# Survey on Intrusion Tolerant Systems in Emerging Technologies

*Nouhad Sanoussi*[1*], and *Ghizlane Orhanou*[1]

[1]Laboratory of Mathematics, Computing and Applications - Information Security,
Faculty of Sciences, Mohammed V University in Rabat, Morocco

**Abstract.** The wide spread of the Internet leads to the expansion of networks and the increase of data. For this reason, the use of emerging technologies became a necessity to cover the needs and the lapses in traditional systems. However, these systems might reveal security flaws and become vulnerable to attacks. Security is therefore essential to ensure the confidentiality, integrity, and availability of system data and services. As the attacks get more sophisticated, the protection gets more difficult. In fact, mechanisms such as Firewalls, IDSs, and IPSs are still ineffective against unknown attacks. Consequently, defense-in-depth security should include intrusion tolerance. The concern of intrusion tolerance is not how to defend or detect the intrusion, but how to mask or restrain the intrusion when the network has been intruded. Using three separate lines of approaches: Detection-Triggered, Algorithm-Driven, and Recovery Based, this paper will present, analyze, and contrast several works in two different environments: Cloud Computing and Software Defined Networks (SDNs), to proceed towards Intrusion Tolerant Systems (ITSs).

**Keywords.** Intrusion Tolerance; Detection-Triggered; Recovery-based; Algorithm-Driven; SDN; Cloud Computing.

## 1 Introduction

Currently, so many technologies are very quickly becoming significant topics in both the scientific community and the general public. Especially with the growth of network applications, even the most isolated people have developed strong bonds with one another, which leads to the expansion of networks and the increase of data. Consequently, the development of new environments is a must such as Cloud Computing which ensures that different applications have access to computing power, storage space, and various software services as required, in order to increase the efficiency of using computing resources, minimize power consumption per service, and effectively prevent the error of computing resources. Furthermore, Software Defined Network (SDN) has emerged as one of the most significant network architectures for streamlining network management and facilitating communication networks, due to its primary characteristic of separating the network control from the data forwarding layer.

Nevertheless, the use of these new environments leads to new security challenges such as man-in-the-middle attacks, denial-of-service attacks (DoS), saturation attacks, etc. The traditional security mechanisms such as firewalls, intrusion detection or prevention systems seem to be insufficient against the new types of attacks and intrusions. In addition, the unavailability of these emergent environments is very crucial and an interruption during few minutes or few seconds could have a big technical and financial impact. Thus, the use of new mechanisms of security is a must, especially the Intrusion Tolerant System ITS, which ensures the continuity of services even in the presence of intrusions. Whereas existing surveys already address ITSs in general technologies. This survey focuses on ITSs in Cloud Computing and SDN, and their classification compared with the approaches proposed in [3].
The remainder of this paper is structured as follows: Section 2 presents the aim of intrusion tolerance systems, techniques, and architectures. Section 3 overviews ITS in Cloud Computing. Section 4 gives a survey of ITS in SDN. Finally, a comparison and a conclusion are given in Section 5.

## 2 Intrusion Tolerance Systems

### 2.1 Intrusion Tolerance Need

Vulnerability and attack are the substantial causes of an intrusion. Vulnerability represents every anomaly in a system components that can be exploited to hack that system and an attack is a malicious attempt to hack the system. In a first stage, an attacker tries to find any vulnerability of the target system which seldom ends in failure. In a second stage, he insert an attack which leads the system components to behave abnormally. Finally, this abnormal way of behaving causes security failure. In order to prevent a security failure, the use of intrusion tolerance is necessary to block the effectiveness of malicious attacks, which means that intrusion tolerance aid the system to recover to a proper state before it falls into a security failure.

* Corresponding author: nouhadsan@gmail.com

### 2.2 Fundamental Techniques

The goal of this section is to present the key techniques used in most researches working on Intrusion Tolerant Systems. The major part of these techniques are extracted from Fault Tolerance techniques:

• Redundancy: As the authors of [1] said, no redundancy, no tolerance. But the use of redundancy alone is deficient. On the whole, redundancy is laying additional resources to a system to use them in need. It can be used in both hardware (replication) and software.

• Diversity per se needs redundancy [1]: to achieve diversity, redundant components should be significantly distinct, such as hardware diversity. The used hardware in the redundant components must be different. The same issue to software diversity by using distinct software and in operating system diversity by using different OS that present, in case they exist, different types of vulnerabilities. The higher the diversity, the lower the risk of system failure, but the greater the complexity of the system.

• Voting [1]: is the solution for differences in output data coming from redundant components, it's based on comparing the redundant responses and come to consensus to choose the proper response.

• Acceptance Test: The concept involves a series of statements that will manifest an exception if the system fails [2].

• Threshold Scheme and Distributed Trust [1]: is also called secret sharing. The concept principally focuses on splitting data D into p chunks that could be dispersed in many emplacements. So it requires k or more chunks to rebuild the data D. Otherwise, it reveals nothing.

• Dynamic Reconfiguration [1]: The principle is to dynamically reconfigure the system without any downtime for the system, so the service can be uninterrupted.

• Indirection [1]: It's generally designed as layers, in which designers insert barriers between clients and servers. The most used indirection systems by ITSs are: virtualizations, proxies, wrappers.

### 2.3 Intrusion Tolerance Approaches

In this section, we present three main different approaches to Intrusion Tolerant architectures (SITAR, MAFTIA, SCIT), focusing on three different approaches [3]: Detection-triggered, Algorithm driven and recovery-based system.

• Detection-Triggered based on multiple levels of defense to increase system survivability. One of them is Intrusion Detection which call up recovery mechanisms to work. One of these architectures is SITAR [4] which relies principally on redundancy, diversity, voting, and automatic reconfiguration.

SITAR architecture contains the following components:

1) Proxy servers, the first point of connection between the clients and the COTS servers, are responsible for accepting or rejecting requests based on security policies.
2) Acceptance monitors, put acceptance test on the requests previously accepted by the proxy servers. If accepted, the requests would be forwarded to the specified Commercial Off-The-Shelf (COTS) server.
3) COTS servers, receive requests, treat them and generate adequate responses, then transmit them again to the acceptance monitors to validate them. Validated responses are delivered to the ballot monitors.
4) Ballot monitors Applies a voting and agreement process to the forwarded responses, the result is the final response that will be sent to the proxy and then to the client.
5) Adaptive reconfiguration module (ARM), collects intrusion signs from other components, evaluates the threats, then applies necessary reconfiguration to ensure system security and performance.
6) Audit control module, controls resources used by the components to prevent intrusions.

• Algorithm-Driven based on the principle of augmenting the level of trustworthiness by constructing layers of more trusted components in a progressive way. It uses intrusion tolerant techniques such as voting algorithm, threshold cryptography, etc.

There are several architectures that fit the algorithm driven approach, but MAFTIA [5] is more appropriate to study in that case.

MAFTIA's intrusion tolerance approach is based on the notion of trust and trustworthiness relationship, its architecture employs intrusion-tolerant techniques to construct layers that contain more trusted components and middleware founded on untrusted hosts and networks.

MAFTIA architecture is divided in at least three layers.

1) The hardware layer contains hosts and networking devices that are generally untrusted such as usual personal computers for example, however, they might include pieces of trusted hardware.
2) The runtime support comprises the Operating System (OS), Java Virtual Machine (JVM), and TTCB which is a distributed trusted component that gives a set of trusted time to middleware protocols for communication.
3) The middleware layer consists firstly of multipoint network based on the physical infrastructure that provides basic secure channels, multipoint addressing, etc. Secondly, the communications support module that ensures principally byzantine agreement (voting), group communications, and threshold cryptography.

• Recovery Based system founded on the principle that as soon as a system comes online, it is compromised, therefore, periodic restoration to a good state is necessary [3].

Self-Cleansing Intrusion Tolerance (SCIT) [6] is a recovery-based model workable to servers that are open to the internet such as web and DNS servers. The main advantages of the SCIT architecture are:

– It ensures the diversity of the server, so the attacker should make more efforts to compromise a system.

– It shortens the exposure time of the server, therefore, rendering the information gathered from the reconnaissance phase inaccurate for the attacker.
– It reduces losses by controlling the time that a server is exposed to the network.

The SCIT architecture is founded on two fundamental components. The server cluster is a group of nodes doing the same tasks and affording the same services but running in different operating system platforms to ensure diversity, thus making malicious exploitation more difficult. Each node is continuously routed through the following lifecycle states as:

– Active: Node is online and accepts/processes any incoming requests.
– Grace Period: Node processes any existing requests, but doesn't accept any new ones.
– Cleansing: Node is offline and undergoes the cleansing to get to a known good state.
– Live Spare: Node has been restored and is ready to come online.

The SCIT controller is the center part that manages server rotation in and out of the cleansing mode.

• Hybrid is the last possible approach that could combine two or more of the approaches discussed before.

### 2.4 Analysis

On the whole, SITAR is effective in tolerating intrusions. In fact, the architecture depends especially on acceptance testing that looks for known intrusion types. While balloting helps the system be resilient against unidentified attackers. Whereas validation of incoming requests has the advantage of allowing the system to continue operating even after a hacked COTS server. Additionally, this design is effective against attacks like cross-scripting, SQL injection, access, violation, DoS attack, and data ex-filtration. Unfortunately, this process will endure additional latency on service time.

Generally, MAFTIA provides a full intrusion-tolerant design based on the concept of trust and trustworthiness from bottom components and other trusted security services. However, the use of Byzantine agreement protocol and secret sharing cryptosystems impacts the response delay. Additionally, the integration of the system into other architectures can be costly and complex in terms of using specified APIs and protocols. MAFTIA design offers tolerance against attacks like access violation, impersonation, web defacing, DoS, and data ex-filtration.

Unlike the two other approaches, SCIT differs significantly because it is basically a recovery-based strategy and doesn't include an intrusion detection element. The plan is predicated on the notion that the system would continue to have vulnerabilities, making successful attacks inevitable. SCIT does not require extra latency time. In addition, the intrusion tolerance is determined by the situation of how much the design may restrict the attacks' impact and keeps the services offered to users available, which depends on the exposure time.

## 3 ITS in Cloud Computing

In this section, we present and analyze three Intrusion Tolerant Systems in Cloud Computing.

### 3.1 Adaptive Cluster Transformation ACT with proactive recovery based ITSs

ACT, presented in [7], firstly aims to maintain a good quality of service (QoS) by using a variable cluster size, and secondly defends the system against DoS attacks by early predicting the incoming massive packets, then replacing the compromised clusters with new ones. ACT architecture is presented as follows:

• VMs: have the same functionality such as providing services, connecting to a secure database, then delivering the same responses to the same requests in the case they are not compromised.
• HW servers: are the point of connection between the VMs and the Internet.
• The central controller: checks the system performance and the external threat, then decides the cluster's size (expansion or reduction) and the rotation state of each VM.

The authors consider that the attacker never modifies any system configuration and all attacks except the DoS attack are stopped by security mechanisms (Firewall, IDS, IPS).

Two main schemes in the ACT architecture are considered:

• **Adaptive Cluster Expansion and Reduction:** To maintain the Quality of Service (QoS), the central controller checks continuously the request-response delay. Then, it decides to increase the cluster size if the delay is longer, otherwise, decreases the cluster size.
• **Adaptive Cluster Substitution:** To avoid system failure due to the sudden increase of incoming packets such as DoS attack. The central controller checks continuously the current response delay of each VM and the number of unprocessed requests.

### 3.2 Hybrid Recovery-Based Intrusion Tolerant System

Proactive recovery-based ITS introduced in [8] depends on the periodic recovery of virtual machines to mitigate intentional intrusions or potential errors. Unfortunately, this solution suffers from managing the exposure time, and the VMs could be attacked easily within their working period. To tackle this problem, the authors in [8] propose a novel hybrid recovery-based ITS that utilizes a scheduling mechanism for availability-driven recovery to decrease the exposure time that leads to higher availability and to prevent performance degradation caused by DDoS attacks.

The architecture of hybrid recovery-based ITS is separated into three primary modules:

• **Hybrid recovery actuator:** uses proactive and reactive recovery on the availability-driven recovery scheduling to ensure the lowest level of exposure

time besides preventing the damage occurred by DDoS attacks.

• **Elastic cluster manager:** ensures providing services even when there is a volumetric DDoS. It contains three processes: Cluster scale-up, Cluster scale-down, and double plus one expansion.

• **The system health monitor:** monitors information concerning the states of VMs.

### 3.3 A Framework for Intrusion Tolerance in Cloud Computing

The Framework for intrusion tolerance in Cloud Computing in [9] is obtained by utilizing and adapting the existing MAFTIA intrusion tolerance framework in the Cloud Computing environment to ensure availability, integrity, and confidentiality.

In [9], the authors propose intrusion tolerance framework, which is based on the layered design of Cloud Computing architecture: User Level, Middleware, and System Level.

The framework is composed of components such as:

• Vulnerability and attack prevention: This component is present in all the layers of Cloud Computing. It introduces mechanisms that ensures authentication, authorization, in addition to firewalls, which prevent attacks.

• Error Processing: consists principally of three components: Event Analysis, Error Detection, and Fault Model.

• Fault Treatment: is in charge of fault treatment at the middleware and system level of Cloud Computing.

• Cloud Security Administration System: for managing and treating security vulnerabilities in the cloud environment.

### 3.4 Analysis

The Quality of Service (QoS) is improved by ACT in conjunction with proactive recovery-based ITSs when there is a flash crowd problem or a volumetric DDoS. However, this method did not address how to regulate exposure time when the cluster size was changed. Additionally, it lacked any defenses against stealthy attacks like application-layer DDoS attacks.

On the other hand, Hybrid Recovery-Based Intrusion Tolerant System employs dynamic cluster resizing and availability-driven recovery, efficiently managed the exposure time to maximize service availability, and maintained a particular level of appropriate reaction time despite workload fluctuations. Additionally, this architecture reduced the impact of volumetric and application-layer DDoS attacks.

Finally, the framework for Intrusion Tolerance in Cloud Computing is based on MAFTIA architecture adapted to Cloud Computing built on the notion of Trust and Trustworthiness connection, which enables to create layers of security services that operate as a top-to-bottom fortress to safeguard applications. The performance rises with the boost in the number of hosts in a datacenter. However, the increase in failed hosts leads to an increase in the total execution time.

## 4 ITS in Software Defined-Networking

Software-Defined Networks (SDN) is an emerging technology that separates the control plane from the data plane. The controller in the SDN control plane has network-wide control capabilities.

The three ITS designs elaborated in this section concern intrusion tolerance at the level of the control plane.

### 4.1 Intrusion Tolerant Architecture for SDN Networks Through Flow Monitoring

The benefits of SDN and intrusion tolerance are both utilized in the proposed system [10]. Indeed, in the Opendaylight SDN controller, an Intrusion Detection Module (IDM) has been developed. IDM's major goal is to change the flows and tolerate intrusion after it has been discovered. This is accomplished by keeping track of the switch's packet count for flows.

In fact, after collecting flow information from the switch and tracking packet counts at various intervals, the controller determines the packet rate. A higher priority flow with the same match criteria is introduced with the action of punting to the controller if the rate exceeds the threshold value. To avoid overburdening the controller with packets, this flow will have a 5-second idle timeout. Thus, the controller receives the subsequent few packets. If a packet is from a known sender, IDM will do deep packet inspection on the punted packets and add a new flow with match criteria of 5-tuple information and action of dropping. Otherwise, a new flow is introduced with the source IP, and an action of the drop is performed if the packet comes from an unidentified source. This will prevent any packets from being delivered by an unknown sender, safeguarding the system.

### 4.2 A Fault-Tolerant and Consistent SDN Controller

Performance and consistency can be balanced through the design of a reliable and fault-tolerant Master-Slave SDN controller. The major goal of this work in [11] is to get the best feasible performance between an SDN Master-Slave controller and a single controller.

The proposed architecture of the Master-Slave SDN controller consists of a master controller and a distributed datastore used to store the Network Information Base (NIB) data by backup SDN controllers:

The master attends messages similarly to how a single controller does so, but with the following two exceptions:

• Each NIB update is divided, stored, and transmitted to the common datastore right away once each incoming message has been processed. A further, successively-increasing ID is utilized for consistency checks.

• Consistency assurance processes rely on NIB-updates divided and buffered from a single reference point, enabling a certain amount of Master independence as well as communication between switch buffers and the slave platform. The goal of the suggested adjustments is to minimize the overhead on the master operation.

### 4.3 ITC: Intrusion Tolerant Controller for Multicontroller SDN architecture

In SDN, the controller is considered as a single point of failure that may return the whole network down in case of a security compromise. This issue is partially tackled by the use of multi-controller mechanisms. However, simply using these mechanisms cannot avoid compromising vulnerable controllers due to their visible nature. In the paper [12], the authors propose to approach the issue of intrusion tolerance in the SDN control plane by first applying a Recovery Based model which assumes that as soon as a system comes online it is compromised; therefore, periodic restoration to a good state is necessary. Secondly, they establish Moving Target Defense (MTD) that provides a proactive defense against adaptive adversaries. The goal of the MTD in the Dispatcher is to constantly shift between multiple controllers with diverse configurations in order to increase the uncertainty for the attacker, in effect, diminishing the information gathered from the control plan during the reconnaissance phase of a potential attack. Finally, they put in place probabilistic models that can contribute to the perception of the performance of self-cleansing intrusion tolerance in the SDN control plane.

### 4.4 Analysis

The concept of intrusion-tolerant architecture for SDNs is involved in monitoring the packet count for flows by using IDM. If the packet rate (flow statistics from the switch) exceeds the threshold value, an idle timeout of 5 secs is set to prevent overload of the controller. Additionally, the process of deep packet inspection will endure additional latency on service time. This design is especially effective against DoS attacks, but it can lead also to availability problems because of dropping all the packets sent from unknown senders.
SDN Master-Slave controller uses a replication scheme joined with a consistency check and a correction mechanism to avoid the single point of failure and to increase the performance when needed. Intrusion Tolerant Controller would be a good solution to enhance security, especially the availability of the SDN controllers due to the integration of the intrusion tolerance approach. However, the proposed solution could face a security problem with a very long exposure time and a performance problem with a very short exposure time.

## 5 Comparison Summary and Conclusion

**Table 1.** Functional Distinctions and Efficiency of ITS designs in SDN and Cloud Computing environments.

| **Attacks Handled** | **Complexity** | **In Which Technology used** |
|---|---|---|
| XSS, DoS SQL Injection Data ex-filtration | High | --- |
| Access violation Impersonation web defacing Data ex-filtration | High | --- |
| limits DoS, web defacing, and Data ex-filtration (Exposure short = Attack limited) | Low | --- |
| Volumetric DDoS | Medium | Cloud Computing |
| Stealthy resource exhaustion attacks DDoS attacks | High | Cloud Computing |
| Authentication and authorization attacks | High | Cloud Computing |
| DoS attacks | High | SDN |
| DoS attacks | High | SDN |
| DoS and DDoS attacks | Medium | SDN |

| **Ref N** | **Approach** | **Techniques** | **Performance Imppact** |
|---|---|---|---|
| SITAR [4] | Detection-Triggered | Redundancy, Diversity, Voting, Acceptance Test Dynamic Reconfiguration Indirection | Impact on Response Time |
| MAFTIA [5] | Algorithm-Driven | Redundancy, Diversity, Voting, Threshold scheme | Impact on Response Time |
| SCIT [6] | Recovery Based | Redundancy Diversity (Optional) | Computing cycles for starting a new server instance |
| ACT [7] | Recovery Based | Redundancy Diversity (Optional) | Computing cycles for cluster expansion and reduction |
| Hybrid Recovery-Based ITS [8] | Hybrid | Redundancy, Diversity, Indirection Dynamic Reconfiguration | Impact on Response Time |
| Framework for IT [9] | Algorithm-Driven | Redundancy, Diversity Reconfiguration, Voting Threshold Cryptography | Impact on Response Time |
| IT Through Flow Monitoring [10] | Detection-Triggered | Acceptance test Indirection | Impact on Response Time |
| Fault-Tolerant and Consistent SDN Controller [11] | Detection-Triggered | Redundancy Threshold scheme | Impact on Response Time |
| ITC [12] | Recovery Based | Redundancy , Diversity, MTD | Computing cycles for new controller |

In this paper, we have presented an analysis of several Intrusion Tolerant Systems in Cloud Computing and SDN. In sum, we have described each architecture, its characteristics, and its techniques used to tolerate intrusions, and have made a brief analysis compared to the different approaches proposed in [3]: Detection-Triggered, Algorithm-Driven, and Recovery Based, in a try to classify each approach.

However, the enhancement of intrusion tolerance in other environments like IoT, Fog Computing... and other SDN planes is still delayed. In fact, every environment has its own concept and characteristic, and its security issues. So adapting the best intrusion tolerance approach is a must.

Table 1 recapitulates the functional distinctions, techniques, and performance between all the ITS designs discussed above, applied either on Cloud Computing or on SDN. It gives, in addition, their intrusion tolerance efficiency against different types of attacks and their level of complexity.

## References

1. F. Wang, and R. Uppalli,C. Killian, MILCOM, 2, pp. 729–734, (2003)
2. I. Koren, C.M. Krishna, Elsevier/Morgan Kaufmann, (2020)
3. Q. Nguyen, A. Sood, IEEE Sec and Priv, 9, no 4, pp. 24-31, July-Aug. (2011)
4. D. Wang, Bharat B. Madan, and S. Kishor Trivedi, SSRS ’03, 23–32, (2003)
5. Paulo E. Veríssimo, Nuno F. Neves, C. Cachin, J. Poritz, D. Powell, Y. Deswarte, R. Stroud, I. Welch, IEEE Security and Privacy, (2006)
6. Y. Huang, D. Arsenault, A. Sood, ARES (2007)
7. J. Lim, Y. Kim, D. Koo, S. Lee, S. Doo, and H. Yoon, J. Supercomput., 66, 2, pp. 918–935, Nov. (2013)
8. B. Jang, S. Doo, S. Lee, and H. Yoon, IEICE Trans. Inf. Syst., E99.D, 4, pp. 1081–1091, (2016)

9. V. M. Karande, A. R. Pais, Advances in Computing and Communications, 193, Springer, pp. 386–395, (2011)
10. B. Manu, A.K. Koundinya, CSITSS, pp. 1–5. (2017).
11. Anjan K. Gonzalez, G. Nencioni, Bjarne E.Helvik, A. Kamisinski, GLOBECOM, pp. 1–6. (2016).
12. N. Sanoussi, K. Chetioui, G. Orhanou, S. El Hajji, Comp and Sec J, 132, (2023