same period in 2002 (Symantec 2003). Meanwhile, statistics from CERT show that the total number of vulnerabilities reported to them in 2002 and 2003 were 4,129 and 3,784 respectively. While the latest figure represents a slight reduction, both figures are dramatically higher than the figure for 2001, which had been just under 2,500 (CERT 2004).

In order to be aware of the vulnerability problem administrators are obliged to monitor multiple sources of information rather than being able to rely upon one reliable source. For example, if the organisation runs operating systems and applications from different vendors, it is necessary to monitor different vendor advisories (e.g. by subscribing to their associated mailing lists). However, the current approaches to alerting only provide a partial solution. For example, Microsoft's Strategic Technology protection system only solves the problem for the recent versions of Microsoft Windows (i.e. it provides no help for other operating systems) (Microsoft 2003). The SANS @RISK: The Consensus Security Alert mailing list provides

multi-vendor information, but only does so on a weekly basis for the most critical vulnerabilities – thereby giving attackers an opportunity to exploit them before the notification is received (SANS Institute 2004).

Of course, in order to achieve effective vulnerability management, a system administrator not only needs to ensure that he receives notification of new vulnerabilities – he must also follow this up by rectifying any that are relevant to his system. However, each of these processes can be further decomposed into a number of sub-tasks. For example, once vulnerability notifications are received, they potentially need to be filtered to exclude those that are not relevant to the system configuration, then classified to group those that pertain to the same software or target, and then prioritized to ensure that the most critical issues receive attention first. Once this has occurred, the task of rectification often requires that appropriate patches are downloaded, and then distributed to and installed upon the affected systems. The traditional requirement has been for administrators to undertake these tasks manually. However, with the dramatic increase in both the number of vulnerabilities and the speed at which they are exploited, this approach is unsustainable because of the major workload implications introduced as a result. This paper therefore considers an architectural design to automate tasks within both the notification and rectification processes, and presents details of a prototype system that has been developed by the authors.

2. An architecture for automated vulnerability resolution

The proposed approach is intended to provide a comprehensive solution to the vulnerability problem, by enabling automation of both the notification and rectification phases. It is also intended to offer a flexible approach that can be customised to suit an administrator's needs, regardless of the vendors and products involved in the target environment.

The first element in the automated agent is automated notification which will reduce the problem of information overload for administrators, by filtration, prioritisation, and classification of the incoming advisories, which will produce only the relevant messages would enable administrators to direct their efforts more effectively, reducing the amount of time lost following up irrelevant material and enabling genuine problems to be addressed more quickly. This will solve part of the problem, but having obtained the relevant information they then have to act upon it. The second element of automation would be an active vulnerability resolver, capable of acting upon the notification data on behalf of the administrator.

The proposals contrast significantly with current marketplace solutions for vulnerability resolution. Although some current scanners do include auto-update features that enable them to be aware of and detect the latest vulnerabilities (eEye 2001), these require specific actions on the part of the product vendors, who must release the associated update for their product. Where products also include 'fix-it' technology, allowing administrators to rectify some issues automatically, these often relate merely to configuration details and do not take care of significant software upgrades or patches (Forristal and Shipley 2001). Some vendors, such as Microsoft, have also introduced automated software update facilities, but some network administrators are loathe to enable this feature in their clients as it may inadvertently harm operations if patches are not applied in a controlled manner.

The overall process associated with the proposed architecture is illustrated in Figure 1 and described below.

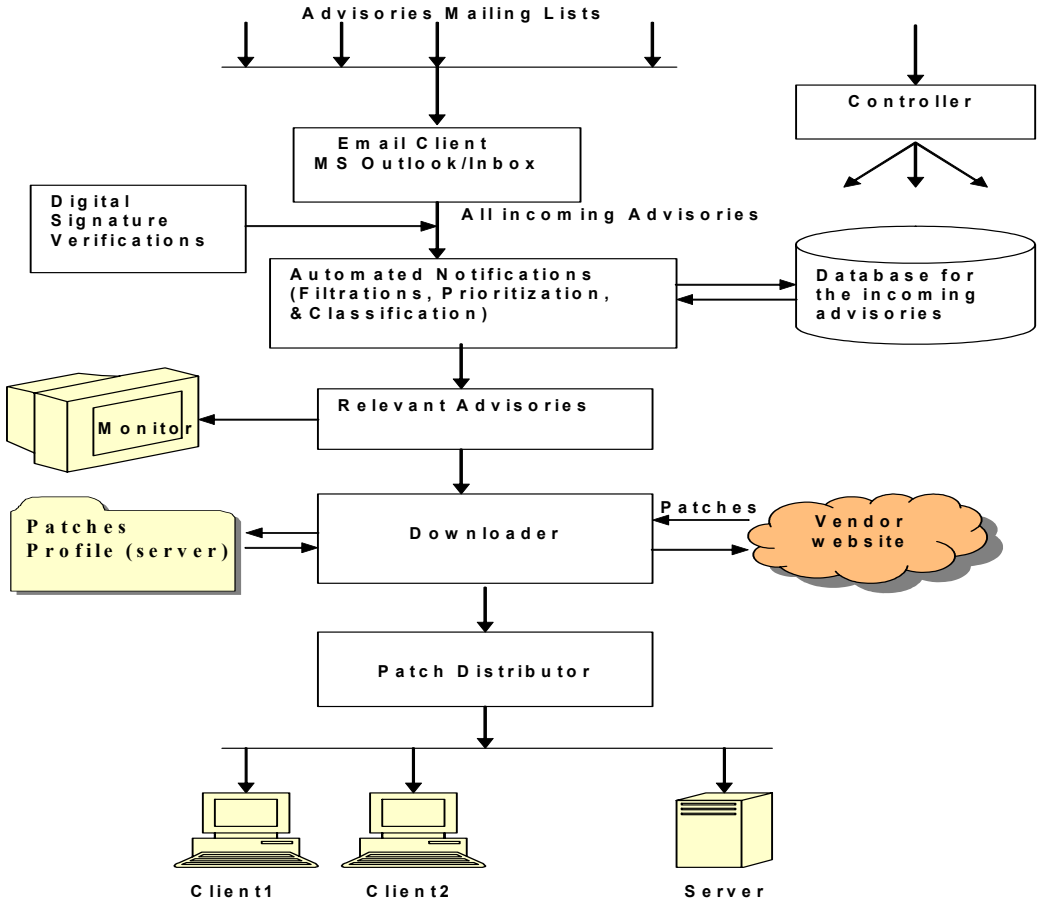

Figure 1 : Automated vulnerability resolution process

- **Email Client.** The *Email Client* receives the incoming advisories from the mail server and saves them in the inbox. In this research, Microsoft Outlook is used to provide this facility.
- **Digital Signature Verification.** The digital signatures on incoming advisories are verified, before passing them to the vulnerability resolver.
- **Automated Notifications.** All incoming mails of the *Email Client* are read, and vulnerability advisories are stored in a profile after verification of the vendor's digital signature. Classification, filtration and prioritisation of the advisories is performed according to administrator-specified criteria. The *Advisories Database* stores all the incoming advisories, along with classification data extracted from the original message.
- **Relevant Advisories.** The advisories that are relevant to the local system are displayed for the system administrator, and passed on for the rectification system to act upon.
- **Downloader.** The *Downloader* acts upon the *Relevant Advisories* by connecting to the vendor web site to download any specified patches, and consults the system administrator's policy for accepting them.
- **Patches Profile.** This module stores patches in a directory, which is later used by the *Patch Distributor.*

- **Patch Distributor.** Accepted security patches are sent to the network clients and servers.
- **Controller.** This module is designed for use by the administrator to configure the notification and rectification modules.

3. A prototype implementation

The architectural approach has been implemented within a prototype system, the key elements of which are described in the sections that follow.

3.1 Automated notification

In the ideal implementation, the notification system is designed to receive and filter messages, using a generic vulnerability-reporting format, which the authors have defined in earlier work (Furnell et al. 2002). However, in order to ensure some level of compatibility with existing advisories, the prototype system allows mappings to be defined between the generic format and the current format of vendor-specific advisories. Once a mapping is defined, the administrator can subscribe to an advisory mailing in the usual manner (e.g. sending a 'subscribe' message to the list, or providing contact details via a subscription web page), and build his own database of relevant advisories, based upon the local network configuration. The database is searchable, enabling the administrator to find the advisories according to desired characteristics (e.g. vendor, operating system, application version, severity, etc.), and then prioritise the resulting rectification. The classification window is illustrated in Figure 2a showing all searchable fields.

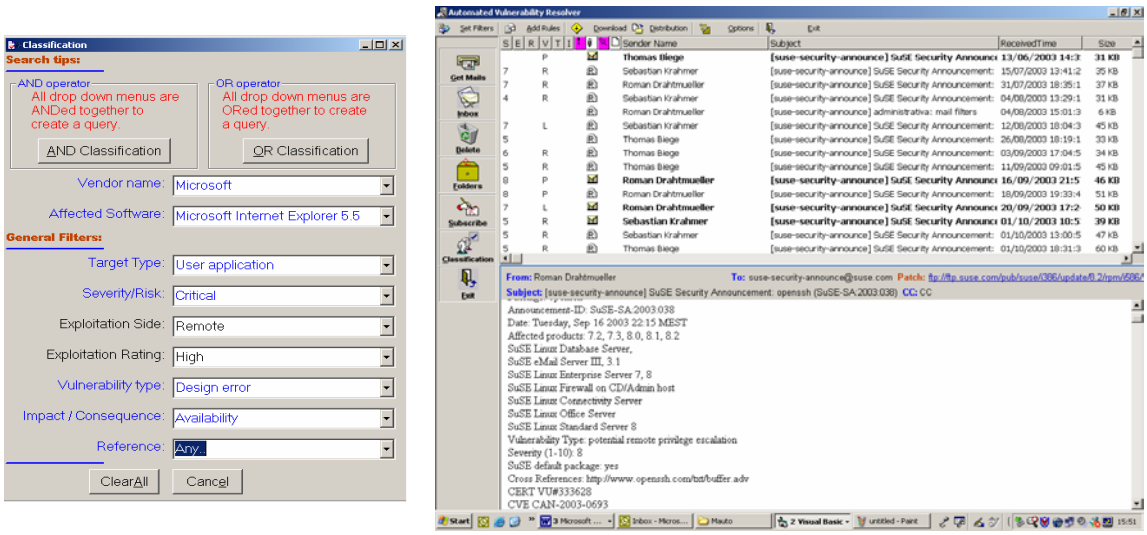

Figure 2: (a) Filtering options and (b) Filtered list of vulnerabilities

The advantage of the approach is that advisories from multiple sources (and originating in multiple different formats), can be dealt with from one interface and in a harmonised format. Figure 2b presents the main interface of the notification system, which is similar to a standard email client, displaying sender name, subject, received time and the like. In addition, however, other fields such as the severity of the vulnerability are extracted from the original

source message, and can be used to filter and order the messages received. This information enables the administrator to judge the importance of the message and can be used to draw his attention to it. Finally, if an associated patch is available, the source URL link is highlighted to the system administrator by displaying it on the upper part of the message form details (as shown in the figure).

3.2 Automated rectification

The second element of the Automated Resolver addresses vulnerability rectification, and the associated sub-architecture is illustrated in Figure 3. The main functionality is split into server and client side elements, as follows:

- **Server side modules**

  - **Controller**. This module enables the administrator to configure the rectification system. On the server side (i.e. the administrator workstation), such configuration would apply to the *Downloader* (e.g. behaviour characteristics to consider/prioritise when downloading the patches), the *Patch profile* (e.g. save all the downloaded patches into a profile directory), the *Network profile* (e.g. client name, location, and IP address) and the *Communicator* (e.g. schedule the time of installation). Meanwhile, on the client side, the Controller configures the *Patch profile* (e.g. save all the downloaded patches into a profile directory) and can initiate the installation of a patch on the client.
  - **Downloader**. This module downloads the patches from relevant messages. It is configurable via the Controller, to download a required patch either manually or automatically. The downloaded patches are saved into the patches profile, irrespective of the operating system of the patch.
  - **Patches Profile (server)**. This module saves the patches on the server (Administrator workstation).
  - **Distributor**. This module distributes the patches to network clients using this IP address, either automatically at a scheduled time, or manually by selecting the patch and then sending it to required clients.
  - **Network Information**. This module is a database for network information, including each machine in the network, its name, location, and IP address.

- **Client side modules**

  - **Client**. This module represents the target machine to which the patch will be sent and installed upon.
  - **Patches Profile (client).** This module saves the patches on the client when received from the server side.

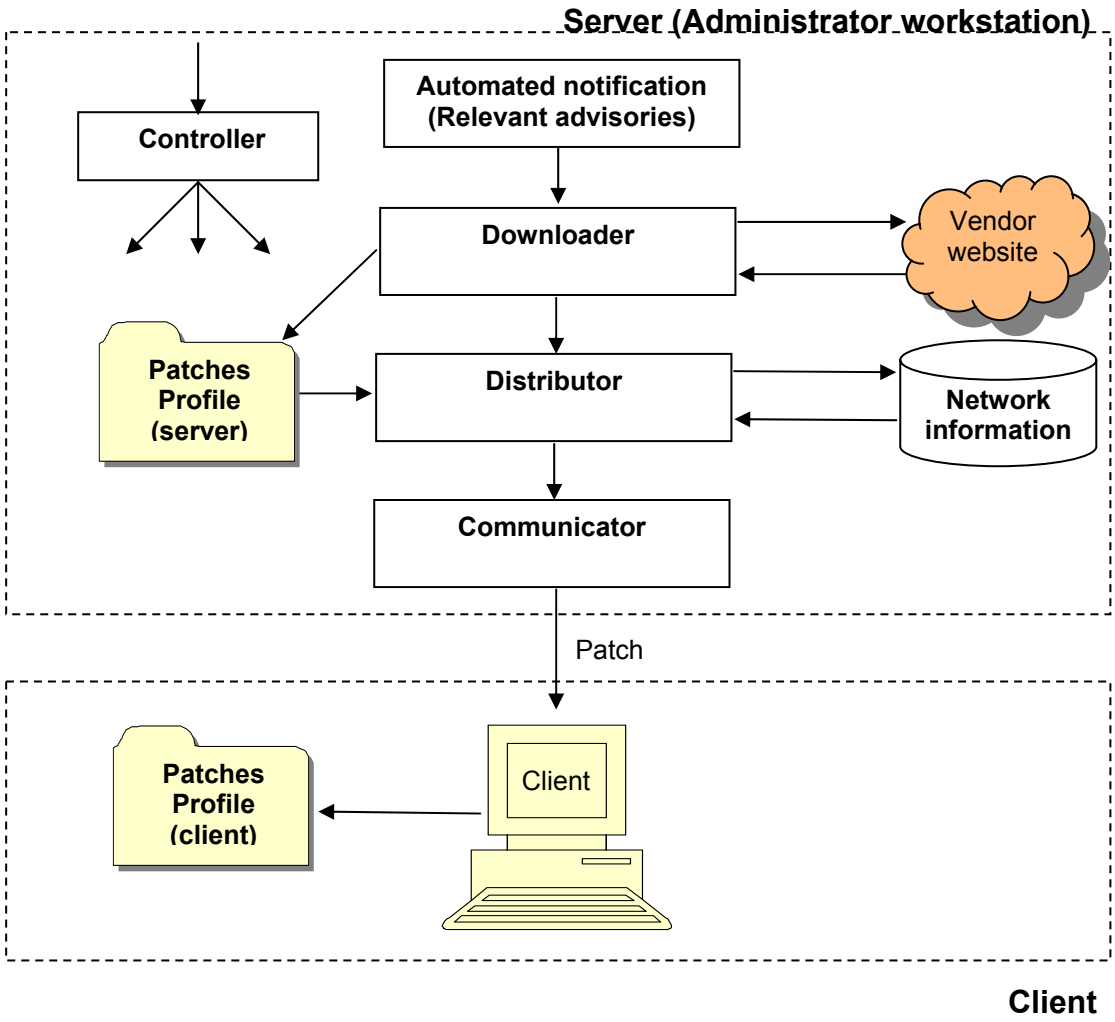

Figure 3 : Automated rectification system

A prototype of the rectification sub-system has been implemented in a referenced network environment on a Windows platform, and provides a proof-of-concept for the ideas presented in the main architectural work. It demonstrates the ability of an automated agent to initiate appropriate rectification actions to a number of actual vulnerabilities, based on the reference environment's network configuration. The main program including the Notification system, Rectification system, Downloader, Distributor, and Communicator are installed on the server (system administrator workstation). The client information (such as name, location, and IP address) is saved in a database on the server, and the administrator can send the patches manually or automatically by nominating the target client(s). The clients run a Communicator program to communicate with the server. After patches are received from the server they are saved into the download profile. After downloading, the administrator can install the patches in the client(s) at any time without any need to connect to the Internet. Figure 4 illustrates the patch distribution window, from which the administrator selects the patch to be deployed, and the client(s) to which it should be sent.

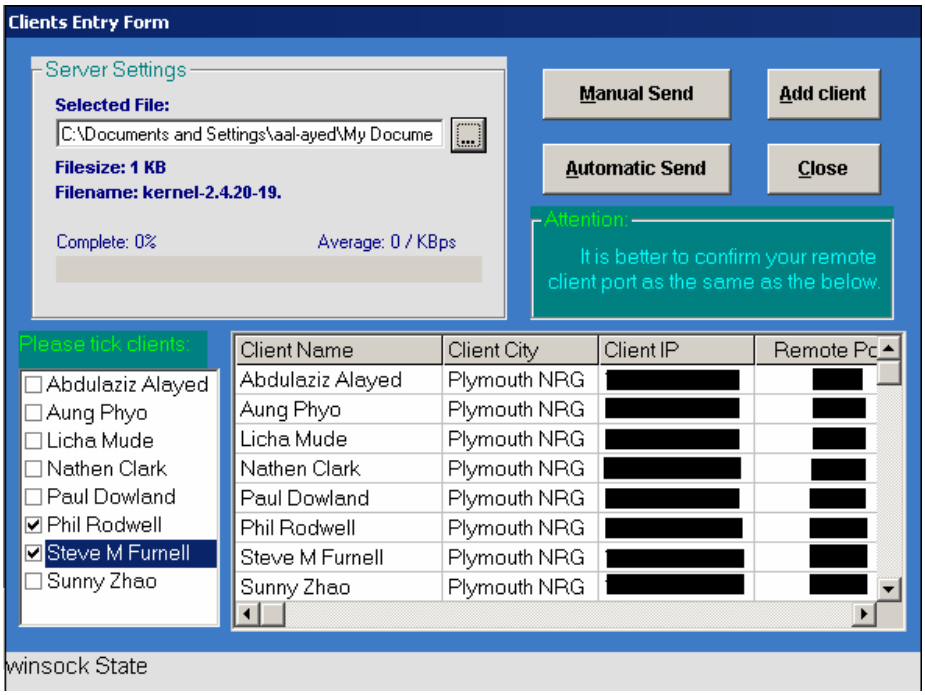

Figure 4 : Patch distribution interface

4. Conclusions

This paper has outlined the architecture of an automated vulnerability resolver, and presented details of a prototype implementation. The automated notification system verifies the source of the incoming advisories, and enables messages from multiple vendors to be processed and prioritised within a single administrative interface. This is supported by an automated rectification system that obtains required patches and allows them to be deployed to client systems.

Although the prototype is functional, there are still some issues to be addressed. Principal amongst these is further development of the rectification system to guard against it causing inconvenience or indirect denial of service to legitimate users (e.g. further options and intelligence within the system to ensure that it does not take the system down to install a patch whilst users are working). Furthermore, the framework must ensure the validity of the patches and corrections that it tries to apply, so as to prevent the rectification agent being misused by hackers as a means of getting the target system to accept malicious code. Nonetheless, the prototype system has succeeded in providing a harmonised front-end for administrators who would otherwise have to manually inspect and filter advisories from multiple sources, and then manually retrieve any associated patches. In this sense, several elements of the traditional administrative overhead have already been reduced.

References

CERT. 2004. “CERT Statistics 1988-2003”, *CERT Coordination Centre*. http://www.cert.org/stats/cert_stats.html

eEye. 2001. “Retina: The Network Security Scanner”, *eEye-Digital Security*. http://www.eeye.com/html/assets/pdf/retina_whitepaper.pdf.

Forristal, J. and Shipley, G. 2001. “Vulnerability Assessment Scanners: Detection Result”, *Network Computing.* 8 January 2001. http://www.networkcomputing.com/1201/1201f1b1.html

Furnell S.M, Al-Ayed A., Barlow I.M., and Dowland P. S. 2002. “Critical awareness-The problem of monitoring security vulnerabilities”, *Proceedings of European Conference on Information Warfare and Security*, 8-9 July 2002, Brunel, UK, pp85-92 2002.

Microsoft Corporation. 2003. “Microsoft Strategic Technology Protection Program”, 2 July 2003. http://www.microsoft.com/security/mstpp.asp.

SANS Institute. 2004. “SANS @RISK: The Consensus Security Alert”, http://www.sans.org/newsletters/risk/.

Symantec. 2003. "Symantec Internet Security Threat Report, Symantec", September 2003. http://ses.symantec.com/PDF/SISTR_sept2003_all.pdf.