UNIVERSITY OF THESSALY

SCHOOL OF SCIENCES
DEPARTMENT OF COMPUTER SCIENCE AND BIOMEDICAL INFORMATICS

PROGRAM OF UNDERGRADUATE STUDIES

DIPLOMA THESIS

Development of an introductory Cyber Security training program

Aristeidis T. Anastou

LAMIA

JUN 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΒΙΟΙΑΤΡΙΚΗ

ΠΡΟΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία υλικού και τεκμηρίωση σχετικα με την εισαγωγική εκπαίδευση σε θέματα ασφάλειας υπολογιστών

Αριστείδης Θ. Ανάστου

ΛΑΜΙΑ

ΙΟΥΝΙΟΣ 2020

BACHELOR THESIS

Development of an introductory Cyber Security training program

Aristeidis T. Anastou

SUPERVISOR
George Spathoulas
Member of laboratory teaching stuff
University of Thessaly

THREE-MEMBER ADVISORY COMMITTEE:

George Spathoulas
Member of laboratory teaching stuff
University of Thessaly

Ioannis Anagnostopoulos
Assistant Professor
University of Thessaly

Athanasios Kakarountas
Associate Professor
University of Thessaly

Examination Date: June 10, 2020

ABSTRACT

Every day we are using technology without thinking, that in many cases our lives depending on it. Cyber crime is on it's rise and the information security industry is in the need of security professionals. Goverments, companies, enterprises and individuals are victims by cyber attacks and scams. Education and staff training is more accessible and affordable with new apps and education platforms. In this paper we propose a platform to introduce students and individuals in the basic concepts of cyber security and penetration testing. Our educational platform is referred in the most known categories of ethical hacking, including type of vulnerabilities and attacks, regarding websites, platforms, apps, cryptographic algorithms and networks. The process that the user learns to attack different vulnerabilities is by choosing the category he wants to learn and start hacking his way through the existing levels. In each level the difficulty is increasing with defences, countermeasures and filters where the user have to bypass them. With this way the player learns by hands on experience, the use of different tools, operating system distributions, ways to identify vulnerabilities, take advantage of them as an attacker and defend them as the defender. For evaluation the user anwers fifty questions before doing the lab and after finish it in order to see his evolution in knowledge.

SUBJECT AREA: Cyber Security training

KEYWORDS: Education, Penetration Testing, Information Security

ΠΕΡΙΛΗΨΗ

Η ψηφιακή τεχνολογία όλο και γίνεται αναπόσπαστο κομμάτι της καθημερινότητας μας. Το κυβερνοέγκλημα αύξανεται κατακόρυφα και μαζί η ανάγκη για επαγγελματίες στο χώρο της ασφάλειας πληροφοριών. Κράτη, εταιρίες, μικροεπιχειρήσεις, ιδιώτες πέφτουν κάθε μερα θύματα κυβερνοεπιθέσεων και εξαπάτησης. Η εκπαίδευση και η κατάρτιση νέων εργαζόμενων γίνεται όλο και πιο προσιτή με διάφορες νέες εφαρμογές, τρόπους εκπαίδευσης και πλατφόρμες. Στο παρακάτω κείμενο παρουσίαζουμε ένα τρόπο για την εισαγωγή σε έννοιες και την εκπαίδευση πάνω σε αυτές, σε θέματα ασφαλείας υπολογιστών. Στην προκειμένη περίπτωση θα αναφερθούμε σε φοιτητές ως αντικείμενα έρευνας αλλά αυτό δεν αποκλείει την ενασχόληση οποιοδήποτε ατόμου που θέλει να ξεκινήσει να εκπαιδεύεται σε θέματα ασφαλείας. Η συγκεκριμένη πλατφόρμα εκπαίδευσης αναφέρεται στις κύριες κατηγορίες ψηφιακών ευπαθειών, επιθέσεων οι οποίες εμπίμπτουν σε ιστοσελίδες, διαδυκτιακές πλατφόρμες, εφαρμογές, κρυπτογραφίκους αλγόριθμους και δίκτυα. Ο τρόπος με τον οποίο καλείται ο κάθε χρήστης να λύση κάθε δοκιμασία και να επιτεθεί(ηθικά) με επιτυχία σε έναν στόχο είναι σαν ψηφίακο παιχνίδι κοινώς CTF(Capture The Flag). Σε κάθε κατηγορία υπάρχουν διάφορες ευπαθείες και στάδια. Σε κάθε στάδιο η δυσκολία αυξάνεται βάζοντας νέες άμυνες, μέτρα προστασίας και φίλτρα, όπου ο χρήστης πρέπει να προσπεράσει. Με αυτόν τον τρόπο ο παίχτης εξοικειώνεται πρακτικά με τη χρήση εργαλείων και λετουργικών συστημάτων χάκινγκ, μαθαίνει να αναγνωρίζει ευπάθειες στα πληροφοριακά συστήματα, τρόπους να τις εκμεταλλεύεται σαν επιτιθέμενος και να τις προλαμβάνει σαν αμυνόμενος. Τέλος ο χρήστης καλείται να απαντήσει σε ενα ερωτηματολόγιο σχετικό με θέματα ασφάλειας πριν ξεκινήσει τις δοκιμασίες και αφού τις τελειώσει ώστε να αυτοαξιολογηθεί βλέποντας την εξελιξή του σε γνώσεις.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Εκπαίδευση σε θέματα ασφάλειας συστημάτων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Εκπαίδευση, Κυβερνοασφάλεια, Ασφάλεια Πληροφοριακών Συστημάτων

ACKNOWLEDGMENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 The problem

Computers and Internet is becoming an inseparable part of our daily lives. Cyber crime is on its rise causing huge problems for society and the need for cyber security professionals is mandatory.But why does cyber security matter much. Cyber crime is borderless and is not targeting only individuals,but companies,industries and governments. The increasing cost of cyber crime especially its role in terrorism makes it one of the biggest threats globally according to PwC [61] [58].

