Global Information Assurance Certification Paper

Solaris Patching

Problems, Solutions, and Open Issues

Julie L. Baumler

14 October 2003

Summary

"Patch your systems" is a basic security and reliability prescription, yet the large number of problems reported from security issues that would have been fixed by patches shows that this prescription is not being followed. Most of the recent news making problems caused by missing patches occurred in the Windows environment. However, unpatched or slowly patched systems are a problem in other operating systems as well. This paper takes a high level look at reasons to patch, the extent to which systems in the wild are being patched and the patch management process; specifically under current versions of Sun's Solaris Operating Environment. After looking at the extent of the problem of unpatched systems and the barriers to patching; I look at some of the strategies and tools, both from Sun and the system administration community, that address these problems and point out areas that are in need of further work. Sun's Blueprint for a high level patching strategy is used as a framework for discussing patching issues throughout this paper.[1]

This paper is not intended to solve what the consulting firm Baroudi Bloor calls "The Patching Problem,"[2] but rather to act as a baseline in determining what has been done well and where further work needs to be done. Where a section of the problem is addressed in depth in other works, I refer to those works. Sun and other patching tool authors have documented their own tools well. Barodi Bloor has addressed the business costs of installing or not installing patches.[3] As a result, these items are outside the scope of this paper. Few solutions exist to ease the administrative burden of researching whether available patches are appropriate for a given system or environment and whether they require special handling. As a result, this area is covered in greater depth. As a system administrator, I have written a number of scripts to ease patch management in my environment. Several of these scripts appear to be unique solutions or demonstrate poorly documented solutions for general patching problems. Where appropriate, I have included these scripts in the appendix. Other open issues in the area of Solaris patching involve appropriate verification of patches, both prior to and after installation, and low cost methods to determine which patches are installed across multiple systems.

Why Patch?

Dan Geer argues very convincingly that patching is not really a security issue, but rather a reliability issue and security is simply a subset of reliability.[4] None the less, the necessity of patching is discussed more often in a security context than a system management or administration context. As Sun's Jody Little puts it "systems that are not kept up-to-date are really exposed to known problems in security, in data loss, in availability ... problems that have already been fixed."[5] From a business standpoint, the consulting firm Barodi Bloor points out that "Any required patch not applied means the system is not running optimally."[6] From a purely security standpoint, Larry Rogers of CERT reports "95% of all network intrusions could be avoided by keeping your computer systems up to date with patches."[7]

The State of Patch Installation

Given such a strong impetus, one would expect administrators to patch their systems daily - reality is far different. In congressional testimony, Richard D. Pethia, Director of the Cert Coordination Center, reported that "We have found that, after a vendor releases a security patch, it takes a long time for system operators to fix all the vulnerable computer systems. It can be months or years before the patches are implemented on 90-95 percent of the vulnerable computers."[8] In November 2002 at Sun's large annual conference, Sun Network, they reported that in a typical month Sun receives 800 to 1000 service tickets[A] for problems that are solved by patches that are already available.[9] This number, by nature, doesn't include the customers who are affected by such problems and find the patch either through research or via the practice of doing a general patch update in the face of any problem.[B]

When I talk to other administrators about security problems from unpatched systems, I often hear an assumption that systems are not patched because the systems owners or administrators do not know any better or do not care enough to patch. Is education the answer? In other words, are the unpatched systems home desktops, end-user managed workstations, test systems, and sandboxes for administrators-in-training or are these production servers belonging to security conscious administrators? Eric Rescola took a disciplined look at whether and when administrators were patching or upgrading their systems by measuring the deployment of fixes to a security issue in Apache's mod_ssl on production internet web servers.[10] He felt that this was a good study of what

---

A Based on ticket numbers issued to me, in 2002 Sun received approximately forty-four thousand tickets a month. Tickets are issued for both proactive service issues and actual problems, so this comes to over 2% of problems reported to Sun.

B Since Sun Service often requires installation of the latest "Recommended Patch Cluster" in order to receive help with software issues, installing these patches in the face of any problem is a common tactic.

security-conscious administrators are doing on production servers because this is security software which must be specifically installed. This vulnerability was announced on 30 July 2002, a week later Rescola found that only twenty-three percent of servers had had a fix implemented, two weeks later he found that this had increased to a bit under a third of affected servers. When an exploit was reported in the wild on 13 September, over sixty percent of affected servers remained vulnerable. This triggered another round of fix implementation, implying that one strategy administrators are using is "wait for known exploit before fixing." Rescola found that many administrators wait about a week between the triggering event (either the bug report or the exploit depending on their strategy) and implementing the fix, but almost all systems that were ever fixed were fixed in the first three weeks after the incident. Rescola's paper analyzes the predictors for fix implementation and their implications for making security vulnerability announcements and issuing patches; this is beyond the scope of this paper. For the purpose of this discussion, Rescola's research shows that those who should (and likely do) know better are not implementing needed patches.

Why Don't Administrators Patch?

Since we can see that people are not patching production systems in spite of the fact that they are minimally security conscious and capable of installing and configuring software, the question arises – why not? Two major reasons are time and risk. It takes time to research patches and create a patch plan; it takes time to run a patch plan through an organization's change management process; it takes time (often system downtime) to install patches; and it take time to resolve any issues that may have been caused by the patch or poor patch implementation.

Patching is inherently risky. Adding patches can result in problems like systems where drivers no longer load, system corruption, missing configuration files,[C] and unbootable systems. Although Sun tests patches extensively before releasing them, every environment is unique. Even in the ideal situations where administrators have one or more levels of test and quality assurance systems in which to implement patches, system differences can still cause a failure in the production environment.[11] Sun customers regularly complain that patching breaks third-party applications.[12] My experience is that Sun has greatly improved the reliability of their patches over the last ten years, but in talking to my peers, the “once bitten, twice shy” attitude seems to prevail.

Many administrators try to reduce the risk by installing only the most important patches, unfortunately, today's minor patch may be the prerequisite to tomorrow's key security fix.[13] Patching can transform a system with a known issue that is not noticeable to the system users to one that is not usable, leading

---

C Code samples to avoid or retrieve missing configuration files and a script to fix a particular type of patchadd induced system corruption are discussed later and included in the appendix.

to an attitude of “if it's not broke, don't fix it.” Expectations for system reliability and performance are often low, further encouraging this attitude. Baroudi Bloor's research shows that “A production server that isn't running quite right because a patch or fix hasn't been applied ... is considered business as usual.”[14]

The patching process is also extremely complex. Each Solaris patch includes a README file which may or may not include special precautions or steps needed to install the fixes in the patch. Some patches need to be installed in a specific order or in conjunction with other patches. Some patches can be installed while the system is running and take effect immediately, others need a system reboot to be effective. Still others need to be installed in single user mode, possibly followed by a reboot. Some patches can only be installed if other patches are not installed (or are removed.) Keeping track of all of this gets complicated fast. Rescola points out that poorly written security alerts can make fixing the problem look harder than it is, leading the administrator to decide it is not worth it. I have seen administrators look at the large number of Solaris patches (over 350 each for Solaris 8 and 9[15]), and decide that they can never keep up, so they don't even try.

