Intrusion Tolerance Approaches in ITUA[1]

Michel Cukier, James Lyons, Prashant Pandey,
HariGovind V. Ramasamy, and William H. Sanders

Center for Reliable and High-Performance Computing
Coordinated Science Laboratory and
Electrical and Computer Engineering Department
University of Illinois at Urbana-Champaign
{cukier, jlyons, prashant, ramasamy, whs}@crhc.uiuc.edu

Partha Pal, Franklin Webber, Richard Schantz,
Joseph Loyall, Ronald Watro, and Michael Atighetchi

BBN Technologies
{ppal, fwebber, rschantz, jloyall, rwatro, matighet}@bbn.com

Jeanna Gossett

The Boeing Company
Jeanna. Gossett@MW.Boeing.com

1. Introduction

The purpose of the Intrusion Tolerance by Unpredictable Adaptation (ITUA) project is to develop a middleware based intrusion tolerance solution that would help applications survive certain kinds of attacks. This paper presents an overview and the key aspects of the ITUA project. We will describe the kind of attacks we are considering, how unpredictability can be used for intrusion tolerance, the architecture of the intrusion tolerant system we are envisioning, and the protocols we are developing to obtain a group communication system that tolerates the intrusions specified in the ITUA project.

2. The ITUA Model

The ITUA project is developing an *intrusion model* that defines a set of possible attacks against a computer system. Each attack in the set is said to be *covered* and an attack not in the set is *not covered* by the model. The attacks covered by the model are the attacks the ITUA project is concerned about defending against. The model identifies some important, but abstract, features of attacks and of their environment.

In particular, the system under attack consists of a set of *security domains*. A security domain implements a boundary that attackers have difficulty crossing. For example, a host can be a domain if an attacker with privileges on one host is not automatically granted privileges on other hosts. Another example of a domain is a LAN with firewalls that insulate it from other LANs.

The attacker proceeds by infiltrating security domains and by corrupting processes within those domains. In the worst case, a corrupt process can behave arbitrarily, so that corruption is equivalent to a Byzantine failure.

A key assumption in the ITUA model is that only *staged* attacks are covered. In a staged attack, the attacker infiltrates some domains before others. This might happen because the attacker needs time to explore the topology of interconnections between security domains, or because domains differ and attacks on them take unpredictable amounts of time. The assumption of staged attacks gives the defense some time in which to react to an attack.

Another assumption is that intrusion detection is possible but imperfect. An assumption about intrusion detection is necessary because a corrupt process can behave arbitrarily. Without detection, an attacker could infiltrate domains in stages, silently corrupting processes, then suddenly cause every process to stop simultaneously. With detection, the defense has some probability of being able to react.

One of the main uses of the ITUA model will be the validation of defensive strategies and algorithms. An effective defense is one that increases the probability and/or the length of time that an application survives an attack. Based on the ITUA model, a key contribution of the ITUA project is to provide unpredictability at different levels of adaptation.

3. Unpredictability in Intrusion Tolerance

Adaptation is crucial for intrusion tolerance. In order to survive an attack, applications must adapt to the damaged environment and to changes in the quality and availability of resources. Recovering from partial failures caused by an attack involves adaptive response. However, in the case of sophisticated attacks carried out in multiple stages, the resilience provided by adaptation can easily be circumvented by an adversary who can predict the adaptive response. For that reason, the ITUA project is proposing to add uncertainty in its intrusion tolerance technology so that the adaptive responses become unpredictable to the attacker.

The ITUA intrusion tolerance technology aims to demonstrate that both application-level adaptive behavior and adaptive responses involving resource allocation and reconfiguration can be made unpredictable to an observer. At the application level, this may mean that an object will decide to communicate with a remote object other than the one the attacker was expecting. One example involving resource reallocation and reconfiguration would be a case in which an attacker has killed a replica on a host, and the replacement replica is started on a host that is chosen in a nondeterministic manner; that would make it harder for the attacker to find and attack the replica as his next step.

4. ITUA Architecture Overview

The architecture for the ITUA infrastructure consists of replicas, managers, and subordinates located in various groups. This section will detail the main components of the ITUA architecture.

The ITUA intrusion tolerance mechanism supports both in-band and out-of-band adaptation. With in-band adaptation, inter-object interaction is intercepted and application-level behavior is altered. This is managed by the QuO [Loy98] adaptive middleware. With out-of-band adaptation, intrusion response and recovery actions that involve manag-

[1] This research has been supported by DARPA Contract F30602-00-C-0172.

ing and configuring system resources are taken independently of the application's inter-object interaction. The ITUA project is implementing a decentralized infrastructure to manage this kind of adaptation, consisting of components known as managers and subordinates.

Each host runs either a manager or a subordinate. Each security domain has one host that runs a manager; the rest of the hosts in that domain run subordinates. All managers form a group called the *manager group*, and all the subordinates in a domain and the manager of that domain form a group called a *subordinate group*. A subordinate's two principal responsibilities are *security advising* and *replication management*. The target of the actions and decisions taken as part of these are always local, i.e. they involve resources associated with the host on which it runs and its network interfaces. However, in some cases (for instance, changing the tolerance mode of a replication group), the impact may ripple through multiple domains, in which case multiple managers and subordinates cooperate. A manager performs all the responsibilities of a subordinate, and also it has some domain-wide responsibilities.

In the *security advisor* role, a subordinate makes use of multiple local sensor-actuator loops to 1) collect information about potential intrusions and anomalous events, 2) make a quick "knee jerk" local reaction to the observed event, and 3) provide the security domain manager and the local replication management operation with host specific information.

In the *replication management* role, subordinates are responsible for starting or stopping replicas of application components. A subordinate may determine, in its security advisor role, that the local host is under attack, and then consequently modify, in its replication management role, the tolerance mode of some groups that have replicas on that local host. Note however, that even though this a local decision, it affects other hosts in other domains too. Changing the tolerance mode of a replication group involves agreement of the members of that group, which involves cooperation of multiple managers and subordinates. The local action merely initiates that process. In order to switch effectively between tolerance of crash and Byzantine failures, the ITUA project is extending the Ensemble [Hay98] group communication system.

5. Group Communication System

Ensemble is a group communication system that tolerates crash failures. It consists of several protocol layers; their combination leads to various protocol stacks. In order to be used in the context of the ITUA project, Ensemble needs to tolerate the failures specified in the ITUA model. We have focused on two important protocols to include the changes in order to tolerate these new failures: the group membership and total-ordering protocols.

5.1 Group Membership Protocol

The Group Membership Protocol (GMP) implements the algorithms for group formation, for the joining of new processes to the group, and for the removal of faulty processes from the group. It maintains information about the current group membership, and attempts to ensure that this information reflects the true physical situation and is identical across all processes in *G*.

If we consider a process group *G* consisting of *n* processes $\{p_1, p_2, p_3 \ldots p_n\}$, our protocol ensures that *G* can tolerate up to $f = \lfloor (n-1)/3 \rfloor$ faults. A process may be removed from *G* if its crash has been detected, if it is slowing down the protocol, or if it is exhibiting "provably bad behavior". In the first two cases, it may be allowed to rejoin *G* at a later time, but in the third case, it is classified as Byzantine and never allowed to rejoin *G*.

There are two main types of messages that can be sent by a process to other processes in the group:

- *GMP-Messages*, which are related to the GMP and contain changes to the membership.
- *Normal Messages*, which are related to the application execution.

The GMP makes use of a reliable broadcast protocol that ensures that a *GMP-Message* broadcast in *G* eventually gets delivered to all non-faulty processes in *G*.

5.2 Total-Ordering Protocol

In the ITUA project, a group of replicas have to maintain consistent state. So, the exchange of information between the replicas must be totally ordered. For this reason we have come up with a total ordering protocol for group multicasts which can be made very efficient in the special case where the group of communicating processes are replicas.

Each group member p_i is associated with a sequence number generator function f_i and a unique initial sequence number s_i. Process p_i generates a new sequence number for a message it wants to broadcast by starting with s_i and applying f_i to the last sequence number p_i had generated. Messages are delivered by processes in the order of increasing sequence numbers. The sequence number generating functions should satisfy the following properties:

- Given a unique assignment of s_i's to each process, the f_i's should generate a group of pair-wise disjoint sets of sequence numbers.
- The universal set of sequence numbers should be the same as the set of all sequence numbers generated by the generating functions of all the group members.
- The distribution of sequence numbers as defined by this set of functions should be closely related to the actual traffic generated by the processes.

6. Current Status

We are currently engaged in the final phase of the design of the overall ITUA architecture. We have started implementing some components: modification of QuO to provide unpredictability, communication infrastructure for the managers and subordinates, and modification of the group membership and total-ordering protocols to tolerate the failures specified in the ITUA model.

More information on the ITUA project can be found at http://www.dist-systems.bbn.com/Projects/ITUA and http://www.crhc.uiuc.edu/PERFORM.

References

[Hay98] M.G. Hayden, "The Ensemble System," Ph.D. thesis, Cornell University, 1998.

[Loy98] J.P. Loyall, D.E. Bakken, R.E. Schantz, J.A. Zinky, D.A. Karr, R. Vanegas, and K.R. Anderson, "QoS Aspect Languages and Their Runtime Integration," *Lecture Notes in Computer Science*, vol. 1511: *Proc. Fourth Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers (LCR98)*, May 1998, Pittsburgh, PA. Springer-Verlag.