Just in the last few years hundreds of millions of credit cards numbers have been stolen.In 2015, 220 million was lost to wire fraud in the United States. In 2019, losses surpassed 1.5 billion, according to WFG National Title Insurance Company[3]. Tens of millions of national security numbers and health records are compromised.Even nuclear centrifuge have been hacked and unmanned aerial drones have been hijacked [86].This is all done by exploiting vulnerabilities in software or hardware [3]. Every day, the AV-TEST Institute registers over 350,000 new malicious programs (malware) and potentially unwanted applications (PUA). [60] Recent data breaches show that businesses of all types, sizes and in all locations are at real risk of a cyber attack at any given moment.

Many data breaches result from phishing scams that introduce malware into network systems[87]. Educating employees regarding the latest tactics used by scammers can help reduce the likelihood that they will click links that expose them to malicious software.[2]Implementing basic data security policies that explain how to properly handle company data is also key to reducing the threat of internal misuse. Organizations should also be more strict about who has access to sensitive data in the first place. These strategies can greatly reduce the impact of human error on cyber security. measures.[45]For example, the UK Cyber Security Breaches Survey 2018, carried out by the UK government and Portsmouth University found that 43percent of UK businesses have experienced a cyber security breach or attack over the last 12 months, with only 20percent of UK companies offering training to staff within the same time frame.

Usually companies have two teams.The RED team which is responsible for attacking the systems and expose vulnerabilities and the BLUE team which is responsible for defending the systems and deploy security patches for vulnerabilities that the red team

exploited.Another way to defend cyber attacks is to seek for information security specialists.These are trained people responsible for designing,testing, implementing and monitoring security measures for companies systems.

Here we propose a method for educating students, security enthusiasts and employees in the field of cyber security introducing them in the basic concepts of information security, vulnerabilities, attack vectors, exploitation techniques and ways to defend against these attacks.

1.2 What is hacking

Hacking is identifying weaknesses in information systems or networks take advantage of them bypass security mechanisms and gain unauthorized access.[26]

1.3 What is a hacker

In computer security, a hacker is someone who focuses on security mechanisms of computer and network systems[53]. While including those who endeavor to strengthen such mechanisms, it is more often used by the mass media and popular culture to refer to those who seek access despite these security measures. That is, the media portrays the 'hacker' as a villain. Nevertheless, parts of the subculture see their aim in correcting security problems and use the word in a positive sense. White hat is the name given to ethical computer hackers, who utilize hacking in a helpful way. White hats are becoming a necessary part of the information security field.[8] They operate under a code, which acknowledges that breaking into other people's computers is bad, but that discovering and exploiting security mechanisms and breaking into computers is still an interesting activity that can be done ethically and legally. Accordingly, the term bears strong connotations that are favorable or pejorative, depending on the context.

A white hat hacker breaks security for non-malicious reasons, either to test their own security system, perform penetration tests, or vulnerability assessments for a client - or while working for a security company which makes security software. The term is generally synonymous with ethical hacker, and the EC-Council,[18] among others, have developed certifications, courseware, classes, and online training covering the diverse arena of ethical hacking.[17]

A black hat hacker is a hacker who ”violates computer security for little reason beyond maliciousness or for personal gain” (Moore, 2005).[19] The term was coined by Richard Stallman, to contrast the maliciousness of a criminal hacker versus the spirit of playfulness and exploration in hacker culture, or the ethos of the white hat hacker who performs hacking duties to identify places to repair or as a means of legitimate employment.[20] Black hat hackers form the stereotypical, illegal hacking groups often portrayed in popular culture, and are ”the epitome of all that the public fears in a computer criminal”.[21]

A grey hat hacker lies between a black hat and a white hat hacker. A grey hat hacker may surf the Internet and hack into a computer system for the sole purpose of notifying the administrator that their system has a security defect, for example. They may then offer to correct the defect for a fee.[21] Grey hat hackers sometimes find the defect of a system and publish the facts to the world instead of a group of people. Even though grey hat hackers may not necessarily perform hacking for their personal gain, unauthorized access to a system can be considered illegal and unethical.

A social status among hackers, elite is used to describe the most skilled. Newly discovered exploits circulate among these hackers. Elite groups such as Masters of Deception conferred a kind of credibility on their members.

A script kiddie (also known as a skid or skiddie) is an unskilled hacker who breaks into computer systems by using automated tools written by others (usually by other black hat hackers), hence the term script (i.e. a computer script that automates the hacking) kiddie (i.e. kid, child—an individual lacking knowledge and experience, immature),[23] usually with little understanding of the underlying concept.

A hacktivist is a hacker who utilizes technology to publicize a social, ideological, religious or political message.

Hacktivism can be divided into two main groups:

Cyberterrorism — Activities involving website defacement or denial-of-service attacks; and, Freedom of information — Making information that is not public, or is public in non-machine-readable formats, accessible to the public. [28]

1.4 Cyber Security Careers Paths

- Chief Infosec Officer

- Cryptographer
- Forensics Expert
- Incident Responder
- Penetration Tester
- Security Administrator
- Security Analyst
- Security Architect
- Security Auditor
- Security Consultant
- Security Director
- Security Engineer
- Security Manager
- Security Software Developer
- Security Specialist
- Security Code Auditor
- Vulnerability Assessor

Different titles but very common jobs.Everybody working in the cyber security industry share the same target.Make more secure the world of information technology. [20]

1.4.1 Information Security Specialist

Daily tasks of security specialist are

- Analyze and establish security requirements for your systems/networks
- Defend systems against unauthorized access, modification and/or destruction

- Configure and support security tools such as firewalls, anti-virus software, patch management systems, etc.
- Define access privileges, control structures and resources
- Perform vulnerability testing, risk analyses and security assessments
- Identify abnormalities and report violations
- Oversee and monitor routine security administration
- Develop and update business continuity and disaster recovery protocols
- Train fellow employees in security awareness, protocols and procedures
- Design and conduct security audits to ensure operational security
- Respond immediately to security incidents and provide post-incident analysis
- Research and recommend security upgrades
- Provide technical advice to colleagues

According to Payscale there are two categories for IT Security Specialists:

- The median salary for an Information Security Specialist is 75,263 (2019 figures). Overall, you can expect to take home a total pay of 47,177 – 119,556.
- The median salary for a Computer Security Specialist is 72,223 (2019 figures). Overall, you can expect to take home a total pay of 39,920 – 107,887.

1.4.2 Penetration testers

A Penetration Tester (a.k.a. Pen Tester or Ethical Hacker) probes for and exploits security vulnerabilities in web-based applications, networks and systems. Simply put, you get paid to legally hack. In this fascinating job, you get to use a series of penetration tools – some predetermined, some that you design yourself – to simulate real-life cyber attacks. Your ultimate aim is to help an organization improve its security. Daily tasks of penetration tester are:

- Perform formal penetration tests on web-based applications, networks and computer systems
- Conduct physical security assessments of servers, systems and network devices
- Design and create new penetration tools and tests
- Probe for vulnerabilities in web applications, fat/thin client applications and standard applications
- Pinpoint methods that attackers could use to exploit weaknesses and logic flaws
- Employ social engineering to uncover security holes (e.g. poor user security practices or password policies)
- Incorporate business considerations (e.g. loss of earnings due to downtime, cost of engagement, etc.) into security strategies
- Research, document and discuss security findings with management and IT teams
- Review and define requirements for information security solutions
- Work on improvements for security services, including the continuous enhancement of existing methodology material and supporting assets
- Provide feedback and verification as an organization fixes security issues

According to Payscale, the median salary for a Penetration Tester is 81,356 (2019 figures). Overall, you can expect to take home a total pay of 49,252 – 134,946. This includes your base annual salary, bonuses, profit sharing, tips, commissions, overtime pay and other forms of cash earnings, as applicable.

1.4.3 Bug bounty

Identifying vulnerabilities in software code can be a tedious and time-consuming process. Many organizations simply don't have the resources to subject their programs to the rigorous scrutiny necessary to identify every single bug or loophole that could be exploited by hackers. In recent years, however, companies have decided to outsource this task through "bug bounty" programs.

These programs encourage well-intentioned hackers to scour web-based software for vulnerabilities and errors, delivering a cash payout when confirmed bugs are identified. Both private companies and government agencies have implemented “bug bounty” policies to help shore up their software security. [77][50][31] There are a lot of resources and platforms on the internet that can teach users the concepts of hacking,pen-testing and red teaming.Also give them except from theory,practical knowledge with a technical experience through challenges and vulnerable machines. [54]

Acording to HackerOne Reduced Traditional Pen Testing Costs HackerOne Challenge costs can be offset, partially or wholly, by reduced cost for other pen testing service providers and internal effort. Interviewees represented a wide range of experiences, from eliminated costs being many times more than what was being paid to HackerOne to HackerOne being slightly more expensive. Notably, organizations can complete Challenges much faster than traditional pen testing, which saves cost and effort and improves remediation. All interviewees viewed the costs being paid to HackerOne for these Challenges as negligible compared to the benefits associated with improved security (discussed later in the study). From the interviews, Forrester heard:“Previously we had scanners-as-a-service doing static and dynamic code scanning. It found some bugs but was 10x to 15x more expensive per bug found and didn’t find everything.” “The comparable work we did in the pilot would have cost six times more with our previous service provider, and they gave us much fewerfindings.”“HackerOne is a much better cost model than red-team pen testing. It is far cheaper to run bug bounties than to do traditional pen testing. [57] And you get much better results.” “Every 1 dollar we spend on HackerOne Challenges would have meant 5 dollars in the past for other pen testing and auditors.”“We avoid the one to two months preplanning that was needed to make sure all system owners were onboard.”“They take care of the triage, which saves my team time. HackerOne tries to recreate the bugs, which is very helpful. Sometimes hackers file a bug, which is just a misunderstanding.”“HackerOne does so much of the overhead work. They provide the online submissions platform, manage the researchers, and provide industry best practices on structuring the programs and making payments. It eliminated the need to recreate the wheel.” For the financial analysis, Forrester assumes:6 | The Total Economic ImpactTM Of HackerOne Challenge The table above shows the total of all benefits across the areas listed below, as well as present values (PVs) discounted at 10 percent. Over three years,the composite organization expects risk-adjusted total benefits to be a PV of 541,577. “If you break it down as bounty payouts compared to the quality of vulnerabili-

ties found and time saved, HackerOne is a much better ROI compared to traditional pen testing companies [98].” Digital services expert, government› To be conservative, the cost for a single pen test from the previous provider was the same for a

HackerOne Challenge with optional Compliance add-on. The organization previously conducted only one pen test per year. Therefore, the savings shown here are equal to one of the two HackerOne Challenges shown in the Analysis Of Costs section of the study. Previous pen testing took two months and required 0.75 of an FTE to manage the testing internally and lead triaging efforts. (The HackerOne-related internal effort, which replaced this is shown in the Analysis Of Costs section of the study.)The amount of cost and effort expended on previous testing can vary widely from greater to less than HackerOne based on the nature of the systems and information being tested and the high-profile status of a company. To account for these risks, Forrester adjusted this benefit downward by 10 percent, yielding a three-year risk-adjusted total PV of 156,784. [16]

2. BACKGROUND AND RELATED WORK

An article from infoworld 36 by Robert Lemos presents the problem that the most of graduate university students have a good understanding of computer science and software engineering but they will lack a solid understanding of computer security and how to make their applications secure. ”Most top computer science programs don’t require students to learn the fundamentals of secure programming an oversight that will continue to hurt application security in the future”, said David Koretz, CEO of security firm Mykonos Software.Most of the universities have only one subject of security in their semesters.

Many companies stated corporate with undergraduate programs to educate students and develop their skills in a security oriented knowledge. Not only wannabe hackers have to build their skills in information security but anyone who wants to be on the top list of candidates in the workforce.A proposed method from big companies and governments is besides the theoretical level also a solid practical experience in exploring security issues, identifying them, and exploit them. Previous approaches with the similar or same techniques were represented by SEED project [24]. They developed many labs for different categories and vulnerabilities like CVE vulnerabilities,known attacks, software security, system security, network security, web security, access control, authentication, cryptography, etc. Their labs are built upon a Linux virtual machine can be used for security learning with a hands on experience. The SEED project has been funded by 3 NSF grants with a total budget of 1.3 million dollars and have been used by over 250 institutes in 30 countries. A publication says that: ”An integrated cybersecurity workforce includes technical and nontechnical roles that are staffed with knowledgeable and experienced people. An integrated cybersecurity workforce can address the cybersecurity challenges inherent to preparing their organizations to successfully implement aspects of their missions and business processes connected to cyberspace.” [70].

A publication by Fred B. Schneider [79] makes a case study about the cybersecurity education in universities and how the teaching methods have to evolve in order to give the students the critical thinking and maturity for the cybersecurity industry. A proposed method from Ronald S. Cheung is a ”Challenge Based Learning (CBL) methodology to cybersecurity education” [13]. The evaluation was made by the students self reporting their skills in computers and security before and after the study.An increasing tend shown in their interest in computer security after completing the labs.An interactive class where the students have to collaborate with their colleagues apply their knowledge

in real world scenarios of cyber attacks and defend against them. YesWeHack 35, a European bug bounty platform, is providing universities and schools with free access to its educational platform YesWeHackEDU. This offer aims to allow educational institutions to hold a practice-oriented cybersecurity training. As of 1st April 2020, all universities and schools can benefit from free licenses of YesWeHackEDU, which are valid until 31st May 2020.Another implemetation in order to enable students to practice IT security anytime anywhere, the Telematics research group of the University of Trier, Germany, is developing an on-line tutoring system, called Tele-Lab IT Security. First, it is a web-based tutoring system which introduces students to fundamental IT security concepts. Second, it provides an on-line virtual laboratory in which students are able to gain practical experience. Based on virtual machine technology, a remote machine can be assigned to a student and administrator rights can be granted to him without endangering the stability and security of the tutoring system itself [34].

A paper by Li-Chiou Chen and Chienting Lin from Pace university propose the method combining theoritical and practical exercise. The students have to complete a hands-on assignments after every lecture. They are downloading an instructions file to follow as doing the assignment step by step. In order to evaluate how effective our students think the courseware is, they solicited students' opinions using an anonymous questionnaire at the end of a semester [12].

Xinli Wang published a paper and the proposed method for understanding in depth the fundamentals of IT security is a hands on experience on a lab environment. The set up of the lab is on a virtual machine using commercial software and covers a variety of categories and vulnerabilities in network security, computer security application security and cryptography. In each category the lab is build in a way to describe first and introduce the user into the topic and then make a walkthrough of the problem and how a malicious user can attack it. The evaluation of the labs were made with students submit a survey questionnaire voluntarily.The results was positive and students show that they enjoyed the process.[93]

SWEET project [11] developed a collection of modules, projects and lab assignments for secure web development. Security components, including modules and labs, are "injected" to existing courses at Towson University [89]. Most of the universities understand the need of the workforce for graduates with high education and solid understanding of IT security principles and a real world experience. That's why news methods and proposes are developed to make this process valuable and more easy to understand.

3. UTH PENTESTBOX

This paper presents a method to introduce students into the concept of ethical hacking and penetration testing.It's an education lab that contains different categories of information security challenges and each category consists of different levels with the difficulty increasing as go through them.Students have the opportunity with this lab to learn the basic types of vulnerabilities(OWASP top 10)[96] and explore advance exploitation techniques. [72]. UTH pentestbox is CTF based lab where users in every category in each level they have to exploit the vulnerability and get the flag. The lab is using a LAMP stack (Linux, Apache, MySQL, PHP) and it is build using a Ubuntu docker image.

But what is a docker?

Docker is a command-line program, a background daemon, and a set of remote services that take a logistical approach to solving common software problems and simplifying your experience installing, running, publishing, and removing software.It accomplishes this using a UNIX technology called containers.Containers are an abstraction at the app layer that packages code and dependencies together. 3.1

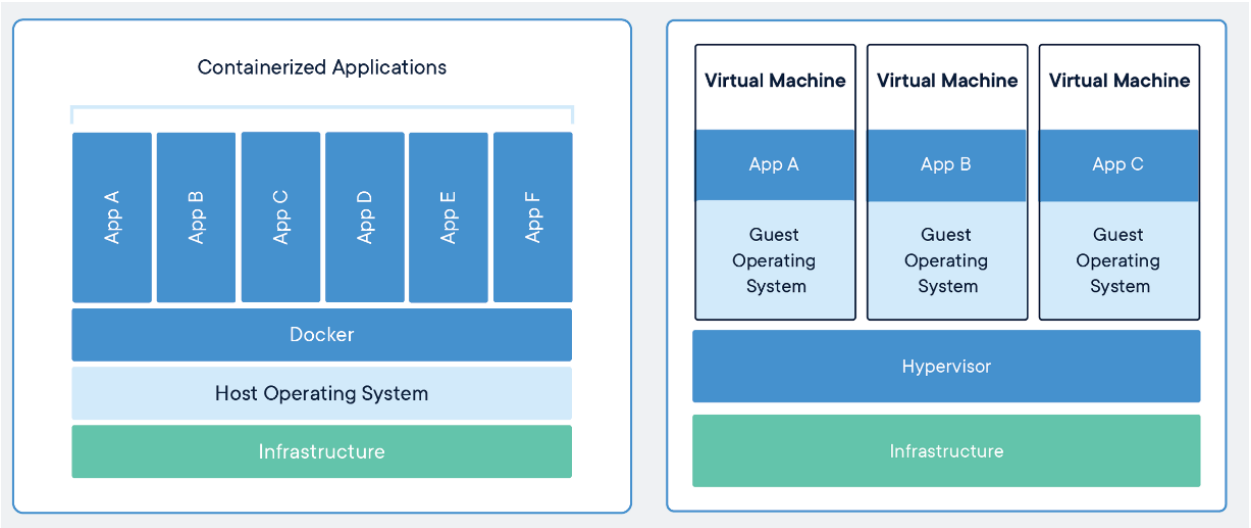

Figure 3.1: Structure of platform

Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.Virtual machines (VMs) are

an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

The final lab that the user has to run is a docker image which is loaded to the host system after installing the docker service. We propose the user to use a specific distribution of Linux Kali Linux 38 which is a Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing. Kali contains several hundred tools which are geared towards various information security tasks, such as Penetration Testing, Security research, Computer Forensics and Reverse Engineering. Kali Linux is developed, funded and maintained by Offensive Security 37, a leading information security training company. Users also can build a docker container with Kali Linux to do the hard work for them and don't have to use virtual machines or troubleshoot any installations. Below is a list with the tools that will be used in the lab for enumeration,exploitation and for exploit development.

- nmap

  Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

- wget

  GNU Wget is a free utility for non-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies.

- curl

  curl is a tool to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS,IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET

and TFTP). The command is designed to work without user interaction.

- hydra

  Hydra is a parallelized login cracker which supports numerous protocols to attack. New modules are easy to add, beside that, it is flexibleand very fast. This tool gives researchers and security consultants the possibility to show how easy it would be to gain unauthorized access from remote to a system.

- john

  john, better known as John the Ripper, is a tool to find weak passwords of users in a server. John can use a dictionary or some search pattern as well as a password file to check for passwords. John supports different cracking modes and understands many ciphertext formats, like several DES variants, MD5 and blowfish. It can also be used to extract AFS and Windows NT passwords.

- nikto

  Examine a web server to find potential problems and security vulnerabilities, including:

  - Server and software misconfigurations
  - Default files and programs
  - Insecure files and programs
  - Outdated servers and programs

  Nikto is built on LibWhisker (by RFP) and can run on any platform which has a Perl environment. It supports SSL, proxies, host authentication, IDS evasion and more. It can be updated automatically from the command-line, and supports the optional submission of updated version data back to the maintainers.

- dirbuster

  Dirbuster is a Web Content Scanner. It looks for existing (and/or hidden) Web Objects. It basically works by launching a dictionary based attack against a web server and analyzing the response.

- wfuzz

Wfuzz has been created to facilitate the task in web applications assessments and it is based on a simple concept: it replaces any reference to the FUZZ keyword by the value of a given payload.

A payload in Wfuzz is a source of data.

This simple concept allows any input to be injected in any field of an HTTP request, allowing to perform complex web security attacks in different web application components such as: parameters, authentication, forms, directories/files, headers, etc.

Wfuzz is more than a web content scanner:

- Wfuzz could help you to secure your web applications by finding and exploiting web application vulnerabilities.
- Wfuzz's web application vulnerability scanner is supported by plugins.
- Wfuzz is a completely modular framework and makes it easy for even the newest of Python developers to contribute. Building plugins is simple and takes little more than a few minutes.
- Wfuzz exposes a simple language interface to the previous HTTP requests/responses performed using Wfuzz or other tools, such as Burp. This allows you to perform manual and semi-automatic tests with full context and understanding of your actions, without relying on a web application scanner underlying implementation.

It was created to facilitate the task in web applications assessments, it's a tool by pentesters for pentesters

- BurpSuite

  Burp Suite Community Edition is a feature-limited set of manual tools for exploring web security. Proxy your HTTPS traffic, edit and repeat requests, decode data, and more. It's packed with power features - including an automated vulnerability scanner, the ability to save your work, and an unthrottled version of burp intruder.

- sqlmap

  sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting,

over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

- Wireshark The most common tool used in network forensics is Wireshark 33 which does live packet sniffing and analysis of them.These can be in real time dynamically or in a saved pcap file where the analysis is happening afterwords to find any abnormal communications on the network or even files.

- tcpdump

  Tcpdump prints out a description of the contents of packets on a network interface that match the boolean expression; the description is preceded by a time stamp, printed, by default, as hours, minutes, seconds, and fractions of a second since midnight. It can also be run with the -w flag, which causes it to save the packet data to a file for later analysis, and/or with the -r flag, which causes it to read from a saved packet file rather than to read packets from a network interface. It can also be run with the -V flag, which causes it to read a list of saved packet files. In all cases, only packets that match expression will be processed by tcpdump.

- iptables

  Iptables and ip6tables are used to set up, maintain, and inspect the tables of IPv4 and IPv6 packet filter rules in the Linux kernel. Sev eral different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

  Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a ‘target’, which may be a jump to a user-defined chain in the same table.

- Links and Websites

  RSA multi attacks tool : uncipher data from weak public key and try to recover private key Automatic selection of best attack for the given public key

- gdb-peda(GNU Linux Debugger with Python Exploit Development)

  GNU Debugger (GDB) is not only good to debug buggy applications. It can also be used to learn about a program’s control flow, change a program’s control flow, and modify the code, registers, and data structures. These tasks are common for a hacker who is working to exploit a software vulnerability or is unraveling the inner

workings of a sophisticated virus. GDB works on ELF binaries and Linux processes. It is an essential tool for Linux hackers and will be used in various examples throughout this tutorial.

- ltrace,strace

  ltrace is a program that simply runs the specified command until it exits. It intercepts and records the dynamic library calls which are called by the executed process and the signals which are received by that process. It can also intercept and print the system calls exe cuted by the program.

  strace is a useful diagnostic, instructional, and debugging tool. System administrators, diagnosticians and trouble-shooters will find it invaluable for solving problems with programs for which the source is not readily available since they do not need to be recompiled in order to trace them. Students, hackers and the overly-curious will find that a great deal can be learned about a system and its system calls by tracing even ordinary programs. And programmers will find that since system calls and signals are events that happen at the user/kernel interface, a close examination of this boundary is very useful for bug isolation, sanity checking and attempting to capture race conditions.

- pwntools 40

  Pwntools is a CTF framework and exploit development library. Written in Python, it is designed for rapid prototyping and development, and intended to make exploit writing as simple as possible.

- ROPgadget 41

  ROPgadget lets you search your gadgets on a binary. It supports several file formats and architectures and uses the Capstone disassembler for the search engine.

Users will be given a Dockerfile to build a Kali Linux container to make the process more easy, portable and light for anyone. The container have the basic tools that they need to do the recon and exploitation of the target.

3.1 Categories

Below is the categories, vulnerabilities and attacks covered in our platform

1. Web
    - Cross Site Scripting XSS
    - SQL injection
    - Server Side Request Forgery (SSRF)
    - XML External Entity (XXE)
    - Local/Remote File Inclusion
    - Remote Command Execution (RCE)
    - Weak Authentication
        - Brute Force Attacks 3.2.8
        - Dictionary Attacks 3.2.8
        - Cookies and Tokens Attacks 3.2.8
2. Crypto
    - Substitution Ciphers
    - XOR
    - AES/DES
    - Hashing functions
    - RSA
3. Forensics
    - Network Forensics
        - Wireshark
    - Firewalls
        - iptables
4. Reversing
    - Use of GNU debugger(GDB)
    - Find strings inside registers
    - Disassemble functions and understand the flow to get the password
5. Pwn

- Binary exploitation
- Buffer Overflow
- ROP
- ret2libc
- FSB

The following table presents a mapping of the Juice Shop's categories to OWASP and CWE (without claiming to be complete). 3.2

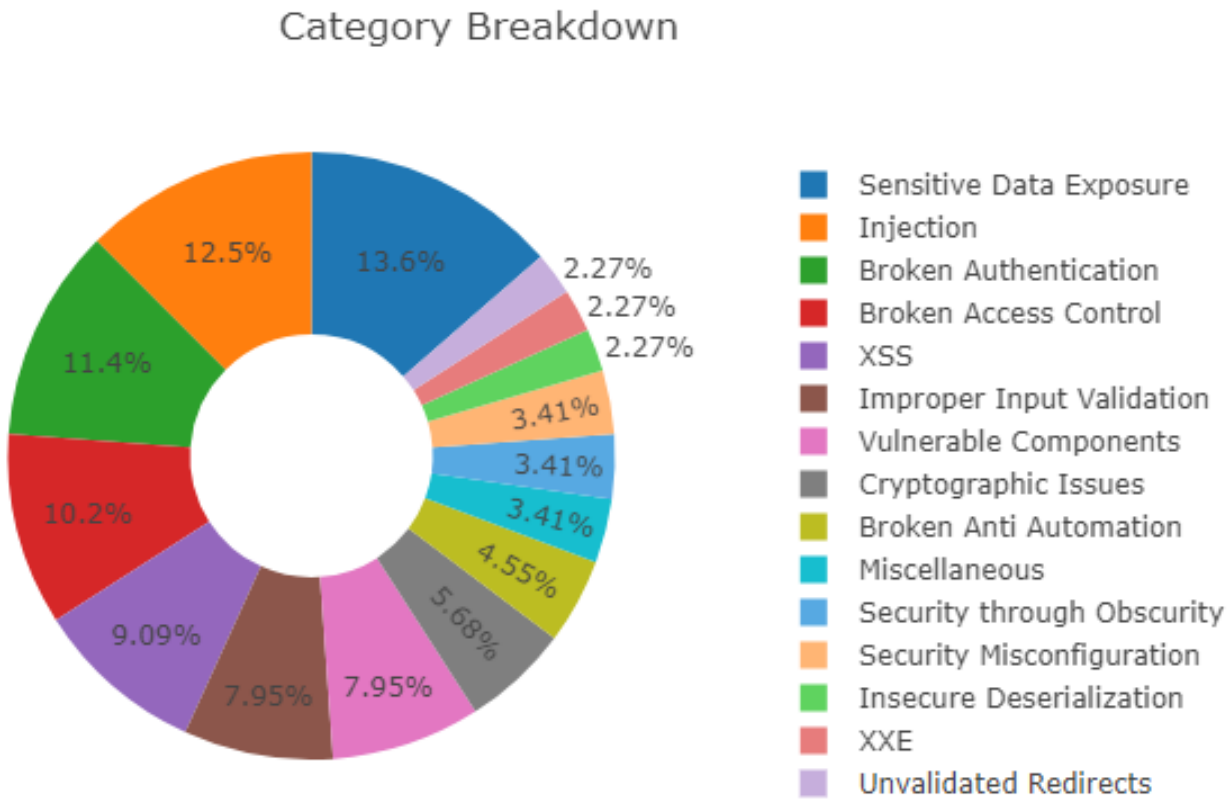

Figure 3.2: Vulnerabilities covered from OWASP's juice shop lab

3.2 WEB

In this category users have the option to test different web vulnerabilities in different level of difficulty. In every level new countermeasures are added in the application so the user

that acts as an attacker has to bypass them.

3.2.1 Cross Site Scripting XSS

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user [4]. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it. An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page [10]. In the lab the user has to exploit the XSS vulnerability in every level and get an alert().The difficulty is increasing in every level so the user starts from simply put a <script>alert()</script> code,bypasssing WAF's and use tools and programs such as Burpsuite to send crafted requests.[56]

In the following image we successfully exploited an XSS by putting in the user input field **<script>alert()</script>.**3.3

3.2.2 SQL injection

SQL injections are among the least understood types of web application attack, but they are also among the most dangerous. A database exposed to the web can be a critical point of information leakage, if the application code designed to enable access is not sufficiently secured. Records can be updated, customer databases can be dumped to text files in their entirety, or malicious attackers can simply choose to delete the whole database using a single command.[66]An SQL injection attack or SQLi consists of insertion or ”injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type

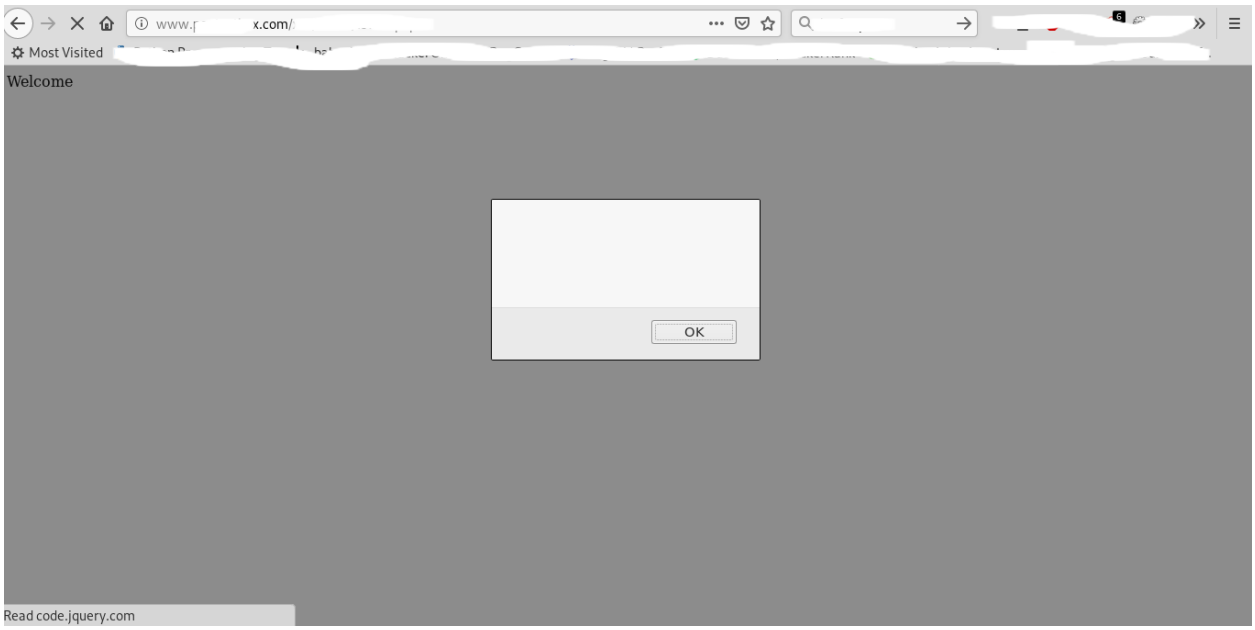

Figure 3.3: XSS attack

of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands[91]. Types of SQLi:

- Classic SQLi
- Error Based SQLi
- Union Based SQLi
- Blind SQLi
- Time Based SQLi
- Boolean Based SQLi
- Second Order SQLi

The lab has ten levels of different types of SQL injection where the users have to exploit the different types of injections from basic to advanced.

