Implementing a Patch Management Process

GIAC Security Essentials Certification Practical Assignment
Version 1.4b, Option 2

By: Scott C. Hull
May 31, 2003

Introduction

The SQL Slammer worm was propagated via the Internet on January 25, 2003. In the first 5 days it caused an approximate $ 1.2 billion in lost productivity, making it the 9th most costly malicious code[1]. In addition to the tangible costs (money, labor, time) realized by organizations exploited by the worm, intangible costs will most certainly be accrued through the loss of public and investor trust in the ability of an organization to secure their systems from attacks. The exploit occurred in spite of the fact that the vulnerability was already known to exist and that Microsoft had released the patch addressing this in July of 2002. That the patch had not been applied should not be a surprise because, according to an article in Network World, Gartner estimates that, "90% of security exploits are carried out through vulnerabilities for which there is a known patch" (Fontana, www.nwfusion.com/news/2003/0505ecora.html). A full 6 months had elapsed before SQL Slammer hit the Internet, and Systems Administrators, to include some of Microsoft's own, had not applied the patch allowing the vulnerability to be exploited. The lack of a patch management plan can be a costly mistake!

The Federal Information Security Management Act, signed into law in 2002,[2] requires Federal Agencies to have patch management processes in place (amongst many other requirements in protecting the Government's computing resources). While it is not the only reason to ensure patching is done, this law does tie an agency's funding to performance and adherence to this law and requires the agency to report on compliance. This case study will describe the approach used in creating and maintaining a patch system for our distributed server environment. This will include the methodology used to prepare the environment, what the environment looked like and actions taken before, during and after implementation, concluding with procedures put in place for ensuring an active patch maintenance plan. It must be noted that this process is not a "one size fits all" but shows our approach in addressing the timely patching of our systems.

System Vulnerabilities

Today's software consists of millions of lines of code and even with extensive testing is released with 5-20 bugs (or flaws) per 1000 lines of code[3]. Windows 2000 was released with roughly 35 million lines of code and based on that, contained an estimated 175,000 to 700,000 bugs! Compare this to the 2001 release of Redhat 7.1 with an estimated 30 million lines of code and a potential 150,000 to 600,000 bugs. Each of these bugs is a possible vector for exploitation.

Keeping up with patching of systems to fix these bugs even in a small environment can be a challenge (Gartner reports that, "IT managers are spending up to 2 hours per day managing patches for their systems" (Fontana, www.nwfusion.com/news/2003/0505ecora.html)). This makes it imperative that a well thought out and documented plan be created to facilitate the patching process and the proper tools needed to help automate this process are chosen. Without this, the ability to maintain the confidentiality, integrity and availability of your computing resources will become near impossible.

Before the process begins, it is important to understand what vulnerabilities are and what is out there to take advantage of them. The definition of systems vulnerability is the absence of some control or countermeasure that help to mitigate the risks to the systems you are protecting. The SQL Slammer worm was not the first worm to take advantage of a known vulnerability where the control (the patch released 6 months earlier by Microsoft) was not put in place by Systems Administrators. In 2001, the Code Red and NIMDA worms wreaked their havoc by taking advantage of multiple vulnerabilities for which patches had been released for[3].

Exploiting the Vulnerabilities

The Internet provides a medium for sharing information and experiences at lightning speed from anywhere in the world. Unfortunately, the same medium is used to share information on how to take advantage of system vulnerabilities. You don't have to be a world-class programmer in order to exploit a system, you just have to know where to look and find out how to do it.

The cycle goes somewhat like this; a system is exploited through some method like a buffer overflow in the operating system software instruction, causing vendors to respond with a patch that nullifies the exploit. The patch is posted on websites along with information on the exploit (how it works, etc) to inform legitimate users that a vulnerability exists and how to eliminate (or mitigate) it. This is where the process breaks down and the point of this case study; the process relies on systems managers to download and apply the patch. The scripts and tools used to exploit the vulnerability are most likely available for download somewhere on the Internet, so when the immediate threat has passed, they are there to be used against the systems of which their respective Systems Administrators failed (or simply ignored) to apply the fix for.

Implementing the Defense

Because of a variety of excuses (too many requirements against too little resources, etc), the patching of internal servers was done with a sort of catch as you can approach; if you had time, go through and patch servers. If not, do it later. This approach meant servers were patched all the way, some of the way or not at all and there was no way to determine for each server which of the three applied. There lacked a systematic approach to ensuring our mission critical resources had the latest patch. The re-occurring theme through this case study is the timely application of patches. To address this issue as well as meet the requirements set forth by the Federal Information Security Management Act (http://csrc.nist.gov/policies/FISMA-final.pdf) , a step by step process was plotted that would leverage the tools we had on hand to bring our servers to the required patching level. In addition, this was made a part of our routine systems administration duties.

Before Implementation

To set the stage of the environment that this process was applied to, the organization has locations all over the Continental United States, Hawaii and Puerto Rico. We have a good number (more than 200) of servers running a variety of Network Operating Systems with the majority being on the Microsoft NT and 2000 platforms.

The organization has gone through various transitions in the area of systems management. Systems were first managed centrally then, as individual business units wanted faster deployment of applications, went to being managed by the business units themselves. The process then reversed itself and the agency's distributed environment went back to being managed by a single team. To add to the mix, we also have a large application development environment, which is done by various groups throughout the organization. We have suffered in the past from a lack of a formal software development life cycle (SDLC) process for distributed applications. This resulted in developers (and even some end users), who had access to a wide variety of software, setting up "servers" and putting out applications. This was done without first ensuring the security of the system and applications were addressed. Even though the application solved a business unit need, the lack of standards in building, deploying and maintaining these systems resulted in an increased risk to system vulnerabilities being exploited.

Because no method existed before going into this, over 200 servers had to be patched to the current level, an extensive number of fixes required installing in order to bring the enterprise up to a current state.

During Implementation

The approach to server management had been to ensure that the servers we managed were available to the users that needed them. As with a lot of IT shops, resources are stretched thin and a growing list of day to day operational requirements stretches them even further. Before implementing the process we now have in place, patching our servers was done in an ad hoc fashion and almost as an after thought. Exacerbating this indifferent approach to patching was the thought that because our network has firewalls, this alleviates us of the need to constantly patch our servers. While this helps mitigate risk of being compromised, this does not provide airtight security. A common statistic used that should concern any systems administrator is that 80-90% of network security incidents come from users inside the network[4]. These users have an advantage over users outside the network as the amount of foot printing, scanning and enumerating is greatly reduced because they already know which systems are available for targeting[5]. Additionally, firewalls should not provide for a lazy approach to patching because external traffic will still be allowed on purpose for business reasons (e.g. access by external customers to the marketing web site). The problem presented then is the lack of a consistent process in ensuring our servers had the latest patches available. Without this, our network had a high level of vulnerability from both internal and external threats.

Before the actual addressing of the patch process itself, a lot of work had gone into creating a more secure foundation in server management. A large part of securing an environment is knowing what is on the network. Because there was no comprehensive listing of the server environment (in part due to the centralized-decentralized-centralized cycle), A database was created that identified every server, the server location, the purpose of the server, the server category and the group the server was used by. This database was then used as part of the standard server build process which had been created to ensure servers were built consistently and addressed all the points of deploying a server (RAID configuration, page file sizing, renaming the admin account, etc).

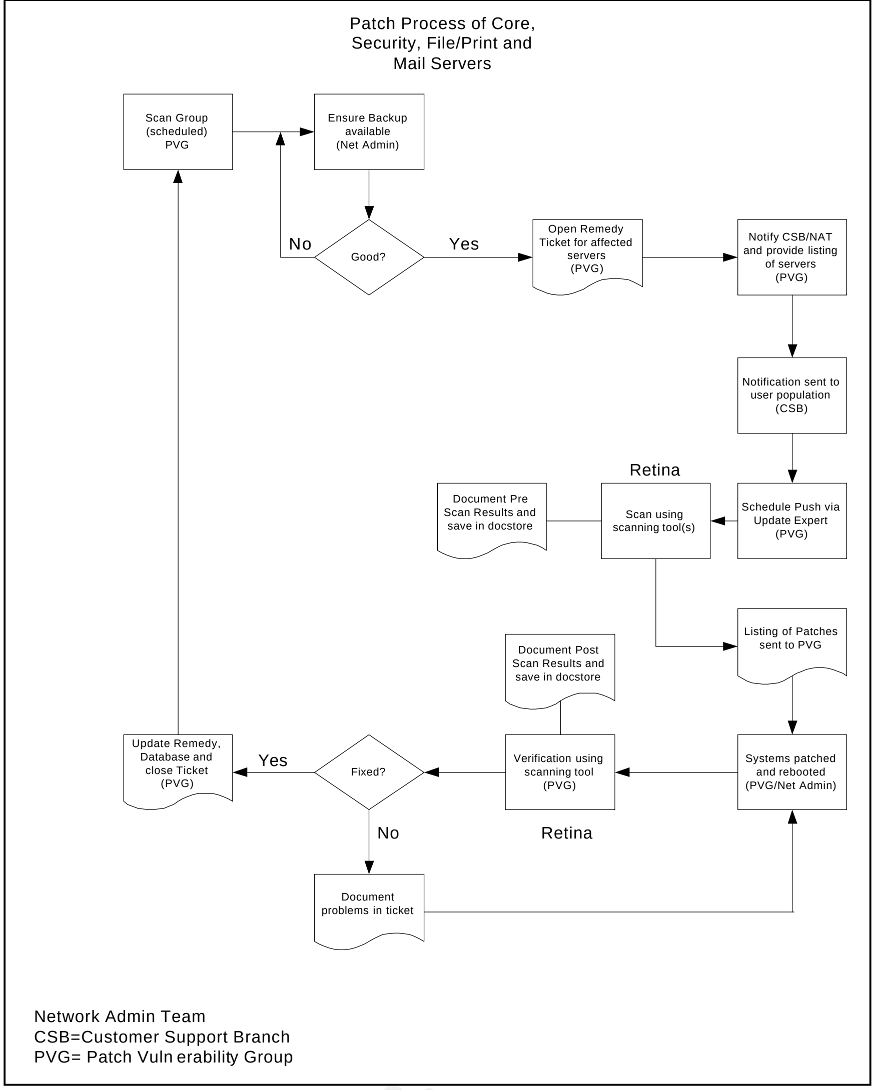
In starting the creation of a formal patch process, I formed a group of team members that could assist in laying out the objective of our patch program, identify the tools available in carrying out the program and the best way to implement the program. This team became the Patch Vulnerability Group as outlined in FIPS Special Publication 800-40[3]. A catalyst to getting a formal process up and running was the Federal Information Security Management Act requirement that could potentially impact our IT funding if not met[2]. The team included myself as lead, the designated security officer for the network and members of the Network Administration team. We also involved members of Customer Support (helpdesk) as they would be our “ambassadors” and ensure the end users were informed of outage times during this process. Our objective was determined to be the creation of a disciplined approach to patch management that was a part of the life cycle of systems management. The process would be implemented in a multi-tiered approach to address our operating environment.

There are a variety of tools available to assist in automating the patching process. This case study is an emphasis on creating a methodical patch management approach, not an endorsement of one tool over the other. We found during the implementation that a combination of tools were best for our environment. I consider the creation of a sound process to be the most important part of creating a patch management system as it lays the foundation that your chosen tools will fit into. The toolkit should include a system scan tool that reliably highlights the systems vulnerabilities before the patching process and whether or not those vulnerabilities exist after the patching process. A tool to do the actual patching is also needed. Because we have a large distributed environment, we required a tool that could centrally download and push the patches to the managed servers. Tools such as windows update (www.windowsupdate.com) and hfnetchk are good for individual or small environments and should be included in the server build process but provide no single view to your environment. The automatic pushing of updates to servers is available with some tools but due to the nature of our environment we felt that this would cause more problems than fix, so we ruled against using this feature. Documentation is considered key to any process and we made this a cornerstone of ours. Documentation ensures that the steps are followed, information is disseminated and outcomes are recorded. Documentation is also used later for such things as formal audits, knowledge bases and creating reports required to management.

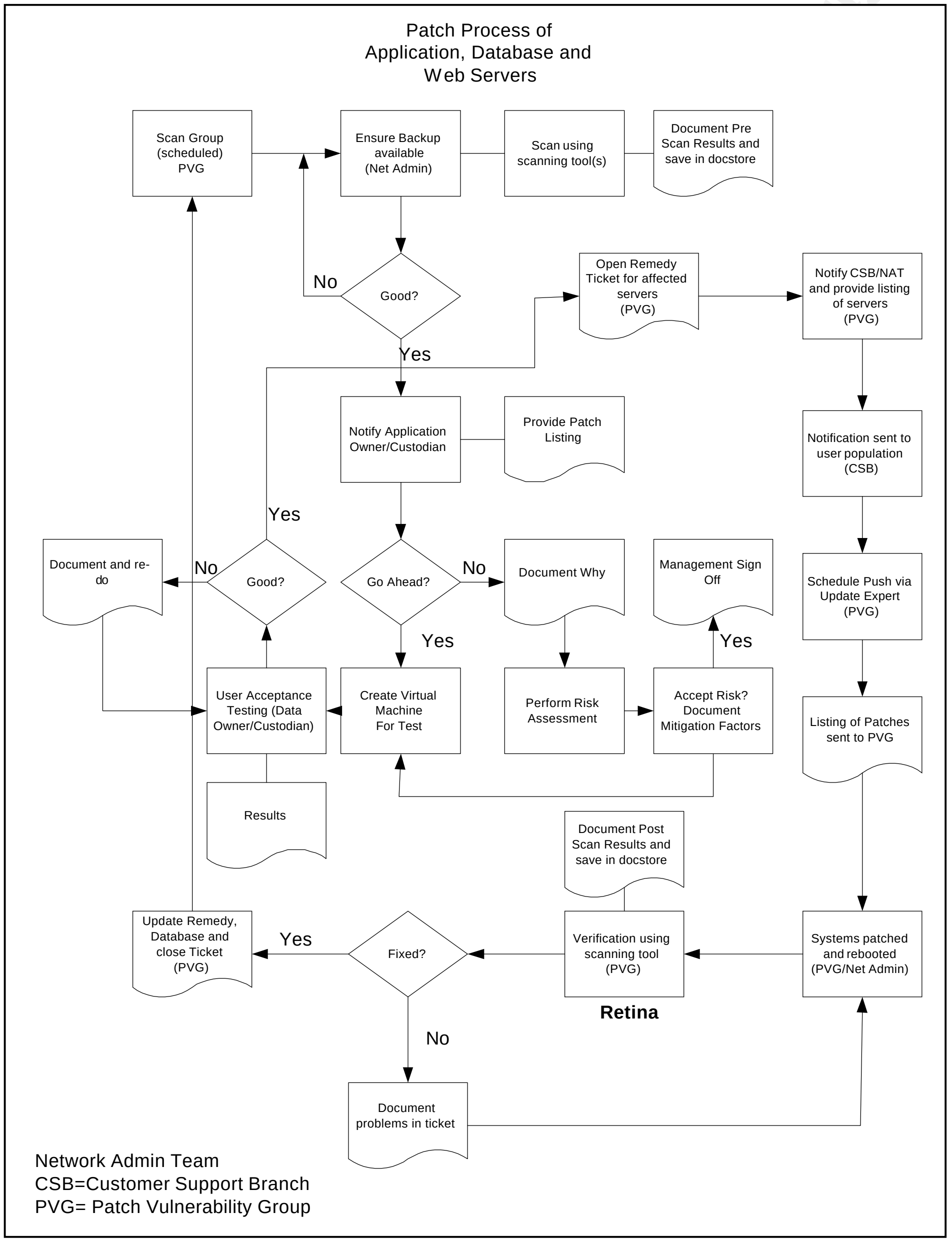
In creating the patch methodology we would use, server categories were created to break the management of the servers up into groups that team members would be responsible for. This helped form how the servers themselves would be patched, as each category of server requires a somewhat different approach to how it was patched. The categories used were Core (Domain Controllers, DNS, WINS, DHCP), File and Print, Mail Servers, Application, Database and Web Servers. It was determined that the Core, Mail File and Print Servers were single use types of servers and the method to patch them would differ from the process in patching the application, database and web servers. The latter group required a good deal of attention to testing the patch and its' impact on the application's availability. The goal was to increase our system security but not at the expense of shutting down our business. I address this fact in flowchart two below by the performance of the risk assessment. We felt that because of the single use purpose of the first group of servers, testing each patch was not required. The impact of a system failure due to a patch application was, in our view, mitigated by redundancies we had built in (e.g. more than one DNS server). Again, this fit our needs but should be evaluated against the environment this methodology would be applied to. The category breakdown also facilitated in increasing our security posture by allowing us to concentrate on smaller groups of servers at a time to bring them up to the current patch level in a short period of time to meet the FISMA reporting requirement timelines.

I took the input from the team and created two process flowcharts. Flowchart one addressed the Core, Security, File/Print and Mail Servers. Flowchart two addressed the Application, Database and Web Servers. In creating the flowcharts, I created a before, during and after flow and identified the group responsible for each step. The following sections are a walk through of the processes.

Flowchart One: Patch Process of Core, Security, File/Print and Mail Servers, the before process begins with ensuring the target system has a recent backup and that the backup is available for restore. The backup administrator ensures the tape is readable and signals the go ahead to the group. The tool we used as the cornerstone of our documentation efforts is our help desk management system, Remedy. A ticket is created which lists the servers to be patched, the patches to be applied and the backups are available. This starts the document trail in this process. For tracking purposes, the ticket number is annotated in the server database for referral, which includes a table for scheduling the server patching, and serves as a quick glance on progress indicators. The next step is to notify our customer support branch. This is important for us in that this branch interacts with the end users on a daily basis. Keeping them informed is critical in ensuring a smooth operating environment and can shorten the time it takes to resolve a problem. The customer support branch is used to also inform the affected population that a server outage will take place and why. If there are issues that impact the scheduled patching, they will arise here. Involving the end user is key. If you involve the user, they get to know why this process is necessary and will help facilitate it by telling you when it can be done. It also gives the end user a feeling of ownership into the process and lends transparency to what we do, "behind the glass wall." Once the schedule has been cleared, the team proceeds with preparing for the patching. A scan is done of the servers that will be patched and the resulting reports stored. The patch process is activated manually and monitored throughout the process to ensure completion. Once the patching is completed, the systems are again scanned with the same tool using the same parameters used in the pre-scan and the results stored. If any vulnerability still exists, the patching process is repeated and re-scanned. Throughout this process, the ticket is continually updated.


Patch Process of Core, Security, File/Print and Mail Servers
Scan Group (scheduled) PVG
Ensure Backup available (Net Admin)
Good?
No
Yes
Open Remedy Ticket for affected servers (PVG)
Notify CSB/NAT and provide listing of servers (PVG)
Notification sent to user population (CSB)
Schedule Push via Update Expert (PVG)
Retina
Scan using scanning tool(s)
Document Pre Scan Results and save in docstore
Listing of Patches sent to PVG
Systems patched and rebooted (PVG/Net Admin)
Verification using scanning tool (PVG)
Retina
Document Post Scan Results and save in docstore
Fixed?
Yes
No
Update Remedy, Database and close Ticket (PVG)
Document problems in ticket
Network Admin Team
CSB=Customer Support Branch
PVG= Patch Vuln erability Group

The second group of servers, the Application, Database and Web Server environment, we needed to involve the application owner a lot more. Since a good deal of the applications involved custom coding which could be impacted by patching, it was imperative that we get buy in as part of our patching process. Towards that end, the process includes a section that notifies the application owner of what patches that will be applied. The owner then has the opportunity to review whether or not any of the changes will adversely affect the application itself. The process allows for the owner to not give the go ahead on patching the system. This will initiate the process of documenting why in as much detail as possible. A risk assessment is then required to be done to assign value to the data on the server, the frequency of the threat and what would happen if the threat were realized. Based on the cost to benefit analysis, the decision is made to whether or not to accept the risk. This latter part requires management sign off as part of ensuring executive level support and that there are no surprises later on. From this part, it should be made clear to the reader that it is important that all parties work together in creating a working, secure environment and not treat each as mutually exclusive requirements. Through the risk assessment, we can work with the owner in bringing the system to an acceptable level of risk and increase the overall security posture, all while maintaining the availability of the application. The owner giving the go ahead to patch does not lift the requirement to test the patch's affects on a server. Through the continued improvement of our application development environment, we have the ability to apply patches to a mirror test environment. For those who cannot afford a separate environment for testing patches, there are software tools that allow the creation of virtual machines. Through this software, you can build multiple "virtual" servers on a single physical server that match the basic configuration of your production environment. The virtual machine is created, software loaded to match the production system and the patch is applied. The application owner then performs a user acceptance test to confirm that the functionality of the application has not been adversely impacted. This is documented and included in the ticket opened next. The remaining steps are the same as in process one.


Patch Process of Application, Database and Web Servers
Scan Group (scheduled) PVG
Ensure Backup available (Net Admin)
Scan using scanning tool(s)
Document Pre Scan Results and save in docstore
Good?
No
Yes
Open Remedy Ticket for affected servers (PVG)
Notify CSB/NAT and provide listing of servers (PVG)
Notify Application Owner/Custodian
Provide Patch Listing
Notification sent to user population (CSB)
Yes
Document and re-do
No
Good?
Go Ahead?
No
Document Why
Management Sign Off
Schedule Push via Update Expert (PVG)
Yes
Yes
User Acceptance Testing (Data Owner/Custodian)
Create Virtual Machine For Test
Perform Risk Assessment
Accept Risk? Document Mitigation Factors
Listing of Patches sent to PVG
Results
Document Post Scan Results and save in docstore
Update Remedy, Database and close Ticket (PVG)
Yes
Fixed?
Verification using scanning tool (PVG)
Systems patched and rebooted (PVG/Net Admin)
Retina
No
Document problems in ticket
Network Admin Team
CSB=Customer Support Branch
PVG= Patch Vulnerability Group

Following these processes, a schedule was developed for each server to be patched. Because the servers had not been consistently patched, the number of vulnerabilities to be addressed was large. Putting in place a methodical, disciplined approach to patching our servers got us moving in the right direction to addressing these vulnerabilities.

Ongoing maintenance

Our first efforts in putting patch management in place addressed filling the gaping hole in our network security. To ensure that this critical part of systems administration would continue to be addressed, a patch operations and maintenance plan was created. The third party patching tool we are using automatically polls for updates and downloads them to our central distribution server. The policy put in place is for every server to be patched at least once every 90 days. The schedule is checked daily by members of the Patch Vulnerability Group and patching executed each night. In addition, a team member monitors security web sites for activities that might require an emergency application of patches to the whole or subset of servers we manage. We also receive automatic notifications of events from the Federal Computer Incident Response Center, (www.FedCirc.gov) which may also require immediate attention outside of our normal maintenance process.

After implementation

After a 4-month period of following this patching methodology, we were able to successfully patch each system or put in place countermeasures that significantly mitigated the potential risks to the system. Patching is also now part of normal operations and not treated as something we will get to. Our system vulnerabilities are being addressed in a routine, methodical manner that has raised the overall security posture of our network.

Conclusion

The process listed above was not an overnight fix. It took a lot of resources and team work in making the process work for our environment and is still an ongoing process to ensure we are hitting our targets. Whether or not you have a large environment or a small one, the following points should be addressed.

Create a formalized plan. The plan should include a standard method to build and deploy servers, documentation procedures and the patch management plan. Make the plan a required part of systems administration and stick to it. The lack of a good foundation creates the ad hoc approach to server administration and increases the likelihood of an exploit from occurring on your servers.

Get management buy-in. Too many times it becomes an inconvenience to go through the patch process, especially if needed files and applications are inaccessible at critical times. By showing that you are involving the end user, you can help mitigate this perceived inconvenience and maintain the security level of your network.

Detail what you have. You will need to know what Operating Systems are out there on your network, where they are located at, what applications run on them and when the last time it was patched. This will augment your ability to react to patching requirements.

Identify, acquire and maintain your toolkit. As pointed out earlier, there are free tools available as well as third party tools that facilitate the process.

Keep in touch with what's out there. There are many sites that maintain current threat alerts. Among them are the Computer Emergency Response Team (www.cert.org) maintained by the Software Engineering Institute at Carnegie Mellon University, and The National Infrastructure Protection Center (www.nipc.gov) maintained by the Federal Bureau of Investigations. Federal agencies can take advantage of the Federal Computer Incident Response Center (www.fedcirc.gov) which is now part of the Department of Homeland Security. FedCIRC also maintains the Patch Authentication and Dissemination Capability (PADC). The PADC allows agency administrators to utilize a database to input their systems into (meeting the know what you have requirement) and provides alerting and reporting services among other things.

Educate the developers and end users. The more they know why patching is important, the more cooperation you will have in carrying out your program.

Maintaining the security of a network environment is a fluid landscape. Start providing some solidity to it by arming yourself with a sound plan.

References

1. Robert Lemos. "Counting the cost of Slammer." CNET News.Com, January 31, 2003.
http://news.com.com/2100-1001-982955.html

2. "Electronic Government"
URL: http://irm.cit.nih.gov/policy/egov.htm

3. Peter Mell, Miles C. Tracy. "Procedures for Handling Security Patches, Recommendations of the National Institute of Standards and Technology". NIST Special Publication 800-40, August 2002.
URL: http://csrc.nist.gov/publications/nistpubs/800-40/sp800-40.pdf

4. Citadel Security Software. “Network Vulnerability Assessment & Remediation”
URL: http://www.citadel.com/Downloads/whitepaperherculesfinal.pdf

5. Stuart McClure, Joel Scambray, George Kurtz. Hacking Exposed: Network Security Secrets & Solutions. Osborne, 1999. 3 – 85.

6. John Fontana. “Ecora boosts patch-management pack”. Network World, 5 May 2003.
URL: www.nwfusion.com/news/2003/0505ecora.html

7. Will Ozier. “Risk Analysis and Assessment”
Harold F. Tipton, Micki Krause. Information Security Handbook, 4th Edition
CRC Press, 2000. 247 – 285.

8. Microsoft. “Microsoft Solution for Securing Windows 2000 Server”. Microsoft, 5 Feb 2003
URL:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/prodtech/windows/secwin2k/default.asp