RegColl: Centralized Registry Framework for Infrastructure System Management

Brent ByungHoon Kang, Vikram Sharma, and Pratik Thanki
– University of North Carolina at Charlotte

ABSTRACT

System administrators are faced with the challenge of managing and compiling information about deployed systems to ensure the maintenance, scalability, security, and overall availability of the infrastructure systems. Recently, securing and monitoring the deployed enterprise infrastructure systems has taken unprecedented importance due to the added accountability now being placed on managing corporate data and private information. Compliance with new regulation has never been stressed more than now – with SOX/Bill 198, GLB Act/PIPDEA, HIPPA to name a few regulations.

Toward this, we present RegColl: A practical framework for registry and system configuration management for monitoring and maintaining the health of corporate infrastructure systems in Windows environment. In RegColl, registry and configuration information are collected at a centralized location so as to inspect and analyze the data for policy compliance, system configuration monitoring, and incident response services.

Introduction

Managing hardware, operating systems, and applications on workstations and desktops in corporate and large networked environments is a big challenge, one that System administrators deal with on a day to day basis. To monitor the health of infrastructure systems, it is imperative to understand system behavior, especially aspects that are related to the upkeep of the system and eventually the whole network. Windows *registry* provides system information to understand the behavior of the deployed applications.

Monitoring the system's registry and configuration information offers important insights in deployed software, patches, drivers, etc. – such entries can be analyzed for vulnerabilities, troubleshooting, and compliance with standards, along with providing documentation in an event that might have reporting implications. Further, by using the registry and administrative tools in Windows, system administrators can provide local or remote support, query registry logs for security analysis, and can apply compliance checks. We extend this facility in centralizing system monitoring services to run audits, apply compliance standard checks, and enforce corporate & legislature policies.

The problem of maintaining healthy configurations of a large installed base and other third party software packages has been recognized as a daunting task [10]. The growing numbers of software and changing system configuration makes it even more difficult to specify an "ideal configuration," which makes troubleshooting and maintenance problems intractable [8]. Aside from compliance and network monitoring issues, system troubleshooting expenses drive the total cost of network ownership, with maintenance and support costs. Largely, troubleshooting cases are due to system misconfiguration [8] and hence, it is one of the issues we propose to tackle with RegColl's centralized registry framework.

Non-compliance with legislature requirements such as Sarbanes-Oxley [16], Gramm-Leach-Bailey (GLB) Act [9], Health Insurance Portability and Accountability Act (HIPAA) of 1996 [11], and others that preserve private information is putting more and more organizations under scrutiny. Such compliance deviations are not only a breach of law but they also affect client confidence. Further, legislatures like OCC (Office of Comptroller of the Currency) [4] and HIPAA require organizations to maintain an information security program that includes "Detecting, Preventing and Responding to Attacks, Intrusions, or other Systems Failures."

Computer intrusions [5] leading to theft of intellectual property or any system compromise in a corporate network can jeopardize users' and employees' data integrity. Furthermore, any incident leading to misuse of confidential data can put a company's reputation at risk. Implication of such incidents may include comprehensive incident response plans and liability to notify the reporting agencies and the affected customers.

Traditionally management plans are devised for Network and IT Infrastructure which cover the security policies, audits, asset inventory, and incident response. Such plans ensure business continuity and disaster recovery in the event of computer and network intrusion or system compromise. Federal Trade Commission (FTC) Safeguards rule [7], OCC and Medical Privacy rules like HIPAA mandate an *incident response* plan which encompasses ways to confirm whether an incident occurred and provide accurate, relevant, and timely information.

We present the benefits of a centralized registry collection and monitoring approach in the next section. Subsequently, we give the architecture details of RegColl framework and its components including the collection mechanism and transfer mechanisms, and we discuss the implementation of RegColl framework. The related work section compares some of the existing technologies and their goals with RegColl framework. We then present the future work section, showing the ongoing development efforts and the proposed upgrades of RegColl framework. We then conclude, revisiting the usage and presenting the practical solutions that might be deployed to realize some of the key benefits of RegColl.

Why Registry Monitoring

To address requirements of system monitoring, configuration management, and incident response, we take a proactive system management. In this approach, we freeze configuration states that can stand comparison to a misconfigured system. In the event of an incident these frozen states provide a historic log of system registries containing restore points that allow for reconfiguration, System rebuild, and incident response.

The centralized registry offers key benefits like Policy enforcement at a centralized location, Audits and Analysis of configuration information, and the overall System Configuration Management. The following sections elaborate these benefits.

Policy Enforcement

Corporate and legislature requirements mandate formulation of a management plan and steps to control access to infrastructure systems, and ensure the security of financial data and non-public information. Centralized repository serves these purposes by providing a single point of policy validation and enforcement. Audits and compliance requirement checks are achieved by deploying compliance and auditing standards on the stored configuration.

- **Remediation** – Centralized configuration offers remediation capabilities to easily and effectively automate and monitor policy consistency requirements. The remediation technique allows for automated and user controlled repair process, for example, re-importing registries to enforce corporate policy across thousands of machines.
- **Restore Points and Time Travel** – Centralized configuration management allows for system restore and time travel by maintaining restore-histories. The causal relationship between the change information ensures a mechanism identifying the restore points and version changes. Such a tracking mechanism would enable reintegration of changes when a disconnected entity rejoins the network. Restore points and configuration change tracking features facilitate rollback to a stable state system state in the event of software misconfiguration.
- **Change Management** – Centralized configuration management detects and responds to planned and unplanned changes by setting standards and policies for change enforcement. Registry and configuration information captures the changes in system's configuration; any planned changes can be verified by studying the configuration change information.

Audit & Analysis

Risk management practices require audits and analysis of configuration information to assess and mitigate any risk arising from vulnerabilities in the system. Analysis of system, registry, and configuration logs is an important aspect of troubleshooting. Furthermore, such techniques take more prominence in rating and certification of processes and organization, and not ignoring that they are legislature requirements as well. Towards this, a centralized repository will serve as a common source for audit and analysis such as:

- **Security Auditing** – The RegColl framework proposes to collect and compare thousands of configuration elements from a networked system to identify and enforce security configurations standards. For example, security relevancy check [18] is done by inspecting security relevant registry keys.
- **Dynamic Auditing** – Centralizing registry and system configurations for *dynamic auditing* allows for systems to be audited seamlessly at a central and secured location on a regular basis. Auditing ensures consistent compliance and provides detailed reporting to address regulatory compliance requirements.
- **Security Analysis** – Centralizing registries give a single source of security and vulnerability analysis for tools developed and discussed by Wang [8] & Wenliang Du [18].
- **Gap Analysis** – These stores prove to be a single source for running gap analysis along with assessment of system configurations for known configuration vulnerabilities including patch inconsistencies.

System Configuration Management

Registries and configuration information is shared and accessed by multiple applications, such as drivers, software, and patches – it is imperative to cache a snapshot of this information so as to establish checkpoints at every change. Registry changes are monitored and pushed to a centralized location as and when new changes are recorded. The monitoring process polls the registry and configuration files and triggers the export of registry database as key entries are changed in the registry. These system-state snapshots are stored into a centralized secure location to analyze system vulnerabilities, run audits, enforce corporate and compliance policy checks, and monitor for any rogue software installations.

Incident Response

Incident response identifies ways to confirm whether an incident occurred and then provide accurate, relevant, and timely information. In such an event, the incident information gives implementation controls to secure the systems and crime scene to protect individual rights established by policy and law. Towards such requirements, a centralized registry and configuration repository provides a clean backup of all registry and configuration changes as they happen.

Corporate Data Protection

Protecting corporate data such as customer information is part of regulations, including the statewide California Database Security Breach Act, the Gramm-Leach-Bliley Act for financial services firms, and HIPAA for health care providers. These regulations are factors that drive corporations to beef up efforts to prevent unauthorized disclosure of sensitive data. In the recent past, several corporations, including large banks (Bank of America and Wachovia), have acknowledged substantial data leaks affecting tens of thousands of customers. Evidently, such leaks originate from inside the corporate office itself, often with data being transferred to or from a range of peripherals such as zip drives, CD/DVD drives and portable printers, to name a few. Furthermore, monitoring system ports and other peripheral devices can be used to block the entrance of viruses and mal-ware, and prevent corporate data from being copied or saved except where and when authorized. RegColl runs on all nodes on all the network clients and interrogates the Registry of every system attached to the corporate LAN. RegColl analyzes the registry's record of activity to identify what interfaces are being used, what peripherals are being attached to the system, and which of those devices are currently active.

RegColl's ability to examine peripheral devices connected to system allows for enforcement of data protection policies. For example, defining a corporate data protection policy, which makes it illegal to copy or move data to portable devices, can be tracked by running the policy enforcement tools. RegColl is not designed to monitor and capture use of peripheral devices; however, it can extend the functionality to incorporate the configuration changes from hardware driver installations.

The next section delineates architectural components of RegColl's framework.

Architecture

The managed entities in this system are workstations, desktop computers, and laptop systems. A process, called RegistryMonitor, runs on the managed entities to collect the registry and configuration information changes. The following sections give the details of each of the components and their interactions. We introduce the version and causal relationship mechanism of the change sets, the collection points, and the collection server components.

Architecture Overview

The RegColl framework introduces a registry and configuration information dissemination mechanism supported with a centralized repository. To ensure better availability of this information and protect against a single point of failure, RegColl introduces a secondary fail over server, which is accessible to the monitor process in the event of primary server failure.

RegColl takes into account the disconnected operations. We assume that some managed entities may enter disconnected mode when mobile. In such an event the registry monitor process collates the information locally with a causal version log. This local storage ensures that no critical changes are disregarded or missed when disconnected. Since RegColl works with causally related changes, the registry and configuration changes are reintegrated with the previously collected versions.

Since RegColl's framework collects and stores system information in a chronological manner, this archive reflects restore points that can be retrieved based on a version identifier. The causality allows for time travel and undo capabilities. Maintaining such a version relationship ensures that a disconnected entity when joining the network can seamlessly integrate the collated change information with the repository. The following subsection provides a systemic view of the RegColl's components.

Collection Points

The registry and system configuration information is collected by the *registry-monitor* process deployed as part of the registry collection framework. The registry-monitor acts as a collection source on each of the infrastructure systems' managed entity to gather registry and system configuration changes. These updates are cached when the registry keys are modified or a new sub-key is inserted or when a new node is created. The centralized framework identifies the keys that are most often updated and reflect critical modifications.

Registry and system configuration files tend to grow with time and configuration changes, such as updating new patches for operating system, security updates and driver installation. These registry logs might swell from a few kilobytes in size to multi megabytes (between 20 KB and 600 MB) as time elapses and configuration changes are made. To overcome the problem of caching and transporting heavy files, RegColl uses a *differencer* to identify changed entries and transport these deltas [1] to a registry collection server.

Often computer systems in a network, especially those in corporate networks, may be disconnected for mobility. Such a scenario mandates local caching of registry change snapshots (referred to as change-sets

in RegColl framework). To prevent the collection point of a disconnected system to make failed attempts to contact the collection server, the cache is polled before each connection attempt to the collection server. After the system is reconnected, any pending change-sets are pushed before a new snapshot is generated (Figure 1).

In order to actively observe the registry changes, monitors that have an operating system hook can provide most efficient cues on the registry changes. RegColl incorporates Microsoft's FileSystemWatcher component to notify (callback) Collection Server in the event of any registry file changes. Since registry information is critical, it is by default authenticated and encrypted using symmetric key cipher and hosted in the registry store.

Collection Server

The registry collector (primary server) acts as a collection point for registry and system configuration file entries. These entries are archived in the history server with version information, and time stamped for chronological ordering. As mentioned, the version and ordering information gives undo and time travel capabilities, since causally related entries can be queried for a particular version. The Collection Server archives the restore point for each system and maintains a mapping between the monitored system ID and the collected configuration files. The fail-over server architecture offers robustness and reliability to the overall framework. Exported registry entries are hosted on the primary registry server; in an event of server outage, the secondary server acts as backup. The primary and secondary servers synchronize maintaining the consistency in registry updates. This data consistency and synchronization is controlled using a causal history based replication framework [3].

Configuration and Registry Repository

Configuration and Registry repository provides a uniform monitoring infrastructure and is the single source of information necessary to develop the best practices for change, problem, and incident management. This source of archived registry & configuration log helps in configuration discovery & dependency mapping, and change tracking serves as the trusted information base for IT operations improvement.

The principal goal of configuration and registry repository is to tag the change-set version and store them in a multi-mapped manner while preserving their chronological order. This repository provides an accurate, always up-to-date configuration repository of the servers, infrastructure software, and dependencies between registries. Centralized registry enables system monitoring services and is a definitive source of automated, application and infrastructure configuration information.

Compliance and Analysis

A registry repository's ultimate goal is to serve as a single point of network monitoring service. The constructive contribution of a centralized registry and configuration store would depend on what tools are employed to query the repository and how frequently the repository is checked for compliance. The frequency of compliance check for policy compliance determines the effectiveness of centralizing the registry.

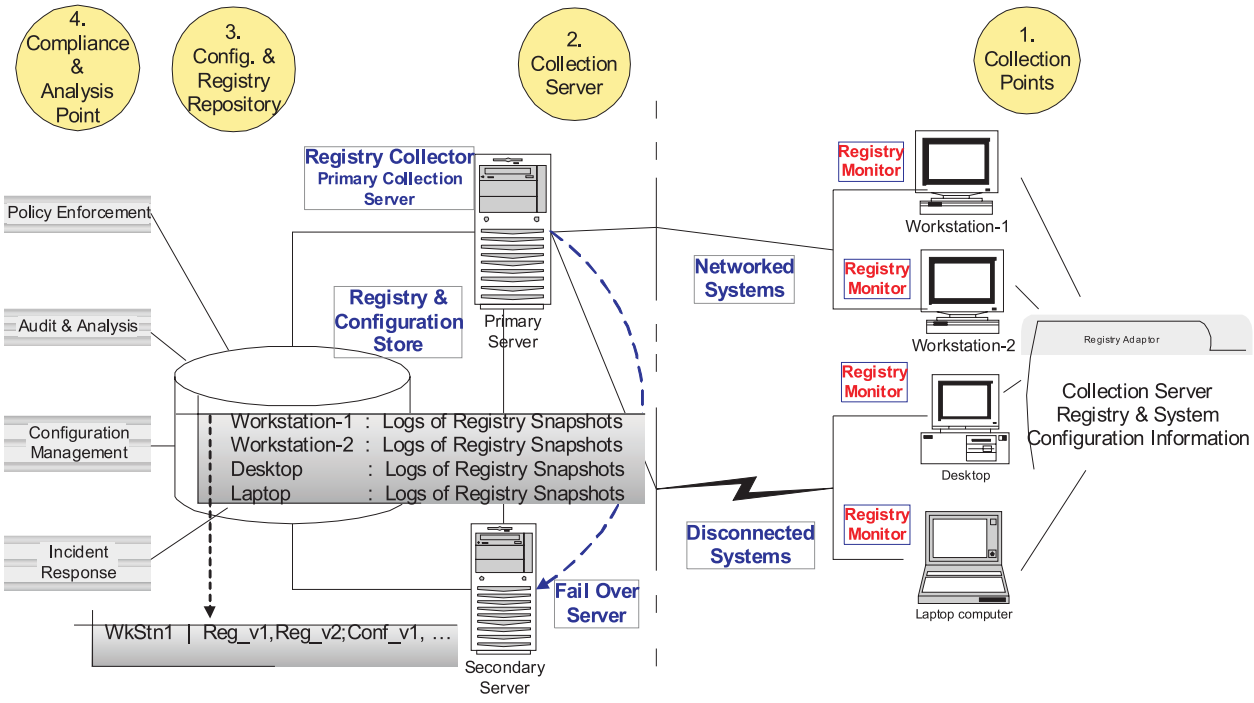

Figure 1: RegColl architecture.

Compliance checks at regular intervals would be able to respond to compliance deviation more effectively. However, a compliance check in the passive mode may be constantly looking for a pattern of malicious behavior. For example, if a policy deviation or deficiency is found, the system may trigger a host of alerts to isolate the host immediately until the host has been remediated. A host of third party software [17] can be employed to check the legislature requirements and perform system analysis.

Security Consideration in RegColl

The centralized registry framework requires entities to establish their identities in the form of public and private key pair. The creation of public-private key pair for each machine is performed as a system registration step. Registry logs that are collected as change sets (at collection points) are signed by the system/node's private key, which are authenticated to enable uploading to the collection server. Signing change sets between the collection-point (Registry Monitors) and the collection server (Registry Collector) establishes confidentiality and proper usage.

The framework enables trust in collection entities by certificate exchange. Entities are governed by their identity keys (private keys); hence, registry information shared is secured from external attacks and malicious updates. Such a trust-framework enables revoking a key in the event of a malicious change-set being generated.

Another important security aspect is the user's system privacy and invasiveness of remote monitoring tools. We contend that our framework is less invasive, since the differencer tool compares and generates deltas for each new registry change. Change set deltas enables RegColl's registry monitoring service to consume orders of magnitude less bandwidth than any other active monitoring process. Further, remote management and monitoring tools (like, Tivoli and OpenView) allow for exclusive control of the system, wherein RegColl framework's registry snapshots always has read access only. The system administrator will never have access to any other file, hence preserving user's privacy.

Registry and Configuration Change Monitoring

We explain the registry and configuration change-set generation mechanism in this section. The change capturing process requires a policy to determine the frequency of changes. We share our experience in determining these policies along with a discussion on Operating System tools used for capturing registry and configuration changes.

Registry change generation synchronizes the local registry changes of managed entities with the remote collection server. These managed entities host a registry monitoring tool called RegistryMonitor, which publishes the change-data from the managed entities based on an ''auto-publish'' policy.

The *publish operation* policy determines how often the registry changes should be published or what should trigger the synchronization. In RegColl's implementation we reviewed publish policies based on ''elapsed time'' and ''transaction size'' of the registry updates. Elapsed time mechanism assumes changes to be comparatively smaller if the time units are maintained small enough. This proved to work fine until the user tries to install large software. For example, installation of Oracle database induces numerous registry changes. We found that large installation generates so much traffic that it affects the efficiency of change-set generation process. Hence, we recommend fine-tuning the change-set generation and publish policies in the event of planned large installations.

In the implementation scenario, another challenge is determining the publish policy. The policy should be set for smaller transaction size or smaller time interval, which inversely affects the change-set size. Too fine granular publishing affects the performance and usage of the system. On the other hand too coarse grained or too many updates are found to produce very large change-set, which are detrimental in exchange operation and hence, proves to be an inefficient use of server resources.

Our experience in determining the publish policy comes from monitoring native file system changes using FileSystemWatcher component on Windows [13] and Inotify file system event monitoring mechanism [6] for Linux. FileSystemWatcher, of Microsoft .NET framework's System.IO namespace gives access to system functionalities such as the FileWatcher utility. The FileWatcher listens to the file system change notifications and raises events when a directory or file in a directory changes. This facility is utilized to monitor registry-file changes.

Differentiating between users initiated change and OS process changes are not discernible to the file watcher. For example, interaction between Windows Explorer (explorer.exe) and user workspace are reported as file changes. Further, any file creation leads to multiple file change notification.

On a Linux based system, a number of swap files are created when user programs open a file. Our experience shows that these files are created and deleted within a few milliseconds. Capturing these temporary files as part of change-set description proves to be unwanted overhead, both in terms of network traffic and change set log size. To manage such copious change notifications, we analyzed both FileWatcher and Inotify change log pattern and developed a filter routine that prunes junk and redundant entries.

RegColl follows a transactional semantics where all changes take place atomically. A time based publish policy will divide the transaction into multiple units should the installation triggered change time is greater

than that of publish time. For example, software like Tomcat web server has a longer installation and configuration time than the default publish time. We ran our experiments on University network configured to trigger notification in the events of file creation, file access and write, file name and directory name change. The size and security attributes changes are also reported. We found that the file system watcher component has certain limitations, especially in reporting what process initiated the registry-file changes.

We contend that monitoring only the registry-file changes will improve the overall performance of synchronizing the configuration files since only the registry changes are propagated to the server reducing the overall network traffic. Even with snapshotting technique, the snapshot size can be prohibitively expensive given that changes happen in number of configuration files; on the other hand, RegColl captures specific registry and configuration changes in a given time window.

Based on our experience in analyzing change notifications, we devised the staging policy where file change notifications are collected for 30 seconds and flushed in an event if no new change is detected in this time window, i.e., the inactivity window. An alternative scheme for pushing the change-set is if the queue reaches a predetermined size (e.g., 100 numbers of changes in our case). Nonetheless the publish policy is configurable.

RegColl Implementation

The Windows registry is a hierarchical structure of key and name value pairs. This structure acts as a central configuration database for users, applications, and other system information. Each key in the registry is a node in the hierarchical structure with one or more sub keys associated with it. This structure binds together configuration information in the name-value pair container associated with each node.

A "registry key" is of importance if it can reflect on configuration changes. If such keys and other system configuration information are collected at one location, it provides a centralized source of registry and configuration information that can be used for system auditing and analysis. To realize this single source of system information repository the following section gives the implementation details of the proposed framework.

Registry Monitor

The centralized registry collection framework requires network wide deployment. The current implementation bundles the collection point interface as Registry monitor batch file, regmon.jar, and regmon.properties files. RegistryMonitor batch file runs the Registry Monitor on the user's local workstation; regmon.jar has the APIs to identify registry changes

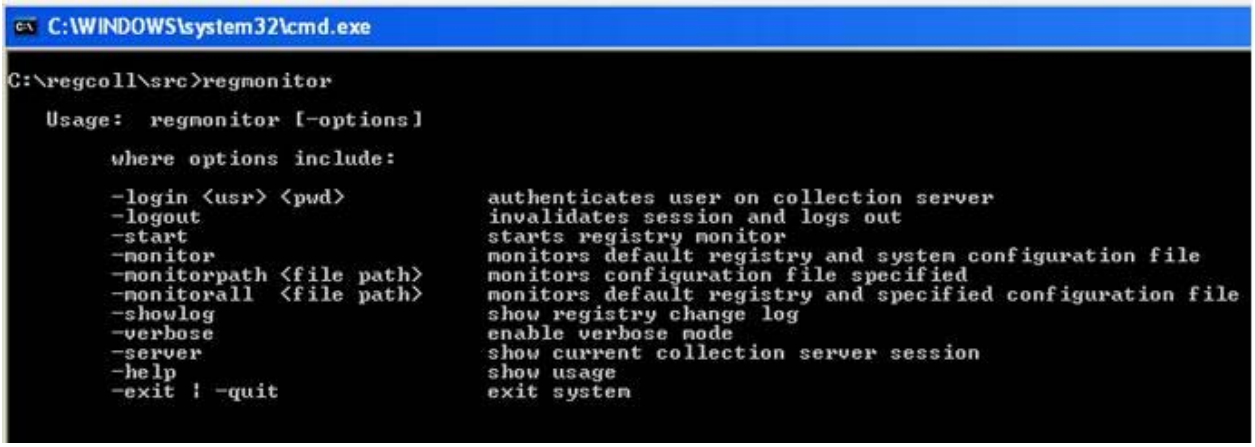

Figure 2: Registry Monitor interface options.

Figure 3: Registry Monitor with "-monitor" command.

and communication protocol implementation for collection-point and collection server communication for pushing changes. The regmon.properties file has collection server information, i.e., collection server's fully qualified domain name (FQDN) and port on which the server is listening.

To fully understand the collection framework, we deployed and tested the system on a local Windows network. The following section gives a deployment scenario of registry monitor and registry collector.

Figure 2 shows the screen shot of Registry Monitor implementation. RegistryMonitor monitors the default registry files; DEFAULT, SAM, SECURITY, SOFTWARE, SYSTEM stored in the windows\system32\config directory and NTUSER.DAT stored in the *user home* directory. Files being monitored could be seen by invoking the ''monitor'' command, as shown in Figure 3.

Installation of any new application on Windows operating system essentially modifies four registry keys, i.e., HKCR (Classes Root Handle Key), HKLM (Local Machine Handle Key), and HKU (Users Handle key).

We tested Registry monitor by installing and uninstalling a few software and drivers. For example, as shown in Figure 5 and Figure 6, Registry monitor when started with the ''-start'' command has successfully identified a change in NTUSER.DAT registry file.

Figure 6 shows the registry modification after the user installed the 'Mozilla FireFox' application. In the event of Registry monitor polling the registry, the FireFox application installation will be identified as a registry change. Registry monitor's file differencer logs the change in the change set as shown in Figure 6.

Registry Monitor Setup

Registry monitor setup entails setting up a monitor code at each of the infrastructure system (what we refer to as, collection point). Each deployment requires setting up of public and private key pairs at the collection point. These keys secure the communication channel between collection points and collection servers, all registry change sets are signed with collection point's private key. The registration process requires the system administrator to authenticate themselves before enabling deployment process. Users set up a secured password for client log on process.

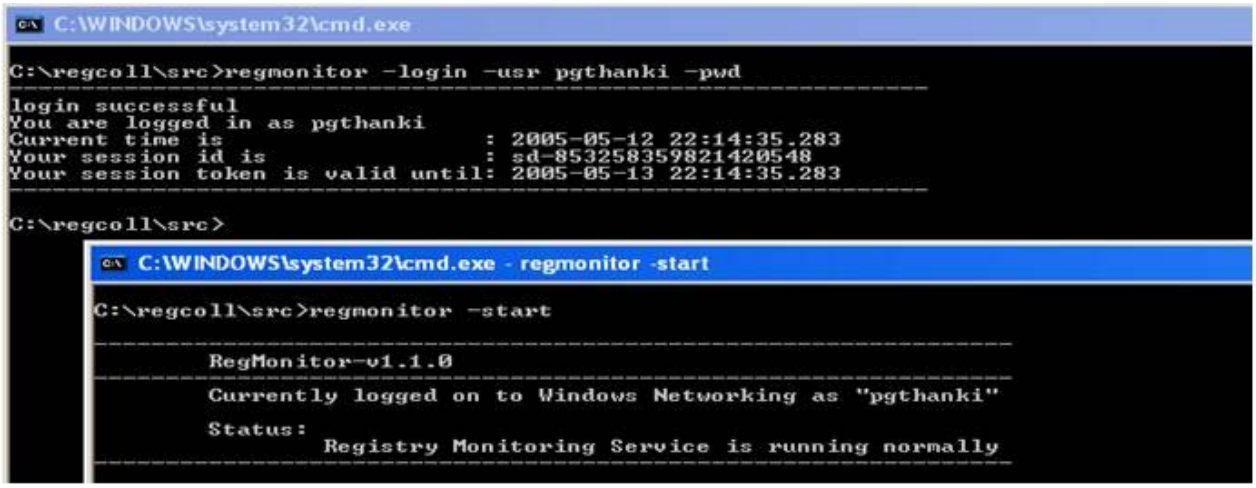

Figure 4: Registry Monitor when invoked with a ''-start'' command.

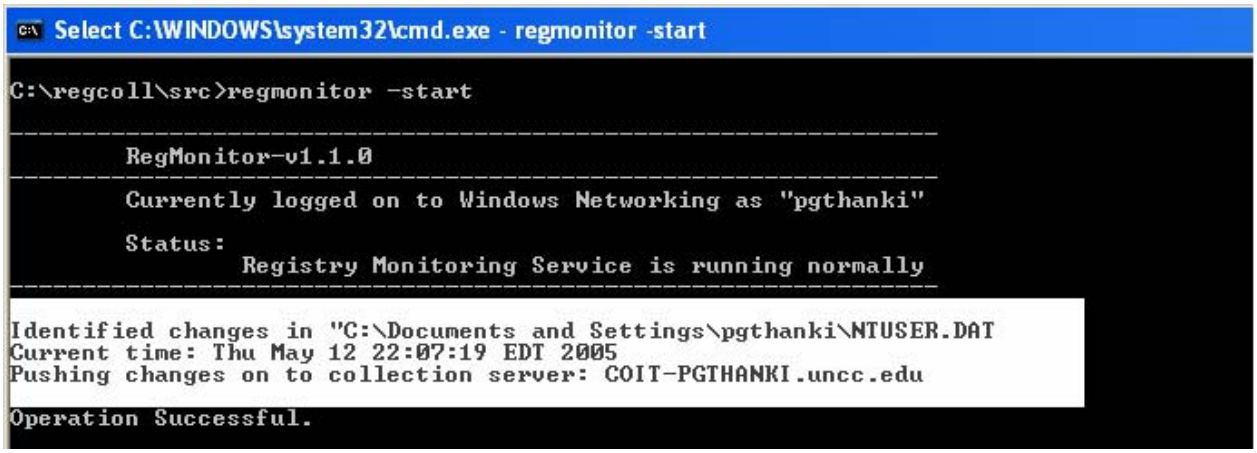

Figure 5: Identified Registry Changes using Registry Monitor.

Registry monitor requires users to authenticate with the collection server before initiating the registry monitor process. The logon process is session bound (Figure 4) and each login returns a session token from the collection server, which is usually valid for a period of 24 hours. The RegistryMonitor interface manages the session ID locally by encrypting it using a pre-established symmetric encryption key (set up during the machine registration process). User can logout of the system by calling ''-logout'' to invalidate the session token. The session token and session IDs are discarded once invalidated.

Registry Collector

The registry collector server maintains a directory structure of registry logs for each collection point (Figure 7). The registry collector captures the change-set snapshots, which are incremental delta information supplied by the collection point to reflect the registry changes. This contains the size of exchanged data at a given time since only the delta of change-sets are moved and not the whole registry or system configuration file. Moving an entire registry snapshot (taken at the time of registry change) is a process and network intensive operation. Hence, to avoid loading the network or system resources, only the change set deltas are generated and uploaded to the collection server.

These deltas give an incremental version change to the registry collection. Since the information traveling on the network is a delta of change set (or simply the changed data blocks), the information exchange is lighter and faster.

The registry collector maintains the version ordering for each change-set supplied by the collection points. These entries are multi mapped with MachineID, collection system's ID and the timestamp they are associated with. Figure-7 shows the collection servers format of storing change-set deltas.

Related Work

Monitoring system configuration information has been employed in many systems. The following are some examples. While these tools share the same idea of utilizing system configuration information as RegColl, they differ in the collection mechanism, such as disconnected mode support and its usages. RegColl collates the system configuration changes into a central location so that the centralized system configuration (e.g., registry) repository could incorporate the useful infrastructure system monitoring services such as compliance checks, incident response endorsement, and corporate policy validation. Similarly, Windows XP system restore [12] can collect registry data in its state snapshots.

The STRIDER [20] project uses differencing of periodic snapshots to reveal any configuration changes in the Windows registry. The registry keys of a failing program are monitored and recorded for analysis, this helps examine and reflect on fault in the program.

UNIX-based tools like Chronus [2] detect configuration error that might induce a faulty state in the system. Chronus captures the failure, inducing state changes to differentiate between working and non-working states. By using binary search, it can diagnose a range of common configuration errors for both client-side and server-side applications. It helps to reveal the specific failure cause, enabling recovery with minimal lost state. Another UNIX-based tool for system change monitoring is discussed in Backtracker [15]. Backtracker uses a change log mechanism and maintains an operating system causal history log. These logs are analyzed to determine the configuration changes which might be caused by the installation of an application or a computer intrusion.

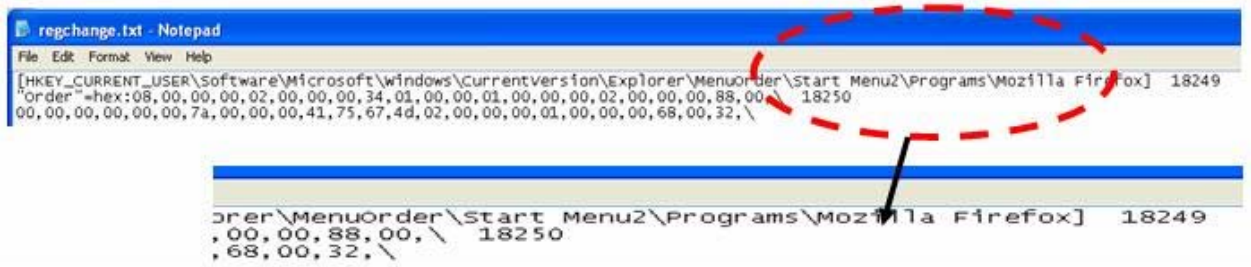

Figure 6: Changes identified by Registry Monitor.

Figure 7: Directory structure of registry collection.

Future Work

RegColl is evolving rapidly and the skeleton is in place. We would like to invite users to try this tool, which can be extended to collect other system info files such as the .ini files on Windows and .rp resource files on UNIX systems. The tool is ready for download; please send an email to bbkang@uncc.edu for registry monitor bundle and/or any further information. This bundle, along with other resources can also be downloaded from the following link: http://coitweb.uncc.edu/~bbkang/ISR/.

Conclusion

The RegColl framework is positioned as a back end collection entity that seamlessly collates the registry and system configuration changes. A fail over server adds reliability to the architecture and provides a backup for change information. Deployment of such a server would obviously be on the network but away from the internet. Further, configuration servers are patently isolated and kept secure. However, we employ security considerations that mandate setting up public & private key pair at each collection point. All change sets are signed by the machines private key.

We delineate some of the useful contributions of system monitoring services that utilize the RegColl framework.

Corporate Policy Enforcement

The collection server can be deemed as a policy enforcement point, where policies are monitored and enforced by third party tools. This central location gives a validation source for such tools; if there is a policy deviation, a policy enforcer tool will generate an alert to the system admin. Audit and configuration are just logging tools, and a configuration management tool will identify if the registry is in good shape based on which rules the system administrator decides to apply. For example, Wang, et al, proposed a tool for troubleshooting misconfigured systems [8] using registry information.

Incident Response

Centralizing registry and configuration information meets the preliminary incident response requirements of documenting and confirming an incident. If an intrusion or system compromise is detected, it will be useful to analyze the changes that happened in the course of intrusion. The incremental change information is constructive in pinpointing specific changes. A compromised system's registry and system configuration information can be compared with a previously identified valid state.

Non-Invasive Monitoring

Active scans of a system's configuration and registry information can be considered more invasive process than capturing registry snapshots. The CPU and network usage of an active monitoring system (e.g., IBM Tivoli) consume order of magnitude more resources than a prescheduled maintenance task or remote monitoring.

Capturing registry information on a timely basis makes registry monitoring a lot less invasive than remote management, since the ''differencer'' produces deltas for each new change in the registry. Registry Monitor's monitoring service consumes orders of magnitude lower bandwidth than an active monitoring process. Remote management allows full control of the system where system configuration changes are accessible for read/write operations remotely. In comparison, the RegColl framework's registry snapshots are read-only. The collection server's administrator never has access to any other files, thereby preserving the user's privacy. The system configuration information is stored in an encrypted form which adds to the overall information protection.

The single source for auditing, analysis, and policy validations capabilities gives more control and monitoring power to system administrators managing large corporate networks and infrastructure systems. Hence, we believe RegColl's centralized registry and configuration framework will be a useful tool in overall infrastructure systems management.

Acknowledgement

The authors would like to thank the following people for their contributions to improve this paper: Gautam Singaraju of Department of Software and Information System, UNC Charlotte, and Vinod Eligeti of Department of Computer Science, Virginia Tech. We take this opportunity to especially acknowledge our shepherd, Yi-Min Wang of Microsoft Research, for his invaluable support in improving this paper.

References

[1] Tridgell, A. and P. Macherras, *The Rsync Algorithm,* Technical report, TR-CS-96-05, Australian National University, http://samba.anu.edu.au/rsync/ , June, 1996.

[2] Whitaker, Andrew, Richard S. Cox, and Steven D. Gribble, ''Configuration Debugging as Search: Finding the Needle in the Haystack,'' *Proceedings of the USENIX Association OSDI '04: 6th Symposium on Operating Systems Design and Implementation*, 2004.

[3] Kang, B., Ph.D. Dissertation, *S2D2: A Framework for Scalable and Secure Optimistic Replication*, UC Berkeley, also in TechReport UCB//CSD-04-1351.

[4] Community Development Resource, Office of the Comptroller of the Currency Administrator of National Banks, http://citeseer.ist.psu.edu/377475.html .

[5] Computer Crime and Intellectual Property Section, http://www.usdoj.gov/criminal/cybercrime/cccases.html .

[6] Dow, Eli M., *Monitor Linux file system events with inotify*, IBM Linux Test and Integration Center, http://www-28.ibm.com/developerworks/linux/library/l-inotify.html?ca=dgr-lnxw07Inotify , 2005.

[7] The Federal Trade Commission (FTC) Safeguards Rule, *Financial Institutions and Customer Data: Complying with the Safeguards Rule*, http://www.ftc.gov/bcp/conline/pubs/buspubs/safeguards.htm , September, 2002.

[8] Wang, Helen J., John C. Platt, Yu Chen, Ruyun Zhang, and Yi-MinWang, *Automatic Misconfiguration Troubleshooting with PeerPressure*, Microsoft Research.

[9] ''In Brief: The Financial Privacy Requirements of the Gramm-Leach-Bliley Act,'' *Federal Trade Commission – Facts for Business*, Available http://www.ftc.gov /bcp/online/pubs .

[10] Larsson, Magnus and Ivica ''Crnkovic, Configuration Management for Component-based Systems,'' *Software Configuration Management – SCM 10*, 23rd ICSE, http://www.mrtc.mdh.se/index.phtml?choice=publications&id=0295 , May, 2001.

[11] Mercuri, R. T., ''The HIPAA-Potamus in health care data security,'' *CACM*, Vol. 47, Num. 7, pp. 25-28, http://doi.acm.org/10.1145/1005817.1005840 , July, 2004.

[12] Microsoft, Inc., *Windows XP system restore*, http://msdn.microsoft.com/library/default.asp?URL=/library/techart/windowsxpsystemrestore.htm, April, 2001.

[13] Microsoft, http://msdn.microsoft.com/library/default.asp?url=/library/enus/vbcon/html/vbconIntroductionToFileSystemComponents.asp, 2004.

[14] *Policy Enforcement tools, McAfee System Protection – McAfee Policy Orchestrator,* http://www.networkassociates.com/us/products/mcafee/mgmt_solutions/epo.htm .

[15] King, Samuel T. and Peter M. Chen, ''Back Tracking Intrusions,'' *Proceedings of the 19th Symposium on Operating System Principles (SOSP 2003)*, Bolton Landing, NY, October, 2003.

[16] Sarbanes-Oxley Act of 2002, HR 3763, PL 107-204, 116 Stat 745, United States Code, 2002, codified in sections 11, 15, 18, 28, and 29 USC.

[17] *Sygate – Policy Enforcement,* http://www.sygate.com/solutions/policy-enforcement.htm .

[18] Du, Wenliang, Aditya P. Mathur, Praerit Garg, ''Security Relevancy Analysis on the Registry of Windows NT 4.0,'' *Proceedings of the 15th Annual Computer Security Applications Conference*, December, 1999.

[19] *Windows NT Workstation Resource Kit – Windows NT Registry*, http://www.microsoft.com/resources/documentation/windowsnt/4/workstation/reskit/en-us/24_reged.mspx .

[20] Wang, Y., C. Verbowski, J. Dunagan, Y. Chen, H.J. Wang, C. Yuan, and Z. Zhang, ''STRIDER: A black-box, state based approach to change and configuration management and support,'' *Proceedings of the USENIX LISA Conference*, October, 2003.