Fragmentation-Redundancy-Scattering : a means to tolerate accidental faults and intrusions in distributed systems

Yves Deswarte

LAAS-CNRS & INRIA

7, avenue du Colonel Roche
31077 Toulouse Cedex
France

Tel: +33 / 61 33 62 88
Fax: +33 / 61 33 64 11
e-mail: deswarte@laas.fr

Abstract

Many distributed systems have been designed to tolerate accidental faults because distribution enables isolation of elements so that error propagation can be prevented or limited. The same approach can be applied to tolerate not only accidental faults, but also intentional operational faults, i.e. intrusions. A distributed system is intrusion-tolerant if it is designed so that any intrusion into a part of this system will not endanger confidentiality, integrity and availability. By intrusion, we mean not only computer break-ins by non-registered people, but also attempts by registered users to exceed or abuse their privileges. in particular, possible malice of security administrators is taken into account.

Fragmentation-Redundancy-Scattering is a technique which enables to recover erroneous data destroyed or contaminated by accidental faults or by intrusions, while preserving the confidentiality of sensitive data. This technique has been applied to different functions of distributed systems such as persistent file storage, security management and data processing.

Introduction

Most of the currently developed secure systems are based on paradigms such as *access control matrix*, *reference monitor*, *security kernel* or *trusted computing base* concepts. These concepts are essentially centralized, in order to keep their implementation simple and verifiable. Such a centralized approach is inconsistent with distribution, local autonomy and concurrency that distributed systems are supposed to provide. Moreover, a centralized implementation of these concepts would constitute a "single point of failure", with respect both to accidental faults (a single site failure can produce a denial of service for the whole distributed system), and to intrusions (a successful intrusion into a single site is sufficient to annihilate the security of the whole distributed system). In addition, administrators may be targets for bribery.

In our approach, the required trust comes from the cooperation and the consensus of a majority of security entities. Each entity can be individually untrusted, as long as a majority of them can be trusted. Therefore, an intrusion into a part of the system will have no consequence on the system security if only a minority of the security entities is affected by the intrusion : this approach is thus "intrusion-tolerant".

More precisely, let us consider a distributed system composed of standard workstations, the *clients*, and of

intrusion-tolerant distributed *servers*. Each server is constituted by a set of untrusted sites, the server being trusted as a whole. That means that an intrusion into the distributed server sites should not endanger the confidentiality and integrity of the sensitive data stored or processed by the server, and should not produce any denial of service: to be successful, an attacker would have to intrude into a majority of the server sites, or bribe a majority of the site administrators. This approach can be envisaged for application servers, such as file servers or data processing servers, as well as for specific security servers, responsible for user authentication and for authorization (i.e. control of access to application servers).

1 Intrusion tolerance

By intrusion we mean a large class of attacks, covering not only computer break-ins by external attackers, but also illegitimate use by registered users [3]. Such intrusions can be classified according to the intruder privileges, or according to the intrusion targets. Intruders can be:

- *external intruders*, i.e. who are not registered as users of the computing system; thus, they have to deceive or by-pass the authentication and authorization mechanisms, or
- *internal intruders*, i.e. who are registered as legitimate users, but who:
  - try to exceed their privileges, for instance by trying to read confidential data or modify sensitive information for which they have no authorized access: to do this, they have to by-pass the authorization mechanisms, or
  - abuse their privileges for some illegitimate (but authorized) actions; for instance a security officer *can* (but *should not*) take malicious actions, or an operator *can* (but *should not*) halt a computer at some inappropriate instant, causing a denial of service.

Intrusion targets can be:

- information confidentiality: the intrusion attempts to disclose confidential information,
- information integrity: the intrusion attempts to create false information or to alter or destroy sensitive information,
- service availability: the intrusion attempts to prevent legitimate users from using the system (denial of service).

2 Fragmentation-Redunadancy-Scattering

In *dependability* terminology [7], intrusions are *intentional operational external faults*, i.e. one particular class of faults. Classically, dependability is obtained by a mixture of fault prevention and fault tolerance. This dual approach can also be applied to intrusions, i.e. we can consider intrusion prevention and intrusion tolerance. Intrusion prevention is the aim of most of physical and logical access control mechanisms. When one considers the possibility that such mechanisms can be defeated, intrusion tolerance techniques can be contemplated.

Our approach to intrusion-tolerance takes advantage of the geographic distribution of a distributed system in order to achieve confidentiality and integrity of sensitive information, and availability for the service. One of the techniques we propose for this purpose is the *fragmentation-redundancy-scattering* technique [2], which consists in first cutting all sensitive information into fragments, in such a way that any isolated fragment does not contain significant information, and second scattering the fragments among the different sites of the distributed system, so that an intrusion into a part of the distributed system give access only to unrelated fragments. Redundancy is added to the fragments (e.g. by replication) in order to tolerate accidental or deliberate destruction or alteration of fragments. The following chapters show how this technique can be applied to different functions.

3 Distributed security management

The fragmentation-redundancy-scattering has been applied to distributed security servers, responsible for user registration and authentication, as well as for authorization (i.e. access control) [1].

3.1 User registration and authentication

An intrusion-tolerant registration and authentication server must tolerate all the kinds of intrusions which have been presented in section 1. This can be done by the use of distribution among a set of authentication sites. Thus, availability of the authentication function can be achieved in spite of failures of one or a minority of sites. Another characteristic is that each site can be managed by a different security administrator: this enables intrusions by a minority of malicious security administrators to be tolerated. Thus, the authentication server can be viewed as composed of a trusted set of

untrusted authentication sites managed by a trusted set of untrusted security administrators.

Registration of a new user in the system must be performed under the control of the security administrator at each authentication site. The user is registered at the first site by the first security administrator, at the second site by the second administrator and so on. At each site, the identity of the user is stored with some authenticator which will be used by the authentication process to verify the claimed identity. The authenticator can be a password, a secret permanent key or a public key. However, except if public keys are used, different authenticators have to be stored on the different authentication sites for the same user. This can be considered as an implementation of the concept of *separation of duty*: one or a minority of security administrators cannot register an illegitimate user and cannot prevent the registration of a legitimate user at a majority of authentication sites. Moreover, if such malicious administrators try to use local information stored on their sites in order to impersonate a registered user, they will fail because they cannot be authenticated by the other sites.

When a registered user wants to access remote servers from a workstation, he/she has to be authenticated by the authentication server: an authentication protocol has to be run between the user site and the authentication sites. This protocol is composed of three phases. In the first phase, the user site attempts an independent authentication with each of the authentication sites. This authentication can be based on classical authentication algorithms, but with different authenticators (or different challenges with public key systems) for the different authentication sites. In this first phase, each authentication site independently decides, for itself, whether or not the authentication attempt succeeds. During the second phase, each authentication site broadcasts its individual decision to all the other authentication sites, and receives the decisions of these other sites. In the third phase, according to the majority of all the received decisions and its own decision, the authentication site authorizes (or not) the session for the user, and confirms this session authorization by sending to the user site a session key (or a ticket containing the session key, depending on the authentication algorithm which has been selected). A different session key is randomly generated by each authentication site. This session key or ticket will be used by the user site to authenticate its requests in the authorization process.

In this protocol, majority voting on the different authentication decisions enables the system to tolerate accidental faults affecting the registration data stored on the different authentication sites or communication faults during the protocol, as well as to tolerate intrusions into a minority of authentication sites which could lead to false local decisions. Moreover, different session keys are generated independently by each authentication site, and sent only to the user site; thus, no intruder, even with the complicity of an administrator, can impersonate the user site (except if he/she breaks the classical authentication algorithms).

3.2 Authorization

The aim of this section is to describe a distributed, intrusion-tolerant authorization server. This server has to store and manage access-rights and to grant or deny user accesses to remote application servers. The server is made intrusion-tolerant using distribution among a set of authorization sites and the application of the following techniques: replication, secret sharing and agreement.

An authorization server has to *check* the rights of users who wish to access remote servers or objects. It also *manages* all user rights for all objects which could be accessed and which need to be protected. When a new object is created by a user or when a new server is installed, the authorization server stores the access rights, the reference (information which enables direct access to the object) and other information, which together form the object or server descriptor. The access right management must obey an *authorization policy*, which has to be implemented by the authorization server.

In our approach, the authorization server is responsible for access right checking and access right management, and when an access from a user site to an object is granted, tickets are distributed by the authorization server to the user site and to the server managing the object, so that the user site can directly access the server for all its requests to that object: when the application server receives a request, it has only to check that this request is allowed by the corresponding ticket.

The authorization protocol is quite similar to the authentication protocol: whatever the user request, first a local decision is taken by each authorization site according to the user access rights which are locally stored; then this local decision is broadcast to the other authorization sites; the authorization decisions received from the other sites are voted on locally (together with

the local decision), and, according to the result of the vote, the user request is locally executed or not. This majority voting technique ensures that a legitimate request cannot be denied and an illegitimate request cannot be granted, unless a majority of authorization data copies have been destroyed or altered.

4 Persistent file server

The application server presented here is a persistent file server, i.e. a server which stores the user files between user sessions [4, 5].The fragmentation and scattering technique when applied to file storage involves cutting every sensitive file into several fragments in such a way that one or several fragments (but not all) are insufficient to reconstitute the file. The level of information granularity is such that the contents of one or several fragments together do not disclose any significant information. The fragments are stored in several copies on different geographically distributed sites, which can be viewed as fragment server machines (see figure 1).

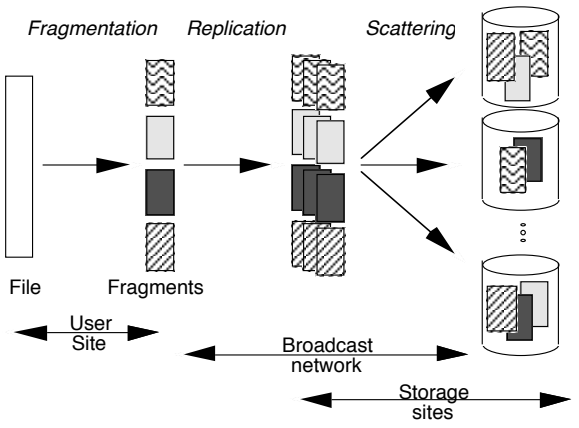

Figure 1: Fragmentation-and-scattering applied to persistent file storage

Conclusion

The intrusion tolerance approach looks very promising for open distributed systems whose elements cannot be all trusted. In particular, the intrusion-tolerant authentication and authorization servers enables a consistent security policy to be implemented on a set of heterogeneous, untrusted sites, administrated by untrusted (but non-conspiring) people.

A prototype of the persistent file server presented in this paper has been successfully developed and implemented as part of the Delta-4 project of the European ESPRIT programme. An intrusion-tolerant security server, gathering the authentication function and the directory and authorization function, is currently being developed for the Delta-4 project; completion of a prototype is planned for the end of 1991. In parallel, application of the fragmentation-redundancy-scattering technique to data processing is explored [6, 8]: in that case, process parallelism and distribution are used to prevent information disclosure, while process replication is used to ensure availability and integrity.

Acknowledgements

The author is grateful to Jean-Claude Laprie, David Powell and Joni Fraga for the original principles of the fragmentation-redundancy-scattering technique as well as to Laurent Blain, Jean-Charles Fabre, Jean-Michel Fray, Pierre-Guy Ranéa and Gilles Trouessin for their contribution to the development of this technique. The work presented here has been developed by the Saturne project, which is a joint project of INRIA and LAAS-CNRS. This research is partly supported by the PDCS (Predictably Dependable Computing Systems) project n°3092 of the European ESPRIT programme, while some aspects have been implemented as part of the Delta-4 (Definition and Design of an open Dependable Distributed architecture) project n°2252 of ESPRIT.

References

[1] Blain L. and Deswarte Y., "An intrusion-tolerant security server for an open distributed system", *Proc. European Symp. in Computer Security (ESORICS 90)*, Toulouse (France), Oct. 1990, pp. 97-104

[2] Deswarte Y, Blain L., Fabre J.-C., " Intrusion Tolerance in Distributed Systems", *Proc. IEEE Symp. on Security and Privacy*, Oakland (Ca.), May 1991, pp. 110-121

[3] Denning D.E., "An intrusion-detection model", *Proc. IEEE Symp. on Security and Privacy*, Oakland (Ca.), April 1986, pp. 118-131

[4] Da Silva Fraga J. and Powell D., "A Fault- and Intrusion-Tolerant File System", *Proc. 3rd Int. Conf. on Computer Security, IFIP/SEC'85*, Dublin (Ireland), Aug.1985, pp. 203-218.

[5] Fray J.M., Deswarte Y. and Powell D., "Intrusion-tolerance using fine-grain fragmentation-scattering", *Proc. IEEE Symp. on Security and Privacy*, Oakland (Ca.), April 1986, pp. 194-201.

[6] Fray J.M. and Fabre J.C., "Fragmented Data Processing: an Approach to Secure and Reliable Processing in Distributed Computing Systems", *Proc. 1st IFIP Int. Conf. on Dependable Computing for Critical Applications (DCCA)*, Santa Barbara (Ca.), Aug.1989, pp. 131-137.

[7] Laprie J.C., "Dependability: a unifying concept for reliable computing and fault-tolerance", in *Dependability of Resilient Computers*, BSP Professional Books, Oxford (UK), Ed. T. Anderson, 1989, pp.1-28

[8] Trouessin G., Fabre J.C. and Deswarte Y., "Reliable processing of confidential information", Proc.7th Int. Conf. on Computer Security, IFIP/SEC'91, Brighton (UK), 15-17 May 1991.