FOREVER: Fault/intrusiOn REmoVal through Evolution & Recovery*

Alysson Bessani, Hans P. Reiser, Paulo Sousa
University of Lisbon, Portugal
{bessani,hans,pjsousa}@di.fc.ul.pt

Ilir Gashi, Vladimir Stankovic
City University, UK
{ec233,ek274}@csr.city.ac.uk

Tobias Distler, Rüdiger Kapitza
University of Erlangen-Nürnberg, Germany
{distler,rrkapitz}@cs.fau.de

Alessandro Daidone
Università di Firenze, Italy
daidone@dsi.unifi.it

Rafael Obelheiro
Universidade do Estado de Santa Catarina, Brasil
rro@joinville.udesc.br

ABSTRACT

The goal of the FOREVER project is to develop a service for *Fault/intrusiOn REmoVal through Evolution & Recovery*. In order to achieve this goal, our work addresses three main tasks: the definition of the FOREVER service architecture; the analysis of how diversity techniques can improve resilience; and the evaluation of the FOREVER service. The FOREVER service is an important contribution to intrusion-tolerant replication middleware and significantly enhances the resilience.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems; D.4.5 [**Operating Systems**]: Reliability; C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Replication, Intrusion Tolerance, Proactive Recovery

1. INTRODUCTION

Threats evolve during system lifetime "because attackers are actively involved in the development of new techniques to inject and, or, activate latent faults in existing systems" [1]. This means that resilient systems need also to evolve in order that attacks do not lead to system failures.

*This work was supported by the EU through NoE IST-4-026764-NOE (RESIST/FOREVER) and by the FCT through the Multiannual Program (LASIGE).

Middleware'08 Companion December 1-5, 2008 Leuven, Belgium

The ideal goal of evolution is the complete removal of vulnerabilities (thus eliminating any chances of an attack causing a failure), but it is well known that such goal is very difficult, if not impossible, to achieve. Nevertheless, one can minimize the number of vulnerabilities by applying security patches to operating systems or by introducing newer (better) versions of the application code. The remaining vulnerabilities may be targeted by attacks and produce failures or intrusions. A resilient system needs to deal with such faults/intrusions, which may be masked through fault-/intrusion tolerance protocols. These protocols are typically run on replicated systems (with each replica redundantly executing client requests), and are able to tolerate the failures of a finite set of f replicas.

Given a sufficient amount of time, a malicious adversary can find ways to compromise more than f replicas. Therefore, if one wants to build a resilient system that is continuously operating, some sort of recovery mechanism will need to be added. The goal would be to detect and recover compromised replicas at a pace faster than the time needed by an adversary to compromise more than f replicas. Arbitrary faults are, however, very difficult to detect. An alternative approach is to calculate the minimum time needed by an adversary to compromise more than f replicas and (proactively) trigger periodic replica recoveries at a faster pace [2]. Note that the time needed to compromise more than f replicas is highly dependent on how diverse replicas are.

The goal of the FOREVER project[1] is to address some of the challenges that arise in this situation by developing a *Fault/intrusiOn REmoVal through Evolution & Recovery* service. This service can be used to enhance the resilience of replicated systems, namely those that can be affected by malicious attacks. In order to achieve this goal, our work pursues three tasks: It defines the FOREVER service architecture, it analyses how diversity may be introduced both in space (replicas) and in time (recoveries), and it evaluates the FOREVER service with stochastic modelling.

The remainder of the paper is organized as follows. The next section describes the FOREVER service. Section 3 discusses diversity management. Section 4 explains our ap-

[1] `http://forever.di.fc.ul.pt/`

proach to evaluate the FOREVER service architecture. Finally, Section 5 concludes.

2. FOREVER SERVICE ARCHITECTURE

Removing faults (or malicious intrusions) requires a timely and reliable recovery mechanism. This recovery support cannot be directly integrated in a traditional middleware layer between operating system and replicated application, as a malicious intruder could easily disable the recovery. A recovery service needs to be implemented in a trusted computing base (TCB).

The simplest recovery procedure is to boot a clean image containing the original operating system and application(s) code, and to obtain the current state from the remaining replicas. Clearly this technique removes the effects of any faults/intrusions that could have occurred before the recovery. However, the adversary may have acquired knowledge before the recovery (e.g., the password of some user, the version of the operating system) sufficient to deploy a more advanced attack after the recovery. Following this reasoning, the adversary may accumulate knowledge over time (days, weeks, months) until it is able to compromise more than f replicas between recoveries. This means that diversity in the space domain should be complemented with diversity in the time domain: recoveries should introduce diversity.

The main objective of the FOREVER service architecture is to extend and generalize previous work on proactive/reactive recovery [3] in order to support application and OS online upgrades (i.e., evolutions), online recovery [4] and self-monitoring. The specific objectives are as follows:

- to allow online upgrades of applications (e.g., installation of new versions), middleware, and operating systems (e.g., installation of security patches). These upgrades may need to be coordinated if they are incompatible with the previous (application or OS) version and the coordination should be done in a way that the impact on availability is minimal;
- to integrate existing work on intrusion detection based on monitoring with virtual machine technology. This allows us to improve the capabilities of (application) self-monitoring and intrusion detection, triggering recoveries when needed.

Based on our previous work [3, 4], we have identified two variants of the FOREVER system architecture, each using a hypervisor (Xen in our prototype) to provide a TCB for the FOREVER service. In the first variant, only the FOREVER service is executed within the TCB, and the remaining replication functionality is part of the middleware in the application domain. The second variant moves more functionalities (group communication, voting, state transfer support) to the TCB. The benefit of simpler and more efficient replication mechanisms comes at the expense of an increased TCB size.

3. DIVERSITY MANAGEMENT

Diversity of replicas is a critical issue, because the system must avoid that an adversary obtains control over more than f replicas simultaneously. The main objective of FOREVER is to analyze how diversity may be introduced effectively both in space (replicas) and in time (recoveries), in this way modifying the vulnerabilities that may be exploited by a malicious adversary.

We intend to use COTS (Components-off-the-shelf) diversity for FOREVER, i.e., each replica uses a different software configuration (OS, JVM, middleware). Our current approach uses the National Vulnerability Database[2] to statistically quantify the diversity of COTS software. While not completely fault independent, our preliminary study indicates that, for example, different COTS operating system families (Linux, *BSD, Solaris, Windows) have only few vulnerabilities in common. This study also indicates the probability of common-mode failures, which we can later use to analytically model the system, in order to quantify if COTS software is sufficiently diverse.

We are currently also studying mechanisms (e.g., instruction-set randomization, address-space randomization) and develop novel heuristics to generate diversity (in either application/OS configuration or runtime environment). These techniques, which may include the modification of OS access passwords, protocols, open ports, authentication method etc., help to increase replica diversity in time.

4. EVALUATION

In ongoing work, we will evaluate the FOREVER service and assess its dependability properties. Models will take into account Byzantine fault-tolerant (BFT) replication, recoveries and evolution (diversity), using as a basis real data from the vulnerability database study mentioned above.

Evaluations will quantify the overhead of enhancing a replicated system with the FOREVER service and evaluate the benefits and the trade-off of introducing diversity in the recovery process.

5. CONCLUSIONS

The FOREVER project is work in progress that makes significant contributions to middleware infrastructures for intrusion-tolerant replication. The FOREVER service architecture allows online upgrades and diversity management of replicas. Future system evaluation will use data from the diversity study to analytically verify the benefits of the proposed service architecture. The FOREVER service thus allows enhancing the resilience of intrusion-tolerant systems.

6. REFERENCES

[1] L. Strigini, P. Palanque, A. Moorsel, A. Pataricza, and M. Banatre. From resilience-building to resilience-scaling technologies: Directions – ReSIST NoE Deliverable D13. DI/FCUL TR 07–28, Department of Informatics, University of Lisbon, 2007.

[2] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks (extended abstract). In *PODC '91: 10th ann. ACM Symp. on Principles of Distributed Computing*, pages 51–59, 1991.

[3] P. Sousa, A. Bessani, M. Correia, N. F. Neves, and P. Verissimo. Resilient intrusion tolerance through proactive and reactive recovery. In *PRDC '07: 13th IEEE Pacific Rim Int. Symp. on Dependable Computing*, pages 373–380, 2007.

[4] H. P. Reiser and R. Kapitza. Hypervisor-based efficient proactive recovery. In *SRDS'07: 26th IEEE Symp. on Reliable Distributed Systems*, pages 83–92, 2007.

[2] `http://nvd.nist.gov/`