Basic aspects in redundancy-based intrusion tolerance

Felicita Di Giandomenico[1] and Giulio Masetti[1,✉]

Institute of Science and Technology "A. Faedo",
Consiglio Nazionale delle Ricerche, 56124, Pisa, Italy[1]
✉giulio.masetti@isti.cnr.it

Abstract. This paper addresses intrusion tolerance in ICT systems (that is, how to guarantee the correct operation even when some of its parts are compromised), by adopting a redundancy approach. As an initial step to the process of incorporating redundancy features to enhance robustness of a system/subsystem to cyber attacks that end up with an intrusion, a conceptual framework encompassing the major aspects to be taken into account, regarding both system and redundancy characteristics, is proposed. This is the basis for reasoning on effective redundancy solutions, so a useful support for defining systems that need to satisfy resilience properties in hostile environments where threats include both intentional attacks and accidental faults.

Keywords: Intrusion tolerance, Redundancy, Diversity

1 Introduction and Related Work

The security of ICT components is increasingly challenged by a variety of attacks, and countermeasures are needed to cope with them. A primary defense against cyber attacks has been traditionally based on preventing intrusions, through methods such as authentication and access control, or on intrusion detection but without systematic forms of processing the intrusion symptoms. Despite the utmost contribution provided by preventive solutions, unfortunately reality shows that their effectiveness is limited and intrusions are permanently happening [14].

Therefore, intrusion tolerance emerged, with the aim to guarantee that a system works correctly even when some of its parts are compromised. Initial intrusion tolerance approaches are dated early ninetees, although they appear as isolated works, mainly focusing on protocols (see [21] for an overview of initial works on this subject). Many other papers addressed this topic in more recent years, some with emphasis on analysis and evaluation of security oriented strategies to manage system resources in order to reach desired reliability/availability levels (e.g., [9, 17, 23]), others on architectural solutions for specific application domains (e.g, [3, 22]) or on proposing variations/extensions to some classical fault tolerance architectures towards secure service (e.g., [16]), or investigating trade-offs between competing requirements (e.g., [8]). Intrusion tolerance in certain contexts can benefit from achievements obtained in other areas, such as multi-party computations and threshold cryptography; however, the downside is that new challenges have to be addressed [5].

The objective of this paper is to start a research activity aiming at a more general and deeper investigation of aspects and practices around redundancy-based intrusion tolerance, so to promote easier access and exploitation of this approach in a variety of system contexts. In particular, with the purpose to provide guidance in the selection and customization of appropriate intrusion tolerance approaches relying on the usage of redundancy, the offered contribution consists in the identification of the various aspects that need to be considered, regarding both system and redundancy characteristics. This is a preliminary, but essential step, to the correct design and deployment of the intrusion tolerance solution, guaranteeing its effectiveness. This work builds upon several, but fragmented solutions existing in the literature, and advances by developing a high level, general reference framework where the several dimensions, to be taken into account when resorting to redundancy-based intrusion tolerance techniques, are indicated and easily exploitable.

The rest of the paper is organized as follows. Section 2 is an overview of redundancy-based intrusion tolerance, and ends with the identification of the four major dimensions composing the proposed conceptual framework for intrusion tolerance, each one then addressed in the subsequent four sections. Specifically, Section 3 describes how cyber attacks are characterized in the proposed framework; Section 4 discusses the system components that are considered the subject of potential attacks; Section 5 characterizes the system under analysis; Section 6 focuses on the types of redundancy that are expected to be more relevant than others for intrusion tolerance. Section 7 draws final considerations and outlines the next steps.

2 Overview of redundancy-based intrusion tolerance

Prompt detection of attacked components is primarily pursued in security, to prevent the successful completion of an attack path, resulting in a high number of (potentially critical) compromised components. This is absolutely correct. However, detection as the only measure in place may be insufficient to reach desired levels of security, so forms of intrusion management are needed [21].

The dependability community has proposed since several decades fault tolerance techniques, with the aim of avoiding service failure in the presence of faults. As described in [2], fault tolerance includes techniques for *error detection* and *system recovery*, these last consisting of *error handling* and *fault handling* mechanisms. Although the kind of faults foreseen in [2] encompasses both accidental faults and deliberate attacks, so targeting dependability and security within a common framework, fault tolerance has been primarily thought as a means for dealing with accidental faults. However, as discussed in [21], the essential concepts are applicable to malicious faults, thus leading to intrusion tolerance. Indeed, the fault model in security resembles similar dynamics as for accidental faults: the presence of a *vulnerability*, which can be exploited with malicious intention by an attacker to launch an *attack* that, if successful, leads to an *intrusion*. The parallel with an example of accidental fault could be the following: a hardware component that is not properly shielded (pseudo-vulnerability) may be penetrated by particles from an electromagnetic field (pseudo-attacker), resulting in an operational fault (pseudo-intrusion) that generates errors in the performed computation. Of course, there are specificities of an intentional attack that need to be carefully analyzed, in order to take the most effective measure to tolerate it. In particular, the *logic* behind the attack requires more attentive reasoning than in the case of pure *accidental* events.

For terminology clarity, and in line with the AVI (Attack Vulnerability Intrusion) fault model in [21], we consider the fault chain *attack* $\rightarrow$ *vulnerability* $\rightarrow$ *intrusion* be the source of the generated error(s), and intrusion tolerance to be inserted as a barrier to prevent the generated error to result into a failure.

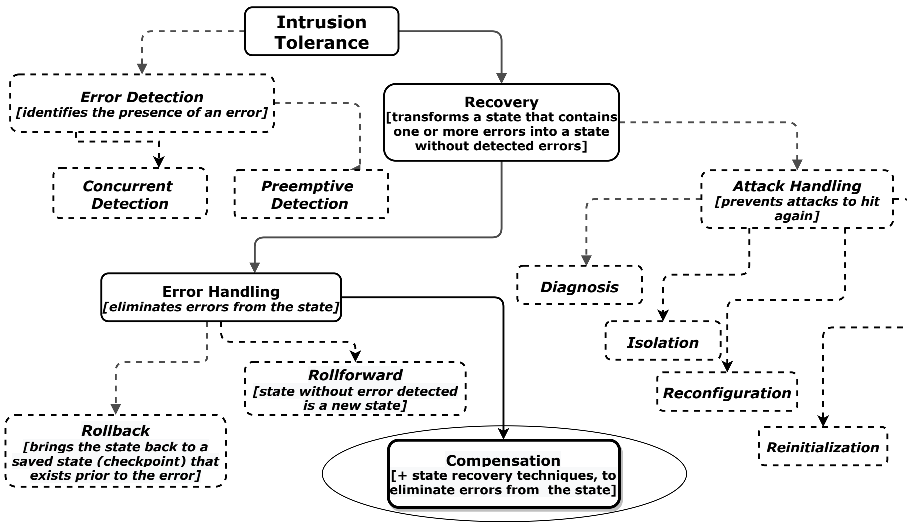
Intrusion Tolerance
Error Detection [identifies the presence of an error]
Recovery [transforms a state that contains one or more errors into a state without detected errors]
Concurrent Detection
Preemptive Detection
Attack Handling [prevents attacks to hit again]
Diagnosis
Error Handling [eliminates errors from the state]
Isolation
Reconfiguration
Rollforward [state without error detected is a new state]
Rollback [brings the state back to a saved state (checkpoint) that exists prior to the error]
Compensation [+ state recovery techniques, to eliminate errors from the state]
Reinitialization

Fig. 1. Intrusion Tolerance, focus on Compensation.

In Figure 1, the organization of fault tolerance mechanisms from [2], but declined in terms of attack tolerance, is proposed, with solid arrows identifying the path of interest in this paper. Specifically, among the wide spectrum of research directions under the hat of *Intrusion Tolerance* as depicted in Figure 1, we concentrate on techniques for handling the errors generated through the intrusion, and in particular on those solutions that adopt redundancy to mask the presence of error(s). They correspond to the category of solutions labelled *Compensation*. Redundancy means the exploitation of more components, to count on enough good results/values that mask, through a properly defined adjudication function that operates the selection among the multiple outcomes of the redundant structure, the presence of compromised computations/stored values/transmitted messages. Well known redundancy-based fault tolerant structures are *Recovery Block* [15] and *N-Version Programming* [1], and we are interested to borrow the concepts underlying them (and other schema as well) in the intrusion tolerance domain. Redundancy in time (i.e., retry of a computation or transmission), typically effective with transient accidental faults, appears inappropriate in the context of attacks (an attack remains solid, or increases along time).

Exploiting redundancy to mask the presence of errors makes error detection not immediately needed, although it remains a fundamental step of the chain, to identify and help confining the intrusion from propagating within the system. Indeed, masking the effect of intrusions can result as the only choice to satisfy other system requirements, for example when sufficiently accurate attack detection implies too large latency for the timing requirements of the application at hand. Moreover, when dealing with stateful computation (that is, the outcome of a computation depends on the input and the internal state of the computing unit), it is advisable to pair masking with state recovery techniques, to eliminate errors from the sate of the failed element(s) of the redundant structure.

Based on the long-dated experience with fault tolerance, and considering the peculiarities of an intentional attack, the proposed general conceptual framework for intrusion tolerance develops along four major dimensions. These last can be considered the *ingredients* to work with in the design of redundancy-based intrusion tolerance mechanisms when facing a specific application context:

– Attack model, that is how the cyber attack is characterized (in Section 3);
– Categories of system components targeted by attacks, that is the identification of which part(s) of the ICT system at hand can be the target of a cyber attack (in Section 4);
– System model and failure assumptions, namely whether the system at hand is a single system component, or a structured, distributed system (in Section 5);
– Type of redundancy, that is the characterization of the forms of redundancy that can be put in place (in Section 6).

3 Cyber attack model

A large variety of cyber attacks have been discovered and studied in the last decades. Several threat repositories exist, such as MITREs ATT&CK [20], CAPEC [18] and CWE [19], where a description on how an adversary can accomplish each individual attack, as well as the consequences of the attack itself, are generally provided. They are certainly helpful support to developers of countermeasures to cope with them, especially how to prevent their occurrence.

From the intrusion tolerance perspective, essentially what is relevant is to know the consequences of the attack, more than the dynamics on which vulnerabilities it exploits and the path to the successful intrusion. These last are vital information to accomplish the attack detection and treatment, but less relevant to carry out masking of the generated intrusion. Attack consequences are then connected with the security attribute that is primarily impacted, namely the well known triad *confidentiality*, *integrity* and *availability*. As a brief recall from [2]: i) confidentiality is preserved in absence of unauthorized disclosure of information; ii) integrity is preserved in absence of improper system alteration; and iii) availability is preserved with readiness for correct service.

Similar to [16], the manifestation of a successful attack on the compromised components can be summarized as follows:

- *Functionality Change*, that is the delivered results are incorrect. This means that the compromised components experience a failure in the value domain. The impact is mainly on the integrity property;
- *Performance Degradation*, that is the results are delivered late or, in the extreme case, they are omitted. This means that the compromised components experience a failure in the time domain. The impact is mainly on the availability property;
- *Information Leakage*, that is sensitive information are revealed. The impact is mainly on the confidentiality property.

Intrusion tolerance techniques by masking the presence of compromised components are mainly directed to preserve integrity and availability. Regarding confidentiality, such techniques are not effective; attack prevention is the reference approach for this property. Therefore, confidentiality is left out from the conceptual intrusion tolerance framework under development in this paper.

4 Categories of system components targeted by attacks

Cyber attacks can be launched to all the components of an ICT system, and typically an attack develops through several of them to be successful and lead to an intrusion. From the attacker perspective, the ICT components can be considered as belonging to three major categories:

- *Computing element*, that is a component that is devoted to perform some kind of functionality, to provide a service to the requesting entity (a user or another component). Operating systems primitives, software applications and enterprise software are typical examples of this category;
- *Communication element*, that is the means through which information is delivered to/from computing elements, users and storage. The internet and the several wireless networks technologies are typical examples of this category;
- *Data storage element*, which includes different storage technologies used to retain digital data within a computer system architecture. The term storage may refer both to a user's data generally and, more specifically, to the integrated hardware and software systems used to capture, manage and prioritize the data. This includes information in applications, databases, data warehouses, archiving, backup appliances and cloud storage.

5 System model and failure assumptions

Existing ICT systems (seen in isolation, or as part of a larger system, or as composed by a number of smaller systems) include a variety of components belonging to categories recalled in Section 4 and are organized according to different models, following methodologies and practices of the reference application domain. Without going in the details, such system models range from monolithic structure to distributed interacting components of different granularities, including SOA (Service Oriented Architecture) [7], microservice and SoS (Systems-of-Systems) paradigm [6]. There are pros and cons with each system model, but this discussion is out of the scope of this paper. The architectural solution is typically chosen on the basis of the functional and non-functional requirements, as well as cost implications. Since flexibility and scalability are among the most relevant requirements to drive the selection of the system architecture, the current trend is to increasingly evolve from the monolithic structure to forms of distributed computation. However, in addition to other considerations, it needs to be mentioned that there are long-lived systems, originally developed as a monolithic architecture, which cannot undergo significant redesign, but need to be enhanced from the resilience perspective. Therefore, the interest in monolithic-based solution appears to be still significant.