Sometimes, patches cannot be installed because they implement a change that is not appropriate for a given environment. For instance, under Solaris 8, Sun's patch 108993-18 (and later versions) changes “*LK*” in the /etc/shadow file from simply the default string used to designate that an account cannot be used for login, into a semaphore meaning that an account cannot be used for login nor for running cron jobs, and removes the ability to script the standard Unix no login behavior.[D] When your environment depends on behavior that a patch changes, you cannot install that patch or any patches that depend on it. Patching, particularly operating system patching, often gets delayed due to other change management issues such as implementing a new application or upgrade. Attempting to track down an application or hardware issue can also cause a freeze of all unrelated changes.

Organizational support can also be a problem. In addition to institutionalization of thoughts like “if it's not broke, why fix it”, organizational policies and procedures often discourage patching. Barodi Bloor shares a common, but counter-productive, business attitude:

> The core requirement of a business is to keep the business running and applying a patch can seem like an extraneous task, often viewed as a cost sink rather than an investment in the reliability and profitability of the business.[16]

Proactive maintenance of any type rarely garners recognition and appreciation within an organization. Time spent on patching is time that is not available for projects and other activities that are more visible and often considered real work. An organization's change management process may be too slow or difficult to

---

[D] This commentary is based on my own research into this problem; this issue is discussed on Sagewire as “Solaris changing authentication logic midstream.”

maneuver to allow regular and proactive patching. My personal experience is that often system users or organizational managers view the planned, scheduled downtime for proactive patching just as negatively as they do unplanned, unscheduled downtime due to system problems. There is a joke I have heard a lot in network administration circles:

> Doing it RIGHT the first time gets the job done.
>
> Doing it WRONG fourteen times is job security.

Organizations that reward “heroes” for fixing outages but ignore the administrators who avoided the problems due to proactive patching encourage that attitude to be taken seriously. Some organizations do not even have anyone available to patch systems; they hire consultants to install and implement new systems and do not make any plans for patching and other maintenance.

Some of the reasons given for not patching or not patching promptly, such as running patches through a change management process, are appropriate, though there may be room for improvement. Others, such as past or current bad experiences with patches breaking applications or changing system behavior, are symptoms of technical or administrative problems, either within user organizations or from the vendor, which need to be overcome.

Getting Around the Barriers to Patching

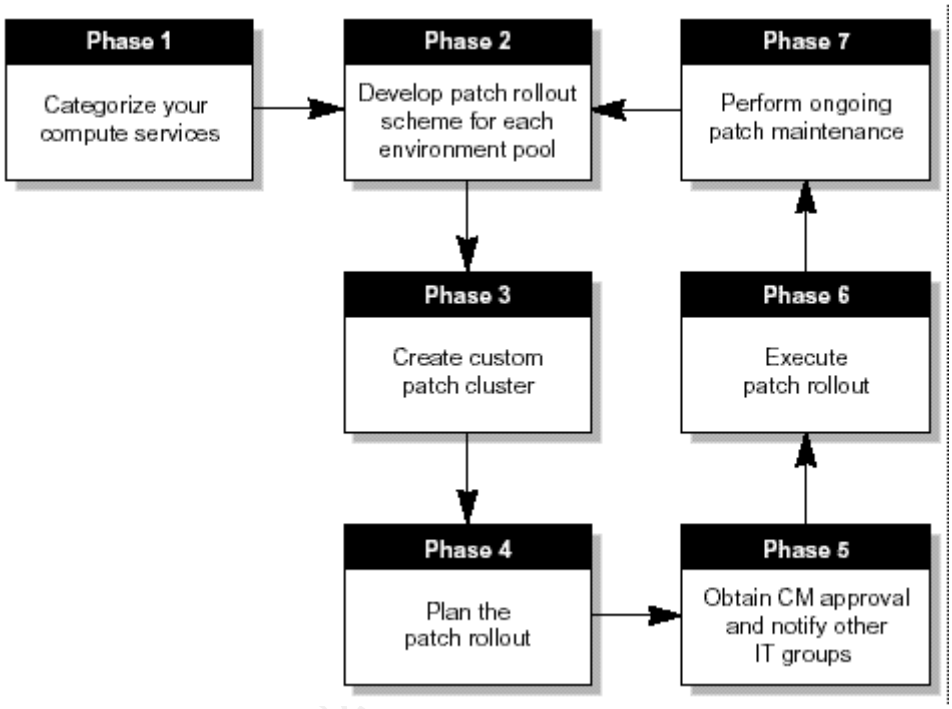
We know that some administrators are installing patches on a timely basis, so the barriers to patching have been at least partially overcome. All of the solutions to patching problems involve applying some combination of intelligence, strategy, and the Unix tradition of scripting anything you are going to need to do more than once. One problem with many of the tools is that no two patching processes or systems are identical and it is difficult to determine which tasks can be fruitfully scripted and which will never be repeated or are so crucial that they should be individually considered and handled on a case-by-case basis. Both Sun and the user community have a number of tools, procedures, and strategies to assist in the patching process. Sun provides a number of patching tools and has provided the user community with a general high-level patching strategy and a recommended low-level patching strategy. There are also a number of community-provided tools that either predate Sun's tools or fill needs not met by them.

Patch Strategies

Sun's High-Level Patch Strategy

In Ramesh Radhakrishnan's January 2003 Blueprint, “A Patch Management Strategy for the Solaris Operating Environment”, Sun Microsystems shares a seven phase high-level patch management strategy.[17] This strategy is intended to be inclusive of the most complex environments and abridged for less complex

environments. This approach makes the strategy particularly useful in providing a framework to discuss the patching process. The strategy is displayed visually in the following diagram:[18]



The strategy diagram gives a high level understanding of the patching process as Sun sees it, but some of the stages deserve some discussion. In phase one, you categorize your systems into sets based on availability requirements. The strategy assumes that within each set there will be some combination of development, quality assurance, test, and production systems for each application or set of applications which can be used for testing patches as well. Phases two through seven are carried out repeatedly for each category of systems. In phase two you develop a lower level patch strategy where you determine your criteria for which patches to install and the speed with which you install them. For instance, in a mission critical environment, you might define a regularly scheduled rollout scheme, a rapid rollout scheme, and an emergency rollout scheme and associated criteria.[19]

Phase three of Sun's strategy, create custom patch cluster, is worthy of in depth consideration. This is an area that is often overlooked and is not well supported by tools and strategies. This phase involves determining which patches to install

on the systems; the order in which they need to be installed to meet patch dependencies; and whether there are any other steps that need to be taken before or after patch installation – such as merging configuration files, turning off services, or rebooting. This stage of patching is time consuming and difficult. Lack of attention to detail at this stage is the source of many patching problems. Yet this effort is often not taken seriously by both system administrators and management. Radhakrishnan suggests using Sun provided tools to come up with a custom cluster using the current jumbo kernel patch, any patches listed in Sun Alerts, and the current Recommended Patches as an initial list of prospective patches to be installed. He mentions the need to read the README files of all the patches to determine the need for their installation and recommends checking your selected patch cluster with your third-party application vendor.[20] Given that the latest Solaris 9 Recommended Patch Cluster contains seventy-three README files with a total of 14,004 lines, this is no small task. My experience is that many problems or potential problems with patches show up on the Sun Managers mailing list prior to or concurrently with their being listed in the README, so checking potential install candidates against the list archives is a good idea as well. Both Radhakrishnan and Baroudi Bloor specify patch cluster creation as a task for senior level system administrators.[21]

Phases four, five, and six of Radhakrishnan's strategy are essentially self-explanatory. Phase four involves developing a schedule for rolling out the different patches onto your systems based on the strategy determined in phase two. Phase five involves running the patches through the organizational change management process. Phase six involves actually installing the patches.

