Software Diversity for Future Systems Security

Abdelouahed Gherbi[1], Robert Charpentier[2] and Mario Couture[2]
[1]École de technologie supérieure
Abdelouahed.Gherbi@etsmtl.ca
[2]Defence Research and Development Canada
{Robert.Charpentier, Mario.Couture}@drdc-rddc.gc.ca

Abstract

Software security represents a major concern as cyber-attacks continue to grow in number and sophistication. One security-weakening factor is related to the standardized software ecosystem that facilitates the spread of malware in systems that share common vulnerabilities. In this overview article, the main concepts associated with diversity and software redundancy are described in the perspective of improving attack resistance. The remarkable progress made in this area, where commercial implementations are now emerging, is also highlighted.

1. Introduction

The security of information systems remains an extremely critical issue despite the good progress that was made in the last 10 years in the fields of software quality and system reliability. It seems that defensive measures cannot catch up to the continuous growth of cyber threats that are not only increasing in number but also in sophistication and scale [1, 2].

It remains extremely difficult to produce fault-free software despite the rigorous quality controls that are generally part of the software development process. These *residual faults* constitute dormant vulnerabilities which would eventually end up being discovered by malicious attackers and exploited to carry out cyber attacks. Moreover, in order to ease the system management, reduce the configuration errors and achieve portability, most of the systems used nowadays run substantially similar software. This is called information technology monoculture [3] [29]. As a consequence, these systems share similar vulnerabilities that facilitate malware propagation and enable large-scale exploitation of these common vulnerabilities.

The Canadian Forces (CF), like most armed forces around the world, are very much concerned by the "conjugation of these risks factors": the increased threat capabilities aiming at vulnerable infrastructures combined with society's dependency on information sharing. Recognizing that cyber attacks are inevitable in the future, a shift from the traditional defensive strategies toward more proactive measures can now be observed. This includes: (a) more rigorous monitoring for earlier attack detection; (b) the capture of legal evidence (cyber forensics) to enable post-event investigation; (c) some semi-automated responses to the most likely attacks; and (d) pre-programmed recovery strategies to minimize the impact of successful attacks.

Among the technologies that have the potential of mitigating the cyber attack risks, "software redundancy" that includes "component diversity" appears to be one of the rare technologies promising an order-of-magnitude increase in system security. The basic idea is simply to have

critical systems implemented in two (or more) instances using sufficiently different sub-systems (e.g. Linux and Unix BSD) so the same dormant vulnerability does not exist in both redundant systems—making it impossible for the attackers to exploit the same vulnerability in both instances simultaneously. Not only does such architecture offer attack resistance, it also greatly improves the monitoring of transactions and the early detection of abnormal behavior by the comparison of both executions. It also enables continuity of services since the replica can handle the user's requests while the first system is targeted, investigated or recovering from a recent attack.

In 2008, Defence R&D Canada initiated a study to evaluate the state-of-the-art in software redundancy implementing technological diversity to mitigate the risk associated with the IT monoculture. The amount of high-quality work that is going on in the scientific community is impressive. This short article gives an overview of the state-of-the-art in system redundancy using different types of diversity paradigms.

2. Redundancy and Diversity combined in a Defense Mechanism

Redundancy is a traditional means to achieve fault tolerance and higher system reliability. This has proven to be valid mainly for hardware because of the *failure independence* assumption as hardware failures are typically due to random faults. Therefore, the replication of components provides added assurance. When it comes to software, however, failures are due to design and/or implementation faults. As a result, such faults are embedded within the software and their manifestation is systematic. Therefore, redundancy alone is not effective against software faults.

Faults embedded in software represent potential *vulnerabilities*, which can be exploited by *external interactive malicious fault* (i.e. attacks) [30]. These attacks can ultimately enable the *violation of the system security property* (i.e. security failure) [30]. Therefore the diversity principle can potentially be used for security purposes. First, diversity can be used to decrease the common vulnerabilities. This is achieved by building a software system out of a set of *diverse but functionally equivalent components*. This in turns makes it very difficult for a malicious opponent to be able to break into a system with the very same attack. Second, the ability to build a system out of redundant and diverse components provides an opportunity to monitor the system by comparing the dynamic behavior of the diverse components when presented with the same input. This enables to endow the system with efficient intrusion detection capability.

