Diversity-based Approaches to Software Systems Security

Abdelouahed Gherbi[1] and Robert Charpentier[2]

[1] Department of Software and IT Engineering
École de technologies supérieure, ÉTS
Montréal, Canada
[2] Defence Research and Development Canada - Valcartier
Québec, Canada

Abstract. Software systems security represents a major concern as cyber-attacks continue to grow in number and sophistication. In addition to the increasing complexity and interconnection of modern information systems, these systems run significant similar software. This is known as IT monoculture. As a consequence, software systems share common vulnerabilities, which enable the spread of malware. The principle of diversity can help in mitigating the negative effects of IT monoculture on security. One important category of the diversity-based software approaches for security purposes focuses on enabling efficient and effective dynamic monitoring of software system behavior in operation. In this paper, we present briefly these approaches and we propose a new approach which aims at generating dynamically a diverse set of lightweight traces. We initiate the discussion of some research issues which will be the focus of our future research work.

Keywords: Security, IT monoculture, Diversity, Dynamic monitoring

1 Introduction

Security remains an extremely critical issue. This is evidenced by the continuous growth of cyber threats [22]. Cyber-attacks are not only increasing in number but also in sophistication and scale. Some attacks are now of nation/state class [24]. This observation can be explained by a combination of a multitude of contributing factors, which include the followings.
The increasing complexity of software systems makes it difficult to produce fault free software even though different quality controls are often part of the software development process. These residual faults constitute dormant vulnerabilities, which would eventually end up being discovered by determined malicious opponents and exploited to carry out cyber-attacks. Moreover, software systems are distributed and interconnected through open networks in order to communicate controls and data. This in turns increases tremendously the risk of attacks. Most importantly, the information systems are running significant similar software. This is called IT monoculture [17]. On one hand, IT monoculture presents

several advantages including easier management, less configurations errors and support for inter-operability. On the other hand, IT monoculture has serious security concerns because similar systems share common vulnerabilities, and consequently, facilitates spread of viruses and malware.

The principle of diversity can be used to mitigate the effects of IT monoculture on software system. Diversity has been used to complement redundancy in order to achieve software systems reliability and fault tolerance. When it comes to security, the approach based on diversity seeks specifically to reduce the common vulnerabilities between redundant components of a system. As a result, it becomes very difficult for a malicious opponent to design one unique attack that is able to exploit different vulnerabilities in the system components simultaneously. Therefore, the resistance of the system to cyber attacks is increased. Moreover, 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.

In this paper we focus on how diversity can be used to generate dynamically a diverse set of light traces for the same behaviour of a software system. To this end, we define a setting which allows running in parallel several instances of a process. All these instances are provided with the same input. Each of these process instances runs on top of an operating system kernel which is instrumented differently to provide traces of the system calls pertaining to different important functionalities of the kernel. We raise in this paper some research question that need to be addressed.

The remaining part of the paper is organized as follows: In Section 2, we introduce the main idea underlying the approaches using software diversity for security purposes. We devote Section 3 to review and evaluate the state-of-the-art approaches based on software diversity to mitigate the risk associated with the IT monoculture. We outline and discuss in Section 4 an approach which aims at enabling the dynamic generation of a *diverse* set of lightweight and complementary traces from a running software application. We conclude the paper in Section 5.

2 Diversity as a Software Security Enabler

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) [2]. These attacks can ultimately enable the violation of the system security property (i.e. security failure) [2]. 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 [11] 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 [9] the different levels of diversity of software and hardware systems and distinguish different dimensions and different degrees of diversity. Bain et al. [3] 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. Taylor and Alves-Foss report in [23] 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.

3 Diversity-based approaches to Software Security

We have undertaken a comprehensive study to evaluate the state-of-the-art approaches based on the principle of software diversity to mitigate the risk of IT monoculture and enable software security [14]. These approaches can be classified into the three main following categories.

3.1 System Integration and Middlware

This category include proposals of software architectures which deploy redundancy combined with software diversity either by using integrating multiple Commercial-Off-The-Shelf (COTs) applications coordinated through a proxy

component or by defining and using a middleware to achieve the same purpose.
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 [10][26], the Scalable Intrusion Tolerant Architecture (SITAR) [28], and Hierarchical Adaptive Control for QoS Intrusion Tolerance (HACQIT) [19].

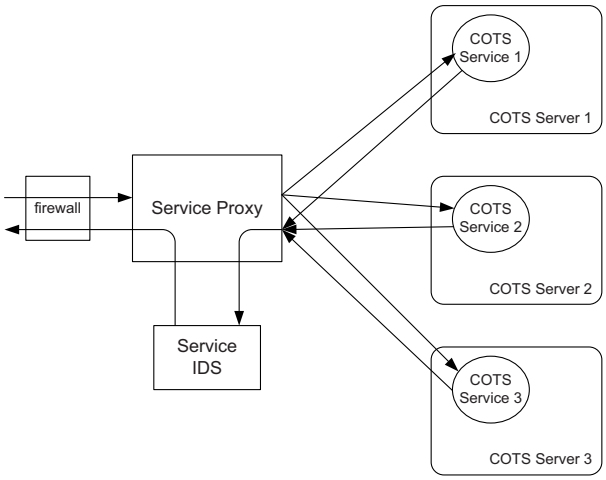

Fig. 1. General Pattern of Intrusion Tolerance Architecture

Middleware-based approaches are much richer since they can provide server coordination between multiple "diverse" applications while hiding the sub-system differences [20]. 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 [18]. 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) [27] 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) [1] [7] is a survivability architec-

ture 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 [7]. 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) [5] 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.

3.2 Software Diversity through Automated 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 [4][16] 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. It is worth noticing that ASR has been integrated into the default configuration of the Windows Vista operating system [30].

Data Space Randomization (DSR) is a different randomization-based approach which aims also at defending against memory error exploits [6]. 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.

3.3 Dynamic Behavior Monitoring

The ability to build a system combining redundant and diverse components provides new powerful capabilities in terms of 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. Several experimental systems used output voting for the sake of detecting some types of server compromising. For example, the HACQIT system [19] uses the status codes of the server replica responses. If the status codes are different the system detects a failure. Totel et al. [25] 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. N-variant systems provide a framework which allows executing a set of automatically diversified variants using the same input [8]. 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. Multi-variant code execution is a runtime monitoring technique which prevents malicious code execution [29]. 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. 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 [12][13].

4 Towards a Diversity-based Approach for Dynamic Generation of Lightweight Traces

The comprehensive dynamic monitoring of an operating system kernel such as Linux kernel is a daunting and challenging task. Indeed, it yields massive traces which are very difficult to be dealt with and in particular to be abstracted correctly to reach systematically meaningful information [15]. The principle of diversity can be potentially leveraged to address this issue. The main idea is to deploy a set of *redundant* Linux nodes running in parallel. This set can also includes deliberately a subset of replicas that are purposefully vulnerable. The diversity is introduced by the fact that the replicas are monitored differently. Indeed, the focus on each Linux kernel replica is put on different (predetermined) perspectives. These include the main kernel services such as memory management, file system management, networking sockets, interrupts, etc.

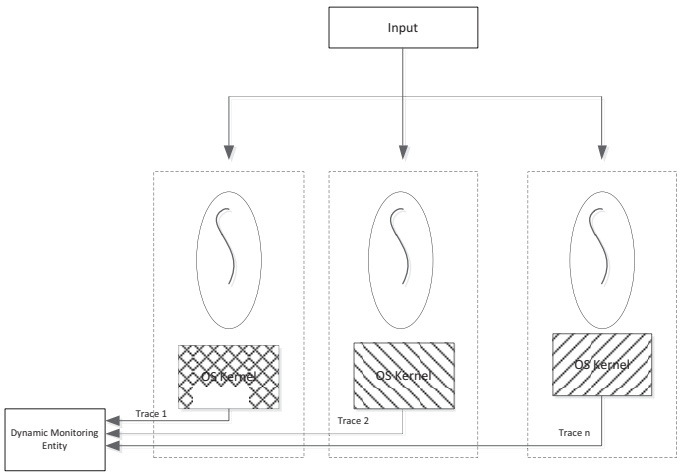

Fig. 2. Architecture for Diversity-based Dynamic Monitoring

The general software system architecture outlined Figure 2 aims at enabling the dynamic generation of a diverse set of traces of the behavior of a software application. Each of these traces is a sub-trace of the whole trace of the software application and it reflects a particular functionality of the operating system kernel. For an software application deployed in this setting N processes are spawn and run in parallel. Each of these processes runs in the environment of an operating system where the kernel has been instrumented to provide the trace of a particular functionality such as the memory management functionality, file management functionality, networking management, input/output drivers etc.

All the instances of the application running are provided with the same input which might be a malicious input (i.e. an attack). The generated traces are collected by a monitoring entity which is in charge of analyzing and correlating them using techniques that need to be investigated as discussed in the following section.

This dynamic monitoring configuration would yield a diverse set of much more lightweight traces. The latter are are sub-traces of the comprehensive trace of the running application. We are interested in investigating several research questions which can be raised using this monitoring setting. These include the identification of correlations between the different traces both in normal (i.e. healthy and secure system) and abnormal situations (system under attack or intrusion) as well as the identification of malicious behavior patterns.

5 Conclusion

Software systems security is a critical issue. An important contributing factor to this issue is the significant similarity in the software used in such systems. This is called IT mono-culture. The mitigation of this issue consists in using diversity which aims at reducing the common vulnerabilities and consequently increasing the difficulty of breaking systems built with diversity in mind.
In this article we focus on how diversity can be deployed to enable software behaviour dynamic monitoring to the end of intrusion detection. We have presented a diversity based approach which aims at generating dynamically traces pertaining to different functionalities of the operating system kernel. These traces, which are the sub-traces of the comprehensive trace of the software application behaviour, are therefore smaller. We are interested to investigate the different correlations and patterns that we can discover between these sub-traces in situation where the software application is healthy and secure and when it is compromised.

References

1. Atighetchi, M., Rubel, P., Pal, P.P., Chong, J., Sudin, L.: Networking aspects in the dpasa survivability architecture: An experience report. In: Fourth IEEE International Symposium on Network Computing and Applications (NCA 2005). pp. 219–222. IEEE Computer Society (2005)
2. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.E.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Sec. Comput. 1(1), 11–33 (2004)
3. Bain, C., Faatz, D.B., Fayad, A., Williams, D.E.: Diversity as a defense strategy in information systems. does evidence from previous events support such an approach? In: Gertz, M., Guldentops, E., Strous, L. (eds.) Fourth Working Conference on Integrity, Internal Control and Security in Information Systems, IICIS'01. IFIP Conference Proceedings, vol. 211, pp. 77–94. Kluwer (2001)

4. Barrantes, E.G., Ackley, D.H., Palmer, T.S., Stefanovic, D., Zovi, D.D.: Randomized instruction set emulation to disrupt binary code injection attacks. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) Proceedings of the 10th ACM Conference on Computer and Communications Security. pp. 281–289. ACM (2003)
5. Bessani, A.N., Reiser, H.P., Sousa, P., Gashi, I., Stankovic, V., Distler, T., Kapitza, R., Daidone, A., Obelheiro, R.R.: Forever: Fault/intrusion removal through evolution & recovery. In: Douglis, F. (ed.) ACM/IFIP/USENIX 9th International Middleware Conference. pp. 99–101. ACM (2008)
6. Bhatkar, S., Sekar, R.: Data space randomization. In: Zamboni, D. (ed.) Detection of Intrusions and Malware, and Vulnerability Assessment, 5th International Conference, DIMVA 2008. Lecture Notes in Computer Science, vol. 5137, pp. 1–22. Springer (2008)
7. Chong, J., Pal, P.P., Atighetchi, M., Rubel, P., Webber, F.: Survivability architecture of a mission critical system: The dpasa example. In: 21st Annual Computer Security Applications Conference (ACSAC 2005). pp. 495–504. IEEE Computer Society (2005)
8. Cox, B., Evans, D., Filipi, A., Rowanhill, J., Hu, W., Davidson, J., Knight, J., Nguyen-Tuong, A., Hiser, J.: N-variant systems: a secretless framework for security through diversity. In: USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium. USENIX Association, Berkeley, CA, USA (2006)
9. Deswarte, Y., Kanoun, K., Laprie, J.C.: Diversity against accidental and deliberate faults. In: Ammann, P., Barnes, B.H., Jajodia, S., , Sibley, E.H. (eds.) Computer Security, Dependability, and Assurance: From Needs to Solutions. p. 171181. IEEE Computer Press, Williamsburg, VA, USA (November 1998)
10. Deswarte, Y., Powell, D.: Intrusion tolerance for internet applications. In: Jacquart, R. (ed.) Building the Information Society, IFIP 18th World Computer Congress,. pp. 241–256. Kluwer (2004)
11. Forrest, S., Somayaji, A., Ackley, D.H.: Building diverse computer systems. In: Workshop on Hot Topics in Operating Systems. pp. 67–72 (1997)
12. Gao, D., Reiter, M.K., Song, D.X.: Behavioral distance measurement using hidden markov models. In: Zamboni, D., Krügel, C. (eds.) Recent Advances in Intrusion Detection, 9th International Symposium, RAID'06. Lecture Notes in Computer Science, vol. 4219, pp. 19–40. Springer (2006)
13. Gao, D., Reiter, M.K., Song, D.X.: Behavioral distance for intrusion detection. In: Valdes, A., Zamboni, D. (eds.) Recent Advances in Intrusion Detection, 8th International Symposium, RAID'2005. Lecture Notes in Computer Science, vol. 3858, pp. 63–81. Springer (2006)
14. Gherbi, A., Charpentier, R., Couture, M.: Redundancy with diversity based software architectures for the detection and tolerance of cyber-attacks. Technical Memorandum TM-2010-287, Defence Reasearch and Development Canada - DRDC Valcartier (2010)
15. Hamou-Lhadj, A.: Measuring the complexity of traces using shannon entropy. In: Fifth International Conference on Information Technology: New Generations (ITNG 2008). pp. 489–494. IEEE Computer Society (2008)
16. Kc, G.S., Keromytis, A.D., Prevelakis, V.: Countering code-injection attacks with instruction-set randomization. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) Proceedings of the 10th ACM Conference on Computer and Communications Security. pp. 272–280. ACM (2003)
17. Lala, J.H., Schneider, F.B.: It monoculture security risks and defenses. IEEE Security & Privacy 7(1), 12–13 (2009)

18. Pal, P.P., Rubel, P., Atighetchi, M., Webber, F., Sanders, W.H., Seri, M., Ramasamy, H.V., Lyons, J., Courtney, T., Agbaria, A., Cukier, M., Gossett, J.M., Keidar, I.: An architecture for adaptive intrusion-tolerant applications. Softw., Pract. Exper. 36(11-12), 1331–1354 (2006)
19. Reynolds, J.C., Just, J.E., Lawson, E., Clough, L.A., Maglich, R., Levitt, K.N.: The design and implementation of an intrusion tolerant system. In: International Conference on Dependable Systems and Networks (DSN 2002). pp. 285–292. IEEE Computer Society (2002)
20. Sames, D., Matt, B., Niebuhr, B., Tally, G., Whitmore, B., Bakken, D.E.: Developing a heterogeneous intrusion tolerant corba system. In: DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks. pp. 239–248. IEEE Computer Society, Washington, DC, USA (2002)
21. Shacham, H., Page, M., Pfaff, B., Goh, E.J., Modadugu, N., Boneh, D.: On the effectiveness of address-space randomization. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) roceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004. pp. 298–307. ACM (2004)
22. Symantec: Symantec global internet security threat report – trends for 2008. Tech. Rep. Volume XIV, Symantec (2009)
23. Taylor, C., Alves-Foss, J.: Diversity as a computer defense mechanism : A panel. In: NSPW '05: Proceedings of the 2005 workshop on New security paradigms. pp. 11–14. ACM, New York, NY, USA (2005)
24. emerging risks team, L.: Digital risks: Views of a changing risk landscape. Tech. Rep. Volume XIV, Lloyd's (April 2009)
25. Totel, E., Majorczyk, F., Mé, L.: Cots diversity based intrusion detection and application to web servers. In: Valdes, A., Zamboni, D. (eds.) Recent Advances in Intrusion Detection, 8th International Symposium, RAID'05. Lecture Notes in Computer Science, vol. 3858, pp. 43–62. Springer (2006)
26. Valdes, A., Almgren, M., Cheung, S., Deswarte, Y., Dutertre, B., Levy, J., Saïdi, H., Stavridou, V., Uribe, T.E.: Dependable intrusion tolerance: Technology demo. In: 3rd DARPA Information Survivability Conference and Exposition (DISCEX-III 2003). pp. 128–130. IEEE Computer Society (2003)
27. Veríssimo, P., Neves, N.F., Cachin, C., Poritz, J.A., Powell, D., Deswarte, Y., Stroud, R.J., Welch, I.: Intrusion-tolerant middleware: the road to automatic security. IEEE Security & Privacy 4(4), 54–62 (2006)
28. Wang, F., Jou, F., Gong, F., Sargor, C., Goseva-Popstojanova, K., Trivedi, K.: Sitar: A scalable intrusion-tolerant architecture for distributed services. In: Foundations of Intrusion Tolerant Systems. IEEE Computer Society, Los Alamitos, CA, USA (2003)
29. Weatherwax, E., Knight, J., Nguyen-Tuong, A.: A model of secretless security in n-variant systems. In: Workshop on Compiler and Architectural Techniques for Application Reliability and Security (CATARS), In the 39th Annual IEEE/IFIP International Conference on Dependable Systems and Network (DSN2009) (2009)
30. Whitehouse, O.: An analysis of address space layout randomization on windows vista. Tech. rep., Symantec (2007)