Phase seven highlights the need for ongoing analysis to keep each set of systems in sync. Over time, emergency patch installation and other system specific changes to meet business needs will cause preproduction and production systems to differ. It is necessary to move these systems back into confirmation with each other so that preproduction testing will take place in an environment as much like the production environment as possible.[22]

The Ideal Patching Environment

The ideal patching environment would consist of a relatively small number of identical, or nearly identical, test and production class systems with frequently available, relatively long, downtime windows[E] and a rich set of installed tools and system management packages. Downtime windows can be virtualized by a sufficiently redundant cluster. Sun's Live Upgrade software on systems with additional disks for an operating system copy can also be used to virtualize downtime windows. Since Sun only supports the use of Live Upgrade when the system has the latest Recommended Patch Cluster installed and I have seen Live Upgrade panic systems twice in the last six months; I no longer use or

---

E Frequency and duration of downtime is related, a more frequent downtime window implies few changes to implement and a shorter duration needed.

recommend this strategy to avoid patch related downtime.[F] The rich set of installed tools and packages is necessary because Sun's patching tools, especially their more recent tools, are easy to use but assume that you have a large number of similar systems with specific management tools installed. For instance, to install Sun's latest patch management tool, Patch Manager, under Solaris 9 on my least minimized system, I had to install four or five additional packages. I would be very unlikely to install these packages on even minimally secure systems. These tools are not designed for, and do not work well with, minimized systems where different services are segregated to their own systems. Minimization is both a common security strategy and an excellent way to reduce the number of patches a system requires.

Sun's Low-Level Patching Strategy

Late last year, Sun issued a white paper, "Solaris Patch Management: Recommended Strategies," which delineates a recommended low-level patch strategy[G] and provides further information on using Sun's patch tools.[23] This general strategy involves running the most current version of the OS that your applications and environment allow and upgrading to the latest Solaris version or Maintenance Update as they are issued (quarterly for current OS versions.) Between quarterly upgrades, install key patches mentioned in Sun Alerts and any patches for problems that you may encounter in your environment.[24] Although Sun's patches are well-tested, particularly the Recommended Patch Cluster and the Maintenance Updates, they still recommend testing in your own environment using a two step (test/production) or ideally three step (test, development, and production) testing and validation environment.[25]

Other Low-Level Patching Strategies

Hopefully by this point, we can reject the strategy of not patching. The "wait for a problem or an exploit in the wild and then install fixes" strategy has also been mentioned. This may be an appropriate strategy in some environments and situations, particularly in cases where testing shows that the fix causes problems with other applications or services in your environment. Using this strategy requires keeping abreast of currently available exploits, a topic beyond the scope of this paper.

F This appears to be a case of the right hand not knowing what the left hand is doing, as Sun suggests using Live Upgrade for patching in several best practices documents. When my system panicked, I was told by Sun Support that using Live Upgrade on a system without the latest patches was unsupported. Live Upgrade can still be useful for test systems. Once you have a patched system, Live Upgrade is an excellent upgrade tool.

G Little's 2002 Sun Network presentation also covers this strategy and references the white paper. An updated version of Little's presentation is scheduled to for release at sunsmart.sun.com in November 2003.

Scott Cromar, in his paper "Configuration and Patch Verification on Solaris Systems," recommends a strategy of leaving working production systems alone, only adding patches as a result of troubleshooting a problem or reported security issues.[26] The exception to this is doing a general patch update consisting of the Recommended Cluster and system specific driver and software patches when a security issue or existing system problem is corrected by a kernel, libc, or libthread patch. Cromar's experience using this strategy is that systems will have the latest Recommended Cluster installed on an approximately quarterly basis. Cromar's paper introduces a toolset, check-advisories.sh and host-audit.sh, to audit a system's compliance with a site's desired security patches and non-patch correction of security issues. Of specific note is the fact that Cromar's patch checking tool attempts to determine whether or not a given patch is not installed because the associated software is missing. Cromar points out that by auditing patch list compliance, you avoid problems due to missing the failure of a single critical patch out of the many patches in the Recommended Cluster[H] or forgetting to include a key patch when building a custom cluster. Cromar acknowledges that these tools require significant maintenance of the configuration files which contain the list of patches, the files they change, and the vulnerabilities they address.

My personal patch strategy is to run minimized systems to limit the patches (and vulnerabilities) that apply and then to install any patches that apply to a given system. I try not to install major patches (such as kernel, libc, libthread, key drivers, and volume management software) within the first week of release unless I am having a problem that they address or some event is going to cause a long change freeze for that system. While I would like to install Maintenance Updates; in practice, I find that by the time one is issued, I have installed all of the patches in them. I manage patch status using a script that takes output from Sun's patchdiag tool to compare the patches installed on all of my systems with the currently available patches to determine which patches are installed.[I] I have

---

H Because Sun's Recommended Cluster is cumulative and exhaustive, installing a cluster should result in a large number of patch failures due to the patch already existing on the system or the package not having been installed on the system, separating the wheat from the chaff in the output and log files is often difficult and time consuming. This is discussed further in the solutions section. The script shorten_patch_cluster in the appendix attempts to partially address this issue.

I I have specifically not included this tool in this paper because it predates, and therefore does not support, downloading signed patches and downloading patches via https. In addition, I suspect that the announced changes to Sun's patch database will break this tool. There is some question about ongoing support for patchdiag as well. My tool does not do much that Patch Manager does not, however because it uses patchdiag, it allows patch management for remote systems much less intrusively (for instance, using ssh to get a copy of showrev and pkginfo output, rather than running a SMC agent on the remote host.) If Sun decides to continue support for patchdiag, I plan to release a

recently started testing Sun's Configuration Service Tracker to keep track of installed patches and other system changes over time in an accessible and disciplined manner.[27] My experience is that this strategy of trying to keep current on all patches has led to higher availability and fewer performance issues.

Patching Problems and Solutions Phase-by-Phase

Phases One and Two

Phase one and two of Sun's high level patching strategy involve mainly management issues. They solve general issues with patching reliability and timing; there are no major open issues within these phases.

Phase Three

Many problems and tools apply to phase three, create a custom patch cluster. Sun's Recommended Patch Clusters and Maintenance Updates (MU's) remove the need to create the cluster altogether, instead providing a ready made tested patch cluster. One problem with Recommended Clusters and MU's is the fact that they are cumulative from the original distribution of the Solaris version they apply to. As a result, a lot of time is spent attempting to install patches that have already been installed on your system. If your operating system is at all minimized, they will also waste time attempting (and failing) to patch packages that are not installed; however, the scripts do handle this much more efficiently than they do patch reinstallation attempts. Both Recommended Clusters and MU's come with a script to install the included patches (and possibly packages in the case of MU's), install_cluster and install_mu respectively. The install_cluster script, in particular, is useful for installing your own patch clusters, and my experience is that Sun service will support this use. Install_cluster expects to find the patches it is to install in the current working directory and an optional listing of the order in which to install patches called patch_order. I have a simple perl script, shorten_patch_order, that will compare the output of showrev -p against a given patch list and remove patches that have already been installed from the patch list. This can be very useful in reducing the time needed to install a patch cluster, or in checking that desired patches are installed. Sun provides several tools to help determine which patches might be appropriate for installation on your system – Patch Manager, Patch Check, PatchDiag, and Sun Management Center Reliability Manager. These tools and their advantages and disadvantages are covered in-depth in Sun's “Solaris Patch Management: Recommended Strategies” paper. Sun contract customers may also be eligible to have a Sun engineer prepare reports listing needed patches. These reports are often referred to as Explorer reports, although Explorer is the information gathering tool, not the reporting tool. I find these reports useful in checking that

---

version of this tool that supports these changes to the community in the future.

the system is as current as I believe it to be, but since these are generally only available on a quarterly basis, I find them of limited use for proactive maintenance. These Explorer based reports are the only tool I have found that do a good job of determining firmware currency. A freely available tool (either from Sun or the user community) that would check firmware and prom version against the appropriate data on Sunsolve would be an excellent addition to the currently available patching tools. As mentioned previously, Cromar's check-advisories.sh can be used to build a system appropriate custom patch list, assuming that you have been maintaining a vuln-dev file for your operating environment version.[28] Another free community developed tool, PatchReport,[J] uses the Sun Recommended patch list, or Sun contract customer patch reports, as a baseline to create a custom patch cluster. Like many of the Sun tools, PatchReport will also install patches.[29] From a completely manual standpoint, all of this information is available on Sun's Sunsolve web site. Customers can sign up to receive email notifications of Sun Alerts and new patches.[30] Contract customers can also ask for notifications of other changes – for instance, I am notified each time changes are made in the patch descriptions collection. Checking this information regularly is useful from both a patching and general system management perspective.

Checking Patch READMEs

None of these tools remove the need to check the patch README files for special installation instructions and to ensure that no localized configuration files or operating system files replaced by third-party applications are replaced by the patch. From my viewpoint, there are a few key pieces of information in the patch README. The summary is useful for identifying what the patch does. The list of bugs fixed is often useful in determining how quickly I need to install a patch on my system. I usually install custom clusters and there is always at least one patch in the cluster that requires installation in single user mode and a reboot. As a result, I only really care about rebooting information if I determine that a new patch should be installed in an expedited manner (for instance it fixes a security vulnerability or is related to a problem we are having in our environment.) However, the special installation instructions can be very important and it is important to check that a patch does not change any customized files. David Cornely names the overwriting of local configuration files “the patch cluster problem” because Sun Recommended Clusters are notorious for overwriting configuration files, particularly in /etc and /kernel/drv.[31] Lately I've been experimenting with using the Solaris Fingerprint Database Companion to compare files I am planning to replace with a patch against files issued by Sun. My assumption is that if a system currently works well using a Sun provided configuration file, any replacement file from Sun will work fine as well; so far that appears to be the case. I use a simple script, get_file_names, to pull the names

---

J The Patch Report documentation (Shamblin) states that it is for Solaris 5.X through 8, but since the Solaris 8 patching tools all exist in Solaris 9, I would expect it to work on Solaris 9 as well.

of files that will be replaced from the README files of all the patches in the current directory. I use sfpC.pl,[32] the Solaris Fingerprint Database Companion, to check md5 checksums of these files against files distributed by Sun.[K] I investigate files that are unknown to Sun individually. This method is still imperfect but is better than I was able to do without these tools.

The other important information in the patch README files is the special install instructions. This includes information about special precautions for certain environments and changes that may need to be made to the system. I have written a simple script, patch_readmes, that uses sed to pull the special install instructions out of the README files of patches in the current directory. This script is very simple and the output is very crude. For instance, I don't really need to see “Reboot after install” or “Reconfigure reboot after install” more than once when I'm checking on a patch cluster; nor do I need to see the term “None” when there are no special install instructions; the script does not filter out any of those extraneous messages.

Although the information is in the README file, I often find that the simplest way to determine which software packages are affected by a patch is to run the pkginfo command on the patch directory (pkginfo -d <patchdir>), this can be directly compared with pkginfo output from the system and includes the package description.

Other Phase Three issues

Another very useful tool for building patch clusters would determine all of the nested patch dependencies given a single patch id or list of patch ids. PatchManager implements this behavior by downloading any dependent patches automatically, but PatchManager makes a lot of assumptions about your environment and tool set that are not appropriate for a security conscious environment. It also does not currently handle patches for bundled software.

Depending on the patch clustering method you choose, patch download can be accomplished at any stage between three and six. One problem with downloading any type of software is that you want to ensure that the software you downloaded has not been compromised. Sun has recently started providing signed patches. To the best of my knowledge, currently the only patch tool that checks signatures is Patch Manager. It is possible to manually check patch signatures using Netscape's signtool[33] and there is room for more tools to support signed patches.

Phases Four and Five

Phases four and five involve mainly planning and managerial tasks, and do not have a lot of unsolved problems inherent to them. There are many free and commercial general-purpose planning and change management tools that could

---

K The get_file_names script and details on how to use it with the Solaris Fingerprint Database Companion are included in the appendix.

be fruitfully applied to these phases. There is no compelling reason to use a tool other than your organization's preferred planning and change management tools.

When putting a patch cluster through the change management process, it is often useful to include information gathered from the patch README files in phase three regarding what the patches do and what the risks are. Cromar mentions specifically that "business users of systems are more likely to give permission for patch upgrades when there is a CERT Advisory number that can be referenced"[34] His tool supports tracking these references.

Phase Six

Phase six involves actually installing the patches. Again, Sun has provided several options for installation tools: patchadd, install_cluster, install_mu,[L] and Patch Manager. One of the main complaints about this stage is that it takes a long time. Casper Dik has written an unsupported patch installation program called fastpatch that does everything patchadd does only faster.[35] Patchreport will install patches using patchadd or fastpatch.[36] Sun's Jody Little reports that a significantly faster version of patchadd for Solaris 9 is targeted for release in update five.[37] Sun could improve patch installation by building a patching process that allows patches to safely be installed in multiuser mode (and ideally without a reboot.) AIX has had this feature for years and it makes patching much simpler to schedule. Little reports that Sun is working on this.[38] Another change I would like to see is reasonable error messages from the patch tools when /usr or another system filesystem is mounted read-only, or even an option to remount read-write.

Resolving Patch Problems

Part of the task of installing patches is resolving any problems with, or caused by, the patching process. One of the reasons I like to use install_cluster to install my custom patch clusters is that it creates a log file with the usual patchadd output and sends a summary to standard output. I find it useful to use the tee command to also save the summary to a file. The most common problems I see with patch installation are missing prerequisite patches and patches that cannot install because one or all of the packages that it references are not installed. The output from patchadd lists missing prerequisites, although it does not list any nested prerequisites. I have discovered that if I run the command "pkginfo -d <patch>", I can usually determine whether or not I need to be concerned about a patch not installing due to missing prerequisites. This is worth checking because occasionally, I discover that a key patch will not install because the fix requires an OS package that has not been installed. All of this could, and should, be

---

L The install_cluster and install_mu scripts actually call patchadd. Install_cluster calls patchadd for each individual patch to be installed, so that if it is interrupted, only a single patch needs to be backed out, rather than the entire cluster, this is both safer and significantly more time consuming.

scripted. There is a lot more that can be done to make checking patch cluster installation easier; for instance, checking that all corequisite patches have installed correctly should be automated.

The other type of patch problems, problems caused by patches, are less straightforward. Resolving this type of problem is much simpler if you have installed patches with the backout option. If you know which patch is causing the problem, you can simply use patchrm and backout that patch. Figuring out which patch is the problem is often much harder – that's why you hope to find these problems in your test, rather than production systems. If you have overwritten a customized configuration file with a patch, you can backout the patch, or you also have the option of just retrieving the configuration file out of the patch.[M]

One problem I've encountered recently is the patching process can damage or delete the pkginfo file for one or more packages. Without these files the system is unpatchable and the only Sun supported solution is to upgrade the operating system. I've included the script that I wrote to recreate these files, fix_pkg, in the appendix.

Phase Seven

Phase seven, perform ongoing patch maintenance, involves making sure that all of your systems have the patch levels you expect them to and that they remain in sync. Cromar's tools allow for auditing to ensure that systems have all the desired patches. Sun provides a diagnostic tool, PCMP – the Patch Comparison Tool, that allows you to compare the patch differences between two systems.[39] As mentioned earlier, I have a tool that creates a web page to display the patch status of multiple systems; unfortunately it depends on services from Sun that are currently in flux or designated end of life. Radhakrishnan discusses the i-status tool, available on a fee basis from Sun Professional Services. i-status displays and compares system configurations.[40] A similar free tool would be very useful.

Conclusion

Patch management is hard, but crucial work. In the Solaris Operating Environment there are a number of tools and strategies, from Sun and the user community, to make this more manageable. Unfortunately, there is no one patch management tool that handles all of the security related issues of patch management well – for instance you need to choose between automatically checking patch signatures and having a tool that functions in a minimized environment. In addition, there are a number of areas where potentially useful tools are missing, particularly in the areas of determining which patches to install and checking that patches have installed correctly.

M Retrieving saved patch files into /tmp so that a configuration file can be retrieved is covered in the appendix.

1 Ramesh Radhakrishnan, “A Patch Management Strategy for the Solaris Operating Environment,” Sun Blueprints Online – January 2003, 20 May 2003 <http://www.sun.com/solutions/blueprints/0103/817-1115.pdf>.

2 Baroudi Bloor, The Patching Problem: It's Costing Your Business Dollars, 22 May 2003, <http://www.baroudi.com/pdfs/patch.pdf>.

3 Baroudi Bloor.

4 Dan Geer, “Patch Work” ;login: August 2003: 27-28.

5 Jody Little and Jacqueline Roundy, “Patch Management Best Practices” SunNetwork. November 2002, 23 May 2003 <http://sunsmart.sun.com>, slide 8.

6 Baroudi Bloor, p. 3.

7 Larry Rogers, “Improving Security – Larry Rogers – Applying Patches”, 1 April 2001, 12 September 2003 <http://www.cert.org/homeusers/apply_patches.html>.

8 Richard Pethia, “Viruses and Worms: What Can We Do About Them?” Testimony of Richard D. Pethia; Director, CERT® Coordination Center; Software Engineering Institute; Carnegie Mellon University; Pittsburgh, PA 15213 Before the House Committee on Government Reform: Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census : Hearing on Worm and Virus Defense: How Can We Protect the Nation's Computers From These Threats? September 10, 2003, 12 September 2003 <http://www.cert.org/congressional_testimony/Pethia-

Testimony-9-10-2003>.

9 Little, slide 8.

10 Eric Rescorla. "Security holes... Who cares?" 21 August 2003 <http://www.rtfm.com/upgrade.pdf>.

11 Baroudi Bloor, p. 6.

12 Little, slide 5.

13 Baroudi Bloor, p. 7.

14 Baroudi Bloor, p. 2.

15 Sun Microsystems, "patchdiag.xref," 12 October 2003, <http://sunsolve.sun.com/pub-cgi/patchDownload.pl?target=patchdiag.xref&method=H>.

16 Baroudi Bloor, p. 4.

17 Radhakrishnan.

18 Radhakrishnan, p. 3.

19 Radhakrishnan, pp. 8-9.

20 Radhakrishnan, pp. 13-14.

21 Radhakrishnan, p. 13. Baroudi Bloor, p. 4.

22 Radhakrishnan, p. 20.

23 Sun Microsystems, "Solaris Patch Management: Recommended Strategies: A White Paper," 2 October 2002, 22 August 2003 <http://www.sun.com/service/support/sw_only/pmstrategies10.02.pdf>.

24 Sun Microsystems, "Solaris Patch Management", p. 1.

25 Sun Microsystems, "Solaris Patch Management", p. 26.

26 Scott Cromar, “Configuration and Patch Verification on Solaris Systems,” 23 February 2003, 22 August 2003 <http://www.sans.org/rr/paper.php?id=921>.

27 Sun Microsystems, “Configuration and Service Tracker (CST),” 12 October 2003 <http://www.sun.com/service/support/cst/>.

28 Scott Cromar.

29 Stokely Consulting, “Solaris Patch Management,” Unix SysAdmin Resources: FAQs, Patches, and Other Info (Sun 3/3), 12 Sep. 2003 <http://www.stokely.com/unix.sysadm.resources/faqs3.sun.html#solaris.patch.mgmt>

30 Sun Microsystems. “Sun Newsletters,” 13 October 2003 <http://www.sun.com/newsletters>.

31 Cornely, David. “Re: Tape drives keep going down upon use on Solaris 8 master server after upgrade.” Online Posting. 28 August 2003. Veritas-bu. 28 August 2003 <http://mailman.eng.auburn.edu/mailman/listinfo/veritas-bu>.

32 sfpC: Solaris Fingerprint Database Companion, vers. 1.2, 30 May 2003 <http://www.sun.com/solutions/blueprints/tools/index.html>.

33 Sun Microsystems, Signed Patches Administration Guide for PatchPro 2.1, March 2003, 10 October 2003 <http://sunsolve.sun.com/patches/spag.pdf>.

34 Scott Cromar.

35 Casper Dik, Fastpatch, 17 September 2003, <ftp://ftp.wins.uva.nl/pub/solaris/auto-install/>.

36 Stokely Consulting.

37 Little, slide 18.

38 Little, slide 27.

39 Sun Developer Connection – The Patch Comparison Utility (PCMP), 12 September 2003, <http://solaris.java.sun.com/tools/pcmp/pcmp.html>.

40 Radhakrishnan, p. 21.

Appendix – Software Tools for Patch Management

Shorten_patch_order – remove installed patches from a list of patches.

```
#!/opt/bin/perl -ni.dist
#################################################################
#
# PROGRAM:              shorten_patch_order
# USAGE:                shorten_patch_order var=val <patch_order file>
#                       Potential vars are:
#                               showrev=<showrev -p output file>
#                               verbose=1
# PURPOSE:              Deletes already installed patches from a
#                       patch order file such as those included with
#                       SUN patch clusters.  This can significantly
#                       improve the speed of patch installation.
#                       The original patch_order file is saved as
#                       patch_order.dist.
# NOTE:                 There is a lot more testing that could
#                       potentially be done to determine whether
#                       or not a given patch should be installed
#                       or not; patchadd does an excellent job
#                       of quickly and efficiently testing such
#                       situations; it is not efficient at testing
#                       whether the patch is already installed -
#                       hence this program.
# AUTHOR:    jlb 20001031
#################################################################
# split into patches and revision number
($patch, $rev) = split("-");

# test to see if revision number matches; occasionally, there
# is a need to back rev a patch; I don't think patchadd handles
# the process automatically, but I believe that in some cases
# it warns if a cancelled higher rev patch is installed
```

```
if ( ! defined( $installed{$patch} ) || ( $installed{$patch} != $rev ) )
{
        # only save patch names if we need to try to install them
        print $_;
        chomp($_);
}

BEGIN{

        # get variable = value from command line
        eval "\$$1=\$2" while $ARGV[0] =~ /^(\w+)=(.*)/ && shift;

        $showrev = "" if ( ! defined $showrev );


        # read in current patches so we know what they are
        &get_current_patches($showrev);

sub get_current_patches {

        my($showrev_file) = @_;

        # determine where our patch data is coming from
        if ( $showrev_file =~ /^$/ )
        {
                # empty showrev file, run showrev -p
                $showrev = "showrev -p |"
        }
        else
        {
                # we gave a filename that contains showrev -p output
                # open it
                $showrev = "<". $showrev;
        }
```

```
		#print "Showrev data comes from $showrev\n" if $verbose;
		# showrev -p gives data in the form:
		### Patch: 106541-10 Obsoletes: 106832-03, [...]

		my($patch, $patch_id, $rev, $junk);

		open( SHOWREV, $showrev) || die "Can't open patch list";
		while (<SHOWREV>)
		{
			($junk, $patch_id, $junk) = split(" ");
			($patch, $rev) = split("-", $patch_id);
			#print STDERR "$patch is at version $rev\n" if $verbose;
			print STDERR "$patch_id\n" if $verbose;

			#$installed{$patch}=$rev;
			# set to rev unless we have multiple versions of this
			# patch installed, then make sure we set this to the
			# latest version
			if ( $installed{$patch} )
			{
print STDERR "PATCH $patch SEEN $installed{$patch} NOW $rev " if $verbose;
				if ( ($installed{$patch} + 0) < ($rev + 0) )
				{
					$installed{$patch}=$rev;
				}
print STDERR "SET TO $installed{$patch}\n" if $verbose;
			}
			else
			{
				$installed{$patch}=$rev;
			}
			#$installed{$patch}= $installed{$patch} ? ($installed{$patch} > $rev ?
$installed{$patch} : $rev) : $rev ;
		}
		close SHOWREV;
}
```

}

get_file_names – read names of files to be overwritten out of patch README's

```
#!/bin/sh
# get_file_names - get the files changed out of the README's for the patches
#                in the current directory
# 200306 jlb
for patch in 1*-[0-9][0-9]
do
                nawk 'BEGIN{infiles=0}
                    /Files included with this patch:/{infiles=1; next}
                    /Problem Description:/{infiles=0}
                    { if (infiles == 1 && $0 ) {print $0}}
                   ' ./${patch}/README.${patch}
done
```

Using get_file_names to find custom files that will be replaced by patches

I only install get_file_names, sfpC.pl and the perl modules it requries on my patch staging server. I usually use ssh to get the md5 signatures off the remote system using the following command string:

```
sh get_file_names | ssh <hostname> sudo xargs -i md5 {} \
                | sfpC.pl -   | grep "0 match(es)"
```

It is necessary to already be an authenticated user of sudo when using sudo with ssh and xargs in this manner (you can leave out the sudo at the risk of missing files that you do not have permission to read.) It is also possible to put data in files instead of using a pipeline. This only identifies files that are unknown to Sun, it does not determine whether you are using the file from the current version/patch level of the operating system, although the database does provide that information.

patch_readmes – pull the special install instructions out of patch READMEs

```
#!/bin/sh
# patch_readmes
# gets special install instructions from all patch readmes in the current
# directory
# jlb 20001229
for i in `ls -d [0-9]*`
do
        /usr/ucb/echo -n "Patch $i : "
        sed -n '/Special Install Instructions:/,$ p' $i/README.$i
        echo ""
done
```

Manually retrieving individual files from patch backout files

When a configuration file is overwritten by a patch, generally, you do not wish to actually back out the patch, you merely want to retrieve your configuration file.

