BUCHAREST UNIVERSITY OF ECONOMIC STUDIES

Faculty of Economic Cybernetics, Statistics and Informatics

Department of Economic Informatics and Cybernetics

APPLIED REVERSE ENGINEERING AND VULNERABILITY RESEARCH IN CYBERSECURITY

INGINERIE INVERSĂ ȘI ANALIZA VULNERABILITĂȚILOR APLICATE ÎN SECURITATEA CIBERNETICĂ

– PhD Thesis –

Summary

Ph.D. Candidate: ȘTEFAN SABIN NICULA

Scientific Supervisor: Prof. RĂZVAN DANIEL ZOTA, Ph.D.

ABSTRACT

This thesis delves into advanced cybersecurity methodologies with a focus on reverse engineering, vulnerability research, and exploitation techniques within Windows operating systems, particularly in the context of the Internet of Things (IoT). The study emphasizes the challenges of analyzing complex, closed-source software environments, with a special emphasis on Windows-based systems, which differ significantly from Unix-based environments in terms of kernel architecture and security mechanisms.

The thesis provides a comprehensive methodology that integrates reverse engineering, fuzzing, and malware analysis to identify and mitigate vulnerabilities in complex software environments. Key aspects include in-depth explorations of Windows kernel security, browser exploitation, and the application of fuzzing techniques to uncover security flaws. The work also proposes innovative solutions such as the use of honeypots for real-time exploit detection, automated debugging of browsers to discover zero-day vulnerabilities, and the integration of machine learning in malware analysis to streamline detection processes.

Windows Host Honeypot with Hypervisor Monitoring is a novel method for detecting exploits in the wild by setting up a Windows host honeypot, combined with hypervisor-based monitoring. Automated Browser Exploit Detection represents the solution that involves using automated debugging and memory sanitization tools to monitor for browser exploits. Automated Malware Analysis with Machine Learning to address the challenge of bulk malware analysis, this approach integrates machine learning with virtual kernel debugging. Reverse Engineering Techniques for Cybersecurity Applications is part of the research that provides a detailed guide on applying reverse engineering techniques to various cybersecurity challenges, including malware analysis, vulnerability research, and binary analysis.

These solutions contribute to the field by offering innovative approaches to proactive cybersecurity, particularly in environments where traditional methods may fall short. The research underscores the necessity of integrating advanced reverse engineering techniques with existing security frameworks to better protect against emerging threats.

KEYWORDS: REVERSE ENGINEERING, VULNERABILITY RESEARCH, EXPLOIT DEVELOPMENT, WINDOWS SECURITY, MALWARE ANALYSIS, FUZZING, BROWSER EXPLOITATION, INTERNET OF THINGS (IOT)

Table of contents

INTRODUCTION

The process of reverse engineering can be applied successfully in many different cybersecurity areas. It represents a workflow and a combination of techniques to understand closed-source software with minimal or no documentation beforehand. The expected results include clear information about the inner workings of software, the internal routines, and the purpose of the analyzed sample. The process includes disassembling a binary, decompilation, debugging, static and dynamic analysis. We often see it being used in actions such as malware analysis and vulnerability research.

The vulnerability research field is an ever-increasing topic that started to get a lot of traction and attention in the last decade. It can be comparable to the widely known bug-bounty programs where researchers are focusing their efforts on testing web applications, mobile applications, networks, or source code review. However, vulnerability research on binaries and desktop applications requires completely different approaches and a similar difference is present in the exploits and vulnerability classes. Still, the two domains have the same goal of discovering and reporting vulnerabilities in different software solutions.

Reverse engineering can be applied in many different areas as is not necessarily related to software on its own, but rather a concept of approaching a closed environment with the purpose of understanding it by constantly probing for results and observing output. As a general description, it could be defined as the deep-level understanding of a self-preserving technology that was intended to be kept private in the form of no documentation or minimal support in debugging the presented solution.

Similarly, in the last decade, we can clearly see an ever-ascending trend the Internet of Things (IoT) industry as more devices are reaching the market every year. The security maturity of the IoT field is increasing, however, judging by the number of publicly reported vulnerabilities affecting those solution, we can note a high chance of identifying easy-to-catch vulnerabilities even in the top vendors.

In terms of dedicated development for IoT software and hardware, the continuous research and development resulted in custom-made solutions intended for specific devices. However, on a

general level, most of the marketed solutions are based on similar operating systems, public libraries, and general software that is used across many industries and not necessarily developed for IoT-related programs.

The thesis tackles a variety of subjects related to *exploitation techniques, operating system internals, the process of reverse engineering, debugging, and tools used*, and includes dedicated chapters for each subject with practical examples. The main purpose of the research is focused exclusively and revolves around the Windows operating system with Intel x86 and x64 CPU support. As opposed to Unix-based environments, Microsoft has a different approach with regard to kernel development and the implementation of kernel-related mechanisms. Ultimately, the high-level concepts are the same as in the case of any other major operating system. My affinity for Windows-based software and the challenge of closed-source kernel components was one of the main motives for completing this research.

The **current level** of documentation and general knowledge for Windows-based exploits is scattered around in various sources. Exploiting development in software and high-value target still represents a learning process based solely on one's curiosity and self-learning drive. Moreover, we can see a big difference in public research, presentations, and trainings with support on Windows-based technologies as opposed to the Unix-based environments, especially significant in the IoT industry. This should not be corelated with a lack of resources of meaningful research on Windows side as in the last 5 years, many more papers have been published tackling this subject with technical-oriented research based on different exclusive scenarios and others that had a more general approach towards it. Being a highly specialized niche of software security represents a challenge when gathering materials and references for a solid workflow and foundation towards achieving a complex level of understanding the reverse engineering process of a highly complex solution such as an operating system kernel or a browser. Furthermore, the exploitation techniques and vulnerability discovery mechanisms are highly customized based on the attack vectors and the exploited environment. With the increasing implementation of protection and mitigation mechanisms, this raises challenges as some techniques are becoming obsolete while others suffer major improvements in order to bypass the protections.

THESIS OBJECTIVES

The **main objective** of the research thesis is to identify, define, test, and apply reverse

engineering concepts over a broad spectrum of low-level software testing with support on the Windows operating system on x86 and x64 architecture from a vulnerability research perspective. This can be achieved by understanding the attack surface, the offensive security approach of external actors when it comes to finding a 0-day vulnerability in complex solutions and the process of leveraging exploitation techniques in order to obtain a fully working exploit code. Another key point is evaluating the abuse and in the wild usage of such exploits and applying the identified techniques in order to research and create a relevant proposal that is able to prevent or detect malicious behavior on the victim software or machine. One final aspect of the research is providing examples and an applicable baseline research that can be used in order to proactively identify vulnerabilities and report them to vendors as to limit the exposure and increase the overall security posture.

The **necessity** of current research is determined by a missing link between various public writings and papers that span the industry, including conference presentations, business and technical blog posts, and academic papers. The aim of the research is to frame multiple contexts of reversing while applying the same baseline knowledge in terms of analysis techniques used, mindset when approaching the debugging perspective, and understanding the target software. Although many examples presented such as browser exploits, kernel attacks, and malware samples are making use of very different approaches and have their own internal knowledge, the process of understanding, debugging, and consolidating remains largely the same. Having knowledge of how to apply vulnerability research in a proactive way can help to elaborate defense techniques that can later mitigate future exploitation attempts or, in some cases, can fully deny a specific technique.

Looking at **the level of knowledge in the field** of 0-day vulnerability research, the depth of technicality and research is very deep; the majority of released exploits and research oftentimes are linked with a vast understanding of internals of such systems, at the point of kernel level of low-level processes including security protections that might conflict with the exploit development process. Each target represents a technical challenge on its own. Having a broad understanding of the overall process equips a researcher with the base knowledge to tackle each project confidently and can further aid in meaningful progress from both the offensive and defensive perspectives. In terms of hunting 0-day vulnerabilities in the wild, most solutions active in the market have proprietary systems, closed source, and have limited to no possibility of customization or the ability to add on-premises detection.

THESIS STRUCTURE AND METHODOLOGY

The thesis is divided into 10 main chapters as follows:

Chapter 1: Introduction is the first chapter presented in the thesis and opens the research topic with an overview of the structure and the research methodology used. The introduction covers general subjects regarding reverse engineering and the main scope of work. It also explains the current level of documentation in the field and the personal motivation for the research while showcasing its necessity. Moreover, a brief abstract is presented for each original solution presented in the thesis.

Chapter 2: Economic impact and analysis model continues with the initial research by taking a look at the economic impact of attacks conducted over the Internet on the overall software landscape. It describes topics such as malware campaigns such as ransomware attacks and the economic damage created, while also taking a look at different campaign starting points. Among common delivery mechanisms, such as social engineering attacks, we can also note the usage of zero-day or n-day exploits for publicly known vulnerabilities. There is also a direct correlation between freelance vulnerability research and public disclosure through various security programs. Moreover, there is an analysis of the direct economic damage between 2021 and 2022, and a considerable increase can be noted.

Chapter 3: IoT and vulnerability research explores the Internet of Things landscape and provides interpreted statistics from developers that include security risks, operating system usage, and trends. The chapter represents the link between the main theme of IoT in the business area and the scope of the research thesis which is focused on the security aspect of specific IoT developments with support for Windows. We identify that a portion of the IoT segment is using Windows Embedded (Windows IoT) as the main operating system, which shares many architectural similarities and internals with the main Windows NT kernel.

Chapter 4: Software vulnerabilities research expands on the idea of conducting research on different vulnerabilities affecting the operating system, binaries, and other complex solutions such as browsers. We list the most common security bugs and their associated particularities. The perspective of binary exploitation is compared with the current protection mechanisms available at the binary and operating system level. There is also a categorization based on the most effective attack and exploitation technique with an emphasis on bypassing the said

security controls. Each type of vulnerability is presented in detail, and, in contrast, the mitigation features implemented on different levels. More advanced methods of bypassing protection often require the usage of multiple security bugs that can be used in conjunction.

Chapter 5: Windows internals security is one of the main support chapters that details the Windows internals and architecture behind the most important routines and features encountered throughout the thesis research. The first key points presented are the mechanisms of elevating a Windows process, a feature that will be mentioned in the next chapters during the exploitation and post-exploitation phase. Windows internals security is a subchapter that follows the architecture of Windows kernel through various topics such as the Object Manager and Security Descriptors, process access tokens and integrity levels, and process execution properties. This research provides a baseline knowledge for the implementation and technical analysis during the practical evaluation of the thesis. Additionally, having prior knowledge of internal processes and routines helped in understanding how different security bugs occur in the Windows kernel and other complex software systems such as browsers.

Chapter 6: Windows virtualization security is a specific Windows internals topic described briefly in the previous chapter and analyzed in more detail. Virtualization-based security represents an edge in protection against various exploitation techniques targeted against the Windows operating system. It offers an additional layer of privilege and adds restriction methods that are harder to bypass as they can become very situational. We also explore the Windows kernel attack surface and take a look at how Windows drivers can be evaluated from a security perspective. The drivers are the link between user mode and kernel mode and many instances, especially in the IoT sphere where devices and sensors are embedded in the IoT solution for the business and functional logic.

Chapter 7: Malware analysis provides details on the malware analysis processes from a defensive perspective, leveraging the operating system knowledge to monitor the activity and successfully reverse engineer a given sample. Details about different evasion mechanisms are presented along with the possible solutions to bypass the protection mechanism that a malware might develop. The first two subsections deal with the static and dynamic analysis model with various practical examples that are most commonly encountered during malware analysis. Moreover, since the C/C++ programming language is one of the most commonly used in malware development, we take a look at useful techniques to facilitate an optimal reverse engineering session. Using the level of knowledge obtained from the previous chapters and the

malware analysis one, a solution based on hypervisor is presented for automatic detection of exploits in the wild. The system can also be used to debug proof of concepts, a technique that can provide valuable information that can help in creating generic detections.

Chapter 8: Fuzzing is a key component of the offensive security research field that can be used to automatically identify security bugs in the code of the analyzed software. There are various types of fuzzing processes and methods. In this chapter, we review some of the main methods of implementing fuzzing in cases where the researcher has access to the code or from a black-box approach. The results show that some of the key components of every fuzzing campaign are the instrumentation methods and corpus collection combined with an optimal harnessing mode.

Chapter 9: Browser exploitation shows a practical exploitation of a public CVE affecting the Chromium browser which is a use-after-free vulnerability in a sandboxed Chromium process. We take a look at the public-source code available and draw conclusions based on the internal debugging process in order to obtain read and write primitives by leveraging internal V8 routines and structures. Additionally, WebAssembly is used to forge a shellcode and place it in the heap memory using heap spraying techniques. Finally, the exploit is executed, and the results show that the Chromium process runs in a sandboxed environment, and an additional vulnerability located at the operating system level is required in order to create a bypass. The results enforce that the security of a system is closely connected by all of its system, and the underlying OS plays a major role in the security of a product.

Chapter 10: Conclusions is the final chapter that presents the conclusions of the thesis, all contributions resulting from the research, the dissemination of publication during the research time, and future work.

This thesis delves into an integrated security analysis framework, which combines reverse engineering, fuzzing, malware analysis, and vulnerability research. Focused on IoT systems using Windows operating systems, the study aims to provide background and methods for uncovering and minimizing vulnerabilities, improving the security posture against sophisticated cyber threats.

The increasing incorporation of IoT devices across various sectors demands increased security due to their complex interconnectivity and the critical nature of their applications. This paper underscores the importance of a holistic security approach by integrating reverse engineering

with traditional security practices to protect IoT systems against vulnerabilities and attacks, particularly in Windows environments. The following methodology structure was followed during the thesis research:

1. Conduct a literature review and obtain technical knowledge

- **IoT and Software Security Challenges:** Current issues and security considerations specific to IoT and software systems.
- **Windows vulnerability research and internals**: An overview of vulnerabilities inherent to Windows used in IoT contexts and details about relevant Windows internal security processes applied to the thesis subject.
- **Fuzzing techniques**: The role of fuzzing in proactive security testing and applied fuzzing campaigns.
- **Malware Analysis:** Examination of malware analysis techniques and the impact of malware attacks on Windows systems.
- **Reverse engineering**: Insights into how reverse engineering can aid in understanding and securing software, focusing on reverse engineering techniques and methods.

2. Research methodology

- **System setup**: Configuration of an experimental IoT environment incorporating Windows OS. Gathering technical knowledge about reverse engineering practices and evaluating operating system processes.
- **Reverse engineering**: Analyze software to identify security flaws in undocumented functionalities.
- **Vulnerability analysis**: Using specialized tools to perform reverse engineering in browsers.
- **Fuzzing Implementation:** Tailoring fuzzing tools to identify vulnerabilities within the operating system.

- **Malware Simulation and Analysis Techniques** Introducing malware to observe interactions with existing vulnerabilities and system defenses.

3. Vulnerability and reverse engineering research

- **Identification of vulnerabilities through reverse engineering**: Techniques to uncover or debug hidden vulnerabilities and security weaknesses using reverse engineering.

- **Impact assessment and mitigation strategies**: Evaluating the risks associated with identified vulnerabilities and proposing robust mitigation and detection strategies.

4. Results and discussion

- **Integration Benefits:** Evaluating how combining reverse engineering with fuzzing and malware analysis provides a deeper understanding of security challenges in the Windows environment.

- **Proposed solutions:** Presentation of original solutions that combine all the techniques discussed in the thesis.

- **Conclusions:** Concluding remarks summarizing critical findings and providing concrete recommendations for security improvement. Outline of suggested protocols and security measures based on the thesis analysis.

5. Future work

- **Advanced reverse engineering techniques**: Further development of reverse engineering methods to enhance detection and mitigation of sophisticated threats.

- **Cross-platform security analysis**: Expansion of the research framework to include IoT systems operating on different platforms.

ORIGINAL SOLUTIONS

The following ideas are presented as possible solution concepts designed to add improvements or resolve the issues identified during the research stage, specifically looking into automated mechanisms to integrate reverse engineering programmatic techniques with complex

interpretations:

1. Identifying exploits used in the wild via a Windows host honeypot and a hypervisor monitoring process. The process of debugging Windows-related kernel vulnerabilities can be used in conjunction with address sanitizers to catch zero-day exploits by exposing specific services on the public internet. Windows logs are very broad a targeting all the system-level services can create a storage and processing problem, however, scoping specific features such as the RDP protocol, SMB or RPC will limit the inbound information at permit automatic debugging without losing performance in the processing power of the operating system. Having important ports like the ones previously mentioned can be a perfect candidate for monitoring against attacks targeting publicly facing Windows machines. Logging outside the honeypot configuration is also a must as the debugged virtual machine can be considered as compromised.
2. Identifying exploits targeting browsers by making use of automated debugging capabilities and memory sanitizers. Similarly, having debugging capabilities and instrumentation when fuzzing on the browser can create an environment for monitoring active exploits and zero-days. A setup can be created similar to the manual instrumentation during fuzzing campaigns with address memory sanitizers enabled in a virtual machine that has automatic URL loading. The URLs can be gathered from public malware repositories with domains known as malicious. Loading such URLs inside multiple browsers and having debuggers and sanitizers active creates an environment where potential undisclosed vulnerabilities can be observed. The internal virtual machine used for this automation must the sandboxed in a strict environment as the detonation process will compromise the entire operating system.
3. Automated malware analysis using machine learning and automated virtual kernel debugging on Windows. Bulk malware analysis is one of the problems encountered during the sample triaging process as a researcher is always limited by resources and time when facing manual work. However, this can be automated using machine learning algorithms based on file similarity and combined with kernel debugging using drivers registered in order to bypass userland anti-detection methods. Through kernel mode monitoring and introspection, the full behavior of the analyzed sample can be analyzed by implementing kernel-level hooks on the Windows API calls made by the sample. Additionally, various static and dynamic flags can be extracted from the execute file in order to create an elaborate view of the sample and provide the means to create detections using a combination of flags and behavioral properties. When using machine learning, the corpus provided for negative

and positive feedback is very important, as important as the relearning process of the algorithm that can be user-based.

4. A walkthrough on how and where to apply reverse engineering techniques for different purposes in cybersecurity, which includes the analysis of PE binary files, a research topic that branches into malware analysis and vulnerability research. Furthermore, having a deep understanding of the underlying operating system that offers direct support on the analyzed topic is presented as a mandatory skill that will improve the analysis process, add new techniques to a researcher's arsenal, and help in translating debugging techniques into automated processes that can ease the binary analysis process.

CONCLUSIONS

Reverse engineering is considered a highly technical subject when dealing with closed-source software as it requires a large time investment and the complexity of the analyzed target scales directly with the effort required to understand and successfully analyze the scope. However, the methodology can be applied to many different areas, as the concepts remain largely the same, but the debugging, tools and internal knowledge differ in each case. The result of reversing, however, can produce very meaningful results that can be later managed based on the scope of the assessment. The research conducted aims to tackle a wide variety of applicable scenarios where reverse engineering can be used in the context of vulnerability research and software debugging for both offensive and defensive purposes. The IoT environment connects many of the technologies that take part in the process of creating a consumer product. The overall security posture of such devices and business applications is defined by the security structure of the entire ecosystem. This includes powerful engines such as browsers and the operating system kernel, all of them exposing different areas that attackers could target. By analyzing the current state of attacks and vulnerability usage in the wild, we can note an ever-increasing trend in the utilization of 0-days for launching potent malware campaigns against Windows-based systems or attacks that exploit vulnerabilities in browsers, sometimes in conjunction with operating system exploits in order to bypass browser protections. The required knowledge and capabilities to investigate such issues are highly technical; however, the results have a large impact on the customer base as the affected products are used by billions of users.

The thesis combines a number of subjects linked by reverse engineering techniques,

understanding of internals, and co-dependencies that enable analysis and research from both offensive and defensive applications. One of the solutions presented provides a setup that can be implemented with hypervisors and heuristic algorithms capable of detecting exploits executed on the system. An example of fuzzing campaign is showcased, and metrics are recorded while in progress in order to display the adjustments made on different components such as the corpus used and the fuzzing harness. A browser exploit walkthrough is researched and the steps required to obtain code execution starting from a use-after-free primitive are researched. All the practical cases and solutions tackled in this thesis are supported by the Windows internals research and the Windows hypervisor protection knowledge described that helped to understand the attack surface and available indicators.

REFERENCES

[1] Chadha, R., Shalom, G.S., Anand, V.K. and Goel, A., 2022, November. A Study on Exploit Development. In 2022 7th International Conference on Computing, Communication and Security (ICCCS) (pp. 1-7). IEEE.

[2] Tandon, A. and Nayyar, A., 2019. A comprehensive survey on ransomware attack: A growing havoc cyberthreat. Data Management, Analytics and Innovation: Proceedings of ICDMAI 2018, Volume 2, pp.403-420.

[3] Digging into the numbers one year after Log4Shel, (Dec.2022), https://www.scmagazine.com/feature/third-party-risk/digging-into-the-numbers-one-year-after-log4shell, retrieved Dec.2022

[4] 2022 hacker powered security report, (Feb.2022), https://www.hackerone.com/resources/i/1487910-2022-hacker-powered-security-report/3, retrieved March 2022

[5] What is the Cost of a Data Breach in 2023?, https://www.upguard.com/blog/cost-of-data-breach, Aug.2023, retrieved Aug.2023

[6] The latest 2023 Ransomware Statistics (updated August 2023), https://aag-it.com/the-latest-ransomware-statistics, retrieved Aug.2023

[7] National Institute of Standards and Technology, ICAT Metabase, https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cwe-over-time

[8] Zerodium Exploit Acquisition Program, https://zerodium.com/program.html, retrieved July 2022

[9] Householder, A.D., Chrabaszcz, J., Novelly, T., Warren, D. and Spring, J.M., 2020, August. Historical Analysis of Exploit Availability Timelines. In CSET@ USENIX Security Symposium.

[10] Goel, A.K., Rose, A., Gaur, J. and Bhushan, B., 2019, July. Attacks, countermeasures and security paradigms in IoT. In 2019 2nd international conference on intelligent computing, instrumentation and control technologies (ICICICT) (Vol. 1, pp. 875-880). IEEE.

[11] Pelaez, A., 9 IoT Operating Systems to Use in 2021 [List & Comparison] (May 2021), https://ubidots.com/blog/iot-operating-systems/, retrieved Nov.2021

[12] 2020 IoT Developer Survey Key Findings (Dec.2020), https://f.hubspotusercontent10.net/hubfs/5413615/2020%20IoT%C2%A0Developer%20Survey%20Report.pdf, retrieved March 2021

[13] 2022 IoT Developer Survey Key Findings (Sep.2022), https://f.hubspotusercontent10.net/hubfs/5413615/2020%20IoT%C2%A0Developer%20Survey%20Report.pdf, retrieved January 2024

[14] 2023 IoT Developer Survey Key Findings (Oct.2020), https://f.hubspotusercontent10.net/hubfs/5413615/2020%20IoT%C2%A0Developer%20Survey%20Report.pdf, retrieved January 2024

[15] Nawir, M., Amir, A., Yaakob, N. and Lynn, O.B., 2016, August. Internet of Things (IoT): Taxonomy of security attacks. In 2016 3rd international conference on electronic design (ICED) (pp. 321-326). IEEE.

[16] Chen, J., Diao, W., Zhao, Q., Zuo, C., Lin, Z., Wang, X., Lau, W.C., Sun, M., Yang, R. and Zhang, K., 2018, February. IoTFuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing. In NDSS.

[17] The Security Development Lifecycle (Chapter 1), May 2006, https://download.microsoft.com/download/f/c/7/fc7d048b-b7a5-4add-be2c-baaee38091e3/9780735622142_SecurityDevLifecycle_ch01.pdf, retrieved Dec.2020

[18] Verizon's "Data Breach Investigations Report (DBIR) 2020", January 2020, https://enterprise.verizon.com/resources/executivebriefs/2019-dbir-executive-brief.pdf, retrieved January 2020

[19] Veracode State of Software Security Report, February 2024, https://www.veracode.com/sites/default/files/2024-02/SOSS-Report-2024.pdf, retrieved February 2024

[20] 2019 Thales Data Threat Report, 2019, https://cpl.thalesgroup.com/resources/encryption/2019/data-threat-report, retrieved Feburary 2020

[21] The Cisco 2020 Data Privacy Benchmark Study, January 2020, https://www.cisco.com/c/dam/global/en_uk/products/collateral/security/2020-data-privacy-cybersecurity-series-jan-2020.pdf, retrieved January 2020

[22] Ponemon Institute's "Cost of a Data Breach Report 2020", 2020, https://www.ibm.com/security/digital-assets/cost-data-breach-report/1Cost%20of%20a%20Data%20Breach%20Report%202020.pdf, retrieved March 2020

[23] CYBERSECURITY The new source of competitive advantage for retailers, May 2018, https://www.capgemini.com/fi-en/wp-content/uploads/sites/27/2018/05/cybersecurity-in-retail-report_v2-10.pdf, retrieved April 2020

[24] The Global State of Information Security Survey 2018, January 2018, https://www.pwc.com/sg/en/publications/assets/gsiss-2018.pdf, retrieved April 2020

[25] The Cost Of Fixing An Application Vulnerability, May 2009, https://www.darkreading.com/cyber-risk/the-cost-of-fixing-an-application-vulnerability, retrieved May 2020

[26] Sushil Kumar, Avinash Kaur, Ashish Jolly, Mohammed Baz, Omar Cheikhrouhou, 2021, Mathematical Problems in Engineering

[27] Mitre CVE Buffer Overflow search result, https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Buffer+Overflow, retrieved May.2019

[28] Erick Leon, Ștefan D. Bruda, Counter-measures against stack buffer overflows in GNU/Linux operating systems., The International Workshop on Parallel Tasks on High Performance Computing, Procedia Computer Science 83, 2016, Volume 83, pages 1301 – 1306

[29] A detailed description of the Data Execution Prevention (DEP) feature in Windows XP Service Pack 2, Windows XP Tablet PC Edition 2005, and Windows Server 2003, (Jul.2017)

https://support.microsoft.com/en-us/help/875352/a-detailed-description-of-the-data-execution-pr evention-dep-feature-in, retrieved Dec.2018

[30] Daniel, M., Honoroff, J. and Miller, C., 2008. Engineering Heap Overflow Exploits with JavaScript. WOOT, 8, pp.1-6.

[31] Liu, D., Zhang, M. and Wang, H., 2018, October. A robust and efficient defense against use-after-free exploits via concurrent pointer sweeping. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 1635-1648).

[32] Sood, A.K. and Enbody, R.J., 2011. Browser exploit packs-exploitation tactics. VIRUS BULLETIN, Barcelona, Spain.

[33] Farah, T., Shelim, R., Zaman, M., Hassan, M.M. and Alam, D., 2017. Study of race condition: A privilege escalation vulnerability. In WMSCI 2017-21st World Multi-Conference Syst. Cybern. Informatics, Proc (Vol. 2, pp. 100-105).

[34] Yan Fen, Yuan Fuchao, Shen Xiaobing, Yin Xinchun, Mao Bing, A New Data Randomization Method to Defend Buffer Overflow Attacks, International Conference on Applied Physics and Industrial Engineering, Physics Procedia 24, Volume 24, Part C, 2012, pages 1757-1764

[35] The magic gadget, (Sep.2016), https://github.com/m1ghtym0/magic_gadget_finder, retrieved Apr.2019

[36] Cheng, L., Ilbeyi, B., Bolz-Tereick, C.F. and Batten, C., 2020, February. Type freezing: exploiting attribute type monomorphism in tracing JIT compilers. In Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization (pp. 16-29).

[37] Ding, Y., Wei, T., Wang, T., Liang, Z. and Zou, W., 2010, December. Heap taichi: exploiting memory allocation granularity in heap-spraying attacks. In Proceedings of the 26th Annual Computer Security Applications Conference (pp. 327-336).

[38] Gadaleta, F., Younan, Y. and Joosen, W., 2010, January. BuBBle: A Javascript Engine Level Countermeasure against Heap-Spraying Attacks. In ESSoS (pp. 1-17).

[39] Position Independent Executables (PIE), (Nov.2012), https://access.redhat.com/blogs/766093/posts/1975793, retrieved Jan.2019

[40] Address Space Layout Randomization, (Mar.2003), https://pax.grsecurity.net/docs/aslr.txt, retrieved Feb. 2020.

[41] Bypassing ASLR - Part I, (May 2015), https://sploitfun.wordpress.com/2015/05/08/bypassing-aslr-part-i/, retrieved Dec.2018

[42] Hardening ELF binaries using Relocation Read-Only (RELRO), (Jan.2019), https://www.redhat.com/en/blog/hardening-elf-binaries-using-relocation-read-only-relro, retrieved Jan.2019

[43] Buffer overflow protection, (Jun.2018), https://en.wikipedia.org/wiki/Buffer_overflow_protection#Canaries, retrieved Jan.2019

[44] Return-to-libc Exploit, (Feb.11), https://medium.com/@nikhilh20/return-to-libc-exploit-aa3fe6fb0d69, retrieved Mar.2019

[45] Bypassing DEP with ROP (32-bit), (Dec.2017), https://bytesoverbombs.io/bypassing-dep-with-rop-32-bit-39884e8a2c4a, retrieved Mar.2019

[46] Rogowski, R., Morton, M., Li, F., Monrose, F., Snow, K.Z. and Polychronakis, M., 2017, April. Revisiting browser security in the modern era: New data-only attacks and defenses. In 2017 IEEE European Symposium on Security and Privacy (EuroS&P) (pp. 366-381). IEEE.

[47] Liu, J. and Xu, C., 2018. Pwning microsoft edge browser: From memory safety vulnerability to remote code execution.

[48] Format String Exploitation-Tutorial, https://www.exploit-db.com/docs/english/28476-linux-format-string-exploitation.pdf, retrieved Apr.2019

[49] Ryan "elfmaster" O'Neill, Learning Linux Binary Analysis, Packt, 2016

[50] How to hijack the Global Offset Table with pointers for root shells, (Apr.2006), https://www.exploit-db.com/papers/13203, retrieved Apr.2019

[51] Smashing the Stack, (Apr.2014), http://phrack.org/issues/49/14.html, retrieved Oct.2018

[52] Ryan "elfmaster" O'Neill, Learning Linux Binary Analysis, Packt, 2016

[53] Reis, C., Barth, A. and Pizano, C., 2009. Browser Security: Lessons from Google Chrome: Google Chrome developers focused on three key problems to shield the browser from attacks. Queue, 7(5), pp.3-8.

[54] Russinovich, M.E., Solomon, D.A. and Ionescu, A., 2012. Windows internals, part 2. Pearson Education.

[55] Pietrek, M., 1993. Windows Internals: The Implementation of the Windows Operating Environment. Addison-Wesley Longman Publishing Co., Inc..

[56] Yosifovich, P., Solomon, D.A. and Ionescu, A., 2017. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more. Microsoft Press.

[57] Windows Access Control For pentesters — part 1, March 2021, https://medium.com/@jasemalsadi/intro-to-windows-access-control-part-1-ff4c20b918e7, retrieved April 2021

[58] Russinovich, M. and Solomon, D.A., 2009. Windows internals: including Windows server 2008 and Windows Vista. Microsoft press.

[59] Provecho, E.F., 2017. Testing User Account Control (UAC) on Windows 10.

[60] Parent Process vs. Creator Process, January 2021, https://scorpiosoftware.net/2021/01/10/parent-process-vs-creator-process/, retrieved January 2022

[61] Brown, K., 2000. Programming Windows Security. Upper Saddle River, NJ: Addison-Wesley.

[62] Govindavajhala, S. and Appel, A.W., 2006. Windows access control demystified. Princeton university.

[63] Access Token, July 2021, https://learn.microsoft.com/en-us/windows/win32/secauthz/access-tokens, retrieved July 2021

[64] Wu, J., Arrott, A. and Osorio, F.C.C., 2014, October. Protection against remote code execution exploits of popular applications in Windows. In 2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE) (pp. 26-31). IEEE.

[65] Eagle, C., 2011. The IDA Pro Book, 2nd Edition. No Starch Press.

[66] Wei, T., Wang, T., Duan, L. and Luo, J., 2010, October. Secure dynamic code generation against spraying. In Proceedings of the 17th ACM conference on Computer and communications security, pp. 738-740.

[67] Calbucci, M., 2000. Windows 2000 Security Descriptors. Dr. Dobb's Journal: Software Tools for the Professional Programmer, 25(11), pp.57-61.

[68] Ferguson, J., 2008. Reverse engineering code with IDA Pro. Syngress.

[69] Probert, D.B., Windows Kernel Internals Windows Service Processes. Microsoft Corporation-http://www. iu-tokyo. ac. jp/edu/training/ss/lecture/newdocuments/Lecrures/11WindowsServices/WindowsServices. ppt.

[70] Assi, M.J., Fahad, A.A. and Al-Sarray, B., 2022. Root Cause Analysis and Improvement In Windows System Based on Windows Performance Toolkit WPT. Iraqi Journal of Science, pp.5046-5057.

[71] Li, X., Wen, Y., Huang, M.H. and Liu, Q., 2011, December. An overview of bootkit attacking approaches. In 2011 Seventh International Conference on Mobile Ad-hoc and Sensor Networks pp. 428-431. IEEE.

[72] Probert, D., 2010. Windows Kernel Architecture Internals.

[73] Willems, C., 2011. Internals of Windows memory management (not only) for malware analysis. None.

[74] Gadaleta, F., Strackx, R., Nikiforakis, N., Piessens, F. and Joosen, W., 2014. On the effectiveness of virtualization-based security. arXiv preprint arXiv:1405.6058.

[75] Deogirikar, J. and Vidhate, A., 2017, February. Security attacks in IoT: A survey. In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 32-37). IEEE.

[76] Isolated User Mode (IUM) Processes - Trustlets, July 2021, https://learn.microsoft.com/en-us/windows/win32/procthread/isolated-user-mode--ium--processes#trustlets, retrieved July 2021

[77] Lukacs, S., Lutas, A.V., Lutas, D.H. and Sebestyen, G., 2014, May. Hardware virtualization based security solution for embedded systems. In 2014 IEEE International Conference on Automation, Quality and Testing, Robotics (pp. 1-6). IEEE.

[78] Xiang, C., Bhagoji, A.N., Sehwag, V. and Mittal, P., 2021, August. PatchGuard: A Provably Robust Defense against Adversarial Patches via Small Receptive Fields and Masking. In USENIX Security Symposium (pp. 2237-2254).

[79] Pearce, L., 2018. Windows Internals and Malware Behavior: Malware Analysis Day 3 (No. LA-UR-18-25467). Los Alamos National Lab. (LANL), Los Alamos, NM (United States).

[80] Custer, H., 1992. Inside windows NT. Microcomputer Applications.

[81] Windows vs Linux, https://echelonlinux.org/, retrieved December 2021

[82] Cerrudo, C., 2005. Hacking windows internals. Blackhat Europe, Amsterdam, Netherlands.

[83] Sahel Alouneh, Mazen Kharbutli, Rana AlQurem, Stack Memory Buffer Overflow Protection Based on Duplication and Randomization, The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks, Procedia Computer Science 21, 2013, pages 250 – 256

[84] Moser, A., Kruegel, C. and Kirda, E., 2007, May. Exploring multiple execution paths for malware analysis. In 2007 IEEE Symposium on Security and Privacy (SP'07) (pp. 231-245). IEEE.

[85] Gandotra, E., Bansal, D. and Sofat, S., 2014. Malware analysis and classification: A survey. Journal of Information Security, 2014.

[86] Sikorski, M. and Honig, A., 2012. Practical malware analysis: the hands-on guide to dissecting malicious software. No Starch Press.

[87] Monnappa, K.A., 2018. Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware. Packt Publishing Ltd.

[88] Yong Wong, M., Landen, M., Antonakakis, M., Blough, D.M., Redmiles, E.M. and Ahamad, M., 2021, November. An inside look into the practice of malware analysis. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (pp. 3053-3069).

[89] Quist, D.A. and Liebrock, L.M., 2009, October. Visualizing compiled executables for malware analysis. In 2009 6th International Workshop on Visualization for Cyber Security (pp. 27-32). IEEE.

[90] Mohaisen, A., Alrawi, O. and Mohaisen, M., 2015. AMAL: high-fidelity, behavior-based automated malware analysis and classification. computers & security, 52, pp.251-266.

[91] Liu, W., Ren, P., Liu, K. and Duan, H.X., 2011, September. Behavior-based malware analysis and detection. In 2011 first international workshop on complexity and data mining (pp. 39-42). IEEE.

[92] Bayer, U., Kirda, E. and Kruegel, C., 2010, March. Improving the efficiency of dynamic malware analysis. In Proceedings of the 2010 ACM Symposium on Applied Computing (pp. 1871-1878).

[93] Prayudi, Y. and Riadi, I., 2015. Implementation of malware analysis using static and dynamic analysis method. International Journal of Computer Applications, 117(6).

[92] Eagle, C., 2004. Attacking Obfuscated Code with IDA Pro. Black Hat.

[94] Or-Meir, O., Nissim, N., Elovici, Y. and Rokach, L., 2019. Dynamic malware analysis in the modern era—A state of the art survey. ACM Computing Surveys (CSUR), 52(5), pp.1-48.

[95] Eagle, C. and Nance, K., 2020. The Ghidra Book: The Definitive Guide. no starch press.

[96] Megira, S., Pangesti, A.R. and Wibowo, F.W., 2018, December. Malware analysis and detection using reverse engineering technique. In Journal of Physics: Conference Series (Vol. 1140, No. 1, p. 012042). IOP Publishing.

[97] Bermejo Higuera, J., Abad Aramburu, C., Bermejo Higuera, J.R., Sicilia Urban, M.A. and Sicilia Montalvo, J.A., 2020. Systematic approach to malware analysis (SAMA). Applied Sciences, 10(4), p.1360.

[98] Sun, H.M., Lin, Y.H. and Wu, M.F., 2006. API monitoring system for defeating worms and exploits in MS-Windows system. In Information Security and Privacy: 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006. Proceedings 11 (pp. 159-170). Springer Berlin Heidelberg.

[99] Ucci, D., Aniello, L. and Baldoni, R., 2019. Survey of machine learning techniques for malware analysis. Computers & Security, 81, pp.123-147.

[100] Lengyel, T.K., Maresca, S., Payne, B.D., Webster, G.D., Vogl, S. and Kiayias, A., 2014, December. Scalability, fidelity and stealth in the DRAKVUF dynamic malware analysis system. In Proceedings of the 30th annual computer security applications conference (pp. 386-395).

[101] Ferreira, D.T., 2023. Automatic binary patching for flaws repairing using static rewriting and reverse dataflow analysis (Doctoral dissertation).

[102] Vasilescu, M., Gheorghe, L. and Tapus, N., 2014, September. Practical malware analysis based on sandboxing. In 2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference (pp. 1-6). IEEE.

[103] Sabanal, P.V. and Yason, M.V., 2007. Reversing C++. Black Hat DC.

[104] Egele, M., Scholte, T., Kirda, E. and Kruegel, C., 2008. A survey on automated dynamic malware-analysis techniques and tools. ACM computing surveys (CSUR), 44(2), pp.1-42.

[105] Ferenc, R., Beszédes, Á., Tarkiainen, M. and Gyimóthy, T., 2002, October. Columbus-reverse engineering tool and schema for C++. In International Conference on Software Maintenance, 2002. Proceedings. (pp. 172-181). IEEE.

[106] Bruce Dang , Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation, Wiley Publishing, 2014

[107] Reverse-Tips, June 2022, https://ppppz.net/2022/06/12/Reverse-Tips/, retrieved June 2022.

[108] Eldad Eilam, Reversing: Secrets of Reverse Engineering, Wiley Publishing, 2005

[109] DRAKVUF Black-box Binary Analysis System, https://drakvuf.com/, retrieved Feb. 2019.

[110] CVE-2020-1135, November 2019, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-1135, retrieved December 2019.

[111] Sutton, M., Greene, A. and Amini, P., 2007. Fuzzing: brute force vulnerability discovery. Pearson Education.

[115] Madjid Kara, Olfa Lamouchi, Amar Ramdane-Cherif, Software quality assessment algorithm based on fuzzy logic, International Journal of ubiquitous systems and pervasive networks (JUSPN), volume 8, issue1, 2017, pages 01-09

[113] Cha, S.K., Woo, M. and Brumley, D., 2015, May. Program-adaptive mutational fuzzing. In 2015 IEEE Symposium on Security and Privacy (pp. 725-741). IEEE.

[114] Chen, C., Cui, B., Ma, J., Wu, R., Guo, J. and Liu, W., 2018. A systematic review of fuzzing techniques. Computers & Security, 75, pp.118-137.

[115] Böhme, M., Pham, V.T. and Roychoudhury, A., 2016, October. Coverage-based greybox fuzzing as markov chain. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 1032-1043).

[116] Böhme, M., Pham, V.T., Nguyen, M.D. and Roychoudhury, A., 2017, October. Directed greybox fuzzing. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 2329-2344).

[117] Pham, V.T., Böhme, M., Santosa, A.E., Căciulescu, A.R. and Roychoudhury, A., 2019. Smart greybox fuzzing. IEEE Transactions on Software Engineering, 47(9), pp.1980-1997.

[118] Zeller, A., Gopinath, R., Böhme, M., Fraser, G. and Holler, C., 2019. The fuzzing book.

[119] Serebryany, K., 2016, November. Continuous fuzzing with libfuzzer and addresssanitizer. In 2016 IEEE Cybersecurity Development (SecDev) (pp. 157-157). IEEE.

[120] Lee, B., Song, C., Jang, Y., Wang, T., Kim, T., Lu, L. and Lee, W., 2015, February. Preventing Use-after-free with Dangling Pointers Nullification. In NDSS.

[121] Österlund, S., Razavi, K., Bos, H. and Giuffrida, C., 2020, August. Parmesan: Sanitizer-guided greybox fuzzing. In Proceedings of the 29th USENIX Conference on Security Symposium (pp. 2289-2306).

[122] Zhu, X. and Böhme, M., 2021, November. Regression greybox fuzzing. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (pp. 2169-2182).

[123] Herrera, A., Gunadi, H., Magrath, S., Norrish, M., Payer, M. and Hosking, A.L., 2021, July. Seed selection for successful fuzzing. In Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 230-243).

[124] Devi, D., Pathak, D. and Nandi, S., 2010. Vulnerabilities in Web Browsers. Indian Institute of Technology., Guwahati., India.

[125] Barth, A., Jackson, C., Reis, C. and TGC Team, 2008. The security architecture of the chromium browser. In Technical report. Stanford University.

[126] Analysis of a Chrome Zero Day: CVE-2019-5786, March 2020, https://www.mcafee.com/blogs/other-blogs/mcafee-labs/analysis-of-a-chrome-zero-day-cve-2019-5786/, retrieved March 2019

[127] Sotirov, A. and Dowd, M., 2008. Bypassing browser memory protections. Proceedings of BlackHat, 2008.

[128] Sayar, I., Bartel, A., Bodden, E. and Le Traon, Y., 2023. An in-depth study of java deserialization remote-code execution exploits and vulnerabilities. ACM Transactions on Software Engineering and Methodology, 32(1), pp.1-45.

[129] CVE-2019-5786 Chrome 72.0.3626.119 stable FileReader UaF exploit for Windows 7 x86, Mar. 2019, https://github.com/exodusintel/CVE-2019-5786, retrieved June 2021

[130] Manhas, S. and Taterh, S., 2018. A Comparative Analysis of Various Vulnerabilities Occur in Google Chrome. In Soft Computing: Theories and Applications: Proceedings of SoCTA 2016, Volume 1 (pp. 51-59). Springer Singapore.