On Teaching Malware Analysis on Latest Windows

Lan Luo
Dept. of Computer Science
University of Central Florida
Orlando, FL, USA
lukachan@knights.ucf.edu

Cliff Zou
Dept. of Computer Science
University of Central Florida
Orlando, FL, USA
czou@cs.ucf.edu

Sashank Narain
Dept. of Computer Science
University of Massachusetts
Lowell, Lowell, MA, USA
sashank_narain@uml.edu

Xinwen Fu
Dept. of Computer Science
University of Massachusetts
Lowell, Lowell, MA, USA
xinwen_fu@uml.edu

Abstract—Microsoft Windows operating systems are the most popular desktop operating systems. 83% of malware attacks target Windows. Windows 10 has a market share of 78.45% out of all Windows versions on market. However, we find security related courses are often taught on Linux or run on older Windows versions. In this paper, we present our practice of teaching malware analysis on the latest Windows (10). We are among the first using the latest Windows (10) for teaching malware analysis. We design the labs and assignments on the pre-configured Windows 10 VM supplemented by the Kali VM. A virtual Cyber Range is created for students to access the two VMs over a cloud. We present our curriculum and learning assessment scheme. Our practice has been validated through surveys on both face-to-face and online classes.

Keywords—malware analysis, Windows, VM, XCP-ng

I. INTRODUCTION

Malware is a major cyber attack and has been actively developed to exploit and infect victim computers. It is reported that there were 1266.70 millions malware programs by September 2021 [1] and more than 114 million (114,312,703) new malware applications were developed in 2019 [2]. Recent malware attacks were the ransomware attacks against Colonial Pipeline [3], computer manufacturer Acer and NBA. Major malware programs include trojans, ransomware, backdoors, worms, viruses, and crypto miners.

The use of the Microsoft Windows operating systems (OSes) and malware attacks against Windows are pervasive. According to StatCounter [4], Windows has a market share of 30.20% out of all operating systems including desktop OSes and mobile OSes as shown in Fig. 1. If only the desktop OS is considered, Windows has a market share of 80.38% as shown in Fig. 2. According to PCMag [5], 83% of malware attacks in the first quarter of 2020 targeted Windows computers. It can be observed that teaching malware analysis on Windows is critical given the large market share of Microsoft Windows and trend of malware attacks against Windows.

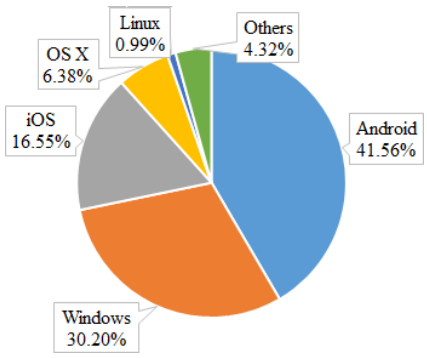

Fig. 1. Major Operating System Market Share in July 2021

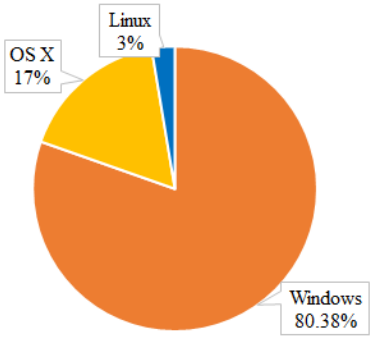

Fig. 2. Desktop Operating System Market Share in July 2021

We performed multiple surveys on student preference of Windows version for learning malware analysis. Every time all students prefer the latest Windows. Fig. 3 shows the result of the survey of 8 students in an online malware analysis class in August 2021. All Students do prefer the latest Windows 10, which is also the dominant Windows version with a market share over 78% according to StatCounter [6] as shown in Fig. 4. The student response is also consistent with Kaylene Williams and Caroline Williams's findings on various ingredients that can improve student motivation [7]. They find that the content shall be *timely*, accurate, useful, and *relevant to students in their life*. Such a realistic environment motivates students to learn.

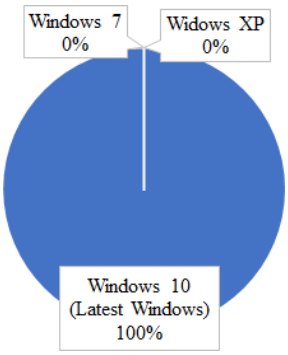

Fig. 3. Windows version preference by students

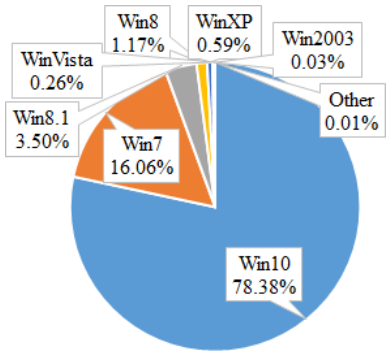

Fig. 4. Desktop Windows Version Market Share in July 2021

However, we find security related courses are often taught on Linux or run on older Windows versions. For example, the widely used SEED hands-on labs for security education are based on Ubuntu. Some malware analysis courses at other institutes run on Windows [8], [9]. Many of those courses are based on the book titled "Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software" by Michael Sikorski and Andrew Honig published on Feb 1, 2012 [10]. All exercises and contents of this book are based on Windows XP while some courses run on Windows 7.

In this paper, we present our best practice of running the malware analysis course on Windows 10. We also plan to migrate our malware analysis course to the latest Windows when there is a new version in the future. The major contributions of this paper can be summarized as follows:

- We are among the first using the latest Windows (10) teaching malware analysis. Our curriculum covers how the attacker can hack into victim computers through software vulnerabilities, how the attacker deploys malware and malware behaviors, and static and dynamic analysis of malware.
- Our practice has been validated through both face-to-face and online classes on malware analysis. We create and pre-configure two virtual machines, one Windows 10 VM and one Kali Linux VM with pre-installed tools. The two VMs can be deployed over desktop computers and can also be accessible online through our virtual Cyber Range, powered by XCP-ng [11] and Xen Orchestra [12]. Our survey results show the classes achieve the objectives of the malware analysis course.

The rest of this paper is organized as follows. We introduce our curriculum in Section II and present the survey of the course in Section III. The paper is concluded in Section IV.

II. Curriculum

In this section, we first present the two VMs used in our malware analysis course. The lab setup is introduced next. We then present our modules on malware analysis and learning assessment scheme at last.

A. Pre-configured Windows 10 and Kali Linux VMs

For malware analysis, a virtual machine (VM) environment is necessary. When a VM is messed up with malware deployed by students, we can roll back to a previous good snapshot. Virtualization software also allows flexible networking of multiple VMs so that we can interconnect all the VMs or isolate particular VMs. VMs can create a safe sandbox for students to play with malware and restart with a clean system conveniently.

For our malware analysis course, we use the Windows 10 VM with pre-installed tools supplemented by the Kali Linux VM. The official Kali VM is pre-installed with a variety of tools including Metasploit. We add BooFuzz [13] onto the Kali VM for network protocol fuzzing. For the Windows VM, we install tools in Table I. We actually performed a survey in a 2021 summer class on malware analysis to see if students want to install tools themselves. Fig. 5 shows the survey result of 8 students. Please note other survey results in this section came from the same group of students. Students do prefer VMs with pre-installed tools, which allow them to focus on learning the knowledge of malware analysis, not distracted by tool installation. Tool installation sometimes can get very complicated because of obsolete links, versions and dependent modules, and may frustrate students very much.

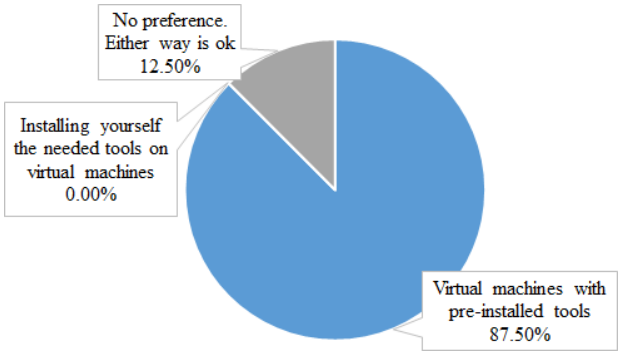

Fig. 5. Student preference of virtual machines with pre-installed tools

We configure networking and security features of Kali Linux and Windows 10 VMs so that there is minimum

configuration by students. The two VMs are created with the Oracle VM VirtualBox and use the *NAT Network* mode so that the two VMs can communicate with each other and access the Internet. We did not find students abuse the Internet although we could have removed Internet access. We disable all Windows 10 *exploit protection* settings, turn all *Windows Defender Firewall* settings off and turn all the *Virus & threat protection* settings off. The *real-time protection* of *Virus & threat protection* settings has to be turned off before running malware samples every time since this setting turns back on after a while automatically.

B. Lab setup

We have used the two VMs in two lab environments. First, the two VMs can be installed on student personal computers or lab computers. The advantage of this setup is the students have full control of the VMs and may experiment on different network configurations. One issue we had with this practice is downloading the two VMs is a big challenge given the large size of the VMs. The .ova (Open Virtual Appliance) file of Kali VM is over 6.0GB and the .ova file of Windows 10 VM is over 25GB. Second, we create a virtual Cyber Range (i.e., a cloud) and allocate the two VMs to students in need. We performed a survey of student preference of the lab environment. Fig. 6 shows each lab environment is preferred by half of the students.

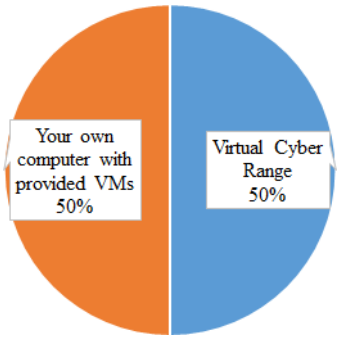

Fig. 6. Student preference of Lab Environment

Due to COVID-19, we created the virtual Cyber Range as follows. All Dell desktops and server in our physical Cyber Range are loaded with XCP-ng hypervisor [11], and managed by Xen Orchestra [12]. VMs can be created on both desktops and server. To access their cloud VMs in the virtual Cyber Range, students need to first connect to the Cyber Range VPN. This process requires each student to request an OpenVPN profile from the administrator.

We also run servers on VMs in the virtual Cyber Range for following purposes: (i) providing administrative services to the networked hypervisors, (ii) easing course delivery and lab experiments for faculty and students, (iii) providing a networked environment to students for course labs and practicing for competitions such as Collegiate Cyber Defense Competition. Note that all software used is open-source and widely used worldwide. Below are a few critical servers. (i) OpenVPN Server (OS: Ubuntu). This server runs scripts that generate OpenVPN certificates, which faculty and students can use to access the Cyber Range remotely. The certificates are auto synced with Dropbox in order to ease sharing with faculty and students. (ii) OpenLDAP Server (OS: Ubuntu). This server manages user credentials for faculty and students accessing the Cyber Range. It runs scripts to ease creation / revocation of credentials for faculty and students whenever required. (iii) VyOS Firewall (OS: VyOS). This is a software-based firewall that is used to control access to all VMs hosted in the Cyber Range that provide administrative capabilities. (iv) DNS Server (OS: Ubuntu, Software: bind9). This server is setup in order to provide name resolution for different services hosted within the Cyber Range (e.g., VPN, LDAP, Gitlab, Samba). Therefore, faculty and students can access the services without having to specifically remember the IP addresses for these services.

C. Malware Analysis Modules

We build the curriculum based on the popular book "Practical Malware Analysis" [10] while significant amount of new content is added. This book has two big issues: (i) All the labs are based on Windows XP and most of them do not work on Windows 10. We have created labs on Windows 10. (ii) The book lacks details on many knowledge units and is hard to understand sometimes although the book explains the high level concepts well. We provide supplemental materials on the missing knowledge units.

1) Module 1: Malware Analysis in Virtual Machines: This module covers why we use virtual machines for malware analysis and different types of virtual machines. We add supplemental materials on VirtualBox and its networking modes. For example, we provide details on installing VirtualBox, VirtualBox Extension Pack and VirtualBox Guest Additions. For networking, we introduce if VMs in different modes can access each other and the Internet.

2) Module 2: Basic Static Techniques: This module introduces basic static techniques, including antivirus scanning, hashing, finding strings, packed and obfuscated malware, portable executable (PE) file format and linked libraries and functions. "Static" means that we are not going to run the malware of interest, just check the static binary of the malware and see anything suspicious and interesting. Finding all the tools in Table I is a daunting task since we need tools working on the latest Windows 10 and need to make sure there is no malware embedded in the tools on the Internet. We carefully find all the needed tools and install them in the Windows 10 VM.

TABLE I. TOOLS ON WINDOWS VM

Static Analysis	
strings	finding strings in binaries
IDA Freeware	Free disassembler
IDA Pro	Professional disassembler
Resource Hacker	resource editor

Static Analysis	
Dependency Walker	finding dependent modules
PEview	determining basic Portable Executable (PE) in formation
PEiD	finding packers, cryptors and compilers for PE files
Regshot	Taking snapshots of the registry for comparison
strings	finding strings in binaries

Dynamic Analysis	
Immunity Debugger	Debugger
mona	Plugin for Immunity Debugger
FrausDNS	A Windows DNS Spoofer for redirecting network traffic developed by us
Process Explorer	Discovering process information
Process Monitor	Combining legacy tools Filemon and Regmon with more features
Wireshark	Network protocol analyzer

Compilation	
MinGW-w64	gcc for 32 and 64 bit Windows, including *gdb*
nasm	Assembler
golink	Linker
Visual Studio	VC++ and others
Python	Python 2.7 and 3

Editor	
Notepad++	a handy editor

Misc	
7zip	Compression and decompression tool
md5deep	md5 hash tool
GlobalProtect	VPN client

Misc	
Sysinternals Suite	troubleshooting tools including listdlls, WinObj, streams
vulnserver	Vulnerable server extended by us
arwin	win32 address resolution program finding the address of a function in a DLL
WinHelp	Opening legacy Windows help files
setdll	Part of Microsoft Research detours toolkits
Other PE editing and viewing tools	*FileAlyzer*, *CFF Explorer*, *PEstudio*, *Exeinfo*
HxD	Hex Editor
shellcode2bin.py	Converting shellcode in hexadecimals to binary
TeamViewer-setup	Remote desktop

3) Module 3: Basic Dynamic Techniques: When we use dynamic techniques for malware analysis, we run the executable of interest and observe what it does. Why do we need dynamic techniques given that we already have static techniques? It is because the basic static techniques really only provide some hints about the executable. When we run an executable binary, we see the results and may know whether the executable binary is malicious and what it might do. This module covers sandboxes, running malware in a .dll file, monitoring with Process Monitor, viewing processes with Process Explorer, comparing registry snapshots with Regshot, faking a network, packet sniffing with Wireshark and using INetSim. The book "Practical Malware Analysis" uses ApateDNS for faking a network as a DNS spoofer. ApateDNS requires the obsolete legacy Windows .net framework. We created a similar Windows DNS spoofer called FrausDNS [14] using the latest Windows .net framework.

4) Module 4: A Crash Course in X86 Disassembly: This module covers the x86 assembly language and related topics. Assembly language is just another computer programming language. There is a one-to-one mapping from assembly language instructions to machine code instructions. An executable can always be disassembled to assembly language code. For malware analysis, we often have to deal with assembly code because we often do not have access to the source code of the malware. We can use tools to disassemble the machine code into assembly code, which is much easier to read. We use Notepad++ as an editor since it is small and easy to use. nasm and Mingw-w64 (gcc) are used to assemble and link the assembly code respectively if no Windows APIs are used and Linux like binary is preferred. nasm and golink are used if Windows APIs are used in assembly code. Our GitHub repository [15] offers a big picture of assembly language and simple working assembly

language code examples, which are missing in the book "Practical Malware Analysis".

5) Module 5: IDA Pro/Freeware: This module introduces the disassemblers IDA Pro/Freeware, covering loading an executable, IDA Pro interface, using graphing options, using cross-references, analyzing functions and extending IDA with plug-ins. We have purchased IDA Pro. However, IDA Freeware is enough for this course, is free, and does not need a license server. IDA Freeware supports x86/x64 processors, and PE (used by Windows), ELF (used by Linux) and Mach-O (Mach object) file formats.

6) Module 6: Debugging: This module introduces the basic concepts and techniques of debugging. *gdb* (the GNU Project debugger) is a very popular debugging tool for Linux. One reason why we introduce this debugger in this module is *gdb* is a command line tool and we may use a command line script/syntax with it. A GUI debugger such as Immunity Debugger may not have this functionality.

7) Module 7: Immunity Debugger: This module introduces Immunity Debugger while the book "Practical Malware Analysis" uses OllyDbg, the ancestor of Immunity Debugger. Immunity Debugger has all the functionalities of OllyDbg and more. It can disassemble the binary, present the assembly code with its memory address, set breakpoints, allow us to step through the code and look at the changes of registers, stack and memory. We often use Immunity Debugger and IDA Pro/Freeware together. IDA Pro/Freeware can tell where the user written code is through its F.L.I.R.T. technology so that we can set a breakpoint at the right place, for example main(), within Immunity Debugger.

Compared with *gdb*, the advantage of Immunity Debugger is the GUI interface so that we do not need to remember so many commands as *gdb* has. But *gdb* has its advantage: it is a command line tool so that it can work with command shell scripts. For example, we can save a complicated command line argument (e.g., a string that contains malicious shellcode) into an environment variable and feed it to a victim program/malware. We then can use *gdb* to debug the program with the special argument.

8) Module 8: Shellcode Analysis: This module covers loading shellcode for analysis position-independent code, identifying execution location, manual symbol resolution, shellcode encodings, NOP sleds and finding shellcode. We use this module to cover the buffer overflow attack in depth to demonstrate how attackers can hack into a computer. We provide examples of buffer overflow attacks in different scenarios, including feeding the malicious string through the command line argument and remotely, and placing the payload before and after the return address of a function in the victim program. We have extended vulnserver [15] so that vulnserver works as a vulnerable chat server with a variety of vulnerabilities including buffer overflow vulnerabilities. In this way, the application scenario is more realistic since students can use vulnserver for chatting. Students may attack vulnserver from the Kali VM remotely and exercise techniques of metasploit.

9) Module 9: Analyzing Malicious Windows Programs: This module covers many aspects of Windows programming, including Windows API, registry, networking APIs, following running malware, kernel vs. user mode, and native API and introduces how malware can maneuver in Windows once it gets into the victim computer through tricks such as buffer overflow attacks. We cover details on how shellcode may use *CreateProcess*() to create a shell at a victim computer so that the attacker can work in a shell of the victim computer.

10) Module 10: Malware Behavior: This module covers many different types of Windows malware and how they are programmed and perform. Once the malware gets into the victim computer, if the attacker is familiar with Windows APIs, it can be trivial to implement various tricks of malware. We also cover the cyberattack cycle, metasploit and Armitage, which is a GUI front-end for the Metasploit Framework. Kali VM is already installed with metasploit and we add Armitage on Kali. Metasploit and armitage can be used to demonstrate the entire cyberattack cycle: (i) Launch scans; (ii) Choose exploits and check which exploits work; (iii) Perform post-exploitation; (iv) Setup and use pivots for future and collaborative attacks.

D. Learning Assessment

We have created the following assignments/labs, projects, discussion forums and exams to assess how well students master the contents. Assessments: (i) Assignment 1–Virtual machines for malware analysis with VirtualBox and Cyber Range; (ii) Assignment 2–Basic static analysis of a Windows keystroke logging malware program, which sends keystroke logs to a Gmail; (iii) Assignment 3–Basic dynamic analysis of the keystroke logging malware; (iv) Assignment 4–A crash course in X86 disassembly with nasm, MinGW-W64 (gcc), golink and provided assembly code examples; (v) Assignment 5–Use of IDA Pro/Freeware disassembling a binary and identifying hardcoded credentials; (vi) Assignment 6–Debugging via *gdb* a program with the buffer overflow vulnerability and observing the stack change; (vii) Assignment 7–Immunity Debugger on a buffer overflow attack that overwrites a local variable on the stack in order to reveal a secret message embedded in a program even if the password is wrong; (viii) Assignment 8–Manually generating shellcode that will be fed into a victim program as a command line argument and pop up the Windows calculator; (ix) Assignment 9–Analyzing a malicious Windows reverse shell malware program; (x) Assignment 10–Use of metasploit to communicate with a legitimate program (e.g. *putty*) embedded with a backdoor created with msfvenom. The term project is performing a buffer overflow attack through *jmp esp*, which is used to jump to the payload in the malicious string. The return address is overwritten with the address of *jmp esp*. We also set up discussion forums for each module and students need to answer two questions for each module. We have the midterm and final exams to assess the overall learning outcomes from the students.

III. Evaluation

In this section, we first present the class setup and then show the survey result evaluating the effectiveness of the curriculum presented in Section II on malware analysis learning.

A. Class Setup

In the malware analysis class in Summer 2021, the students read the textbook ("Practical Malware Analysis"), notes and slides and optionally chat with the instructor for an hour each week. The assessments include discussion forum questions and assignments for each module, one term project and the midterm and final exams.

We have performed multiple anonymous surveys on similar classes over Google Forms. The results are similar. We accidentally lost data of early surveys and will focus on the survey conducted in August 2021 in this section. There were 8 students and all of them responded to the survey.

B. Effectiveness of Modules on Malware Analysis Learning

We ask students to rate their knowledge level of malware analysis before and after they take this class. "1" means no knowledge and "5" means expert on malware analysis. Fig. 7 and Fig. 8 show the results. The average knowledge level on malware analysis before the class is 1.125 while the average knowledge level after the class is 3.25. Fig. 9 shows the survey result of the 8 students for the question "Would you recommend the class to others?". It can be observed students do like the course and would recommend the class to other students. We ran an online GenCyber high school summer camp using our virtual Cyber Range on weekdays from Jul. 12–23, 2021. our GenCyber curriculum contains the software security module including the buffer overflow attacks and metasploit. The students used the Kali VM to attack the vulnserver chat server. The attack scripts in Python were provided to the students although they had to change the victim IP, port and generate the corresponding shellcode through *msfvenom*. Fig. 10 shows the survey result for the question "I learned a lot about cybersecurity". 46 students responded and 95.65% of them answered "Strongly Agree" or "Agree".

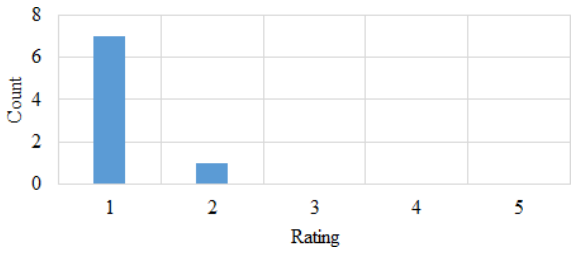

Fig. 7. Please rate your knowledge level of malware analysis BEFORE you take this class

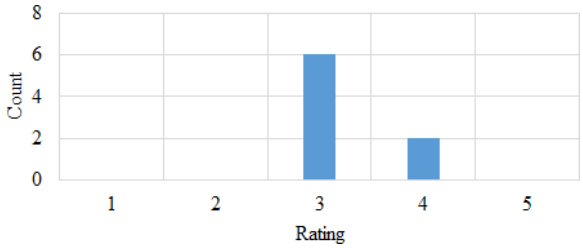

Fig. 8. Please rate your knowledge level of malware analysis AFTER you take this class

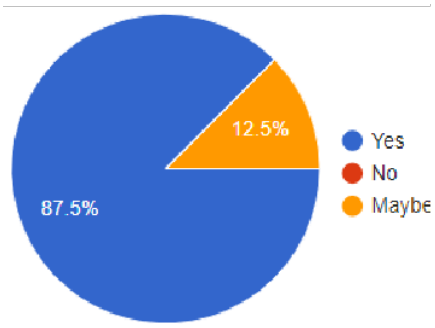

Fig. 9. Would you recommend the class to others?

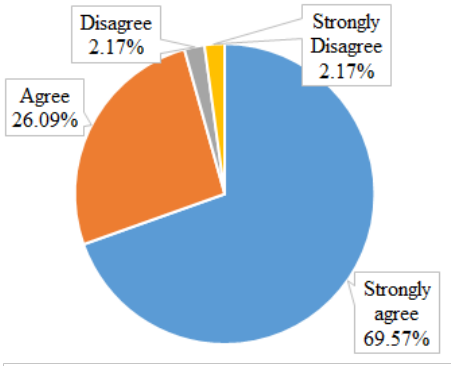

Fig. 10. GenCyber: I learned a lot about cybersecurity

IV. Conclusion

In this paper, we present our practice of teaching malware analysis on the latest Windows (10). Our curriculum is built upon the popular "Practical Malware Analysis" book. However, the book is old. All the labs are built on Windows XP and there are a lot of missing technical details. We create corresponding assignments/labs on Windows 10 and supplement the book with details. We have offered the course both face-to-face and online with pre-configured Windows 10 VM and Kali VM. Students may access the two VMs in the cloud over our virtual Cyber Range powered by open source tools XCP-ng and Xen Orchestra. Our practice has been validated by surveys on the online and face-to-face courses on malware analysis.

ACKNOWLEDGEMENTS

This research was supported in part by US National Science Foundation (NSF) Awards 1931871 and 1915780, and NSA GenCyber 21-MA-UMLx-UV-S1. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] AV-TEST GmbH - The Independent IT-Security Institute, Magdenburg Germany, "Malware," 2020, last accessed September 13 2021. [Online]. Available: https://www.av-test.org/en/statistics/malware/

[2] AV-TEST GmbH - The Independent IT-Security Institute, Magdenburg Germany "Security report 2019/2020," 2020, last accessed September 13 2021. [Online]. Available: https://www.av-test.org/en/news/facts-analyses-on-the-threat-scenario-the-av-test-security-report-2019-2020/

[3] "The 10 biggest ransomware attacks of 2021," JUNE 2021, last accessed September 13 2021. [Online]. Available: https://illinois.touro.edu/news/the-10-biggest-ransomware-attacks-of-2021.php

[4] StatCounter, "Operating system market share worldwide," 2021, last accessed 16 August 2021. [Online]. Available: https://gs.statcounter.com/os-market-share

[5] J. Cohen, "Windows computers were targets of 83% of all malware attacks in q1 2020," August 2020, last accessed 16 August 2021. [Online]. Available: https://www.pcmag.com/news/windows-computers-account-for-83-of-all-malware-attacks-in-q1-2020

[6] StatCounter, "Desktop windows version market share worldwide," 2021, last accessed 16 August 2021. [Online]. Available: https://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide

[7] K. C. Williams and C. C. Williams, "Five key ingredients for improving student motivation," Research in Higher Education Journal, pp. 104– 122, 2011.

[8] RPISEC, "Malware analysis - csci 4976," 2015, last accessed 16 August 2021. [Online]. Available: https://github.com/RPISEC/Malware/

[9] DSU, "Csc 428 - reverse engineering," last accessed 16 August 2021. [Online]. Available: https://catalog.dsu.edu/preview_course_nopop.php?catoid=27&coid=17244

[10] M. Sikorski and A. Honig, Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, 1st ed. USA: No Starch Press, 2012.

[11] XCP-ng, "Turnkey open source hypervisor," 2021, last accessed 18 August 2021. [Online]. Available: https://xcp-ng.org/

[12] xen-orchestra, "Turnkey solution for xenserver and xcp-ng," 2021, last accessed 18 August 2021. [Online]. Available: https://xen-orchestra.com/

[13] J. Pereyda, "boofuzz: Network protocol fuzzing for humans," 2021, last accessed 17 August 2021. [Online]. Available: https://boofuzz.readthedocs.io/en/stable/

[14] C. MoralesGonzalez, "Frausdns–a windows dns spoofer," 2021, last accessed 18 August 2021. [Online]. Available: https://github.com/ChrisM09/FrausDNS

[15] X. Fu, "Malware analysis," 2021, last accessed 18 August 2021. [Online]. Available: https://github.com/xinwenfu/Malware-Analysis