The first step is to identify the patch that changed the file. Using grep against the patch README files is usually the most efficient way to do this. You also need the name of the package that includes the file. Greping for the filename in /var/sadm/install/contents will provide this. Next you want to copy the saved patch archive into files in /tmp using the following commands:

```
zcat /var/sadm/pkg/<PACKAGE_NAME>/save/<PATCH_NUMBER>/undo.Z \
                > /tmp/datastream
mkdir /tmp/<PACKAGE_NAME>
pkgtrans /tmp/datastream /tmp/<PACKAGE_NAME>
```

The missing configuration file should be somewhere in /tmp/<PACKAGE_NAME>; most likely /tmp/<PACKAGE_NAME>/reloc/<full_path_name_to_file>.

fix_pkg – retrieving missing pkginfo files after patching

```
#!/bin/sh

#  * WARNING * WARNING * WARNING * WARNING * WARNING * WARNING *
#
#  THIS PROGRAM IS COMPLETELY UNSUPPORTED AND MAY BREAK YOUR SYSTEM
#
#  * WARNING * WARNING * WARNING * WARNING * WARNING * WARNING *

##############################################################################
# PROGRAM:              fix_pkg
# USAGE:                fix_pkg <PKGNAME>
# PURPOSE:              Certain types of patchadd failure result in a missing
#                       /var/sadm/pkg/*/pkginfo for one (or more) packages
#                       without this file, the system becomes unmaintainable as
#                       any patches or packages that depend on the affected
#                       package cannot be added.
#                       The ideal and Sun supported solution is to upgrade the
#                       server, but that may not be able to be accomplished in
#                       a timely manner, in addition upgrade tools such as LU
#                       are not supported from a system that is not up-to-date
#                       on patches.
#                       This program recreates a best guess pkginfo file and
#                       returns the system to a maintainable state.
#
#                       Symptoms of the problem:
#
#                       In Solaris 8, you get an error message when running
#                       showrev -p:
#                                       showrev: /var/sadm/pkg/SUNWcsu/pkginfo \
#                                                       - No such file or directory
#
#                       In all affected versions, patchadd is unable to install a
#                       patch either reporting directly that the pkginfo file
#                       is damaged or broken or that one or more patch packages
```

```
#                       associated with the patch are not installed, but
#                       the associated files are installed and listed in
#                       /var/sadm/install/contents and /var/sadm/pkg/<PKG> exists.
#
#                       Cause of the problem:
#
#                       The Sun support center told me that this happens as a result
#                       of installing patches (particularly kernel patches) in
#                       anything other than Single-User mode.  I and many other
#                       admins I know regularly install patches on quiescent but
#                       non-single-user mode systems and yet this problem is very
#                       rare.  All the systems where I've seen this problem occur
#                       have a history of the patchadd process having received a
#                       SIGINT or SIGKILL during patch installation - patchadd
#                       traps these signals and backs out any partially installed
#                       patches, but may not always restore the pkginfo file
#                       appropriately.  However, I have successfully used
#                       SIGINT to stop a running patchadd process with no ill
#                       effects.
# EXIT VALUES:          0 if any pkginfo is restored
#                       1 if an error occurs
#
# NOTE:                 The following command can be used to get a sample of signatures
#                       for use with the Solaris Fingerprint Database
#                       (http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl)
#                       to ensure that you really do have a valid but broken package:
#                        md5 `grep " ${PKG}$" /var/sadm/install/contents | nawk 'NF=10 &&
#                                       $2 ~ /^[fe]$/{print $1}' | tail -256`
#
# AUTHOR:   Julie L Baumler 20030918-20030926
################################################################################

usage(){
        echo "USAGE:"
        echo "          $0 <pkg>"
        echo "            - <pkg> is the name of a broken package (missing pkginfo)"
}
```

```
TMP="/var/tmp" # what to use for temporary files

# Check for appropriate usage
if [ $# -ne 1 -o $1 = "help" -o $1 = "-help" -o $1 = "-?" ]
then
        usage
        exit 1
fi

# Determine which package we are working on
PKG=${1}

# Perform some validity testing
if [ ! -d /var/sadm/pkg/${PKG} ]
then
        echo "ERROR: There is no directory for ${PKG} in /var/sadm/pkg" >&2
        exit 1
fi

grep "${PKG}[ $]" /var/sadm/install/contents > /dev/null
if [ $? -ne 0 ]
then
        echo "ERROR: ${PKG} does not occur in /var/sadm/install/contents" >&2
        exit 1
fi

if [ -f /var/sadm/pkg/${PKG}/pkginfo ]
then

        echo "ERROR: ${PKG} already has a pkginfo file" >&2
        exit 1
fi

# Log what we are doing
/bin/logger -p user.notice -t `/bin/basename $0` rebuilding pkginfo file for ${PKG}
```

```
# Move to the package directory
cd /var/sadm/pkg/${PKG}

#
# Find the most recent patch to be added, extract the pkginfo file from it
#
# This is currently using modification time, which is not an ideal test
# as the patchadd backout process can create multiple files with the
# same mtime - it does seem to be good enough though.
# Ideally, we would pick the best match pkginfo file, but that involves
# a lot more processing. (Best match defined as the listing of the most
# known to be installed patches.)
#
# This doesn't get you anything if you haven't previously patched the
# package, however since you can only get into this  situation if you
# have patched, it's not really an issue.  If you have only patched with
# the no save option, you are just out of luck
#
zcat `ls -t save/1*/undo.Z  | tail -1` > ${TMP}/$$datastream$$
if [ -f ${TMP}/$$datastream$$ ]
then
                mkdir ${TMP}/$$pkg$$
                pkgtrans -i ${TMP}/$$datastream$$ ${TMP}/$$pkg$$ > /dev/null 2>&1 << HERE
1
HERE
                rm ${TMP}/$$datastream$$     # cleanup
fi

if [ -f ${TMP}/$$pkg$$/${PKG}/pkginfo ]
then
                echo "Restoring previous pkginfo file"
                # remove the data that we'll need to reset
                # and move restored pkginfo into the target directory

                # ACTIVE_PATCH and ACTIVE_OBSOLETES are set during
```

```
		# patch installation for backout purposes and
		# don't seem to have a use once a patch is installed
		# but aren't necessarily removed
		# remove them here, because the values may be outdated
		# PATCHLIST is populated with current data later

		egrep -v "PATCHLIST|ACTIVE_PATCH|ACTIVE_OBSOLETES"
${TMP}/$$pkg$$/${PKG}/pkginfo > ./pkginfo.$$

else
		echo "WARNING: No pkginfo file in previous patch save file"
		echo "created pkginfo will contain only the PATCH_INFO_ lines"
		echo "gather remaining data from the Solaris CD or original source"
		incomplete="YES"
		touch ./pkginfo.$$
fi

rm -r ${TMP}/$$pkg$$ # cleanup

# Get patch information
/bin/showrev -p 2>/dev/null > ${TMP}/showrevp.$$

# create the PATCH_INFO_* line for any patches that aren't listed
# in the pkginfo file but were save installed
for patch in `(cd ./save; ls -dt 1*)`
do

		grep "PATCH_INFO_${patch}" ./pkginfo.$$ > /dev/null
		if [ $? -eq 0 ]
		  then
			# we've already got info on this patch in pkginfo
			# echo "${patch} is already in pkginfo"
			continue;

		fi
```

```
		#echo "Adding ${patch} to pkginfo"


		# Add the information about this patch

		grep "Patch: ${patch}" ${TMP}/showrevp.$$ > /dev/null
		if [ $? -ne 0 ]
		then
			# we don't have information about this patch
			# because it only applies to this package
			# so, put in stub data
			# NOTE: this data could be determined from
			# the patch's README file
			echo "PATCH_INFO_${patch}=Installed: Thu Jan  1 01:02:03 PST 1970
From: unknown Obsoletes: Requires: Incompatible:" >> ./pkginfo.$$

		else

			#Get the install date (or close estimate) in the form
			#Day Mon ## HH:MM:SS PST YYYY
			# just assume it's a Monday, PST, and 2003 if we need to
			date=`ls -ld save/${patch} | nawk '{ year=($8 ~ /:/)?2003:$8;
			    $8=($8 ~ /:/)?$8:"05:01";
			    printf("Mon %s %2d %s:01 PST %d\n", $6, $7, $8, year);}'`

			echo "PATCH_INFO_${patch}=Installed: $date From: "`hostname`""`grep
"Patch: ${patch}" ${TMP}/showrevp.$$ | sed "s/Patch: ${patch}//" | sed 's/Packages:.*$//'` >> ./pkginfo.$$
		fi
done

rm ${TMP}/showrevp.$$ # cleanup

# Determine the patches that have been installed
# PATCHLIST is usually set before the PATCH_INFO lines (or the most
# recent set of PATCH_INFO lines), but patchadd, patch install/* scripts, ...
# don't use line relative location to find the variables they need
# they use the variable name
```

```
echo "PATCHLIST="`grep PATCH_INFO ./pkginfo.$$ | sed 's/PATCH_INFO_\(.*\)=Installed.*$/\1/'` >>
./pkginfo.$$


# Copy the pkginfo file in if it's complete
# Tell user where it is if it's incomplete
# Double check that we're not clobbering an existing pkginfo
if [ "X${incomplete}" = "XYES" ]
then
        # we didn't find an old pkginfo file from a patch
        partial_name="./pkginfo."`/bin/basename $0`"."`/bin/date +%Y%m%d%H%M%S`
        echo "Partially restored pkginfo file saved as ${partial_name}"
        mv ./pkginfo.$$ ${partial_name}
        exit 0


elif [ ! -f ./pkginfo ]
then
        mv ./pkginfo.$$ ./pkginfo
        exit 0
else
        echo "ERROR - something else created pkginfo while "`/bin/basename $0`\
                " was running" >&2
        partial_name="./pkginfo."`/bin/basename $0`"."`/bin/date +%Y%m%d%H%M%S`
        echo "Restored pkginfo file saved as ${partial_name}"
        mv ./pkginfo.$$ ${partial_name}
        exit 1
fi
```

Bibliography

Baroudi Bloor. The Patching Problem: It's Costing Your Business Dollars. 22 May 2003. <http://www.baroudi.com/pdfs/patch.pdf>.

Cole, Eric, Jason Fossen, Stephen Northcutt, and Hal Pomeranz. SANS Security Essentials with CISSP CBK. Version 2.1. Vol. 2. GIAC Certification Series. The SANS Institute, 2003.

Cornely, David. "Re: Tape drives keep going down upon use on Solaris 8 master server after upgrade." Online Posting. 28 August 2003. Veritas-bu. 28 August 2003. <http://mailman.eng.auburn.edu/mailman/listinfo/veritas-bu>.

Cromar, Scott. "Configuration and Patch Verification on Solaris Systems." 23 February 2003. 22 August 2003. <http://www.sans.org/rr/paper.php?id=921>.

Dik, Casper. Fastpatch. 17 September 2003. <ftp://ftp.wins.uva.nl/pub/solaris/auto-install/>.

Geer, Dan. "Patch Work." ;login: August 2003: 27-28.

Little, Jody and Jacqueline Roundy. "Patch Management Best Practices." SunNetwork. November 2002. 23 May 2003. <http://sunsmart.sun.com>.

"New Patch Delivery Infrastructure" SunSpectrum InfoExpress Newsletter, May 2003. 22 August 2003. <http://www.sun.com/service/support/infoexpress/may2003/0503-05.html>.

Pethia, Richard. "Viruses and Worms: What Can We Do About Them?" Testimony of Richard D. Pethia; Director, CERT® Coordination Center; Software Engineering Institute; Carnegie Mellon University; Pittsburgh, PA 15213 Before the House Committee on Government Reform: Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census: Hearing on Worm and Virus Defense: How Can We Protect the Nation's Computers From These Threats? September 10, 2003. 12 September 2003. <http://www.cert.org/congressional_testimony/Pethia-Testimony-9-10-2003>.

Radhakrishnan, Ramesh. "A Patch Management Strategy for the Solaris Operating Environment." Sun Blueprints Online – January 2003. 20 May 2003. <http://www.sun.com/solutions/blueprints/0103/817-1115.pdf>.

Rescorla, Eric. Security holes... Who cares? 21 August 2003. <http://www.rtfm.com/upgrade.pdf>.

Rogers, Larry "Improving Security – Larry Rogers – Applying Patches". 1 April 2001. 12 September 2003.

<http://www.cert.org/homeusers/apply_patches.html>.

Sagewire. "Solaris changing authentication logic midstream." 21 August 2003. <http://sagewire.sage.org/Articles/03/07/27/057258.shtml>.

sfpC: Solaris Fingerprint Database Companion, vers. 1.2. 30 May 2003. <http://www.sun.com/solutions/blueprints/tools/index.html>.

Shamblin, Joe. PatchReport (rev. 2.x) -- Solaris patch tool. 22 August 2003. <ftp://x86.cs.duke.edu/pub/PatchReport/index.html>.

Stokely Consulting. "Solaris Patch Management". Unix SysAdmin Resources: FAQs, Patches, and Other Info (Sun 3/3). 12 Sep. 2003. <http://www.stokely.com/unix.sysadm.resources/faqs3.sun.html#solaris.patch.mgmt>.

Sun Managers. <http://www.sunmanagers.org>.

Sun Microsystems. Configuration and Service Tracker (CST). 12 October 2003. <http://www.sun.com/service/support/cst/>.

Sun Microsystems. Signed Patches Administration Guide for PatchPro 2.1. March 2003. 10 October 2003. <http://sunsolve.sun.com/patches/spag.pdf>.

Sun Microsystems. "Sun Newsletters," 13 October 2003 <http://www.sun.com/newsletters>.

Sun Microsystems. patchdiag.xref. 12 October 2003. <http://sunsolve.sun.com/pub-cgi/patchDownload.pl?target=patchdiag.xref&method=H>.

Sun Microsystems. "Solaris Patch Management: Recommended Strategies: A White Paper." 2 October 2002. 22 August 2003. <http://www.sun.com/service/support/sw_only/pmstrategies10.02.pdf>.

Sun Microsystems. Sun Developer Connection – The Patch Comparison Utility (PCMP). 12 September 2003. <http://solaris.java.sun.com/tools/pcmp/pcmp.html>.

Sun Microsystems. Sunsolve Patch Portal. <http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access>. <http://sunsolve.sun.com/private-cgi/show.pl?target=patchpage>.

Acknowledgement

I would like to thank James Vermillion, my Sun Systems Support Engineer, for pointing out the Baroudi Bloor article and discussing all of my questions, complaints, and ideas about the patching process over the last few years.

Last Updated: July 25th, 2017

Upcoming Training

SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event