Nicholis Bufmack

Secure Storage Servers
CS 591 Term Project Report
05/11/2007

Table of Contents

1.0 Introduction

Host based intrusion detection systems (HIDS) work by focusing their attention on the internal state of a computer system. By monitoring the system state over time and analyzing details of system activity, the HIDS hopes to detect unauthorized system activity. The result of such monitoring is to either alert an administrator that something is amiss or to provide for some sequence of actions (such as running a security script) that will allow for prevention or alleviation of the security enacting event. In the best of cases, the system's security policy will have provided protection and the action performed by the HIDS will simply be an alert detailing the successful warding of the attack. In other instances, it will be an alert with the most dire news that a system administrator can receive: a successful break-in has occurred. In such a case, the need for a solid method of disaster recovery is paramount. [1]

Recovery mechanisms employed in the aftermath of a successful break-in typically involve a few basic steps: isolation of the system, determination of the details of the break-in, mitigation of the security policy holes that provided the opportunity for the break-in, determination of the effects of the break in, and restoration of the system to the state of operation it possessed before the break-in. [1] It is the last of these that will be the focus of this project.

The technique behind disaster recovery is typically to create a database containing details of file and system objects that should be monitored. In the case of file system objects, the HIDS creates a checksum or MD5 (or similar) hash that can be used to uniquely identify the contents of the file. In effect, a digital signature of the file is created. When coupled with other file information, such as creation date, last access time, etc., a snapshot is generated that can be compared with another snapshot generated at a later date to see if the file has been

modified or altered from a previous version. [1] Note that his does not provide for an absolutely foolproof method for guaranteeing that a file has not been modified by an unauthorized user: in particular, it has been recently shown that the there is more than an insignificant possibility that an intruder can modify a file and still generate a previous digital signature. [2]

In operation, the disaster recovery scenario is as follows: Lock the system and the file object database, scan the file system, create new checksums, compare the checksums to the original file system object information contained in the file system object database, restore or remove any modified files, update the database, and then release the system and file locks. [1]

As one would expect, this is anything but a fast, easy process. Simply generating the checksum information can be extremely slow process, especially when one considers a network file server with many users, each with their own home directory containing numerous files. [1,2] While improvements in digest generation algorithms can certainly speed up this process, as can improvements in hardware disk access, there still remains a limitation in how much improvement can be made via this route. What is needed is another methodology for disaster recovery.

2.0 Problem Statement

Despite the method employed, disaster recovery systems suffer from a few core problems. The three that shall be the focus of this project are as follows:

- The time needed to scan the system's files and generate file system object signatures can be time expensive. If the scan is done real-time, there will be significant overhead, diminishing the responsiveness of the system. [1]
- The system being scanned must be under some level of access restriction while the file system scan is being performed. This may be a complete system lock such that no action can be performed (as would be necessary if a restore is being performed) or a file level lock if the scan is being performed on a single file in real-time. [1]
- The file system object signature database and the archive restoration store must be secured. If unauthorized access is gained to either, then the entire system's role has been destroyed. In such a case, system recovery becomes impossible. [1]

The proposed problem to be solved by this project is then to develop a real-time disaster recovery system that provides for:

1. Minimal impact on overall system performance.
2. Minimal file scanning and signature generation time.
3. Minimized or non-existent file and system object locking restrictions.
4. Maximizes the security of the file system object signature database and restoration archive store.

3.0 Proposed Solution

The general solution proposed by this project is to leverage a distributed file system, the file replication service associated with distributing the files between servers, and a restore point service that will allow for selective archiving and restoration of file system objects. Additionally, a separate storage server, functioning on a one-way replication channel, will provide secure storage services for the file system object database and for the file system restoration archives.

Specifically, the solution will function under very specific software constraints. The constraints, while seemingly confining the solution to a single platform, are not especially limiting. A recent report has indicated that Microsoft Windows Server technology will dominate the server market by 2008. [3]

The platform under which the solution is to run includes:

- Microsoft Windows Server 2003 Release Candidate 3
- Microsoft NTFS (NT File System)
- Microsoft DFS (Distributed File System)
- Microsoft VSS (Volume Shadow Service)
- Microsoft SQL Server 2005
- Microsoft WMI Scripting using the Windows scripting language
- Microsoft ADS (Active Directory Service)

In summary, the solution will provide for a Windows domain composed of a primary domain and one or more file servers. The individual machines will provide for the NTFS file system. Underpinning the domain will be DFS, providing for two way replication between the various file servers, ensuring homogeneity between replicated volumes.

Separate from the file servers, a secure storage server running VSS will be the target of one-way file replication. File signatures will be stored in the SQL Server and functionalized by a WMI interface written in the Windows scripting language.

Hardware necessary is the minimal needed to run the core software packages and an Ethernet LAN providing network connectivity. Since virtualization could provide for separation of the servers on the same machine, it may not be necessary to have redundant physical machines. In this solution, it is not specified as to whether virtualization is used.

Under this solution, the secure storage server described will provide for real-time, minimal impact, secure disaster recovery services for a Microsoft Windows domain. Exact details follow in the next section, as do expected benefits of the solution, and expected limitations.

4.0 Functionality

A server is prepared with Windows 2003 Server R3. The server is setup as a domain controller controlling a single domain on an Ethernet LAN; it will be referred to as DC. Three additional servers are prepared with the same operating system and added to the domain. Two of the servers, referred to hereafter as FS1 and FS2, respectively, will be file servers. The other server, which we will call SSS, will be the secure storage server.

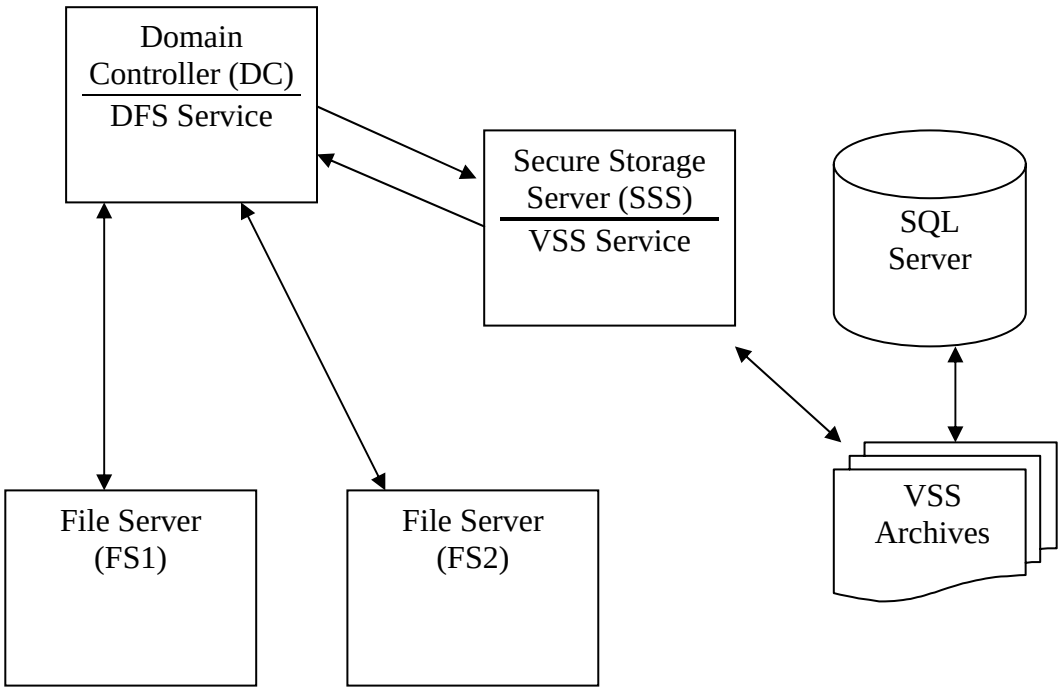
SSS is setup to run VSS and SQL Server. Additionally, the WMI scripting that will be done, will run on this server, interfacing with Windows objects existing only on this server and executing stored procedures created within the SQL Server.

DC is configured to run DFS. Replication is two way between FS1, FS2, and DC. This is to provide for homogeneity between files on FS1, FS2, and DC. One-way replication is setup between DC and SSS; the replication is initially configured to replicate *to* SSS. This will provide for a secure replication channel to the secure storage server.

On SSS, VSS is setup to backup the replication share that is the target of the one-way replication between DC and SSS. Granularity is set at the file level. VSS is setup to backup file system objects whenever a change has been made. The WMI scripting package is coded to intercept the messages from VSS, extract the file signature, contact the SQL Server, and run a stored procedure that will insert the information into the appropriate database and tables. Additional scripting will provide for the reverse process: stored procedures can be run in the database, returning file signatures fulfilling a specific criteria (such as when changed), and the appropriate files can be restored from the VSS archives.

Figure 1 below shows the topology of the proposed solution.

Figure 1: Topology



The solution described above consists of two states: before recovery and recovery. The topology of the system remains the same no matter which state the system is in with a single exception: the direction of replication under which DFS operates is changed. Details of each state are discussed in the following paragraphs.

In the before recovery state, the system is in normal use. Users access their files and run their programs. In functionality, the various files on FS1 and FS2 included in the replication rules on DC are replicated between DC, FS1, and FS2 two-way: no matter what server is connected to, the same files are on each file server. This is accomplished via the file replication service underpinning the DFS service.

At the same time, files are replicate one-way to SSS. When an event is generated on SSS indicating that a file has been changed, VSS is triggered to create a backup of the new replicated file, and the WMI interface script triggers the appropriate stored procedure in the database. This process occurs in real-time without any action or management being necessary from the system administrator. Additionally, the backup is not incremental: each backup of a file system object is a new backup object and does not overwrite a previous backup.

When an intrusion event has occurred and a restoration is deemed appropriate, the system enters the recovery state. During this state, some normal services are suspended temporarily: for example, users can not access their home directory files, but may be able to use print services, depending on the degree of penetration and the judgment of the system administrator. In any event, DFS replication becomes one-way from DC to FS1 and FS2. The direction of replication is also reversed between DC and SSS; replication now occurs from SSS to DC. Next, the WMI interface script accesses the SQL server database to find the signatures of the VSS file backups that match the restoration criteria and initiates a restoration

of the appropriate files. Since the direction of the replication has been altered in the way mentioned above, the new files will replicate back into the domain. After replication has finished, the DFS replication flows are restored and the system enters the before recovery state.

5.0 Benefits

Specific benefits should be realized from the proposed solution to the problem, each of which address the aforementioned problem statement. Specifically, these are:

- The solution provides for real-time file signature generation and backup creation without any negative impact on the core domain servers or their functionality. All processing occurs on the secure storage server, not on servers providing services to users of the system.
- Because of efficiencies inherent in Microsoft's DFS, bandwidth utilization is minimized during scanning, backup, and restoration processes [4,5].
- In addition to the security provided for by the one-way replication to the secure storage server, the secure storage server can be put on a separate subnet behind a firewall.
- During restoration, it is not necessary to make the entire system unavailable. Depending on the specific situation, it may only be necessary to restore files found to be altered, leaving the other files and services available to system users. In any event, all processing involved in restoration is done only on the secure storage server.
- The entire system is automated. The only component requiring human interaction is the determination of what criteria to invoke during a restoration.

Other benefits may certainly exist and my become evident during an implementation of the solution.

6.0 Limitations

Certain limitations exist in this system. Limitations other than these may be found during the implementation of the solution.

- The solution will only work on the platform indicated. It also will not work on stand-alone systems. An ADS domain is required. [5]
- The solution will not work with non-Windows operating systems, such as Linux' ext3. [5]
- The solution will not work with some Windows operating systems, such as FAT and FAT32. [5]
- It is not clear as to the effect of the solution viz. a viz. the preservation of file system metadata such as alternate data streams. [4]

7.0 Future Work

The obvious task remaining to be done is to actually implement the system. Implementation should be on geographically isolated servers and use as many file servers as possible. Additionally, a large amount of files should be in each DFS replication share. Metrics should be devised, such as impact on system responsiveness and bandwidth impact, that can measure the true impact of the solution. Finally, a simulated intrusion event should be created to test the overall functionality of the system.

8.0 References

[1] *Host Based Intrusion Detection Systems.* Retrieved April 1, 2007 from Wikipedia website. http://en.wikipedia.org/wiki/Host-based_intrusion_detection_system

[2] *MD5*. Retrieved May 4, 2007 from Wikipedia website. http://en.wikipedia.org/wiki/Md5

[3] *Windows Server to Own 60% Market Share by 2008.* Retrieved April 26,2007 from Win Insider website. http://www.wininsider.com/news/?7124

[4] Windows 2003 Support Site. Retrieved April 2, 2007 from Microsoft TechNet website. http://support.microsoft.com/?scid=ph;en-us;3198

[5] Windows 2003 Server Technology. Retrieved April 2, 2007 from Microsoft TechNet. http://technet2.microsoft.com/windowsserver