©2010 International Journal of Computer Applications (0975 - 8887)
Volume 1 – No. 24

An Analysis of Mechanisms for Making IDS Fault Tolerant

Perminder Kaur Dhavleesh Rattan Amit Kumar Bhardwaj

ABSTRACT
This paper is a survey of the work, done for making an IDS fault tolerant. IDS are prone to various attacks and it becomes the natural primary target of hostile attacks with the aim of disabling the detection feature and allowing an attacker to operate without being detected. This paper suggests that intrusion detection system (IDS) must be fault tolerant; otherwise, the intruder may first subvert the IDS then attack the target system at will. Making an IDS fault tolerant is a challenging task.

1. INTRODUCTION
Fault tolerance is a means of achieving dependability, working under the assumption that a system contains faults, and aiming at providing the specified services in spite of their presence. The ubiquity of Internet has continually increased the incidence of exploitation on the vulnerabilities of computer systems and networks. Furthermore, the computing environment has shifted from the traditional centralized computer systems to the networked information systems (NIS), and unfortunately, the NIS is subject to frequent intruder attacks. The current focus of IDS research includes efficiency (i.e., reducing the computing resources consumption), accuracy (i.e., design of a 'better intrusion detection algorithm) and coverage (i.e., detecting more attack types). These issues are important; however, an IDS may be attacked first. After it has been subverted, the system is left defenseless. Hence, it is important to make an IDS fault tolerant.

This paper is organized as follows: section 2 covers analysis of existing mechanisms, section 3 covers results and conclusions of our research.

2. ANALYSIS OF PREVIOUS WORK
A survey on fault tolerance techniques, for IDS, can be found in [1]. Some surveys on the architecture for Integrity checking and intrusion tolerant server are there in [2-5].
Papers on fault tolerance mechanisms for Network Intrusion Detection System are found in [6-10].
Disabling the intrusion-detection system can happen in the following ways:

Denial-of-service attacks. Denial-of-service attacks are a powerful and relatively easy way of temporarily disabling the intrusion-detection system. The attack can take place against the detector, by forcing it to process more information than it can handle (for example by saturating a network link). This usually has the effect of delaying detection of the attack or, in the worst case, of confusing the detector enough so that it misses some critical element of the attack. A second possibility is to saturate the reaction capability of the operator handling the intrusion-detection system. When the operator is presented with too many alarms, he can easily miss the important one indicating penetration, even if it is present on the screen.

Evasion of the detection. Several techniques have been developed to evade detection of an attack by intrusion-detection systems. Network-based tools, the most popular tools today, particularly suffer from these attacks involving hand-crafted network packets:

1. Attack by IP fragmentation. Intrusion-detection systems have diffculties reassembling IP packets. Therefore, splitting an attack artiffcially into multiple packets creates a mismatch between the data in the packet and the signature, thus hiding the attack.

2. Attack via the TTL (Time To Live). By altering the TTL of IP packets, it is possible to make the intrusion-detection system see packets that will not arrive at the target of the attack. By inserting fake data into the communication stream, an attacker can interleave the attack with bogus information, thus hiding the attack from the intrusion detection system while the target correctly reconstructs this attack data and reacts to it.

Karl N. Levitt & Steven Cheung[1] have given some common techniques in fault tolerance and security.
These are:

1. Redundancy.
2. Majority voting.
3. Sending packets over multiple communication paths.
4. Storing critical files in more than one site.
5. Using multiple servers for authentication, Error detection or correcting codes.
6. Cryptography.
7. Heterogeneity (e.g. N-version programming) Having heterogeneous hosts and routers which run different communication protocols; cost: standardization of protocol and OS.
8. Error containment Access control, firewalls.
9. Detection System Diagnosis (e.g. active probing for faults) IDS, anomaly and misuse detection, auditing, testing or monitoring by

site administrators, virus scanners, integrity checking.

If the likely faults affect a single protected component, only then the Redundancy is effective e.g., a processing element. Moreover, fault masking prevents the fault from inducing errors that propagate beyond the component that suffered the fault. There seems to be a related concept in the security domain. If a computer on a network is compromised by an attacker, it should be difficult for him to use this compromised machine as a base to attack other machines. Access control mechanisms and firewalls associated with network components can block or at least limit the spread of attacks.

Architecture For Integrity Checking:

Integrity represents whether or not an agent has been modified from its original state This agent could be a device driver, a kernel security agent (such as a firewall), a security service (such as VPN), an OS kernel invariant or any other program. Today's advanced viruses and worms attack software running in memory to circumvent operating system protections. Such attacks often disable intrusion detection systems in order to execute malicious payload.

Y. Peggy Shen, Wei-Tek Tsai, Sourav Bhattacharya, Ting Liu[3] have proposed a system architecture to enhance the attack tolerance of IDS through integrity cheking.

The System uses the anomaly detection and sandbox techniques to detect intrusions of the IDS. The anomaly detection technique first establishes the normal program behavior ("self '), then detects deviation from the normal program behavior. The definition of self is defined as finite numbers of sequence of system calls in the running processes of an application program.

It is a real-time intrusion detection system. It has three major components the Integrity Checker (IC), the IDS Monitor (IDM) and the Neighborhood Watcher (NW).

Integrity Checker (IC) - The IC detects unauthorized modification and replacement of executable and configuration files. The IC does that by checking these files periodically. The IC computes 32- bit CRC values for each executable file and configuration file at the system initialization time as well as runtime. If any files are modified or replaced, the runtime computed CRC values will be different from the original CRC values.

Intrusion Detection Monitor (IDM) -The IDM monitors the normal program behavior of the IDS processes/threads, and verifies that the IDS is operating within the sandbox. IDM sends out the monitoring results of the previous frame to all the **NWs** in the group. If the **NW** fails to receive the results in a frame, it increments the strike-counter by one. The strike-counter is used to accommodate the asynchronous nature of the **NWs,**

Neighborhood Watcher (NW) - The **NWs** are responsible to monitor IDMs located in the network. The IDM and **NW** transmit heartbeat messages to each other periodically. In other words, they monitor each other periodically. If the **NW** detects that the IDM has been compromised, it sends a warning message to the security personnel and other **NWs.**

The advantage of this system is that it can detect intrusions of **IDS** as well as itself in a real-time manner. Architecture enhances the attack tolerance of **IDSs.** The architecture is a hybrid of distributed, redundant and cross-corroborating techniques. The design of the system is flexible and scaleable.

Gene H. Kim and Eugene H. Spafford[5] describes the design and implementation of the Tripwire tool. They analyzed various security tools, and provide a model for building security tools with similar goals. The goal of integrity checking tools is to detect and notify system administrators of changed, added, or deleted files in some meaningful and useful manner.

Tripwire uses interchangeable "signature" (usually, message digest) routines to identify changes in files, and is highly configurable. It uses two inputs: a configuration describing file systemobjects tomonitor, and a database of previously-generated signatures putatively matching the configuration. Selection-masks (described below) specify file system attributes and signatures to monitor for the specified items.

Intrusion Tolerant Architecture for IDS:

Dan Gorton[4] in his thesis work provides an intrusion tolerant architecture for IDS. The architecture used is composed of four major components:

Application servers, Tolerance proxies, IDS, and a Firewall

- The redundant application servers are used to provide contents to requesting web browsers. Different hardware, operating systems, and applications are used to minimize the risk of all web servers being vulnerable to the same attack or failure modes.
- The tolerance proxies are then used to provide a secure front-end to the application servers. They mediate client requests to one or more application servers depending on the currently selected security policy.
- The IDS is used as one part of the monitoring subsystem
- The firewall is used to minimize the exposure of the intrusion tolerant system. Only web requests are allowed to pass through from the outside.

In the result of his thesis he showed that it is possible to use different fault tolerant mechanisms, e.g. redundancy and diversity, to be able to tolerate some degree of intrusions.

Fault Tolerance Mechanism For IDS:

Various mechanisms have been proposed for making an IDS fault tolerant. I have analyzed some of the research papers published on the area of concern.

Lindonete Siqueira and Zair Abdelouahab[6] have proposed an adaptive fault tolerance mechanism for Network Intrusion Detection System based on Intelligent Agents. Agents collect information related to hosts by monitoring different systems and using the collected information the following actions can be taken:

1. Detect agents which are still active.
2. Detect agents to be replicated.
3. Detect the action of malicious agents.

By using a list of capacities for each agent , and monitoring the actions that are accomplished by each agent of the system , malicious agents can be detected.

R.Shashikumar and L.C.S. Gouda[7], provide a reconfigurable IDS architecture to provide confidentiality, data integrity, authentication and nonrepudiation. The architecture was implemented based on the FPGA hardware. The reconfigurable hardware unit processes the TCP three way handshakes and the Server and Client TCP stream reassembly. Five important states (CLOSED state, SYNSENT state, SYN-RECV state, ESTABLISHED state and EXCHANGE state) are examined to build up the proper TCP three way handshakes needed for the TCP connection. During the building of the TCP connection, the control signals “Division”, “Flag-vulnerability” and “Established” will be the output to the downstream units. The division signal controls the Converger unit In this process, attacks such as Stealthyscan and half TCP connection can be identified.
The autonomous restructuring algorithm is designed to handle the faults that most frequently occur due to gate oxide shorts or metal to metal shorts and provides the feature of self-healing, with built-in autonomous restructuring units.
The results obtained confirms that the system is fast and is ideally suited for monitoring high speed networks and provides improved security to the shared resources on Internet and Intranet. By parallelizing the tasks of reassembling TCP packets on the server and the client on a FPGA the performance of the IDS is greatly improved.

Pabitra Mohan Khilar, Jitendra Kumar Singh, Sudipta Mahapatra[8] propose a failure detection service that uses a heartbeat based testing mechanism to detect failure and take the advantage of cluster based architecture to forward the failure report to other cluster and their respective members.
Failure detection algorithm maintains a heartbeat receive table for each member node in each clusterhead. When a heartbeat from a particular member is received, a new freshness point is calculated using the arrival time of this heartbeat and previous heartbeat messages and new timeout period is set equal to this freshness point.
(i) In every heartbeat interval **THB** each member node sends a heartbeat message to the clusterhead.
(ii) If heartbeat from a particular member is received within the timeout period TTM, clusterhead first saves the arrival time t of this heartbeat message according to its local clock. Then a new freshness point is calculated using the arrival time of this heartbeat and previous heartbeat messages and new timeout period is set equal to this freshness point.
(iii) If the heartbeat from a particular member is not received within the timeout period TTM then that node is considered as failed by the CH. The CH broadcast the firm failure message containing ID of the node to the group.
When a gateway node GW receives this message it forwards this message to the clusterhead of the neighboring clusters.
Results show that complexity of the message(bandwidth utilization) increases linearly with the number of nodes. Local detection time is independent of the number of nodes. This approach is linearly scalable in terms of consensus time.

Liwei Kuang, Mohammad Zulkernine[9] propose an intrusion-tolerant mechanism for network intrusion detection systems (NIDS) that employ multiple independent components. The mechanism monitors the detection units and the hosts on which the units reside and enables the IDS to survive component failure due to intrusions. As soon as a failed IDS component is discovered, a copy of the component is installed to replace it and the detection service continues. We implement the intrusion-tolerant mechanism based on the CSI-KNN-based NIDS and evaluate the prototype in the face of component failures. The results demonstrate that the mechanism can effectively tolerate intrusions.

3. RESULTS AND CONCLUSION

The results of the above analysis can be summarized based upon the following evaluation criteria used for fault tolerance:

1. Availability of the resources in the hosts (memory, disk space, etc.) i.e denial of service.
2. Reliability i .e. Mean time between break-ins, covert channel capacity

The most widely used mechanisms for fault tolerance can be summarized as:

1. Replication Of software agents.
2. Employing Redundancy in processing elements.
3. Integrity checking for self healing.
4. Using Reconfigurable hardware and restructuring architectures.

