Methodologically Upgrading A Production System

Submission for Track 1, Version 1.4b
by
Resat Otan Ayan

Revision History

Date	Version	Description	Author
4/21/2003	1.0	Document Creation	R. Otan Ayan

Table of Contents

Introduction 4

A Security Hole Is Found! 4

Update Now? 6

The Joys Of Owning A Test Environment 7

Executing Test Scripts In A Test Environment 9

Let's Test 10

Plan Of Attack 11

Game Time 11

Let's Recap 12

References 13

Introduction

System administrators face this challenge on a regular basis. They receive security updates and expected to perform this task seamlessly, without affecting production work on servers, workstations, client systems located locally or remotely.

Patching and upgrading systems has been and still is a revolving task. System administrators receive security alerts, messages, and notifications of available software updates countless times on an irregular basis. We perform the patch, the upgrade, and the fix. A day later, we are informed of another security patch. What happens intermittently, when you receive the notification of a patch and when the patch is successfully applied to your production system? Simply applying the security patch to the production system(s) without executing a set of trial runs of the patch increases the risk of downtime to the user community of your system(s).

Administrative work covers multiple day-to-day operations. As Mandy Andress states, "Proper deployment and periodic updates are necessary to help ensure systems are protected…"[1] This paper attempts to outline the process an administrator should follow after a security patch has been released. Since this process is a consistently repeatable task, a Standard Operating Procedure (SOP) can be revised and enhanced as needed.

A Security Hole Is Found!

"Where do you get your security updates?" asks Mike Fratto writer for Network Computing. There are multiple resources to read and sign up for.

VulnWatch: Vulnerability Disclosure List www.vulnwatch.org
BugTrag Mailing List: www.securityfocus.com/archive/1
NWC Internet Threat Level: www.nwc.com/alertcon/
Security Alert Consensus Newsletter: www.sans.org/nwcnews/
Security Alert Consensus Archives: archives.neohapsis.com

Mike suggests that "most of the notifications that get to your hands could be the result of a heightened hype or what is presented as a high security risk, may not be a risk to your machines at all…"[2] so it is important to identify if a particular security patch should be applied to your systems. Depending on your vendors to alert you of the updates that apply to your systems is one of the first and most important steps you can take.
Below are a few helpful vendor based websites to keep up-to- date:

Novell: http://support.novell.com/filefinder/security/index.html
Microsoft: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/
Netscape: http://wp.netscape.com/security/index.html?cp=brictrpr3
Symantec: http://securityresponse.symantec.com/avcenter/security/Advisories.html
Real Networks: http://service.real.com/help/faq/security/
In addition to proactively listening, reading, and browsing for security-related

[1] Andress, p. 148.
[2] Fratto, p. 23.

information, one may sign up for automated electronic alerts, distributed via email or other means. This is a recommended communication protocol, but it adds an extra step where you are required to authenticate the information validity. One method of applying authentication is through the use of 'signatures'. There are several groups and vendors already utilizing this authentication process with the help of freely available tools, such as PGP (Pretty Good Privacy). A free version of PGP can be obtained from www.pgp.com or www.pgpi.com. PGP software is comprised of multiple sub-toolsets built for the encryption of email messages, disk drives, or network communications. In addition to these available functions, PGP can provide a level of authentication to the validity of an unencrypted message.

When you receive a message signed by an author using PGP, you will notice that the beginning of the message states the following:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
```

The message body will be reflected and the message will end with a few lines identifying that the message has concluded and displaying the message signature:

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.1 (GNU/Linux)

iQA/AwUBPqRZ/lcHflx+x7VOEQIpyACbBqk3zfY4GeuUtcHP/lnX2KWsxJkAoL/m
9nS8Eki4KU/RQo6aT2O2+J7A
=NAfa
-----END PGP SIGNATURE-----
```

In order to authenticate the validity of the message, a few additional steps are required. Assuming you have already obtained a compatible version of PGP, you will need to obtain the author's public key. You will be able to locate most authors published PGP keys on the common PGP servers, or the author will provide them using some other means. For example you can obtain the keys for the SANS (System, Audit, Network, Security) Institute from their website at: http://www.sans.org/key.txt or from the PGP servers at: http://pgp.dtype.org:11371/pks/lookup?op=get&search=0xA1694E46.
In PGP version 8, you can import the PGP keys by simply using "Decrypt & Verify" function on the key text. Once you have obtained and imported the keys to your local key ring you can configure PGP to perform the validation automatically. If the message content is validated appropriately, you will receive a message from PGP stating that the message is intact in its original form:

```
*** PGP SIGNATURE VERIFICATION ***
*** Status:   Good Signature from Invalid Key
*** Alert:    Please verify signer's key before trusting signature.
*** Signer:   Author's Name <author@emailaddress.com> (0xA1694E46)
*** Signed:   4/16/2003 8:23:40 AM
*** Verified: 5/4/2003 11:57:52 AM
*** BEGIN PGP VERIFIED MESSAGE ***
```

A similar message stating that the message has been altered would be provided, if there were a problem with the verification process:

```
*** PGP SIGNATURE VERIFICATION ***
*** Status:   Bad Signature from Invalid Key
*** Alert:    Signature did not verify. Message has been altered.
*** Alert:    Please verify signer's key before trusting signature.
*** Signer:   Author's Name <author@emailaddress.com> (0xA1694E46)
*** Signed:   4/16/2003 8:23:40 AM
*** Verified: 5/4/2003 12:00:16 PM
*** BEGIN PGP VERIFIED MESSAGE ***
```

You will notice that on both of these examples, PGP states that the key utilized is "Invalid". This occurs because you have not signed the Public Keys that are obtained from the author, after importing them to your local key ring. Ensuring that the Public Keys have not been altered, you can sign them and receive complete verification:

```
*** PGP SIGNATURE VERIFICATION ***
*** Status:   Good Signature
*** Signer:   Author's Name <author@emailaddress.com> (0xA1694E46)
*** Signed:   4/16/2003 8:23:40 AM
*** Verified: 5/4/2003 12:04:13 PM
*** BEGIN PGP VERIFIED MESSAGE ***
```

Update Now?

You have received a high priority notification that your system(s) needs a particular security patch. You have performed a validation of the message and obtained the appropriate patch. Should you go ahead and apply it to your production system now?

Before you patch your production system, you will need to ensure that the patch won't negatively affect your systems' functionality. There are a few ways to approach this problem. You can accept the risk and directly apply the patch to your system, or you can minimize the risk of any adverse affects by first testing your new security patch against a test system. We will follow a process where we will first test and then apply the patch, accordingly.

An important point needs identification before proceeding with the testing process. Pre-patch production environment statistics/information need to be captured, such as existing response rates, performance measurements, and event logs. Simply, we need to "base-line" our environment before installing the patch in order to determine any differences (improvements or defects) related to the patch.

Several defect-tracking tools are available, including freeware up to very capable fee-based tracking tools. A useful resource for obtaining a sample list of defect-tracking tools is the following URL: http://www.testingfaqs.org/t-track.htm. You will need to log as much detail about existing functional and technical defects in your production system and also duplicate them in your production and test environments to ensure that they still exist and are reproducible. In addition to tracking functional and technical defects within your production system, you will also need to obtain performance measurements and existing event logs of your system. You can use the "Performance" monitor on a Windows-based system to log the existing utilization. Besides tracking utilization based "counters" within this application, it is also advisable that you track queues for available system components. For example, you will log the utilization of the CPU's by selecting the "% Processor Time" counter from the "Processor" performance object. In addition, select any related queue-based counters such as the "Processor Queue Length" counter from the "System" performance object. The main reason for this selection is that you want to be aware of the average utilization of your systems. Looking at queue lengths will provide you more in-depth information as to how long tasks are waiting to be processed. It would be ideal and necessary to capture these measurements on a "regular" business day. Captures must represent an extended amount of time to provide sufficient data to reflect the day-to-day utilization of your system(s). Do not forget to save existing Event Logs. The most standard logs in a Windows system are "System", "Security", and "Application" logs. There could be additional logs if your computer performs additional tasks, such as being a DNS (Domain Name Server) for your domain. It is advisable that you record and save all logs since the last system reboot. This will provide a good collection of data identifying existing errors and warnings.

