A Framework for Producing Effective and Efficient Secure Code through Malware Analysis

Abhishek Kumar Pandey[1], Ashutosh Tripathi[2]
Alka Agrawal[4], Rajeev Kumar[5*], Raees Ahmad Khan[6]
Department of Information Technology
BBA University, Lucknow-UP, India

Mamdouh Alenezi[3]
College of Computer and Information Sciences
Prince Sultan University
KSA

***Abstract*—Malware attacks are creating huge inconveniences for organizations and security experts. Due to insecure web applications, small businesses and personal systems are the most vulnerable targets of malware attacks. In the wake of this burgeoning cyber security breach, this article propositions a framework for a complete malware analysis process including dynamic analysis, static analysis, and reverse engineering process. Further, the article provides an approach of malicious code identification, mitigation, and management through a hybrid process of malware analysis, priority-based vulnerability mitigation process and various source code management approaches. The framework delivers a combined package of identification, mitigation and management that simplifies the process of malicious code handling. The proposed framework also gives a solution for reused codes in software industry. Successful implementation of the framework will make the code more robust in the face of unexpected behavior and deliver a revolutionary stage wise process for malicious code handling in software industry.**

***Keywords*—Malware analysis; reuse code; framework; static analysis; dynamic analysis; reverse engineering; manual analysis**

I. Introduction

The present cyberspace is imploding with attacks and breaches. Easy access to internet and quality less security mechanism has created much unusual and dangerous vulnerability in the current digital world. Malware is the biggest threat to the cyber world in a current situation [1]. Malware is the software that has some malicious or harmful set of operation or instructions in their source code for performing a malicious activity in a system or network [2]. Malicious software's have hidden malicious features. With name or structure, they are like normal useful software but after execution they perform harmful activities on the system. Millions of computer users are targeted by more than thousands of different malware daily. According to a study [3], in every 39 seconds, a malware attack is executed in the world.

The personal and professional tasks of today's digital generation are now software based and any software is made with source codes. Instructions and operations written by a coder into a particular language for execution on the computer are called source code [4]. Enormously growing speed of software industry is daily producing more than a hundred of new software for the users. Unfortunately, the malware creators take advantage of this huge population of software. Malware creators facilitate their malicious software with genuine software for more user accessibility. Every malicious code has some harmful features but they also have some good codes and the purpose of this framework is to provide good codes from malicious code for reuse in the industry with malware analysis. The growing and expansive rate of software industry creates the need to reuse codes for coders with some improvements instead of writing a new one.

The culture of reuse code is growing very fast in the software industry because reusing the codes reduces the efforts and, more essentially, saves on the time invested in the project. The time that a coder spends on a project is very valuable and if reusing of code reduces that valuable time, it is a great option for programmers. While working on malicious codes, it is very important to understand the harmful malicious activity of code for effective mitigation and that is the reason behind using malware analysis in the identification process.

The first segment of this article discusses the significance of reuse code in the business; the second segment characterizes the need of malware investigation. In third segment, the authors characterize the system for extricating secure great codes with malware examination and besides this portray the need and criticalness of the structure. After this clarification, in the last segment, the authors posit the conclusion and enunciate efforts directed towards future work.

II. Previous Research Initiatives

Authors of the proposed study find that many researchers provides the research article on malware analysis and portray various different type of frameworks for enhancing the malware analysis approach. In order to deliver the proposed framework authors find the following previous research initiatives.

Belal Amro provides a malware analysis technique for mobile devices that gives an analysis study of various malware analysis approaches on operating systems like android and IoS [13]. The paper focuses on frequently used phone set vulnerabilities and tries to assess their possible solution through malware analysis.

S. Chuprat et al. provides a framework and its implementation in big data environment. The proposed framework in this paper delivers an approach that analyzes and predicts the future threat of malware attack in big data

*Corresponding Author.

platform [14]. Authors of this study also provide some data in order to validate their results and framework workflow.

G. Hamsa et al. provides an analysis of various malware identification techniques and assess their pros and cons on different standards [15]. The exhausting review of paper on malware techniques provides a path for future researchers of malware and malware analysis.

The authors of this proposed study finds that there is lack of literature which is discussing the whole malicious handling approach under one roof. Many researchers provide various effective and novel approaches in order to enhance the identification and detection of malwares through malware analysis. But it is also evident that there is very less amount of literature is available that is discussing about the malicious code vulnerability identification, mitigation and management. Proposed framework will help the industry and future researchers in order to produce some useful codes through a hybrid approach associating malware analysis.

III. Importance of Reused Codes

Software industry is growing voluminously. Coders code new logics and functions every day but a new code takes too much time and efforts to be coded. Every coder faces some challenges like understanding user requirements, time of completing a project and so on [5]. Reuse of existing code eases the coders' tasks in multiple ways. Reuse of code gives a key to the coder for easily understanding the needs of client. Thus, the time of completing the project is much less when compared to the time invested in writing new codes. Embedded system development has secured an important place in the software industry in the last decades and average time duration of completing an embedded system project is a minimum of 12-14 months [6]. The phrase 'time is money', is indeed most apt for the software industry. Any product line is worthwhile only if satiates the end user's needs in a given timeframe. The immensely competitive pace at which the companies churn out products in the software arena must meet the time targets. This necessitates the reuse of code in software development. Automated Program Repair (APR) approaches also open a door and create the demand of reusable codes for creating patches and findings bugs. The basic work process of APR's are totally depends on reusable codes [16]. This type of scenario also refers to the need of effective framework that produces some useful codes from malicious vulnerable codes.

IV. Why Malware Analysis?

Malware are increasing at an alarming rate for several reasons. These reasons create many challenges and issues for the cyber expert. According to the study of Forbs Magazine, 25% of Malware target the financial information of users [9]. The study also shows that the number of hacked account a hacker has, this makes it easy for hackers to exploit. Malware analysis helps in objective identification of malware or malicious code. There are three main techniques of malware analysis (i) Static Analysis (ii) Dynamic Analysis (iii) Reverse Engineering.

Many researchers have proposed their ideas on malware analysis methodologies. Yuhei Kawakoya et al. shows the methodology of malware analysis with the help of sandboxing and API calls analysis. The paper tells the process of taint assisted malware analysis and enhances the malware identification steps [7]. Kamla Kant Sethi et al. gives a framework of malware analysis for classifying the malware with identification of the malware by using Sandboxing Tools and Machine level learning tools for extracting exact information about malicious software [8]. Li Li et al. portray a systematic review on the need for static analysis in malware detection but the methodology that is described in the paper uses the automated tools for static code analysis of malware. The methodology uses the call graph analysis technique to examine the calls, variables, and classes of code and other significant attributes of a source code [11]. Christian Camilo et al. shows the significance of machine level approach in their paper and focuses the whole analysis process of malware on machine learning for better results [12].

These research studies are based on enhancing the malware analysis process for better results. However, there is a need for mitigation of malicious codes also and further research initiatives must pivot on this. Every researcher needs to focus on the useful codes written with malicious codes for helping the software industry by providing codes for reuse as well as identifying malware and mitigating them.

V. Framework

Malware analysis deals with the study of how malware functions and about the possible outcomes of infection given by a specific malware. When an attacker writes a malicious application code, he also uses or writes some good code for hiding the malicious activity of that application and also for increasing the user acceptability of application. This is akin to steganography. The objective of this framework is to extract or retrieve the good code from malicious code for reuse. The Framework is a full package of identification, mitigation and managing the code by combining malware analysis for extracting useful codes. The authors have classified this framework into three phases (1) Monitoring (2) Mitigating and (3) Managing. A brief explanation of these three phases is enumerated below:

A. Phase 1: Monitoring Phase

Objective of this phase is to understand the purpose, functionality and structure as well as the vulnerabilities of the malware for extracting good codes and easy mitigation and management. Monitoring phase is a combination of all three methodologies of malware analysis (static analysis, dynamic analysis and reverse engineering). The developer uses automated analyzers in the monitoring phase for detecting and examining the malware easily and this is done in considerably less time. The authors categorize the monitoring phase into three sub-phases that are shown in Fig. 1 and described as follows:

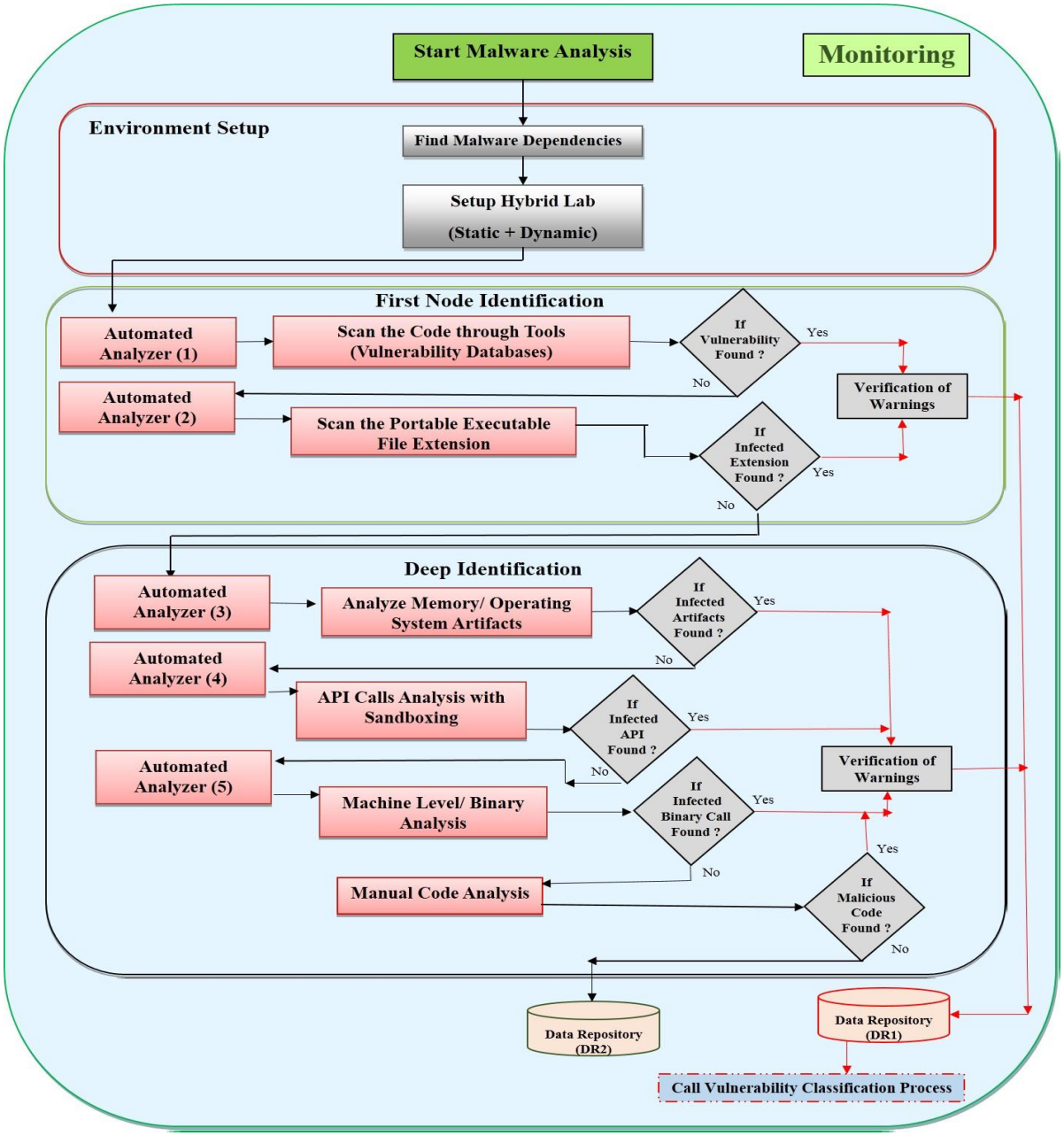

Fig. 1. Monitoring Phase.

1) Environment setup: First sub-phase of the monitoring phase is environment setup. In this sub- phase, the authors set up an environment for executing the analysis process. Dynamic analysis of malware is always done under some restricted environment for a better and secure outcome. The dynamic analysis deals with malware at motion. The following are the processes that an examiner takes while setting the analysis environment.

- Find Malware Dependencies: It is very important in dynamic analysis to run all features and services of malware for understanding the objective and finding the vulnerabilities clearly. So it is important to find malware dependencies and install them in the lab to perform dynamic analysis process.
- Setup Hybrid Lab (Static + Dynamic): After finding dependencies, set up a hybrid lab which is a mixture of the static and dynamic lab for further analysis. The static analysis deals with malware at rest, it means in this process malware is not executed on the system. Static analysis is fully secure and harmless examination process of malware, but dynamic analysis deals with malware at motion. In dynamic analysis, malware is executed on the system under a controlled environment. Reverse engineering is complementary for dynamic analysis.

2) First node identification: Second sub-phase of the monitoring phase is first node identification. This phase deals with some common methods to find out malware vulnerabilities. This phase uses signature-based identification methods for recognizing old malware classes. Steps that are taken in this sub-phase are enlisted below:

- Scan the Code through Tools (Vulnerability Databases): In this step we scan the code through various old vulnerability database (by tools) for finding the match. If the vulnerability is found then we verify the warning manually and if manual verification is also found to be yes, then we save that vulnerability into Data Repository (DR1) and, if not, then we go for the next step which is scanning the portable executable file extension.
- Scan Portable Executable File Extension: In this step, we scan portable executable file extension by various tools for finding the infected extension and if the infected extension is found by a tool, we verify the warning manually. If a warning is yes, then we save the vulnerability into DR1 otherwise we go to the next sub-phase which is Deep Identification.

3) Deep identification: Third sub-phase of monitoring phase is deep identification. In this phase, the examiner analyzes the malware by various industry level professional methods and finds the vulnerabilities. Steps that are taken in this sub-phase are:

- Analyze Memory/Operating System Artifacts: In this step, experts analyze memory/operating system artifacts both manually and by the tools. If something is detected, the examiner verifies the warning first. If the warning is yes, he saves that vulnerability into DR1 and if it is no, then the expert proceeds to the next step which is API Calls Analysis with Sandboxing.
- API Calls Analysis with Sandboxing: In this step, the expert uses sandboxing tools and with the help of that tool the examiner analyzes API calls for malicious API's. If the tool finds any malicious API then it blinks the warning and after that the expert verifies the warning. If the warning is true then the expert saves it to DR1. If false, then the next step begins which is Machine Level/Binary Analysis.
- Machine Level/Binary Analysis: After using all static and dynamic method in the last automated analysis, examiner uses reverse engineering method for finding vulnerabilities in the code. In this step, the expert uses reverse engineering malware analysis tools for finding the malicious binary calls. If the tool shows the warning, the developer verifies that warning with an expert. If a warning is yes then the expert saves that vulnerability into DR1 repository. If not, then he goes for next step which is the Manual Code analysis.
- Manual Code Analysis: In this step, the examiner analyzes the malware code and finds vulnerabilities and malicious piece of code manually. If the analyst finds malicious code or vulnerability, he calls for superior checking (verify warning) and if the senior coding expert will verify the warning to be true, then the analyst saves that vulnerability in DR1. Should it be false, then he saves this code into a new repository called the Data Repository DR2 as a vulnerability-free code.

B. Phase 2: Mitigation Phase

This phase is to mitigate the detected/identified vulnerabilities in the codes. The step-by-step process of mitigation of vulnerabilities is depicted and elucidated in Fig. 2:

1) Vulnerability classification: In this step, the analyst classifies the vulnerabilities that are discovered in previous phase. Afterwards, the analyzer goes for the next step which is measuring the Priority of vulnerabilities (Quantitatively).

2) Measuring the priority of vulnerabilities: In this step, the examiner evaluates the priority of vulnerability, quantitatively and mitigates these vulnerabilities according to their severity level. If severity level of the vulnerability is high then the analyst removes it. If the severity level of the vulnerability is medium, then the analyst repairs the codes. If the severity level of the vulnerability is low, then the analyst tries to fix the issue. After mitigating the vulnerability issues in the codes, the examiner saves the code in Data Repository (DR3) and calls for the managing phase.

C. Phase 3: Managing Phase

Objective of this phase is to manage mitigated code (DR3) and vulnerability-free code (DR2) for future reuse. Management of extracted code is very necessary because while a coder uses an old code in reuse, it is always a challenge for the programmer to manage that code5. This phase will help the coders in the industry by reducing their work (Managing Code) slightly. The authors categorize the monitoring phase into three sub-phases that are shown in Fig. 3 and are described as follows:

1) Maintaining the codes (As per requirement specification): In this step, examiner follows recent trending process of software development industry which is also called managing code. With the help of a good coder, an examiner manages the codes from (DR2) & (DR3) and after successful management of code, the expert saves the managed code into Data Repository (DR4).

2) Verification of the functionality: After successful management of code, the expert verifies the functionality of the code. If the analyst finds any functionality issue, then the analyst directly calls for maintenance of the coding process, and if no issue is found then the examiner goes for next step which is Measuring the Complexity of Design.

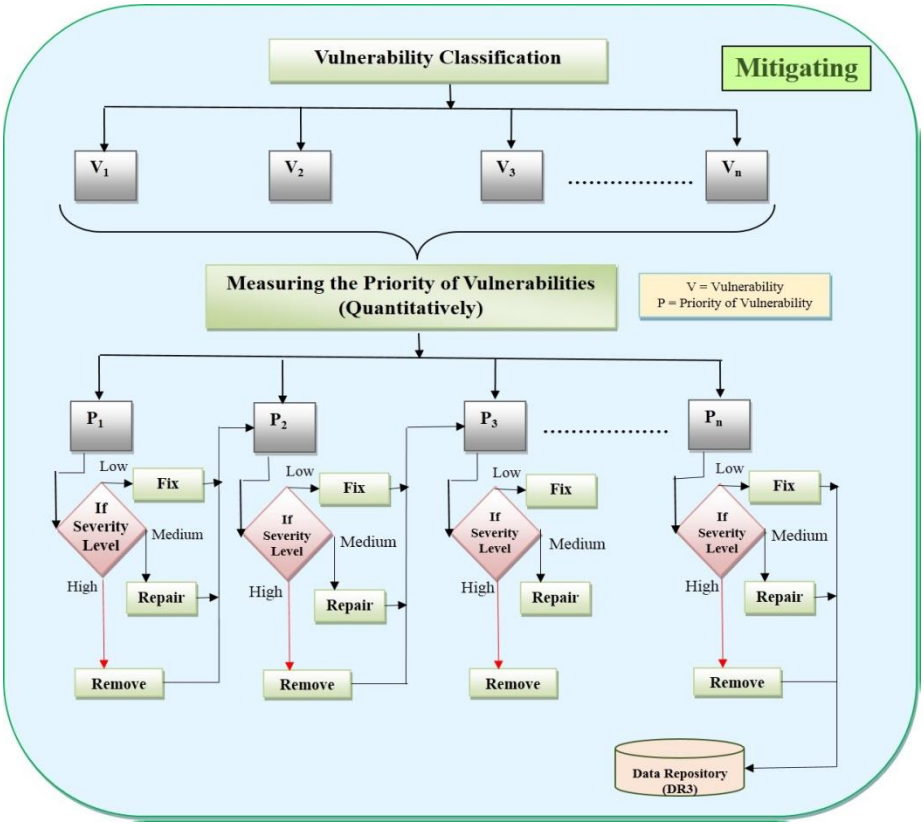

Fig. 2. Mitigating Phase.

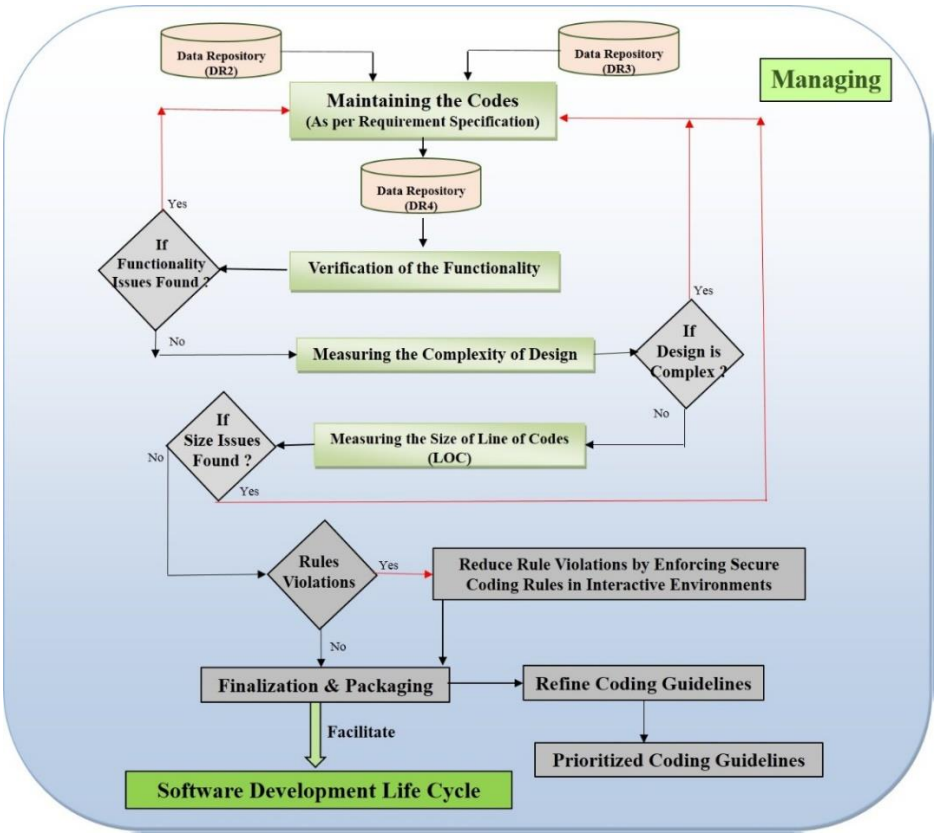

Fig. 3. Managing Phase.

3) Measuring the complexity of design: In this step, the analyst assesses the complexity of the design of code and if the examiner finds any issue in the complexity of design, then the examiner directly calls for maintaining the coding process. Otherwise the next phase is followed which is measuring the Line of Codes (LOC).

4) Measuring the size of Line of Codes (LOC): In this step, the analyst measures the line of codes for assessing the size of code. If the examiner finds any size issue in the code, the expert directly calls for maintaining the coding process and if no issue is found then the expert goes for next step.

5) Rule violation: In this step, the expert checks the rule violation of code, if the result is yes, and rules are violated more than acceptance, the expert reduces rule violation by enforcing the Secure Coding Rules in interactive environment. After this process, the examiner goes for Finalization & Packaging step and if minimum rules are violated, they are acceptable. So, the examiner goes for the next step called-Finalization & Packaging.

6) Finalization and packaging: In this step, the expert finalizes the code and prepares it for use by facilitating it with software development life cycle. This process helps the industry developers in their projects by providing ready to use managed codes.

7) Refine coding guidelines: This step is for coders who are interested in writing secure codes. In this step, the examiner provides the guidelines for a coder for writing secure code after analyzing the full malicious code.

8) Prioritize the guidelines: This step will help the coders to understand the provided guidelines easily by arranging the guidelines according to their priority or need in programming.

VI. Significance of the Framework

Signature-based identification of malware was very useful and effective for last 10 years but in the current scenario, Corrado Aaron Visaggio and his team from Italy developed an engine that alters and modifies the malware code automatically and misinforms the signature-based analyzers [10]. The engine works on the shape of the malicious code, not on the behavior of the code. This sort of improvement creates the need for a full bundle with the blend of each of the three investigation forms and furthermore needs to take a shot at the code for malware analysis. The framework is providing all the necessary requirements that are needed in the current situation of malware analysis and software industries.

The framework is focusing on the clear and perfect vulnerability identification mechanism with the help of malware analysis techniques. For mitigating these vulnerabilities, the framework uses prioritization and severity assessment methods. After mitigation comes managing and for this the secure code framework is produced which manages the code. Thereafter, an expert programmer then assesses the complexity, reliability and size of code for easy reusability. After all these steps, the framework provides the guidelines for future developers and facilitates the produced code into the software development life cycle for further uses. If we look at this framework deeply, it is a full bundle supply of ready to use codes. The framework provides the following features for developers and researchers.

- The framework provides ready to use, a maintained code for developers for their existing projects, if the code is compatible with their project.
- The framework gives well-structured and accurate malware analysis procedure for finding code vulnerabilities.
- The framework is able to identify the malicious codes and mitigate these vulnerabilities. Furthermore, it produces secure code for industry reuse.
- The framework provides the procedure for providing secure reused codes with the help of three-phase framework and creates an easy approach for the coder to reuse code.
- The framework also provides the time feasible method for identification, mitigation and managing the malicious code.

VII. Conclusion and Future Work

Malware coders are attempting to increase their area of infection and impact of harm very massively. Evidently, security mechanism of web is penetrated on a daily basis with huge number of malware attacks occurring every day. Advancement of malicious codes on a daily basis is creating big gap in old identification and examination methodologies for malware. Besides this, a large number of software is also creating the challenge for coder in development of new logics and functions every day. This kind of challenge has increased the significance of reuse codes in the industry. The framework is shown in Fig. 4. It maps the phase-wise steps to produce secure codes from malicious code with the help of malware analysis. The framework will help in identification of malware and then mitigating the malicious vulnerabilities, moreover managing, mitigating, securing, and producing no vulnerable code for industry reuse. Successful implementation gives the direction for future analysis and suggests the guidelines for coders. The implementation of the intended framework will help the researchers to develop a useful and reliable strategy for producing or writing secure codes for future work on this proposition.

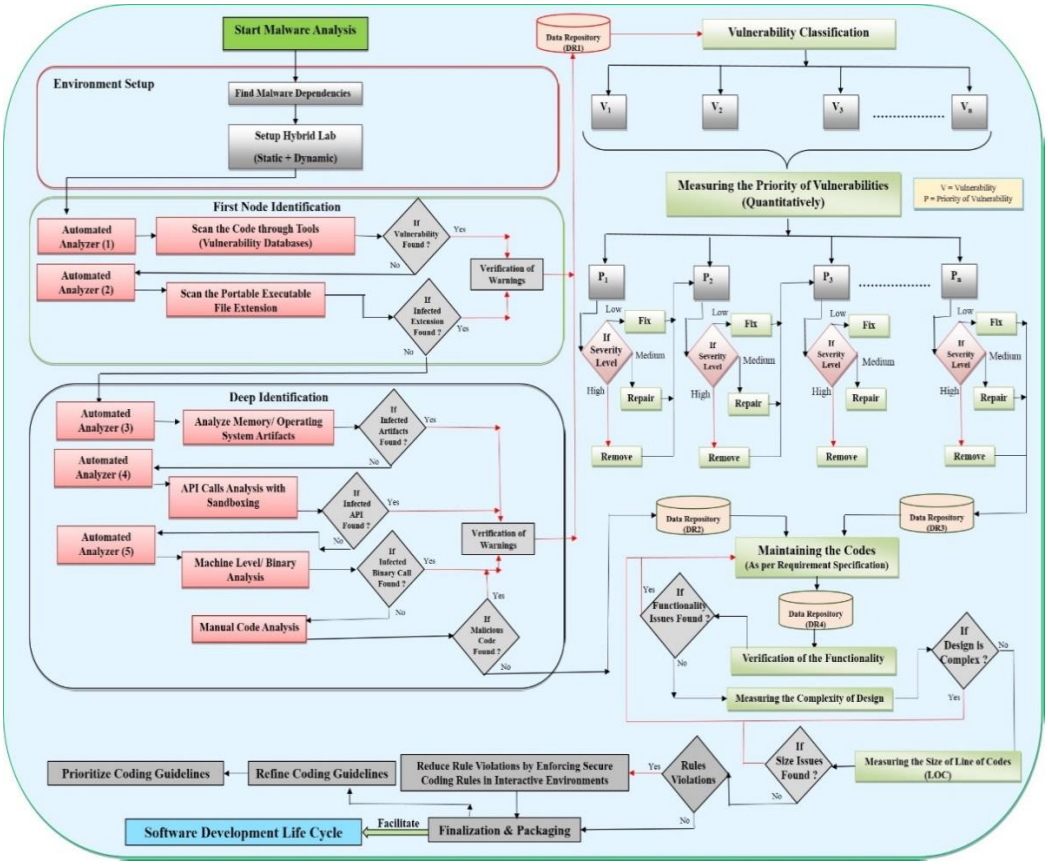

Fig. 4. Framework for Producing Secure Code through Malware Analysis.

ACKNOWLEDGMENT

Authors are thankful to the College of Computer and Information Sciences, Prince Sultan University for funding this research study.

REFERENCES

[1] Pandey, A. K., Tripathi, A. K., Kapil, G., Singh, V., Khan, M. W., Agrawal, A., Kumar, R., & Khan, R. A. Trends in Malware Attacks: Identification and Mitigation Strategies. In M. Husain, & M. Khan (Eds.), Critical Concepts, Standards, and Techniques in Cyber Forensics (pp. 47-60). Hershey, PA: IGI Global, 2020.

[2] All about malware, Available at: https://www.malwarebytes.com/malware/.

[3] CyberSecurity Statics and Facts For 2017- 2018, Available at: https://privacy.net/cybersecurity-statistics/.

[4] Source Code, Available at: https://www.techopedia.com/definition/547/source-code.

[5] The Challenges of Code Reuse (How to Reuse Code Effectively), Available at: https://www.perforce.com/blog/qac/challenge-code-reuse-and-how-reuse-code-effectively.

[6] Why Code Reuse Matters, Available at: https://www.apress.com/de/blog/all-blog-posts/why-code-reuse-matters/15477476.

[7] YuheiKawakoya, EitaroShioji, Makoto Iwamura, Jun Miyoshi. Taint-Assisted Sandboxing for Evasive Malware Analysis. Journal of Information Processing; Vol.27 297-314, 2019

[8] Kamlakant Sethi, Shankar Kumar Chaudhary, Bata Krishna Tripathy, Padmalochan Bera. A Novel Malware Analysis Framework for Malware Detection and Classification using Machine Learning Approach. 19th International Conference on Distributed Computing and Networking, Varanasi. 2018.

[9] Cybercrime: 25% Of All Malware Targets Financial Services, Credit Card Fraud Up 200%, Available at: https://www.forbes.com/sites/zakdoffman/2019/04/29/new-cyber-report-25-of-all-malware-hits-financial-services-card-fraud-up-200/#7f4932eb7a47.

[10] A Group of the researchers from the Iswatlab team at the University of Sannio demonstrated how is easy to create new malware that eludes antimalware, Available at: https://securityaffairs.co/wordpress/51714/malware/evading-antimalware.html

[11] Li Li, Tegawende F. Bissyande, Mike Papadakis, Siegfried Rasthofer, Alexandre Bartel, Damien Octeau, Jacques Klein, Yves Le Traon. Static Analysis of Android Apps: A Systematic Literature Review. Journal of Information and Software Technology Elsevier; Vol. 88 pp 67-95, 2017.

[12] Christian CamiloUrcuqui Lopez, Andres Navarro. Framework for Malware Analysis in Android, Sistemas&Telemática, 14(37), 45-56, 2016.

[13] Belal Amro. Malware Detection Techniques For Mobile Devices, International Journal of Mobile Network Communications & Telematics, Vol.7, No.4/5/6, 2017.

[14] Suriayati Chuprat, Aswami Ariffin, Shamsul Sahibuddin, Mohd Naz'ri Mahrin, Firham M. Senan, Noor Azurati Ahmad, Ganthan Narayana, Pritheega Magalingam, Syahid Anuar, Mohd Zabri Talib. Malware Forensic Analytics Framework Using Big Data Platform, Springer Nature Switzerland, 881, pp. 261–274, 2019.

[15] G. Hamsa, Deepti Vidyarthi. Study And Analysis Of Various Approaches For Malware Detection And Identification, Vol 1, Issue-10, 2013

[16] Qi Xin and Steven P Reiss. "Better Code Search and Reuse for Better Program Repair". In: Proceedings of the 6th IEEE/ACM International Conference on Genetic Improvement. 2019.