From Byzantine Fault Tolerance to Intrusion Tolerance
(a Position Paper)

Alysson Neves Bessani
LaSIGE, University of Lisbon, Faculty of Sciences – Lisbon, Portugal

***Abstract*—A system is said intrusion-tolerant if it maintains its security properties despite some of its components being compromised by a malicious adversary. Although the implementation of these systems usually requires the use of Byzantine fault-tolerant (BFT) protocols, they are not a complete solution. Besides BFT replication, there are several other techniques such as proactive recovery, diversity and confidential operation that are needed to implement these systems. Many of these techniques present interesting open problems that need to be addressed before a complete intrusion-tolerant system could be build and deployed.**

***Keywords*-Intrusion Tolerance, Byzantine Fault Tolerance**

I. Introduction

An *intrusion-tolerant* system is one that maintains its security properties (i.e., *confidentiality*, *integrity* and *availability*) despite some of its components being compromised by an adversary [1]. The term was coined by Fraga and Powell in 1985 and was almost forgotten for 15 years due to the prohibitive performance costs of the mechanisms required to implement these systems. Byzantine fault-tolerant (BFT) replication is perhaps the most notorious of such mechanisms. It is used to protect the integrity and availability of a system whose components are attacked and intruded by a malicious and intelligent adversary.

In the last decade, several papers showed that it is indeed possible to implement BFT replication with low overhead (e.g., [2], [3]), and thus this barrier was removed. Although initially intrusion tolerance was one of the key motivations for BFT replication, researchers realized that there are many additional problems in dealing with malicious adversaries and start positioning BFT as a solution to deal with non-malicious arbitrary faults like heisenbugs and data corruption on memory and disks. This position paper surveys some of the progress made on BFT and intrusion tolerance in the last years and point out some open problems that still need to be addressed in order to build practical systems that can operate despite attacks and intrusions perpetrated by a malicious adversary.

II. Intrusion-Tolerant Systems

A Byzantine fault-tolerant system is a system composed by n components that can operate correctly even if $f < n$ components suffer arbitrary faults.

The main difference between an intrusion tolerant and a BFT system is that the faults are explicitly assumed to be caused by a malicious adversary. This difference may appear to be of little impact, but it increases the complexity of any implementation since several new requirements need to be addressed:

- Many protocols are very efficient in favorable executions (executions with synchrony, no faults and no concurrency), however, the fact that a malicious adversary will be working against the protocol can make these favorable executions very unlikely. In consequence, the system needs to operate reasonably well under favorable and unfavorable execution.
- Common vulnerabilities are the bane of intrusion-tolerant systems. So, if one wants to resist attacks from a malicious adversary, extensive use of diversity is needed.
- When a machine is compromised, the system should be able to detect its misbehavior and react in order to clean the intrusion. This is a fundamental requirement that needs to be observed since, if an adversary was able to compromise f or less machines, the probability that he/she eventually find a way to compromise one more is high.
- In some systems, confidentiality is also a requirement and BFT replication cannot ensure it. For them, techniques such as threshold cryptography are needed to ensure that no replica stores the clear state, but only fragments of it that are of little use individually (i.e., only by at least $f + 1$ replicas cooperating one can recover information about the system state).

Given these extra requirements, and specially the fact that an intrusion-tolerant system cannot live with compromised machines forever, we propose a definition for this type of system:

Definition 1 *A replicated intrusion-tolerant system is a replicated system in which a malicious adversary needs to compromise more than f out-of n components in less than T time units in order to make it fail.*

This definition is similar to the guarantees provided by BFT systems with proactive recovery [2], and assumes the notion of periodic intrusion cleaning taking into account two parameters: f and T. Both these parameters are of little use if the previously mentioned requirements are not satisfied.

III. A Roadmap for Intrusion Tolerance

Here we present a list of solved, half-solved and open problems that should be solved for implementing intrusion tolerant systems based on distributed algorithms [4]. Our objective here is not to make this list (and the associated references) neither extensive nor complete, but give some pointers for the state of the art taking into account the space constraints.

A. Solved Problems

These are the problems we consider solved. However, it is worth to point out that there is always room for improvement.

High-performance BFT replication: During the last decade many papers proposing BFT protocols capable of offering high-performance in specific environment conditions were published (e.g., [2], [3]). These papers showed that a BFT replication protocol performance overhead can be made very modest in terms of both latency and throughput. The main conclusion about this is that the protocols performance should not be a concern anymore.

Resource-efficient BFT replication: Some works showed that it is possible to reduce the number of replicas of a BFT protocol from $3f+1$ to $2f+1$ or even $f+1$. There are two ways to do this. The first one is through the separation of the agreement from the execution phase of BFT replication [5]. This simple idea allows the use of $3f+1$ replicas to implement total order delivery of requests to $2f+1$ executing replicas. Recently this result was extended to use only $f+1$ executing replicas taking into account VM technology [6], [7]. The second way to mitigate the resource cost of replication is to employ trusted components to constraint the behavior of malicious replicas [8]–[10]. With these hybrid architectures it is possible to implement BFT replication using only $2f+1$ replicas. Some of these papers also require a trusted component as simple as an authenticated monotonic counter, which can be implemented using the trusted platform module [10].

Proactive recovery: Recovering replicas is a practical way to allow a replicated system to tolerate an unbounded number of faults during its lifetime. Proactive recovery [2], [11] aims to periodically and proactively rejuvenate replicas even if they are correct to clean potential undetected intrusions. The main advantage of these systems is that the adversary can stay in control of some replicas only during a window of vulnerability, even if its perpetrating a stealth intrusion and could not be detected. Proactive recovery together with BFT replication represent a interesting starting point for implementing an intrusion tolerant system that satisfies our definition.

Recent works showed that many proactive recovery systems can be attacked and delayed, giving time for an intelligent adversary to compromise more than f machines and takeover the system [12]. To deal with these vulnerabilities it is necessary to increment the replicated system with a synchronous subsystem capable of triggering timely recoveries without interference of attackers [13]. This paradigm was latter extended to deal with reactive recoveries and applied to a stateless intrusion-tolerant firewall, showing that it is practical for certain classes of critical services [14].

B. Half-solved Problems

These are challenges that were already addressed and reasonable solutions exists, but there are still gaps to be worked on.

Effectiveness of diversity: Common vulnerabilities and bugs that affect more than one replica of a replicated system can break the assumption that the system tolerates f faults since a single fault (a common vulnerability) can bring down more than one replica. To deal with this problem, the use of diverse replicas is a common assumption. There is a common understand that the use of N-version programming [15] is too costly for practical systems, however, opportunistic diversity of operating systems and database management systems were shown to be very effective to avoid common faults and bugs on these types of components [16], [17].

Despite these encouraging results, there are still areas that deserve more work. First, some commonly used components such as Java virtual machines and cryptographic APIs did not have their diversity evaluated. Second, most replication protocols are only evaluated on LANs, which obviously does not support the assumption of diversity regarding natural disasters or bad administration. It would be interesting to implement and evaluate a BFT protocol using diverse software on replicas deployed in different administrative domains.

Performance robustness of BFT protocols: Recent papers described several weakness of BFT protocols [18], [19] showing that under certain conditions their performance may be unacceptably bad. Those same papers, among others, propose ways to solve these problems and implement robust state machine replication. Alternatively, there are recent works pointing towards the avoidance of aggressive optimizations and target the stability of protocols, i.e., the theoretical performance (e.g., number of steps, message complexity) of the protocol in best and worst case is bounded and small [20].

C. Open problems

These are open problems that still need more work.

Intrusion reaction: For each protocol and service it is possible to define a set of intrusion detection rules and find measures to react to them. A question that could be asked is *given a specification of a protocol, how to detect misbehaviors automatically and react to them?* There are solutions for simple protocols [21], but it is still not clear how to extend these solutions for complex multi-tiered systems [22].

Time-bounded state transfer: Imposing time bounds on the system, like we did in our definition of intrusion tolerant system definition, require a hybrid distributed system model with a synchronous subsystem [13]. With such subsystem it is possible to bound the time the adversary have to compromise more than f replicas if there is an upper bound on the time needed to recover a replica. The problem is that one of the steps of the recovery is state validation and/or transfer (if current copy is not valid), which is usually done on the asynchronous part of the system, through the BFT protocol. Consequently, this part of the recovery can be attacked and delayed by the adversary, and thus such bound on state transfer time does not exist in current architectures. To avoid losing state, some systems stop the service and call for human intervention if state transfer could not be completed in a certain time (e.g., [2]).

A possible solution would be to transfer state through the synchronous network used to coordinate the recoveries. Unfortunately it will not solve the problem since the interaction between the synchronous and asynchronous part of the system can take an arbitrary amount of time, and the problem could persist. This problem, while subtle, makes it impossible to design a robust system that matches our definition of intrusion tolerance.

Diversity management: Assuming we have a pool of diverse configurations for our replicas, with different operating systems, database management systems, virtual machines, etc., how to choose the best configuration for a set of replicas? This question could only be answered assuming that we have access to studies that show how diverse the different software components we have [16], [17]. However, if proactive recovery is employed, not only intrusions but latent vulnerabilities need to be clean from the replicas after a recovery. It means that the pool of diverse configurations, will be exhausted after some time, unless one keep upgrading it with patches and explore the potential differences between different versions of the same system. Besides that, a non-homogeneous executing environment can have bad effects on the performance of replication protocols [23], which make difficult for system managers to understand if the different behavior of replicas over time are due to the diverse platforms, workloads or attacks and intrusions[1]. All these concerns complicate the use of diversity on intrusion-tolerant systems, and clearly show that more work is needed here.

Confidential operation: The techniques discussed so far are useful to protect the integrity and availability of an intrusion tolerant system, but not its confidentiality. A still open question is how to ensure the confidentiality of the state of intrusion tolerant system even if up to f machines are compromised. There are specific solutions for storage-like services [24], [25], but no general solution for state machine replication. Recent breakthroughs on homomorphic cryptography (e.g., [26]) can open interesting avenues for work on this area.

Gracefull degradation: The intrusion tolerance definition we use is very strict in the sense it does not accomodate temporary downtime or degraded operation. An interesting research topic that demands further research is the definition of weaker notions of intrusion tolerance considering that sometimes the adversary may be powerfull enough to break the f and T bounds defined for the system.

[1]This can also have interesting effects on protocol parameters such as timeout values.

IV. Final Remarks

In this paper we defined an intrusion-tolerant system using a fault threshold f and a time bound T for the adversary to compromise more than f system replicas. Based on this definition we surveyed solved problems, half-solved problems and open problems regarding the implementation of this kind of system. We hope this paper clarifies the difference between BFT and intrusion tolerance and show that there are still work to do on half-solved and open problems in order to fulfill the vision of an intrusion-tolerant system.

Acknowledgments

Many of the ideas here presented were developed during years of discussion with many members of the Navigators research group, from the University of Lisbon. More specifically, I would like to thank Miguel Correia, Paulo Sousa, Nuno Neves and Paulo Veríssimo. This work was partially supported by the EC through project FP7-257475 (MASSIF) and by the FCT through the Multiannual and the CMU-Portugal Programmes, and the project PTDC/EIA-EIA/100894/2008 (DIVERSE).

References

[1] J. S. Fraga and D. Powell, "A fault- and intrusion-tolerant file system," in *Proceedings of the 3rd International Conference on Computer Security*, 1985, pp. 203–218.

[2] M. Castro and B. Liskov, "Practical Byzantine fault-tolerance and proactive recovery," *ACM TOCS*, vol. 20, no. 4, pp. 398–461, 2002.

[3] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative Byzantine fault tolerance," in *Proc. of the 21st ACM Symp. on Operating Systems Principles (SOSP'07)*, Oct. 2007.

[4] Q. Nguyen and A. Sood, "A comparison of intrusion-tolerant system architectures," *IEEE Security & Privacy*, vol. 9, no. 2, pp. 18–25, Mar./Apr. 2003.

[5] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin, "Separating agreement form execution for Byzantine fault tolerant services," in *Proc. of the 19th ACM Symp. on Operating Systems Principles - SOSP'03*, Oct. 2003.

[6] T. Wood, R. Singh, A. Venkataramani, P. Shenoy, and E. Cecchet, "ZZ and the art of practical BFT execution," in *Proceedings of the 6th ACM SIGOPS/EuroSys European Systems Conference - EuroSys'11*, Apr. 2011.

[7] T. Distler and R. Kapitza, "Increasing performance in Byzantine fault-tolerant systems with on-demand replica consistency," in *Proceedings of the 6th ACM SIGOPS/EuroSys European Systems Conference - EuroSys'11*, Apr. 2011.

[8] M. Correia, N. F. Neves, and P. Veríssimo, "How to tolerate half less one Byzantine nodes in practical distributed systems," in *Proceedings of the 23rd IEEE Symposium on Reliable Distributed Systems*, Oct. 2004, pp. 174–183.

[9] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz, "Attested append-only memory: Making adversaries stick to their word," in *Proc. of the 21st ACM Symp. on Operating Systems Principles (SOSP'07)*, Oct. 2007.

[10] G. S. Veronese, M. Correia, A. Bessani, L. Chung, and P. Verissimo, "Minimal Byzantine fault tolerance: Algorithm and evaluation," Department of Computer Science, University of Lisbon, DI/FCUL TR 09–15, Jun. 2009. [Online]. Available: http://hdl.handle.net/10455/3153

[11] L. Zhou, F. Schneider, and R. Van Rennesse, "COCA: A secure distributed online certification authority," *ACM TOCS*, vol. 20, no. 4, pp. 329–368, Nov. 2002.

[12] P. Sousa, N. F. Neves, and P. Verissimo, "Hidden problems of asynchronous proactive recovery," in *Proc. of the Workshop on Hot Topics in System Dependability – HotDep'07*, June 2007.

[13] P. Sousa, N. F. Neves, and P. Veríssimo, "How resilient are distributed f fault/intrusion-tolerant systems?" in *Proceedings of Dependable Systems and Networks – DSN 05*, Jun. 2005, pp. 98–107.

[14] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo, "Highly available intrusion-tolerant services with proactive-reactive recovery," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 4, pp. 452–465, 2010.

[15] A. Avizienis and L. Chen, "On the implementation of N-version programming for software fault tolerance during execution," in *Proceedings of the IEEE COMPSAC*, Chicago, IL, USA, Nov. 1977, pp. 149–155.

[16] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "OS diversity for intrusion tolerance: Myth or reality?" in *Proc. of the Int. Conf. on Dependable Systems and Networks (DSN'11)*, Hong Kong, 2011.

[17] I. Gashi, P. Popov, and L. Strigini, "Fault tolerance via diversity for off-the-shelf products: A study with SQL database servers," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 280–294, Oct-Dec 2007.

[18] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making Byzantine fault tolerant systems tolerate Byzantine faults," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design & Implementation*, Apr. 2009.

[19] Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Byzantine replication under attack," in *Proceedings of the IEEE International Conference on Dependable Systems and Networks*, Jun. 2008.

[20] A. Bessani, M. Correia, and P. Sousa, "Active quorum systems," in *Proc. of the Workshop on Hot Topics in System Dependability – HotDep'10*, Oct. 2010.

[21] A. Haeberlen, P. Kuznetsov, and P. Druschel, "Peerreview: Practical accountability for distributed systems," in *Proc. of the 21st ACM Symp. on Operating Systems Principles (SOSP'07)*, Oct. 2007.

[22] P. Pal, F. Webber, and R. Schantz, "The DPASA survivable JBI - a high-water mark in intrusion-tolerant systems," in *1st Workshop on Recent Advances on Intrusion-Tolerant Systems – WRAITS 2007*, Apr. 2007.

[23] M. Castro, R. Rodrigues, and B. Liskov, "BASE: Using abstraction to improve fault tolerance," *ACM Transactions Computer Systems*, vol. 21, no. 3, pp. 236–269, Aug. 2003.

[24] A. N. Bessani, E. P. Alchieri, M. Correia, and J. S. Fraga, "DepSpace: a Byzantine fault-tolerant coordination service," in *Proceedings of the 3rd ACM SIGOPS/EuroSys European Systems Conference – EuroSys'08*, Apr. 2008.

[25] M. A. Marsh and F. B. Schneider, "CODEX: A robust and secure secret distribution system," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 34–47, Jan.-Mar. 2004.

[26] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, pp. 97–105, March 2010.