On the Coverage of Proactive Security: An Addition to the Taxonomy of Faults

Timo Warns

Carl von Ossietzky University of Oldenburg
Software Engineering Group
Graduate School *Trustworthy Software Systems*

Abstract:
Intrusion tolerance is a recent approach to deal with intentional and malicious failures. It combines the research on fault tolerance with the research on security, and relies on the means of proactive security. The development of a fault-tolerant system requires the explication of its underlying fault assumptions. In this context, we propose a viewpoint on faults that discriminates whether they are timely exploitable and, therefore, are not covered by proactive security.

1 Introduction

Proactive security has become one of the most important means for the development of intrusion-tolerant systems. Intrusion tolerance is a recent approach to deal with intentional and malicious failures. It can be considered as a special case of fault tolerance as its main idea is to prevent system failures by tolerating single even undetected intrusions with means of redundancy. It especially addresses security related threats and, therefore, is located at the borders of research on fault tolerance and security.

The development of a fault-tolerant system requires to specify the underlying fault assumptions explicating the type and number of faults to tolerate. Hence, the specification process includes the modelling of faults often based on fault taxonomies. Avižienis et al. proposed such a taxonomy with *elementary fault classes* [ALRL04]. From our point of view, the modelling of faults is one of the most important means to avoid unnoticed vulnerabilities.

Sousa et al. recently identified a previously unnoticed vulnerability within a number of current intrusion-tolerant systems [SNV05]. The vulnerability is caused by the systems' asynchronous model allowing to violate assumptions of proactive security. As our contribution, we abstract from such vulnerabilities and identify a class of

This work is supported by the German Research Foundation (DFG), grant GRK 1076/1.

faults related to the timeliness of exploitation that is not covered by proactive security. We present a new viewpoint and two elementary fault classes for the taxonomy of Avižienis et al. [ALRL04] as it does not cover the identified fault classes yet.

The paper is organised as follows. First, we summarise the approach of proactive security in section 2 considering proactive secret sharing as an example. The fault taxonomy of Avižienis et al. [ALRL04] is briefly introduced in section 3. We present the viewpoint with its fault classes in section 4 and conclude in section 5.

2 Proactive Security

Proactive security is an approach to tolerate long-term attacks by means of distribution of trust and periodic refreshment [CGHN97]. Its main idea is to distribute and periodically refresh items of a system, e.g., pieces of a secret or software components, to remove the effects of intrusions. Usually, the choice, which items to refresh, does not depend on prior intrusion detection. Due to its advantages, it has been suggested for various contexts, e.g., secret sharing and recovery of program code of components. Exemplarily, we describe proactive secret sharing.

Secret sharing is a cryptographic technique to attain the security of data. A secret sharing scheme is a (k, n)-threshold scheme, which divides a sensitive data item into n shares in a way that k shares are required to reconstruct the original item, but less than k shares do not reveal any information about the item [Sha79, Bla79]. Distributing the shares to different components with independent failure modes enables a system to tolerate up to $k-1$ intrusions during the lifetime of the secret.

The lifetime of the secret may be undesirably large as vulnerability window to tolerate component failures. A solution is to periodically refresh the shares as proposed by Ostrovsky and Yung [OY91]. Proactive secret sharing "renew[s] the shares without changing the secret, in such a way that any information learned by the adversary about individual shares becomes obsolete after the shares are renewed" [HJKY95]. Hence, an adversary needs to break into more than $k-1$ components within the period between two renewal phases. The vulnerability window is shrinked from the lifetime of the secret to the span between refreshments. Furthermore, proactive secret sharing is used to recover lost or corrupted shares without violating the confidentiality of other shares.

3 Fault Taxonomy

Fault-tolerant systems avoid failures by means of redundancy. Avižienis et al. state that the "class(es) of faults that can actually be tolerated depend(s) on the fault assumption that is being considered in the development process [...]" [ALRL04].

They classify faults according to a taxonomy with eight basic viewpoints leading to *elementary fault classes* as shown in figure 1.

The fault classes discriminate faults according to (a) their phase of creation or occurrence (b) the place of their origin (c) their phenomenological cause (d) whether they affect hardware or software (e) whether they were introduced maliciously (f) whether they were introduced intentionally (g) the capabilities of their originators (h) and their persistence. We note that the classes do not address time properties regarding exploitation. There is no distinction between faults that are exploitable in a timely manner and those that are not.

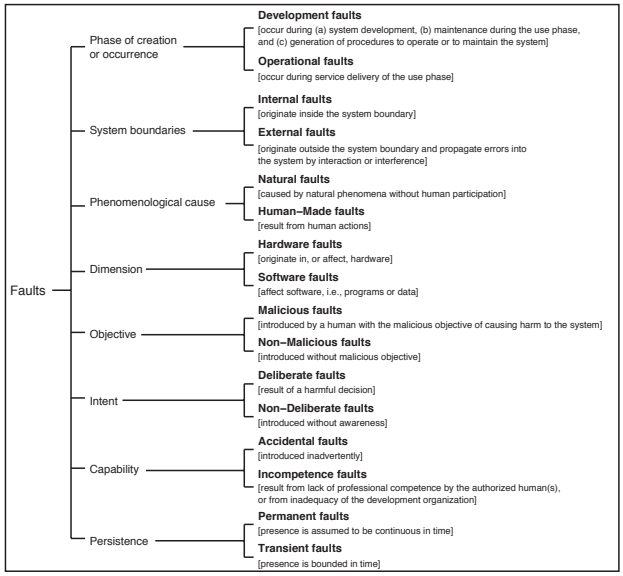

Figure 1: The elementary fault classes [ALRL04]

4 Timely Exploitation

Proactive security tolerates intrusions under the assumption that their number is bounded within a vulnerability window. For example, proactive secret sharing with a (k, n)-scheme keeps the confidentiality of the sensitive data item as long as an adversary intrudes less than k components with shares between two refreshment phases.

Sousa et al. found a vulnerability within a number of current intrusion-tolerant systems that rely on an asynchronous model [SNV05]. They showed that the vulnerability window (defined in terms of time) cannot be fixed in such systems, because no assumptions regarding time can be made with a purely asynchronous model. They presented an attack that violates the tolerance bounds of proactive security

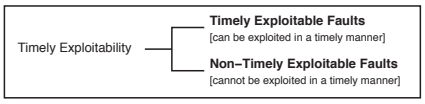

Figure 2: Viewpoint *Timely Exploitability*

due to enlargement of the vulnerability window. In other words, they were able to exploit too many vulnerabilities in a timely manner.

The components of an intrusion-tolerant system are required to handle a number of normal service invocations greater or equal to the bound for intrusions to tolerate. For example, the components participating in a (k, n)-scheme must be able to handle k invocations between two refreshment phases at least, because otherwise an authorized client would not be able to reconstruct the original data item by collecting k shares at all. Hence, the period between refreshments is bounded below.

This limitation opens the way for another kind of intrusions enabling the adversary to exceed the number of faults tolerated by proactive security. If the time complexity of the exploitation is below a certain bound, the adversary can carry out more intrusions within the vulnerability window than tolerated. The time complexity of exploitation depends on various factors that may change over time. For example, acquisition of knowledge by the adversary may enable him to reduce the time complexity of his attacks. Thus, the adversary can produce a system failure even if the vulnerability window is fixed.

This kind of faults can be illustrated with a common type of vulnerabilities: software flaws leading to *buffer overflows*. A buffer overflow is a failure activated by maliciously altered input overwriting data beyond the end of a buffer, which is supposed to accept the input. Such flaws require a certain amount of knowledge to be exploited, e.g., the adversary needs to know at least the service containing the flaw and the structure of the input required for a successful attack. However, if the attacker gathers the knowledge, the buffer overflow is usually easy to activate. Often the time complexity of buffer overflows equals the time complexity of normal invocations of a service. Hence, an attacker is able to carry out a number of buffer overflows equal to service invocations of a normal client exceeding the bounds of proactive security.

Proactive security only tolerates intrusions that do not violate its tolerance bounds by timely exploitation. As the development of fault-tolerant systems requires to specify which types of faults are tolerated, timely exploitation needs to be one viewpoint of a fault taxonomy. As this is not a part of the taxonomy of Avižienis et al. yet, we propose a new viewpoint with two fault classes as illustrated in figure 2.

The viewpoint *Timely Exploitability* addresses the timely exploitation of a fault. A *timely exploitable fault* is a fault that can be exploited in a timely manner allowing to violate tolerance bounds. The violation may be caused due to an enlargement of

a vulnerabilility window or an unexpected short time period of an successful attack. It is notable that the decision to which of both classes a fault belongs depends on its context. A software flaw within a component may be timely exploitable within one system, but is not in another one. Examples for timely exploitable faults are insecure default configurations and software flaws leading to buffer overflows.

5 Conclusion

We identified a viewpoint and two elementary fault classes that were not covered by the taxonomy of faults previously. They address the timely exploitation of faults and support the discrimination of vulnerabilities that are tolerated by proactive security and those that are not. We consider this addition helpful to reason about intrusion-tolerant systems and their capabilities.

However, we are aware that the methodology of fault tolerance requiring to specify fault assumptions and coverage raises many questions when applied to security related threats. For example, security fault models are very complex due to heterogenous aspects, e.g., human factors and covert channels. From our point of view, it is an open question how to model such security related threats thoroughly.

References

[ALRL04] A. Avižienis, J.-C. Laprie, B. Randell, and C. E. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[Bla79] G. R. Blakley. Safeguarding Cryptographic Keys. In *Proc. AFIPS 1979 National Computer Conference*, pages 313–317. AFIPS, 1979.

[CGHN97] R. Canetti, R. Gennaro, A. Herzberg, and D. Naor. Proactive Security: Long-term protection against break-ins. *RSA Laboratories' CryptoBytes*, 3(1), 1997.

[HJKY95] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *Proc. of the 15th Annual International Cryptology Conference on Advances in Cryptology*, pages 339–352, London, UK, 1995. Springer-Verlag.

[OY91] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks (Extended Abstract). In *Proc. of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 51–59, New York, USA, 1991. ACM Press.

[Sha79] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[SNV05] P. Sousa, N. F. Neves, and P. Veríssimo. How Resilient are Distributed f Fault/Intrusion-Tolerant Systems? In *Proc. of the 2005 International Conference on Dependable Systems and Networks*, page to appear, June 2005.