

Design issues of 'Vulnerabilities and Suspicious behavior detection system' in Storage Area Network (SAN)

Mr. Sandeep Abhang¹, Mr. Sachin Deshmukh², Dr. Ulhas Shinde³, Dr. Ratnadeep Deshmukh⁴
 M I T college of Engineering¹, S. P. Women's Engineering college³, Department of CS and IT, Dr. B. A. M. University^{2,4},
 Aurangabad (MS) - 431001, India.
 sandeep_abhang@yahoo.com¹, Ratnadeep_deshmukh@yahoo.co.in³

Abstract: Current accelerated demand of business continuity of operations in 24×7 work environment has increased the importance and amount of the data stored on computers and Storage Area Networks (SAN). A lot of security issues are pending related to the performance and ability of such high-speed network architecture. There is tremendous risk in the available intrusion detection systems and the security issues are remaining pending. This paper takes a review of existing available IDSs and describes various important design issues and some suspicious activities, Vulnerabilities in to the SAN. In the last section of the paper we represented important useful conclusions on above review for future design of IDSs for SAN.

Keywords Intrusion detection, SAN, SVC, IDS, WDM, RAID, FC,

1. Introduction to Storage Area Networks (SAN)

Storage Area Network (SAN) is made-up of a high speed Network for connecting data servers to storage devices, a management layer for setting up and maintaining connection between these server and Storage Devices (SDs) [1]. It contains three type of component: The Servers, Storage Volume Controllers (SVC) and SDs. Where storage is not directly connected to network clients and even not directly connected to the servers. All SDs are interconnected. The figure 1 shows the basic architecture and require infrastructure for the SAN, Servers are connected to SAN in order to serve client requests and to access data that are located at the Storage Devices (SD). SDs includes heterogeneous storage devices such as optical disks, RAID arrays, tape backups, which are effective for storing large amounts of information and backing up data online in e-commerce, online transaction processing, electronic vaulting, data warehousing, data mining, multimedia Internet / intranet browsing, and enterprise database managing applications. The SVC is the main component that manages the server and SDs interactions.

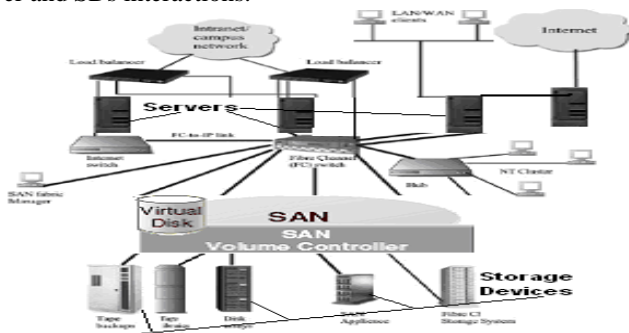


Figure 1. Storage Area Network

It virtualizes SDs into a common pool and allocates storage to host systems from that common pool. By analyzing an interaction scenario between the different system entities (client, server, SVC and SDs), the SVC should maintain a set

of structures in order to serve incoming requests [2]. There are mainly three structures that should be available and managed at each SVC, a mapping table that holds for each entry the transaction and its originator, a table that holds for each entry of the server and its associated SDs, that are logically attached to it and a structure that ensures the mapping of each high level resource (e.g. file, directory) to its physical allocation at the SD (e.g. File allocation Table). We found the two major problems with the existing design of the SAN i.e. Intrusions in side the SAN and

1. Disaster recovery. Due to the intrusions number of suspicious activities can grow and attacks are made on the SAN like, several common intruder actions, such as adding backdoors, inserting Trojan horses, tampering with audit log, LUN masking attacks, Zone hopping, Session hijacking, Man in the middle attacks. Another problem is that the SANs are designed to work within campus or limited distance. In the recent past natural disasters like earthquakes, fires, floods, power outages and manmade disasters like terrorist attacks, human errors have increased which leads to severally destructive in limited distance environment. Data center at National / International level requires high level of security. Need of extending SAN over large distance has become essential to protect data against loss or damage. For disaster recovery the network topology with high-performance solutions like iSCSI, Wavelength Division Multiplexing (WDM) transport networks are upcoming. But due to the intrusions dealing with suspicious activity and Vulnerabilities became very hectic and essential as number of attackers increasing day by day.

2. Mathematical Modeling of SAN

2.1 Mathematical Modeling of transaction of SAN

In [1] a mathematical model for the Transaction in the SAN is described. A task that is supported by an SD may be modelled as: $ts_j = fs_j(o_1, \dots, o_n)$, where $o_1 \dots o_n$ are the n elementary requests that are supported by an SD s_j and fs_j represents the function that processes these requests. A transaction is a set of tasks that are interrelated and that may be represented as follows: $T = \{t_1, t_2, \dots, t_s\}$, T where the relation T defines an ordering on the set of tasks. The processing of an elementary request moves the SD from an input state s_{in} to an output state s_{out} that both belong to the set of possible states denoted as S . S is composed of two sub sets: SL and SM . SL represents the set of states that are legitimate for an SD. SM is the set of states that are considered malicious for an SD.

The latter is composed of two sub sets that are the following: ST and SF . This classification is due to the fact that the processing of malicious requests may lead the SD to a failure state or moves it to another malicious state. In this way, ST represents the set of tolerated states and SF is the set of failure states The processing of an elementary request o that belongs to a task is tolerated if the following conditions are satisfied: sino 2 ST and souto 2 ST . The end of the intrusion tolerance process is achieved if : sino 2 ST and souto 2 SF . A transaction is considered malicious if the following condition is satisfied : 9 t 2 T and o 2 t | souto 2 ST .The end of the tolerance process is achieved if the following condition is true: 9 t 2 T and o 2 t | souto 2 SF . The decision is modelled as follows: DT = {dt1 , ..., dtm} and dt1 = {_o1 , ..., _on}. As illustrated by the aforementioned conditions, DT is determined based on {dti} where i = 1, ..., n (n is the number of tasks that belong to a transaction T). In a recursive manner, each dti is determined based on the set of decisions associated to elementary requests {_oi}. From this representation, we can deduce that the set of decisions associated to elementary requests has a direct effect on task decisions and indirectly are considered in the elaboration of the global decision DT.

2.1.1 Performance Metrics

The performance of the SAN is calculated by following metrics under symmetrical traffic.

Average throughput per SAN node.

Queuing delay, Recovery point Objective (RPO), Recovery time Objective (RTO) ,Network Rate ,Total Storage Area Network throughput. The performance of the WDM unidirectional fiber ring is calculated as follows , Latency = propagation + transmit

- Transmit = size / bandwidth , Propagation = distance / Speed of light ,Transfer time = RTT+1 / (bandwidth * transfer size) ,Throughput = Network load / Transfer time

Normalized Network load: It is the ratio of the sum of all nodes transmission rates to the total network transmission capacity. At the load of 1, the summation of all nodes transmission rates reaches the total network transmission capacity.

The network load can calculated by following formula,

$$L = \sum_{i=1}^R \sum_{j=1}^N (R_{nij} * U_{ij}) R_w * N_w$$

Where,

R – No. of the ring, N- No. of the nodes, R_{nij} –transmission rate of node j on ring I ,U_{ij} –Utilization of node j for ring I , R_w -Wavelength rate, N_w -Number of wavelength,

3. Suspicious behavior and Intrusions in SAN

The mostly the suspicious contents are the viruses, detectable via its signature. Viruses cannot disable the internal scanners of servers even after infecting clients. Two other examples of suspicious content are large numbers of “hidden” files or empty files. Hidden files have names that are not displayed by normal directory listing interfaces and their use may indicate that an intruder is using the system as a storage repository, perhaps for illicit or pirated content [3]. A large number of empty files or directories may indicate an attempt to exploit a

race condition by inducing a time-consuming directory listing, search, or removal.

3.1 Vulnerabilities

Attackers obtain access to a machine by exploiting vulnerability in a remotely accessible service. Exploiting vulnerability usually involves overflowing an unchecked buffer or similar programming error to run arbitrary code of the attacker’s choice on the victim machine [4]. If the exploited service is not privileged, the attacker may then attempt to leverage a flaw in the operating system of the machine to escalate his privilege.

3.2 Attack model

Most of the attacks are automated attacks resulting from self-propagating worms. These worms spread in an autonomous fashion and do not need to create interactive sessions. Nevertheless, not every attack is automated. Some attackers use semi-automated attacks: they first scan large IP address spaces in search of a machine running a service with a vulnerability they believe they can exploit. Once they have found such a machine, they use an exploit to execute code of their choosing on the machine. This code will often create a backdoor that will enable the attacker to interactively access the machine. He can be moved by mischief or fun, in which case once he has compromised the machine, he may tamper with it. Once attackers have compromised SAN, they want to retain the ability to access it in the future to use it for the aforementioned purposes. As a result, they will try to remain undetected for as long as possible. An attacker who is able to gain super user status on a machine may also modify the operating system. If no mechanism to protect the integrity of the kernel is present, he may for example load kernel modules to change the structure of the operating system and hide services he may be running.

3.2.1 Shellcode. Shellcode is difficult to construct, and has several major restrictions placed on it. The size of the shellcode should be small, as it has to fit within the buffer being attacked. Because the attacker cannot predict the exact memory location where the buffer will exist, the ‘shellcode’ must also be position independent. Moreover, the shellcode is often injected in buffers that are processed by functions that handle strings, such as the standard C library function strcpy. In such cases, the shellcode cannot contain characters that will be regarded as ending the string by the processing function. Shellcodes are often not engineered to exploit a specific vulnerability. Even if they are, they can often be slightly modified to fit another situation. As a result, they are heavily reused by attackers and there are far fewer variations of shellcode existing in the wild than there are exploitable vulnerabilities.

3.2.2 Backdoors

Generally Shellcodes are designed to create a primary backdoor, by which the attacker can obtain the interactive session and he may execute more sophisticated commands on SAN. There are two kinds of backdoors.

3.2.2.1 Primary Backdoors

In [4] only four different types of primary backdoors are discussed that are shellcodes create: user creation, bindshell, connect-back shell and findsocket shell. These backdoors

leverage relatively simple mechanisms. User Creation type of primary backdoor involves creating a privileged account on the machine with a user name and password known to the attacker. The attacker can then gain access to this machine by logging in as the user through regular channels. This requires modifying access control files for the system. On a Linux system, this means modifying the system wide password file `/etc/passwd` or shadowed password files, `/etc/shadow` also needs to be modified. Alternatively, service specific mechanisms such as ssh key files can be manipulated. Bindshell is another method involves spawning a root shell and binding its input and output streams to a socket listening on a port of the attacker's choosing. The attacker then uses a program such as telnet to connect to the port to communicate with the shell. They refer to this technique as bindshell, named after some functions that we found doing this in various exploits that they have analyzed. Authors also note that the bindshell could either be created by the shellcode directly, or by having the shellcode modify the configuration of TCP wrapper services such as `inetd` or `xinetd` to spawn such a shell. Yet a third method involves initiating a connection back to the attacker's machine, and then starting a shell with its input and output streams tied to the new connection. This has the advantage that it will bypass firewalls that block incoming connections to unauthorized ports, but will allow outgoing connections that are initiated from within the trusted network. Finally a fourth method cleverly reuses the connection the vulnerable service had been using and connects the shell's input and output streams to that connection. This method is very similar to the bindshell, with the exception that no new socket is created. Instead, the shellcode simply searches for the existing socket and uses that instead. Since the traffic now flows over the same connection that the attack arrived on, the interactive session is able to bypass any firewall defenses, and would generally not trigger any detection systems that scan for new network flows. They refer to this backdoor technique as a find-socket shell, since the shellcode has to find the existing socket to create the backdoor.

4. Responding to intrusions

Storage IDS should operate in a way that will not interfere with a valid, running system. Since a detected "intruder action" may actually be legitimate user activity (i.e., a false alarm), our default response is simply to send an alert to the administrative system or the designated alert log file [8]. There are, however, more active responses that a storage IDS could trigger upon detecting suspicious activity. When choosing the proper response, the administrator must weigh the benefits of an active response against the inconvenience and potential damage caused by false alarms. One reasonable active response is to slow down the suspected intruder's storage accesses. For example, a storage device could wait until the alert is acknowledged before completing the suspicious request. It could also artificially increase request latencies for a client or user that is suspected of foul play. Doing so would provide increased time for a more thorough response, and, while it will cause some annoyance in false alarm situations, it is unlikely to cause damage. The device

could even deny a request entirely if it violates one of the rules, although this response to a false alarm could cause damage and/or application failure. Liu, et al. proposed a more radical response to detected intrusions: isolating intruders, via versioning, at the file system level [10]. To do so, the file system forks the version trees to sandbox suspicious users until the administrator verifies the legitimacy of their actions. Unfortunately, such forking is likely to interfere with system operation, unless the intrusion detection mechanism yields no false alarms. Specifically, since suspected users modify different versions of files from regular users, the system faces a difficult reintegration problem, should the updates be judged legitimate. Still, it is interesting to consider embedding this approach, together with a storage IDS, into storage systems for particularly sensitive environments. A less intrusive storage-embedded response is to start versioning all data and auditing all storage requests when an intrusion is detected. Doing so provides the administrator with significant information for post-intrusion diagnosis and recovery. Of course, some intrusion-related information will likely be lost unless the intrusion is detected immediately, which is why Strunk 6 et al. argue for always doing these things (just in case). Still, IDS-triggered employment of this functionality presents a useful trade-off point.

5. Intrusion Detection systems:

Intrusion Detection is performed for the most proposed solutions at the system and network levels. Few works have addressed security issues in the SAN environments especially in the intrusion detection field.

Related study:

In [2] the novel approaches for storage based intrusion detection are discussed, how features of state-of-the-art block storage systems can be used for intrusion detection and recovery of compromised data. They present two prototype systems. First they present a real time intrusion detection system (IDS), which has been integrated within a storage management and virtualization system. In this system incoming requests for storage blocks are examined for signs of intrusions in real time. The intrusion detection schemes can be deployed as an appliance loosely coupled with a SAN storage system. The major advantage of this approach is that it does not require any modification and enhancement to the storage system software. In this approach, they use the space and time efficient point-in-time copy operation provided by SAN storage devices. Authors also present performance results showing that the impact of ID on the overall storage system performance is negligible. According to this approach, detection is performed at the storage level and integrated within a storage management and virtualization system. SIDSs ensure detection even if the system is compromised but detection efficiency depends on the set of rules that define attack signatures. In this context,

In [3] Cooperative Intrusion Detection and Tolerance System (CIDTS) is proposed. CIDTS detects and tolerates attacks by the cooperation of several components at the network, host and disk level. It contains a component, called the system call interceptor that ensures cooperation at the host level and introduces modifications to storage requests before

forwarding them to the disk. These modifications consist in introducing information about the process associated to the storage request and the decision made by cooperating host and network level detection components. Among these components, authors have mention the request handler that handles incoming storage requests after checking them against a set of rules by another component referred to as the violation rules monitor. The rules and generated alerts are stored in a special area of the disk that is called the protected area.

In [4] Intrusion Detection and Tolerance System (ISIS) is presented. The proposed solution is based on: The management of two areas (virtual area and protected area) at each storage node, the cooperation of detection modules running on each SAN component and the use of distributed set of rules that are updated and managed in a secure manner ISIS attempts to detect intrusions made by a remote attacker—someone who should not have access to the system under attack. ISIS makes no assumption on how an attacker is able to execute code on the vulnerable machine.

In [12] numbers of specific warning signs visible at the storage interface are discussed. Examination of 18 real intrusion tools reveals that most can be detected based on their changes to stored files. They describe and evaluate a prototype storage IDS, embedded in an NFS server, to demonstrate both feasibility and efficiency of storage-based intrusion detection.

In [17] how self-securing storage improves intrusion survival by safeguarding stored data and providing new information regarding storage activities before, during, and after the intrusion are discussed. Specifically, it focuses on three ways that self-securing storage contributes: helping to more quickly detect intrusions, providing easily accessible information for diagnosing intrusions, and simplifying and speeding up post-intrusion recovery. First, a self-securing storage server can assist with intrusion detection by watching for suspicious storage activity. By design, a storage server sees all requests and stored data, so it can issue alerts about suspicious storage activity as it happens.

5.1 Design issues for intrusion detection system:

Followings are the major issues while designing the IDSs

5.1.1. Intrusion Prevention

Generating security alerts time to time is very important for Prevention.

For Prevention from Intrusion the rules can be develop for keep watching following attributes

For metadata:

Inode modification time	Data modification time	Access time
File permissions	File owner	File group
Link count	Device number	Inode number
File type	File size	File names

For data:

5.1.2. Intrusion detection diagnosis

Once an intrusion is detected and stopped, the administrator interested in what happened, determining how the intruder

gained access to the system, when they gained access, and what they did after got in. in some systems administrators usually perform only a cursory review of the post-intrusion system, hoping that the intruder overlooked some obviously incriminating evidence. The administrator who wishes to dig deeper into the system in search of answers to these questions is faced with a daunting task. First, he must scour the free space of the storage system in search of disk blocks from deleted data and log files that have not yet been overwritten.

5.1.3. Action on Intrusion

Storage-based intrusion detection can quickly and easily notice several common intruder actions, such as adding backdoors, tampering with audit log contents. Such activities are exposed to the storage system even when the client system’s OS is compromised. Second, In current systems, little information is available for estimating damage and determining how he gained access. Because intruders can directly manipulate data and metadata in conventional systems, they can remove or obfuscate traces of such activity.

5.1.4. Report after Intrusion

The report about each and every Intrusion entry or action if any must be done all the time. The report must contain the nature of intrusion, harms of it, Vulnerabilities, location, cause of it, solutions to overcome.

5.1.5. Data preservation

In available current systems, the mechanics of Data preservation involves finding separate media on which temporarily store the users’ data, then performing the normal reformat, reinstall, restore, and finally restoring this recent version of the users’ data. There also exists the challenge of determining whether this data is safe to keep at all since the intruder could have modified it.

5.1.6. Recovering data

Without examining the contents of files, it is difficult to determine what application level change was made. it would be possible , in some situations, untangle changes made by a legitimate user and an intruder within a single file. For instance, the intruder may have planted a macro virus within a word processing document. A tool that understands the format of such files could remove the virus while leaving the other data intact. Modern virus detectors are able to handle this specific case, but in general, this sounds like a daunting task—creating application-specific recovery tools for data files. However, a small number of applications create most files in a given system. Therefore, a few utilities will cover most of the data. An additional use for these application-specific recovery tools is for bootstrapping recovery and diagnosis of complex programs, such as databases. Consider a database application that maintains a log of operations, capable of undoing or redoing its operations. A recovery tool could potentially validate the contents of the database against the database’s log, ensuring that any changes made to the database contents were made through the proper interface. The database’s built-in validation and auditing could then be

Appends in the files	Any modifications
----------------------	-------------------

used to dig deeper into the changes.

6. Conclusions:

Managing a SAN is still a major challenge there is tremendous risk in the available intrusion detection systems and do not apply efficiently to the SAN environments due to the use of static rule and lack of cooperation between detection modules. The security issues are remains an investigated topic and seeking for solution.

1. Storage-based intrusion detection is not free. Checking rules comes with some cost in processing and memory resources and more rules require more resources. In configuring a storage IDS, some thing must balance detection efforts with performance costs for the particular operating environment.

2. Even if the CIDTS is deployed in the SVC and storage devices, it is not suited for SAN environments because detection is performed in an autonomous manner without cooperating with the remaining CIDTSs. Therefore, additional processing should be introduced for host level and disk side components in order to ensure detection in SAN environments.

3. Storage IDS can produce some false positives. Such as "watch these 100 files for any modification," they will occur only when there are legitimate changes to a watched file, which should be easily verified if updates involve a careful procedure.

4. The general anomaly detection is also a hectic job and like any IDS, a storage IDS will fail to spot some intrusions.

5. Storage IDS cannot notice intrusions whose actions do not cause odd storage behavior.

Using network-based and host-based IDSs together with storage IDS, can increase the odds of spotting various forms of intrusion.

6. Storage IDSs embedded within individual components of decentralized storage systems are unlikely to be effective. For example, a disk array controller is a fine place for storage-based intrusion detection, but individual disks behind software striping are not. Each of the disks has only part of the file system's state, making it difficult to check non-trivial rules without adding new inter-device communication paths.

7. In current systems, the administrator is poorly equipped to understand the intrusions action because, once an intruder gains control of the computer system, no information can be trusted; the intruder has the ability to erase and obfuscate incriminating evidence

References:

- [1] Yacine Djemaiel, Noureddine Boudriga. Dynamic detection and tolerance of attacks in Storage Area Networks 22nd IEEE International Conference on Advanced Information Networking and Applications, IEEE Computer Society, pages 377-380, March 2008, 978-0-7695-3096-3/08/2008 IEEE
- [2] M. Banikazemi, D. Poff, and B. Abali. Storage-based intrusion detection for storage area networks (sans). In Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05), pages 118-127, Washington, DC, USA, 2005. IEEE Computer Society.
- [3] Y. Djemaiel, S. Rekhis, and N. Boudriga. Cooperative Intrusion Detection and Tolerance System. In Proceedings of 12th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2005), pages 279-283, Tunis, Tunisia, December 2005.
- [4] Lionel Litty. Hypervisor-based Intrusion Detection, Graduate Department of Computer Science University of Toronto 2005
- [5] Vladimir V. Rabov 'Storage Area Networks (SAN) WLO40-sample.cls April 3,2003

[6] A. G. Pennington, J. D. Stunk, J. L. Griffin, C. A. Soules, G. R. Goodson, and G. R. Ganger. Storage based intrusion detection: Watching storage activity for suspicious behavior. In Proceedings of the 12th USENIX Security Symposium, 2003.

[7] 'Storage Area Network (SAN) based on WDM for Disaster recovery operation', proceedings of CESSE 2007, WASET August 2007, Berlin Germany.

[8] J. S. Glider, C. F. Fuente, and W. J. Scales. The software architecture of a san storage control system. IBM Syst. J., 42(2):232-249, 2003.

[9] "IBM system storage SAN volume Controller" IBM's Redbook SG24-6423-05, pp. 12.

[10] Z. Xiao and L. Zhanhuai. Research on security of storage area network. In Proceedings of InfoSecu04, pages 238-239, Shanghai, China, 2004.

[12] Adam G. Pennington, John D. Strunk, John Linwood Griffin, Storage-based Intrusion Detection Watching storage activity for suspicious behavior Oct 2002 School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213

[13] Taisir E. H. El-Gorashi, Bernardi Pranggono, and Jaafar M. H. Elmighani 'WDM Metropolitan Sectioned Ring For Storage Area Networks Extension with Symmetrical and Asymmetrical Traffic'. IEEE ICC 2006 Proceedings.

[14] Taisir E.H. El-Gorashi, Bernardi Pranggono, Jaafar M.H. Elmighani, 'Multi-Wavelength Metro WDM Sectioned Ring for SAN Extension under Hot Node Scenario and Variable Traffic Profiles' 1-4244-0236-0/06/ 2006 IEEE, ICTON 2006

[15] Bernardi Pranggono, Member, IEEE, Jaafar M.H. Elmighani, Senior Member, IEEE 'Performance Evaluation of INSTANT - A Metro WDM SAN under Balanced and Unbalanced Traffic Conditions' 0-7803-9236-1/05 2005 IEEE

[16] Theodoro M. Wong, Richard Golding, 'Self Management for Distributed brick based storage' A0508-013, IBM research report, August 2005, IBM research division.

[17] John D. Strunk, Garth R. Goodson, Adam G. Pennington, 'Intrusion Detection, Diagnosis, and Recovery with Self-Securing Storage' School of Computer Science Carnegie Mellon University, PA 15213, May 2002, Pittsburgh.