Another important aspect associated to the structure and operation of an ICT system is the assumed *failure model* for the system components (due to accidental faults and/or intentional attacks). In line with the discussion on effects of attacks in Section 3, experienced failures can be in the *value domain* (an incorrect value is delivered/transmitted/stored) or in the *time domain* (a value that violates the time constraints is delivered/transmitted).

6 Type of redundancy

Redundancy for fault tolerance purposes was initially exploited in the form of replicas, that is identical copies of the same system component, both hardware or software, to cope with accidental faults. The problem of common mode failures, mainly originated by design faults, raised attention on the dependencies among replicas such that a problem in one replica is actually present also in all the other replicas. Research on how to obtain diverse components, in order to promote sufficiently high probability of failure independence, was then triggered, leading to a shift of the paradigm of redundancy from replicas to variants, which are functionally equivalent components developed with some form of *diversity*.

In the security context, redundancy in the form of replicas has been recognized as not effective: the attacker can easily exploit the same attack to compromise all the replicas. Therefore, forms of diversity are needed.

As presented in [14], *diversity* in an ICT system can affect several aspects (the axes of diversity), ranging from software components (application level functionality, as well as supporting middleware features, libraries and lower level OS primitives) to hardware components, but also location, administrative and security methods. Indeed, diversity to hardware and software components has been practised since long time, through a variety of solutions. A thorough discussion of practices for enhancing diversity in software designs is in [13], including methods such as employment of diverse development teams, enforcing functional diversity and adopting different development environments (e.g., different programming languages, compilers, run-time supports, etc.). Data diversity, mainly consisting in using random perturbations of inputs or algorithm specific re-expression of inputs, is another form of diversity that can be applied alone, or in conjunction with other methods to develop diverse fault intrusion systems.

Administrative diversity plays a more important role when security is targeted rather than in case of accidental faults only. In fact, it is well known that many security compromises are carried out through social engineering [24]. Therefore, distributing the components of an intrusion tolerant system across different administrative domains, applying different security management policies, helps to mitigate the exploitation of social engineering by an intruder.

Concerning location diversity, consisting in placing several hw/sw components in different sites, it is another important defense against both physical and cyber threats (e.g., [3]).

Note that security methods are a category of system components that take great advantage of the diversity principle. As for application functions, using several security methods to enforce a given security attribute (e.g., encryption techniques), increases the chance that the attribute is not violated if a subset of these methods is compromised. Since security methods play a critical role, increasing their security is a priority when designing and deploying them.

A few considerations about diversity in relation with intrusion tolerance are in the following. It has been already recalled that degrees of diversity can be obtained in a variety of practices. However, when attacks are to be counteracted, practices such as just the adoption of different OS or programming languages may result too light. Functional diversity looks certainly more effective, although more expensive in general. Moreover, diversity at one layer leads to states of the underlying layers which are different, even if they are not diversified. This suggests that the higher is the level where diversity is applied and the greater is the expected benefit in terms of protection against an intruder. Moreover, since general categories of attacks can be identified and the efficacy of defenses varies between them, the decision on where and how to apply diversity methods for security should take into consideration such knowledge rather than simple enforcement of randomly picked techniques, to improve in efficacy.

Another aspect to consider cautiously is the granularity of the units that are selected as the entities to replicate for intrusion tolerance. Keeping the size of such units small has benefits in terms of error propagation and diagnosis, but reduces the possibility of diversity. In addition, small size means a higher number of units that are made redundant, thus increasing the attack surface.

Of course, diversity introduces also new problems. The major ones are:

- inexact voting: correctly working diverse variants will often produce different, but correct results. So, a generalized decision function is required, typically a distance function over the space of possible results, and rules to group results into consensus groups. Such definition may be application specific, making it more difficult to create a standard (sw or hw) voter component.

- Consistency among correct variants: decision points (e.g., expressions like if $x > a$ in a program) in the versions may be source of inconsistent actions taken by correct variants. Bypassing this problem by having versions decide on a consensus value of the variable before the decision point has the drawback of potential reduction in terms of diversity and performance degradation.

7 Final considerations and next steps

The system structure and the failure model, together with the resilience properties the system has to satisfy (e.g., a specified level of reliability and/or availability) drive the identification of which system components are primarily critical and deserve to be made more robust to undesirable events. When redundancy-based intrusion tolerance solutions are selected for such purpose, the same information is helpful to decide on the degree of redundancy (failure assumptions play a primary role), on the adjudication function for the redundant structure (able to cope with diverse redundancy), as well as on how to organize the redundancy from the operational perspective (sequential versus parallel execution, also depending on the presence of real-time constraints).

In this paper, we focused on redundancy-based intrusion tolerance with the objective of identifying the basic aspects around an ICT system that need to be considered in order to make an effective choice of the redundancy structure to adopt. The discussed conceptual framework, graphically depicted in Figure 2, encompasses four identified dimensions that are primarily relevant when building redundancy-based intrusion tolerance.

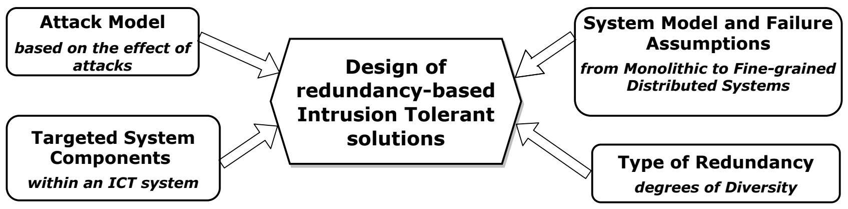
Attack Model
based on the effect of attacks
Targeted System Components
within an ICT system
Design of redundancy-based Intrusion Tolerant solutions
System Model and Failure Assumptions
from Monolithic to Fine-grained Distributed Systems
Type of Redundancy
degrees of Diversity

Fig. 2. Proposed conceptual framework for redundancy-based Intrusion Tolerance.

Moving from this initial, but important identification of major aspects that have to be considered in combination when addressing the employment of redundancy-based intrusion tolerance, the final goal of this research line is to provide guidelines to practitioners on which would be suitable solutions to adopt for the system at hand. To reach this goal, in the next steps we will:

i) revisit from the security perspective the classical redundancy-based fault tolerant approaches, namely the two extremes N-Version Programming [1] and Recovery-Block [15], together with the several variants in between (such as NSCP [12] and SCOP [4]). Literature review will be the starting point for this investigation, such as [16];

ii) discuss the relationship between the aspects overviewed in this paper and the characteristics of the intrusion tolerance solutions identified at the previous point. For example, in a microservice environment a somewhat natural way to deal with variants of a computational component is through N-version programming, with the adjudicator cautiously designed [1]. If the diversity degree of the variants constituting the redundant structure is such that the adjudication function would not be sufficiently accurate, while the application domain is more prone to define effective acceptance tests on the results provided by a variant, then sequential execution according to the Recovery Block paradigm would be preferable. However, assumptions on communication attacks are to be taken into account when deciding the redundancy structure: sequential execution of the available variants can be defeated by the communication attacks, while parallel execution is a more robust alternative. To continue with exemplifications, the assumption on the attack model leads to deciding on the degree of redundancy: if the impact of the attack is omission, lower redundancy is required than in the case of failure in the value domain. Relationships as those just presented will be carried on in a systematic way.

We believe that this research activity will be practically useful to developers of nowadays complex systems employed in critical domains. Among others, the BIECO project aims at developing a holistic framework, populated by a variety of methods and techniques to understand and manage the cybersecurity risks and threats in complex interconnected ICT systems. The BIECO's vision is to target a variety of domain sectors, thus calling for as much as possible general solutions, suitable to be customized to fit specific contexts. The proposed approach to redundancy-based intrusion tolerance constitutes an initial contribution in BIECO, as well as to other similar initiatives.

Acknowledgment This work was supported by the project BIECO (`www.bieco.org`) that received funding from the European Unionâs Horizon 2020 research and innovation programme under grant agreement No. 952702.

