International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Study of Code Obfuscation Techniques for the Security of Software Components.

Miguel Rodríguez-Veliz[1], Yulier Nuñez-Musa, Roberto Sepúlveda-Lima[2]

Submitted: 26/05/2023 **Revised**: 06/07/2023 **Accepted**: 27/07/2023

Abstract: The objective of the study is based on analyzing documentary information referring to code obfuscation techniques for the security of software components, from the perspective of the State of the Art. The research design is developed under a qualitative approach, of a documentary nature, in which referential information on the methods and techniques to carry out the processes of data assurance in computer systems is analyzed, based on the aforementioned theoretical and the analysis of the information collected without the manipulation of variables. The population under study is based on documents such as scientific articles, research papers and bibliographic reference material. To collect the data, content sheets are used, which represent the basis for the content analysis of the sources consulted. With the analysis of the results of these studies, it is intended to carry out an investigation of the theoretical aspects that support the obfuscation of security codes from its origins and how it has gradually advanced in its implementation in software.

Keywords: *source code, obfuscation, software, techniques.*

1. Introduction

In today's world, the massification of web services (Web Service) as software component elements that serve as the basis for the preparation of more advanced and complex computer systems, require the search for the automation of processes and the use and protection of information, promoting that every day innovates in the development of applications with the purpose of covering the diversity of user requirements.

In this situation, applications become vulnerable to potential cyber attacks; In addition, the illegal appropriation of source codes of certain software, can represent essential damages both for the application of the computer system, and for the organization that produces it, since losses are generated before the born competitor through the advancement in innovation and product development; the competitor can detect possible failures in the product and take advantage of them to discredit both the product and the manufacturer and; The most essential in the computer field, such as the decryption of modern algorithms, which can have an impact on their modification, including direct attacks on encryption mechanisms.

And, that is why accessing the source code of a software or application increases the possibilities of the so-called "cracks", violating anti-piracy systems (software records through keys, expiration dates, hardlocks, etc.). Therefore, it becomes essential to handle the terminology related to obfuscation, which according to the Royal Spanish Academy, can be defined in a general way as the deliberate concealment of the meaning of a certain thing, making it more confusing and complicated to interpret, to avoid understanding it.

Hence, three types of mechanisms or protection measures to counter attacks on applications are highlighted: ethical (amnesty, appeal and shareware) that seek to emphasize the incorrectness of this activity and its consequences; legal ones (copyright, patent and software license) based on possible sanctions and; technical mechanisms, among which stand out the diversity of code, watermarks, resistance to software modification (code encryption and self-verification of integrity) and obfuscation, the latter being the one that allows to guarantee the privacy of the software; That is, it keeps the structure, its operation and the data handled by the software hidden from an attack.

Accordingly, for this research the State of the Art will initially be addressed, where Cappaert (2012) in his work *Code obfuscation techniques for software protection*, states that:

"Code obfuscation is a set of transformations that turns a program into a functionally equivalent, but unintelligible one, making it difficult to understand and reverse engineer. Code obfuscation applies one or more code transformations that make code more resistant to analysis and manipulation, but preserve its functionality."

And, is that code obfuscation is such a wide area for exploration, that it has advanced by leaps and bounds towards systems relatively simple to use, but at the same time with a complex level of defense against attacks; This applies even to physical components (hardware) in which elements are used internally in the circuits to generate a

[1]*Facultad de Ciencias Informáticas, Universidad Técnica de Manabí, Portoviejo, Ecuador.*
[2]*Facultad de Ingeniería Informática, Universidad Tecnológica de la Habana 'José Antonio Echeverría', La Habana, Cuba*
E-mail: miguel.rodriguez@utm.edu.ec

"logical obfuscation" that prevents access to the constituent information of the so-called chips.

So much so, that according to Berón et al (2006) they point out that software obfuscation is a technique to hide the flow of control of software, as well as data structures containing sensitive information. It is also applied for the mitigation of the threat of reverse engineering, so it is considered a technique to package and compress a malicious program allowing it to go undetected. The obfuscation technique can occur in a simple way, converting strings to hexadecimal or as complex as developing custom algorithms for detailed translations.

This contrasts with what Aucsmith (1996) pointed out who proposed the use of multiple threads of execution in the software to confuse potential attackers when determining which is the correct thread to follow. This has an impact on the fact that the deterministic execution of control obfuscation algorithms (with the same input always offers the same output) allowing attackers to obtain more information, which facilitates a satisfactory attack. Therefore, taking this factor as a premise, obfuscation is approached with a non-deterministic behavior (with the same input it offers multiple outputs or results, without knowing it in advance, following a certain probability distribution) in the design of the protection mechanism.

In the same vein, Ferrari et al (2020) in their article referring to *ATENOS: A program to improve security in WSDL,* point out that competing companies can learn the know-how and get to copy the design to offer similar and competitive services. This goes beyond simply competing, this implies the possibility of security attacks such as information espionage, client impersonation, command injection and denial of service, in addition to the possibility that attackers learn about the data exchanged and the invocation patterns of WSDL documents, in which both formal information of the source code is safeguarded, as informal related to identifiers, comments, documentation, etc.

Given this situation, as noted in the document published by the Ponemon Institute in 2016, called the *2016 Data Breach Cost Study*, they consider that the impact of a data breach can be disastrous for an organization and can include losses of trust and customer loyalty, financial penalties and other consequences. And, despite the high risk of threats, companies continue to be victims of data breaches worldwide, which generates instability in the environment of the protection of the data that organizations own, process and store.

Likewise, while external threats represent a high priority, the threat to sensitive data also comes from insider threats. Among them, the threat of employees stealing customer information, personally identifiable information or credit card data are real due to the fact that, in most cases, privileged users, such as system administrators or database administrators, have authorized access to the data. Often, the actual data from the production environment is copied to the non-production environment, which is less secure and not managed with the same security controls as the production environment, and the resulting data can be exposed or stolen.

Faced with this scenario, obfuscation techniques offer different ways to ensure that data remains protected from falling into the wrong hands, and fewer people can access sensitive information, while meeting the requirements of the organization. In this sense, this article analyzes the documentary information referring to the obfuscation techniques of security codes of software components, from a theoretical vision through the investigation of referential information of the techniques for information assurance.

2. Materials and Methods

Obfuscation

It is the technique of transforming the code into another less readable and difficult to understand to the human being, but maintaining its functionality, in this way the code is modified to hide its original purpose and drown the aggressor of information (Chen et al., 2017). Hence, in the technological world, obfuscation is also known as data masking, considered as the process of replacing sensitive information existing in test or development environments with information that looks like real production information, but does not serve anyone who wants to misuse it.

So much so, that source code obfuscation is considered a technique consisting of altering the source code of a program or software, in such a way that it is difficult to understand or modify by a person who examines it without authorization. Such a technique is commonly used in the software industry to protect intellectual property and prevent reverse engineering. There are several source code obfuscation techniques that can be used to achieve this goal, among them can be described the most common:

- Variable Renaming: This technique consists of changing the names of variables, functions and methods of the source code to random or meaningless names. This makes it more difficult for an attacker to understand the code and understand how it works.
- Instruction obfuscation: This technique involves modifying the instructions in the source code to make them more difficult to understand. For example, unconventional arithmetic operators or masking techniques can be used to make code difficult to understand.

- Insertion of false code: This technique consists of inserting false code into the source code. This code has no effect on the execution of the program, but it makes it difficult to understand the actual code.
- Control flow obfuscation: This technique consists of modifying the control flow of the program, so that it is difficult to follow. For example, conditional deviation techniques can be used to jump to different parts of the code based on certain conditions.
- String encryption: This technique involves encrypting the text strings used in the source code. This way, the strings are not readable in the binary file and it is more difficult for an attacker to find sensitive information.

And, the technological development in the assurance and protection of software has been increasing continuously in recent years; As code compilation has evolved far beyond simply translating computer software into an executable one. So much so, that applications are written in high-level languages because of the possibilities they represent, given the advantages they offer. However this has certain disadvantages that make code compilation implicitly include numerous optimization techniques ranging from removing dead code, optimal register allocation, and efficient mappings to the architecture's target instruction set.

Therefore, in the stages of compilation and / or software development is where different source code transformations can be performed and applied; To which many techniques have been developed to maximize the obfuscation of code and the different parts that make up a software, making its analysis extremely difficult. Some as simple as encoding the name of identifiers, but code obfuscation offers many more possibilities and varieties. An efficient obfuscation is constituted of one or more code transformations that in turn transform a software in such a way that it is much more difficult to apply reverse engineering, being the only restriction for these transformations, the fact that they are manual or automated, since what is sought is the preservation and functionality of the original software.

Thus, Collberg et al. (1997) point out that code transformations to obfuscate a program can be divided into four main classes, such as:

- Lexical or design transformations: These affect information in code that is unnecessary for its execution and that reduces the information available to a human reader.
- Control flow transformations: These types of transformations act by modifying the control flow of the program.
- Data flow transformations: They operate on the basis of the data structures used in the program.
- Preventive transformations: These attempt to stop the correct functioning of code decompilers or deobfuscators.

Given these actions, those responsible for ensuring the source codes of digital environments, ensure that test users or developers do not have access to see real production data, trying as much as possible that as long as what they are seeing seems real and coherent. Therefore, obfuscation techniques are used to protect data from the source code matrix for the protection of sensitive information and masking identifiable information with realistic values, allowing companies to mitigate the risk of data exposure.

One of the main adverse factors that occur in software, mainly those that are executed in client, is that the source code responsible for hosting the execution logic must be downloaded to be compiled and executed, which means that the work carried out by software developers is exposed to any user who runs the application in their client tool; Therefore, the foundation that is implemented for code obfuscation implies:

- Intellectual property: It is presented when the manufacturer or the technical team in charge of implementing software code, does not want it to be copied, modified or implemented in different systems without prior authorization.
- Information security: If the developed code is obfuscated and additional tools are provided that increase its security levels in the protection of information, it makes it difficult to analyze the code in search of system vulnerabilities, making it difficult to apply reverse engineering tasks to the developed software.
- Performance: Depending on the obfuscation technique applied, advantages can be visualized in obfuscation, such as software performance, since some techniques lighten the weight of the code and make it much easier for tools to execute them.

Given the theme raised, the study of source code obfuscation techniques for the security of software components is outlined, from a methodological perspective in which the design of the research governs its basis in a qualitative approach, of a documentary nature, in which information referred to the case study is analyzed. To do this, it starts from theoretical documentation and carries out the analysis of the information obtained without the manipulation of variables to generate a process of inquiry that guides in the contextualization and shaping of the State of the Art. The referred approach is approached with a collection of data without numerical measurement to discover or refine research questions in the research process (Hernández et al., 2006).

Similarly, it is established as a non-experimental research, which for Hernández et al. (2001) is nothing more than one that is carried out without deliberately manipulating variables. Simply put, it is an investigation where the independent variables do not intentionally vary. Likewise, Tamayo and Tamayo (1994) cited in Martínez (2018) point out that the research is descriptive, since it records, analyzes and interprets information from the composition or processes of the phenomena.

In the same vein, Arias (2012) states that a research technique consists of the particular procedure of obtaining information, using data collection instruments. Those instruments should respond to considerations of reliability, validity and objectivity. The techniques and instruments used in the development of the research are described below:

- Content tabs: content tabs allow you to preserve the data that is obtained in an organized and visible way. They record both the information found in the documents, as well as the thoughts, comments and arguments resulting from the analysis of the reading. For this, documentary sources are selected where the central topics to be investigated are found: obfuscation, techniques and methods, software components, security (Pulido, 2007).
- Bibliographic file: a register is made where the core elements that identify the documents are noted, since, due to their nature, the data vary, some of these elements include the author's information, title of documents, publisher and place of publication among others (Gaos, 2002).
- Content analysis: it is applied to analyze written, sound or visual documents such as: newspapers, magazines, books, movies, television programs, advertising, among others. Seeking to make a description of the categories product of the information found in the bibliography consulted and from there to be able to base the final considerations (Giroux & Álvarez, 2004)

In this sense, the research carries out a qualitative analysis of bibliographic references that form the foundations of the documentary base of the research, guided by a database search process of certified and digitized journals such as the Elsevier Scopus platform, among other academic databases, delimiting the period of data collection in the last ten years, of which we selected 30 studies that correlate with this research. The following is a list of these investigations:

Table 1. Search equations in Elsevier Scopus database

Searches	Search Equations	Database
1	(TITLE-ABS-KEY ("technique obsfuscation security") AND TITLE-ABS-KEY (code)) AND PUBYEAR > 2013	Elsevier Scopus
2	(TITLE-ABS-KEY ("obsfuscation codes software") AND TITLE-ABS-KEY (components)) AND PUBYEAR > 2013	Elsevier Scopus
3	(TITLE-ABS-KEY ("web obsfuscation software") AND TITLE-ABS-KEY (source code)) AND PUBYEAR > 2013	Elsevier Scopus

Inclusion and exclusion criteria

In order to achieve the objective of this research, the inclusion and exclusion criteria for the selection of primary and secondary studies were defined based on the research questions posed.

Inclusion criteria

- Studies whose title, keywords, contain one or more terms related to the research questions and the abstract include information on source code obfuscation, encompassing the techniques and methods applicable to the assurance of software components.
- Studies on the application of obfuscation techniques in web services.
- Studies whose conclusions have been presented in a coherent manner from the theoretical or methodological point of view.

Exclusion criteria

- Studies that do not contain information related to the subject to be investigated.
- Studies that in the title, abstract and keywords include a term not related to the topic to be investigated.
- Studies not indexed in scientific publications.
- Documentary sources such as books.

The searches were generalized, so we proceeded to separate the phrases "security obfuscation technique" and, "software code obfuscation", detailing that, when used without any symbology, they generate a large number of results, which, later, when reviewed indicate that many studies are not related to the related topic, since it is oriented to the approach in the security of software components. In this sense, we proceeded to carry out a direct sampling of the studies that represent these phrases at a general level, pointing out the obfuscation techniques, the source codes and, specifically, in the software components.

Table 2. Results of the different searches

Searches	Search Equations	Items found	Selected
1	(TITLE-ABS-KEY ("technique obsfuscation security") AND TITLE-ABS-KEY (code)) AND PUBYEAR > 2013	434	14
2	(TITLE-ABS-KEY ("obsfuscation codes software") AND TITLE-ABS-KEY (components)) AND PUBYEAR > 2013	423	5
3	(TITLE-ABS-KEY ("web obsfuscation software") AND TITLE-ABS-KEY (source code)) AND PUBYEAR > 2013	265	11

Therefore, an exploration was detailed using the phrases in both Spanish and English in the Elsevier Scopus database, referencing studies since 2013, to handle a breadth of the evolution of the theme, which, in the case "technique obsfuscation security", together with the logical operator AND with the word "code". Likewise, a second search was carried out, where the phrase "obsfuscation codes software", without quotation marks, and the word "components", joined by the logical inclusion operator AND were carried out and, finally, a third search was carried out, where the words "web obsfuscation software" were used, without quotation marks, with the phrase "source code", without quotation marks, emphasizing that the three search criteria were applied in Spanish and English languages in the referred platform. As in the previous searches, the logical operator AND was used, for which a total of 1,122 open access article type documents were obtained that are related to the three searches carried out, of which the number of thirty (30) studies (Table 3) that are related to the research questions was taken as a representative sample for the research.

Table 3. Studies related to the proposed objectives. First search criterion

I am a student	Title	Author	Year	Fountain
E01	Enhance virtual-machine-based code obfuscation security through dynamic bytecode scheduling	Kuang, K., Tang, Z., Gong, X., Fang, D., Chen, X., & Wang, Z.	2018	ScienceDirect, computers & security 74 (2018) 202–220
E02	Dynamic defenses in cyber security: Techniques, methods and challenges	Zhen, Y., Li, Z., Xu, X, & Zhao, Q.	2022	Digital Communications and Networks 8 (2022) 422–435
E03	Diversification and obfuscation techniques for software security: A systematic literature review	Hosseinzadeh, S., Rauti, S., Laurén, S., Makela, J., Holvitie, J., Hyrynsalmi, S., &	2018	Information and Software Technology 104 (2018) 72–93
E04	ObfSec: Measuring the security of obfuscations from a testing perspective	Menéndez, H., & Suárez-Tangil, G.	2022	Expert Systems With Applications 210 (2022) 118298

