A Survey on Aims and Environments of Diversification and Obfuscation in Software Security

Shohreh Hosseinzadeh, Sampsa Rauti, Samuel Laurén, Jari-Matti Mäkelä, Johannes Holvitie, Sami Hyrynsalmi, Ville Leppänen

Abstract: *Diversification and obfuscation methods are promising approaches used to secure software and prevent malware from functioning. Diversification makes each software instance unique so that malware attacks cannot rely on the knowledge of the program's execution environment and/or internal structure anymore. We present a systematic literature review on the state of-the-art of diversification and obfuscation research aiming to improve software security between 1993 and 2014. As the result of systematic search, in the final phase, 209 related papers were included in this study. In this study we focus on two specific research questions: what are the aims of diversification and obfuscation techniques and what are the environments they are applied to. The former question includes the languages and the execution environments that can benefit from these two techniques, while the second question presents the goals of the techniques and also the type of attacks they mitigate.*

Key words*: Software security, diversification, obfuscation, systematic literature review (SLR)*

INTRODUCTION

To make software secure, the risk caused by the vulnerabilities in software needs to be alleviated. Instead of eliminating the security holes, in this work we focus on making the exploitation of software vulnerabilities harder. For this purpose, our research concentrates on two promising software security approaches.

The first approach is *obfuscation*. Obfuscation transforms the program code so that it remains functionally identical to the original code but is much more difficult for the adversary to understand [7]. Reverse-engineering the code will become more onerous and costly [23]. Obfuscation does not guarantee that the attacker cannot tamper with the targeted program (given enough time, any obfuscation can be undone), but it makes this task significantly more difficult to accomplish, adding an extra layer of defence. There are several obfuscation methods exist that scramble different parts of the code during different stages of software development process.

The second approach, *diversification*, means changing the internal structure of software in order to create uniquely diversified copies of it [11]. Now computers can run unique instances of the same program that all have the exactly same functionality but differently diversified code. Software monoculture, where programs have identical internal structure on a huge amount of machines, is broken, and malware counting on the known internal implementation details is rendered useless [28]. It is worth noting that in order to preserve the trusted programs' access to the essential resources, we have to propagate the diversification to all trusted libraries and applications in the system.

Diversification makes it more arduous for malware to exploit vulnerabilities and launch a successful attack. Even if a malicious program somehow succeeds in running its code on one computer, this attack only works on that specific computer. Because of unique diversification secrets, a costly analysis has to be performed on each computer separately in order to attack it. Because of this, diversification especially decreases the risk of automated large-scale attacks.

CompSysTech'16, 23-24 June 2016, Palermo, Italy
 ISBN 978-1-4503-4182-0/16/06...$15.00
http://dx.doi.org/10.1145/2983468.2983479

We conducted a systematic literature review of the papers discussing obfuscation and diversification techniques in the context of software security. In this study, we address two research questions. First, what is the *aim* of obfuscation or diversification. Second, for what *environments* are the obfuscation or diversification approaches proposed in order to improve security. This tells us what exactly is being protected – for instance, what is the programming language and the execution environment the techniques are applied to.

Of course, this setting of the study still leaves out some other interesting research questions. For example, what are the different obfuscation techniques usually used to protect software and how effective and costly these approaches are. However, to illustrate the wide usability of obfuscation and diversification techniques in different domains (e.g., IoT [15] and cloud computing [14]), we believe the aims and environments are the most interesting questions to discuss in this limited space. Also, different obfuscation transformations have already been studied to some extent elsewhere [6].

The rest of the paper is organized as follows. In the second section, we explain the method of the study. The third section discusses the aims of obfuscation and diversification approaches presented in the literature. The fourth section covers the environments where these approaches have been used. Finally, we present the conclusions and discuss future work.

METHOD OF THE STUDY

The method of the study we used in this research work is Systematic Literature Review (SLR). As defined by Kitchenham et al. [17], an SLR is a means of research to identify, evaluate and interpret all the studies related to one field of research. A systematic review has the benefit that it collects, classifies and maps the scattered studies. Therefore, it is a suitable way to pinpoint research gaps, which could give baselines for future research.

In this paper, we systematically review the papers that are published between 1993 and 2014 and study obfuscation and diversification techniques with the objective of securing the software. By collecting and analyzing the papers, we try to identify what kind of problems are being solved with the help of these techniques. In other words, for what purpose they are used, what type of software security attacks they thwart, and what type of environments and languages are these techniques applied to.

In order to collect the data systematically, we followed a protocol suggested by Kitchenham et al. [18]. In this process, shown in Figure 1, we first defined a set of search strings and searched into the top databases of the field, including IEEEXplore Digital Library, ACM Digital Library, Wiley online library, ScienceDirect, dblp, and SpringerLink (Phase I). Then we excluded the irrelevant studies based on their titles (Phase II), after that based on their abstracts (Phase III), and finally based on their full texts (Phase IV). Using the snowballing technique we found the missing studies (Phase V). To decide on inclusion or exclusion of the papers, we set some inclusion criteria. A study is included, if it answers all the inclusion criteria positively, and it is excluded otherwise. We used the following four inclusion criteria:

- Is the study related to software development/production?
- Is the study aims at improving the software security?
- Is the study related to obfuscation/diversification?
- Is obfuscation/diversification proposed/discussed/used with the goal of enhancing the security in code/program/software?

As the result of systematic search, we had 209 papers published in this field of study. From the final set of papers, we extracted the data, analyzed them and classified them, in order to answer our research questions.

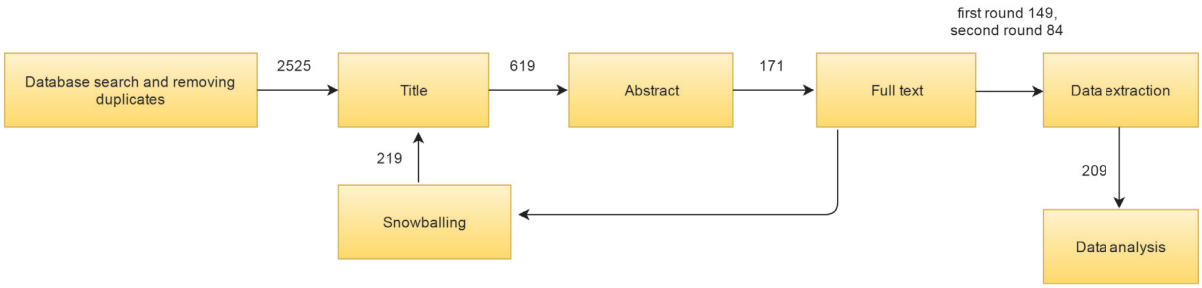

Figure 1: The Systematic Search and Selection Process used when conducting the systematic literature review.

There have been some other studies similar to this research work published previously [2, 19]. However, several factors make our SLR different from the existing works: First, we are following a systematic process for data collection. Second, we consider both obfuscation and diversification techniques, as we believe they are closely related. Finally, we study the papers in which security is the main scope of study.

AIMS OF OBFUSCATION AND DIVERSIFICATION

Obfuscation and diversification are techniques used for improving security of software. The rationale behind these two techniques is not to remove the security holes in the software but to prevent (or at least make it difficult) an adversary from taking advantage of the vulnerabilities. In the literature reviewed, typically, these two techniques were used for making the reverse engineering difficult [4], countering the static and dynamic analysis of the code [25], hiding some fraction of the code or some data inside the code [26], hiding or faking the original control flow of the program [9, 30], protecting the mobile agents against a malicious host, and preventing occurrences of large-scale [8] and targeted attacks.

Among all, in the following we discuss the top high level goals that the papers were trying to achieve, through obfuscation and diversification techniques. We will then continue by presenting the top attacks that were successfully mitigated with the help of these two techniques.

1. *Preventing malicious reverse engineering of the software:* the most common aim behind the use of obfuscation and diversification techniques is to make the program's code more complex, and harder to read/comprehend. As a result of this complication the act of reverse engineering of the program's logic becomes harder [22, 24]. Reducing the understandability of the software makes it resistant against unwanted modification [12].

2. *Preventing unauthorized alteration of the software:* tamper resistance approaches attempt to make it harder for an attacker to modify the program. For this purpose, these techniques can use methods that make comprehending the program difficult. In this scenario, obfuscation is helpful method for creating tamper-resistant programs [1, 21].

3. *Hiding some data in the code:* obfuscation can be used to conceal static non-executable data in the code, for instance hiding watermarking information, cryptographic keys, and static integers [26].

4. *Preventing the vulnerabilities to be widely spread:* Typically, the exploits work by gaining knowledge on the program's internal structure and its vulnerabilities, and exploiting those vulnerabilities. Obfuscation makes it challenging and costly (in terms of time and effort) for the attacker to achieve this knowledge. Furthermore, diversifying the program internals lowers the chance of large-scale attacks, that is, the same attack model is less likely to work on multiple targets [8].

We acknowledge that there are some overlaps between the different classes we presented here. Still, we tried to categorize the research works based on their high-level aims of using obfuscation and diversification techniques, in order to answer the question that what types of real-world problems are being solved by using the aforementioned techniques.

Attacks mitigated

As discussed, obfuscation and diversification techniques have widely been used to impede malware and defend against a wide range of attacks. Among all, the following are the most common type of attacks that the studied research works were aiming to mitigate, with the help of these two techniques: buffer overflow [29], code injection attacks [16], Return-Oriented Programming (ROP) [13], JIT spraying attacks [10], slicing attacks [10], insider attacks [27], piracy [3], large scale attacks [8], and web application attacks (including SQL injection and cross-site scripting (XSS) attacks [5]).

Figure 2 shows the distribution of these attacks in the reviewed papers.

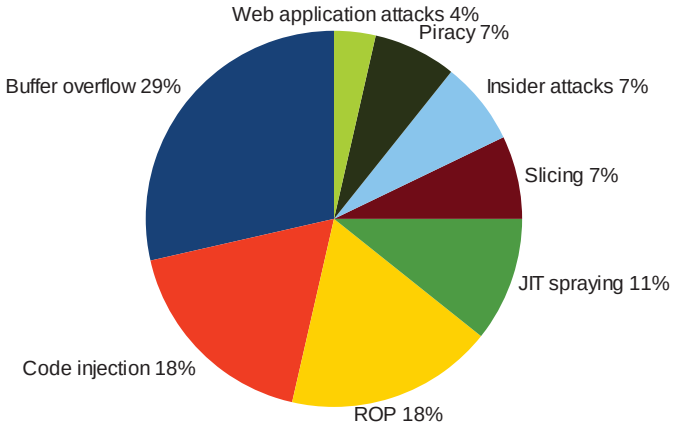

Figure 2: Different types of attacks that could be mitigated with the help of obfuscation/diversification techniques

ENVIRONMENT

To analyze the environments obfuscation or diversification were applied to, we examined each study from two standpoints. Firstly, the execution environment in which the technique is being employed and secondly, the language that was the target of technique.

Execution environments

Different execution environments secured by obfuscation or diversification are shown in Table 1. We can see that the largest group of papers describe approaches that are applied in any environment. This reflects the broad applicability of obfuscation and diversification in many software layers and different platforms. As any computer system runs native code on the lowest level of abstraction level, it is not surprising that the second largest group of obfuscation and diversification approaches deals with native code.

The other execution environments are split quite evenly into smaller groups. These are general classifications for devices and operating systems based on their uses, such as server, mobile and desktop. Cloud platform's third party services are distinguished from generic servers. Embedded platforms that are not mobile are also in their separate category.

Languages

Figure 3 shows the languages the obfuscation and diversification approaches are applied to. The languages are also divided into rough categories based on their runtime models

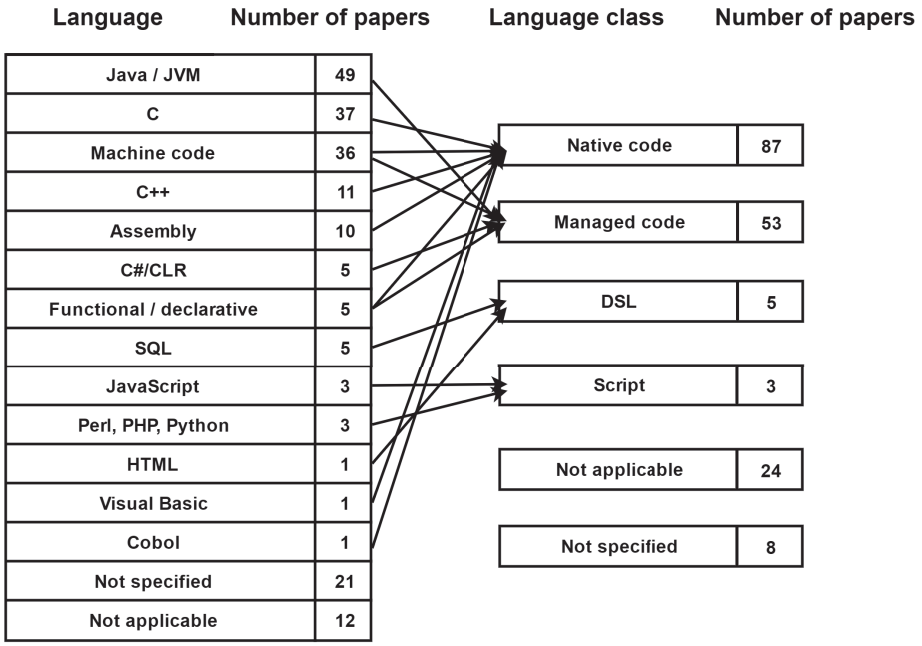

Figure 3: Languages that obfuscation and diversification techniques are applied to.

and semantic abstractions. For example, some programming languages such as C are usually compiled to native code while other languages like Java are transformed into bytecode that is run on a virtual machine. The class of each language is indicated by the arrows in Figure 3.

Taking a look at language categories, we can see that a majority of the cases fall into two language types. First, there is the native code category that contains all the local applications written in systems programming languages. Second, the managed code group is a category for applications that are written in some high-level run on a virtual machine (for instance, the Java programming language run on Java Virtual Machine). Two smaller groups identified in our study were domain specific languages like SQL and scripting languages such as Perl.

On the language level, Figure 3 lists all the proposed target languages, each placed in its own group. This provides sufficient granularity, even though some languages (like languages executed on JVM and functional languages) were still bundled together because of their similar execution environments. We can see that Java, C/C++ and Assembly languages are most often chosen as target for obfuscation or diversification.

For several papers, a language related to the proposed obfuscation approach could not be specified. Many studies also proposed very generic solutions covering a broad group of languages. In some cases, the specific language or class, or the applicability of obfuscation technique could not be determined. Some papers also presented abstract high-level solutions for which a language is not applicable. Moreover, for many obfuscation or diversifi-

Table 1: Environments for diversification and obfuscation approaches.

Target environment	Number of studies
Any	67
Any native	65
Server	8
Cloud	6
Distributed/agent based	6
Mobile	6
Embedded	3
Desktop	2

cation approaches targeting machine code binaries or bytecode the original language cannot be determined [20].

CONCLUSIONS AND FUTURE WORK

We presented a study of aims and environments of obfuscation and diversification techniques. Obfuscation and diversification are used for many different purposes in the context of security, mainly to prevent reverse-engineering and unauthorized modification of software. The two techniques have also been shown to mitigate a wide range of attacks. When it comes to the environment, the study shows the broad applicability of obfuscation and diversification for different languages, at several software layers and in various environments.

Even though this study is a carefully conducted systematic literature review, it still has some limitations.

As future work, a broader study covering more research questions will be conducted. In addition to aims and environments that we have studied in this work, there are various research questions worth considering. For instance, what is the current status of this field of study? This includes data like types of publications, number of published studies each year, the distribution of studies between the academia and industry. Also the obfuscation and diversification methods used provide an interesting topic to study. The exact targets that are obfuscated or diversified in the code, the levels and stages in development where techniques are applied, and the cost and effectiveness of techniques are going to be explored in our future studies.

REFERENCES

[1] Balachandran, V., Keong, N. W., and Emmanuel, S. Function level control flow obfuscation for software security. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2014 Eighth International Conference on* (July 2014), pp. 133–140.

[2] Baudry, B., and Monperrus, M. The multiple facets of software diversity: Recent developments in year 2000 and beyond. *ACM Comput. Surv. 48*, 1 (Sept. 2015), 16:1–16:26.

[3] Chakraborty, R., Narasimhan, S., and Bhunia, S. Embedded software security through key-based control flow obfuscation. In *Security Aspects in Information Technology*, M. Joye, D. Mukhopadhyay, and M. Tunstall, Eds., vol. 7011 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, pp. 30–44.

[4] Cho, S., Chang, H., and Cho, Y. Implementation of an obfuscation tool for c/c++ source code protection on the xscale architecture. In *Software Technologies for Embedded and Ubiquitous Systems*, U. Brinkschulte, T. Givargis, and S. Russo, Eds., vol. 5287 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 406–416.

[5] Christodorescu, M., Fredrikson, M., Jha, S., and Giffin, J. End-to-end software diversification of internet services. In *Moving Target Defense*, S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds., vol. 54 of *Advances in Information Security*. Springer New York, 2011, pp. 117–130.

[6] Collberg, C., Thomborson, C., and Low, D. A Taxonomy of Obfuscation Transformations. Tech. Rep. 148, The University of Auckland, 1997.

[7] Collberg, C. S., and Thomborson, C. Watermarking, tamper-proofing, and obfuscation - tools for software protection. *IEEE Transactions on Software Engineering 28*, 8 (2002), 735–746.

[8] Cox, B., E., D., Filipi, A., Rowanhill, J., Hu, W., Davidson, J., Knight, J., Nguyen-Tuong, A., and Hiser, J. N-variant systems: A secretless framework for security through diversity. In *Usenix Security* (2006), vol. 6, pp. 105–120.

[9] Darwish, S. M., Guirguis, S. K., and Zalat, M. S. Stealthy code obfuscation technique for software security. In *Computer Engineering and Systems (ICCES), 2010 International Conference on* (2010), IEEE, pp. 93–99.

[10] Drape, S., Majumdar, A., and Thomborson, C. Slicing aided design of obfuscating transforms. In *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on* (July 2007), pp. 1019–1024.

[11] Forrest, S., Somayaji, A., and Ackley, D. Building diverse computer systems. In *Operating Systems, 1997., The Sixth Workshop on Hot Topics in* (May 1997), pp. 67–72.

[12] Goto, H., Mambo, M., Shizuya, H., and Watanabe, Y. Evaluation of tamper-resistant software deviating from structured programming rules. In *Information Security and Privacy*, V. Varadharajan and Y. Mu, Eds., vol. 2119 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 145–158.

[13] Gupta, A., Kerr, S., Kirkpatrick, M., and Bertino, E. Marlin: A fine grained randomization approach to defend against rop attacks. In *Network and System Security*, J. Lopez, X. Huang, and R. Sandhu, Eds., vol. 7873 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 293–306.

[14] Hosseinzadeh, S., Hyrynsalmi, S., Conti, M., and Leppänen, V. Security and privacy in cloud computing via obfuscation and diversification: A survey. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)* (2015), pp. 529–535.

[15] Hosseinzadeh, S., Rauti, S., Hyrynsalmi, S., and Leppänen, V. Security in the internet of things through obfuscation and diversification. In *Computing, Communication and Security (ICCCS), 2015 International Conference on* (Dec 2015), pp. 1–5.

[16] Jiang, X., Wang, H. J., Xu, D., and Wang, Y. Randsys: Thwarting code injection attacks with system service interface randomization. In *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on* (Oct 2007), pp. 209–218.

[17] Kitchenham, B. Procedures for performing systematic reviews. *Keele, UK, Keele University 33*, 2004 (2004), 1–26.

[18] Kitchenham, B., and Brereton, P. A systematic review of systematic review process research in software engineering. *Information and Software Technology 55*, 12 (2013), 2049 – 2075.

[19] Larsen, P., Homescu, A., Brunthaler, S., and Franz, M. SoK: Automated software diversity. In *Security and Privacy (SP), 2014 IEEE Symposium on* (May 2014), pp. 276–291.

[20] Lee, B., Kim, Y., and Kim, J. binob+: A framework for potent and stealthy binary obfuscation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security* (New York, NY, USA, 2010), ASIACCS '10, pp. 271–281.

[21] Majumdar, A., and Thomborson, C. Manufacturing opaque predicates in distributed systems for code obfuscation. In *Proceedings of the 29th Australasian Computer Science Conference - Volume 48* (Darlinghurst, Australia, Australia, 2006), ACSC '06, Australian Computer Society, Inc., pp. 187–196.

[22] Monden, A., Monsifrot, A., and Thomborson, C. Obfuscated instructions for software protection. Tech. rep., 2003.

[23] Nagra, J., and Collberg, C. *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. Pearson Education, 2009.

[24] Qin, J., Bai, Z., and Bai, Y. Polymorphic algorithm of javascript code protection. In *Computer Science and Computational Technology, 2008. ISCSCT '08. International Symposium on* (Dec 2008), vol. 1, pp. 451–454.

[25] Schrittwieser, S., and Katzenbeisser, S. Code obfuscation against static and dynamic reverse engineering. In *Information Hiding*, T. Filler, T. Pevný, S. Craver, and A. Ker, Eds., vol. 6958 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, pp. 270–284.

[26] Sivadasan, P., and Lal, P. S. Securing SQLJ source codes from business logic disclosure by data hiding obfuscation. *arXiv preprint arXiv:1205.4813* (2012).

[27] Tunc, C., Fargo, F., Al-Nashif, Y., Hariri, S., and Hughes, J. Autonomic resilient cloud management (arcm) design and evaluation. In *Cloud and Autonomic Computing (ICCAC), 2014 International Conference on* (Sept 2014), pp. 44–49.

[28] Williams, D., Hu, W., Davidson, J., Hiser, J., Knight, J., and Nguyen-Tuong, A. Security through diversity: Leveraging virtual machine technology. *Security Privacy, IEEE 7*, 1 (Jan 2009), 26–33.

[29] Xu, J., Kalbarczyk, Z., and Iyer, R. Transparent runtime randomization for security. In *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on* (Oct 2003), pp. 260–269.

[30] Yao, X., Pang, J., Zhang, Y., Yu, Y., and Lu, J. A method and implementation of control flow obfuscation using seh. In *Multimedia Information Networking and Security (MINES), 2012 Fourth International Conference on* (Nov 2012), pp. 336–339.

ABOUT THE AUTHORS

PhD student Shohreh Hosseinzadeh, PhD student Sampsa Rauti, Research assistant Samuel Laurén, PhD student Jari-Matti Mäkelä, PhD student Johannes Holvitie, Postdoctoral researcher Sami Hyrynsalmi, Prof. Ville Leppänen, Department of Information Technology, University of Turku, Finland. E-mail: {shohos,sjprau,smrlau,jmjmak,jjholv,sthyry, ville.leppanen}@utu.fi.