References

1. Avizienis, A.: The N-version approach to fault-tolerant software. IEEE Transactions on Software Engineering **11**(12), 1491–1501 (1985)
2. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing **1**(1), 11–33 (2004)
3. Babay, A., Tantillo, T., Aron, T., Platania, M., Amir, Y.: Network-attack-resilient intrusion-tolerant SCADA for the power grid. In: 48th Annual IEEE/IFIP Int. Conference on Dependable Systems and Networks (DSN). pp. 255–266 (2018)
4. Bondavalli, A., Di Giandomenico, F., Xu, J.: A cost-effective and flexible scheme for software fault tolerance. Journal of Computer Systems Science and Engineering **8**, 234–244 (1993)

[1] As a separate voter, possibly enclosed within the root of trust [16], or a voting logic distributed onto the variants [10], maybe introducing also camouflage units [11]

5. Brandão, L.T.A.N., Mouha, N., Vassilev, A.: Threshold schemes for cryptographic primitives. Tech. Rep. NISTIR8214, National Institute of Standards and Technology (2019)
6. Ceccarelli, A., Bondavalli, A., Froemel, B., Hoeftberger, O., Kopetz, H.: Basic Concepts on Systems of Systems, pp. 1–39 (2016)
7. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design (2005)
8. Gashi, I., Povyakalo, A., Strigini, L.: Diversity, safety and security in embedded systems: Modelling adversary effort and supply chain risks. In: 12th European Dependable Computing Conference (EDCC). pp. 13–24 (2016)
9. Haphuriwat, N., Bier, V.M.: Trade-offs between target hardening and overarching protection. European Journal of Operational Research **213**(1), 320–328 (2011)
10. Hardekopf, B., Kwiat, K., Upadhyaya, S.: Secure and fault-tolerant voting in distributed systems. In: IEEE Aerospace Conference Proceedings. pp. 1117–1126 (2001)
11. Kwiat, K., Taylor, A., Zwicker, W., Hill, D., Wetzonis, S., Ren, S.: Analysis of binary voting algorithms for use in fault-tolerant and secure computing. In: 5th International Conference on Computer Engineering Systems. pp. 269–273 (2010)
12. Laprie, J.C., Arlat, J., Beounes, C., Kanoun, K.: Definition and analysis of hardware- and software-fault-tolerant architectures. Computer **23**(7), 39–51 (1990)
13. Littlewood, B., Strigini, L.: A discussion of practices for enhancing diversity in software designs. Tech. Rep. DISPO, Centre for Software Reliability, City University (2000)
14. Obelheiro, R.R., Bessani, A.N., Lung, L.C., Correia, M.: How practical are intrusion-tolerant distributed systems? Tech. Rep. TR-06-15, Dept. of Informatics, Univ. of Lisbon (2006)
15. Randell, B.: System structure for software fault tolerance. IEEE Transactions on Software Engineering **1**(2), 220–232 (1975)
16. Rodriguez, M., Kwiat, K.A., Kamhoua, C.A.: Modeling fault tolerant architectures with design diversity for secure systems. In: IEEE Military Communications Conference (MILCOM). pp. 1254–1263 (2015)
17. Tarraf, D.C., Kamhoua, C.A., Kwiat, K.A., Njilla, L.: Majority is not always supreme: Less can be more when voting with compromised nodes. In: IEEE 18th International Symposium on High Assurance Systems Engineering (HASE). pp. 9–12 (2017)
18. The MITRE Corporation: Common Attack Pattern Enumeration and Classification (CAPEC). `https://capec.mitre.org/data/definitions/3000.html`
19. The MITRE Corporation: Common Weakness Enumeration (CWE). `https://cwe.mitre.org`
20. The MITRE Corporation: Mitre attack. `https://attack.mitre.org`
21. Veríssimo, P.E., Neves, N.F., Correia, M.P.: Intrusion-tolerant architectures: Concepts and design. In: Architecting Dependable Systems. pp. 3–36 (2003)
22. Vöelp, M., Verissimo, P.: Intrusion-tolerant autonomous driving. In: IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC). pp. 130–133 (2018)
23. Wang, L., Ren, S., Korel, B., Kwiat, K.A., Salerno, E.: Improving system reliability against rational attacks under given resources. IEEE Transactions on Systems, Man, and Cybernetics: Systems **44**(4), 446–456 (2014)
24. Winkler, I.S., Dealy, B.: Information security technology? ... Don't rely on it - a case study in social engineering. In: 5th USENIX UNIX Security Symposium (USENIX Security). pp. 1–6 (1995)