Therefore, diversity has naturally caught the attention of the software security research community. The seminal work presented by Forrest *et al.* in [4] promotes the general philosophy of system security using diversity. The authors argue that uniformity represents a potential weakness because any flaw or vulnerability in an application is replicated on many machines. The security and the robustness of a system can be enhanced through the deliberate introduction of diversity. Deswarte *et al.* review in [5] the different levels of diversity of software and hardware systems and distinguish different dimensions and different degrees of diversity [6]. Bain *et al.* [7] presented a study to understand the effects of diversity on the survivability of systems faced with a set of widespread computer attacks including the Morris worm, Melissa virus, and LoveLetter worm. Ammann *et al.* [8] report on a discussion held by a panel of

renowned researchers about the use of diversity as a strategy for computer security and the main open issues requiring further research. It emerges from this discussion that there is a lack of *quantitative* information on the cost associated with diversity-based solutions and a lack of knowledge about the extent of protection provided by diversity.

Three main levels of security enhancements based on diversity and redundancy can be distinguished: first at the architecture level, where replicas of critical sub-systems are introduced to maintain service delivery even when one sub-system fails; second at the code level, where some program transformations are made to diversify replica; and finally a fully monitored combination of diversified components cleverly assembled in a secure architecture. In the sequel, these approaches are discussed further.

3. Redundancy Obtained by Multiple Instances Running in Parallel

Two categories of software architectures implementing redundancy can be distinguished. The first category uses a proxy to coordinate multiple COTS (Commercial-Off-The-Shelf) applications while the second one uses a middleware to achieve the same purpose. Noticeably, some commercial products implementing such strategies can now be found on the market like "*everRun*" for Windows by Marathon Technologies [www.marathontechnologies.com].

3.1 Multiple COTS applications coordinated by a proxy

The software architectures described in this section implement the architectural pattern depicted in Figure 1. This approach is ideal for a system integration of COTS components or legacy and closed applications aiming to deliver the services. The servers are shielded from the user side through proxies. Monitoring and voting mechanisms are used to check the health of the system, validate the results, and detect abnormal behavior. Examples of this approach include the Dependable Intrusion Tolerance (DIT) architecture [9, 10], the Scalable Intrusion Tolerant Architecture (SITAR) [11], and Hierarchical Adaptive Control for QoS Intrusion Tolerance (HACQIT) [12].

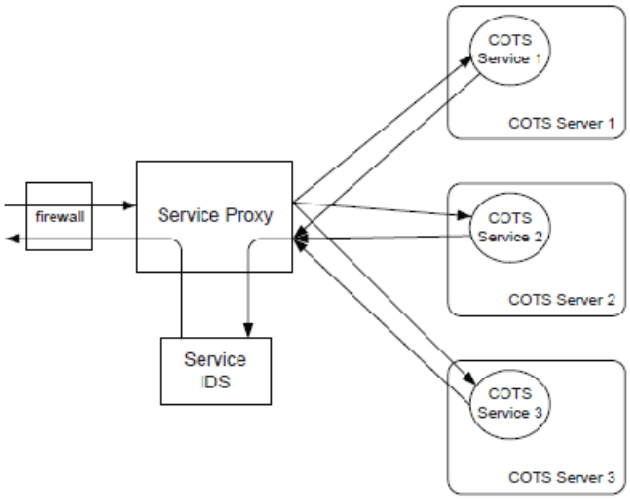

Figure 1: General Architectural Pattern of Intrusion Tolerance

3.2 Multiple applications assembled through middleware

Middleware-based approaches are much richer since they can provide server coordination between multiple "diverse" applications while hiding the sub-system differences [13]. Several intrusion tolerant software architectures are part of this category.

The Intrusion Tolerance by Unpredictable Adaptation (ITUA) architecture is a distributed object framework which integrates several mechanisms to enable the defense of critical applications [14]. The objective of this architecture is to enable the tolerance of sophisticated attacks aiming at corrupting a system.

Malicious and Accidental Fault Tolerance for Internet Applications (MAFTIA) [15] is a European research project which targeted the objective of systematically investigating the *tolerance paradigm* in order to build large scale dependable distributed applications.

The Designing Protection and Adaptation into a Survivability Architecture (DPASA) [16, 17] is a survivability architecture providing a diverse set of defense mechanisms. In this architecture diversity is used to achieve a defense in depth and a multi-layer security approach [17]. This architecture relies on a robust network infrastructure which supports redundancy and provides security services such as packet filtering, source authentication, link-level encryption, and network anomaly sensors. The detection of violations "*triggers"* defensive responses provided by middleware components in the architecture.

Fault/instrusiOn REmoVal through Evolution and Recovery (FOREVER) [18] is a service which is used to enhance the resilience of intrusion-tolerant replicated systems. FOREVER achieves this goal through the combination of recovery and evolution. FOREVER allows a system to recover from malicious attacks or faults using time-triggered or event-triggered periodic recoveries.

4. Diversity obtained by Program Transformations

Diversity can be introduced in the software ecosystem by applying automatic program transformations, which preserve the functional behavior and the programming language semantics. They consist essentially in randomization of the code, the address space layout or both in order to provide a probabilistic defense against unknown threats. Three main techniques can be used to randomize software.

The Instruction Set Randomization (ISR) technique [19, 20] changes the instruction set of the processor so that unauthorized code will not run successfully. The main idea underlying ISR is to decrease the attacker's knowledge about the language used by the runtime environment on which the target application runs. ISR techniques aim at defending against code injection attacks, which consist in introducing executable code within the address space of a target process, and then passing the control to the injected code. Code injection attacks can succeed when the injected code is compatible with the execution environment.

Address Space Randomization (ASR) [21] is used to increase software resistance to memory corruption attacks. These are designed to exploit memory manipulation vulnerabilities such as

stack and heap overflows and underflows, format string vulnerabilities, array index overflows, and uninitialized variables. ASR consists basically in randomizing the different regions of the process address space such as the stack and the heap. Noticeably, ASR has been integrated into the default configuration of the Windows Vista operating system [22].

Data Space Randomization (DSR) is a different randomization-based approach which aims also at defending against memory error exploits [23]. In particular, DSR randomizes the representation of data objects. This is often implemented by applying a modification to the data representation, such as using an XOR operation for each data object in memory against randomly chosen mask values. The data are unmasked right before being used. This makes the results of using the corrupted data highly unpredictable. The DSR technique seems to have advantages over ASR, as it provides a broader range of randomization: on 32-bit architectures, integers and pointers are randomized over a range of 2^{32} values. In addition, DSR is able to randomize the relative distance between two data objects, addressing a weakness of the ASR technique.

5. Higher Resistance obtained by Combining Redundancy and Diversity

The ability to build a system combining redundant and diverse components provides new powerful capabilities. One of them is the advanced monitoring of the redundant system by comparing the behavior of the diverse replicas. This endows the system with efficient intrusion detection capabilities not achievable with standard intrusion detection techniques based on signatures or malware modeling. Moreover, with the introduction of some assessment of the behavioral advantages of one implementation over the others, a "meta-controller" can ultimately adapt the system behavior or its structure over time. These futurist concepts are now prototyped in several R&D projects like those briefly described below.

5.1 Intrusion Detection using Output Voting

Several experimental systems used output voting for the sake of detecting some types of server compromission. For example, the HACQIT system [9] uses the status codes of the server replica responses. If the status codes are different the system detects a failure. Totel *et al.* [24] extend this work to do a more detailed comparison of the replica responses. They realized that web server responses may be slightly different even when there is no attack, and proposed a detection algorithm to detect intrusions with a higher accuracy (lower false alarm rate). These research initiatives specifically target web servers and analyze only server responses. Consequently, they cannot consistently detect compromised replicas.

5.2 Behavior monitoring in N-Variant Systems

N-variant systems provide a framework which allows executing a set of automatically diversified variants using the same inputs [25]. The framework monitors the behavior of the variants in order to detect divergences. The variants are built so that an anticipated type of exploit can succeed on only one variant. Therefore, such exploits become detectable. Building the variants requires a special compiler or a binary rewriter. Moreover, this framework detects only anticipated types of exploits, against which the replicas are diversified.

5.3 Multi-variant Execution Environment

Multi-variant code execution is a runtime monitoring technique which prevents malicious code execution [26]. This technique uses diversity to protect against malicious code injection attacks. This is achieved by running several slightly different variants of the same program in lockstep. The behavior of the variants is compared at synchronization points, which are in general system calls. Any divergence in behavior is suggestive of an anomaly and raises an alarm.

5.4 Behavioral Distance

The behavioral distance approach aims at detecting sophisticated attacks which manage to emulate the original system behavior including returning the correct service response (also known as mimicry attacks). These attacks are thus able to defeat traditional anomaly-based intrusion detection systems (IDS). Behavioral Distance achieves this defense using a comparison between the behaviors of two diverse processes running the same input. It measures the extent to which the two processes behave differently. Gao *et al.* proposed two approaches to compute such measures [27, 28].

6. Concluding Remarks

A few modern Operating Systems integrate some level of diversity to improve internal security and a few COTS packages are emerging that implement redundancy extension into traditional architectures. It seems that system architects should now consider more systematically redundancy or component diversity for critical systems that are operated in hostile environments. In many instances, the cost of security failures may well justify the additional complexity and the associated deployment and operating costs. The exploitation of both features simultaneously remains mostly experimental at this time but the very strong promises that such architectures make will continue to justify R&D in this field.

References

[1] Symantec (2009), *Global Internet Security Threat Report – Trends for 2008*, (TR XIV), Symantec.

[2] Emerging risks team, Lloyds (2009), *Digital Risks: Views of a Changing Risk Landscape*, (TR XIV) Lloyd's.

[3] Lala, Jaynarayan H.; and Schneider, Fred B. (2009), *IT Monoculture Security Risks and Defenses*, IEEE Security & Privacy, 7(1), pp. 12–13.

[4] Forrest, Stephanie; Somayaji, Anil; and Ackley, David H. (1997), *Building Diverse Computer Systems*, Workshop on Hot Topics in Operating Systems, pp. 67–72.

[5] Deswarte, Yves; Kanoun, Karama; and Laprie, Jean-Claude (1998), *Diversity against accidental and deliberate faults*, In *Computer Security, Dependability, and Assurance: From Needs to Solutions*, IEEE Computer Society Press, pp. 171–181.

[6] Obelheiro, Rafael R.; Bessani, Alysson N.; Lung, Lau C.; and Correia, Miguel (2006), *How Practical are Intrusion-Tolerant Distributed Systems?* (TR0615) Universidade de Lisboa.

[7] Bain, Charles; Faatz, Donald B.; Fayad, Amgad; and Williams, Douglas E. (2001), *Diversity as a defense strategy in information systems*. In IFIP Proceedings, Vol. 211, Kluwer, pp. 77–94.

[8] Ammann, Paul; Barnes, Bruce H.; Jajodia, Sushil; and Sibley, Edgar H. (Eds.) (1999), *Computer Security, Dependability, and Assurance: From Needs to Solutions: Proceedings 7–9 July 1998, York, England, 11–13 November 1998, Williamsburg, Virginia*, IEEE Computer Society Press.
[9] Deswarte, Yves; and Powell, David (2004), *Intrusion tolerance for Internet applications*, In *Proceedings of the IFIP 18th World Computer Congress, August 22–27, 2004, Toulouse, France*, Kluwer, pp. 241–256.
[10] Valdes, Alfonso; Almgren, Magnus; Cheung, Steven; Deswarte, Yves; Dutertre, Bruno; Levy, Joshua; Saïdi, Hassen; Stavridou, Victoria; and Uribe, Tomas E. (2003), *Dependable Intrusion Tolerance: Technology Demo*, In Proceedings of the 3rd DARPA Information Survivability Conference and Exposition, April 22–24, 2003, Washington, DC (DISCEX-III 2003), IEEE, pp. 128–130.
[11] Wang, Feiyi; Jou, Frank; Gong, Fengmin; Sargor, Chandramouli; Goseva-Popstojanova, Katerina; and Trivedi, Kishor (2003), *SITAR: A Scalable Intrusion-Tolerant Architecture for Distributed Services*, In Proceedings of the Foundations of Intrusion Tolerant Systems (OASIS'03), IEEE, pp. 359–367.
[12] Reynolds, James C.; Just, James E.; Lawson, Ed; Clough, Larry A.; Maglich, Ryan; and Levitt, Karl N. (2002), *The Design and Implementation of an Intrusion Tolerant System*, In Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN 2002), IEEE, pp. 285–292.
[13] Sames, David; Matt, Brian; Niebuhr, Brian; Tally, Gregg; Whitmore, Brent; and Bakken, David E. (2002), *Developing a Heterogeneous Intrusion Tolerant CORBA System*, In Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN 2002), IEEE, pp. 239–248.
[14] Pal, Partha Pratim; Rubel, Paul *et al.* (2006), *An architecture for adaptive intrusion-tolerant applications*, Software: Practice and Experience, 36(11–12), pp. 1331–1354.
[15] Verissimo, Paulo; Neves, Nuno Ferreira; Cachin, Christian; Poritz, Jonathan A.; Powell, David; Deswarte, Yves; Stroud, Robert J.; and Welch, Ian (2006), *Intrusion-tolerant middleware: the road to automatic security*, IEEE Security & Privacy, 4(4), pp. 54–62.
[16] Atighetchi, Michael; Rubel, Paul; Pal, Partha Pratim; Chong, Jennifer; and Sudin, Lyle (2005), *Networking Aspects in the DPASA Survivability Architecture: An Experience Report*, 4th IEEE International Symposium on Network Computing and Applications, IEEE, pp. 219–222.
[17] Chong, Jennifer; Pal, Partha Pratim; Atighetchi, Michael; Rubel, Paul; and Webber, Franklin (2005), *Survivability Architecture of a Mission Critical System: The DPASA Example*, In Proceedings of the 21st Annual Computer Security Applications Conference, IEEE, pp. 495–504.
[18] Bessani, Alysson Neves; Reiser, Hans P.; Sousa, Paulo; Gashi, Ilir; Stankovic, Vladimir; Distler, Tobias; Kapitza, Rüdiger; Daidone, Alessandro; and Obelheiro, Rafael R. (2008), *FOREVER: Fault/intrusiOn REmoVal through Evolution & Recovery*, ACM/IFIP/USENIX 9th International Middleware Conference, Association for Computing Machinery (ACM), pp. 99–101.
[19] Kc, Gaurav S.; Keromytis, Angelos D.; and Prevelakis, Vassilis (2003), *Countering code-injection attacks with instruction-set randomization*, Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM, pp. 272–280.
[20] Barrantes, Elena Gabriela; Ackley, David H.; Palmer, Trek S.; Stefanovic, Darko; and Zovi, Dino Dai (2003), *Randomized instruction set emulation to disrupt binary code injection attacks*,

Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM, pp. 281–289.
[21] Shacham, Hovav; Page, Matthew; Pfaff, Ben; Goh, Eu-Jin; Modadugu, Nagendra; and Boneh, Dan (2004), *On the effectiveness of address-space randomization*, In Proceedings of the 11th ACM Conference on Computer and Communications Security, ACM, pp. 298–307.
[22] Whitehouse, Ollie (2007), *An Analysis of Address Space Layout Randomization on Windows Vista*, Technical Report Symantec.
[23] Bhatkar, Sandeep and Sekar, R. (2008), *Data Space Randomization*, Vol. 5137 of Lecture Notes in Computer Science (LNCS): 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July 10–11, 2008, Paris, France (DIMVA 2008), Springer, pp. 1–22.
[24] Totel, Eric; Majorczyk, Frédéric; and Mé, Ludovic (2006), *COTS Diversity Based Intrusion Detection and Application to Web Servers*, In LNCS Vol. 3858: 8th International Symposium on Recent Advances in Intrusion Detection, September 7–9, 2005, Seattle, Washington (RAID 2005), Springer, pp. 43–62.
[25] Cox, Benjamin; Evans, David; Filipi, Adrian; Rowanhill, Jonathan; Hu, Wei; Davidson, Jack; Knight, John; Nguyen-Tuong, Anh; and Hiser, Jason (2006), *N-variant systems: a secretless framework for security through diversity*, In Proceedings of the 15th conference on USENIX Security Symposium, USENIX Association.
[26] Weatherwax, Eric; Knight, John; and Nguyen-Tuong, Anh (2009), *A Model of Secretless Security in N-Variant Systems*, In Proceedings of the 39th Annual IFIP Conference on Dependable Systems and Network (DSN 2009).
[27] Gao, Debin; Reiter, Michael K.; and Song, Dawn Xiaodong (2006), *Behavioral Distance for Intrusion Detection*, LNCS Vol. 3858: 8th International Symposium on Recent Advances in Intrusion Detection, September 7–9, 2005, Seattle, Washington (RAID 2005), Springer, pp. 63–81.
[28] Gao, Debin; Reiter, Michael K.; and Song, Dawn Xiaodong (2006), *Behavioral Distance Measurement Using Hidden Markov Models*, RAID'06, LNCS Vol. 4219: 9th International Symposium on Recent Advances in Intrusion Detection, September 20–22, 2006, Hamburg, Germany (RAID 2006), Springer, pp. 19–40.
[29] Birman, K.P.; Schneider, F.B. (2009), "The Monoculture Risk Put into Context," IEEE Security & Privacy, vol.7, no.1, pp.14-17
[30] Avizienis, Algirdas, Laprie, Jean-Claude, Randell, Brian, and Landwehr, Carl E. (2004), Basic Concepts and Taxonomy of Dependable and Secure Computing, IEEE Trans. Dependable Sec. Comput., 1(1), 11–33.