E05	MOJI: Character-level convolutional neural networks for Malicious Obfuscated JavaScript Inspection	Ishida, M., Kaneko, N. & Sumi, K.	2023	Applied Soft Computing 137 (2023) 110138
E06	Different Obfuscation Techniques for Code Protection	Camargo-Ruiz, D., Tovar-Zambrano, E., & Niño- González, J.	2015	Procedia Computer Science 70 (2015) 757 – 763
E07	Detection of obfuscation in java malware	Kumar, R., & Essar, A.	2016	Procedia Computer Science 78 (2016) 521 – 529
E08	Looking for Criminal Intents in JavaScript Obfuscated Code	Cerutti, F., Barattieri, D., Gringoli, F., & Lamperti, G.	2022	Procedia Computer Science 207 (2022) 867–876
E09	Obfuscation of Documents using Randomly Generated Steps	Irimia, C., Irimia, R., Milea, R., Ilas, S., Vasiliu, A., & Iftene, A.	2022	Procedia Computer Science 207 (2022) 1581–1590
E10	Android application forensics: A survey of obfuscation, obfuscation detection and deobfuscation techniques and	Zhang, X., Breitinger, F., Luechinger, E., O'Shaughnessy, S.	2021	Forensic Science International: Digital Investigation 39 (2021) 301285
E11	ERMDS: A obfuscation dataset for evaluating robustness of learning-based malware detection system	Jia, L., Yang, Y., Tang, B., & Jiang, Z.	2023	BenchCouncil Transactions on Benchmarks, Standards and Evaluations
E012	Techniques of injection and obfuscation of PE files in the memory of a process and the analysis of this in Windows	Fagoaga-Sancho, J.	2020	Polytechnic University of Madrid. School of Computer Engineering. Degree Project.
E13	Analysis of malicious code concealment techniques for evasion of end-user protection systems and NIDS	Cabrera Laguapillo, M. S., & Larco Andrade, Y. C.	2018	Reposito Digital – EPN. National Polytechnic School. Quito, 2018.
E14	ATENOS: A Program to Improve Security at WSDL	Ferrari, A., Bernardis, E., Berón, M., Bernardis, H., Tinoco, M., Bustos,	2018	National University of San Luis. Department of Computer Science. Faculty of Physical, Mathematical and Natural Sciences, San Luis,
E15	Remote Management Software Prototype	Calderón-Ojeda, A., & Morocho-Méndez, G.	2020	Universidad Politécnica Salesiana Sede Cuenca. Systems Engineering. Technical Project. Cuenca, Ecuador.
E16	Encryption process with idea algorithm and code obfuscation in web services	Camargo-Ruiz, D., Tovar-Zambrano, E., & Niño- González, J.	2022	Magazine Links: Science, Technology and Society. Bogotá D.C., Colombia.

E17	Web Services Security	Bernardis, E., Bernardis, H., Berón, M., & Montejano, G.	2017	National University of San Luis. Department of Computer Science. Faculty of Physical, Mathematical and Natural Sciences, San Luis,
E18	Viewing functions as token sequences to highlight similarities in source code	Chilowicz, M., Duris, E., & Roussel, G.	2013	Science of Computer Programming 78 (2013) 1871–1891
E19	ENDMal: An anti-obfuscation and collaborative malware detection system using syscall sequences	Lu, H., Wang, X., Zhao, B., Wang, F., & Su, J.	2013	Mathematical and Computer Modelling 58 (2013) 1140–1154
E20	Techniques and Tools to Regulate Security in WSDL-Based Web Services	Bernardis, H., Bernardis, E., Berón, M., Riesco, D., & Pereira, M.	2018	National University of San Luis. Department of Computer Science. Faculty of Physical, Mathematical and Natural Sciences, San Luis,
E21	A taxonomy of assets for the development of software-intensive products and services	Zabardast, E., Gonzalez-Huerta, J., Gorschek, T., Smite, D.,	2023	The Journal of Systems & Software 202 (2023) 111701
E22	Darknet traffic classification and adversarial attacks using machine learning	Rust-Nguyen, N., Sharma, S., & Stamp, M.	2023	Computers & Security 127 (2023) 103098
E23	AMA: Static Code Analysis of Web Page For The Detection of Malicious Scripts	Seshagiri, P., Vazhayil, A., & Sriram, P.	2016	Procedia Computer Science 93 (2016) 768 – 773
E24	Combining multiple granularity variability in a software product line approach for web engineering	Horcas, J., Cortiñas, A., Fuentes, L., & Luaces, M.	2022	Information and Software Technology 148 (2022) 106910
E25	Phishing websites detection using a novel multipurpose dataset and web technologies features	Sánchez-Paniagua, M., Fidalgo, E., Alegre, E., &	2022	Expert Systems With Applications 207 (2022) 118010
E26	A machine learning approach to detection of JavaScript-based attacks using AST features and paragraph vectors	Ndichu, S., Kim, S., Ozawa, S., Misu, T., & Makishima, K.	2019	Applied Soft Computing Journal 84 (2019) 105721
E27	BinGold: Towards robust binary analysis by extracting the semantics of binary code as semantic flow graphs (SFGs)	Alrabaee, S., Wang, L., & Debbabi, M.	2016	Digital Investigation 18 (2016) S11-S22
E28	Plagiarism Detection for Java Programs without Source Codes	Anjali, V., Swapna, T.R., & Jayaraman, B.	2015	Procedia Computer Science 46 (2015) 749 – 758

E29	Android Malware Detection based on Vulnerable Feature Aggregation	Roy, A., Singh Jas, D., Jaggi, G., & Sharma, K.	2020	Procedia Computer Science 173 (2020) 345–353
E30	Guidelines for developing Web GIS applications	Acosta-Correa, B.S., & Yanza-Hurtado, A.V.	2013	FOLLOW. University of Girona. VII Free Software Conference.

The aforementioned studies are related from the theoretical perspective with the notions for the analysis of obfuscation techniques, investigating their elements to study the processes of source code assurance for software components, establishing the basic criteria for the contextualization of the State of the Art based on the types of obfuscation and how to implement it in technological tools. So much so, that the interaction between web services is established and how the programs provide them with the levels of encryption to increase security against potential cyber attacks, protecting in turn the source code matrix which represents the fundamental characteristic of the software and even the manufacturer.

3. Conclusions

Source code obfuscation represents the practice of transforming the source code of a software into a form in which the code is difficult for humans to understand or read, but is machine-readable. Given this, the primary objective of obfuscating is considered to protect both intellectual property and trade secrets of companies, seeking to prevent reverse engineering. Therefore, when analyzing from the vision of the State of the Art of referential information, it can be detailed that there are various techniques and tools applicable to programs, among them we have the obfuscators of names of variables and functions, which seek to rename variables and give random names to the functions, thus making it difficult to understand the source code.

Also, there are flow control obfuscators, which basically mix the flow control of the software, changing the order of the instructions or incorporating new ones that are not even necessary. There are also literal obfuscators, which encrypt literal values such as strings, numbers, and other expressions. Also, the structure obfuscators are presented, in which it is sought to divide the obfuscated software into smaller parts to mask the hidden relationships between them. As well as code transformers, which involve the application of code transformation algorithms, which include flow control obfuscation, name obfuscation, debugging information, and merging code blocks.

In general, source code obfuscation techniques have advanced significantly in recent years, and have become an important component for the protection of intellectual property and the security of the software industry, being clear also that obfuscation does not guarantee complete protection, and must be used in conjunction with other security techniques for adequate protection.

References

[1] Acosta-Correa, B.S., & Yanza-Hurtado, A.V. (2013). Guidelines for the development of Web GIS applications. FOLLOW. University of Girona. VII Free Software Conference.

[2] Alrabaee, S., Wang, L., & Debbabi, M. (2016). BinGold: Towards robust binary analysis by extracting the semantics of binary code as semantic flow graphs (SFGs). Digital Investigation 18 (2016) S11-S22

[3] Anjali, V., Swapna, T.R., & Jayaraman, B. (2015). Plagiarism Detection for Java Programs without Source Codes. Procedia Computer Science 46 (2015) 749 – 758

[4] Aucsmith (1996). Tamper resistant software: An implementation. In Proceedings of the First International Workshop on Information Hiding. Pages 317-333. London, UK.

[5] Bernardis, E., Bernardis, H., Beron, M., & Montejano, G. (2017). Security in Web Services. National University of San Luis. Department of Computer Science. Faculty of Physical, Mathematical and Natural Sciences, San Luis, Argentina

[6] Bernardis, H., Bernardis, E., Berón, M., Riesco, D., & Pereira, M. (2018). Techniques and Tools to Regulate Security in Web Services Based on WSDL. National University of San Luis. Department of Computer Science. Faculty of Physical, Mathematical and Natural Sciences, San Luis, Argentina

[7] Berón, M., Henriques, P., Varanda, M., & Uzal, R. (2006). Tools for understanding programs, June 2006.

[8] Cabrera Laguapillo, M. S., & Larco Andrade, Y. C. (2018). Analysis of malicious code concealment techniques for evasion of end-user protection systems and NIDS. 124 sheets. Quito: EPN.

[9] Camargo-Ruiz, D., Tovar-Zambrano, E., & Niño-González, J. (2015). Different Obfuscation

Techniques for Code Protection. Procedia Computer Science 70 (2015) 757 – 763

[10] Camargo-Ruiz, D., Tovar-Zambrano, E., & Niño-González, J. (2022). Encryption process with idea algorithm and code obfuscation in web services. Magazine Links: Science, Technology and Society. Bogotá D.C., Colombia.

[11] Cappaert, J. (2012). Code obfuscation techniques for software protection. Katholieke Universiteit Leuven.

[12] Cerutti, F., Barattieri, D., Gringoli, F., & Lamperti, G. (2022). Looking for Criminal Intents in JavaScript Obfuscated Code. Procedia Computer Science 207 (2022) 867–876

[13] Chen, Z., Jia, C., & Xu, D. (2017). Hidden path: dynamic software watermarking based on control flow obfuscation. International Conference on Computational Science and Engineering and International Conference on Embedded and Ubiquitous Computing. Ed. IEEE, vol. 2, pp. 443-450, ISBN: 1-5386-3221-7

[14] Chilowicz, M., Duris, E., & Roussel, G. (2013). Viewing functions as token sequences to highlight similarities in source code. Science of Computer Programming 78 (2013) 1871–1891

[15] Collberg, C., Carter, E., Debray, S., Huntwork, A., Kececioglu, J., & Linn, C. (2004). Dynamic path-based software watermarking. Conference on programming language design and implementation. Ed. ACM, vol. 39, pp. 107-118, DOI: 10.1145/996841.996856, ISBN: 1-58113-807-5

[16] Fagoaga Sancho, J. (2020).Techniques of injection and obfuscation of PE files in the memory of a process and the analysis of this in Windows systems. Final Project / Final Degree Project,E.T.S. of Computer Engineers (UPM), Madrid, Spain.

[17] Ferrari, A., Bernardis, E., Beron, M., Bernardis, H., Tinoco, M., Bustos, M., & Riesco, D. (2018). ATENOS: A Program to Improve Security in WSDL. National University of San Luis. Department of Computer Science. Faculty of Physical, Mathematical and Natural Sciences, San Luis, Argentina

[18] Hernández, F., et al (2001). Research methodology. McGraw-Hill Publishing. Mexico. https://www.uca.ac.cr/wp-content/uploads/2017/10/Investigacion.pdf

[19] Hernández, R., Fernández, C., Baptista, M. (2006). Definitions of quantitative and qualitative approaches, their similarities and differences. Research Methodology, 2–23. https://investigar1.files.wordpress.com/2010/05/1033525612-mtis_sampieri_unidad_1-1.pdf

[20] Horcas, J., Cortiñas, A., Fuentes, L., & Luaces, M. (2022). Combining multiple granularity variability in a software product line approach for web engineering. Information and Software Technology 148 (2022) 106910

[21] Hosseinzadeh, S., Rauti, S., Laurén, S., Makela, J., Holvitie, J., Hyrynsalmi, S., & Leppanen, V. (2018). Diversification and obfuscation techniques for software security: A systematic literature review. Information and Software Technology 104 (2018) 72–93

[22] Ponemon Institute (2016). Report on the cost of Data Gaps 2016. IBM Security.

[23] Irimia, C., Irimia, R., Milea, R., Ilas, S., Vasiliu, A., & Iftene, A. (2022). Obfuscation of Documents using Randomly Generated Steps. Procedia Computer Science 207 (2022) 1581–1590

[24] Ishida, M., Kaneko, N. & Sumi, K. (2023). MOJI: Character-level convolutional neural networks for Malicious Obfuscated JavaScript Inspection. Applied Soft Computing 137 (2023) 110138

[25] Jia, L., Yang, Y., Tang, B., & Jiang, Z. (2023). ERMDS: A obfuscation dataset for evaluating robustness of learning-based malware detection system. BenchCouncil Transactions on Benchmarks, Standards and Evaluations

[26] Kuang, K., Tang, Z., Gong, X., Fang, D., Chen, X., & Wang, Z. (2018). Enhance virtual-machine-based code obfuscation security through dynamic bytecode scheduling. ScienceDirect, computers & security 74 (2018) 202–220

[27] Kumar, R., & Essar, A. (2016). Detection of obfuscation in java malware. Procedia Computer Science 78 (2016) 521 – 529

[28] Lu, H., Wang, X., Zhao, B., Wang, F., & Su, J. (2013). ENDMal: An anti-obfuscation and collaborative malware detection system using syscall sequences. Mathematical and Computer Modelling 58 (2013) 1140–1154

[29] Menéndez, H., & Suárez-Tangil, G. (2022). ObfSec: Measuring the security of obfuscations from a testing perspective. Expert Systems With Applications 210 (2022) 118298

[30] Ndichu, S., Kim, S., Ozawa, S., Misu, T., & Makishima, K. (2019). A machine learning approach to detection of JavaScript-based attacks using AST features and paragraph vectors. Applied Soft Computing Journal 84 (2019) 105721

[31] Palella, S. & Martins, F. (2018). Methodology of quantitative research. Third Edition. FEDUPEL. http://gc.scalahed.com/recursos/files/r161r/w23578w/w23578w.pdf

[32] Royal Spanish Academy (2018). http://www.rae.es/

[33] Roy, A., Singh Jas, D., Jaggi, G., & Sharma, K. (2020). Android Malware Detection based on Vulnerable Feature Aggregation. Procedia Computer Science 173 (2020) 345–353

[34] Rust-Nguyen, N., Sharma, S., & Stamp, M. (2023). Darknet traffic classification and adversarial attacks using machine learning. Computers & Security 127 (2023) 103098

[35] Sánchez-Paniagua, M., Fidalgo, E., Alegre, E., & Alaíz, R. (2022). Phishing websites detection using a novel multipurpose dataset and web technologies features. Expert Systems With Applications 207 (2022) 118010

[36] Seshagiri, P., Vazhayil, A., & Sriram, P. (2016). AMA: Static Code Analysis of Web Page For The Detection of Malicious Scripts. Procedia Computer Science 93 (2016) 768 – 773

[37] Zabardast, E., González-Huerta, J., Gorschek, T., Smite, D., Alegroth, E., & Faberholm, F. (2023). A taxonomy of assets for the development of software-intensive products and services. The Journal of Systems & Software 202 (2023) 111701

[38] Zhang, X., Breitinger, F., Luechinger, E., & O´Shaughnessy, S. (2021). Android application forensics: A survey of obfuscation, obfuscation detection and deobfuscation techniques and their impact on investigations. Forensic Science International: Digital Investigation 39 (2021) 301285

[39] Zhen, Y., Li, Z., Xu, X, & Zhao, Q. (2022). Dynamic defenses in cyber security: Techniques, methods and challenges. Digital Communications and Networks 8 (2022) 422–435

[40] Mr. B. Naga Rajesh. (2019). Effective Morphological Transformation and Sub-pixel Classification of Clustered Images. International Journal of New Practices in Management and Engineering, 8(01), 08 - 14. https://doi.org/10.17762/ijnpme.v8i01.74

[41] Kulkarni, A. P. ., & T. N., M. . (2023). Hybrid Cloud-Based Privacy Preserving Clustering as Service for Enterprise Big Data. International Journal on Recent and Innovation Trends in Computing and Communication, 11(2s), 146–156. https://doi.org/10.17762/ijritcc.v11i2s.6037

[42] Dhabliya, D., Soundararajan, R., Selvarasu, P., Balasubramaniam, M. S., Rajawat, A. S., Goyal, S. B., . . . Suciu, G. (2022). Energy-efficient network protocols and resilient data transmission schemes for wireless sensor Networks—An experimental survey. Energies, 15(23) doi:10.3390/en15238883