Global Information Assurance Certification Paper

Bouncing Back From a Blaster Disaster: How One Company Used Free Tools to Build an Effective Patch Management System

John Verseman
GIAC Security Essentials (GSEC) Practical Assignment
Version 1.4c – Option 2
10/21/04

ABSTRACT

This paper tells the story of how SoftCo Corporation, a software development firm, introduced centralized patch management in an environment with few controls on its user community and a decentralized IT management infrastructure – and did it using primarily free tools. There are plenty of resources available that do a great job of describing how to set up things like Software Update Services, configure Automatic Updates and use Baseline Security Analyzer and this paper will touch upon those things. However this paper is really meant to be a supplement to those materials. The two important themes that run through this paper center around how these tools were adapted to function in an environment that is not very security friendly and how the data that was generated by these tools was used to drive results and communicate those results to management.

This paper starts off by describing the security environment in the summer of 2003 and the effects that MS Blaster had on the environment and the culture. In the wake of these viruses, the Corporate Security team was faced with the challenge of ensuring patches were deployed to all systems at SoftCo so that the events like Blaster and Slammer would not happen again. To further complicate issues, Corporate Security was given no budget to fix these problems. They had to rely on free and/or existing tools.

In the wake of these viruses, SoftCo used Active Directory, Software Update Services and Baseline Security Analyzer along with Microsoft Excel and Access to create a patch management program that not only distributed patches to machines, but measured the progress and allowed Corporate Security to find un-managed devices on the network.

It is important to stress that the events and techniques described in this paper are real. However the name SoftCo is obviously a fake, some of the details of SoftCo's business are purposefully vague and where necessary the data used in examples has been altered so as to protect the company's identity.

An Overview of SoftCo's Environment and Culture

SoftCo Corporation is a software development/consulting firm made up of roughly 5000 people. SoftCo develops and implements an enterprise-wide solution for its clients. SoftCo's mission is to transform the way its clients do their business through the use of SoftCo's enterprise software. Thus the consulting side is just as important as the software development side. Approximately 3000 of SoftCo's employees work at SoftCo's world headquarters while the other 2000 of SoftCo's employees are virtual – they work at client sites or from their homes. SoftCo also operates a few remote offices that have minimal staff in them. SoftCo is mostly a Windows shop. Approximately 90% of SoftCo's systems are Windows-based. The rest of the systems are a mixture of AIX and VMS.

As a technology company, SoftCo strives to create "a culture of innovation." Creating a "culture of innovation" means empowering the user community with tools and information to make decisions and explore new ideas. Speed is also a critical part of SoftCo's culture. There is very little tolerance for bureaucracy. That means that individual groups are empowered to build their own infrastructure to support their development efforts. How that plays out it practice is that the user community in general has a lot of freedom on the network.

- All users have full administrative rights to their assigned devices. This decision was made primarily to accommodate the legitimate needs of SoftCo's large developer community to be able to develop and test software on their own machines. However this policy applies to all users.
- IT functions are decentralized. Corporate IT handles HR, Finance and various other apps used internally. Corporate IT also distributes workstations to the user community. Engineering has multiple autonomous IT infrastructure groups to support development and testing.
- There is a strong aversion to patch management from within the company. Many systems are maintained by developers and this creates a problem because the developers' first priority is to get their programs to work. Patches are perceived as extra work that provides little benefits and leads to more problems than it solves due to Microsoft's early troubles distributing high-quality patches.

In the summer of 2003, SoftCo had developed a manual process for managing patches on its corporate Windows servers. This process was developed as a result of the SQL Slammer virus. While it was a step in the right direction, there were still many obvious flaws in the process. First and foremost, this process only covered a small portion of the total systems at the company. These were primarily the Finance and HR Systems, Exchange Systems and file servers. Second this was a manual process. Essentially it was a monthly "patch-party" where all the corporate system administrators would come to work in the middle of the night on a Saturday night and go patch all the systems by going to Windows Update to download and install the patches on each of the servers.

The Blaster Event and its Impact on SoftCo

SoftCo's Corporate Security Team knew that there were a lot of holes in the patch management process at SoftCo – namely that only a small percentage of the systems were getting patched. However the Corporate Security team was still a relatively young and small group and did not have much authority beyond Corporate IT. So in the summer of 2003, the Corporate Security team developed a notification process to distribute security bulletins to system administrators across the company's various IT groups: Corporate as well as the multiple Engineering departments. This process was Microsoft focused. Corporate Security would monitor for new Microsoft Security bulletins, do the research and re-classify the vulnerability as it applied to SoftCo.

