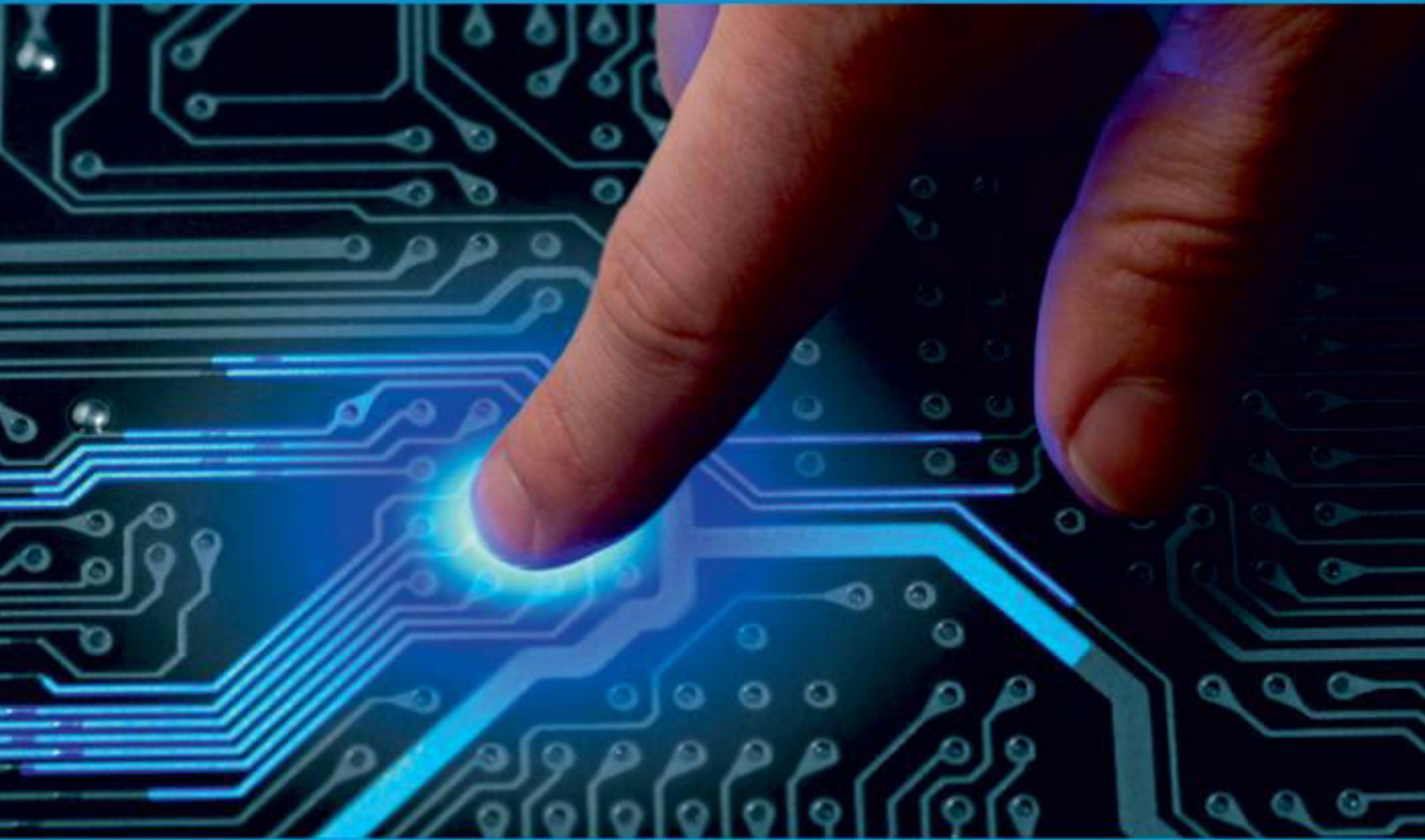
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 2, February 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1202053 |

Cypher Shield: A Simple Approach to Foil Reverse Engineering

N Nithish, Murugan R

Student of MCA, Dept. of CS & IT, Jain (Deemed-to-be-University), Bengaluru, India

Professor, Dept. of CS & IT, Jain (Deemed-to-be-University), Bengaluru, India

ABSTRACT: In this paper, we introduce Cypher Shield, a user-friendly approach to protect our malware from the prying eyes of reverse engineers. By integrating cypher code execution techniques, we cloak crucial parts of the code in encrypted layers, creating a puzzle that's tough to crack during static analysis. We examine the technical details and practical considerations, examining how Cypher Shield successfully shields our malware. Through real-world case studies, we show that your software can execute seamlessly without falling victim to reverse engineering attempts. These findings highlight Cypher Shield as an accessible and effective solution, empowering developers to bolster the security of their software applications against the persistent threat of reverse engineering. We will perform it in real live example by encrypting with a marshal algorithm for the malware.

I. INTRODUCTION

In today's fast-paced world of ever-evolving cybersecurity threats. Malicious software, or malware, stands as a formidable adversary, posing a direct threat to the fundamental core of computer systems across the globe. Cybercriminals persistently engage in the art of decoding and manipulating malware, exploiting vulnerabilities, understanding detection mechanisms, and devising elusive variations that effortlessly slip past traditional security barriers. In this dynamic digital arena, Cypher Shield emerges as a reliable defender, offering a solution that is not only effective but also user-friendly, specifically tailored to fortify malware against the persistent threats of reverse engineering.

By intricately weaving cypher code execution techniques into its framework, Cypher Shield transforms critical sections of malware code into encrypted puzzles, skillfully resisting attempts at deciphering during static analysis and extending the duration of reverse engineering. This ingenious strategy disrupts the typical playbook of reverse engineers, introducing a layer of complexity that significantly hampers their progress.

As we embark on this exploration, we will delve into the intricacies of Cypher Shield, unpacking its technical nuances, practical applications, and the potential impact it can have on enhancing malware resilience. Through the adoption of Cypher Shield, developers can empower their malware with a dynamic defence mechanism – a shield not only protects their creations but also challenges the established norms in the perpetual dance of cybersecurity cat-and-mouse. Brace yourself for a journey into a new realm of safeguarding your digital creations with the formidable Cypher Shield.

II. MALWARE ANALYSIS

The process of understanding the behaviour of malware is called malware analysis by this we can protect as well as eliminate the malware now we will see the classification of malware analysis

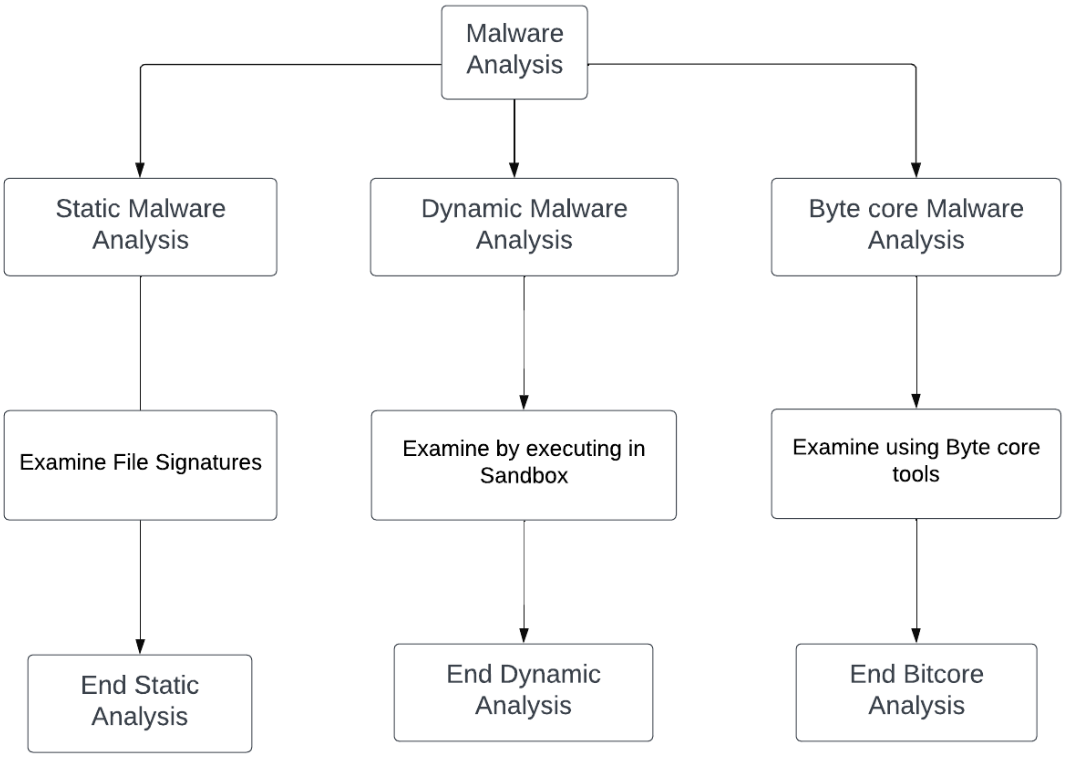


Figure 1. Classification of Malware Analysis

Static Malware Analysis:

we will analyze the malware through the file signature of the malware. Now we will see exactly how it is going to work in real life. At first, when we come to know that a particular file has a file signature over it we will extract the file signature of the malware file and store it in the database and whenever we use basic antivirus software it scans the entire file signatures of that particular system and compare with existing malware file signatures of that system by that process it will declare as malware if normal file signature is matched with malware file signature but we will never execute the malware over here. These are the main aspects of the Static Malware Analysis

- Determine the file type either Windows based or Linux-based.
- Determine through crypto hashes like md5, sha256
- String search
- Determine the fuzzy hash
- Submitting online antivirus
- Inspecting File Dependencies

These are the main points that will play a crucial role in static malware analysis

Dynamic Malware Analysis:

Instead of comparing file signatures over here, we will directly execute the malware in a safe and secure environment like a sandbox. Now we will see in detail how this process will work over here. We will create a safe environment in the docker with minimum support of OS and basic requirements to execute the file over here by the file execution process we will come to know whether the file is normal or malware. We can see the functionality of the source code in such a way that at what point it started to do malicious activity in the system and we can even see the network monitoring in the system like TCP/UDP connection over the system with traffic details and everything will be given as detailed in the graph. Now we will see the basic elements of Dynamic Malware analysis.

- Monitoring the process
- Monitoring the file activities
- Monitoring the network
- System call Tracking

Byte core Malware Analysis:

Instead of executing the malware, we will see the entire source code of the malware file so that we will understand every module of the file of source code and we will come to the nature of malware. We will see clearly how this going to work. At first, we will take the exact package of the file and we will use tools like Integer analysis and upload the file over it so it will show the exact source code of the malware this process is also called Reverse Engineering we can also do one more step to make an efficient way to analysis ie.., Memory analysis in this we will de-assemble the file in the assemble level so that we will file exact spot of the memory block of the file where it tries to do malicious activity over the system. For the particular block of memory address, we will copy it in a Hexadecimal and we will store that particular file is called as Yara Rule. So this is called Byte core Malware Analysis

- Running Process
- Network Communication
- Loaded Modules like Library in code
- Are there any code injections
- API Cookies

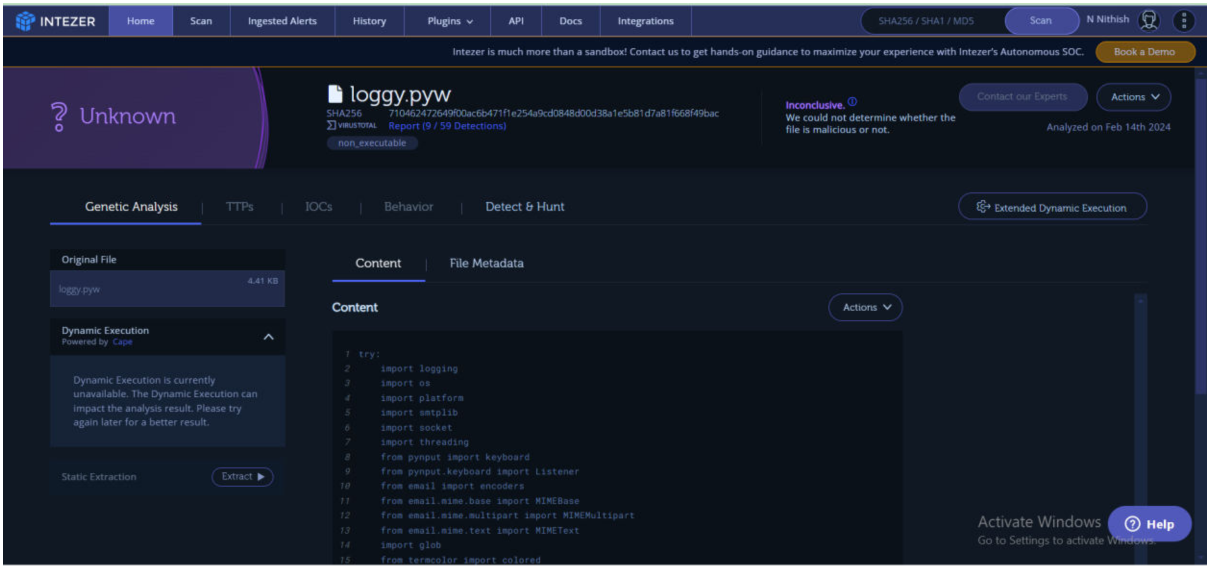
Now we know that working of Byte Core malware analysis but our main focus of the paper is Avoiding reverse engineering through encryption of the code. For that, we will use the Marshal Algorithm and RSA Algorithm

