TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

Implementation of a Hands-on Attack and Defense Lab on Insecure Direct Object References

Master's thesis

Ephrem Getachew Demesa (165600IVCM)

Supervisor: Margus Ernits

Tallinn 2018

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Ephrem Getachew Demesa

13.12.2018

Abstract

This thesis aims to implement a web-based hands-on attack and defense lab for insecure direct object reference vulnerability that provides accessibility regardless of time and location constraints. In this work, a web-based grade management system with multiple IDOR vulnerabilities was implemented. Based on this web application, the lab exercises were created including the automated scripts to evaluate learners' solutions.

The methodology used to create hands-on practical course was ADDIE instructional design model. To design the course, the learning objectives were composed. Based on the learning objectives, the prototype was implemented along with the evaluation script, the necessary materials, evaluation methods, and delivery methods. Evaluation of the prototype was conducted by a selected matching pilot group and correction was made by incorporating their feedbacks. Additional evaluation and validation were made by students of the Web Application Security (I901) course from TalTech IT College. Subsequently, lab testing results, time duration and progress were monitored and feedbacks were collected. After validation was performed, the web-based hands-on attack and defense lab has been suggested and accepted by the Instructor to include it to the web application security (I901) course curriculum.

Finally, the web-based hands-on attack and defense lab was implemented and the result is publicly available as an open source contribution in GitHub (https://github.com/EphremG/ctf). The lab was designed in such a way that it can be applied for different projects and additional exercises. In addition, the process of adding new challenges was also included in the design of the prototype. Moreover, the prototype is integrated with the RangeForce system where all the necessary modules and evaluation scripts are readily incorporated. However, it is practically possible to use the prototype without the RangeForce system.

Testing of the prototype requires registration to RangeForce (www.rangeforce.com) followed by redeeming of the following promotion code: testi-idor2-pe4phohX.

Key words: attack and defense lab, IDOR, web application security

Annotatsioon Abstract in Estonian

Annotatsioon - Praktilise ründe ja kaitse labori loomine ebaturvalise objekti otseviite haavatavusele

Antud magistritöö eesmärgiks on luua praktiline, kaitse- ja ründesuunitlusega õppelabor ebaturvalise objekti otseviide (IDOR) nõrkuse õppeks, mis oleks juurdepääsetav sõltumatult ajast ja asukohast.

Autor teostas veebipõhise hinnete haldamise süsteemi, mis sisaldab mitmeid IDOR tüüpi nõrkuseid. Antud süsteemi baasil loodi praktilised laboratoorsed tööd, mis sisaldavad automaatset hindamist ja õppuri abistamist.

Praktilise õppelabori loomisel kasutati ADDIE õpidisaini mudelit. Määratleti õpiväljundid, mille alusel loodi prototüüplahendus koos automaatse hindamise ja tagasiside skriptidega ning õppematerjalid, hindamismeetodid. Töö käigus loodi labori arhitektuur ja kirjeldati meetodid, mille abil saab luua täiendavaid ülesandeid. Loodud prototüüplahenduse hindamiseks kasutati pilootgruppe ja nende tagasiside alusel muudeti laborit. Lõpliku hindamise viidi läbi õppeaine Veebirankenduste turvalisus (I901) raames TalTech IT Kolledžis. Labori testimise tulemusena mõõdeti õppesessiooni kestvust, õppuri edenemist ja koguti tagasiside. Labori hindamisel kaasati õppeaine õppejõud Andres Käver ja labor lisati õppeaine Veebirakenduste turvalisus (I901) laborite nimekirja.

Veebipõhine ründe- ja kaitseotrarbeline e-õppe labor on avalikult kättesaadav GitHub aadressilt (https://github.com/EphremG/ctf) kasutades vaba tarkvara litsentsi.

Loodud laborit saab taaskasutada ja laiendada, ning selleks on töös esitatud protsess ja juhised, mis võimaldavad luua uusi harjutusi.

Lisaks on labor integreeritud RangeForce süsteemi, kuigi loodu on kasutatav ka iseseisva üksusena. Labori testimiseks looge kasutaja RangeForce süsteemis (https://rangeforce.com) ja kasutage kupongi **testi-idor2-pe4phohX**

Võtmesõnad: ründe- ja kaitseotstarbeline labor, IDOR, veebirakenduste turvalisus

Acknowledgements

I would like to express my sincere gratitude to my supervisor Margus Ernits for his invaluable advice and guidance in this thesis work. I am also deeply indebted to Rolad Kaur for his support in the integration of the target system with the RangeForce platform.

Finally, I would like to acknowledge with gratitude the support and love of my parents and friends throughout this work and my whole life.

Abbreviations

API	Application Programming Interfaces
ASP	Active server pages
CTF	Capture the flag
Cx	Checkmarx enterprise tool
DVWA	Damn vulnerable web application
HTTP	Hypertext transfer protocol
IDOR	Insecure direct object references
JSP	Java server pages
OWASP	Open web application security project
PHP	PHP Hypertext preprocessor
RIPs	PHP security static code analysis scanner
SQL	Structured query language
VM	Virtual machine
VTA	Virtual teaching assistant
WAF	Web application firewall
XSS	Cross-site scripting
ZAP	Zed attack proxy

Table of Contents

List of Tables

List of Figures

1. Introduction

Web applications contain security bug that may lead to critical security breaches on the availability, integrity and confidentiality of a system. In a web application, the three essential security properties are input validation, integrity and correctness of the logic [1]. Failing to meet this requirements can cause vulnerability and successful exploitation of the web application.

The Open Web Application Security Project (OWASP) is a non-profitable organization which works to enhance the security of software [2].This thesis focuses on one of OWASP Top 10 2013 vulnerability list, namely Insecure Direct Object References (IDOR).

“IDOR occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files ”[3]. IDOR is one of OWASP top 10 security vulnerabilities next to SQLi, XSS and Broken authentication.

It is not uncommon for a software developer to expose a reference for the internal implementation object as URL or in the form of parameter [4]. If there is no proper access control check, a malicious user easily manipulate these references to gain or/and access unauthorized data. Thus, IDOR represents the absence of the authorization level checks.

This research will demonstrate the real impacts of IDOR vulnerability by implementing vulnerable web application using web technologies (HTML, PHP and MySQL). In this thesis, a web-based grade management system was implemented using PHP programing language and multiple IDOR vulnerabilities were included in the study. Different approach might be followed to create hands-on lab, however, in this thesis, hands-on lab challenges were created using CTF (Capture the Flag) approach. CTF is a standard term used to create competition between security professionals and/or students to learn about cyber security [5]. The purpose of the vulnerable system was to assess the performance of existing security tools in finding IDOR. Three existing source code analysis tools (SonarQube, RIPs, and Checkmarx) and blackbox testing tools (w3af, owasp zap and nikto) were applied in this study and scans were conducted accordingly. Publicly available similar CTF projects were selected for the study and the function and limitation of each existing automated solutions were observed.
Finally, the challenge was integrated with RangeForce cloud-based cyber security training platform, which provides virtual teaching assistance.

1.1 Motivation

Although many blackbox and whitebox security scanning tools are developed to find security vulnerabilities, no one came up with a complete solution to successfully find IDOR, as the pattern is different from one web application to another web application [4]. Hence, IDOR have been one of the major web application flaw and cause for security breaches. It is also possible to check its occurrence in many web application from different bug bounty reports which are disclosed in public and private bug bounty programs such as Hackerone, Facebook and Google [6]. In addition, nowadays frameworks such as Python Django come up with built-in mitigation techniques for XSS, SQL-injection and CSRF; however, it is not possible to solve IDOR issue by default or using any automated tools unless authorization is properly implemented during development.

On the other hand, awareness training through CTF is considered as a possible approach to practice finding and fixing such vulnerability. However, there is a luck of publicly available IDOR lab which contains both attack and defense exercises. Therefore, the purpose of this paper is to find a structured approach to design a prototype for hands-on attack and defense CTF challenges. In addition, it can exist in most of web application (PHP, ASP, JSP) and security tools are not a complete solution to detect or find this vulnerability.

1.2 Problem statement

The lack of publicly available practical IDOR attack and defense training platform indicates the need for better and effective training programs for academic purpose or for people who are studying and/or working in IT profession. In fact, there are some universities teaching web application security and plenty of existing best practice guidelines are in the internet which helps you to understand IDOR vulnerability and prevent the vulnerability. However, the problem in most cases are luck of practical scenarios. One good reason could be the challenge to teach practical exercises for a large number of students in a class. Therefore, an alternative practical web application security trainings that incorporate real case scenarios is necessary to solve such problems.

Existing intentionally vulnerable web application which are used for CTF challenges are not suitable to create different IDOR challenges. This is because, IDOR is a permission related vulnerability, and therefore, the vulnerable web application system must be designed in such a way that allows different users to access the system with different roles and privileges. In this regards, therefore, it is essential to design a vulnerable web application.

Although CTF challenge is universally acceptable in security training, only few exercises are available for IDOR vulnerability. In addition, most of IDOR challenges are designed only to improve penetration testing skills from the attacker point of view. However, practical defense challenges for IDOR are not included to the publicly available CTF exercises. In this context, open source defense exercises for IDOR would be a novel contribution.

1.3 Contribution

1.3.1 Research Design

This thesis work follows instructional design model to identify the outcomes of the instruction, to develop scope and sequence, and to establish effective evaluation. Out of different instructional design models, ADDIE instructional design model were used to develop hands-on practical exercise.

1.3.2 Research contribution

The first contribution was implementing a prototype of vulnerable web application system that later was used to create different practical lab exercises. The prototype was first scanned with the existing static source code analysis tools followed by blackbox security scanning tools. The results were analyzed and existing security solutions were evaluated.

The practical exercise is integrated with RangeForce system where cloud based virtual teaching assistant (VTA) are provided. RangeForce VTA provides better learning environment in which the author contributes the IDOR attack and defense lab which also includes the necessary reading materials, hints and evaluation scripts for automatic score point to mark when a given task is solved. Moreover, using RangeForce platform, the lab would be available without time and location constraint. In addition to RangeForce, it is also available as an open source in GitHub for a user who wants to download and run in their own local server.

1.3.3 Design evaluation

The prototype of this research work was evaluated in two phases. First, a pilot group who are experts in web application security provided their feedback about the training exercise content and its correctness. After collection of each feedback from the pilot group, it was possible to improve the system in a much better way. In the second phase, the learning outcome of the prototype was evaluated with TalTech IT college students, who are studying web application security course. The purpose of the second phase evaluation was to assess the knowledge gained by the participants.

1.4 Target audience

This thesis research is useful and applicable for IT students and professionals who are responsible for software development, quality assurance, secure software design, and penetration testing.

1.5 Scope

The scope of this thesis is limited to building a prototype for IDOR CTF challenges that is capable of running broken web application written in PHP language. This was accomplished by defining documentation, design, and a reference implementation. The defense exercises require learners with the basics of PHP programming language experience.

1.6 Thesis outline

This thesis work is categorized into six chapters. Chapter one discusses the overview of thesis including the problem statement, proposed solution and the author contribution in this research. Chapter two presents the literature background about IDOR. In Chapter three, the capability of existing blackbox security scanning tools and static code analysis tool to find IDOR flaw from the proposed vulnerable web application prototype is evaluated. Chapter four discusses the research design methodology including the analysis, design, development, implementation and evaluation of the hands-on lab. Chapter five discusses about the summary of the thesis work. Finally, the last chapter suggests for future work.

2. Literature

This section presents literature about IDOR, the meaning of it, a brief explanation on how it occurs, and some stories from companies who were attacked with this vulnerability and additional information.

2.1 Insecure DOR vulnerabilities

IDOR has been regarded as a serious web application vulnerability. Unlike other vulnerabilities such as XSS and SQL-Injection, identifying IDOR vulnerability is a bit challenging using automated tools. This is because to successfully attack using this flaw, we need to distinguish the flawed interface as well as the pattern to spot an insecure object. In order to identify an interface that provides access to sensitive contents, code review and website walk-through must be done [4].

Even though Insecure DOR is not a new vulnerability, the impact of this vulnerability is critical when it occurs. Insecure DOR is a permission-related problem and cannot be fixed automatically or by default as the permission use-cases vary from web application to web application [7]. Figure 1 illustrates IDOR vulnerability.

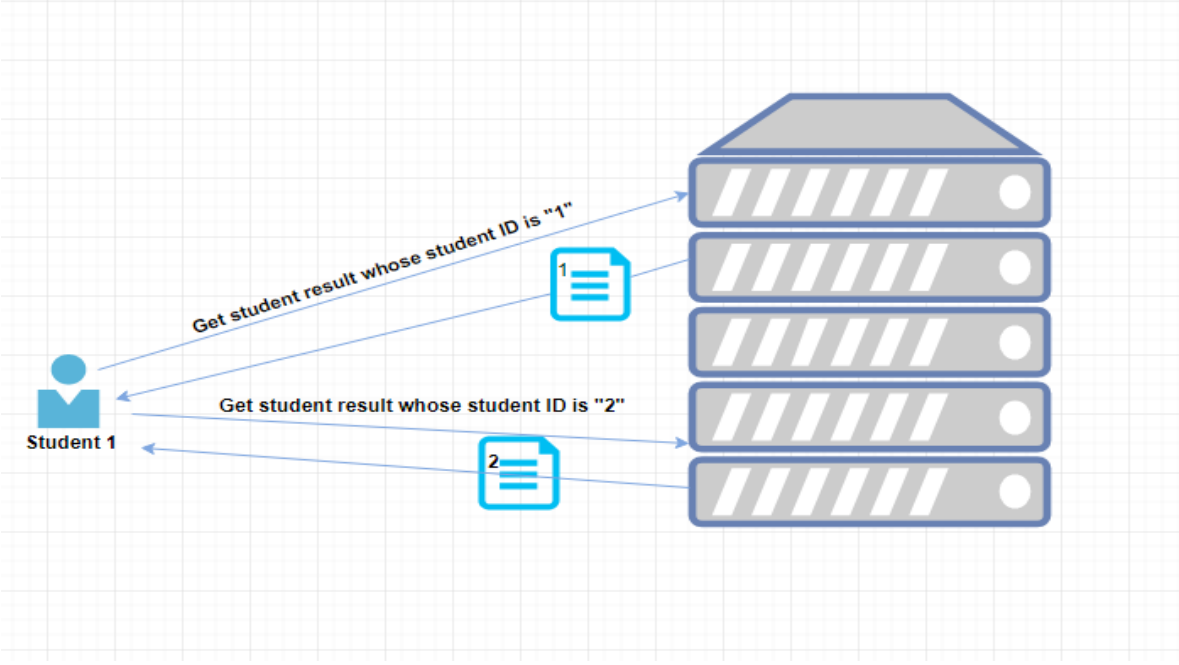

Figure 1: Example of IDOR vulnerability

This type of vulnerability has been disclosed several times. It is one of the most common publicly-disclosed security flaws [7]. IDOR vulnerability allows malicious user to access an

account that belongs to another user. The impact of IDOR vulnerability leads to account takeover, change-delete users´ data or access or modify private data [8].

IDOR has a major role in large security breaches and scandals worldwide. The following are a few cases worth mentioning:

- Recently US Security Researcher discovered IDOR case regarding Samsung [9].
- More than 1.5 Million records from yahoo database were deleted by one Egyptian hacker using IDOR [10].
- Turkish security researcher hacked Apple Developer's site using IDOR and revealed 275000 registered third-party developers [11].

2.2. Attack vectors for IDOR

This section discusses the most common attack vectors for IDOR vulnerability.

2.2.1. Missing access level control

In OWASP Top 10 2017, Missing Function Level Access Control and IDOR are merged together and called Broken Access Control [12]. Missing function level access control is much of an authorization issue where the application just checks for the user being authorized or not. it checks if user A is logged into the application while user A makes a request to access resource C. However, it does not check whether user A is allowed to access resource C or not. Although, this is just one of the example it may also mean certain Application Programming Interface (API) which allows access to all resources to all users and does not have an access control mechanism implemented [13].

2.2.2. Parameter modification

A web parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc. Usually, this information is stored in cookies, hidden form fields, or URL Query Strings, and is used to increase application functionality and control [14] In fact, most of IDOR attacks are performed by using parameter tampering technique.

2.2.3. SessionID prediction

The session prediction is defined as an attack which focuses on predicting SessionID values that permit an attacker to bypass the authentication schema of an application. By analyzing and

understanding the SessionID generation process, an attacker can predict a valid SessionID value and get access to the application [15]. This is one of the bad practices for developers. If the SessionID is predictable, a malicious user uses different techniques such as brute force or fuzzing attack to gain unauthorized access or to bypass authentication. Figure 2 shows an example of predictable SessionID.

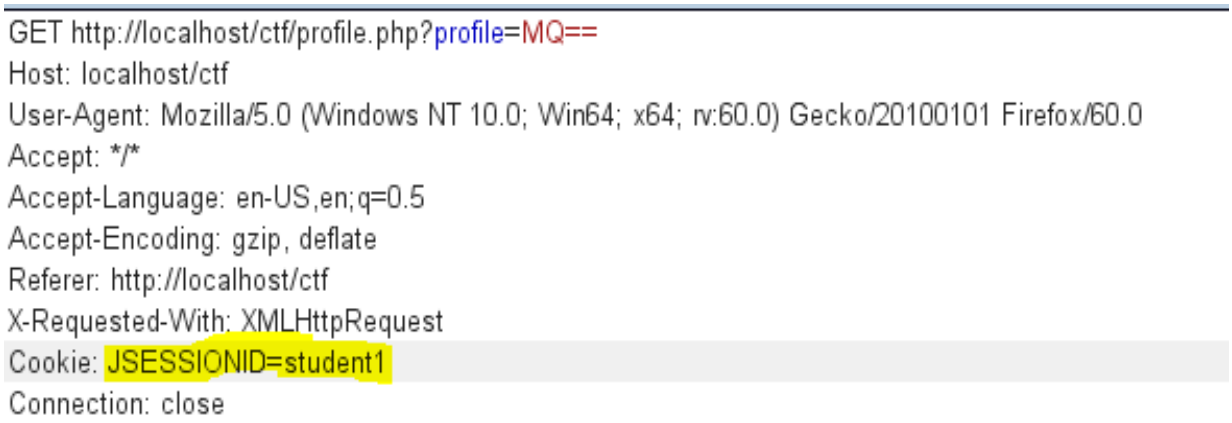

Figure 2: Example of predictable SessionID

2.2.4 Local file inclusion and path traversal

Path traversal vulnerability allows the user to see the directory list. This means, to successfully attack with IDOR, one can get advantage of path traversal to predict parameter values such as file name. "When some parameter gets name of the file to be displayed and input is not checked against path traversal, it is possible to access system files such as /etc/passwd on Linux or boot.ini on Windows operating system" [16]. Example of file inclusion is shown below:

http://www.someapplication.com/viewfile.php?name=../../../../../../../etc/passwd

2.3 How to find IDOR injection points

According to Bugcrowd article [7], the major all possible requests for IDOR vulnerability tests by checking all the features of the target system. The common places are:

- HTML source code
- JS files
- API connection
- GET and POST Request

2.4 Defensive techniques

There are plenty of existing best practice defensive techniques [4, 7, and 18]. Some of these techniques are:

- The use of indirect object reference helps to prevent from getting access to unauthorized resources.
- For each requested record, make sure that the user has the right privilege.
- Make sure that data tampering can be detected.
- Encrypt legitimate values to make it more difficult for an attacker to predict the pattern and find other possible keys.

Finding IDOR using automated tool is challenging since patterns can be varied from web application to web application. In this work, some of the existing solutions (whitebox and blackbox security scanning tools) are considered in order to understand the existing problems and their solutions.

Another approach is to teach/learn from hand-on lab exercise to improve developer's mistakes. One of such practices is using Capture the Flag (CTF) exercises, which provides challenges that are universally acceptable in the form of exercises with the aim to teach security vulnerability through hands-on attack and defense exercises.

3. Existing solutions

The first section of this chapter discusses about the existing whitebox and blackbox security scanning tools which are used to scan for web application security vulnerabilities. Those tools are evaluated based on their ability to find IDOR from a broken web application which have multiple IDOR vulnerabilities. For this purpose, broken web application using PHP programming language is implemented and it is used to evaluate existing security tools. The selection of PHP programming language is based on its global recognition.

Security scanning tools are categorized into two, namely Whitebox and Blackbox security scanning tools. Whitebox Scanning tools are used to check the source code or the internal implementation of the system [19]. Whitebox testing helps to find bad implementation in the source code. On the other hand, Blackbox tools scans the system without knowing any information about the internal implementation [19]. In this thesis, three Source Code analysis tools as well as Blackbox testing tools are assessed.

3.1 Source code analysis tools

The aim of source code analysis tools is to find vulnerabilities from source code implementation. For this reason three well known source code analysis tools namely, SonarQube, Checkmarx and RIPs are selected in this study. For this work, any source code analysis tool which does not support PHP is not a preferable since the vulnerable web application was implemented in PHP language. Therefore, the selection of source code analysis tools was made based on their support for PHP programming language and global recognition.

The purpose of this source code scan is to find IDOR using the above selected tools.

3.1.1 SonarQube (community version)

SonarQube is developed by SonarSource and it is used as an open source platform for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on more than twenty programming languages [20]. Since SonarQube supports PHP programming language, it can be used to scan the vulnerable web application source code and test if SonarQube successfully finds IDOR vulnerabilities. Figure 3 shows SonarQube scan results.

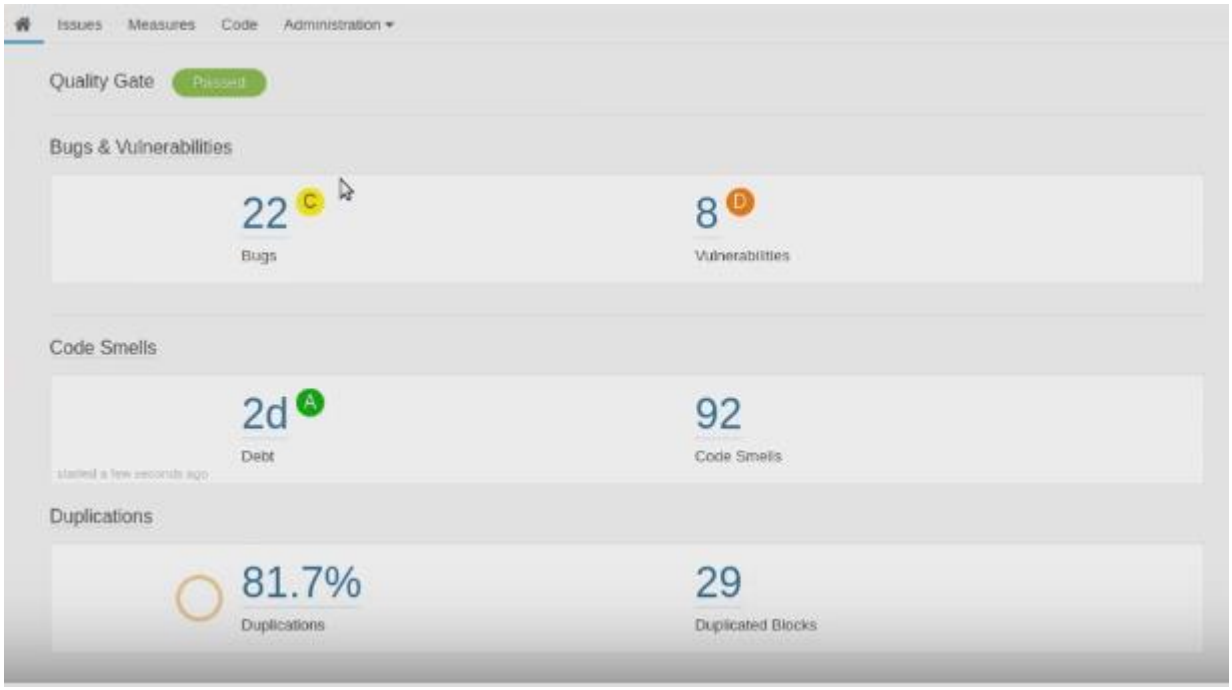

Figure 3: SonarQube scan result

As can be seen from the scan results, most of the findings were related to code quality issues. In addition, some security vulnerabilities which are not related to IDOR vulnerabilities were identified. Therefore, it can be concluded that SonarQube failed to find authorization related vulnerabilities.

3.1.2 RIPS (community version)

RIPS is one of the most popular PHP source code analysis software that is used to automatically detect security vulnerabilities in PHP applications [21].

In order to evaluate RIPS the same vulnerable web application used for SonarQube was scanned. Figure 4 depicts the RIPs scan result.

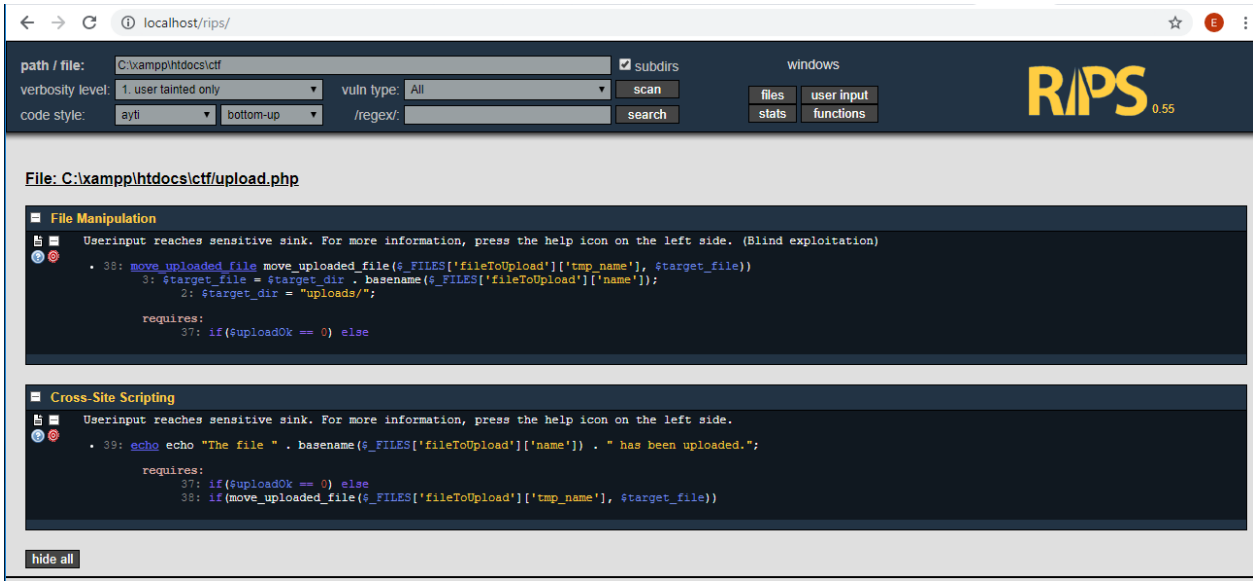

Figure 4: RIPS source code scan result

The scan results indicates that RIPS failed to find IDOR flaws from the given target, but was able to find other vulnerabilities (XSS and File manipulation), which are out of the scope of this thesis. Moreover, there is no way to query own rules in order to enforce advanced scanning. Thus, this shows that RIPS is not a preferred solution to find IDOR from source code.

3.1.3 Checkmarx (enterprise version)

Checkmarx is “a provider of state-of-the-art application security solutions: static code analysis software, seamlessly integrated into development process” [22]. This tool is one of the most expensive tool which costs approximately $15,000 a year per license [22]. Checkmarx has different products, but in this thesis the focus will be on CxSAST. Figure 5 and 6 show Checkmarx scan results.

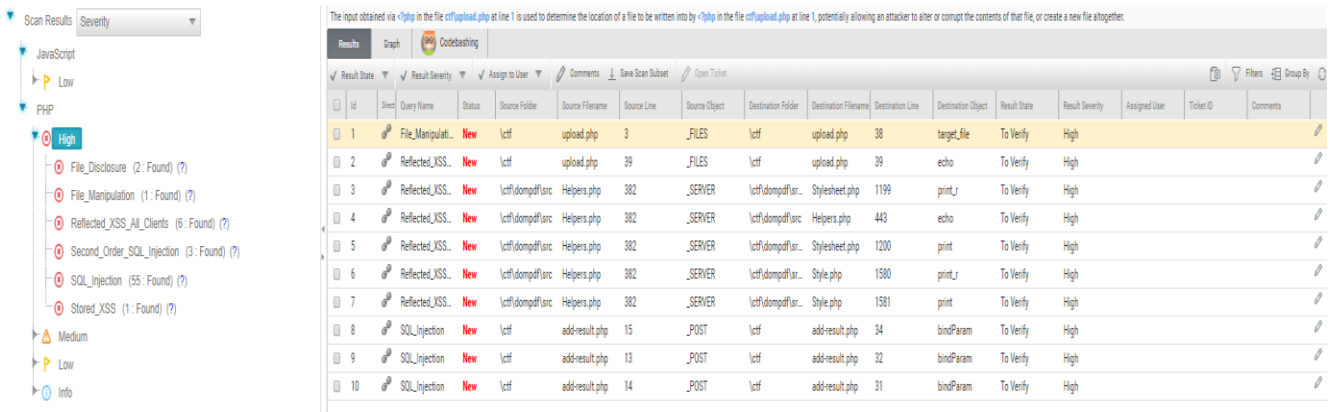

Figure 5: Checkmarx scan result 1

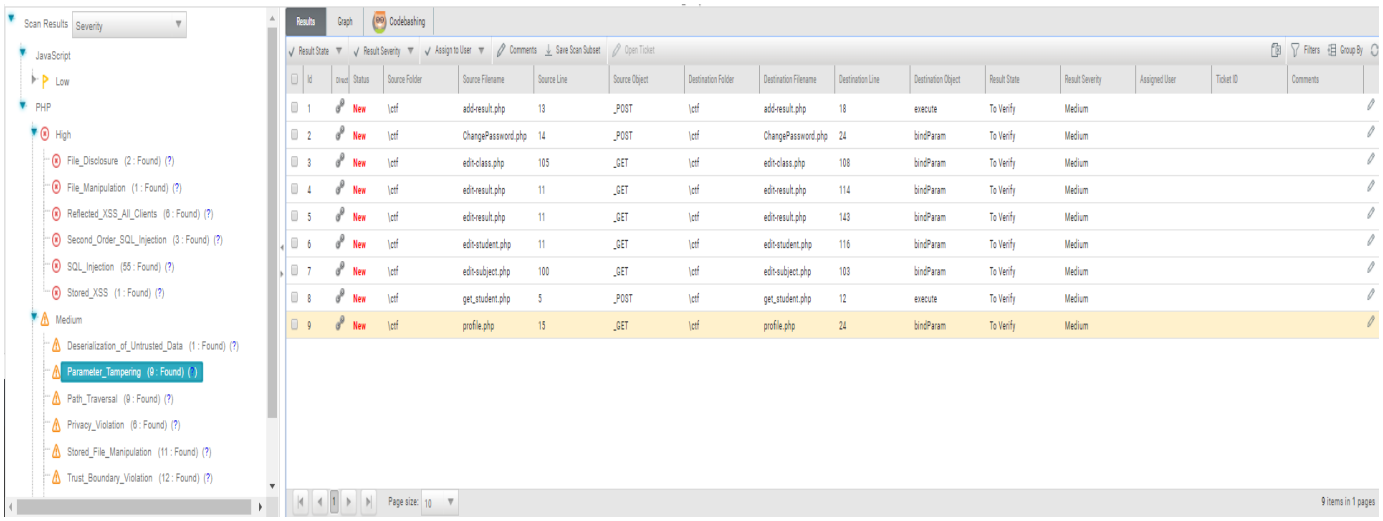

Figure 6: Checkmarx scan result 2

It can be observed that the majority of the finding was related to SQLi and XSS which also include some false positive results. Due to the scope of this thesis, only IDOR related findings are discussed here. Checkmarx was able to find some of IDOR vulnerabilities relate to parameter tampering, which is one of the major cause for IDOR attack. If incremental integer IDs are used to reference an object without proper authorization control, CxSAST detects easily from the implementation. However, if the ID values are encoded, Cx does not detect such type of vulnerability. In addition, one of the vulnerability from the broken web application which allows unauthorized user to delete all comments records from database was not detected by Cx. In general, Cx has a better scan results as compared to other two open source analysis tools. However, Cx is not a complete solution to find authorization vulnerabilities from source code.

To sum up, among the three different source code analysis tools used, two of them failed to find any IDOR vulnerability from source code. However, Cx scan showed better results, although it did not find all of the vulnerabilities. Therefore, based on the above results, it can be concluded that a complete reliance on source code analysis tools is not a good practice to find IDOR vulnerabilities.

3.2 Blackbox security scanners

There are several web application security scanners which are open source and enterprise (paid) version. In this thesis, open source tools namely, w3af, OWASP ZAP and NIKITO have been used. After scanning the prototype application, the strength of those tools in finding IDOR flaws is evaluated. The tools are selected since they are available as open sources and they are commonly used by many security researchers to scan web application security.

3.2.1 w3af

w3af is one of the most widely used penetration testing tool for attacking and auditing web applications. The main purpose of this tool is to find and exploit web application vulnerabilities [23]. Evaluation of the w3af tool is done in Ubuntu Linux operating system, hence w3af is installed in Ubuntu Linux OS. Figure 7 shows the scan result of w3af.

Figure 7: w3af scan result

The results indicate that w3af was not able to find IDOR or any broken authorization related vulnerabilities from the given target. As a result, this tool is not a good choice to scan insecure direct object references vulnerabilities.

3.2.2 OWASP ZAP

Zed attack proxy (ZAP) is “an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by people with a wide range of security experience and as such is ideal for developers and functional testers who are new to penetration testing” [24]. In OWASP ZAP spider scanning is used to scan the whole system automatically. Figure 8 shows ZAP scan results.

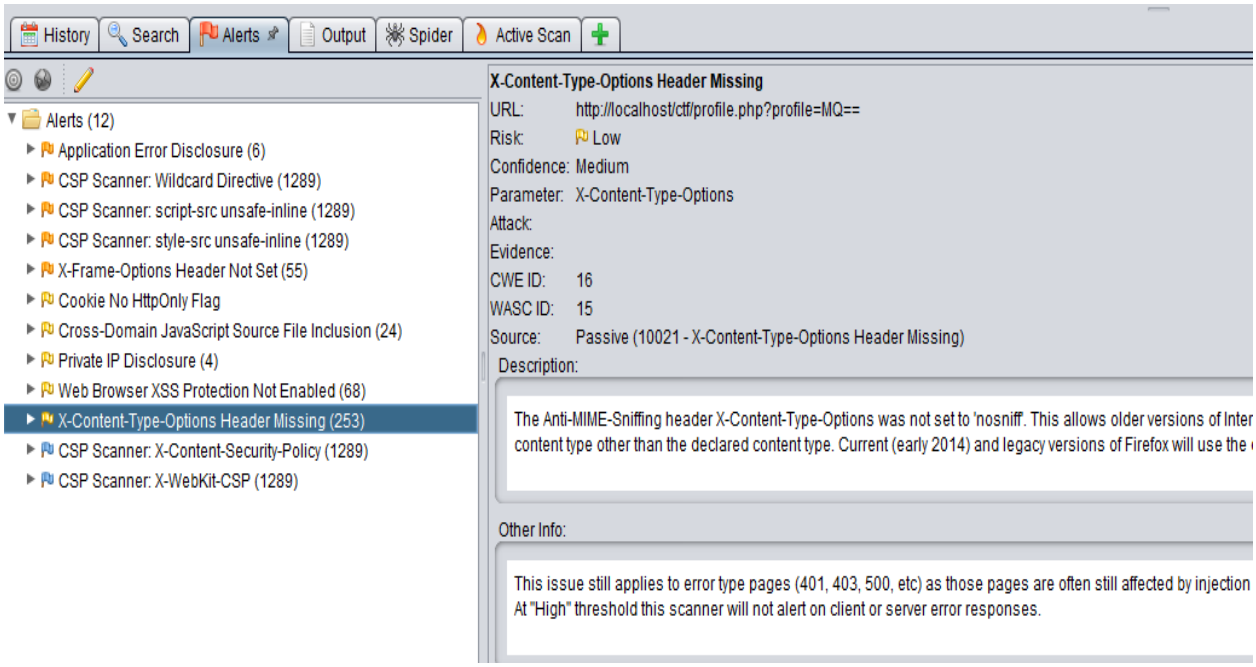

Figure 8*:* ZAP scan result

Similarly, the scan result shows that ZAP failed to find IDOR from the vulnerable web application. However, this tool is suitable for injection vulnerabilities such as XSS and SQLi but not for broken authorization flaws. One of the most important feature of ZAP proxy is the possibility of inspecting and tampering the requests before sending to server. In such a way, there is a possibility of finding IDOR, however, it is not fully automated, thus it needs human interaction.

3.2.3 NIKTO

Nikto web application vulnerability scanner is a security auditing tool which tests a lot of possible security issues on a website including XSS and SQLi [25]. Since this tool is coming with Kali Linux system, scanning the vulnerable prototype has been done in Kali Linux. Figure 9 shows nikto scan results.

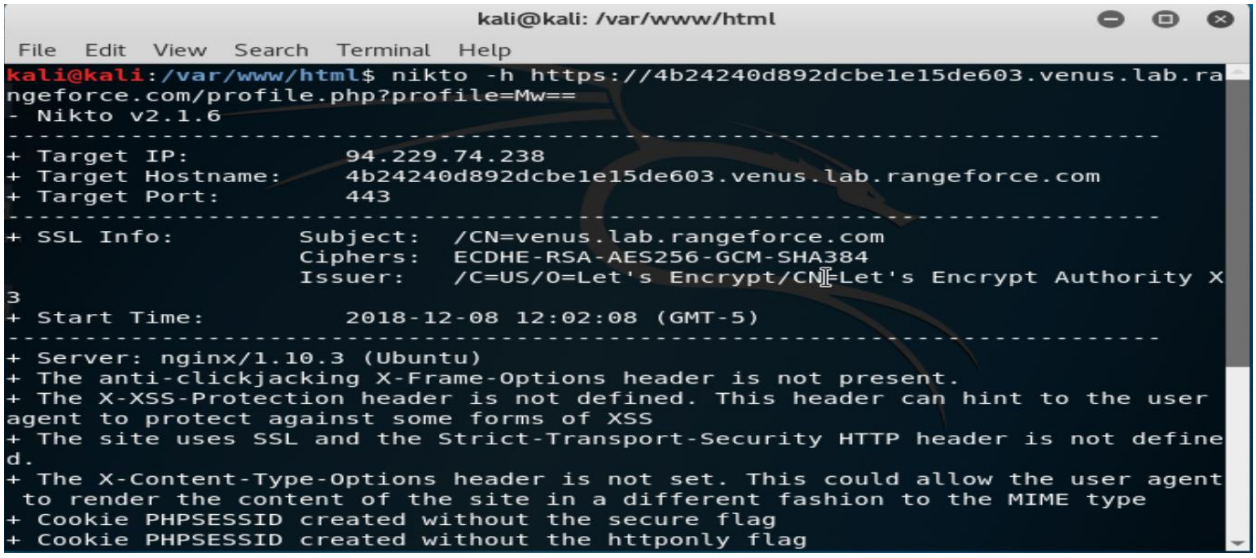

Figure 9: Nikto scan result

Nikto is more easy and efficient to use. Even though this tool failed to find IDOR vulnerability, it is capable of finding lots of vulnerabilities in the web applications. In this regard, it can be suggested that Nikto is highly efficient for webserver scanning and other types of web application vulnerabilities but not for authorization related vulnerabilities such as IDOR or missing function level access control vulnerabilities.

3.3 Web application firewall (WAF)

A web application firewall (WAF) is an application firewall for HTTP applications that applies a set of rules to an HTTP conversation. These rules, in general, include common attacks such as cross-site scripting (XSS) and SQL injection. While WAFs protect servers, proxies generally protect clients. [26]

There are many available open source WAF. However, in this paper we will see four open source WAF. These are: ModSecurity [27], NAXSI [28], WebKnight [29] and Shadow Daemon [30]. In this work, detailed discussion about each WAF is not covered, but instead the list of requests that could be filtered by each of the above-mentioned WAF is considered.

- ModSecurity filters the following payloads:
  - Cross-site scripting
  - Trojan
  - Information leakage
  - SQL injection
  - Common web attacks
  - Malicious activity

- WebNight filters the following payloads:
  - Buffer overflow
  - Directory transversal
  - SQL injection
  - Blocking bad robots
  - Hotlinking
  - Brute force

- Shadow Daemon filters the following payloads:
  - SQL injection
  - XML injection
  - Code injection
  - Command injection
  - XSS
  - Backdoor access
  - Local/remote file inclusion
- NAXSI filters the following payloads:
  - cross-site scripting
  - SQL injection attacks.

From the above list, it can be observed that most of injection attacks are filtered using WAF. However, unlike injection attacks, to successfully attack IDOR vulnerability, it is not required to use any malicious payload to perform unauthorized action. In other words, none of the above mentioned WAF helps to detect or filter IDOR attacks.

To summarize, finding IDOR or authorization related vulnerability using the state of the art automated tools was difficult. By using Checkmarx, it was possible to find integer ID parameter tampering which is related to IDOR vulnerabilities. However, it is still possible to bypass Cx checks if the parameters are encoded. Hence, it can be concluded that automated tools are very helpful to find different flaws from the source code, however, permission related flaws are challenging to be found using the above mentioned tools.

3.4 Existing CTF challenges

To prevent IDOR vulnerability, the alternative option is to give awareness training for student/developer. According to Stohr-Hunt's study, the best way to deliver awareness training is with the hands-on practical learning method [31]. Capture the flag (CTF) is one of universally acceptable ways to create practical attack and defense exercise [5]. Each of the existing cyber challenges has one or more of the following drawbacks:

- IDOR exercise is not included
- Assistant teacher is required to conduct the training
- The broken web app prototype is not suitable for IDOR.
- It is not freely available
- It is unavailable for academic use
- It does not include mitigation/fixing exercises

3.4.1 bwapp

bwapp is used to discover and prevent web application vulnerabilities. bwapp is useful to conduct successful penetration testing and ethical hacking projects. A unique feature of bwapp is that it has over 100 web bugs and covers all major known web vulnerabilities, including all risks from the OWASP Top 10 project. Hence, it has the advantage in that it covers a wide range of vulnerabilities [32].

Limitation or bwapp

bwapp has Insecure Direct Object references exercise, however, all the challenges are designed to show the offensive ways. It does not guide or give a hint to successfully find the vulnerability, Hence, in this case, an instructor must be there to assist the students to complete each task. Moreover, it does not provide or teach the way to mitigate a vulnerability.

3.4.2 Security shepherd

Security Shepherd is one of OWASP project that aims provides students with manual penetration testing skills. One of the interesting thing in Security Shepherd is that it presents reading material for security risk concepts before any challenge is started. The lesson provides a user with help in layman terms about a specific security risk and assists them in exploiting a textbook version of the issue [33].

Limitation of Security Shepherd

Although Security shepherd is available for academic use, it only focuses on teaching penetration testing skills. It does not contain defensive exercise. Besides, the material provided by security shepherd does not give any mitigation solution for IDOR challenges.

3.4.3 DVWA

DVWA stands for Damn Vulnerable Web App is an open source project from OWASP. It is an intentionally damn vulnerable implemented using PHP/MySQL. The main goals of this project is for professional to assess their level of security knowledge [34]. In addition, this can be used to learn about most of OWASP top 10 security vulnerabilities. It also helps developers to better understand the best practice mitigation techniques. Moreover, it can also be used by teachers/students to teach/learn the most common web application security vulnerabilities in a classroom environment [34].

Limitation in DVWA

Even though DVWA is designed nicely to help security professionals to test their own skills, it does not include any IDOR exercises. Therefore, this cannot be a good choice to solve the problem statement of this work.

3.4.4 OWASP WebGoat

WebGoat is also another open source project from OWASP which is used for web application security lessons. The program is a demonstration of common server-side application flaws. The exercises are designed for people to learn about application security and penetration testing techniques [35].

Limitation of OWASP WebGoat

Although WebGoat has IDOR challenges, its main focus is only to teach penetration testing skills without considering how it can be mitigated. Moreover, an instructor is needed to follow up and assist students if they need hint or help.

3.4.5 OWASP Juice shop

OWASP Juice shop is defined as an open source application that contains a large number of hacking challenges of varying difficulty for a user to exploit the underlying vulnerabilities. Juice shop is a recent project which is designed using JavaScript technologies and is available

for an academic use. One of the important feature in OWASP Juice is the scoreboard which is used to track hacking progress [36].

Limitation of Juice Shop

OWASP Juice shop challenges focuses only to teach penetration testing skills without considering how it can be mitigated. An Instructor is needed to guide students to complete each task. There is no guiding hint or reading material available. Moreover, IDOR exercise is not present, thus not a good choice for IDOR training.

In conclusion, the majority of the CTFs discussed do not include IDOR exercises. In addition, none of them are designed to teach how to fix IDOR vulnerability.

4. Methodology

When outlining courses for teaching, a particular method needs to be decided for development. This helps to define the major objectives that define the goal of the training, outline requirements for the learner, define target audience and validate the efficiency of the training. According to Gardner [37], instructional design process may involve the following steps:

- must conduct a needs research and analyze target group needs.
- determine whether these needs can be fulfilled by learning; and exactly how.
- they write the learning objectives and conduct research to see what the outcomes are.
- they assess each trainee's entry skills and knowledge.
- based on all of the above analyses and outcomes, an Instructional Designer should choose the instructional strategies and training techniques and select the media formats appropriate for the training.
- after the course is over, they need to follow-up participants and make sure the course has been beneficial and sufficient for their future personal and professional growth.

Although there are different existing instructional design models to create training materials, ADDIE model is one of the most popular of all [38]. In this work, ADDIE model has been used as the instructional design model to create a hands-on lab exercise. The ADDIE model is commonly used by instructional designers and training developers. The five phases, i.e, analysis, design, development, implementation, and evaluation, represent a dynamic, flexible guideline for building effective training and performance support tools [39]. Figure 10 shows the five phases of ADDIE model.

Figure 10: Five phases of ADDIE Model [40].

4.1 Analysis

In the ADDIE model, the analysis phase is used to study the requirements prior to the content creation. Analysis helps to gain a clear understanding of primary audience of the course, physical and organizational constraints, technical requirements, accessibility of the course, and evaluation criteria [39].

A. Need analysis

- As a result of informal interviews held with developers and IT professionals, it was revealed that several difficulties were experienced within the existing security scanning tools. As the authorization pattern is different from a web application to another, it is difficult to find IDOR vulnerability using automated tools. In addition, publicly available IDOR exercises missed the defense exercises.
- With the increasing spread of online CTF exercises, learners' interest in new methods and their willingness to learn from attack and defense exercise were among the factors that led to teach this course online.
- A lab that would work regardless of time and location constraint

B. Student analysis

It was important to conduct analysis regarding of learner who would exercise the online learning platform including their academic results, their experience in programming

language and their level of education. The student analysis can make the exercise to be more productive and increase learner's motivation [41].

The minimum technical requirements are:

- The learner should have basic understanding of web technologies such as: HTML, PHP and database query languages. Although the vulnerable web app is implemented using different frameworks such as Bootstrap, JavaScript and jQuery, previous experience with frontend languages or frameworks are not required.
- Basic understanding of HTTP Requests including GET and POST requests i.e. how the web browser (client) communicate with web server using GET and POST requests.
- Since the database is implemented using MySQL database, the learner should be comfortable with basics of MySQL query i.e. select, insert, delete and update statements.
- For attacking scenarios, the learner should be able to understand basics of HTTP proxy, which is widely used by web app security test. i.e. intercept, view and modify HTTP requests and responses.
- Basic understanding for URL encoding/decoding.

The above-mentioned technical requirements are a self-assessment for a student. Hence, the lab contents are designed considering the fact that the learner fulfils the minimum skill sets.

C. Content analysis

The content of IDOR exercise is composed from different online materials mainly from research paper, slides, blogs and security write-ups.

D. Structure analysis

The course to be taught would have a web based online structure where factors related to the course would be organized accordingly.

E. Online environment analysis

For the learning platform, possible online cyber range platforms were investigated. From the available options, the most appropriate cyberrange platform for the exercise is selected.

F. Technical analysis

The technical equipment and software to be used were listed, for that reason students are required to have personal computer and internet connection. The online cyber range

platform should have the software requirements, including Ubuntu OS, code editor, web browser, python, PHP, nginx and MySQL server.

G. Assessment criteria

It is also important to determine the evaluation of the assessment criteria. After completing the hands-on lab exercise, the student skills will be measured by the number of tasks solved and hints used.

4.2 Design

As per the guideline of ADDIE model, design phase is "a process which includes responding to the questions of how to carry out the objectives and strategies determined in the analysis phase" [39].

4.2.1 Learning objectives

This component defines the expected goal of the hands-on lab exercise by defining the knowledge acquired by a student because of instruction. The main objectives of IDOR hands-on lab are:

- To understand IDOR Vulnerability and the risks it imposes if it exists.
- To teach where Insecure DOR occur and to find from the vulnerable web application.
- To teach how to mitigate/patch the vulnerability using PHP programming language.

Table 1 describes the objectives of IDOR attack and defense exercise and it covers the initial learning objectives of the hands-on lab exercises with a possibility to add new objectives in the future.

Table 1: Learning objectives of the lab

Learning objective	Description	Validation
Student will be able to find confidential information from vulnerable web application.	Learn how to take advantage of IDOR to see confidential information from the system.	Automated Script verifies when the student submits the flag.
Student will be able to identify encoded parameter and tamper parameter to find other students' private information from vulnerable web application.	Learn to decode parameter and understand the way to find confidential information.	Automated Script verifies when the student submit the flag.
Student will be able to identify direct object reference and delete all records from vulnerable web application.	Learn the impact of IDOR in deleting pages and removes all data from database using IDOR flaw.	Automated Script verifies when the student submit the flag.
Student will be able to implement access control check in SQL query	Learn to check access control before executing the sql query	Automated Script verifies automatically when the vulnerability is fixed.
Student will be able to implement access control check for direct object references using PHP programming language.	Learn secure implementation of session to apply access control check	Automated Script verifies automatically when the flaw is fixed.

A. Outline reading materials

There are different ways to outline training materials, but the basic principles helps to understand how to logically and creatively structure a training that achieves the learning objectives. In order to successfully complete the attack and defense exercises, a clear instructions and reading materials should be provided for a learner. The reading

materials is breakdown into three segments: “understanding IDOR vulnerability”, “how to find IDOR”, and “how to Fix IDOR vulnerability”.

B. Design the score evaluation method

The evaluation of learner performance is measured based on the number of tasks completed by the user. This can be validated using an automated python script, which checks for the correct solutions.

C. Design system

The quality of IDOR attack and defense lab depends on the selection of the vulnerable web application. Therefore, based on the learning objectives, a grade management system is selected. The system has three different roles: student, teacher, and admin. For each roles, different permissions are assigned. Table 2 shows the permission of each roles.

Table 2: The system permission assigned per role

Role	**Permission**
Admin	➢ Add/Edit/Delete Student ➢ Add/Edit/Delete Teacher ➢ Add/Edit/Delete course and class ➢ Assign student to course ➢ Assign teacher to course ➢ Change Password
Student	➢ Change their own password ➢ View exam result ➢ View own profile ➢ Add/Delete comments to Blog page
Teacher	➢ Manage class routine ➢ Add/Edit/Delete student result ➢ Manage own password

D. Delivery method

For delivery method, it has been proposed to publish the training in online rangeforce platform. In addition, it will also be available in GitHub from https://github.com/EphremG/ctf as an open source project contribution. This will give a better availability for anyone interested to download at any time to learn/teach about insecure direct object references. Evaluation script will be in a separate folder (https://github.com/EphremG/CheckerScript). If one needs to run in localhost, then evaluation scripts should be executed manually. However, for simplicity and

easy integration rangeforce platform is used. Rangeforce was selected for many reasons including its easy integration, user-friendliness and necessary modules such as SolutionAPI and Scoreboard are readily included.

4.3 Development

This phase is a foundational step because it ensures the information gathered in the previous phase to be transmitted to the learner. Therefore, it is the phase which includes the preparation of the platform to be used [41].

A. Developing the prototype

The system used to create IDOR attack and defense lab is web based grade management system. The main tasks will be created on student privilege which includes viewing student result, profile and posting/deleting to the blog page. Figure 11 represents the sequence diagram for student.

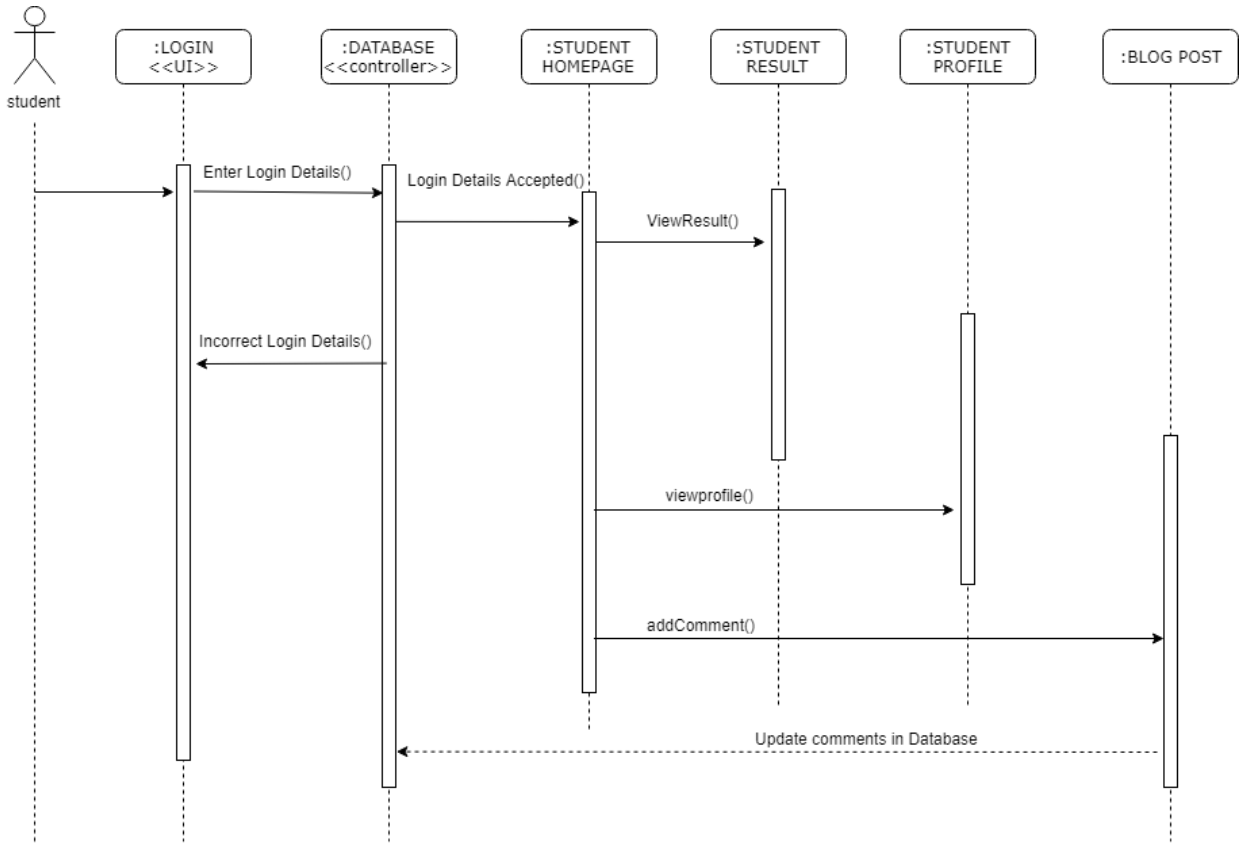

Figure 11: Student sequence diagram

B. Preparation of exercises

I. IDOR attacking

The hands-on lab exercises consists of different challenges. In the following sections, three different exercises namely, finding confidential information, manipulating encoded parameter, and deleting records from database are discussed.

Challenge 1: Finding confidential information

For this challenge the learner will get a student privilege login access and will be asked to find another students personal information such as other students grade result. In this task, the learner will understand about hidden input inspecting and parameter tampering to complete the task. Finally, by submitting the flag, the student gets a point.

Challenge 2: Manipulating encoded parameter

One of the most common developer's misconception about IDOR is that encoding the parameter solves IDOR issues. This is in fact a best practice but not a root solution. Encoding the parameter values prevents hacker from easily predicting the parameter, however, by using a tool it is possible to decode and understand the pattern. Therefore, with this challenge the learner will be able to understand ways to tamper encoded parameter and manipulate the values. In this challenge, the learner first needs to use the same login credentials to access the system. The aim of this Scenario is to teach the learner that not all encoded parameters in the URL are not free from IDOR vulnerability. To complete this task, the learner needs to use base64 decoder tool, which is freely available online.

After decoding the parameter values, the learner will understand the pattern used to display student profiles. After that, by incrementing/decrementing the ID values in encoded format, the learner will find the hidden flag/solution. Once the flag is submitted, the point will be given.

Challenge 3: Deleting values from database

Developers use different approaches to protect against authorization issues in a group message or blog pages. However, every user has the right to delete only his/her own comment/post. Nevertheless, owing to insecure implementation of the prototype, it is possible to inspect the inputs and delete other users' comments as well. In fact, this is one of the most common type of bug bounty findings [7].

For this challenge, the task is to delete another student's comment from the blog post. The prototype was designed to allow student to delete his/her own comment/post. However, since the blog page is vulnerable for Insecure DOR, it is possible to delete other users' post as well.

Up on deleting all comments and posts, the flag will appear in the page. By submitting the flag, student will get point.

II. IDOR defense

The learning objectives of defense scenario is to understand the vulnerability from the given report/instructions and to fix the flaw. In this part, the learner focuses on the mitigation techniques only. It is possible to apply different approaches to prevent an application from IDOR vulnerability. The most common approaches are collected from the following studies: [4] and [18].

Validation

Data validation helps to confirm that the user input is clean, correct and useful data. Input validation is a very vital process to protect the system against different types of attacks, such as path traversal, and most of injection attacks i.e. SQL injection, cross-site scripting injection, command injection.

Sanitization

This is also one of the best method to solve SQL injection issues. Even though this is a best practice, it is difficult to defend IDOR attack only by sanitizing user input. However, this practice helps to mitigate other types of vulnerability which can be an attack vector for IDOR.

Unpredictable values

This is very important technique to prevent against IDOR attack [4]. For an attacker to successfully test IDOR, he/she has to know two different parameter values of different users. An example can be two users each having access to different objects with the same or different privileges. If the attacker easily predicts possible inputs or the iteration, it can easily be tested for IDOR attack. Hence, making it harder to guess is one approach to prevent hackers from successfully attacking your system. This can be implemented in different ways: encoding, hashing, concatenating string and numbers for the unique parameter values.

Implementation of indirect object mapping

An alternative approach to prevent direct object reference vulnerabilities is by mapping original values such as ids, keys, etc with cryptographically strong random values [4]. All the mapping

implementation will be on the server side, thus, the actual value will not be exposed to the user. This is a better mitigation than logical validation [18].

Implementation of access control

The key solution for IDOR vulnerability is implementing access control check for all directly referenced objects to ensure the user is authorized for the request object [4]. For the defense challenges, the student use this mitigation approach.

The purpose of defense challenges is to fix the previous vulnerabilities in the attacking challenges. Hence, the source code can be available only once the student successfully completed the attacking challenges. The purpose of hiding the source code is to prevent students from cheating the flag in the source code. For this purpose, the desktop screen has two different windows as shown in Figure 12.

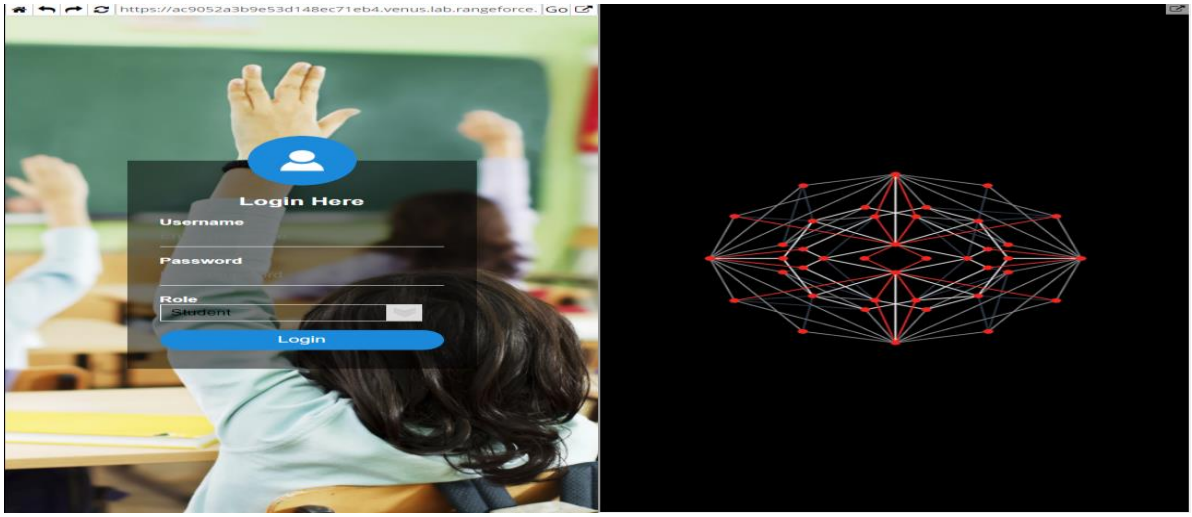

Figure 12: windows screen before completing the attacking challenges

In the left side, the target system will be visible for the student, from that the attacking exercises is completed. On the right side window, the source code is shown using code editor tool. However, the right window will be visible only upon completing all the attacking challenges. The main reason is to protect students from cheating the solution in the source code.

Figure 13 shows how the windows looks after finishing the attack challenges.

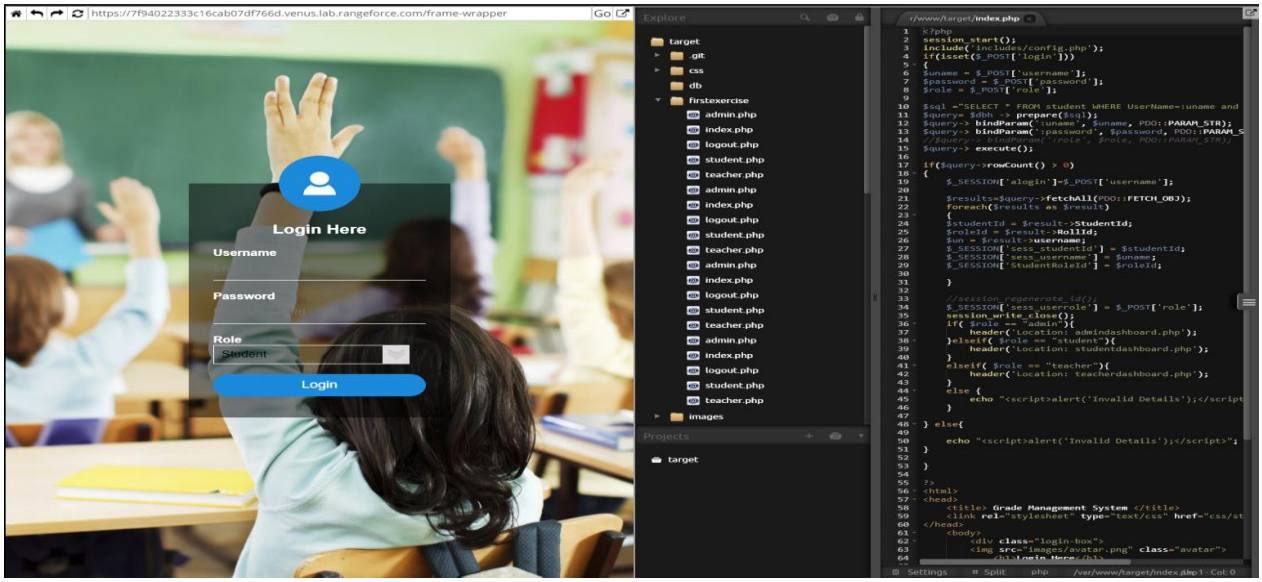

Figure 13: Windows screen after completing attacking challenges

C. Testing

According to ADDIE model, development phase testing is very critical [40]. This is because initially many plans seems ok, however, when it comes to reality, different issues begin to appear. Therefore, this phase is used to ensure if the design process was successful. For this purpose, the initial testing was conducted and the design matches the learning objectives.

D. Editing

This phase is important to ensure that there is no distractor in the contents that can impact the integrity of the course which also includes spelling checking from the instruction and reading material.

Evaluation script

In this phase, evaluation method for the student solution and score calculation is developed. For this purpose the author implemented checker script using python. Brief details on the checker script will be discussed in the next section.

Score calculation

The lab contains two different exercise namely: IDOR attack and IDOR defense. The weight of each task worth different points. Defense challenges require more effort, hence, 40% point is given for completing IDOR attacking challenges and 60% is given for completing IDOR Defense Challenges. Finding the ideal scoring formula is out of the scope this thesis. Therefore, the scoring points were chosen because of its simplicity to implement and understand.

Prototype flexibility

This prototype allows existing scenarios to be reconfigured or allows for entirely new IDOR scenarios to be created. In addition, this prototype is also open for additional scenarios with new vulnerability types such as XSS, SQLi, Path Traversal and so many other web application security vulnerabilities.

Process of developing new Challenges

The prototype does not contain all the IDOR related exercise. But it is open to add new tasks on top of the existing exercises. Figure 14 defines the process of adding new challenges.

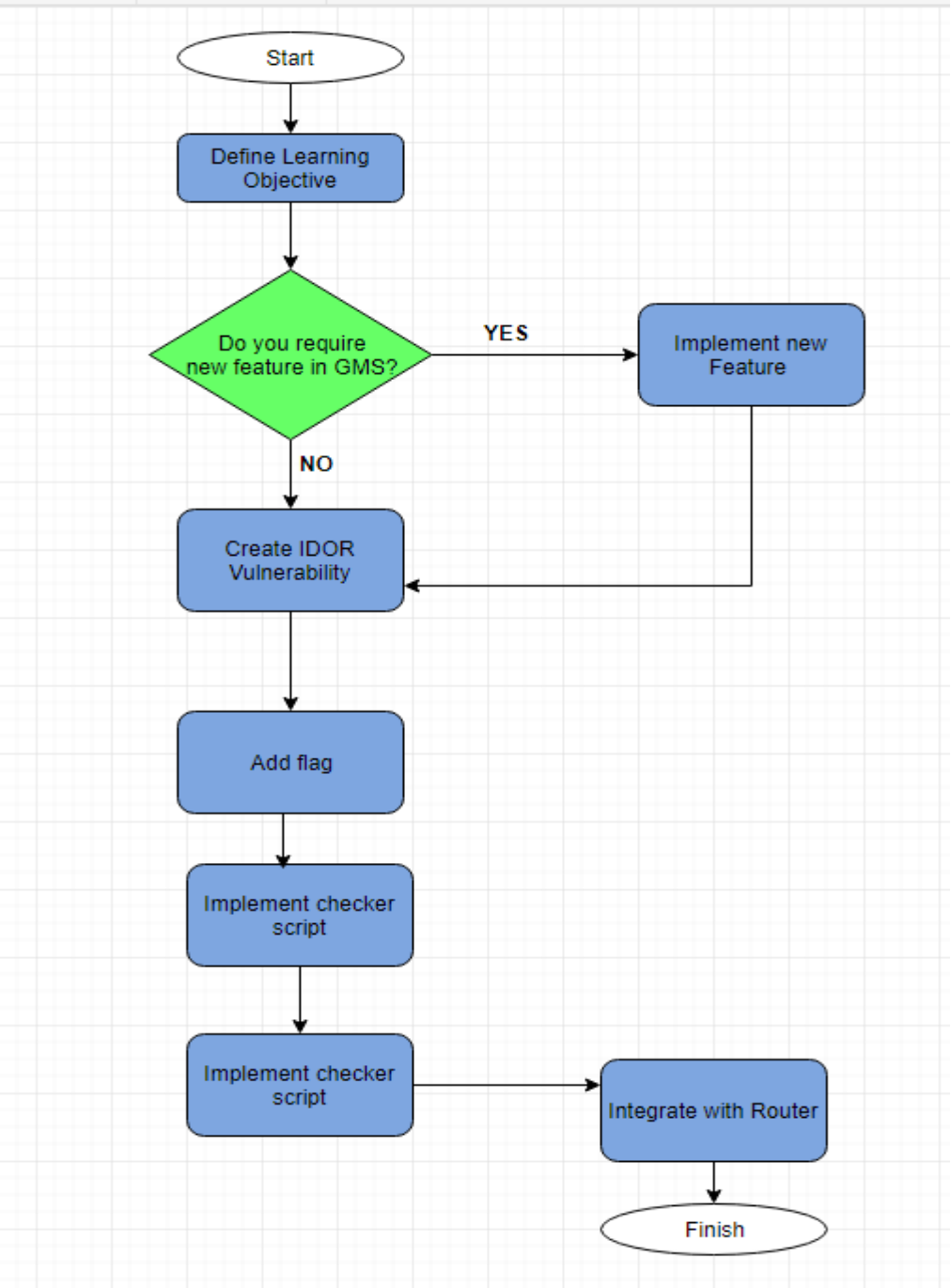

Figure 14: Process of adding new challenges

To understand each process/step, one can consider the following example of adding a new download feature which helps students to download their results.

Step 1: Define learning objectives: understand where IDOR exist in download page

Step 2: Create download.php and implement download functionality to download student grade result.

Step 3: Create IDOR vulnerability in download.php. For example one student to download other students' grade reports.

Step 4: Create flag which helps to give a point when the student successfully completes the task/s.

Step 5: Implement checker script, which is used to evaluate the defense task solution.

Step 6: Integrate step 5 script with router machine to automatically evaluate solutions.

4.4 Implementation

According to ADDIE guidelines, implementation is the fourth stage of the ADDIE model. In this stage, the design phase is implemented and evaluated accordingly [40].

The hands-on-lab is implemented as an open source and can be downloaded from GitHub public repository: https://github.com/EphremG/ctf. The main purpose in open source contribution is to open a door for others to collaborate and share in the future. In addition, it is also open to modify and integrate these challenges into other projects.

More importantly, for simplicity and virtual accessibility, it is integrated with RangeForce platform. RangeForce platform offers several benefits by providing a different functionalities. These include:

- ➢ Login authentication for learner
- ➢ Provide API to integrate for feedback and check
- ➢ Provide source code editing tool for defense task/s
- ➢ Provide a suitable way to include the necessary reference materials for learners to read
- ➢ Provide scoreboard

The architectural design of the hands-on-lab is demonstrated in Figure 15.

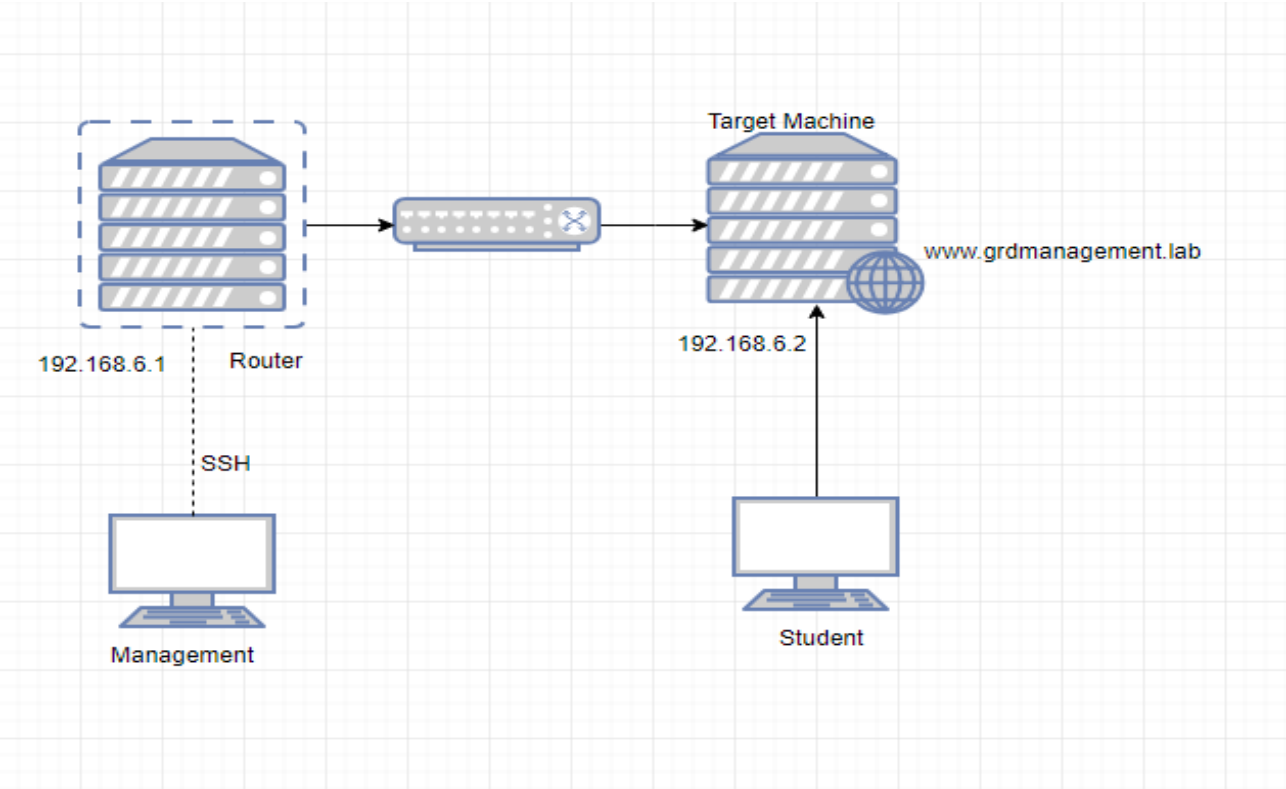

Figure 15: Network architecture for the lab

To successfully start the lab, the following steps are performed:

Step 1: Clean Ubuntu 18.04 machine is cloned in the target machine

Step 2: All needed softwares (Nginx, MySQL, PHP, etc.) are installed on the target machine

Step 3: From GIT repository the prototype and checker script are cloned to target machine and router machine respectively.

Step 4: All configurations should be done (Virtual host created for target, editor set up).

- ➢ Setup the web target in /var/www
- ➢ Run checkerscript from router machine every 5 seconds (This will start only after the attacking tasks are completed)

Step 5: Lab successfully starts

For security reasons student do not have access to router machine because checkerscript is installed in the router machine, thus, student might abuse the checkerscript.

Evaluating IDOR attacking solutions

Evaluating IDOR attacking solution is implemented in JSON format. The JSON file is first integrated with Rangeforce SolutionAPI. The JSON file contains the list of questions, descriptions and also the correct answers. The complete list of tasks with their description, correct answer and hints are stored in JSON format, referred from the following link.

https://github.com/EphremG/JSON/blob/master/task.json

Therefore, for attacking exercises, the learner need to submit their solution/flag through the provided form and hit the submit button. Then, SolutionAPI compares the student solutions with the correct answer and returns true/false. True means, the answer is correct and then point will be added, or false means an incorrect answer, hence the learner should redo the task.

Evaluating IDOR defense solutions

Evaluating defense side solution is quite different from the previous solution. In this case, the author implemented python script to evaluate student's defense tasks solution.

First task

In this task, the vulnerability exist on result.php page. Logged in student was able to see other students' exam results simply by changing from URL or inspecting the hidden form and changing the values. Therefore, the task is to patch this vulnerability. In order to solve this issue, the student needs to implement access control using session in PHP. After successfully logged in, the student ID will be stored in session. Before displaying the student's result, there has to be a validator to check if the form/user input request is from the same logged in students ID. The script basically does the attacking steps: first login using student1 login credential, then changing the student roleid parameter and check if the content of the page display Students2 result.

Part of the code used is shown as follows:

```
url = 'http://192.168.6.2/index.php'
values = {
    'login': 'letmein',
    'username': student1,
    'password': student1,
    'role' : 'student'
}
r = s.post(url, data=values)
```

```
url2 =
'http://localhost/ctf/result.php?rollid=2&class=1'
r2 = s.get(url2)
if Student2 in r2.content:
    print('You have not fixed the vulnerable Yet! :(' )
else:
    print(‘Successfully fixed’)
```

The complete code can be found in appendix 1.

Second task

In this task, the vulnerability is in profile.php page, on which students can see their profile details such as: full name, email, date of birth and so on. In this part of IDOR defense task, the students are expected to fix the vulnerability by implementing access control check. However, in this case the parameter is encoded with base64, which means it is less predictable as compared to the previous task, but not secured. Therefore, if an attacker decode the parameter, he/she will be able to predict and tamper the parameter to find other students profile details. Hence, the script is fundamentally checking if it can be possible to login in with student1 and able to retrieve student2 profile details. If successful, it indicates that the vulnerability is not patched correctly.

Part of the script is as follows:

```
url = 'http://localhost/ctf/index.php'
values = {
        'login': 'letmein',
        'username': ‘student1’,
        'password': 'student1',
        'role' : 'student'
}
r = s.post(url, data=values)
url2 = 'http://localhost/ctf/profile.php?profile=Mg=='
r2 = s.get(url2)
if  'Abeni' in r2.content:
        print('You have not fixed the vulnerable Yet! :(')
else:
        print(‘Correctly patched!’)
```

The complete code is found in appendix 2.

Third task

In this task, the vulnerability exists in deletecomment.php page. This flaw allows a single student to delete the entire records of students' comments and posts. Therefore, the learner should be able to fix this vulnerability by implementing a proper access control. One way to solve this flaw is through validating the user privilege and the access request ID; that way it is possible to control proper authorization which the user has the right to delete the requested reference. The solution is evaluated by using automated script.

Part of the script is as follow:

```
sql= """delete from comment where 1"""
mycursor = connection.cursor()
mycursor.execute(sql)
connection.commit()
sql= """INSERT INTO comment (id, studentId, comment, Name)
VALUES (1, 1, 'Student1 Comment', 'Ephrem'), (2, 2, 'Student2
Comment', 'Abeni')"""
mycursor = connection.cursor()
mycursor.execute(sql)
connection.commit()
print(mycursor.rowcount, "record inserted")
```

In the previous attacking task, student removed all comments. Hence, we need to make sure all values are deleted from comment table. Then the automated script inserts two comments with two different student account for testing purposes. After this step, the next step is to check if student1 is able to delete student2 comments. The following part of the code covers this checks.

```
s = requests.Session()
url = 'http://192.168.6.2/index.php'
values = {
    'login': 'letmein',
    'username': 'ephrem',
    'password': 'ephrem',
    'role' : 'student'
}
r = s.post(url, data=values)
url = 'http://192.168.6.2/deletecomment.php?commentid=2'
url = s.get(url)
```

```
if 'ABENI' in str(url.content):
    print('Congratulations, You have Successfully fixed the issue! :)')
    exit (0)
else:
    print('You have not fixed the vulnerable Yet! :(')
    exit (1)
```

The complete code can be found in appendix 3.

4.5. Evaluation

Evaluation is the last phases of ADDIE model in which the final validation and improvement to the training content correctness will be conducted [40]. In this thesis, to successfully validate the hands-on lab exercise, evaluation was conducted into two phases. First evaluation was done with pilot group who are experienced in this topic. In the second phase, 26 participants from TalTech IT College students enrolled for Web Application Security course were participated for validation process.

4.5.1 Evaluating with pilot groups

In the first phase, five people qualifying with the minimum skills set were selected to evaluate and give feedback about the exercise content and correctness. Selection of this pilot groups was based on the following questions:

1) Are you familiar with HTML?
2) Are you familiar with PHP programing language?
3) Are you familiar with basics of SQL query?
4) Are you familiar with HTTP requests (GET and POST methods)?
5) Do you know Insecure Direct Object References?
6) Do you have experience with CTF?

4.5.2 Feedbacks from pilot group

The overall pilot group response showed positive feedbacks and the lab worked correctly with some minor issues. Majority of the suggestions were considered to be relevant and immediate improvement was done. The most common suggestions were:

- ➢ Comments should be added to the source code
- ➢ Hints should be included to complete tasks
- ➢ It takes too much time to start the lab
- ➢ unnecessary/unused codes should be removed from the source code

- GUI is not fully responsive
- Additional challenges should be included.

Most of the suggestions were considered important and correction was made accordingly. The first feedback was important because without comments it is difficult to understand how the system was implemented and this might be one of the main issues why it took more time to complete the defense exercises. Because of the feedback, a brief explanation in comments was provided for all necessary lines of codes.

The second feedback was also very important because the main purpose of hints was to teach/help learners in case of any difficulty. However, as suggested by one of the pilot group, hints should not only provide a solution but also solutions with easy steps and best practices. Moreover, the purpose of VTA is to assist learners with hints and materials in case any help is required. Therefore, hints are included.

The third comment was about the time taken to start the lab content. Depending on the internet connection bandwidth, the lab might take different time to start the exercise. This is because when starting the lab, clean ubuntu 18.04 machines are cloned. After that VMs are cloned, all needed software (Nginx, MySQL, PHP etc) are installed on the machines and the GIT repositories are cloned. Then, all configurations are done (virtual host created for target, editor setup etc.). When everything is completed then the lab starts. In addition, the time taken to start this lab was tested with wifi connection and it takes not more than 1 min 20 seconds. Therefore, considering the time taking to start the lab, additional five more minute was added to the total time set to complete all the tasks.

The other relevant feedback was to remove unused codes which was left by mistake during the testing phase. Correction was made following the feedback . In addition, there was GUI issues and the necessary CSS was added to make it responsive. However, since the prototype was designed as a proof of concept to solve the problem statement, designing fancy GUI was not a priority, but it can be considered for future work.

The last feedback was to add more IDOR vulnerability on API request, POST method and session cookie tampering. However, owing to limited time and resources the last suggestion will be considered in the future work.

4.5.3 Feedback for the checker script

No one was able to find any major issue on the checker script, indicating that it was impossible for anyone to bypass the checker validation to get point without solving the given tasks. However, some minor improvement was suggested and, it will be considered in future work.

4.5.4 Evaluation with participants

In the second phase, the evaluation was conducted with a group of students in TalTech IT College who are enrolled for Web application Security course. In total, 26 students participated for this lab. To evaluate this work, the following steps were conducted: Pre-training assessment, post-assessment (reactions) and post-assessment (learning).

Pre-training assessment

The purpose of this assessment is understand student's previous experience for IDOR attack and defense. This also covers participants' current competencies and learning needs. The survey consists of the following questionnaire:

i) Are you familiar with IDOR(Insecure Direct Object References)?

(1) Extremely Familiar
(2) very familiar
(3) Familiar
(4) Not so familiar
(5) Not at all familiar

ii) To what extent are you confident about your skills to find IDOR vulnerability from PHP based web application?

(1) Extremely Familiar
(2) Very familiar
(3) Familiar
(4) Not so familiar
(5) Not at all familiar

iii) To what extent are you confident about your ability to fix IDOR vulnerability if it exists in PHP based web application?

(1) Extremely Familiar
(2) Very familiar
(3) Familiar
(4) Not so familiar
(5) Not at all familiar

The survey results indicate that majority of the participants have insufficient knowledge about IDOR. On the other hand, six participants were familiar with IDOR attacking techniques. However, they had insufficient knowledge for fixing the vulnerabilities and only three participants said they were very familiar both in finding and fixing IDOR vulnerabilities. Therefore, it can be concluded that the need for IDOR training was essential.

Post-assessment (reactions)

In this step, participants' reactions after exercising the hands-on lab was collected. These covers their experience to the training environment, learning objectives, reading materials, instructions & hints, and the overall satisfaction and comments. The complete survey questionnaires can be found in Appendix 4.

The survey results show that majority of the participants responds with positive satisfaction. A few participants also proposed to add more time for IDOR defense challenges and ten more minutes was added. It was also noted that those of participants with less experience in PHP had faced a difficulty to understand the prototype source codes. For this reason, additional hints were added in the source code to easily understand where the fixes should be implemented.

The time taken to complete all tasks vary from one student to another student. For those participants who attempted once, the time taken to complete all tasks was longer. However, for students who tried more than once, they were able to complete all tasks in less than 25 minutes. The main reason could be, in the attacking tasks, the stundets were able to memorize the solution and during their next attempts they could directly submit the solution and focus only on the defense tasks, which saved them lots of time.

The average time taken for participants to complete all tasks in the first attempt was less than 45 minutes. Which means the time assigned to this lab was good enough. Participants who attempted more than once had less time.

However, there is one student who took him more than three hours. There could be many reasons, but one good reason might be he started the lab and left before ending the mission, this means that the time will not stop counting until he end the mission. Participant's time duration to complete the lab was summarized in appendix 5.

Table 3 shows summary of feedbacks from the survey. Although the total student participant for this lab was 26, only 24 of them participanted for the survey.

Table 3: Summary for participants feedback

	Agree	Neither agree or disagree	Disagree	Strongly disagree	Not applicable
The instruction was clear and easily understandable	24	-	-	-	-
The reading material was good enough to complete the tasks	22	2	-	-	-
The hint provided was very helpful	19	-	4	1	-
The time given to complete each task was enough	20	-	4	-	-
The comments in the source code helps to understand the code easily	23	-	1	-	-
The overall learning objectives are met	23	-	1	-	-
It was possible to complete the defense challenges with basics skills of PHP programming language	18	2	4	-	-
Is it possible to complete the tasks without reading the provided reading materials?	2	1	2	19	-

Post-assessment (learning)

The survey questions from 9 -11 in appendix 4 were used to assess participants' knowledge gained from IDOR attacking and defending lab. Survey questionnaires can be found in Appendix 4 question 9 -11.

Compared with the pre-training survey, the final survey feedback showed tremendous improvement on participant's confidence on IDOR attack and defense skills. This proves that the training was both important and successful.

Contribution for a course curriculum

After incorporating the feedbacks collected from participants, this lab was suggested and accepted by TalTech IT college instructor (Andres Kaver) to include it to the web application security (I901) course curriculum.

5. Conclusions

Insecure direct object references vulnerability is one of the owasp Top 10 vulnerability lists since 2004. Although insecure DOR is not a new vulnerability, the impact of this vulnerability is significant when it occurs. The main purpose of this thesis is to create hands-on attack and defense lab for insecure direct object reference vulnerability that can be accessed regardless of time and location constraints.

The study in this work implicate that existing automated security tools are not complete solutions to find IDOR vulnerability. IDOR vulnerability exist in different web application unless the systems are manually tested.

This paper work covers the most common issues where IDOR exists and the various ways to fix the vulnerability using PHP web application. Three existing source code analysis tools (SonarQube, RIPs, Checkmarx) and blackbox testing tools (w3af, owasp zap, and nikito) were selected, and scans were conducted. The limitation of each existing automated solutions was observed. Among the chosen tools, Checkmarx was the only tool to find parameter tampering vulnerabilities. However, it was observed that it is possible to trick Checkmarx by using the encoded parameter.

It was suggested that awareness training can help to find and fix IDOR vulnerability. Existing attack and defense CTF challenges were selected for the study, and their functions and limitations for IDOR exercise was outlined.

In this study, the methodology used to create hands-on practical course was by using the ADDIE instructional design model, which consists of course analysis, design, and development, implementation, and evaluation. The need analysis, student minimum technical skillsets, and the assessment criteria were developed. The learning objectives were developed to design the course. Based on the learning objectives, the prototype was designed including, necessary materials, evaluation methods, and delivery methods. Finally, the lab architecture was designed in such a way that would also allow anyone to modify or contribute additional exercises for future work. The prototype also includes the process of adding new challenges.

The prototype was evaluated in two phases. The first evaluation was conducted with a pilot group who are experts in web application security, important feedback for the training exercise content and correctness was collected. After collecting important feedbacks, an immediate improvement was made to the prototype. In the second part, the lab exercise was evaluated with participants from TalTech IT college students who are studying the web application security course (**I901**). The purpose of the second phase evaluation was to assess participants' reactions after exercising the hands-on lab.

The survey result shows that the majority of the participants respond with positive satisfaction. Some minor feedbacks was considered for future works. After incorporating the feedbacks collected from participants, this lab was suggested and accepted by TalTech IT college instructor (Andres Kaver) to include it to the web application security (I901) course curriculum.

It was observed that IDOR vulnerability is easier to find in manual pentesting. However, when relying only on fully automated tools, there is a high probability of missing most of IDOR vulnerabilities. In this context, the practical hands-on lab provides the most effective method to find and fix IDOR vulnerability.

In conclusion, this thesis work contributes a web-based hands-on attack and defense lab for insecure direct object reference vulnerability that is suitable for academic purpose or self-improvement. A scalable web application prototype is implemented which allows existing scenarios to be reconfigured or allows for entirely new IDOR scenarios to be created. Most of publicly available CTF exercises are designed to improve penetration testing skill. However, this thesis work offers both attack and defense exercises for IDOR vulnerability. Furthermore, issues related to time and resource constraints are solved by integrating the prototype with RangeForce cloud-based system.

6. Suggestions for future work

The exercised is limited to a few exercises, therefore, it is important to define additional learning objectives and implement new IDOR related challenges for both attacking and defending challenges. In addition, as suggested by some of the pilot group, GUI can be re-modified in such a way that it impresses and attracts more learners.

In OWASP top 10 2017, IDOR and Missing function level access control are now merged together and called Broken Access control. Therefore, in the future work, it is possible to include additional challenges for missing function level access control vulnerability.

Although the attacking techniques/steps for IDOR are very similar in different programming languages, in the defense part, the syntax and built-in functions are quite different. Therefore, implementing the prototype in another web language such as java, python, asp.net are considered essential to teach practical defense in different programming languages.

The evaluation scripts are integrated with RangeForce and automatically points are given for student upon fixing the vulnerabilities. However, if the prototype is installed in localhost, the evaluation script must be executed manually. The automation of the manual work/execution is interesting to consider in future studies.

REFERENCES

[1] Xiaowei Li and Yuan Xue (2011). A Survey on Web Application Security. Department of Electrical Engineering and Computer Science. Vanderbilt University. [Retrieved June 02 , 2018]

[2] Open web application project (OWASP), Retrieved June 03, 2018, from https://www.owasp.org/index.php/Main_Page

[3] Top 10 2007-Insecure Direct Object Reference. (n.d.). Retrieved June 05, 2018, from https://www.owasp.org/index.php/Top_10_2010-A4-Insecure_Direct_Object_References

[4] Kumarshrestha, A., Maharjan, P. S., & Paudel, S. (2015). Identification and Illustration of Insecure Direct Object References and their Countermeasures. *International Journal of Computer Applications, 114*(18), 39-44. doi:10.5120/20082-2148

[5] Cyber Security Capture The Flag. (June 15, 2018). Retrieved from https://blogs.cisco.com/perspectives/cyber-security-capture-the-flag-ctf-what-is-it

[6] Prasad, P. (n.d.). *Mastering modern web penetration testing: master the art of conducting modern pen testing attacks and techniques on your web application before the hacker does !* Retrieved June 20, 2018.

[7] Medhat, E., Smith, A., T., M., & S. Web Applications Attacks: Insecure Direct Object Reference. Retrieved June 28, 2018, from https://latesthackingnews.com/2017/07/09/web-applications-attacks-insecure-direct-object-reference/

[8] Houston, S. (2017, November 09). How-To: Find IDOR (Insecure Direct Object Reference) Vulnerabilities for large bounty rewards. Retrieved July 05, 2018, from https://www.bugcrowd.com/how-to-find-idor-insecure-direct-object-reference-vulnerabilities-for-large-bounty-rewards/

[9] Metzger, M. (2017, March 15). Samsung Leaking Customer Information – Hacker Noon. Retrieved July 15, 2018, from https://hackernoon.com/samsung-leaking-customer-information-9b7e2dcb006d#.vvmbm3mkf

[10] Critical flaw in Yahoo allows Hacker to delete 1.5M records. (2014, March 01). Retrieved July 18, 2018, from http://securityaffairs.co/wordpress/22687/hacking/critical-flaw-yahoo.html

[11] Arthur, C. (2013, July 22). Apple Developer site hack: Turkish security researcher claims responsibility. Retrieved July 18, 2018, from https://www.theguardian.com/technology/2013/jul/22/apple-developer-site-hacked

[12] Robert Abela, R. (2017, December 18). OWASP Top 10 for 2017. Retrieved July 25, 2018, from https://www.netsparker.com/blog/web-security/owasp-top-10/

[13]Highbrow(June 27, 2016) Difference between missing function level access control and insecure direct object references. Retrieved Aug 5, 2018, from https://www.quora.com/What-are-the-differences-between-missing-function-level-access-control-and-insecure-direct-object-references

[14] Web Parameter Tampering.. Retrieved August 6, 2018, from https://www.owasp.org/index.php/Web_Parameter_Tampering

[15] Session Prediction.. Retrieved August 10, 2018, from https://www.owasp.org/index.php/Session_Prediction

[16] Vanja Suhina(2007) Exploiting and automated detection of vulnerabilities in web applications, Retrieved August 13, 2018 from http://www.zemris.fer.hr/~sgros/publications/diploma_thesis/suhina_vanja_seminar.pdf

[17] "CWE-639 - Security Database." *Security-Database*, Retrieved August 20, 2018 from www.security-database.com/cwe.php?name=CWE-639.

[18] Hui Wang. "Preventing Insecure Direct Object References in App Development." Retrieved August 21, 2018 from *http://www.cs.tufts.edu/Comp/116/Archive/fall2014/Hwang.pdf*.

[19] "Black Box Testing Vs. White Box Testing: Key Differences. Retrieved August 25, 2018 from www.guru99.com/back-box-vs-white-box-testing.html

[20] "About SonarQube source code analysis tool" *SonarQube*, Retrieved August 29, 2018 from www.sonarqube.org/about/

[21] "About RIPs source code analysis tool", RIPs Scan Retrieved September 3, 2018, from http://rips-scanner.sourceforge.net/

[22] "CxSAST Static Application Security Testing." Retrieved September 4, 2018, from *Checkmarx SAST*, www.checkmarx.com/products/static-application-security-testing/

[23] "w3af tool" Retrieved September 6, 2018, from http://w3af.org/

[24] Testing Tools, Zed Attack Proxy(ZAP). Retrieved September 8, 2018, from https://www.owasp.org/index.php/Appendix_A:_Testing_Tools

[25] Keane, J. C. (2012, October 26). Nikito Web Application Scanner. Retrieved September 10, 2018, from https://www.madirish.net/547

[26] Web Application Firewall. (2016, October 18). Retrieved September 11, 2018, from https://www.owasp.org/index.php/Web_Application_Firewall

[27] ModSecurity: Open Source Web Application Firewall.. Retrieved September 15, 2018, from https://www.modsecurity.org/

[28] Nbs-system.. Nbs-system/NAXSI. Retrieved September 15, 2018, from https://github.com/nbs-system/naxsi#we-need-your-help

[29] WebNight Web Application Firewall. Retrieved September 16, 2018, from http://www.aqtronix.com/?PageID=99

[30] Shadow Daemon WAF. Retrieved September 18, 2018, from https://shadowd.zecure.org/overview/introduction/

[31] Stohr-Hunt, P. M. (1996). An analysis of frequency of hands-on experience and science achievement. Journal of Research in Science Teaching, 33(1), 101-109.

[32] BWAPP. . Retrieved September 25, 2018, from https://sourceforge.net/projects/bwapp/

[33]OWASP Security Shepherd. Retrieved September 26, 2018, from https://www.owasp.org/index.php/OWASP_Security_Shepherd

[34] DVWA - Damn Vulnerable Web Application, Retrieved September 27, 2018, from http://www.dvwa.co.uk/

[35] OWASP-Webgoat, Retrieved September 28, 2018, from http://www.vulnspy.com/webgoat-8/webgoat_8:_a_deliberately_insecure_web_application/

[36] OWASP Juice Shop. Retrieved September 30, 2018, from https://www.owasp.org/index.php/OWASP_Juice_Shop_Project

[37] What does an Instructional Designer do? (2014, August 19). Retrieved October 2, 2018, from https://vivaelearning.com/what-does-an-instructional-designer-do/

[38] Treser, M. (2017, July 20). Getting To Know ADDIE. Retrieved October 6, 2018, from https://elearningindustry.com/getting-know-addie-analysis

[39] ADDIE Model. Retrieved October 6, 2018, from https://www.instructionaldesign.org/models/addie/

[40] Kurt, S. (2017, August 29). Retrieved November 8, 2018, from https://educationaltechnology.net/the-addie-model-instructional-design/

[41] Durak, Gürhan, and Murat Ataizi. "The ABCs of Online Course Design According to Addie Model." *Universal Journal of Educational Research*, vol. 4, no. 9, 2016, pp. 2084–2091., doi:10.13189/ujer.2016.040920.

APPENDICES

Appendix 1: Solution checker script for task one

```
import requests
 s = requests.Session()
 url = 'http://192.168.6.2/index.php'
 values = {
                'login': 'letmein',
                'username': 'ephrem',
                'password': 'ephrem',
                'role' : 'student'
 }
 r = s.post(url, data=values)
 #print(r.url)
 url2 = 'http://192.168.6.2/result.php?rollid=2&class=1'
 r2 = s.get(url2)
 #print(r2.status_code)
 if 'syntax error' in str(r2.content):
                print('1 You have not fixed the vulnerable Yet! :(')
                exit(1)
 elif str(r2.content) == '':
                print('Content is empty :(')
                exit(1)
 elif 'Abeni' not in str(r2.content) and str(r2.content) != '':
                print(' Congratulations, You have Successfully fixed the issue! :)')
                exit(0)
 else:
                print('3You have not fixed the vulnerable Yet! :(')
                exit(1)
```

Appendix 2: Solution checker script for task two

```
import requests
    s = requests.Session()
    url = 'http://192.168.6.2/index.php'
    values = {
                'login': 'letmein',
                'username': 'ephrem',
                'password': 'ephrem',
                'role' : 'student'
    }
    r = s.post(url, data=values)

    url3 = 'http://192.168.6.2/profile.php?profile=MQ=='
    check = s.get(url3)
    url2 = 'http://192.168.6.2/profile.php?profile=Mg=='
    r2 = s.get(url2)
    if 'Abeni' in str(r2.content):
                print('You have not fixed the vulnerable Yet! :(')
                exit(1)
    elif 'Ephrem' not in str(check.content):
                print('You have not fixed the vulnerable Yet! :(')
                exit(1)
    elif 'syntax error' in str(r2.content):
                print('You have not fixed the vulnerable Yet! :(')
                exit(1)
    elif str(r2.content) == '':
                print('2You have not fixed the vulnerable Yet! :(')
                exit(1)
    else:
                print('Congratulations, You have Successfully fixed the issue! :)')
                exit(0)
```

Appendix 3: Solution checker script for Task three

```
import mysql.connector
import requests
connection = mysql.connector.connect(
 host="192.168.6.2",
 user="root",
 passwd="rootroot",
 database="srms"
)
sql= """delete from comment where 1"""
mycursor = connection.cursor()
mycursor.execute(sql)
connection.commit()
sql= """INSERT INTO comment (id, studentId, comment, Name) VALUES (1, 1,
'Student1 Comment', 'Ephrem'), (2, 2, 'Student2 Comment', 'Abeni')"""
mycursor = connection.cursor()
mycursor.execute(sql)
connection.commit()
print(mycursor.rowcount, "record inserted.")
s = requests.Session()
url = 'http://192.168.6.2/index.php'
values = {
                'login': 'letmein',
                'username': 'ephrem',
                'password': 'ephrem',
                'role' : 'student'
}
r = s.post(url, data=values)
url = 'http://192.168.6.2/deletecomment.php?commentid=2'
url = s.get(url)
if 'syntax error' in str(url.content):
                print('1 You have not fixed the vulnerable Yet! :(')
                exit(1)
```

```
elif str(url.content) == '':
        print('2You have not fixed the vulnerable Yet! :(')
        exit(1)
elif 'ABENI' in str(url.content):
        print('Congratulations, You have Successfully fixed the issue! :)')
        exit (0)
else:
        print('You have not fixed the vulnerable Yet! :(')
        exit (1)
```

Appendix 4: Survey questions

1. The instruction was clear and easily understandable
    a. Agree
    b. Neither agree or disagree
    c. Disagree
    d. Strongly disagree
    e. Not applicable
2. The reading material was good enough to complete the tasks
    a) Agree
    b) Neither agree or disagree
    c) Disagree
    d) Strongly disagree
    e) Not applicable
3. The hint provided was very helpful
    a) Agree
    b) Neither agree or disagree
    c) Disagree
    d) Strongly disagree
    e) Not applicable
4. The time given to complete each task was enough
    a) Agree
    b) Neither agree or disagree
    c) Disagree
    d) Strongly disagree
    e) Not applicable
5. The given comment in the source code helps to understand the code easily
    a) Agree
    b) Neither agree or disagree
    c) Disagree
    d) Strongly disagree
    e) Not applicable
6. The overall learning objectives are met

a) Agree
b) Neither agree or disagree
c) Disagree
d) Strongly disagree
e) Not applicable

7. It was possible to complete the defense challenges with basics skills of PHP programming language
    a) Agree
    b) Neither agree or disagree
    c) Disagree
    d) Strongly disagree
    e) Not applicable
8. Is it possible to complete the exercise without referencing the provided reading materials?
    a) Agree
    b) Neither agree or disagree
    c) Disagree
    d) Strongly disagree
    e) Not applicable
9. How do you feel about the training content?
    a) Extremely interested
    b) Very Interested
    c) Somewhat interested
    d) Not so interested
    e) Not at all interested
10. How confident are you with your skills for the basics of IDOR attack and defense?
    a) Extremely confident
    b) Very confident
    c) Somewhat confident
    d) Not so confident
    e) Not at all confident
11. Other feedback

Appendix 5: Participant activity

Lab count	IDOR Lab – first finish	Attempts	Progress	Duration	Last activity
1	2018-11-23 14:10:48	2	100	299.951	2018-11-23 14:10:48
1	2018-11-23 13:34:56	3	100	367.682	2018-11-23 13:34:56
1	2018-11-29 19:39:04	2	100	524.276	2018-11-29 19:39:04
1	2018-11-23 13:59:02	2	100	557.808	2018-11-23 13:59:02
1	2018-11-23 13:39:38	7	100	675.993	2018-11-23 13:39:38
1	2018-11-23 15:26:32	4	100	991.635	2018-11-23 15:26:32
1	2018-11-23 12:48:04	1	100	1619.57	2018-11-23 12:48:04
1	2018-11-23 12:52:00	1	100	1701.13	2018-11-23 12:52:00
1	2018-11-23 14:03:35	1	100	1719.143	2018-11-23 14:03:35
1	2018-11-23 14:32:06	1	100	1754.463	2018-11-23 14:32:06
1	2018-11-14 8:12:16	62	100	1885.07	2018-11-28 10:38:03
1	2018-11-23 12:58:58	1	100	1889.818	2018-11-23 12:58:58
1	2018-11-23 13:17:50	1	100	2514.536	2018-11-23 13:17:50
1	2018-11-23 13:50:51	1	100	3610.838	2018-11-23 13:50:51
1	2018-11-23 13:36:46	1	100	4020.929	2018-11-23 13:36:46
1	2018-11-23 13:37:04	1	100	4056.535	2018-11-23 13:37:04
1	2018-11-29 19:29:10	1	100	4286.161	2018-11-29 19:29:10
1	2018-11-23 15:50:32	1	100	4679.647	2018-11-23 15:50:32
1	2018-11-23 17:52:02	1	100	13504.304	2018-11-23 17:52:02

Appendix 6: Participant feedbacks

The first part was easy, the second a bit harder because of PHP.
The lab is useful, specially in the PHP part, because is the part that really must be done in order to prevent cyber attacks. On the the other hand my PHP knowledge is not so good, and therefore some more hints would make it easier.

The difficulty level was quite ok. The hints in the text for the implementation of the countermeasures and the comments in the source code helps a lot. For non web devs and also for Junior devs, it is perfect to show why it is important to do such checks and how to implement it right. I liked the example with the base64/UTF encodings, because I saw it that often in the internet and the even think it is save till now. The questions and the part for the solution are one time the same. So the same text is mentioned at both parts. In the end after finishing the mission, it showed me that I could redo it, but I finished it with 100%. The file explorer opened at least after any refreshes, was not clear where is it and how to "activate" it. All in all it shows different ways of IDOR and was funny to hack and fix it.

The lab was challenging and okay. A bit hard for me as my experience with backend language is mostly with Java not with PHP but I managed to solve it. Also I learnt about using encoding in protecting the URL input (although, as we saw, it can be also vulnerable to abuse).

I participated in the IDOR lab and finished it. The lab I would say it's not either too hard or easy:
it depends how much you know the Insecure Direct Object Reference topic but still I think that even though I have average knowledge I would not say the lab is too easy. I would give a recommendation to the author to separate the html files with php files and make more php files for its php codes after it is easier to have that instead of having in one file both HTML and PHP code. overall the lab was well planned and well implemented for example, I did not have any issue or bug regarding when a task should get completed which is the case with some other labs

It is a good starting point to learn to snoop around when it comes to finding security loopholes in web application, the interface looks really nice and appealing. I would appreciate if more concise material that isn't necessarily lengthy but packed with great resource about best practice

when it comes to web application security be added. In all it was a good ride and awesome experience.

The lab was the medium difficulty, however, introduced the main concepts of IDOR and challenged to fix the issues. Different parts of the lab helped to understand and discover the vulnerabilities and consequences and different ways of solving the problem.
The user interface of the lab was somewhat confusing because the browser window was not wide enough to display the navigation menu, but opening browser page from the lab in the new window helped. Also, the comment page was initially empty, thus the task to delete other comments was hard to validate (only the flag was displayed). Overall, I really enjoy rangeforce labs!

This test has the expected level of difficulty, neither too hard nor too easy. It was globally interesting.

The lab is interesting but it will be hard for me to judge the difficulty because I already know all those things. From my own experience exploitations tasks were easy and patching the code was ok, maybe because the comments in the code are helping too much.

The rangeforce lab was loading for long time, But the exercise was nice. :)