USENIX Association

Proceedings of the 17th Large Installation Systems Administration Conference

San Diego, CA, USA
October 26–31, 2003

USENIX
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Radmind: The Integration of Filesystem Integrity Checking with Filesystem Management

Wesley D. Craig and Patrick M. McNeal – The University of Michigan

ABSTRACT

We review two surveys of large-scale system management, observing common problems across tools and over many years. We also note that filesystem management tools interfere with filesystem integrity checking tools. Radmind, an integrated filesystem management and integrity checking tool, solves many of these problems. Radmind also provides a useful platform upon which to build further large-scale, automatic system management tools.

Introduction

We present "Radmind," an open source suite of Unix command-line tools and a server designed to remotely administer the filesystems of multiple, diverse Unix machines. At its core, Radmind checks filesystem integrity. Like Tripwire [Spafford], it is able to detect changes to any managed filesystem object. Radmind, though, goes further than just integrity checking. Files are stored on a remote server so that Radmind can reverse changes if filesystem integrity is lost. Changes can be made to managed machines by updating files on the Radmind server.

Each managed machine may have its own *load-set* composed of multiple, layered *overloads*. Overloads consist of a list of filesystem objects and any associated file data. This allows, for example, the operating system to be described separately from software packages.

One of the great strengths of Radmind is that it does not enforce a management ideology. It allows a system administrator to manage many machines using roughly the same skill set she already uses to manage one machine. It also provides a useful filesystem integrity check that neither undermines nor is undermined by filesystem management.

State of the Art

In the recently published "Experiences and Challenges of Large-Scale System Configuration," Anderson, et al. [Anderson 03] review several large-scale system management tools: SUE [CERN], cfengine [Burgess], LCFG [Anderson 02], and several ad hoc solutions. They identify several common problems:

- There is no method to determine that a running node has diverged from its specification. Without this information, the only option is to reinstall.
- The complexity of existing configuration tools requires extensive knowledge, above what may be expected of a newly hired but experienced system administrator.
- New operating systems and new operating system versions are a major problem.
- Server management is typically ad hoc, not part of the large-scale configuration management system.
- There is no support for disconnected nodes, e.g., laptops.

Seven years earlier in "An Analysis of UNIX System Configuration," [Evard 97] Evard describes many of these same problems. He also notes that large-scale configuration management has been "an area of exploration for at least ten years." While many sophisticated tools have been made available in the last seventeen years, the same problems are still prevalent.

Also of note is the degree to which filesystem integrity checking conflicts with these system management tools. Groups like SANS and CERT list filesystem integrity checking as one of the basic procedures that all system administrators should use to help secure their computers. However, if a cluster is running both a filesystem integrity tool for intrusion detection and, e.g., rsync [Tridgell] for software updates, each time an update occurs, every machine will report a security event. For large clusters, these reports are noise in which real problems may be lost [Arnold]. In order to update the managed systems without triggering security events, the system management tool must be aware of (or integrated with) the intrusion detection tool.

Integrating intrusion detection with system management affords several additional capabilities that make it attractive. While filesystem integrity checking tools detect the changes that an attacker has made to the filesystem, they can also detect certain types of data corruption caused by operating system and hardware errors, e.g., a flaky disk driver or controller. Because filesystem integrity checking tools scan virtually all of the nonvolatile filesystem, they detect the changes that the system administrator makes as well. For example, they can detect what "make install" has done, or RPM [Bailey] , or even "vi." Once captured, this information can be propagated by the system administrator to manage an entire cluster of machines.

Synctree [Lockard], also written at the University of Michigan, is able to leverage filesystem integrity information for the purpose of system management. However, Synctree requires AFS as a file transfer mechanism and requires AFS system:administrator ("root" in AFS) access. In a non-AFS environment, it would be an excessive burden to bring up an AFS cell just for system management. Moreover, a non-AFS-dependent mechanism would be required to securely manage the AFS servers.

Despite Synctree's shortcomings and minimal adoption, it demonstrates that combining integrity checking with filesystem management is a viable methodology. Below, we demonstrate the advantages of this methodology and how it can be used to resolve the issues that have been plaguing automatic system administration for nearly two decades.

Divergence

Because Radmind includes a filesystem integrity checking tool, fsdiff (filesystem difference), that traverses the nonvolatile portions of the filesystem, it is possible at any time to compare a live node with a well-defined specification. Any modifications, additions or deletions to the local filesystem are reflected in the output of fsdiff. Therefore, it is possible to determine a priori the correctness of a node. When a node diverges from its specification the system administrator has the option to audit the *transcript* (Figure 1) of differences.

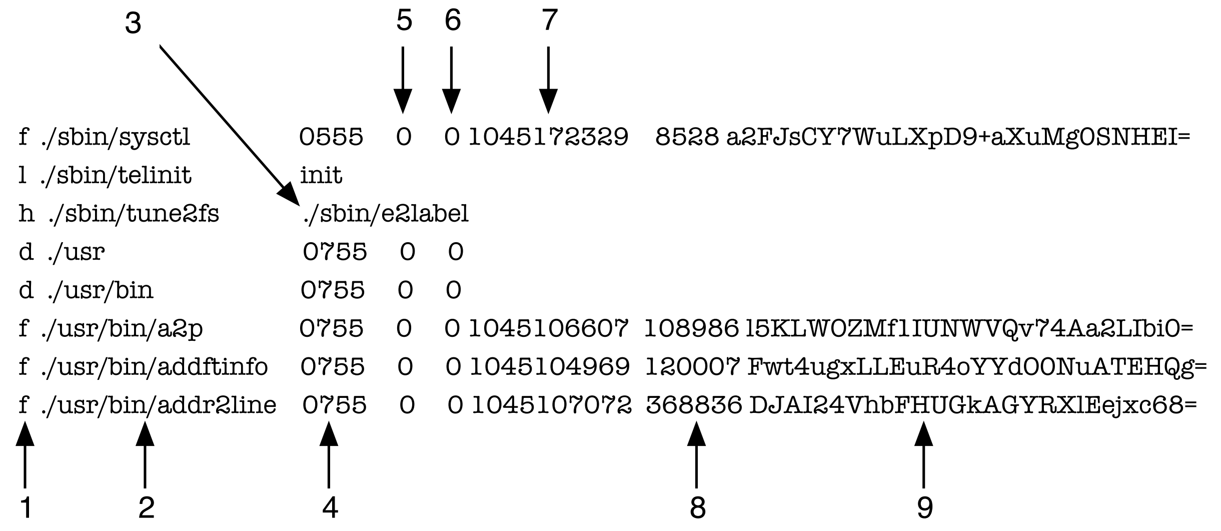
Without this information, the system is always running in an indeterminate state. When the system administrator responds to aberrant behavior from a particular node, she cannot distinguish intrusion, hardware failure, software bugs, staff error, etc., from system divergence. The only way to eliminate system divergence as a possible source of error is to reinstall.

Radmind, on the other hand, can utilize a difference transcript to return the node to its specification. A separate tool, lapply (loadset apply), reads a transcript and modifies the filesystem accordingly. lapply removes, modifies and creates filesystem objects, downloading files from a Radmind server as necessary. Since the transcript lists only differences, lapply makes the minimum changes required.

Interestingly, if the specification has changed, the same procedure can be used to install updates onto nodes. The specification consists of a series of transcripts and a listing of their precedence, known as a *command file* (Figure 2). The ktcheck (command file and transcript check) tool calculates a checksum of the local copy of the command file and transcripts, compares it with a checksum returned by the Radmind server, and optionally updates the local copies. When ktcheck indicates that the command file or transcripts have been updated, differences from the specification are treated as intentional updates rather than anomalies in the filesystem. The ability to differentiate a system update from a security event makes the security event reporting valuable, particularly in a large-scale environment.

Complexity

Meta-configuration tools like LCFG, PAN [Cons], and cfengine each implement their own declarative language to manage the contents of configuration files. They do not directly address the management of the remainder of the filesystem. For example, cfengine deployments use other tools, e.g., RPM, to manage system binaries. In order to use cfengine with



```
f ./sbin/sysctl        0555  0  0 1045172329   8528 a2FJsCY7WuLXpD9+aXuMg0SNHEI=
l ./sbin/telinit       init
h ./sbin/tune2fs       ./sbin/e2label
d ./usr                0755  0  0
d ./usr/bin            0755  0  0
f ./usr/bin/a2p        0755  0  0 1045106607 108986 l5KLWOZMf1IUNWVQv74Aa2LIbiO=
f ./usr/bin/addftinfo  0755  0  0 1045104969 120007 Fwt4ugxLLEuR4oYYdOONuATEHQg=
f ./usr/bin/addr2line  0755  0  0 1045107072 368836 DJAI24VhbFHUGkAGYRXlEejxc68=
```

Figure 1: Fragment of a Linux From Scratch base transcript. 1. File type. Shown are regular files (f), a symbolic link (l), a hard link (h), and directories (d). 2. Pathname. Transcripts are sorted by pathname. 3. Link target. 4. Mode, listed in octal. 5. Owner ID. 6. Group ID. 7. mtime. 8. Size in bytes. 9. Base64 encoded SHA-1 cryptographic checksum.

RPM, all software must be packaged, even if it is built from source. Unfortunately, the system administrator is then required to learn two complex applications and is still unable to fully verify the integrity of the non-volatile filesystem.

The above meta-configuration tools have semantic knowledge of the contents of configuration files. This implicit knowledge forces the system administrator to use the tool's language to update or create a configuration file, preventing her from using her preferred method. If the system administrator uses a tool like Webmin [Cameron] in a cfengine environment, cfengine will simply overwrite her changes. There is no provision in cfengine for capturing complex changes, saving them, and automatically incorporating them into a given configuration.

In a Radmind environment, fsdiff is able to detect any changes, regardless of how the changes were made. This permits the system administrator to use whatever tool is appropriate to make changes. A separate tool, lcreate (loadset create), reads fsdiff's output and stores the transcript and any associated files to a Radmind server (Figure 3). Once the loadset is on the server, the system administrator can either merge the new loadset into an existing loadset or treat it as an overload. By not merging, the system administrator can easily test new loadsets while retaining the option to back them out. This allows a system administrator to manage a large number of diverse machines using normal system administration tools plus Radmind without requiring additional scripting or programming.

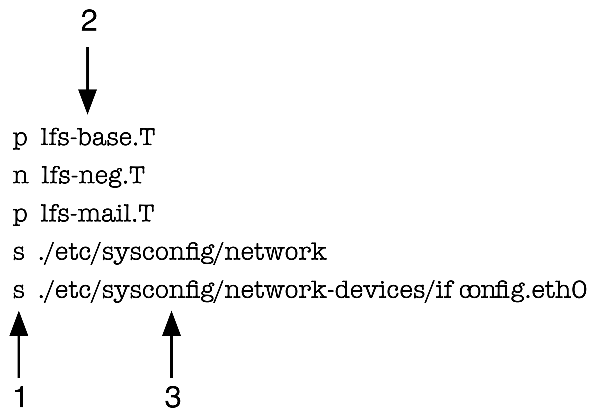
Portability

During an operating system upgrade, configuration files, system binaries, and the kernel are updated. Tools like cfengine are unable to directly manage system binaries and the kernel. This makes operating system upgrades challenging. Since cfengine's classes, scripts and edits contain the semantics of the system configuration files, adding support for new operating systems often involves writing substantial new code. Current meta-configuration tools support a limited number of operating systems, and a system administrator must wait to deploy a new operating system until its semantics have been incorporated. Once a version of cfengine supports the target operating system, the system administrator still must revise her site-specific cfengine scripts and configuration files.

By contrast, Radmind has no knowledge of the contents of files. Files are instead treated as simple byte-streams. Porting Radmind to new platforms is easy, typically requiring only recompilation. In unusual cases such as Mac OS X's HFS+ multi-forked files or Solaris "door" files, support for new filesystem objects may need to be added to the code base.

While treating managed files as byte-streams greatly improves portability, it also has some drawbacks. Some applications use a single file for configuration, e.g., inetd uses /etc/inetd.conf. With Radmind, two inetd.conf configurations that differ by only a single line require two separate files. In an environment with even moderate diversity, maintaining the common portions of many inetd.conf files is a chore. There are many solutions to this problem. xinetd [xinetd], for example, may load the contents of a directory for its configuration. This allows the common configuration to be shared amongst a large number of machines with the differences represented by separate whole files. Or, to use Radmind with traditional inetd, inetd's startup script could synthesize /etc/inetd.conf from a directory containing inetd.conf fragments.

A common rationale for encoding the semantics of configuration files into meta-configuration tools is the ability to make a single change and affect an entire infrastructure. For instance, one could specify that a



```
p lfs-base.T
n lfs-neg.T
p lfs-mail.T
s ./etc/sysconfig/network
s ./etc/sysconfig/network-devices/ifconfig.eth0
```

Figure 2: Command file for a Linux From Scratch mail server. 1. Transcript type. Listed are positive transcripts (p), a negative transcript (n), and special files (s). 2. Transcript name. 3. Special file pathname. Positive transcripts list the managed portion of the filesystem. Negative transcripts define the unmanaged portions of the filesystem. Transcripts are listed from lowest to highest precedence. The special transcript, built from the list of special files, always has the highest precedence.

machine should be an NTP server, and simultaneously that other machines running multiple operating systems should use that NTP server. This ultra-abstract configuration management is another source of complexity and non-portability inherent in meta-configuration tools. The problem described above is more easily managed by removing the configuration to a service location mechanism such as ZEROCONF [IETF], DHCP, or DNS. Evard makes a similar observation, advising system administrators to ''migrate changes into the network and away from the host'' [Evard 97].

Servers

Large-scale configuration management tools are most popular in computing clusters, labs, and other environments where hardware and software diversity is very low. However, even when widely deployed, these systems are often not used to manage servers. As an example, the 100 nodes of a computing cluster might be managed while the two or three supporting servers are built by hand. While it is easy to share one image for all of the cluster nodes, existing tools have difficulty managing the differences between servers and between cluster nodes and servers.

With Radmind, fsdiff is able to automatically detect and report the differences between machine specifications. A system administrator can install a server with the same base loadset as a compute node, install additional software required for, e.g., NTP, and capture those changes as an overload. This overload plus the cluster base loadset can then be applied to any machine to produce an NTP server.

When machines are exactly alike except for host-specific files, an overload is not necessary. Such machines can share one command file, listing the appropriate transcripts and any *special files*. When ktcheck runs, host-appropriate special files will be referenced. In this way, machines can share the bulk of their specifications while retaining their individuality.

Laptops

Laptops are particularly challenging due to their volatile network configuration. Mobile computers are on the network only intermittently. Moreover, when they are connected, the available bandwidth is often low. Configuration management tools are also challenged by the potentially weak security of the network and the fact that machines may need to be updated from any IP-address.

Since all Radmind functions are initiated by the managed machines, updates can be triggered by the user when he knows that the network will be available. Moreover, the tools fail gracefully when the network is unavailable. Radmind also minimizes network traffic in the case where the loadset has not changed. If changes are detected, Radmind transfers only the necessary files.

For network security, Radmind supports SSL-encrypted links. This allows nodes on insecure networks to be updated securely. SSL certificates can also be used

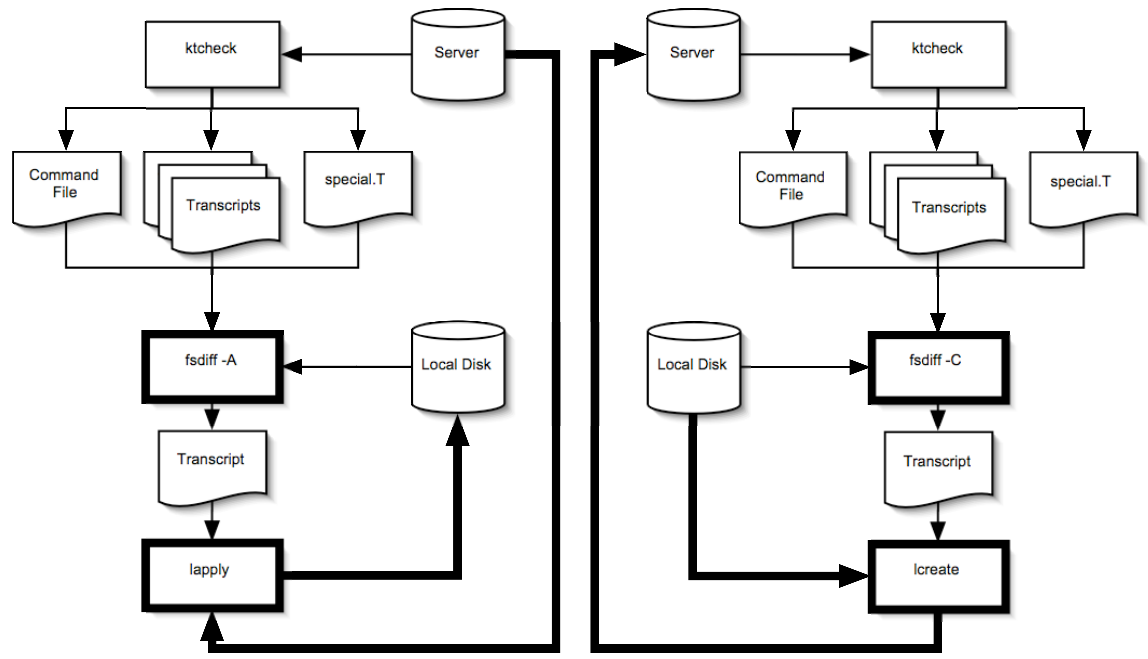


Figure 3: Overview of Radmind tools and data flow. Like many classic Unix tools, fsdiff's input and output are in the same format. fsdiff is able to output two types of transcripts, *apply-able* (left) and *create-able* (right). Apply-able transcripts describe the changes needed to make a filesystem match a specification. Create-able transcripts describe the changes needed to make a specification match a filesystem.

to authenticate both the Radmind server and the managed clients, regardless of DNS or IP-address variation.

Future Work

Radmind does not directly address the initial installation of machines. Instead, most system administrators install a minimal operating system on each machine before it can be managed with Radmind. More sophisticated solutions are possible, e.g., using PXE [Intel] or NetBoot [Apple]. In addition, work has been done to create a "Radmind CD." These CDs are built from a production base loadset and an autoinstall overload. Using this methodology, creating new CDs to support additional hardware is trivial.

Since Radmind only manages the filesystem, other methodologies are required to manage the master boot record of a machine, processes, etc. Any one of these activities is relatively easy to layer on top of fsdiff, either before or after lapply is run. It is difficult to automatically determine all of the correct actions to take when large filesystem changes are made. Tools like sowhat [Couch] are helpful for determining interdependencies among filesystem objects, but again, leave much manual work to the system administrator. Automatically determining the correct order of process stops and starts is nontrivial. Like automatic system administration in general, this area is ripe for further exploration.

Conclusion

Radmind is an open source, complete solution to the large-scale filesystem management problem. Its external requirements are small, and the tools themselves are easy to use – there is no additional language to learn. Radmind's security is based on OpenSSL, both for SSL/TLS encryption and authentication and for support of cryptographic hash functions such as SHA-1 and RIPEMD-160. Radmind is freely available from http://radmind.org .

Author Information

Wesley D. Craig joined the University of Michigan in 1987, where he designed and wrote netatalk. He is currently the Senior IT Architect and Engineer for the University's Research Systems UNIX Group where he manages the team that runs the University's central LDAP directory, e-mail, and charged-for printing systems. Reach him electronically at wes@umich.edu .

Patrick M. McNeal earned his BSE in computer engineering from the University of Michigan. Upon graduation, he joined the University's Research Systems UNIX Group as a software engineer. He works on the Radmind project along with other e-mail-related services. Reach him electronically at mcneal@umich.edu .

References

[Anderson 02] Anderson, P. and A. Scobie, "LCFG – The Next Generation," *UKUUG Winter Conference*, http://www.lcfg.org/doc/ukuug2002.pdf , 2002.

[Anderson 03] Anderson, P., G. Beckett, K. Kavoussanakis, G. Mecheneau, and P. Toft, "Experiences and Challenges of Large-Scale System Configuration." *GridWeaver Project*, http://www.epcc.ed.ac.uk/gridweaver/WP2/report2.pdf , March, 2003,

[Apple] Apple Computer, Inc. http://www.apple.com .

[Arnold] E. Arnold, "The Trouble With Tripwire: Making a Valuable Security Tool More Efficient," http://www.securityfocus.com/infocus/1398 , June 6, 2001.

[Bailey] Bailey, E., "Maximum RPM (RPM)." MacMillan Publishing Company, August, 1997.

[Burgess] Burgess, M. and R. Ralston, "Distributed Resource Administration Using Cfengine," *Software: Practice and Experience*, Vol. 27, 1997.

[Camero] Camero, J., "Webmin: A Web-based System Administration Tool for Unix," Freenix 2000, *USENIX Annual Technical Conference*, 2000.

[CERN] CERN, *SUE*, http://wwwpdp.web.cern.ch/wwwpdp/ose/sue/doc/sue.html .

[Cons] Cons, L. and P. Poznanski, "Pan: A High-Level Configuration Language," *Proceedings LISA XVI*, USENIX Association, 2002.

[Couch] Couch, Alva L., "Global Impact Analysis of Dynamic Library Dependencies," *Proceedings LISA XV*, USENIX Association, 2001.

[Evard 97] Evard, R. "An Analysis of UNIX Machine Configuration." *Proceedings LISA XI*, USENIX Association, 1997.

[IETF] IETF, *Zeroconf Working Group*, http://www.ietf.org/html.charters/zeroconf-charter.html .

[Intel] Intel Corporation, "Preboot Execution Environment (PXE) Specification," September 20, 1999.

[Lockard] Lockard, J. and J. Larke, "Synctree for Single Point Installation, Upgrades, and OS Patches," *Proceedings LISA XII*, USENIX Association, 1998.

[RPM] RPM, http://www.rpm.org .

[Spafford] Kim, G. and E. Spafford, "Experiences with TripWire: Using Integrity Checkers for Intrusion Detection." *Proceedings System Administration, Networking, and Security, III*, Usenix Assoc., 1994.

[Tridgell] Tridgell, A. and P. Mackerras, "The rsync Algorithm," Australian National University, *TR-CS-96-05*, June, 1996.

[xinetd] *xinetd*, http://www.xinetd.org .