Proposing An Educational Resource for Exploit Detection of Common Vulnerabilities via Packet Analysis

Omid Bodaghi
omid.bodaghi@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Tahera Fahimi
tahera.fahimi@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Amir Abbas Asgari
amirabbas.asgari@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

ABSTRACT

Packet analysis plays a crucial role in network security and is one of the must-have skills for network security engineers. It can be used for anomaly detection and analysis of the occurred attacks on almost all the systems that interact with each other over a computer network. In this project, we aim to generate some educational resources for students learning network security to help them gain experience in packet analysis and network vulnerability detection.

In this project, we select a vulnerable web server, named WebGoat, exploit all its injection and broken access control vulnerabilities, and meanwhile capture the transmitted packets. In addition, we exploit "MS08-067" vulnerability existing in Windows XP.

The final prepared materials consist of a set of captured network traffic files related to the discussed vulnerabilities, accompanied by a couple of worksheets that help students find how and where the attacks have happened. We expect the students to be able to detect the procedure of the performed attack in each pcap file, and hopefully, it could improve their skills in packet analysis and vulnerability detection.

1 INTRODUCTION

Network security is one of the most prominent areas of interest both to scholars and industry people. The importance of web application security is significantly increased due to the growing range of web services. Most web applications have unknown and critical bugs that may lead to security faults and make them vulnerable to attacks[2].

One of the effective approaches to address the security issues arising in large-scale and complex computer networks is network traffic analysis. The included techniques within this approach are becoming increasingly popular because of their inherent benefits, such as greater threat detection, automation, and scalability. From another point of view, network traffic analysis, and especially, packet analysis, could also be employed as a useful educational tool to teach security materials in a riddle-like interactive way.

Some web vulnerabilities are more exploited than others. Based on the open web application security project (OWASP) Top Ten [10] for 2021, we chose to investigate the following three network vulnerabilities: *broken access control*, *injection*, and *vulnerable and outdated components*, which are the 1st, 3rd, and 6th on the OWASP list, respectively.

In our proposed approach, we aim to provide students with some network packets captured during different scenarios where one end of the communication (i.e., the attacker) has successfully compromised something pertaining to the other side (i.e., the victim). Then we expect the students to locate the scope of the captured packets that are directly engaged with the attack. The scope may include: (i) where the attack has been initiated, (ii) how it has proceeded, and (iii) where it has finished. We also want them to be able to discuss the technical aspects of the attack and to determine the security asset(s) being compromised. We hope that the students could attain a clearer and more practical vision of some of the security notions and well-known vulnerabilities.

In Section 2, we review some former studies on the domain of network security. In Section 3, we express the main elements and tools we have utilized to develop our teaching materials. Section 4 includes a detailed description of the performed attacks and the captured traffic of each one. Section 5 discusses an overview of how to analyze the captured packets and expectations from the students. In Section 6, we briefly introduce the main artifacts of this study. Finally, Section 7 presents the conclusions and recommendation for future studies.

2 RELATED WORKS

There is a variety of educational resources and tools to educate people interested in network security, with various levels of experience. In this section, we will review the most significant and well-known resources. Then, we will concentrate on packet analysis resources to find the advantages of our project over them.

Books: Given the significance and high demand in network security, numerous books have been written to help readers understand the concepts of this field. Based on their methodology, books can be divided into two categories: theoretical books and experimental ones. As some examples in the first category, Stallings [13] and van Oorschot [14] concentrate on theoretical issues, detailed definitions of the protocols and the standards, cryptography usages in network systems, etc. But they do suffer from a lack of experimental exercises. Regarding the second group of the books, Rawat [12] and Sanders [11] aim to teach students experimental skills about the network security concepts like setting up and using firewalls and VPNs, collecting packets, etc. These books give students the opportunity to learn network security through experimentation. However, due to the complexity and vast range of topics, which cannot be covered in a single semester and may be challenging for students, as well as the lack of specific prepared assignments and projects, these books might not be a good choice for the beginners.

Capture the Flag (CTF): CTF competitions [19] are fantastic resources for learning security concepts and gaining more experience. CTF is a type of contest consisting of a set of questions where people try to find a flag using their knowledge of security. While CTF provides a setting for people to compete with each other on designed challenges, it cannot be a suitable option for university course assignments. Typically, the questions are designed for professionals and require creativity. In addition, questions are often not limited to a single topic, so it is not a practical tool for teaching network security to the student in an organized manner.

Intentionally vulnerable web applications: To become a security engineer, students must deal with real-world scenarios and be able to freely test various attacks on a platform without any legal concerns. As a result, several purposefully vulnerable web applications have been developed to allow beginners to practice attacking them. Some of the most well-known ones are WebGoat created in Java, and Damn Vulnerable Web Application (DVWA) created in PHP [15]. These two web applications contain various web vulnerabilities to teach students about network security, and there are some websites [16] to explain the vulnerabilities and attacks. We used an intentionally vulnerable web application in our project to generate packets during the attack procedure; however, our teaching approach is based on packet analysis, and not exploiting vulnerabilities.

Packet analysis: There are some resources about packet analysis in CTF contests that could be useful for beginners [6, 17]. In addition, there are some packet analysis assignments from universities around the world [7–9]. These materials, however, only provide a brief explanation of packet analysis. Additionally, a lot of them only attempt to teach network security protocols without concentration on attacks and vulnerabilities.

The lack of comprehensive educational resources for pcap [18] analysis of real-case attacks motivates us to prepare such resources. In this project, we first try to attack a web server and record the transmitted packets. Then, we provide the students with a pcap file containing the captured packets and ask them to examine the packets to determine the precise steps taken during the attack. With this approach, they can obtain knowledge of packet analysis while also learning details about exploiting vulnerabilities like injection and broken access control. More importantly, some of the selected topics include more than one pcap file for analysis, which enables students to view security from different perspectives, identify various controls against threats, and learn diverse strategies to decrease the chance of a successful attack.

3 METHODOLOGY

In this section, we discuss the outline of what is expected from the students as the main goals of the designated assignments. Moreover, we describe the tools and techniques we have used to perform our attacks against the victim, and record and save the transmitted data.

3.1 Assignment goals

Each assignment consists of a couple of pre-captured network packet sequences, each saved as a pcap file. These pcap files are accompanied by some documentation to help the students start engaging with each assignment. We provide a brief description for both groups about the category of attacks pertaining to each assignment.

There are two main aims designed for the students to achieve within each assignment. The first is to locate the packets directly related to the occurred attack. The second goal is to have the students explain the technical details, what the adversary's exact idea was, how they employed that idea in a successful attack, and what asset(s) of the victim was compromised (e.g., some sort of data leakage or unauthorized access to some resources). Each compromise can relate to one or more sides of the famous security triad, i.e., confidentiality, integrity, and availability (aka CIA).

We have prepared our educational materials for two different levels of elementary and intermediate students, and the activities expected from each group are a little different.

For the elementary level, we assume the target audience has a basic knowledge of security notions but has neither a prior experience with packet analyzing tools nor deep knowledge of different well-known types of security threats and vulnerabilities. Hence, we provide them with a step-by-step help guide consisting of little questions/quests for this group to help them trace the captured packets and accomplish the goals. In addition, this group of students faces a more refined version of the saved packet sets where most of the irrelevant packets (which could be confusing for this audience) have been removed from the pcap files. So, one of the two general expectations we discussed before, i.e., locating the packets directly related to the attack, is almost already solved in this version of the assignments.

Regarding the intermediate version, unlike the elementary one, we provide the students with all the captured packets without any prior sanitization. Also, there are no step-by-step guidelines for solving each assignment. So, the students only have the raw pcap files and a brief general description of the relevant category of attacks.

3.2 WebGoat

The victim side of all the attacks (except for the vulnerable components assignment) is a web application called *WebGoat* [5] that is being run as a web server on the localhost of a virtual machine during the experiments. WebGoat is an intentionally vulnerable web application developed for educational purposes, which includes a categorized drop-down menu of famous vulnerabilities. Each category consists of a few subcategories, and by selecting a subcategory, the user confronts a sequence of numbered web pages. Each page either consists of an interactive task that the user should accomplish (the application itself states whether it has been a successful trial or not) or merely some descriptions.

Motivated by its easy deployment, simple user interface, and well-formed structure discriminated by different categories and subcategories of web vulnerabilities, we decided to choose WebGoat as the main platform to produce our teaching materials.

After selecting the topic of each assignment from the pool of WebGoat subsections, the basic procedure for generating the result pcap file is as follows:

- First, we execute tcpdump command *tcpdump -w <file_name >.pcap -i <interface_name>* to start capturing packets transmitted over the target network interface;
- Second, we exploit the intended vulnerability as per the design of WebGoat;
- Before terminating the capture process of tcpdump in the terminal, we usually perform some miscellaneous actions so that the raw pcap file could be a little more challenging to analyze for the intermediate level students with whom these files are directly provided.

- Finally, after capturing the packets, we double-check the saved files via Wireshark to ensure that the data has been recorded correctly.

Regarding the vulnerable components assignment, we did not use the corresponding section of the WebGoat because of the lesson structure. More precisely, by merely investigating the attack packets, students are not able to figure out the procedure of the attack and need additional resources like source code, Docker images, etc. So, we decided to follow a different scenario for this threat, which is discussed in more detail in Section 4.3.

3.3 tcpdump

In this project, we need a secure and simple tool to capture network traffic. tcpdump is an open-source command-line tool available for most Unix-like operating systems (including Linux, Solaris, OpenBSD, etc.) to read, save, and analyze the data packets transmitted over network interfaces.

It is worth mentioning that capturing packets on some interfaces requires superuser privilege in the Linux operating system. Because of security concerns, we prefer to run a program with as few components as possible under root privilege. So, tcpdump seems a better option than some other ones having a graphical user interface (GUI) such as Wireshark which, of course, looks more user-friendly.

3.4 Wireshark

Wireshark is another open-source and free network packet analyzer, which, in opposition to tcpdump, is a cross-platform one running through a GUI [3].

Although we talked about the security disadvantages of running tools like Wireshark to capture packets from the network interfaces, we still use this software for another purpose. Actually, after capturing and saving the network packets sent and received over the target interface, we use Wireshark to verify if the packets were captured and saved appropriately. We also use this software to modify the saved packets for difficulty adjustment of the exercises as discussed in more details in Section 6.

4 EXPLOITING THE VULNERABILITIES

This section is organized based on the major vulnerability categories we have investigated, i.e., injection, broken access control, and vulnerable components. Each subsection starts with a brief description of the vulnerability category and continues with the details of exploits.

4.1 Injection

Injection is the 3rd vulnerability in OWASP Top Ten for 2021. In this vulnerability category, the adversary injects some "code" in a request that expects to only receive "data" from the user. This enables the attacker to perform malicious actions caused by the lack of input validation and parameterized calls on the server side which leads to interpreting received data as code. Therefore, the attacker can access, change, or corrupt data in an unauthorized way.

A1_1.pcap:
(i) A SELECT query is intended to retrieve the information of a specific user whose last name is received from the client as a string and is directly inserted between the start and the end of the formatted query. This dynamic query, i.e., a query generated at runtime, is vulnerable to an SQL injection.
(ii) As per the exercise goal, entering the input as *<some_alphanumerics>' OR '1'='1* causes the server to return the data of all users saved on its database rather than only a single target user.
A1_2.pcap:
(i) A two-field form has been designed that takes an employee's name and Transaction Authentication Number (TAN) to show their data from a table including all employees' records.
(ii) The subtle difference between this task and the previous one is the use of the AND operator between the two input values in the query. So, to exploit the server's flaw in this scenario, the first input value does not matter (e.g., just some alphanumeric characters) but the second one is in the form of *<some_alphanumerics>' OR '1'='1';--*. The trailing part "--" comments out the part of the query after that, totally preventing it from being interpreted as part of the SQL command.
(iii) This way, the information of all the employees is shown to the malicious client which means the confidentiality is compromised.
A1_3.pcap:
(i) The input fields that the WebGoat client should feed to the server are identical to the previous task. Although the exploit idea is similar (i.e., SQL command chaining), the attacker's target is to tamper with the saved contents of the database.
(ii) The first input value does not matter. The second field "auth_tan" is the one to which we feed an input like

$$\begin{aligned}&< some\_alphanumerics >'; \; UPDATE \; employees\\ &SET \; salary = 1 + (SELECT \; MAX(salary)\\ &FROM \; employees)\\ &WHERE \; last\_name =' < the\_attacker\_username >'; --.\end{aligned}$$

(iii) The described injected SQL command sets the malicious user's salary as one unit more than the maximum existing salary, ensuring their salary would be more than any other one after the command is run. Hence, it is clear that the integrity of the database is violated.
A1_4.pcap:
(i) There is a single input field by which the client can search a phrase among the system's logs.
(ii) Apart from the described legitimate usage of the provided input field, we feed the field with a value such as *<some_alphanumerics>'; DELETE * FROM access_log; --*, which completely clears all the recorded data from the table containing the system's logs.
(iii) Therefore, the adversary can clear every left traces, meaning that they have successfully disabled an availability aspect of the system.
A1_5.pcap:
(i) The path toward finding the actual vulnerability of the web server is trickier here. There is an actual double-tab panel for login and registration. The educational goal is to illegally log in as a specific user account from which we only know the username.

Through a couple of trials and errors with the login panel, it seems that it is not vulnerable to a SQL injection. So, we try our

chance with the register panel which takes a username, an email address, a password, and its confirmation to create an account for the new user. After trying some different combinations of logical operators with conditional statements, we can indirectly infer knowledge from the server's response to our boolean queries. More precisely, when the server returns the message "User <our_query_contents> created, please proceed to the login page.", it means the query has been resolved to a False value. The server *thinks* there is no current entry with the same username as the client's input. On the other hand, when the server responds with a message like "User {0} exists, please try to register with a different username.", means that the query value has been computed as True, unlike the previous case.
(ii) Now, the attacker attempts to find out whether the letter at position i of the victim user's password matches their guess. To this aim, the attacker can brute force the password step by step via a simple and short Python snippet that sequentially sends HTTP requests to the server to crack the target password character by character. This reduces the exponential search space of all the possible values to a linear one.
(iii) So, the script cracks the password in just a couple of seconds. When the i-th character of the password is successfully hacked, a new row of queries starts to be sent for $i + 1$. The final terminating condition is when no character (among all the ones we know the password consists of) matches the n-th position of the password, which implies the password is actually of length $n - 1$. Then, we simply enter the cracked password into the login panel along with the victim's username and impersonate them.
A1_6.pcap:
(i) To check for SQL injection vulnerabilities, we enter the input *<some_alphanumerics>'; SELECT * FROM user_system_data;--*. We are unable to attack the server with the above input due to the server's response "*Using space is not allowed!*"
(ii) Now we need to determine whether or not we can insert malicious data without using any space characters. To rewrite the above query with no spaces, we create another one like *<some_alphanumerics>';/**/SELECT/**/*/**/FROM/**/user_system_data;--* . Note that "/*" and "*/" indicate the boundaries of a SQL comment. By providing this string as an input, we can successfully retrieve the users' accounts. This compromises confidentiality of the system and gives the attacker access to the unauthorized data.
A1_7.pcap:
(i) First, we enter *<some_alphanumerics>'; SELECT * FROM user_system_data;--* input exactly like the first step of the previous task. We can see that spaces are not allowed here.
(ii) To see if we can pass the input validation control, we examine the input *<some_alphanumerics>';/**/SELECT/**/- */**/FROM/**/user_system_data;—-*. According to the server's result, the server has removed SELECT and FROM phrases from the provided input. It may indicate that the server has changed its input validation approach in comparison to the previous task.
(iii) Now, we enter *<some_alphanumerics>';/**/SESELECTLECT/**/*/**/FRFROMOM/**/user_system_data;--*. In fact, we embed a SELECT in a SELECT string and also do the same for FROM. Hence, after removing one SELECT and one FROM, there would still be another pair of untouched SELECT and FROM. By submitting this input, we successfully attack the web server. We can conclude that the server's strategy here is to check the input and remove keywords like SELECT and FROM (but not recursively).
A1_8.pcap:
(i) We want to retrieve the first portion of the "webgoat-prd" server's IP address. To identify the vulnerability, we first engage with the form. As we check, there is no SQL injection vulnerability regarding the input field.
(ii) Now, we should check other features of the application by observing the requests and responses. Therefore, we try the browser's "Inspect" feature and go to the "Network" tab. Here, we check the HTTP requests and responses to analyze the server's functionality. We observe that to sort a column like "mac" the generated request is *http://127.0.0.1:8000/WebGoat/SqlInjectionMitigations-/servers?column=mac*. Most likely, the server sorts the records using a SQL query similar to *SELECT * FROM <table_name> ORDER BY <column_name>*.
(iii) Additionally, we are aware that CASE WHEN can be used inside of an ORDER BY clause in SQL, as in the example *SELECT * FROM stores ORDER BY CASE WHEN <condition> THEN <column_name_1> ELSE <column_name_2> END;*. Here, we rank the columns based on <column_name_1> if the criteria are met, and based on <column_name_2> otherwise. With this information, we can take advantage of the server. When we enter an invalid column for a test (such as "ABC") an error occurs. Fortunately, based on the server's error message response we can figure out the target table name is "server", and the query to sort columns matches with our guess in step (ii).
(iv) Now, the attack plan is to specify a condition in "CASE WHEN" and then check the outcome. We can infer that a specific condition is true if it is ordered by <column_name_1>, and is flase otherwise. The query can therefore be *(SELECT substring(ip, 1, 1) FROM servers WHERE hostname = "webgoat-prd") = 1*. By observing how the result is sorted, we can determine whether or not the first digit of the IP address is 1. So, we can iterate digit by digit over possible IP addresses and find the correct one based on the sort results. The process is continued and the ultimate result is 104.

In all the last three attack, the confidentiality of the system is compromised.

4.2 Broken access control

Broken access control is ranked as the first vulnerability based on the OWASP Top Ten for 2021. Access control enforces a policy such that users cannot act outside of their intended permissions. In other words, different users in a system have different access levels. For example, an administrator can view all users' information while a regular user can only view their own information. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limitations. In this attack, we will log in with the username "tom" which belongs to an authenticated user in the system. However, by using the privileges of this user, one can find some different pieces of information that are not supposed to be accessed by them. Also, the user finds other users' IDs by checking the JavaScript code running on the webpage, so they can query for any other users and view their information.

As it is shown in the WebGoat, the attacker logs in with the provided username and password (username is "tom" and password is "cat"). At first, nothing is suspicious but when he opens the JavaScript codes from the browser's "Inspect" menu, he can see some hidden data features within the body of the response. The user can find his ID by uncommenting the hidden features while ID and any other hidden information should not be viewed by any user. So, the user launches a brute-force attack on ID field in order to gather other users' information.

4.3 Vulnerable components

Vulnerable components refer to outdated pieces of software that have security issues and are not maintained correctly. These components can be part of a library, framework, or any other application. It is significant to update the code components to prevent security issues because they are used as building blocks of many other applications and a vulnerability in one component may also leave the system vulnerable. In this section, we chose a traditional vulnerability in Windows XP because it involves a high level of risk and is applicable to the current insecure systems to drop a shell. This attack is possible because of outdated components.

First, we need to have two virtual machines in our laboratory, one Kali Linux and one Windows XP. Because Windows XP has lots of other vulnerabilities, it is recommended to connect it to the network over a bridge connection or behind a Network Address Translation (NAT). In the real-world scenario, the attacker needs to find the victim's IP address which can be done with the help of Nmap. Nmap [4] can search the whole network and use a signature database to recognize some characteristics of devices like their operating systems. The attacker runs command *nmap -sV <attacker_IP>/<subnet_mask>* to find the victim's IP address and uses Metasploit project in Kali. The attack target is known as "MS08-067" vulnerability, which is a remote code execution [**?**]. An attacker who successfully exploits this vulnerability could gain a complete remote control of the affected system.

In order to conduct the exploitation, the attacker uses the Metasploit project and loads the module "MS08-067" by running commands *service metasploit start*, *msfconsole*, and *search ms08-067*, respectively. The name of the submodule used by the attacker is "exploit/windows/smb/ms08_067_netapi" that is loaded by the command *use exploit/windows/smb/ms08_067_netapi*. With *show options* command, the attacker can learn what the module needs as input to exploit. The mandatory fields are "RHOST", "LHOST", and "Payload", respectively, referring to the target Windows XP IP address, Kali Linux IP address, and any compatible payload available for this exploit. The attacker can set these values via the command *set <field> <value>*. Now with the command *exploit*, Kali uses the module "MS08-067" to drop a shell in Windows XP host. If Windows firewall is disabled (we have assumed it for simplicity), then there is almost no protection against this attack [1].

5 PACKET ANALYSIS

In this section, we take a glance at generated packets and discuss the main idea(s) of each pcap file. Additionally, we propose our expectations from the students based on the evidences in the packets.

5.1 Injection

In general, almost every injection attack involves a suspicious value fed to a request. For example, a special set of characters like ";", "--", etc. appeared in a first name input field looks weird in sense of the intended expectations of that field. Seeking for the contents of HTTP requests is generally useful to find clues about this type of attack since this is where the malicious client "injects" the unintended data to the server's back-end.

The exploit goals of tasks A1_1.pcap and A1_2.pcap are inherently the same. So, by checking the transmitted packets of these two tasks, the unusual input values containing an always-true conditional statement preceded by a logical "OR" might take the attention of the network analyzer. This may let they figure out that the client is actually trying to retrieve all the records within a specific table of the database.

The main idea behind the attacks done in tasks A1_3.pcap and A1_4.pcap is also similar to that of A1_2.pcap and it is basically the SQL command chaining. Looking at the captured network data available in these two files, it is expected from the students to respectively find the "UPDATE" and "DELETE" phrases within the saved HTTP requests sent to the server over the POST method. Obviously, these both look odd in the context of the usually expected input of either a TAN or a phrase to be searched among the previous logs.

Although the task of A1_5.pcap still lies within the scope of SQL injection, understanding the details of the attack seems to be trickier for the students. Because as discussed in Section 4, the performed exploit includes some heuristic trials and errors followed by an automated sequence of HTTP requests issued by a Python snippet. Therefore, the students should first look up the packets for the initial trial and errors to figure out what kind of vulnerability the attacker has found on the server side. Then, they face a series of similarly structured request sequences (where each sequence pertains to cracking one of the password characters). By carefully investigating the outline of these sequences, they are expected to understand that the attacker has been trying to brute-force the password letter by letter (and not all at once), which is the major idea behind this task.

In the process of discovering vulnerability and attacking of tasks A1_6.pcap and A1_7.pcap, students should identify the initial relevant packets. They may see that the user's initial HTTP POST request uses strange characters instead of a typical username. Then, they can examine the response to determine the sensitive information. This examination leads to the understanding of the server's defense plan and attacker's strategy. At the end of these tasks, we anticipate that the students would learn that using input validation without prepared statements is not enough.

In the captured file of A1_8.pcap task, the first packets related to the attack are not threatful at all, and the adversary uses them to figure out the server's functionality. Therefore, in order to determine the attack methodology, students should examine more packets and compare them to one another. They should also be familiar with advanced SQL queries, so some links and hints are included in the assignment file. Based on the boolean condition in CASE WHEN queries, the result will be sorted in two different ways, and the adversary can use this knowledge to retrieve data out of their access. Here, The adversary uses the "substr()" function as a part of the condition contained within the URL to obtain the IP address of the given server.

This vulnerability is different from the previous ones. Here, the server might use prepared statements, so we are unable to attack the server via an input form. After completing this assignment, students would comprehend that even if the server employs prepared statements, without an input validation system, it may still be susceptible to injection. As well, they should learn that usually they need to analyze a batch of packets, rather than a single packet, to find out the attack's strategy.

The last three tasks should teach students the necessity and importance of using prepared statements and input validation at the same time.

5.2 Broken access control

As we go through the packets in the pcap file, we find that a user is authenticated to the system using a GET request body. The authentication step is correct (both the request and response are acceptable). By exploring more packets, we can see the user is requesting different IDs through sending different GET requests, which is a malicious activity. For some ID values, the user could achieve other users' details from the responses.

5.3 Vulnerable components

Through investigating the pcap file, we find that a user was scanning the network. This activity is considered malicious unless the user is a network administrator.

We expect the students to go through the captured HTTP packets, and find the first packet with a suspicious payload, i.e., a payload included in "MS08-67" module from the Metasploit project. So the students can detect an attack happened on Windows XP that aims to exploit "MS08-067" vulnerability.

6 RESULTS

Generating a novel educational resource for students to analyze network traffic packets of an occurred attack is the main goal of this project. As an artifact, we have designed three assignments with one or more pcap files for each one. Now, we explain the main idea of each assignment in what follows:

Assignment 1: This assignment is about injection attacks. The first section of the guide file includes an introduction to SQL injection as well as some useful links about MySQL syntax. Then, we have eight subsections, each one including some questions and hints to help students analyze the packets regarding one pcap file. In this assignment, we have two concepts: SQL injection and SQL mitigation. The first five pcap files are about SQL injection and organized from easy to challenging. The remaining pcap files, which deal with SQL mitigation, are arranged in the ascending order of difficulty. By the end of this assignment, we anticipate students to get familiar with Wireshark, read data from packets, acquire a plausible knowledge of SQL injection, learn basic mitigation strategies, and obtain basic skills in packet analysis.

Assignment 2: This assignment is mainly about the broken access control and has a similar guide file to that of assignment 1. Additionally, we created a pcap file by exploiting broken access control section in WebGoat.

Assignment 3: In this assignment, instead of using WebGoat, we use Windows XP as a vulnerable system. Windows XP has a vulnerability called "MS08-067" that allows an attacker to drop a shell and attain administrator access.

Please take note that we have taken into account two levels of students: elementary and intermediate. For the first group, the pcap files are filtered to include only the relevant packets, and there are more hints in the guide file. But for the second group, the pcap files include all the captured packets and there might also be some minimal hints in the guide files.

7 CONCLUSION AND FUTURE DIRECTIONS

In this project, we developed a teaching resource that undergraduate and graduate students enrolled in courses related to network security can use as course assignments. This was necessary because of the lack of packet analysis exercises suitable for interested students and the importance of this skill in network security. These resources can teach students two crucial skills in network security, i.e., exploiting vulnerabilities and packet analysis. To generate packets, we select an intentionally vulnerable web application named WebGoat, and we exploited injection and broken access control vulnerabilities. In addition, a vulnerability in Windows XP, "MS08-067", is selected for the vulnerable components category. As the main artifacts of this project, we created three assignments, each including a guide document and some pcap files, that help students understand the procedure of the attacks by analyzing the captured packets for each category.

REFERENCES

[1] Barath. 2021. *Metasploit Basics for Beginners – Exploiting Windows XP (MS08–067) with Metasploit (Kali Linux) – Part 1.* https://www.getastra.com/blog/security-audit/how-to-hack-windows-xp-using-metasploit-kali-linux-ms08067/

[2] José Fonseca, Nuno Seixas, Marco Vieira, and Henrique Madeira. 2014. Analysis of Field Data on Web Security Vulnerabilities. *IEEE Transactions on Dependable and Secure Computing* 11, 2 (2014), 89–100. https://doi.org/10.1109/TDSC.2013.37

[3] Wireshark Foundation. 2022. *Wireshark.* https://www.wireshark.org/

[4] nmap organization Gordon Lyon. 1997. *nmap.* https://nmap.org/

[5] Michael Jones, John Bradley, and Nat Sakimura. 2015. WebGoat). RFC 7515. https://doi.org/10.17487/RFC7515

[6] Marcelle Lee. 2019. *CTF Challenge Walkthrough: Network Traffic Analysis, Rogue User.* https://marcellelee.medium.com/ctf-challenge-walkthrough-network-traffic-analysis-rogue-user-f87ae923130

[7] National Institute of Technology (NIT). [n. d.]. *Wireshark as a Network Protocol Analyzer.* http://lokeshchouhan.com/gallery/experiment2dcl.pdf

[8] University of Toronto. [n. d.]. *Wireshark Lab.* https://www.cs.toronto.edu/~ahchinaei/teaching/2016jan/csc358/Assignment1w.pdf

[9] University of West Indies. [n. d.]. *Packet Analysis Exercise.* https://comp3911.info/exercises/14/

[10] OWASP. 2021. *OWASP Top Ten.* https://owasp.org/www-project-top-ten/

[11] Danda B. Rawat. 2013. *Computer and Network Security: An Experimental Approach.* CreateSpace Independent Publishing Platform.

[12] Chris Sanders. 2013. *Applied Network Security Monitoring: Collection, Detection, and Analysis.* Syngress.

[13] William Stallings. 2016. *Network Security Essentials: Applications and Standards.* Pearson.

[14] Paul C. van Oorschot. 2021. *Computer Security and the Internet: Tools and Jewels from Malware to Bitcoin.* Springer Nature.

[15] Shubham Vashist. 2018. *Top 5 (deliberately) vulnerable web applications to practice your skills on.* https://resources.infosecinstitute.com/topic/top-5-deliberately-vulnerable-web-applications-to-practice-your-skills-on/#:~:text=Metasploitable%202%20%E2%80%93%20Metasploitable%202%20is,mainly%20used%20for%20network%20testing.

[16] Jan Verner. 2020. *OWASP WebGoat Solutions.* https://github.com/vernjan/webgoat

[17] welchbj. 2021. *PCAP Analysis*. https://github.com/welchbj/ctf/blob/master/docs/pcap.md
[18] Wikipedia. (Last visit: Dec 5, 2022). *PCAP*. https://en.wikipedia.org/wiki/Pcap
[19] zaratec. 2022. *CTF Practice*. https://zaratec.io/ctf-practice/