5. Fault detection using Heartbeat messages in multiagent systems.

The result of the evaluation of the above mechanisms based upon above criteria is shown in table 1 below:

Table 1: Evaluation results

Sr. No.	**Mechanisms for Fault Tolerance**	**Availability**	**Reliability**
1.	Replication Of software agents.	High	High
2.	Employing Redundancy in processing elements.	Appropriate	Low
3.	Integrity checking for self healing.	High	Appropriate
4.	Using Reconfigurable hardware and restructuring architectures.	High	High
5.	Fault detection using Heartbeat messages in multiagent systems	High	Low

From the above analysis I can conclude that intrusion detection system (IDS) must be fault tolerant; otherwise, the intruder may first subvert the IDS then attack the target system at will and the main requirements for making an IDS fault tolerant are:

Timeliness - the system shall detect intrusions of **IDS** in a timely fashion. Since the **IDS** protects the computer systems and networks, a compromised **IDS** makes the target system's door wide open for intruders. A compromised **IDS** needs to be detected and reported immediately.

Scalability - the system shall be scaleable in the sense that it should work in a network of few workstations or hundreds of servers, with few **IDSs** or hundreds of IDS.

Flexibility - the system shall be flexible. Some IDSs employ centralized detection algorithms, but some distributed detection algorithms. Since the system protects **IDSs,** thus, it must accommodate both the centralized and distributed IDSs.

Accuracy - the system shall detect intrusions accurately. It is essential to reduce the false alarm rate. When the false alarm is high, the security personnel are overwhelmed with the false alarms. Worst yet, he or she must plow through all the false alarms to hunt for intrusions.

Resilience to Subversion - the system shall resist subversion. If the system is compromised, then the IDS is in danger of being attacked. Thus, it is vital that the system has built in self-protection mechanism.

4. REFERENCES

[1] K.N. Levitt, S. Cheung**,** "Common Techniques in Fault-Tolerance and Security," Proc. of the Dependable Computing for Critical Applications 4, pp. 373-377, 4-6 Jan. 1994,

[2] L. Catuogno, I. Visconti,"A Format-Independent Architecture for Run-Time Integrity Checking of Executable Code." Proc. of the Third International Conference on Security in Communications Networks, 2002

[3] Shen, Y.P. Tsai, W.-T. Bhattacharya, S. Liu, T," Attack Tolerant Enhancement Of Intrusion Detection Systems." Proc. 21st Century Military Communications Conference, Vol 1, pp. 425-429.

[4] Dan Gorton, "Extending Intrusion Detection with Alert Correlation and Intrusion Tolerance" Masters Thesis, 2003

[5] G. H. Kim, E. H. Spafford, "The design and Implementation of Tripwire: A File System Integrity Checker". Proc. Conference on Computer and Communications Security,Vol 2, pp. 18-29, November 1994

[6] Lindonete Siqueira and Zair Abdelouahab," A Fault Tolerance Mechanism for Network Intrusion Detection System based on Intelligent Agents (NIDIA)." Proc. The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06) , Vol 00, pp. 49-54, 2006

[7] R.Shashikumar and L.C.S. Gouda," Self-Healing Reconfigurable FPGA Based Fault Tolerant Security Model for Shared Internet Resources" IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.1, January 2009

[8] Pabitra Mohan Khilar, Jitendra Kumar Singh, Sudipta Mahapatra," Design and Evaluation of a Failure Detection Algorithm for Large Scale Ad Hoc Networks Using Cluster Based Approach" Proc. 2008

International Conference on Information Technology , Vol 00, pp.153-158, 2008.

[9] Liwei Kuang, Mohammad Zulkernine, “"An Intrusion-Tolerant Mechanism for Intrusion Detection Systems," Proc. 2008 Third International Conference on Availability, Reliability and Security, pp.319-326, 2008.

[10] C. KO**,** “Execution Monitoring of Security-Critical Programs In **A** Distributed System: **A** specification Based Approach”, Ph.D. Dissertation, Computer Science Department, University of California at Davis, 1996.