The reason why you would want to obtain this information from a production environment is slightly different from why you would want to obtain the same information from your test environment. Data obtained from production will be utilized at a later point to investigate any issues or a question arising after a security patch is applied. Data obtained from the test environment is for knowledge on how the system reacts to the security patch, before it is applied to the production environment. Therefore, you will need to perform a set of tasks against your test environment while capturing the performance and event logs, where this is not needed for your production environment.

Tasks include functional test scripts from your end-users. These tests can be automated and executed against your test environment for a period of time to simulate the production usage. During this simulation, you will need to capture all identified performance and event logs. Several tools can help you in this arena, a sampling of which is available on the following website: http://www.testingfaqs.org/t-load.htm. Keep in mind that you will execute these tests pre- and post-patch installation, without making any modifications to your test environment or your test scripts to ensure that your comparisons represent the same conditions.

The Joys Of Owning A Test Environment

In a perfect world, you could/would duplicate your production environment exactly in a test lab. You would have the same number and type of servers with the same amount of CPU's, memory, disk drives, software versions, etc.. You would also have the same network equipment in your test lab as is used in production. You would have enough resources to simulate the production workload and system utilization. Since we are not talking fiction here, let's realize that your test lab will hardly duplicate your production environment. Therefore, decisions regarding your test environment setup need to be made. You will need to identify priority needs and build your test environment accordingly. "It has been said that if experimentation involves more than seven variables it is an art not a science."[3]

Realism #1: You will need to obtain comparative hardware. The comparison can occur on a node-to-node basis. You may need to ensure that you have as many nodes in your test environment as exist in your production environment. This will ensure that your comparisons will be compartmentalized and you will be able to distinguish particular node(s) affected after a security patch is applied. The purchase of actual hardware for each node that you may want to consider is not necessary. A possible software solution for utilizing multiple nodes in a limited test environment utilizes virtual nodes and can be found at the following URL:

VMWare: http://www.vmware.com

For each test node, you may need to consider a minimum set of functional memory requirements. In the production environment, you may be utilizing large sets of memory. In your test environment, however, you may need only enough memory to support your test scripts. This logic can be applied to the rest of the testing components, with some caveats. If you think you may do well in your test environment with a single CPU vs. a multi CPU, it may be wise to still go with a multi-CPU model to ensure the core system files match the production environment. You will need to obtain adequate harddrive space to handle your test dataset, but the most important issue is that the test drivers must match the system drivers used in your production environment. You may also emulate a mock infrastructure as your "production like" network:

Storm by Shunra: http://www.shunra.com/products/storm/storm_1.php

In addition to the backend system, the client machines must also be built on similar hardware to that used in production. It is easier to obtain client hardware than backend server hardware, although the same special attention (comparable hardware selection) needs to be given if the exact client hardware cannot be used.

Realism #2: After you have obtained your test hardware, software must be configured per the production environment. It is prudent to utilize the same operating system type, version, installers, and previous patches. It will be useful here that you selected a similar hardware choice for your test environment, since you should strive for the same device drives in your test environment as in your production environment. Building the same replication and data access links in your test environment as in your production environment is also an important step and should not be overlooked. Client software should reflect production configurations. If possible, client configuration should imitate

[3] Blakemore

the actual client machines.

You can perform these installations one by one following a written checklist/inventory that you have developed using the production environment. You also have another choice where you can use utilities to speed up this build process. There are several tools available that will speed up your build process, which may require repetition (Did you think you will build your test environment only once?).

Remote Deployment Utility by HP:
http://h18013.www1.hp.com/products/servers/management/rdu/description.html#benefits
Ghost by Symantec: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=3

Realism #3: The dataset(s) utilized in test environments is the most common variable that is overlooked. Data employed in your test environment matters. Your test cycle may provide faulty defects if data are not enough, consistent, or manipulated to fit into your test environment. Data amounts are usually not as important for functional testing. You should have a minimum amount of "production-like" data in order to execute your functional test scripts. It is simple enough to copy production data vs. creating a set of mock data for your test environment. Some of tools that may be utilized in your environment to produce adequate mock data are sample data generators, as follows:

Datatect: http://www.datatect.com/Product/product.htm
Datafactory by Quest: http://www.quest.com/datafactory/

For testing purposes, you may want to use actual production data. Caution: be sure to check your organization's policy on this. You will know if you have placed the correct amount and type of data into the test environment. This becomes apparent when comparing defects found in your test environment to your production environment.

Realism #4: There is one last major item requiring awareness. Does your system have connections and/or dependencies on other systems in your production environment? How are these systems represented in your test environment? Will their functionality be affected with this security patch? Since these connections and dependencies do play a significant role in most environments, you will need to duplicate this environment and set boundaries of your test environment accordingly, by including these interfaces.

Executing Test Scripts In A Test Environment

After you have built your test environment, you need to ensure that it is functioning as expected. You have already obtained a set of test cases and test scripts that you can execute against your test environment. As you know, there is no end to security patch releases and you know that you will be repeating this process over and over again. It may make sense at this point to automate your test scripts. If somebody has already automated the test scripts, you are in luck. If not, you will need to make the choice of executing manual test scripts or automated test scripts. It may make sense to shift this work to a group of test analysts, if there is such a group of professionals existing in your company. There are a vast amount of tools to help you in the automated testing arena. You may be interested in looking at SQA Silk Test, Rational Robot, or some others that are listed at the following site: http://www.testingfaqs.org/t-gui.htm

Let's Test

After all the preparation, now you can apply the obtained security patch(es) to your test environment. Make sure to carefully read the instructions packaged with the patch and document the planned steps you will take based on the following:

> Each software vendor will have unique instructions on how to install their patches (usually the process is very simple). Be careful to follow their instructions, as patches must often be installed in a set sequence for the entire process to work.[4]

While you are applying the security patch, be sure to record any changes to your original documents and plans. In addition, plan to utilize your defect-tracking tool for any ill affects of the patch against the test environment.

Of course, tools that help you apply these patches remotely and to a large amount of systems can be found on the Internet to assist in the installation process. Larry Seltzser suggests "keeping all the Microsoft Windows systems on a large network up to date with Service Packs and security patches is usually a nightmare for administrators."[5] He continues on to discuss a tool called HfNetChkPro, which seems to be a useful alternative to manually installing patches. HfNetChkPro is provided by Shavlik Technologies. You may access a stripped down version of this tool from Microsoft's site.

> Hfnetchk by Microsoft:
> http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=34935A76-0B20-4F91-A0DE-BAAF969CED2B
> Hfnetchkpro by Shavlik: http://www.shavlik.com/

You can obtain further information about these tools by reviewing a paper published by Joe Maloney, for GSEC certification. Mr. Maloney discusses many pros and cons for the use of these tools, but all in all, he makes the point of why we need these types of tools to perform patch upgrades. He states that "…there are too many patches to keep up with, limited budgets and time constraints…"[6] and that is the main reason why these tools are indispensable to your arsenal. Keep in mind that patch management/application into your environments does not mean automation. The big picture is about obtaining time, control, and information[7].

During and after execution of your base-lined test scripts, perform a comparison of your

[4] Computer Security Resource Center

[5] Seltzser

[6] Maloney, p. 1

[7] Dorn, slide 37

event logs, defects, response rates, and performance findings to the originals collected from the test environment. With the results in hand, you should be able to make an informed decision as to how to proceed with the security patch.

Although almost done, there are a few more steps remaining. After you have installed the security patch, it would be wise to check if you can reverse the process, in the event that something goes wrong during execution against the production system(s). These steps may already exist in the recovery process section of your Business Continuity Plan. If you uninstall a problem security patch, make sure you re-execute your test scripts and confirm that the results are identical to those obtained prior to the security patch installation.

Possibly you may not have the option to remove the security patch, but you should still have a process to follow. You may wish to depend on your backups, or even require your vendor to provide an uninstall option. Either way, you should plan for the worst.

Plan Of Attack

A step-by-step plan regarding what happens during a security patch upgrade is a crucial item. You may leave this to your project manager if you have such staff in your organization. You will need to itemize individual tasks and identify an overall time frame allocated for this process. Assign each task to appropriate personnel. You will need to also identify and document each process with individual checkpoints. Do not forget to obtain proper approval of the plan and to notify the user community of planned downtime. All of this planning can easily be documented, as you may have guessed, with a set of tools that are available to you and project managers. The most common planning tool is called Microsoft Project: http://www.microsoft.com/office/project

In your plans, start the actions with the execution of full backups prior to the update. Obtain confirmation of a successful backup via a brief audit.

Game Time

Preparation is the key. You have planned for the worst and you are ready to apply your security patch(es). The best advice to be given here is to stick to your plan. It may not be flawless and you may need to modify it depending on any unplanned roadblocks. But make sure you follow your plan as closely as possible.

When to apply the patch is a point for discussion. Hopefully enough time has passed that any defects within the patch have been located and updated accordingly. A calculated approach as to when to apply a patch have been discussed and studied.

> Patch too early, and one might be applying a broken patch that will actually cripple the system's functionality. Patch too late, and one is at risk from penetration by an attacker exploiting a hole that is publicly known. Balancing

these factors is problematic.[8]

If it is an important security patch, you should proceed and apply the patch. After you have applied the patch, you have only a few remaining steps to perform. You will need to make sure you do not identify any new defects. If you do, you will need to diagnose the new defects. It may have been a modified configuration file that occurred during the patch application, or it could be as simple as a stopped service. After you have compared all base-lined data and executed all your test scripts, rest assured that your system is ready for prime time. You will need to notify the user community as to the availability of the system and document your lessons learned.

Let's Recap

1. Baseline your production environment by obtaining the current defects, configuration details, logs and performance metrics.
2. Create your test environment, building it comparable to your production environment.
3. Baseline your test environment, making sure defects and measurements obtained match to the ones obtained from production environment.
4. Apply your security patch to the test environment.
5. Re-check your baseline that you have obtained in step 3 above.
6. Make sure you can remove the security patch out of the test environment.
7. Re-check again your baseline to the one you have obtained in step 3 above.
8. Identify and document step by step the procedures to apply the security patch to your production environment.
9. Execute your procedures against the production environment.
10. Update your process with identified improvements as needed.

Congratulations, you are now ready to repeat this process again, when you receive next week's security patch! This time, you'll be all the more knowledgeable and efficient.

[8] Beattie, et al. p. 1

References

[1] Andress, Mandy. CISSP Exam Cram. Scottsdale: The Coriolis Group, LLC, 2001. 148.
[2] Fratto, Mike. "Security and Sensationalism." Network Computing March 21, 2003: 23.
[3] Blakemore, Jacqui. "Setting up servos using software." Industrial Technology. June 1998.
URL: http://216.239.37.104/search?q=cache:keypd-badrUC:www.industrialtechnology.co.uk/1998/jun/blakemor.html+setting+up+servos+using+software&hl=en&ie=UTF-8
[4] "Beginners Guide to Updating a Network." Computer Security Resource Center.
URL: http://csrc.ncsl.nist.gov/patches/updatingnetwork.html
[5] Seltzer, Larry. "OS Patch Management." PC Magazine. September 17, 2002.
URL: http://www.pcmag.com/article2/0,4149,480780,00.asp
[6] Maloney, Joe. "Patch Management in a Windows Environment Revisited." GSEC Research Project, Posted Practical. February 9, 2003.
URL: http://www.giac.org/practical/GSEC/Joe_Maloney_GSEC.pdf
[7] Dorn, Susan. "Patch Management: What is it? Why do I need it?" RingMaster Software. May 15, 2002.
URL: http://www.cinap.com/pdfs/WhyAPM.ppt
[8] Beattie, Steve, et al. "Timing the Application of Security Patches for Optimal Uptime." November 2002.
URL: http://wirex.com/%7Ecrispin/time-to-patch-usenix-lisa02.ps.gz