Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry

[1]Adeosun, Titilayo Helen
Department of Accounting
University of Ilesa, Nigeria
thadeosun@gmail.com

[2] Olajide, Blessing Olajide
Department of Computer Engineering
Federal University Wukari, Taraba State Nigeria.
olajideblessing55@gmail.com

[3] Adeosun, Olajide Olusegun
Department of Computer Science
Ladoke Akintola University of Technology, Ogbomoso, Nigeria.
ooadeosun@lautech.edu.ng

ABSTRACT

Bank critical systems are systems that are not permitted to fail during their operation schedule; they are expected to perform operation 24 hours a day without any interruption. This trailed that all its constituent embedded systems (nodes) must be optimized to an appreciable degree of reliable functionality. This study focused on how banks service delivery can be sustained uninterruptedly in order to serve their customers optimally and guard against cost and disasters which may result from computer system failure during operation. The requirement constraints of embedded systems application design make software-defect masquerading faults inevitable as the nodes collaboratively relate to ensure a dependable service delivery of the critical systems. This is because embedded systems networks are closed network and therefore masquerading faults only emanate from defects in the software of the nodes. To remedy this attack that threats seriously the critical systems reliability and dependability, an algorithm was proposed. The proposed solution involves three main stages: the pre-authentication stage, authentication stage and the auto-repair stage which ensure the system auto-reparability from faulty state. The network simulation of the proposed solution was conducted using Matlab and the results shows that the reliability of fault-tolerant critical system were found to be 0.9999 all through the trial. This is so because all the starting nodes survive till the end of the critical system's operation.

Key words: *Algorithm, Critical system, Embedded system, Masquerade fault, Fault-tolerant*

Introduction

The demand for continuously available electronic system increases everyday not only in the banking sector but in all sectors of the economy. Transaction process, communication system and critical processes all require non-stop, fault tolerant operations. The need to guard against computer threats or failure in banks should not be handled with levity due to its consequential effect which may be costly and irreparable. Banks' products and services are in varying degree and are delivered via computer systems especially now that Nigeria is a cashless economy. Banks have invested greatly and deployed information and communication technology in their service delivery, this afford customers access to their bankers and desired product and services through alternative channels regardless to their geographical locations without necessarily entering into the banking halls. These channels are driven by systems which cannot afford to fail. They include: automated teller machine (ATM), point of Sale (POS), online banking, electronic fund transfer (EFT), telephone banking to mention a few. According to GeekforGeek, (2024), Fault tolerant in system design is important for it provides timely and reliable services, enhance users experience and satisfaction as customer demand for accessible and uninterrupted services. Fault not detected early always result in disruption of operations and loss of revenue to the bank. A dissatisfied customer will always switch from one bank to another. Designing a fault tolerant or highly available system can be achieved by following four basic principles: redundancy, fault isolation, fault detection and annunciation and online- repair. There are dearth literatures on this study, the extant works did not really address fault tolerant as it affect the banking sector. It is against this backdrop that this paper intends to examine the strategy that can be employed to militate against system failure without service interruptions.

Kranz, (2023) opined that fault tolerance systems are designed to compensate for multiple failures. Such systems automatically detect failure of the central processing unit (CPU) input/output (I/O) subsystem, memory card, mother board, power supply or network components. The failure

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

point is identified and a backup component or procedure is immediately takes its place with no loss of service. The idea is to prevent the crash of key systems and networks and focus on issues related to uptime and down time. Fault tolerant can be provided at the software level, hardware level or enabled by combination of the two. At the software level the operating system provides an interface that allow a programmer to checkpoint critical data at predetermined points within a transaction. At the hardware level it is done by duplexing each hardware component, disks are mirrored, multiple processors are grouped together and their outputs are compared for corrections. When an anomaly occurs, the faulty component is identified and taken out of the operation but the machine continue to function as usual.

A critical system is a system a system technology that is deemed by the entity to be of particular importance. It is defined as a system which it is failure may pose a risk to the life of an individual or individuals (Science direct.com). The constituent nodes of the critical systems work collaboratively to ensure the efficient delivery of the critical system overall services. In critical systems, dependable output delivery among its nodes is crucial to overall system dependability. Most nodes often have strict dependability requirements that are subjective to factors such as cost, size constraints or the system architecture (Pfeifer, 2000). These factors cause nodes to use a single shared network for all local communication among them. The use of the shared network if not properly governed will make the appearance of masquerading fault inevitable.

Traditionally, masquerading faults has been viewed as a malicious attack, rather than a fault tolerance problem. This is a misconception because embedded system's network are typically closed (i.e., their networks are not accessible to outside intruders).They usually do not include security methods for preventing malicious attacks because there is no interface for any snooping to take place. Masquerading fault can only be initiated by nodes within the systems that malfunction to take on the identity of another legitimates node to disrupt the sequence of operation of the nodes as pre-organized. These malfunctions are caused by logical errors that are not easily visible to the programmer at the time of the systems coding or configuration. Other causes could be from exposure of the critical system to harsh environment, long time use or the need to upgrade the systems software for interoperability with new critical systems. The masquerading faults that occur as a result of logical defects in the software of the

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

nodes are called software-defect masquerading faults. Some other software-defect faults includes replay faults, black hole faults, gray hole attack and so on (Haiyan and Kongjun, 2016)

Software-defect masquerade faults are threat to dependable system output delivery. It is a fault mode where a software-defect causes one node to send an output as having come from another node. Vipul, etal., (2015) described the development of a fault-free reliable software as a daunting task in spite of meticulous planning and well documented processes, occurrence of certain defects are inevitable. Unfortunately, many embedded system designs do not address this particular failure mode. This fault mode cannot be overlooked because they result to either grinding the critical system operation to a halt or slightly-off-specification problem in service delivery. Hence this paper presents a proactive algorithm to detect, prevent and repair nodes from masquerading in the critical system.

Literature Review

Sutar and Pawar (2015) proposed cryptographic security mechanism to prevent masquerading among the control units in vehicles. Various embedded control units within the vehicle are connected together by Controller Area Network (CAN) protocol which has no authentication field. For this reason they are prone to different attacks and their work utilizes a software based approach: Message Authentication Code (MAC) algorithm to prevent masquerading attack. Sarika and Verghese (2013) proposed an algorithm for detecting a particular phishing attack called tab nabbing using distributed software agents. Tab nabbing is an attack which prompts the user to enter login particulars to well-known websites by imitating those websites when the webpage is idle for some time. The distributed agent's method is based on the integration of autonomous distributed agents with strong level of intelligence. A distributed multi-agent system presents a prodigious capacity for high level of learning, distribution of tasks and responsibilities, fault recovery, and adaptation to new changes. The work consists of agents in 3 levels and they communicate with each other as needed. The level1 agents checks the URL of the webpage and confirms whether it is not prohibited and checks the layout of webpage and confirms whether it is not changing when the webpage is idle for some period. The level 2 agent identifies the webpage as a phishing page or a legitimate page. In this mechanism level1 agent acts as an agent for detecting URL mystification and detects tab nabbing attack. Level 2 agent is acting as a web watcher when the threshold values of actual web pages differ from the values received from its phished webpage.

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

Manikantan, Yu and Tricha (2009) suggested an aware detection (CAD) approach that adopts two strategies, hop-by-hop loss observation and traffic overhearing. Each intermediate node observes the behavior of its previous–hop and next-hop neighbors to detect misbehaving nodes. Software fault tolerance utilizes static and dynamic methods that include checkpoint and rollback recovery to provide a correcting mechanism to tolerate errors that emanated from software or program errors. Fault tolerance techniques such as Process-Level Redundancy (PLR) are used for such transient faults because hardware based fault tolerance technique is expensive to deploy. Other types of fault tolerance techniques in distributed system include fusion based technique and replication based fault tolerance technique (Arif and Murat, 2015).

Aghaei, etal., (2011) propose client transparent fault tolerance model for web services which applies N-version and active replication techniques to recognize server masquerading faults and redirect requests to reserved backup server in order to reduce the service failures. Santos, Lung and Montez (2005) also suggested a fault tolerance infrastructure that added an extra layer acting as proxy between client requests and service provider's response to ensure client transparent faults tolerance. Luis (2006) observed that Controller Area Network (CAN) is been used in some safety-critical applications and was able to point out that CAN inherent event-triggered transmission mode does not favor dependability. In other words it is easier to detect errors and build fault-tolerant mechanisms for time-triggered communication protocols.

Fault Tolerance algorithmic approach

This work approach to solve software-defect masquerading faults in critical systems gives a step by step process for modeling a masquerade fault-tolerant critical system. The processes were grouped into three modules. The modules include the pre-authentication stage, authentication stage and the auto-repair stage. All the modules were deployed on a time-triggered network with time triggered protocol and architecture (TTP/A) of star topology to manage the basic communication of the nodes on the shared channel within the critical system. This approach used TTP/A because it is very popular and being used in abstracting embedded system networks as illustrated by Figure 1. It is functionality includes clock synchronization and a redundant star coupler that mirrors the primary star coupler. This protocol gives the nodes the ability of self-initiation and ensures that the star coupler do not become a central point of failure on the network.

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

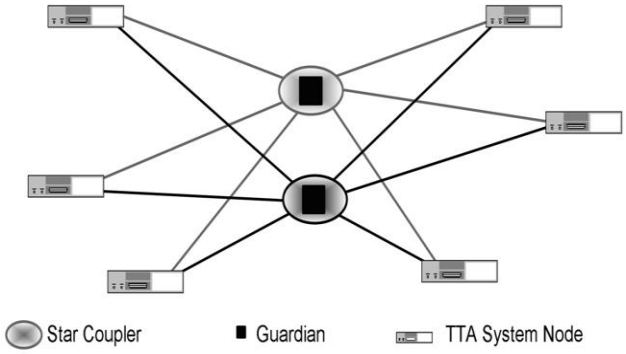

Figure 1: Topology of Time Trigger Architecture (TTA)-Star in Critical System
(Source: Kopetz, 2001)

The proposed fault-tolerant algorithm identifies masquerading nodes and stops their effect from propagating into the embedded critical-system shared network. It does this by replacing the faulty nodes with their replicas that have the initial correct credential to transmit at the correct time slot. The algorithm utilizes authentication service and light weight agent (LWA) service to detect masqueraders and to ensure the node's availability, reliability and dependability in service delivery to the shared network, a triple modular redundancy (fault tolerance mechanism) was also deployed at each node within the critical system to perform node repair. Therefore, each node is abstracted as a module of three healthy identical nodes:

(a) Node Authentication Module

(b) Node Health Monitor module

(c) Nodal Fault –tolerant System Module

These modules were deployed on the bus guardian that is resident on the star coupler in TTA of star topology architecture to accomplish the set objectives.

Fig 2: Model of the fault Tolerant Strategy Modules

Pre-authentication stage

During configuration, each node is designed at start up dynamically to obtain a secret key (SK) and an identity (Id). The secret key is only known to the node and the bus guardian in the star couplers. The secret keys are not broadcasted periodically in the message description

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

list (MEDL) as the Id and number of Slot (NoS) to other nodes on the network. The NoS is the overall number of available time frame shared by the nodes during a standard operation schedule of the critical system. At the Pre-authentication stage the algorithm uses two servers to execute its process. The servers are:

(i) Authentication Server (AS)

(ii) Ticket Granting Server (TGS)

The AS keeps the database of the assigned secret key (SK) and their corresponding Id. A symmetric cryptosystem for using the secret key to encrypt the Id of the nodes requesting authentication is also available on this server. The algorithm uses mono-alphabetic cipher substitution as opined by Caeser Cipher for encryption and decryption in the AS. The TGS is the server that accepts the validated credentials from the nodes after the node has decrypted the encrypted Id with its own copy of the SK. It then adds a session key to the credential before giving it back to the node. The credential that is given back to the node contains the SEK, Id and the SEK. The SEK is the addition of Node Time Slot (TS) and number of slot (NoS). The SEK is used to gain access to the shared channel at the timeslot indicated in it. This is made possible by a variable called timeout counter inside the node that counts down the SEK until it becomes zero then the node is granted access to the shared channel. i.e., Timeout counter = SEK-1 until it eventually becomes zero. This happens concurrently on all the nodes before transmission. The first node to count to zero transmit first and the next to get to zero do so after the first has finished the use of the channel in its time frame and so on in that manner till the last node in a normal fault free scenario. The TGS also keeps a database of node Id versus the additional of node time slot known as SEKguide. This database is used to retain a copy of the assigned SEK to the node as a guide.

Authentication Stage

When the timeout counter of a node equals zero and it wants to transmit, it first presents its SEK to the TGS, then the TGS uses the SEKg to compare the tendered SEK parameters. If there is any discrepancy in the SEK to the stored SEKg then the node attempting transmission is reported to be masquerading. This report is made by the Light weight agents. The light weight agents are mobile agents with restricted intelligence and have the ability to migrate from one node to another to effect changes on some systems configuration. At the instance of a node attempting to transmit, the lightweight agent at this stage migrates from that node to-fro to the

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

star coupler and compares the SEKg with the SEK. If the SEK and the SEKg equals then it permits the node to transmit. If not, the node is denied the use of the shared channel.

Auto-repair Stage

After the masquerading node has being identified and prevented from using the shared network at the wrong TS by the authentication stage. The auto-repair stage replaces the faulty node with its replica that is viable and healthy. This module implements a Triple Modular Redundancy (TMR) fault tolerance mechanism to achieve its task. The Lightweight agents (LWAs) from the authentication stage reports to the faulty node that one of its integral nodes is faulty. The TMR module then uses the disagreement detector to ascertain the faulty node and service is transfer to one of the remaining two spare. Here, the nodes do not fail or freeze and so the critical system does not experience any down time. The faulty nodes are self-repaired.

The Node

A node is made up of three identical nodes in parallel. This forms a node TMR module (Figure 2). Generally in this write-up a subsequently a node is used to imply a node TMR module. The individual node has the capability to perform self-reliantly the assigned task within a critical system. A single node is abstracted as a finite state machine with six different states and can transit from one state to another base on the fault tolerant logical function. The states are COLD state, INITIAL state, ACTIVE state, FAULT state, TIMEOUT state and the FINAL state. At any instance of a node-shared network communication only a node in the TMR module is active while the others are on Standby.

To help the disagreement detector to select a replica to replace a masquerading node a Basic Fault-tolerant System (BFS) switching concept is employed.

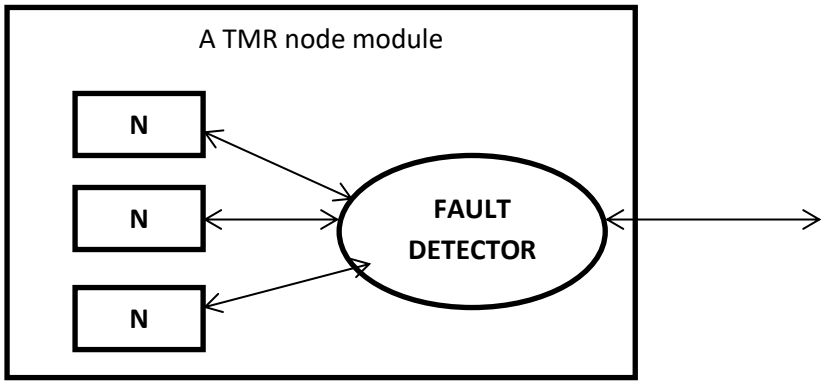

Figure 2: TMR Structure of the Node Module

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

Fuzzy rule For the Proposed Fault-tolerant Algorithm

Masquerading fault occurs when there is a logical problem in the software of the node that causes some of the node variables like the noS and TS which consequently result to a node having the wrong session key SEK. Fuzzy rules were therefore developed to form the decision logic of the constituent stages of the Fault-tolerant algorithmic approach as described:

(a) Fuzzy Rule for Pre-Authentication Stage.

Rule 1: If Node id is decrypted **THEN** issue SEK

Rule 2: If Node id is not decrypted **THEN**Don't issue SEK

(b) Fuzzy Rulefor Authentication Stage

Rule 1: If SEK =SEKg**THEN** no masquerade

Rule 2: If SEK < or >SEKg**THEN** masquerade

(c) Fuzzy Rule for Auto-repair Stage

Rule 1:If Decision logic = 1 **THEN** do nothing

Rule 2: If Decision logic = 0 **THEN** Repair the node

The process flow is as depicted by Figure 3.

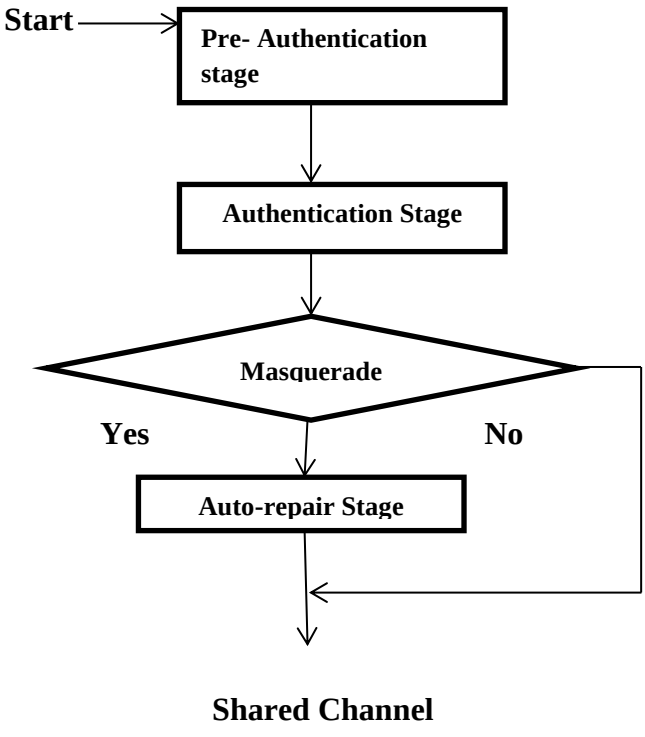

Figure 3: Flow Diagram of the Algorithmic Process

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

The Proposed Fault- tolerant Algorithm for Solving Masquerading Faults in Critical Systems

A node has four variables which are:

(a) Secret key (SK)

(b) Node Id (Id)

(c) Session Key (SEK) = TS + noS where TS is time slot and noS is slot number

(d) Timeout counter

The timeout counter allows a node to know when is its turn to transmit because a node transmit when Timeout counter =SEK – 1 until Timeout counter eventually equals zero. Also the nodes operate in six different states which are as explained:

{**COLD State**}

It's the state of all nodes before the critical machine is turned on

{**INITIAL State**}

When the critical system is turned on, the node transit to the initial state. At this state the pre-authentication stage processes are performed as follows;

Input id *// from the node*
Encrypt id *// by the AS*
If Node decrypt id
Then issue SEK and move to {ACTIVE state} *// by Authentication stage*
Else
Don't issue SEK *// by Authentication stage*

{**ACTIVE State**}

Input SEK *// from the Node*
If (SEK =SEKg) then Decision logic = 1 move to {TIMEOUT state}
Else (SEK >or <SEKg) then Decision logic = 0 move to { FAULT state}

{**FAULT State**}

(SEK> or <SEKg) then Decision logic=0
Begin {
Repair the Node
}
Return to {INITIAL state}

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

{TIMEOUT State}

if (SEK= SEKg) begin {

SEK -1 until SEK =0

}

Then transmit on the shared channel then move to the {FINAL state}

{FINAL State}

At this state the Node has completed its operation successfully within the allotted time slot and control to transmit is transferred to the next node to transmit at its own time slot.

Note that pre-authentication stage processes are performed at the {INITIAL state}, Authentication stage processes at the {ACTIVE state} and {FAULT state} and auto-repair stage processes at the {FAULT state}.

Result and Discussions

This study employed simulation as a means of supporting the use of the proposed algorithm to solve software defect masquerading fault in critical system network. The simulation tool used is Matlab version 2012b. It provides a pictorial and graphical editor to build models for various scenarios needed for the fault-tolerant critical system model. All the three module's operations are modeled in an object–oriented approach which gives intuitively easy mapping to real embedded critical system's network. The Simulink model experimented with four nodes in a time-triggered network as shown in Figure 4.

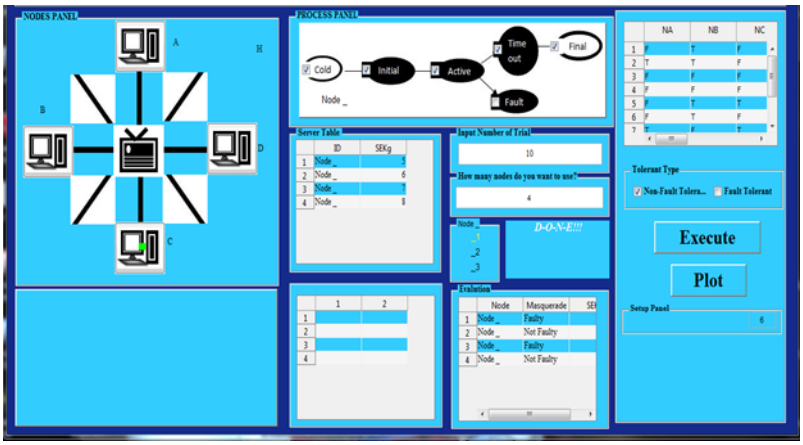

Figure 4: Simulink Model of the Software-defect Masquerade Fault-Tolerant Critical System.

A system is said to be fully reliable, if all the starting nodes survives till the end of the critical machine's operation.

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

$$\text{Reliability (R)} = \frac{\text{NumS}}{\text{NumO}} \quad (1)$$

where:

NumS is the number of survived nodes; and

NumO is the original number of working nodes at the start of the critical system.

Table 1: The Software-Defect Masquerading Fault-tolerant Critical System (Four Nodes)

NA	NB	NC	ND	NumF	NumS	Reliability	Time(s)
T	T	F	T	1	3	0.9999	10
F	F	T	F	3	1	0.9999	25
F	T	T	T	1	3	0.9999	29
F	F	F	T	3	1	0.9999	37
T	T	F	F	2	2	0.9999	54
T	T	F	T	1	3	0.9999	66
T	T	F	T	1	3	0.9999	74
T	T	T	T	0	4	0.9999	79
T	T	T	T	0	4	0.9999	85

Discussion of findings

This implies that for an ideal fault-tolerant critical system the NumO must be equal to NumS. The simulation result gotten for the successive runs of the model of the masquerade fault-tolerant critical system gives the results on Table 1. The results of the four-node fault-tolerant critical system showed that when NumF is 2 and NumS is 2 the reliability is 0.9999. Likewise, when NumF is 3 and NumS is 1 the reliability is 0.9999. The reliability of fault-tolerant critical system were found to be 0.9999; this gives 0.9999 all through the trial. This is so because all the starting nodes survive till the end of the critical system's operation. With these results, this work has achieved the aim of optimizing the reliability of the critical systems by solving the problem of software-defect masquerading among the critical system's nodes. This is in line with Kranz, (2024) who stressed that fault tolerant environment restore service

Citation: Adeosun T.H, Olajide B.O & Adeosun O.0 (2024). Fault Tolerant Strategy: A Tool for Militating against Masquerading Threat in Banking Industry. LAUTECH Journal of Accounting, Finance and Contemporary Management Research 1(1), 57-70

instantaneously and a high availability environment strives for 99.999 percent of operational service.

Conclusion

This work presents an algorithm, simulation and a prototype implementation of a proactive fault-tolerant solution to software-defect fault management among critical system nodes**.** Correspondingly, the results from the simulated fault-tolerant critical system model showed that the approach of this research to solve masquerade fault an embedded system network of the critical system is viable if properly deployed in an embedded system network. It is capable of providing a reliable, dependable and always-available critical system to organizations, industries, banks and every other social sector that hinges on computer automation in delivering their services.

References.

Ademaj, A. (2002). Slightly-off-specification failures in the time-triggered architecture. Proceedings of the Seventh IEEE International Workshop on High Level Design Validation and Test (HLDVT'02), Cannes, France, pp7-12.

Aghaei, S., Khayyambashi, M. and Nematbakhsh, M. (2011). A fault tolerant Architecture for web services. Innovations in Information Technology (IIT). International Conference, pp53-56.

Arif, S. and Murat, A. (2015). Fault tolerance mechanisms in distributed system, *International Journal of Communication, Network and System Sciences*. 8, pp471-482

Fortinet, (2024). Definition, benefits, component of, and consideration for fault tolerance system. https://www.fortinet.com> fault…t

GeeksforGeeks. Fault tolerance in system design. https://www.geeksforgeeks.org> f…. last updated in July 2024.

Haiyan, L. and Kongjun, B. (2016). A new authentication solution for prevention of some typical attacks in ad- hoc network. *International Journal of Security and Its Application.* 10(7) pp333-344.

Kranz, G.(2023). Fault tolerance. Disaster recovery facilities and operations. techtarget.com

Kopetz, H. (2001). A comparison of TTP/C and FlexRay Technische University¨at Wie, pp1-5.

Luis, A. (2006): Safety-critical automotive systems new developments in CAN in W4: Design Issues in Distributed Communication-Centric Systems. University of Aveiro, LiU-Tek-Lic-2006:58, pp1-7.

Manikantan, S., Yu C. and Tricha, A. (2009). Channel-Aware Detection of Gray Hole Attacks in Wireless Mesh Networks. IEEE global Communication Telecommunication Conference, pp1-6.

Pfeifer, H. (2000). Formal verification of the TTP group membership algorithm in IFIP TC6/WG6.1 International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing and Verification, FORTE/PSTV 2000, Pisa, Italy, pp3-18.

Sutar, T. and Pawar, S. (2015). Security for controller area network (CAN) Bus in vehicle communication system. *International Journal for Scientific Research and Development* 3(8). pp551-553.

Sarika, S. and Verghese, P. (2013). Distributed software agents for antiphishing. *International Journal of Computer Science Issues (IJCSI)*. 10(3) .No 2, pp1-4.

Santos, G., Lung, L. and Montez, C. (2005). FT web: A fault tolerant infrastructure for web service. EDOC Enterprise Computing Conference, 9th IEEE International, pp95-105.

Spice work. High availability Vs fault tolerance: 3-key difference-spice work. https://www.spicework.com>high….

Vipul, V., Manohar, L. and Sureshchandar, G. (2015). A framework for software defect prediction using neural network. *Journal of Software Engineering and Applications.* http:/dx.doi.org/10.4236/jsea.2015.88038, pp384-394.

Wikipedia. https://en.wikipedia.org>wiki>c….