We can think that there are so many algorithms over there in the current world but why Marshal algorithm and RSA Algorithm

The main unique feature of the Marshal Algorithm is code obfuscation means that we will convert the source code of the file into hexadecimal so that the code will be converted into assembly level it can be executed in the normal system but the plain text will remain over here so for that we will use RSA with salting to make it more secure when we perform reverse engineering over the application it will not show the source code so that it will be hard to break the malware through the reverse engineering process.

Now we will see exactly how this process is going to work out for Malware by performing the reverse engineering for the keylogger encrypted with the Marshal Algorithm.

At first, we will execute the code without encryption and will see how it's going to work the file name is logged.pyw and it is a Keylogger malware that was specially built for Windows OS to capture the keystrokes from the victim PC. We are going to do reverse engineering for that malware using Integer analysis to decode the malware



Intezer is much more than a sandbox! Contact us to get hands-on guidance to maximize your experience with Intezer's Autonomous SOC.

loggy.pyw
SHA256 710462472649f00ac6b471f1e254a9cd0848d00d38a1e5b81d7a81f668f49bac
VIRUSTOTAL Report (9 / 59 Detections)
non_executable

Inconclusive.
We could not determine whether the file is malicious or not.

Analyzed on Feb 14th 2024

loggy.pyw 4.41 KB

Dynamic Execution
Powered by Cape

Dynamic Execution is currently unavailable. The Dynamic Execution can impact the analysis result. Please try again later for a better result.

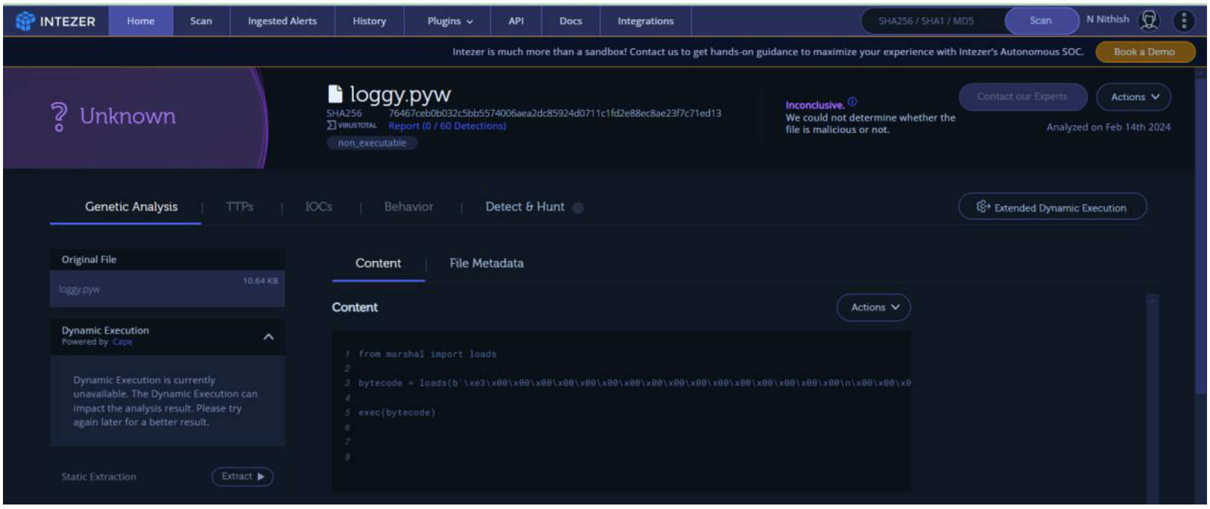
Content

```
try:
    import logging
    import os
    import platform
    import smtplib
    import socket
    import threading
    from pynput import keyboard
    from pynput.keyboard import Listener
    from email import encoders
    from email.mime.base import MIMEBase
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    import glob
    from termcolor import colored
```

Figure 2. File Execution without cyphering it

After executing the code without encryption, we can see here that the program how the malware was built with every specific detail of the loggy.pyw and we can see the actual code he required to develop the malware and we can even understand which programming language is used to create the malware so that we can see what is the logic behind the malware and its motive of the malware

Now we will use the Marshal Algorithm and RSA Algorithm for that loggy.pyw file so we can see how that code was converted into cypher text of hexadecimal code which can be easily understood by the machines and not by humans.
In the code actually, we converted the functionality of the code to hexadecimal but they still text will exist for that we are going to use the RSA algorithm with a public key as well as a private key so that our code will exist as a cypher when we perform the reverse engineering.



Intezer is much more than a sandbox! Contact us to get hands-on guidance to maximize your experience with Intezer's Autonomous SOC.

loggy.pyw

SHA256 76467ceb0b032c5bb5574006aea2dc85924d0711c1fd2e88ec8ae23f7c71ed13

VIRUSTOTAL Report (0 / 60 Detections)

non_executable

Inconclusive. We could not determine whether the file is malicious or not.

Analyzed on Feb 14th 2024

Original File: loggy.pyw 10.64 KB

Dynamic Execution Powered by Cape

Dynamic Execution is currently unavailable. The Dynamic Execution can impact the analysis result. Please try again later for a better result.

Static Extraction

Content

```
from marshal import loads

bytecode = loads(b'\xe3\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\n\x00\x00\x0

exec(bytecode)
```

Figure 3. File Execution with Cypher Code

By this, we can avoid reverse engineering for malware by making the source code into cypher code in an executable way so that the code will be executed in the victim system at the assembly level of the code.

III. RESULT

Delving further into malware analysis, we explored diverse classifications, encompassing static analysis, dynamic analysis, and byte core analysis. Recognizing the paramount importance of mitigating reverse engineering risks, we introduced the concept of code obfuscation. This involves transforming the source code into a hexadecimal format using the Marshal Algorithm, a strategic manoeuvre to deter prying eyes from understanding the code during reverse engineering attempts. As well as we use the RSA algorithm to convert the basic text to cypher so that it will be hard to do reverse engineering for that malware

A practical illustration using a keylogger malware (loggy.pyw) demonstrated the impact of employing the Marshal Algorithm and RSA Algorithm. Through the encryption of the source code into cypher code, we showcased the proactive measure of preventing reverse engineering, ensuring that the code executes seamlessly at the assembly level without divulging its underlying logic.

IV. CONCLUSION

In conclusion, the synergistic integration of Cypher Shield and the Marshal Algorithm forms a robust defence against reverse engineering, offering a pragmatic and effective solution to fortify digital creations amid the ever-evolving landscape of cybersecurity threats
A hands-on example featuring a keylogger malware (loggy.pyw) showcased the impact of deploying the Marshal Algorithm and RSA Algorithm. By encrypting the source code into cypher code, we demonstrated a smart move against reverse engineering. The code executed smoothly at the assembly level, keeping its secrets safe.
In a nutshell, Cypher Shield, coupled with strategic analysis and the craftiness of the Marshal Algorithm, provides a sturdy defence against reverse engineering. It's not just about security; it's about giving developers the upper hand in the constantly evolving world of cybersecurity threats. As developers continue their dance with cyber threats, these tools emerge as stalwart guardians, rewriting the rules and ensuring our software stays resilient against adversarial endeavours.

REFERENCES

1. Sikorski, M., Honig, A. (2012). Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. United States: No Starch Press.
2. Blazy, O., Fouque, P.-A., Jacques, T., Lafourcade, P., Onete, C., & Robert, L. (2022). MARSHAL: Messaging with asynchronous ratchets and signatures for faster healing. Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing.
3. N. R. Sai, T. Cherukuri, S. B., K. R. and A. Y., "Encrypted Negative Password Identification Exploitation RSA Rule," 2021 6th International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2021, pp. 1-4, doi: 10.1109/ICICT50816.2021.9358713. keywords: {Data privacy;Databases;Media;Solids;Decoding;Cryptography;Password;Encrypted Negative positive identification;parallel key rule;hashed positive identification},
4. Intezer. (n.d.). Intezer Analyze. https://analyze.intezer.com/
5. J. Sabu, A. S, A. Gopan, G. S and S. Murali, "Advanced Keylogger with Keystroke Dynamics," 2023 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 2023, pp. 1598-1603, doi: 10.1109/ICICT57646.2023.10134044. keywords: {Computer hacking;Keystroke dynamics;Keyboards;Passwords;Credit cards;Software;Servers;Keyloggers;Keystroke Dynamics;Typing pattern},

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING