A Laboratory for Hands-on Cyber Threat Hunting Education

Jinpeng Wei
Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC, USA
jwei8@uncc.edu

Bei-Tseng "Bill" Chu
Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC, USA
billchu@uncc.edu

Deanne Cranford-Wesley
Davis iTEC Cyber Security Center
Forsyth Technical Community College
Winston-Salem, NC, USA
dwesley@forsythtech.edu

James Brown
Davis iTEC Cyber Security Center
Forsyth Technical Community College
Winston-Salem, NC, USA
jbrown@forsythtech.edu

***Abstract*—Cyber threat hunting has emerged as a critical part of cyber security practice. However, there is a severe shortage of cybersecurity professionals with advanced analysis skills for cyber threat hunting. Sponsored by NSA, the University of North Carolina at Charlotte (UNC Charlotte) and Forsyth Technical Community College (Forsyth Tech) have been developing freely-available, hands-on teaching materials for cyber threat hunting suitable for use in two-year community college curriculum, 4-year universities curriculum, as well as for collegiate threat hunting competitions. Our hands-on labs focus on exercising a set of essential technical skills (called the threat hunting skill set) in an enterprise environment and they are modeled after real-world scenarios. Our lab environment contains real threats (e.g., malware) against real software (e.g., Operating Systems and applications), and real security datasets. These labs are designed to help a student learn how to detect active and dormant malware, analyze its activities, and assess its impact. These labs also teach a student how to search and probe for anomalies in a variety of datasets using multiple analytical skills, such as statistical analysis. In this paper, we present the design and implementation of our hands-on labs.**

Keywords—cyber hunting, hands-on labs, malware analysis, security data analytics, virtualization, Nice Framework, Nice categories, cyber defense

I. Introduction

Cyber threat hunting has emerged as a critical part of cyber security practice [1][2][3][7][8][9][11][13][17]. For example, in a survey of 494 IT professionals by SANS Institute, 86% of respondents are interested in threat hunting; about 75% said more aggressive threat hunting had reduced their attack surface; however, more than 40% do not have a formal threat hunting program in place [13]. Threat hunting has gained a lot of attention in the community, for example, it was extensively discussed at RSA Conference [14] and InfoSecurity Magazine [15]. There are cyber threat hunting training materials in industry [10] (e.g., SANS Institute [4][5] and Focal Point [16]) but they are expensive and have limited coverage. In academia, one prominent example of free hands-on labs for security education is SEED [12][18]. However, SEED does not cover cyber threat hunting. We use these existing projects and their practices as great examples for us to learn from when we build teaching materials in the area of Threat Hunting in this project.

Colleges and universities have not yet focused on developing threat hunting education material to prepare students for this important area. Threat hunting differs from many traditional cyber security activities such as cyber defense, penetration testing, and forensics. It is a highly unstructured task that demands deep technical know-how, data analytics savvy, and out of the box thinking [20]. Efforts in defining cyber security knowledge units, e.g. NICE and NSA/DHS CAE-CD, have identified basic cyber security skills. However, cyber threat hunting requires students to develop analytical skills that integrates/synthesizes basic cyber security skills. We envision that labs developed can be used in a variety of ways in a typical cyber security curriculum. Some labs can be used as capstone projects in traditional security courses such as introduction to information security, network security, and computer forensics. Other labs may be used in a cyber threat hunting class or used in competitions.

Like many other sectors, automation using Artificial Intelligence is impacting the cyber security industry. System administration jobs are being reduced by automated management tools. Security orchestration and workflow automation is also reducing the need for human intervention in cyber defense. At the same time, there is increased demand for cybersecurity professionals with more advanced analysis skills. Cyber threat hunting is an example of advanced analysis skills in great demand.

Our hands-on labs focus on exercising a set of essential technical skills (called the threat hunting skill set, detailed in Section II.A) in an enterprise environment and they are modeled after real-world scenarios. Our lab environment contains real threats (e.g., malware) against real software (e.g., Operating Systems and applications), and real security datasets. These labs are designed to help a student learn how to detect active and dormant malware, analyze its activities, and assess its impact. Moreover, these labs teach a student how to search and probe for anomalies in a variety of datasets using multiple analytical skills, such as statistical analysis, machine learning, and data visualization. Our labs are designed at different difficulty levels suitable for use by two-

This work is supported by National Security Agency through grant number H98230-17-1-0354.

979-8-6425-4896-7/20/$26.00 ©2020 CISSE

year community college students, 4-year university students, as well as for collegiate threat hunting competitions.

In terms of research method, we first identify the set of technical skills necessary for threat hunting by surveying related work (in both industry and academia) and considering CAE-C Knowledge Units (KUs), next we design, build, and test hands-on labs that cover these technical skills at different difficulty levels (guided by Bloom's Taxonomy [22]). Finally, we evaluate the effectiveness of our labs by collecting feedback from users.

The rest of this paper is organized as follows. Section II presents the design of our hand-on labs, including the threat hunting skill set, threat detection and analysis labs, and security data analytics labs. Section III describes the virtualization based implementation of our hands-on labs. Section IV discusses the current status of our project and possible future work, and Section V concludes the paper.

II. Design of the Threat Hunting Labs

A. The Threat Hunting Skill Set

In general, successful threat hunting requires several technical skills shown in Fig. 1. Threat hunting is often triggered by either an internal incident (e.g., a warning generated by the IDS) or external threat intelligence (e.g., report about a new threat actor, such as WannaCry). Security incidents are important indicators of potential threats. Therefore, incident analysis is an important first step in effective threat hunting. Similarly, a threat hunter must know how to assess threat intelligence information from the community. The hunting for potential threats relies on analysis of security related data, such as event logs, DNS requests, and packet captures. For threats that can either modify security log data or attack security tools directly, memory forensics is an alternative way to locate malware. Once malware is identified, a threat hunter must analyze the malware in order to understand its history, impact, and capabilities (i.e., what it can potentially do in the future). Our hands-on exercises cover as many such skills as possible, and we give priority to the skills at the top of the list.

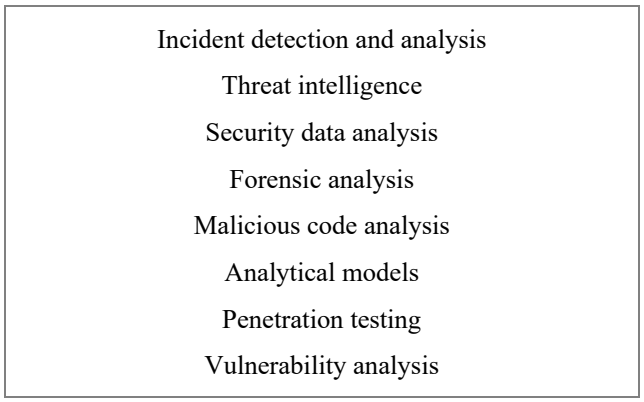

Fig. 1. Threat Hunting Skill Set (Ordered by Priority)

B. Challenges and Design Decisions

The central challenge of our lab design is how to construct representative threat scenarios suitable for training. For example, what should be our criteria of choosing malware samples? How much security data should be used in data analytics exercises so that the data does not tell a toy story yet time required to finish the hands-on labs is reasonable?

We have to find the balance between relevance and safety: on one hand, the malware used should represent the state of the art in offensive technologies; on the other hand, this malware must be under control during the lab exercises, so that studying it does not accidentally cause damage. We use the following strategies: (1) employ virtual machines (VMs) to run the malware so that we can avoid damaging the local system (i.e., the malware does not run in a real computer with actual user data); (2) isolate the lab VM from the campus network, so that the malware cannot attack and propagate to the campus network; (3) give the lab VM limited networking capabilities: network access is often necessary to observe the malware's interesting behaviors, however, it can also enable the malware to cause damage (e.g., attacking real hosts on the Internet). Therefore, we use fake DNS servers and network service simulators to give the malware an illusion that it can access the Internet to accomplish its missions. There may still be trade-offs if some malware requires an external C&C server but we cannot perfectly simulate the C&C server. In general, there are a few well-understood challenges in malware analysis, such as obfuscation, anti-debugging, and anti-virtualization, that demand more research. However, they are out of the scope of this paper.

We also strike a balance between relevance and feasibility. For example, there exist publicly available security datasets such as the Sandia Dataset [19] that are collected from real computer networks. However, directly using such datasets in a hands-on lab may not be feasible because such datasets are too complicated for a student to analyze in a few weeks, for example, the Sandia Dataset includes more than one billion network events. Therefore, we must use a trimmed down version of the original dataset so that analyzing it becomes doable in a student project.

If we do not find the right balances discussed above, either students are not able to learn the state-of-the-art of cyber threats, or they do learn but at the risk of spreading malware. Furthermore, if the labs take too much of the students' time, some may be frustrated and give up. On the other hand, if the labs are too easy, the students will not be sufficiently challenged. To minimize such risks, our solution incorporates virtualization-based lab technology that represents the state of the art in threat analysis; we also make our labs customizable in terms of the amount of security data, so that we can revise the labs based on students' feedback.

979-8-6425-4896-7/20/$26.00 ©2020 CISSE

C. Threat Detection and Analysis

1) Brief Introduction

Our labs in this topic area help a student learn how to detect active and dormant malware (either on disk or in memory), analyze its activities, assess its impact, and minimize its damage. They cover a skill set that includes incident detection, malicious code analysis, memory forensic analysis, and security data analysis. These skills can be mapped to the following CAE-C Knowledge Units: Digital Forensics, Network Forensics, Software Security Analysis, and Software Reverse Engineering, and the CAE classification can be Protect and Defend.

2) Hands-on Assignment

We create hands-on exercises that represent real-world threat scenarios in an enterprise environment. Each hands-on exercise covers a set of threat hunting skills that are needed to deal with a representative malware. The exercise is created by installing representative malware into a lab environment (i.e., a virtual machine). A student uses the lab virtual machine as the starting point of investigation, and his or her task is to uncover what has happened and submit a detailed report, without any knowledge of the particular malware installed in the exercise. From each exercise, the student is expected to employ a set of skills to "solve the puzzle".

Our virtualization-based lab design is a viable solution because we have used the same technology in our recent teaching at UNC Charlotte (e.g., in a course ITIS 6330/8330 Malware Analysis).

As an illustration, a student may use Process Explorer to inspect all running processes. If he/she recognizes a suspicious process from its name, he/she can find out the corresponding executable file on disk and perform static analysis; he/she may also perform dynamic analysis on a suspicious process. Lab T1 in Table I provides an example of this kind of easy projects. There are multiple static analysis tools available, such as PEiD that can detect whether an executable file is packed, Dependency Walker that can show all imported functions by the executable file, CFF Explorer that can show all parts of the executable file, and IDA that can disassemble the executable file. There are also multiple dynamic analysis tools, such as OllyDbg and Windbg that can debug the suspicious process at instruction level, Process Monitor that can monitor all library calls (e.g., Win32 APIs) made by the suspicious process, Process Explorer that can show many runtime attributes of the suspicious process (such as strings in memory, DLLs loaded, and objects created), Regshot that can detect any changes to the Windows Registry during malware execution, ApateDNS that can catch all DNS requests from the suspicious process, and Wireshark that can capture all network packets. We install all these analysis tools in the lab environment to make them readily available to the student.

The above scenario represents an easy case for identifying malware execution (i.e., malware does not obscure its process name). In reality, there will be harder cases that require different approaches for detection. For example, the full path of malware executable can be detected if it uses the registry key

HKEY_LOCAL_MACHINE\SOFTWARE
\Microsoft\Windows\CurrentVersion\Run

to achieve persistence. As another example, a malware that runs as a service may be detected if the service has a suspicious description or the DLL that implements the service has a suspicious name. We develop exercises that expose different scenarios to the student. Lab T2 in Table 1 provides a concrete example project with medium difficulty level.

Even stealthier malware (such as Lab T3 in Table I) can hide malicious files/processes/network connections from user-level investigation tools such as Process Explorer. In that case, a student may analyze the memory in order to detect hidden objects. For example, the student can use Windbg to uncover hidden malicious processes, hidden network connections, hidden DLLs, hidden malicious services, hidden device drivers (kernel modules), and malicious code injected into benign processes (e.g., through process hollowing). This method is especially useful for detecting malware's kernel-level activities, such as hooking of the System Service Descriptor Table (SSDT) and legitimate device drivers' IRP function tables.

TABLE I. EXAMPLE LABS IN THREAT DETECTION AND ANALYSIS

Lab Name	Difficulty Level	Description
Lab T1	Low	We install a malware file ocl.exe (MD5: 251f4d0caf6eadae4534 88f9c9c0ea95) in the lab VM. Using Task Manager, a student can see a process named ocl.exe, then she can use Process Explorer to locate the executable file on the disk. Next, she can use ApateDNS to see that the malware connects to an external server every 30 seconds. Using Wireshark, she can know that the malware tries to connect to port 9999, and if she runs Netcat on an external host listening on port 9999 and configures ApateDNS accordingly, she can see that the malware starts a reverse shell once the connection is successful.
Lab T2	Medium	We install a keylogger (MD5: 24ce99418862cb0c04e46fba24 5596ab) in the lab VM. This malware disguises itself under an innocuous name javaw.exe, records keystrokes and saves them in a file, persists over reboot, contacts a C&C server at total-updates.com, and acts upon several commands (such as "Update", "Upload KeyLogs"). A student can use multiple analysis tools such as ApateDNS, Wireshark, System Monitor, Process Explorer, Process Monitor, and OllyDbg to discover and analyze this malware. The student also can write a Python program to

Lab Name	Difficulty Level	Description
		simulate the C&C server. Further details of the analysis steps are shown in Table II.
Lab T3	High	We install one worm (MD5: a230a1bd2a8446e5d0a91e7dde44 c29f) from the Storm family to prepare the lab VM. This worm employs rootkit technology to hide malicious files, registry entries, and processes. A student can use Windbg to analyze the memory of the given VM. She needs to detect the modifications of two entries in the SSDT (NtQueryDirectoryFile and NtEnumerateValueKey) by the malware and discover the logic of the rootkit (e.g., what kind of files and registry entries it hides). She also needs to find the malicious device driver file and analyze it (using IDA and Windbg) to discover more details of the malware, such as which processes are running stealthily and which security products the malware tries to prevent from running.

TABLE II. ANALYSIS STEPS FOR LAB T2

Tool	Student Action	Observation
Process Explorer	Inspect process names	No process with a suspicious name
ApateDNS	Configure the tool to resolve any domain name to the host's IP address	Periodic requests for domain total-updates.com
Wireshark	Capture traffic	Periodic TCP SYN packets to the host's IP address on port 80, without TCP SYN-ACK packets from the host
Netcat on the host	Listen on port 80	A HTTP POST message is received, which includes the username, hostname, and group name of the analysis VM
System Monitor (sysmon)	Enable network monitoring	One process makes a network connection to the host IP address on port 80, and the process name is javaw.exe
Process Explorer	Find the start location of javaw.exe	At the location there is another file named Log.txt, and its content is logged keystrokes, such as "dir [enter]"
Process Monitor (procmon)	Trace the API calls made by javaw.exe	Confirm that javaw.exe invokes WriteFile ("Log.txt")
OllyDbg	Attach to javaw.exe, set breakpoint at WriteFile, and use the call stack to locate malware code that makes such calls	Confirm how javaw.exe logs keystrokes and saves them in Log.txt
OllyDbg	Set breakpoint at InternetOpenA and use the call stack to figure out where in the malware such APIs are invoked	Confirm how javaw.exe contacts the host at port 80 and how the reply from the host affects its execution, and find out the commands that the malware understands, such as "Update", "Upload KeyLogs"
Python scripts on the host	Develop a Python based web server that responds with "Update", "Upload KeyLogs", etc	Confirm that when the server replies "Upload KeyLogs", the malware sends encoded content of Log.txt. Confirm effects of other commands

The student can also perform event log analysis to hunt for malware, such as tracking malware installations, recognizing suspicious services, and finding evidence of malware execution. A student can first apply regular expression based filtering of event logs to reduce the amount of entries to be inspected, and then use heuristics to identify suspicious processes (such as a common system process name that is misspelled) and intrusions (such as many crashes of Adobe Reader and alerts from anti-virus). Next, the student can apply more advanced techniques such as timeline analysis and data visualization to recognize higher level malware semantics such as cyber kill chain phases (e.g., lateral movement) through patterns of events. The second area of our project focuses on the training of Security Data Analytics skills (see Section II.D).

3) Assessment

The student needs to submit a report of discoveries for each lab. The report is graded based on the completeness and clarity of the submission. In terms of completeness, the reported findings are checked against the ground truth, i.e., how far it is towards detecting and understanding the malware used to create the exercise. A typical report would cover (1) when (and how) the breach occurred, (2) rogue processes, application code injections, or persistent rootkits involved, and (3) activity, impact and capability of malware involved. In terms of clarity, the report must mention the tools/methods that are used in the hunting to glean each piece of discovered information. Depending on the nature of the exercise, the report can be centered around a few questions given in the exercise. Each lab exercise will have detailed rubrics for grading.

If the lab exercise is used in a competition, then the time that a student uses to work out the solution can also be a grading factor.

D. Security Data Analytics

a) Brief Introduction

Successful cyber threat hunting takes cyber threat intelligence, various logs, packet captures, and alerts from traditional IDS/IPS and firewalls as input to find threats and anomalies within the organization's networks and systems. Therefore, the ability to analyze security-related data is essential for cyber threat hunting. Our security data analytics labs cover sets of analytical skills to search and probe for anomalies in a variety of datasets, such as basic search, statistical analysis, aggregation, machine learning, data mining, and data visualization.

4) Hands-on Learning Assignment

We give students a number of log data sets with a description of the problem scenario. We also give them the necessary data analysis tools. They are expected to analyze log data to detect malicious activities in the system.

Depending on the difficulty levels of the assignments, we may provide additional information in the lab. For example, we can directly give the baseline profile of a system (i.e., when there are no intrusions), so the students can skip the step that learns the baseline profile.

Example scenario. As an illustration of our design of the lab exercises, we use the following scenario. C0mp@ny is a medium sized company with its headquarter located in Charlotte, North Carolina, USA. It has its offshore offices in Paris, London and Luxembourg. The Charlotte office employs around 100 employees. The company has four departments: Human Resource (HR), Research, Information Technology (IT), and Finance. On every work-day each employee logs onto their office machine. Employees can log on to their account either from home or office using the proper credentials and a secure connection. They can access documents shared with them or documents they have been given authorized access. They can use devices (e.g., printer, fax, and telephone) and other company resources available to them. After working hours, they need to log out of the machines. The system keeps the logs of login/logout times, actions performed, accessed devices, and GPS coordinates from where the employee is logged on. A small snapshot of the system log data is shown in Fig. 2. Every entry contains the date, time-stamp, employee ID, employee nickname, action, resource, IP, location latitude, location longitude and the operation status. This kind of log data will be given to the lab participants.

6/1/15	10100	Adara	8:46	login	AuthServer	192.168.1.2	35.0114253	-80.807878	success
6/1/15	10174	Quinlan	8:46	access	Server15	192.168.1.6(	35.228719	-80.82172	success
6/1/15	10165	Megan	8:48	login	AuthServer	192.168.1.51	35.228719	-80.82172	success
6/1/15	10120	Castor	8:49	login	AuthServer	192.168.1.15	35.228719	-80.82172	success
6/1/15	10178	Sacha	8:51	login	AuthServer	192.168.1.5(	35.1656996	-80.943506	success
6/1/15	10192	Wanda	8:51	login	AuthServer	192.168.1.71	35.228719	-80.82172	success
6/1/15	10104	Amanda	8:52	access	Server16	192.168.1.8	35.228719	-80.82172	success
6/1/15	10116	Bryar	8:53	print	Printer7	192.168.1.6	35.228719	-80.82172	success

Fig. 2. A Snapshot of the System Log File

Example Lab S1: detect anomalies regarding access time. Difficulty level: Low. The focus of this lab is access time (i.e., the period when an employee logs into the system). Students will analyze the employee activity log to detect incidents in which a previous employee tries to access company resources after he/she has left the company. The students are given an employee detail table (which indicates the starting and ending dates of each employee) and system log data such as the one in Fig. 2. The student is expected to write programs (e.g., in Python) to detect the anomaly and this lab provides the programming environment for Python and any other needed programming languages.

Example Lab S2: detect access location anomaly. Difficulty level: Medium/High. In this lab, students identify anomalous login locations from the given dataset. An employee can log in from within a radius of 10 miles from home or office in the headquarter (HQ), Paris, London, or Luxembourg; any other login locations are suspicious. This exercise gives the student system log data such as the one in Fig. 2, together with employee information (such as name, home address, office location, and typical routine). Students are expected to use tools like Google Maps to generate a map of access attempts using the GPS Coordinates in the log, add office locations and employees' home locations on the map, and finally search for any access attempts that deviate from the norm for the individual. In our example scenario, an access from Eastern Asia should be detected as abnormal, assuming that no employee lives in or travels to that area. Since the amount of data could be large (with tens of thousands of log entries), the student is expected to write a program using the Google Maps API in order to automate the data handling and anomaly detection.

5) Datasets for the Learning Assignments

In order for the Data Analytics labs to be meaningful, we must have realistic security data. We develop tools that generate security related data (such as access logs) based on different data models. For example, we generated the system log entries in Fig. 2 based on the example scenario about C0mp@ny, such as the number of employees and the branch offices; specifically, when generating the locations (office or home) from which an employee accesses the company network, we model the probability distribution of "working from home" as the Normal distribution with mean 0.5 and standard deviation 0.287. More details of our data generation process can be found in [20].

6) Assessment

Students need to submit a report of discoveries. The report is graded based on the completeness and clarity of the submission. In terms of completeness, the reported findings is checked against the ground truth, i.e., how far it is towards detecting and understanding the anomaly or attack(s) reflected in the given security data. A typical report would cover (1) efforts to validate data to ensure only reliable data is used in the analysis, (2) when the anomaly or attack(s) occurred, (3) the evidence for an anomaly, (4) rogue processes or other attack artifacts generated, if applicable, and (5) activity and impact of the attack(s), if applicable. In terms of clarity, the report must mention the tools/methods that are used in the hunting to glean each piece of discovered information. Depending on the nature of the exercise, the

report can be centered around a few questions given in the exercise. Each lab exercise has detailed rubrics for grading.

If the lab exercise is used in a competition, then the time that a student uses to work out the solution can also be a grading factor.

III. Implementation of the Threat Hunting Labs

We build and host hands-on labs on two dedicated servers, and these labs can be remotely accessed from inside a browser[1]. The dedicated servers are virtualized using VMWare, and each hands-on lab environment is contained in one virtual machine (VM). Each of our servers can host about ten lab environments (VMs), which means that each server can support ten students at once. Each lab environment is based on Ubuntu, which in turn uses VirtualBox to run different labs. Each lab environment also runs a FastX web server to support remote desktop access from inside a browser. We use LDAP to centrally manage user accounts among the lab environments. Since our dedicated servers are connected to the campus network, we configure firewall policies to ensure that these servers are logically isolated from the rest of the campus network. The lab VMs have private IP addresses, so they cannot be directly reached from the Internet; we use a Nginx proxy (which has a public IP address) in front of them to allow remote access to the lab VMs. A more detailed diagram of our lab environment is shown in Fig. 3.

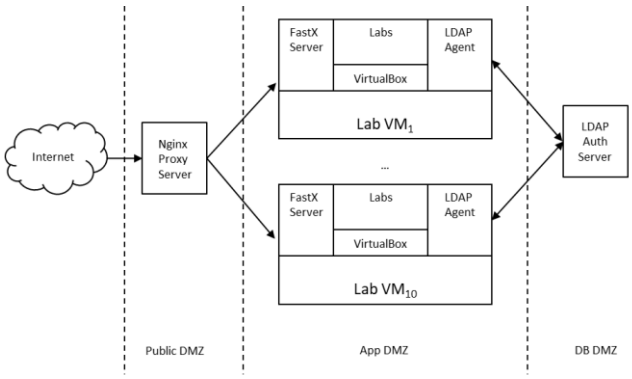

Fig. 3. The Topology of Our Lab Environment

Moreover, we provide a web portal[1] that allows participating students to request access to our labs. Upon receiving access requests, we create accounts for the students after a vetting process. Next a student can login to our lab environment from his/her browser. Once this is done, he/she can view the list of available labs, read lab manuals, and launch the labs on top of VirtualBox to finish the exercises. We set up the environment for each lab, including restoring the states of required virtual machines and configuring the network to interconnect the virtual machines. We also need to address lab time scheduling issues among students because our server can run only a limited number of lab VMs concurrently. We solve this problem by expiring each student account within a reasonable period that is long enough for the student to finish the lab exercise.

[1] The URL to our cyber hunting labs is https://sites.google.com/uncc.edu/cyberthreathunting/home

Each lab exercise is packaged in one or more VirtualBox virtual machines that (1) mimic the IT environment to be analyzed, and (2) have common analysis and development tools installed. Note that these virtual machines are nested VMs inside VMWare VMs, and they are the core of our online laboratory.

More specifically, the lab virtual machines (VirtualBox VMs) contain security analysis tools such as debuggers (e.g., OllyDbg and Windbg), disassemblers (e.g., IDA), basic static analysis tools (e.g., CFF Explorer, Dependency Walker, PEiD, PEview, UPX, Resource Hacker), basic dynamic analysis tools (e.g., Process Monitor, Process Explorer, Regshot, WinObj Object Manager, ApateDNS, Netcat, iNetSim, and NtTrace), packet sniffers (e.g., Wireshark), Sysinternals, and the ELK stack (Elasticsearch, Logstash, Kibana) for log data analysis [6].

The lab virtual machines also contain programming tools or environments such as gcc, Visual Studio, and NetBeans. Each tool is installed together with the operating system that it depends on. For example, if the lab requires iNetSim [21] that runs only in a Linux environment and more programming is needed then gcc can be installed in the same virtual machine for students who prefer to write C code. We avoid using commercial IDEs.

We prepare a student manual for each lab, which includes (1) description of the lab exercise and necessary background information for the lab, such as system configuration and the security policy; and (2) questions for the student to answer. We also provide an instructor manual for each lab, which includes the ground truth, answers to the lab questions and instructions with screenshots to follow in order to obtain those answers. Depending on the purpose of using the lab (e.g., course assignment or competition), the solution can be used differently. For example, when the lab is used as an assignment, the instructor can give a grade based on the student's answers to the lab questions, and the solution can be used by the student as a learning guide. When the lab is used in a competition, the student will need to write a lab report in addition to answering the questions, and the grading is based not only on the answers but also on how close the report is to the ground truth.

IV. Status of the Project and Future Work

We have created six labs and they are in the beta testing phase. Some labs (e.g., the Security Data Analytics labs) have been used internally in our classroom teaching, while others are to be evaluated. We plan to evaluate this project from the following aspects:

First, we plan to evaluate the quality of our course material design, such as whether the labs cover essential skills in threat hunting and whether the labs have appropriate difficulty levels. We will seek feedback from colleagues both inside and outside our universities and external experts in member companies of our NSF funded Industry-University Cooperative Research Center (IUCRC) in Configuration Analytics and Automation.

979-8-6425-4896-7/20/$26.00 ©2020 CISSE

Second, we will collect data on student learning outcomes. We plan to introduce the labs into existing courses and collect feedback from (1) students and faculty in our current programs in two participating institutions (UNC Charlotte and Forsyth Technical Community College), (2) students and faculty in other institutions (e.g., Purdue University and University of Tennessee at Chattanooga) who are interested in using our labs, and (3) cyber security professionals. We will use the students' performance data to evaluate the labs. For example, how easy or hard for the students to follow the lab instructions, how much the labs reinforced the knowledge taught in the classrooms, and how much the labs increase the students' knowledge and awareness of threat hunting.

V. Conclusion

We have described the design and implementation of freely-available, hands-on labs for cyber threat hunting education, built by the University of North Carolina at Charlotte and Forsyth Technical Community College. Commercial training materials for cyber threat hunting are expensive and thus not accessible to the general student population. Through our educational labs, we aim to help alleviate the severe shortage of cybersecurity professionals with advanced analysis skills for cyber threat hunting.

References

[1] https://community.hpe.com/t5/Protect-Your-Assets/Staffing-a-successful-cyber-threat-hunting-team-Part-1-Cyber/ba-p/6915902#.WK2ulfiJFaQ. Detailed white paper link: https://www.hpe.com/h20195/v2/GetPDF.aspx/4AA6-8216ENN.pdf

[2] https://community.hpe.com/t5/Protect-Your-Assets/Staffing-a-successful-cyber-threat-hunting-team-Part-2-How-cyber/ba-p/6921509#.WK2uqfiJFaQ

[3] Staffing a successful cyber threat hunting team, Part 3: skills to look for in a threat hunter. https://community.hpe.com/t5/Protect-Your-Assets/Staffing-a-successful-cyber-threat-hunting-team-Part-3-Skills-to/ba-p/6923381#.WK2uzviJFaQ

[4] SANS. Incident response and threat hunting curricula. https://www.sans.org/curricula/incident-response-and-threat-hunting

[5] FOR508: Advanced digital forensics, incident response, and threat hunting. SANS. https://www.sans.org/course/advanced-incident-response-threat-hunting-training

[6] https://github.com/comperiosearch/vagrant-elk-box

[7] Tim Bandos. Seek evil, and ye shall find: a guide to cyber threat hunting operations. https://digitalguardian.com/blog/seek-evil-and-ye-shall-find-guide-cyber-threat-hunting-operations

[8] Introduction to threat hunting teams. Federal Virtual Training Environment (FedVTE).

[9] https://www.infocyte.com/blog/2016/6/17/threat-hunting-fad-or-essential-cyber-security-tactic

[10] Peter Stephenson. Threat hunting and analysis training session. https://its.ny.gov/eiso/19th-annual-cyber-security-conference/InteractiveTrainingSessions

[11] HPE. Hunting today - using your existing technology to hunt for cyber threats.

[12] http://www.cis.syr.edu/~wedu/seed/

[13] Eric Cole. Threat hunting: open season on the adversary. SANS Institute InfoSec Reading Room. https://www.sans.org/reading-room/whitepapers/analyst/threat-hunting-open-season-adversary-36882

[14] SANS lethal threat hunting and incident response techniques. RSA Conference, February 2017.

[15] Destiny Bertucci. The rise of the threat hunter. InfoSecurity Magazine, February 2017. https://www.infosecurity-magazine.com/blogs/the-rise-of-the-threat-hunter/

[16] Data analytics training. https://focal-point.com/services/advisors/data-analytics/data-analytics-training

[17] What is threat hunting? the emerging focus in threat detection. https://digitalguardian.com/blog/what-threat-hunting-emerging-focus-threat-detection. Accessed on 27th March, 2017

[18] Wenliang Du, SEED: hands-on lab exercises for computer security education, in IEEE Security & Privacy, Vol. 9, No. 5, pp. 70-73, Sept.-Oct. 2011. doi: 10.1109/MSP.2011.139

[19] A. D. Kent, Comprehensive, multi-source cybersecurity events, Los Alamos National Laboratory, http://dx.doi.org/10.17021/1179829, 2015.

[20] Md Nazmus Sakib Miazi, Mir Mehedi Pritom, Mohamed Shehab, Bill Chu and Jinpeng Wei. The design of cyber threat hunting games: a case study, Proceedings of the 26th International Conference on Computer Communication and Networks (ICCCN 2017), July 31-Augus 3, 2017, Vancouver, Canada.

[21] INetSim: Internet services simulation suite. http://www.inetsim.org/requirements.html

[22] Bloom's taxonomy. https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy