Development of a framework for secure patch management

Lech J. Janczewski and Andrew M. Colarik
The University of Auckland, Department of MSIS, Private Bag 92019, Auckland, New Zealand,
Tel: 64 9 373 7599, ext 87538/83048, Email: lech@auckland.ac.nz or acolarik@auckland.ac.nz

ABSTRACT

With the growing expansion of Internet connectivity and usage by consumers and merchants, financial institutions and governmental entities, the requirement for stable and secure software is being elevated to the legislative and judicial levels. No longer can software manufacturers disregard this growing requirement in fulfilling their commercial obligations. The issuance of a patch is the beginning and not the end of a software developer's obligations to its customers and all subsequent parties impacted by its product.

In this paper, the authors provide a framework detailing the components of a secure patch management system, a discussion on the necessity of managing and securing each phase/component, and some basic patch issuance concerns with regards to the supporting legal environment.

INTRODUCTION

Books, journals and mass media are full of stories about people trying, intentionally or unintentionally, to cause damage to information systems. Every year, several internationally renowned organizations (like CSI/FBI or CERT) produce detailed reports indicating the nature of these activities [CSI/FBI, 2001]. An elaborate taxonomy of attacks [Denning, 1999] as well as many methods of handling threats has been created and developed by security organizations.

Generally, attacks prevention and handling could be accomplished in two major ways: implementation of the full risk analysis, like that promoted by Common Criteria [2000] or following the baseline approach [von Solms, 1996]. The first is very costly and detailed, while the second allows for a quick increase of system security, not necessarily tailored for a particular application. What is important, however, is the fact that both approaches consider development of a security policy as a fundamental requirement.

Security policy is usually divided into three distinctive parts [Forch, 1994]: prevention activities, handling of security alerts, and disaster recovery (security in time). These parts address detailed issues related to handling threats against hardware, software, personnel and organizational matters (security across domains).

One particular issue spans both (time and domains) dimensions of the security policy: introduction of new software components into a system. These new parts are generally prepared by the primary developers as a result of:

- Discovering some errors in the existing software,
- Enhancing the system with options previously nonexistent,
- Changes in the client requirements or environment.

In the past, contacts between a software developer and a user were very intimate as they usually were members of the same business organization. They may have even shared the same room. In contrary, more and more users are implementing products "off-the-shelf" and the contact between the developer and the end user is quite loose. They are usually far away, both in real distance as well as in the organizational space. Hence, the user must have detailed information about his/her system and communicate this to the software developer. On the other hand, the developer must provide good information to the user. Both sides must be aware of the way in which the proper selection and transport of the upgrades, method of their installations, payments for said upgrades, and any applicable warranties. Any fault in this process could have disastrous effects on the IT system. Thus, creating a dramatic situation for the whole business organization.

This paper proposes a framework for conducting secure patch management, examines some of the technical and legal management issues in the distribution and installation of patches, and suggests several ways of improvement.

SECURE PATCH MANAGEMENT SYSTEM COMPONENTS

For the purposes of this paper we define secure patch management as a system incorporating the following components (refer to Figure 1): A patch event notification, an integrity and authentication check of the patch event notification, a patch version applicability verification, patch transport and delivery, a patch file integrity verification, a trial installation of the patch, patch deployment, and recipient audit component.

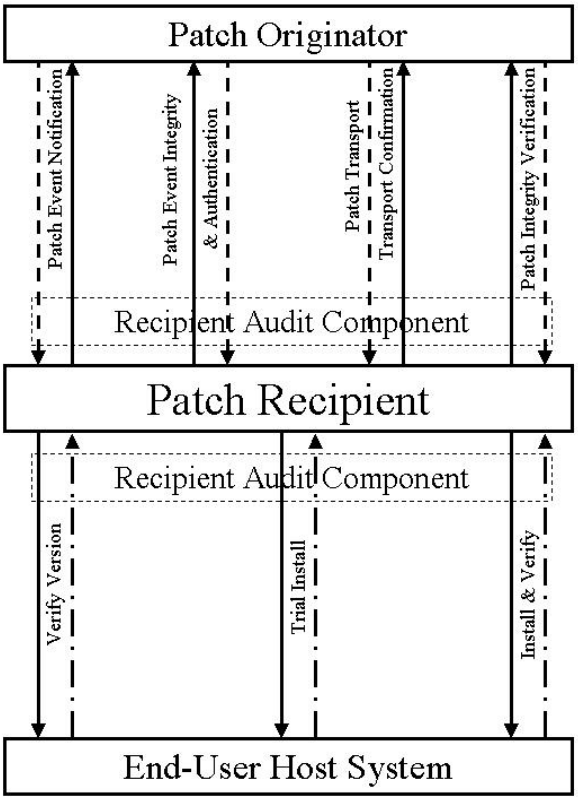

Figure 1: Secure Patch Management System framework

Apart from the technical procedures, listed above, there are other, equally important issues related to the distribution and installation of software patches. These procedures result from the fact that in the majority of cases the producer of the patches and their receivers are separate business entities. Therefore, there is a need to set up their mutual obligations under the law, such as the forms of payment for the patches, installations, liabilities, etc.

This paper will concentrate mostly on the technical issues listed above, but a brief summary of the business problems will be presented at the end.

DISCUSSION OF THE MAJOR ELEMENTS OF THE FRAMEWORK

Patch Event Notification

The first and perhaps the most important component is the patch event notification. This component performs the function of making the user aware that a patch has been developed and is available for deployment in a timely manner. Without such timely notification of the issuance of a patch, a patch becomes functionally useless.

To facilitate the notification process, a set of procedures or the establishment of a notification system is required to ensure timely notice of the existence of a newly issued patch. This system may be procedural, semi-automated, or fully automated. This system may come in the form of a push, a pull, or a combination of the two processes. The evening news broadcasts the availability of the latest patch to correct of security flaw is an example of a push process. Where the user visits the respective software manufacturer to determine the latest news of software updates is an example of a pull process. An e-mail service that an individual subscribes to and may post to notify its entire subscription list of the availability of a new patch is an example of a combination process.

The level of detailed information contained within the event notification should facilitate an informed and corrective response from the end user or system administrator. Upon receipt of a patch issuance, the responsible party wishing to apply the patch should be able to make an assessment as to the significance of the patch if applied to the existing system. Details such as software name and version, any operating system distinctions that the patch was intended to service, the location of the patch, the date of issue, any acquisition or download requirements, and perhaps any additional information required for the decision that the patch is required and necessary to the continued use of the software product.

The patch event notification should be made available to all responsible parties whose functions are maintaining the smooth operation of the host system where the software product resides, i.e. system administrators or end-users.

Patch Event Integrity and Authentication

Once a patch event notification has been received, its integrity and authentication should be seriously considered. This commonly overlooked step is to confirm that the patch event notification actually came from the software manufacturer and has not been modified either intentionally or through error. Data origin authentication allows sources of received data to be verified as claimed and data integrity is used to detect unauthorized changes in data during transmission [Tripunitara, 1998].

When the availability of a patch has been communicated through the public and/or private media networks such as television, it is not uncommon to hear of hoaxes or statements made with bad information. Traditionally, upon becoming aware of a new patch, users visit the manufacturers web site and provide their own verification of the patch's existence. Newsgroups provided a more focused topical dissemination of patch availability but rely upon interested third party contributions. Industry standards of authentication are rarely found in use with newsgroup postings. Another alternative are numerous management services that provide timely notification of issued patches and updates. Companies such as Net Infrastructure [2002], PatchLink [2002], and Mission Critical Linux [2002] are a few examples of the type of management services available. These services are generally provided by e-mail notification but may include remote operational updates.

To prevent unscrupulous individuals from misinforming users, cryptographic methods may be utilized to provide patch event authentication [Stallings, 1999]. Mechanisms such as private and public key exchange, and digital certificates that are processed through professional trusted third parties [VeriSign 2002], have been utilized in this capacity. Hinde [1999] states that “an 'electronic signature' is any substitute for a handwritten signature on and electronically generated document”. A handwritten signature does not necessarily have the same capacity to ensure the origin and integrity of data as a digital signature but can serve to authenticate data in some way.

Patch Version Verification

At this point, a user has received a patch event notification through some dissemination vehicle and has verified the integrity and authenticity of the patch event notification. The user must determine that the information contained in the patch notification is applicable with regards to the software and its current revision level. It may be stated that the version level of any given installed software shall dictate which patch file or files are required.

When a patch event notification occurs, the patch version must be compared to the existing installation software version. This is generally done through the use of manuals, installation diskettes and CD-ROMs, version references listed within the menus of the software, and stored within the operating system registry. In addition to manually referencing the software version, third party applications and diagnostic utilities can provide an accurate version level such as Norton System Works [2002].

In a secure and automated approach to version verification, it would be appropriate and practical to verify that the patch and software versions match at the notification, acquisition, and installation phase of an update. Customized software must also be considered when deciding the applicability of any given patch. Assuming that the patch is indeed required for the continued operation of the intended software product, successful patch transport becomes the next issue.

Patch Transport and Delivery

Historically, a CD-ROM or floppy diskette would be delivered via the postal authority. The medium may be delivered via certified mail or by special courier. In the age of the Internet, electronic delivery occurs by downloading the patch file(s) via the manufacturer's web site using file transfer protocol (FTP). Acquiring the patch may also be completed via newsgroups and e-mail distribution as an attachment file to the communication. But all of these methods bring into question the integrity of the patch file(s). The least likely method for receiving corrupted or tampered patch files is by receiving a CD-ROM or floppy diskette by mail. While it is unlikely that a subversive individual would be able to intercept and modify the files, corruption at the time of disk replication has been a diminishing yet reoccurring problem. In addition, what is to stop some enterprising individual or firm from creating official looking diskettes and mailing them in some professional manner to a system administrator? An uninformed employee may simply perform the update and unwillingly become an accomplice to creating a backdoor pipeline that may be used to exploit a company's computer resources.

Electronic delivery via the Internet provides less assurance of integrity. Web sites and FTP servers can be spoofed, session requests may be diverted, files may be modified prior to download and during download, or

have additional files bound to the patch file(s). What is needed to improve total system security is some form of patch integrity verification. In the case of receiving an update patch via mail service, additional customer information may be provided or integrated into the shipping invoice or incorporated into the original software install. For example, the installation software and updates issued by Great Plains Dynamics [2002] integrates the customer name into the authorization code. For electronic delivery of patches, use of a generated hash function or message digest that is based on the original patch file may provide a means for patch integrity verification. Essentially, a hash function or message digest is a mathematical function that takes a variable-length input string and converts it into a fixed-length binary sequence. It is a uniquely, identifiable digital fingerprint designed in such a way that it is hard to reverse the process by making it difficult to find two strings that would produce the same hash value [Schmidt, 1990].

The manufacturer of the patch could provide access to a pre-generated hash function based on the original patch. This would permit the user to validate that the patch file has not been modified by anyone since its original creation. A user would be able to generate a hash function from the downloaded file and compare it with the original provided by the manufacturer. A match would verify the integrity of the patch.

Patch Trial Installation

The smooth and consistent operation of the software system is one of the primary objectives of the system administrator and the end user. It is very easy to assume that a patch that is provided by the manufacturer will only improve the software's performance. However, experience teaches that no software system operates in a vacuum. Hardware, other software, operating systems, and other available utilities impact the total information processing system environment. Manufacturers of software can only attempt to provide stable products in a continuously evolving, diverse, and customized system environment. As a result, an aspect of securing the integrity of the system in which the software operates is to provide a test bed for the trial installation of a patch. SAP software has a system test area known as the sandbox or test system. SAP recommends that updates and modifications to the system are to be implemented in the sandbox before full deployment [SAP, 2001]. In addition, products such as GoBack 3 by Roxio [2002] provide users the ability to uninstall updates when system integrity needs to be restored.

Patch Deployment

Once the patch installation has been tested and deemed appropriate for the system, patch deployment should be prompt and complete. Regardless of whether the patch is deployed manually, through semi-automated, or fully automated means, access control and authorization must be a considered component of this phase. A distributed system of checks and balances may wish to be employed that allows the responsibilities of patch acquisition and installation to be divided between individuals to ensure that all procedures have been followed prior to installation.

Recipient Auditing of the Components

The last component in a secure patch management system is one that should be fully integrated into all aspects of secure patch management. This component is the audit function. “System audit logs provide information about usage characteristics on a computer system” [Schultz et al, 2001] and are a key component in transmission controls [Aggarwal et al, 1998]. When an event notification occurs a log entry should document the receipt and details of the notification. The resulting authentication of the event notification should also be documented. This will provide a comprehensive list of valid and in valid sources for notification. The version verification that the patch file is applicable or non-applicable should be included in the audit log. Whether a software patch is received by mail or electronic means, the source information and acknowledgment of delivery should also be included in the log. All patch integrity checks should be fully documented and recorded. All trial installations of the patch and the resulting consequences should be recorded along with a system snapshot of available hardware and software components. Once the patch has been deployed, all appropriate documents and manuals should be updated to reflect the current version. Lastly, a distributed storage and access of all audit logs should be implemented to ensure that the documentation trail is not altered or deleted without the highest level of access.

DISCUSSION OF THE SEMI-TECHNICAL COMPONENTS OF THE FRAMEWORK

An important issue, which so far has not been well researched, is the problem of where to store all the available patches. In the case of such companies like Microsoft or IBM, producing and announcing new patches to the existing systems is relatively simple. Large companies such as these may produce patches that even the most unexperienced user would be able to direct their search efforts for upgrades to the well-known software manufacturer.

In our opinion, there is not a straightforward solution to this problem unless big companies control and coordinate all points of entry.

Another issue surfaces and becomes more complicated when dealing with open source systems like Unix or Linux. It is generally left to the user to determine which of the available patches developed to resolve a given issue is the best selection. The question that lies at the core for this type of approach is how does a user know that a patch produced by company A has or has not incorporated new changes that may have been incorporated by a patch developed by company B?

A good secure patch management system must be scalable and accommodate diverse access to patches regardless of the size of manufacturer or the size of the recipient. In the case of a small installation, patch introduction would be relatively simple: usually an owner/user installs basic software components from one source manufacturer, such as Microsoft or Apple Computer. But in the case of company-wide installations, there may be hundreds of different components, sources, and locations. Servicing updates could become an extremely challenging task for even a large organization. This concept suggests the introduction of automation into the process, including the receiving of information about the availability of a patch, its download, and its testing, and installation.

Another open research question is that of organizing libraries containing information about the available patches and the patches themselves. At present we are vigorously researching this particular issue.

The possibility of automating the process of delivery and installation of software patches raises an interesting technical as well as legal issue: to what extent would a user allow the supplier to penetrate their installation? This type of legal issue emerged with the introduction of Windows 95 and continues with Windows XP. Microsoft has included a clause in its accept/decline portion of its licensure agreement stating that it may remotely examine and update the configuration of a user's system. In addition, this issue extends to push technologies such as Java applets and ActiveX modules that promote the acquisition and delivery of a user's system information.

A discussion of the legal aspects will take place in the next section (non-technical components), but there are some additional technical issues hat need to be considered. The first is how to set up the smooth exchange of information between the user of a system and a developer, and to do so such that unauthorised reads by a third party would be prevented. This also needs to be conducted such that confidential information of the accessed party is not disclosed. One possible solution could be to produce a sort of a system passport or system ID that would contain a listing of all components and versions, and be located separately from the real system, i.e. getting access to this file would not compromise the security of the rest of the system. One may expect that such a solution would be eagerly promoted by the software developers but would be strongly opposed by software users, especially those who are using non-registered software.

DISCUSSION OF THE NON TECHNICAL COMPONENTS OF THE FRAMEWORK

Most of the non-technical issues concentrate on the problem of validating electronic documents, i.e. delivery and deployment. In the case of more advanced users or bigger installations, successful deployment of a patch management system would require the automation of the process. The deployment of automating the entirety of the patch management system implies that the parties involved in the process would trust the electronic-only-data for the clearing of their financial and business obligations.

When electronic-only transaction systems are deployed, it is in domestic markets that business obligations may mainly be executed and enforced. Problems begin when there is a need to cross international borders. This can be illustrated in the best way by comparing the transaction policies of dot-com companies. Amazon.com is readily accepts orders coming from outside the USA [Amazon.com 2002], but is reluctant to ship goods outside the USA territories (books excluded).

Another issue, which is not yet solved, is the validity of digital signatures, necessary in electronic commerce for such things as contracts, integrity validation of files, and others. While many countries have introduced related legislation, like USA [Digital Signature, 2000], Ireland or Poland [Zalewski, 2001], others have not. These laws are also not entirely clear as to international jurisdiction.

Yet another quite interesting issue, which has not yet been solved, is of setting the date and location of an electronic transaction. A document signed between two parties, one located in New Zealand and the other in United Kingdom could bear a date/time stamp of 6th of April at 6:00 pm in New Zealand or 5th of April, 6:00 am in UK. Which date/time stamp is valid? This question is not as insignificant as one may suggest. 12 hours difference means that having $1M for 12 hours may produce a gain or losses of around $68 (with 5% interest per annum). The functional aspects of transaction agreements create a series of time constraints that may not so easily be resolved when they transcend international borders.

As a software manufacturer, consider all of these non-technical issues in the context of supplying a software fix, patch or update. If such a firm conducts business in a manner that exposes its customer to additional risks

resulting in damages, hasn't the firm failed to fulfil its fiduciary responsibility? If the technology is available to secure the communication transactions and the firm does not utilize these, is it not in some way responsible for subsequent damages? If a patch is installed and it causes the system integrity to fail, isn't the firm culpable in the resulting crash?

We believe that the very nature of the software industry and electronic transactions across international borders renders many of these issues moot while licensure agreements shield the manufacturer, and practical jurisdiction is rare. It may very well take several hallmark legal cases to void/modify the licensure agreement and an international treaty on jurisdiction to remedy this situation.

CONCLUSIONS

In this paper we have presented a framework for securing patch management, a collection of important issues in patch management, a discussion and series of research questions that are being pursued. These issues and questions can be divided into three, quite separate, domains:

- Technical problems related to identifying the patch and transporting it safely to the destination, and its deployment,
- Confidentiality issues resulting from the need to exchange important information between the supplier of the patches and their end user,
- Inadequacy of the law to handle the exchange of information in the electronic form only, especially in the case of international contacts.

We signalled the important issues within each of these domains. We believe that the future of the efficient utilization of patch management systems lies in the automation of the process. However to be successful, it requires finding solutions to all of the problems mentioned in this paper. At the University of Auckland, we are currently researching these issues, with special emphasis on the technical aspects of secure notification, transporting, and deploying patches, resulting in the development of a refined architecture encompassing all the functions mentioned in this paper.

REFERENCES

[Aggarwal et al], Aggarwal, R., Rezaee, Z., Soni, R., *Internal control considerations for global electronic data interchange*, International Journal of Commerce & Management, Volume 8, No. ¾, 1998.

[Amazon.com, 2002], http://www.amazon.com/help/Payment Methods - We Accept *and* Shipping restrictions

[Common Criteria, 2000], httm:// www.csrc.nist.gov/cc/ccv20/ccv2list.htm.

[CSI/FBI, 2001], *2002 Computer Crime and Security Survey*, http://www.gocsi.com/press/20020407.html.

[Digital signature, 2000] http://www.mbc.com/db30/cgi-bin/pubs/LMZ-E-SIGN.pdf

[Denning, 1999], Denning, D., *Information Warfare and Security*, Addison Wesley, 1999.

[Forch, 1994], Forch, K., *Computer Security Management*, Boyd & Fraser, 1994.

[Great Plains Dynamics, 2002], http://www.greatplains.com/

[Hinde, 1999], Hinde, S., *Step into a secure New World - Compsec '99 Report*, Computers and security, Volume 18, No. 8, 1999.

[Mission Critical Linux, 2002], http://www.missioncriticallinux.com

[Net Infrastructure, 2002], http://www.netinfra.com/patching.htm

[Northon System Works, 2002], http://www.symantec.com/sabu/sysworks/basic/

[Patch Link, 2002], http://www.patchlink.com

[Roxio, 2002], http://www.roxio.com/en/products/datarecoverypc.jhtml

[SAP, 200] SAP R/3 Upgrade Guide. SAP Labs, Inc. 2001. http://wwwtech.saplabs.com/docs/sysadmin/upgrades.pdf

[Schultz et al, 2001], Schultz, E., Proctor, R., Mei-Ching Lien, Gavriel Salvendy, G., *Usability and Security An Appraisal of Usability Issues in Information Security Methods*, Computers & Security, Volume 20, No. 7, 2001.

[Schmidt, 1990], Schmidt, D., *GPERF: A Perfect Hash Function Generator*, Second USENIX C++ Conference Proceedings, April, 1990.

[Stallings], Stallings, W., *Network Security Essentials*, Prentice Hall, 1999

[Tripuntara, 1998], Tripunitara, Spafford, E., *Issues in the incorporation of securities services into a protocol reference model*, Fifth ACM Conference on Computer and Communications Security, 1998.

[Verisign, 2002], http://www.verisign.com

[von Solms, 2996], von Solms, R., *Information Security Management: the Second Generation*, Notes on Information Security Management, IFIP, 1996.

[Zalewski, 2001], Zalewski, T., 2001, *Front wewnetrzny (Internal front)*, Polityka, No 41, 2001