Corporate Security created the following vulnerability classification table to provide guidance on how patches should be applied at SoftCo Corporation:

Vulnerability Rating	**Rating Characteristics**	**Recommended Action**
Critical	A virus or worm which exploits the vulnerability is in the wild or a virus outbreak is in progress or imminent. A security breach that exploited the vulnerability has been otherwise identified	Patch all affected system immediately.
High	The vulnerability can be exploited without any user interaction (such as opening and email or going to a hostile web site).	Immediately test the patch and apply to affected systems within on week.
Medium	The vulnerability requires user interaction (such as opening an email or visiting a hostile website) to be exploited, but is otherwise easily exploitable. Likelihood of exploit being used in the wild by a worm or hostile web sites is high.	Certify patch and apply to affected systems within three weeks.
Low	Vulnerability exploit is extremely difficult or unlikely. Exploit of vulnerability has minimal impact on critical systems	Certify patch and apply to affect systems at the next maintenance opportunity.

In July 2003, Microsoft released Security Bulletin MS03-026, detailing a vulnerability in the RPC service that runs on all windows systems. Corporate Security immediately recognized the gravity of the situation and published an alert to all internal IT owners.

******SECURITY ALERT BULLETIN******

Buffer Overrun in RPC Interface could allow code execution (MS03-026)
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp

SEVERITY

HIGH: ***Certify and deploy patch as soon as possible.***

AFFECTED AREAS

Windows NT 4.0
Windows NT 4.0 Terminal Services Edition
Windows 2000
Windows XP
Windows Server 2003

Impact of vulnerability:
Run code of attacker's choice. Any system that can connect to TCP 135 can exploit this vulnerability. No special user privileges are required.

Mitigating Factors:
An attacker has to be able to send packets to TCP Port 135 – this port is blocked by the firewall protecting systems from direct attacks from the internet.

RECOMMENDED ACTION

ALL SYSTEMS RUNNING AFFECTED VERSIONS OF WINDOWS
1) Deploy patch to development and certification environments
2) Certify patch as soon as is possible
3) Apply patch to production systems

DETAILS

For Complete technical details, see the Microsoft Security Bulletin at:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp

******If you feel this bulletin has not been sent to someone who should see it, please forward it******

Once we became aware that this buffer overflow had been successfully exploited, another reminder was sent to system owners indicating the situation was becoming more serious and urging them to patch their systems.

It is also important to note here that when MS03-026 was originally released, a patch for that vulnerability for Service Pack 2 on Windows 2000 was not available. Here is where bad policy by Microsoft and bad policy by SoftCo created a recipe for near disaster. In July 2003 SoftCo was focused on certifying and deploying Windows XP to all end-users. Our current build was Windows 2000 Service Pack 2. Service Pack 3 for Windows 2000 was never even a serious candidate for deployment because traditionally Service Packs had been seen as highly problematic to certify and deploy while providing little benefit. Microsoft, in an effort to keep customers on the most current version of their software did not originally make the patch available for Windows 2000 Service

Pack 2. Thus if customers wanted continued support via patches, they would have not choice but to upgrade.

As a result, the Corporate Security team found itself in a race against the clock. Thanks to the patch process that did have in place we were able patch all our internal infrastructure systems by the time Blaster hit. However certifying and deploying a service pack to over 5000 users, many of who were remote, and with no automated software management system, in less than 30 days proved to be an insurmountable task and Blaster hit before we were able to deploy Service Pack 3 and the patch for Windows 2000. Lastly, Corporate Security had no real way to enforce it's guidelines for installing patches in a timely manner. Thus server owners did not heed Corporate Security's warnings and did not patch their systems.

Microsoft finally did make a patch available for Windows 2000 SP2 on August 12th. But Blaster.A hit on August 11th. Fortunately, the Corporate IT Server infrastructure was patched thanks the process we had in place. As a result, our critical infrastructure systems withstood the virus and our core network services and enterprise systems remained available. But Corporate Security ultimately lost the race to patch the rest of the company. The result was three weeks of scanning patching and re-scanning. We had further issues with remote associates bringing infected computers on to client sites thus infecting our clients. SoftCo lost countless dollars cleaning up after Blaster and the impetus for automated patch management was born.

In the wake of blaster a few critical things happened:

- Management finally understood why patch management was important. They made it a priority and the CIO mandated that all systems must be patched regularly.
- Corporate Security was officially given the responsibility of virus and patch management. Previously, these issues were left to the system administrators and the help desk. While Corporate Security would still need to work with Administrators and the help desk to ensure the work was done, the responsibility for maintaining the infrastructure, ensuring the integrity of our process and holding system owners responsible now lied directly with Corporate Security.

Our primary constraint was that we had no budget for the project. We could devote man-hours, creativity, spare hardware and free tools to the effort, but we could not go out and spend any money.

The Solution: SUS, MBSA, MS Access, Partnerships and Policy

What Blaster made obvious was that SoftCo's highest risk systems were Windows systems. Windows Systems comprised 90% of the IT environment. Further all our virtual users were on Windows and they were using those systems to connect to our clients networks. Lastly Windows, being the dominant

operating system on the market is a very popular target for hackers. Based on these risk factors – especially the damage that can be caused when a SoftCo system infects one of its clients – Corporate Security decided to focus on implementing a Windows based patch management system in order to reduce our highest risk area. We considered including VMS and AIX patch management in this process, but ultimately decided to address it as a separate project because we felt that those systems were at a reduced risk and trying to address those systems would slow down the progress we needed to make on our Windows systems.

As mentioned, our primary constraint was cost. Being a relatively new team, we had no real budget. Our mandate was to fix the problem, but we would be given no dollars to fix it. What we could use was tools and equipment we had on-hand and we could use free tools. Give those cost constraints, our decisions were easy. We would use Microsoft's Software Update Services solution to distribute patches because it was freely available. We would validate the integrity of our environment by using Microsoft's Baseline Security Analyzer because it was freely available. We would use tools that we already had available to us – Active Directory and Microsoft Access - to enforce Automatic Updates settings, deploy the Automatic updates client and analyze our scan data to ensure we were delivering results.

Active Directory

One thing SoftCo did have going for us was that most of our systems were in Active Directory. Those systems not in Active Directory were in a trusted domain and were in the process of being moved into Active Directory. Thus we were able to take advantage of Group Policy to configure and manage how most systems would behave with respect to Software Update Services. For those systems not in Active Directory yet, we would at least be able to get the privileges necessary through the Windows trust relationships to scan those systems to ensure they were patched.

SUS

To deploy SUS we were able to find spare hardware: one server class machine and one workstation class machine, to run Software Update Services. The workstation class machine would be used to deploy patches to certification systems, while the server class machine would be used to deploy patches to production and end user systems.

With respect to certifying end user systems, we created a separate Organizational Unit for our Certification systems in Active Directory. This OU was pointed to the Certification server. The certification process tested the install and configuration process of the automatic updates client, the download and install process for the patches and finally that the patches did not break any applications on our base device image. In order to expedite the certification process, application testing was done at a "high-level." That means that we did

not do full regression testing on all our applications anytime we wanted to release a patch. Rather, we did basic functionality testing of our enterprise applications. This made sure that the patches did not break any functionality used broadly by the user community. It also allowed us to certify patches faster.

Another problem we faced with SUS was that we had a highly mobile workforce. We had roughly 2000 users that worked primarily on the road at client sites. When they did connect back to SoftCo via a VPN, it was usually for file transfer purposes, to check email (which very often was done with Outlook Web Access) or to access an enterprise application for a specific purpose then get out. On average, remote users did not spend much time connected directly to SoftCo's network. We were therefore concerned about whether remote users would actually get the patches once they were released. In order to give our remote users the best chance that they would get updates in a timely manner, we made our SUS server available on the Internet.

In order to reduce risk of system compromise, we took the following security precautions on our production SUS server.

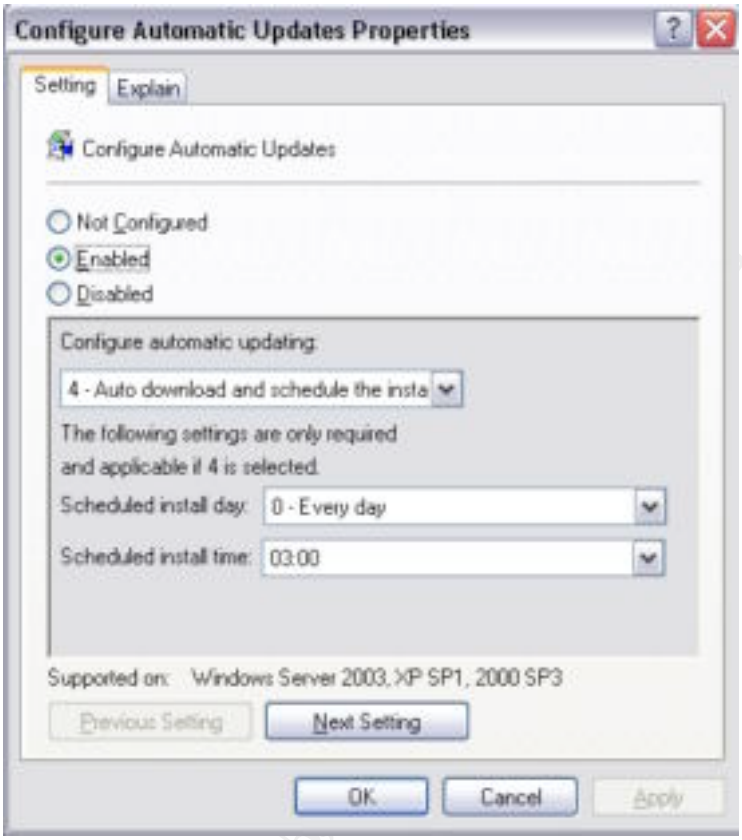
- The firewall was configured only to allow port 80 traffic to the server from the Internet.
- The server was not a member of our production Active Directory domain. It was a stand alone system.
- The administrator account was renamed and a long, complex password was given to the account.
- This was a dedicated system running SUS only and we ran the IIS Lockdown Tool as part of the SUS installation.

With the SUS Server set up and ready to go, we were ready to start configuring the SUS Clients.

The Automatic Updates Client

Workstations: Our automatic updates deployment was happening during the middle of a Windows XP rollout and it was important to ensure both platforms were protected. We decided to split XP and Windows 2000 systems into their own organizational units. Since our base Windows 2000 image was at Service Pack 2, we needed to ensure the automatic updates client was deployed to those systems. Microsoft provided a free msi file to install Automatic Updates on Windows 2000 Service Pack 2 systems. We assigned this file to all systems in the Windows 2000 organizational unit via group policy. This ensured that all systems got the client. Further, we provided instructions to manually install the automatic updates client on end user systems. The help desk was then instructed to verify the client installation on every call they received. This approach helped us find systems that had fallen through the cracks.

Automatic updates clients were configured via group policy as follows:



The client was set up for automatic download and scheduled install at 3:00AM daily. Additionally clients were configured to point towards our production SUS Server and the NoAutoRebootWithLoggedOnUsers key was set to 1. These settings allowed users the option of installing the updates once downloaded and if they declined, the updates would install on the next boot.

Servers: Since SoftCo does not have established maintenance windows, we could not configure automatic updates on servers to automatically install patches. Further we had all our servers in one OU. So when we dealt with the servers we decided to configure automatic updates to automatically download the patches when they became available and notify that the patches were ready to be installed. That meant that the patching process was still manual, but it was at least a bit more streamlined. Instead of going to Windows update and downloading the right patches to be installed, Administrators now just logged on and clicked the automatic updates icon. It provided a good point of compromise to allow us to push patches to our autonomous engineering groups – we would not force an install on those systems, but we would distribute the patches. The system owners then could determine the best time for them to install the patches.

Certification was accomplished on the server side by rolling patches progressively through environments. At the time Automatic Updates was implemented, we had a good communication process between the server team, the development teams and the certification teams. Developer and certification analysts would be notified when systems were patched and they were in turn responsible for testing the stability of the back-end systems and troubleshooting any issues.

Validating the integrity of the environment

Simply deploying SUS was not enough. Indeed, deploying SUS was only the beginning of the project. The most important thing we had to do was to validate to management that we were getting results – and we could just wait till we survived the next worm better than Blaster to know we were protected. Management wanted to be sure the patches were successfully being deployed to the user population. Further we had to hold system owners accountable for patching their systems. That means we had to have a good way to follow up with them to ensure they were applying the patches in a timely manner. This is where MBSA came in. We used MBSA in hfnetcheck mode to run scans of the entire network and output the results to a tab separated file. These results were imported into an Access database and analyzed. What we found is that we could get a wealth of information about our environment through this process.

- We could tell which machines were missing the most patches
- We could tell which patches were not getting installed
- We were able to find rouge Windows machines by analyzing the error messages output by MBSA
- We could compare the systems that MBSA found against the systems covered by our Anti-Virus servers and from that find systems that were not covered by anti-virus.

As mentioned earlier, one thing SoftCo had going for it was that the different IT fiefdoms that existed were not entirely separate. Everyone was either in an OU in Active Directory or in a trusted domain. This allowed Corporate Security, along with CIO support to mandate that all systems had to allow Domain Administrators in the Local Administrator group of every device. This was a critical requirement to simplify and centralize the scanning process. Corporate Security was also given the authority to remove devices from the network that did not comply with our policy.

Configuring the Scans

We configured Microsoft Baseline Security Analyzer in hfnetchk mode to scan by subnet. Our subnet design segregated networks by building and floor. Data Centers also had their own groups of subnets as well as branch offices. Remote access clients also had their own group of subnets.

Following is a sample batch file we used:

```
@echo off

REM This script will run an MBSA Scan for Building 1

sleep 5
net use e: \\server02\reporting

REM First Floor
c:\MBSA\mbsacli -hf -v -t 8 -s 2 -fq c:\Scripts\ignore.txt -nosum -r 10.1.11.1-10.1.12.254 -x c:\MBSA\mssecure.xml -sus "http://susserver.SoftCo.com" -o tab -f e:\TabFiles\Building1_1.tab
Sleep 10

REM Second Floor
c:\MBSA\mbsacli -hf -v -t 8 -s 2 -fq c:\Scripts\ignore.txt -nosum -r 10.1.21.1-10.1.22.254 -x c:\MBSA\mssecure.xml -sus "http:// susserver.SoftCo.com" -o tab -f e:\TabFiles\Building1_2.tab
Sleep 10

REM Third Floor
c:\MBSA\mbsacli -hf -v -t 8 -s 2 -fq c:\Scripts\ignore.txt -nosum -r 10.1.31.1-10.1.32.254 -x c:\MBSA\mssecure.xml -sus "http:// susserver.SoftCo.com" -o tab -f e:\TabFiles\Building1_3.tab
sleep 10

REM Fourth Floor
c:\MBSA\mbsacli -hf -v -t 8 -s 2 -fq c:\Scripts\ignore.txt -nosum -r 10.1.41.1-10.1.42.254 -x c:\MBSA\mssecure.xml -sus "http:// susserver.SoftCo.com" -o tab -f e:\TabFiles\Building1_4.tab
sleep 10

net use e: /del
```

As you can see from the batch file, one scan per floor was performed. Following is a description of the switches that were used:

-hf: Use Baseline Security Analyzer in hfnetchk mode

-v: Display details for Patch Not Found, Warning and Note Messages

-t 8: Limit the number of threads when the scan was executing to 8. This slowed down the scans and reduced the chance of the scans interfering with regular network traffic

-s 2: Supress Note and Warning Messages. We were only concerned with those patches that registered as Not Found.

-fq c:\Scripts\ignore.txt: This was used to create an ignore file for known false positives and patches that were not necessarily a concern.

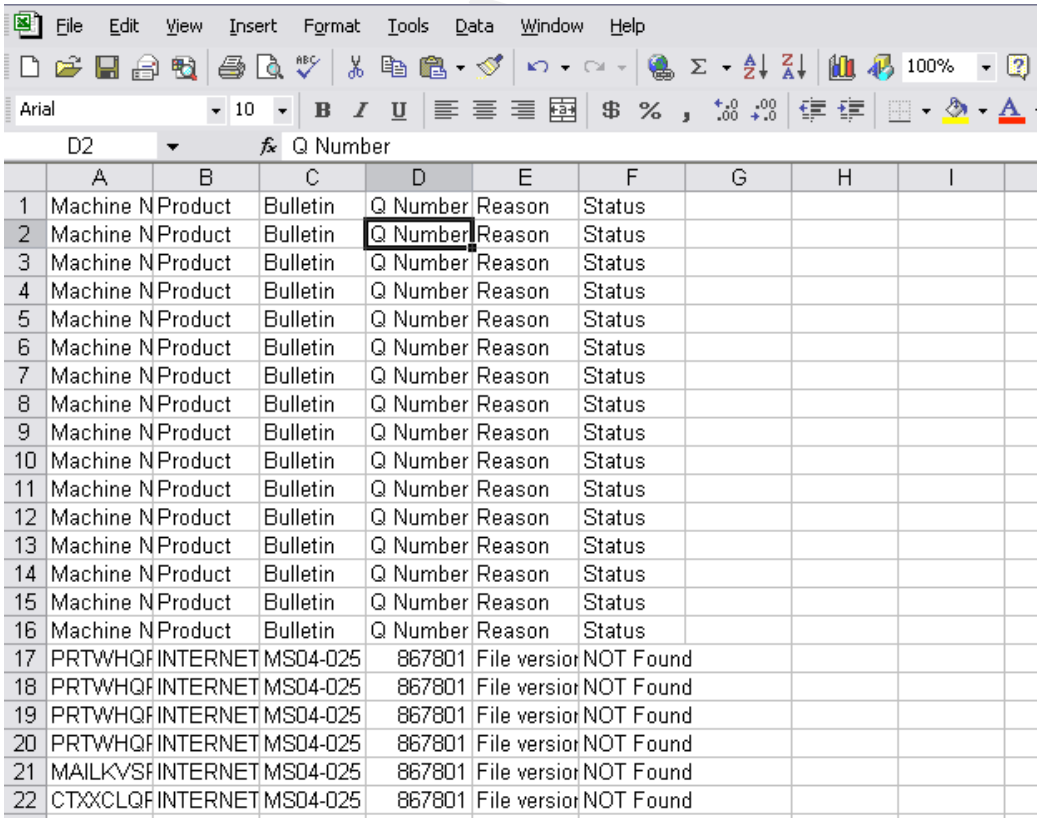
-x c:\MBSA\mssecure.xml: Specify the mssecure.xml file to use

-sus http://susserver.SoftCo.com: Check against updates approved in the Corporate SUS Server. We were only interested in measuring what we had already approved.
-o tab: Output the data in tab separated format
-f <file specification>: Output the data to the specified file

Once the scans had run, we were left with a large number of tab separated files. In order to grab the data from these files, we decided to import them into an Access database for analysis. Since there were a large number of tab files, we made the decision to combine them first before importing them to make the import process go more smoothly. Combining the files was accomplished by using the "type" command and redirecting the output to another text file. For example:

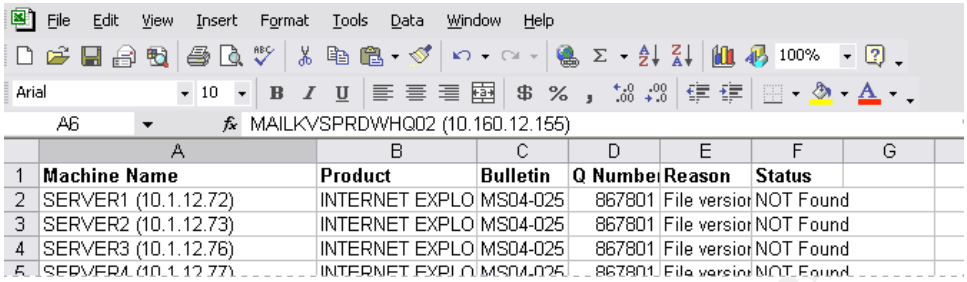
Type subnet1.tab >> end_user_subnets.tab
Type subnet2.tab >> end_user_subnets.tab
And so on…

These commands were added to a batch file so that could be run with one command. Once the compiled file was put together, we opened it up in Excel to make a few modifications prior to importing into access. First, we sorted by KB number descending and removed all the extra column headings.



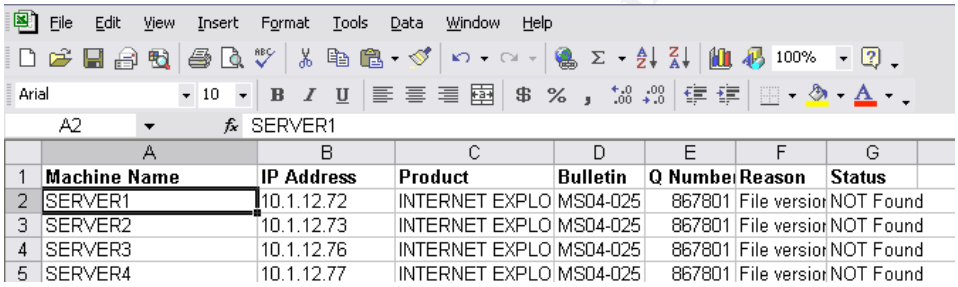
	A	B	C	D	E	F	G	H	I
1	Machine N	Product	Bulletin	Q Number	Reason	Status			
2	Machine N	Product	Bulletin	Q Number	Reason	Status			
3	Machine N	Product	Bulletin	Q Number	Reason	Status			
4	Machine N	Product	Bulletin	Q Number	Reason	Status			
5	Machine N	Product	Bulletin	Q Number	Reason	Status			
6	Machine N	Product	Bulletin	Q Number	Reason	Status			
7	Machine N	Product	Bulletin	Q Number	Reason	Status			
8	Machine N	Product	Bulletin	Q Number	Reason	Status			
9	Machine N	Product	Bulletin	Q Number	Reason	Status			
10	Machine N	Product	Bulletin	Q Number	Reason	Status			
11	Machine N	Product	Bulletin	Q Number	Reason	Status			
12	Machine N	Product	Bulletin	Q Number	Reason	Status			
13	Machine N	Product	Bulletin	Q Number	Reason	Status			
14	Machine N	Product	Bulletin	Q Number	Reason	Status			
15	Machine N	Product	Bulletin	Q Number	Reason	Status			
16	Machine N	Product	Bulletin	Q Number	Reason	Status			
17	PRTWHQF	INTERNET	MS04-025	867801	File versior	NOT Found			
18	PRTWHQF	INTERNET	MS04-025	867801	File versior	NOT Found			
19	PRTWHQF	INTERNET	MS04-025	867801	File versior	NOT Found			
20	PRTWHQF	INTERNET	MS04-025	867801	File versior	NOT Found			
21	MAILKVSF	INTERNET	MS04-025	867801	File versior	NOT Found			
22	CTXXCLQF	INTERNET	MS04-025	867801	File versior	NOT Found			

The next issue we had to deal with was that the Machine Name and IP address were placed in the same column as shown below:



	A	B	C	D	E	F	G
1	**Machine Name**	**Product**	**Bulletin**	**Q Numbe**	**Reason**	**Status**	
2	SERVER1 (10.1.12.72)	INTERNET EXPLO	MS04-025	867801	File versio	NOT Found	
3	SERVER2 (10.1.12.73)	INTERNET EXPLO	MS04-025	867801	File versio	NOT Found	
4	SERVER3 (10.1.12.76)	INTERNET EXPLO	MS04-025	867801	File versio	NOT Found	
5	SERVER4 (10.1.12.77)	INTERNET EXPLO	MS04-025	867801	File versio	NOT Found	

We decided we wanted the ability to run queries by subnet, if necessary, to determine if there was a certain area of the network which needed more attention than other areas. We used Excel's "text-to-columns" functionality to create a new IP Address column, as shown below:



	A	B	C	D	E	F	G
1	**Machine Name**	**IP Address**	**Product**	**Bulletin**	**Q Numbe**	**Reason**	**Status**
2	SERVER1	10.1.12.72	INTERNET EXPLO	MS04-025	867801	File versio	NOT Found
3	SERVER2	10.1.12.73	INTERNET EXPLO	MS04-025	867801	File versio	NOT Found
4	SERVER3	10.1.12.76	INTERNET EXPLO	MS04-025	867801	File versio	NOT Found
5	SERVER4	10.1.12.77	INTERNET EXPLO	MS04-025	867801	File versio	NOT Found

Lastly, we used the find and replace functionality to remove the ")" from the end of each IP Address. We now had our nicely formatted spreadsheet ready to import into Access for further analysis.

Next, the file was imported into an access database so we could run queries to measure our progress. We used the Import function in Access located under File -> Get External Data. Upon initial creation of the table we made sure Access used the first row as column names and we let access choose its own primary key.

Once the file was imported, we queried for the following things:

The total number of missing patches (Vuln Query): This was reported on primarily for its "thud" factor. Telling management that we are missing 20,000+ patches across the organization was a good way to get their attention.

Here is the SQL for the query from Access:

```
SELECT Vuln1.[Machine Name], Vuln1.[IP Address], Vuln1.Status,
Vuln1.Product, Vuln1.Bulletin, Vuln1.[Q Number]
FROM Vuln1
WHERE (((Vuln1.Status)="NOT Found"))
ORDER BY Vuln1.[Machine Name];
```

This was a simple query just to get us any row where the status of the patch was "NOT Found." The total number of rows returned by this query gave us the total number of missing patches at SoftCo and the resulting aforementioned "thud" factor. It was also used to calculate the average number of missing patches per device.

The list of all systems with missing patches, sorted in descending order (VulnPerServer): This was our hit list. We attacked systems on this list from the top down to ensure our most vulnerable systems are getting attention.

Following is the SQL Statement for this query:

SELECT [Vuln Query].[Machine Name], [Vuln Query].[IP Address], Count([Vuln Query].Status) AS CountOfStatus
FROM [Vuln Query]
GROUP BY [Vuln Query].[Machine Name], [Vuln Query].[IP Address]
ORDER BY Count([Vuln Query].Status) DESC;

In order to ensure we were querying only for missing patches, we built this query on the aforementioned Vuln Query. This query grouped all the Status NOT Found messages by machine name then counted them and sorted the data in descending order.

Following is a small sample of the resulting output:

Machine Name	IP Address	CountOfStatus
SERVER43	10.3.31.154	28
SERVER8	10.1.12.211	25
SERVER20	10.2.11.23	25
SERVER17	10.4.41.122	25
SERVER32	10.1.22.33	24
SERVER44	10.6.21.213	23

This query armed us with the necessary information to find our highest risk unpatched systems and get them up to date. This was also the key to holding systems owners accountable. Now when we released a bulletin telling people to patch, we could actually find the people who ignored us and target them. We could also take this information to management should we have to deal with anyone who did not want to cooperate with us. This query proved to be very effective in reduce the number of unpatched and often forgotten systems in our environment.

A list of all missing patches, sorted in descending order: This list enabled us to analyze which patches were not getting deployed and perform secondary checks.

Following is the SQL Statement for this query:

SELECT [Vuln Query].Bulletin, [Vuln Query].[Q Number], Count([Vuln Query].Status) AS CountOfStatus, [Vuln Query].Product

FROM [Vuln Query]
GROUP BY [Vuln Query].Bulletin, [Vuln Query].[Q Number], [Vuln Query].Product
ORDER BY Count([Vuln Query].Status) DESC;

Conceptually, this query is the same as our VulnPerServer query. The only difference is the "GROUP BY" clause. Instead of grouping by machine name and IP address, we now grouped by bulletin, Q(or KB) number and product.

Following is a sample of the output from this query:

Bulletin	Q Number	CountOfStatus	Product
MS04-025	867801	64	INTERNET EXPLORER 6.0 FOR WINDOWS SERVER 2003 GO
MS04-024	839645	52	WINDOWS XP PROFESSIONAL SP1
MS04-022	841873	51	WINDOWS 2000 SERVER SP4
MS04-019	842526	51	WINDOWS 2000 SERVER SP4
MS04-020	841872	51	WINDOWS 2000 SERVER SP4
MS04-024	839645	51	WINDOWS 2000 SERVER SP4
MS04-023	840315	51	WINDOWS 2000 SERVER SP4

This query gave us a good snapshot of patches were not being deployed. This then gave us an opportunity to validate the results to determine if we were seeing false positives in our scans or if there was a problem in patch distribution.

Making use of MBSA Error Messages
During our research, we also learned a key thing about the error messages that Baseline Security Analyzer gives. Following is a list of the most common errors we received on our scans:

- Error: 200 - System not found. Scan not performed.
- Error: 235 - System not found, or NetBIOS ports may be firewalled. Scan not performed.
- Error: 261 - System found but it is not listening on NetBIOS ports. Scan not performed.
- Error: 301 - SystemRoot share access required to scan.
- Error: 451 - Admin rights are required to scan. Scan not performed.
- Error: 553 - Unable to read registry. Please ensure that the remote registry service is running. Scan not performed.

Upon researching these messages, we found that Error's 200 and 235 almost always meant that a system was not using that IP address. We learned that error 261 almost always was a piece of network gear or another non-Windows system. The remaining errors (301, 451 and 553) were almost always a case of the scan not having administrative access. Given this knowledge, we were able to come up with a few more queries that gave us an even better understanding of our environment.

The total number of systems scanned: This query was designed to give us a count of all the Windows based systems our scans found.

Following is the SQL for this query:

SELECT Vuln1.[Machine Name], Vuln1.[IP Address]
FROM Vuln1
WHERE (((Vuln1.Reason) Not Like "Error: 2"))*
GROUP BY Vuln1.[Machine Name], Vuln1.[IP Address];

For this query we looked for any machine name and IP Address pair where the Reason field did not contain an error in the 200 range. Thus we had a count of roughly all the systems on the network that were windows systems, regardless of whether we were able to successfully scan a system.

This number was then used in conjunction with our total missing patches count to give us an approximate number of missing patches per Windows device on the network. The measurement of average number of missing patches per device gave us another indicator of the relative health of our Windows systems with respect to missing patches.

List of Systems with Access Denied Type Errors: This query was used to find those systems that Corporate Security was not able to successfully scan.

Following is the SQL Statement for this query:

SELECT Vuln1.[Machine Name], Vuln1.[IP Address], Vuln1.Reason
FROM Vuln1
WHERE (((Vuln1.Reason) Like "Error: 3" Or (Vuln1.Reason) Like "Error: 4*" Or (Vuln1.Reason) Like "Error: 5*"));*

This query gave us another “hit list” of systems Corporate Security could go out and investigate. Once we found owners of the systems, we could work with them to make sure our scans were able to run on the machines successfully. This query was an outstanding tool in helping us identify rouge systems on the network.

Lastly, we also were able to get a list of systems that reported to our anti-virus servers. This list was imported to another table in access and a query was run to find systems that responded to scans that were not reporting to any anti-virus servers. Using this information, we were able to identify systems that either were not using our corporate virus protection solution or those systems that had a broken anti-virus client.

The Real Test: SASSER

By May of 2004, we had our processes working fairly well. We had enough data to know our patch management systems were working. We were typically able to certify and deploy patches to the organization within approximately two weeks of their release. We had been pro-actively finding and patching our highest risk unpatched systems. Because of how we used scans to identify systems without anti-virus, we increased our coverage and discovered client/server communication issues with our anti-virus system that were subsequently fixed.

On May 1st, 2004, the SASSER worm was released exploiting one of the vulnerabilities described in MS04-011 released in April. Thanks to our vulnerability classification process, we immediately recognized the urgent nature of this vulnerability. We were able to execute on our certification processes and within two weeks we were at 80% compliance and growing. Moreover, our most vulnerable systems, our end users, were 90% patched during that two week period. By the time SASSER hit, we had 5 total infections. It was a long day or two for the security team, but for the rest of the company it was but a tiny blip.

Implementing patch management the way we did effectively reduced our greatest area of risk – virus activity on unpatched systems. We were also able to implement patch management in a way that let system owners retain the control over their systems to which they had become accustomed. The measuring processes we used provided the accountability we needed to ensure system administrators were following the guidelines that we set out for them.

References

Microsoft Corporation. "Software Update Services Deployment White Paper." January 2003. URL: http://www.microsoft.com/windowsserversystem/sus/susdeployment.mspx

Microsoft Corporation. "Microsoft Baseline Security Analyzer (MBSA) 1.2.1 is Available." Rev 20.0. August 18th, 2004 URL: http://support.microsoft.com/default.aspx?kbid=320454

Microsoft Corporation. "Buffer Overflow in RPC Interface Could Allow Code Execution (823980)." V2.0 September 10th 2003. URL: http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx

Microsoft Corporation. "Security Update for Windows (835732)." Version 2.1. August 10th, 2004. URL: http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx

Cruz, Martin. Trend Micro Corporation. "WORM_MSBLAST.A." August 16th, 2003 URL: http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_MSBLAST.A&VSect=T

Sapaden, Bernard. Trend Micro Corporation. "WORM_SASSER.A" May 6th, 2004: URL: http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_SASSER.A&VSect=T