In the following image we see the error that occures in the database when the put a single quote in the id parameter.The responding error is **You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ”-1” LIMIT 0,1’ at line 1**

3.4

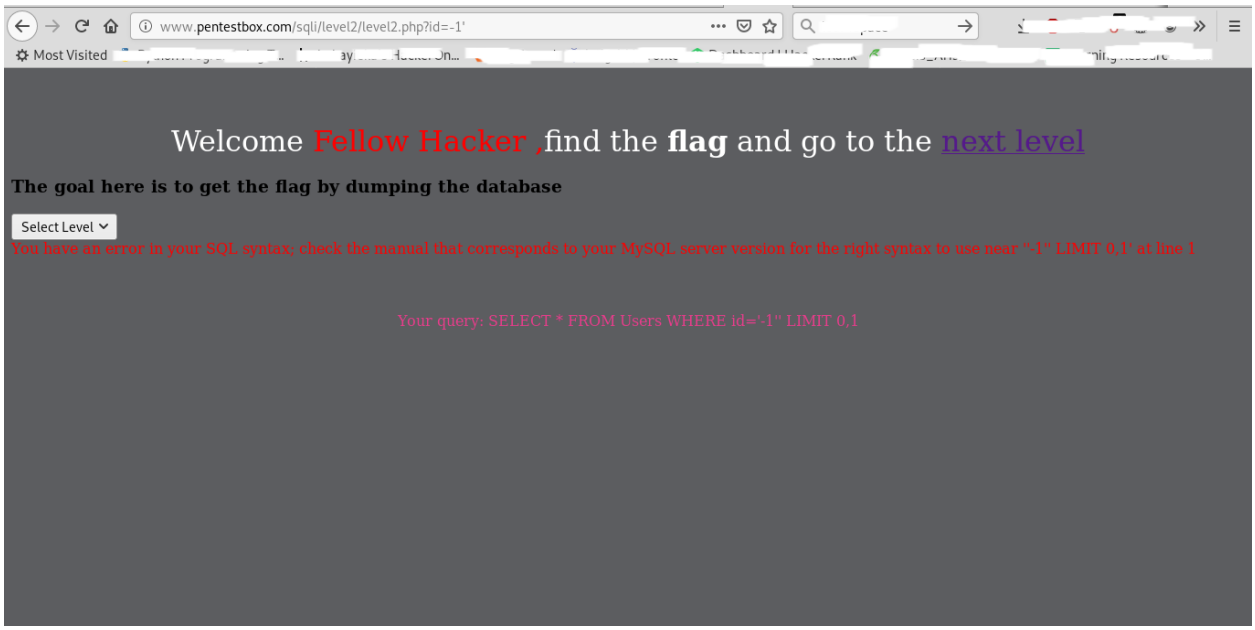

Figure 3.4: SQL injection attack

3.2.3 Server Side Request Forgery (SSRF)

Server Side Request Forgery (SSRF) is a vulnerability that gives an attacker the ability to create requests from a vulnerable server. SSRF attacks are commonly used to target not only the host server itself, but also hosts on the internal network that would normally be inaccessible due to firewalls. SSRF allows an attacker to:

Scan and attack systems from the internal network that are not normally accessible Enumerate and attack services that are running on these hosts Exploit host-based authentication services

As is the case with many web application vulnerabilities, SSRF is possible because of a lack of user input validation. For example, a web application that accepts a URL input in order to go fetch that resource from the internet can be given a valid URL such as http://google.com But the application may also accept URLs such as:

- http://localhost
- http://10.0.0.1
- file:///localhost/example.txt

- file:///path/to/file
- 127.0.0.1:22

When those kinds of inputs are not validated, attackers are able to access internal resources that are not intended to be public [55] What can SSRF do:

- SSRF to Reflected XSS
- Try URL schemas to read internal and make server perform actions (file:///, dict://, ftp://, gopher://,git://..)
- Scan for internal networks and ports.
- Access to metadata services Amazon EC2 and OpenStack, this may allow the attacker to gain access to metadata services, which can be used to gain access to sensitive information, sometimes including credentials or private keys.

An example of XSS from SSRF is to fetch an .svg file that has embedded javascript and when it loads this file will pop an alert() thanks to http://brutelogic.com.br/poc.svg

In the lab there are five levels with SSRF vuln pages where the user has to fetch an internal port and get the flag.

3.2.4 XML External Entity (XXE)

An XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts.

The XML 1.0 standard defines the structure of an XML document. The standard defines a concept called an entity, which is a storage unit of some type. There are a few different types of entities, external general/parameter parsed entity often shortened to external entity, that can access local or remote content via a declared system identifier. The system identifier is assumed to be a URI that can be dereferenced (accessed) by the XML processor when processing the entity. The XML processor then replaces occurrences of the named external entity with the contents dereferenced by the system identifier. If the

system identifier contains tainted data and the XML processor dereferences this tainted data, the XML processor may disclose confidential information normally not accessible by the application. Similar attack vectors apply the usage of external DTDs, external stylesheets, external schemas, etc. which, when included, allow similar external resource inclusion style attacks.

Attacks can include disclosing local files, which may contain sensitive data such as passwords or private user data, using file: schemes or relative paths in the system identifier. Since the attack occurs relative to the application processing the XML document, an attacker may use this trusted application to pivot to other internal systems, possibly disclosing other internal content via http(s) requests or launching a CSRF attack to any unprotected internal services. In some situations, an XML processor library that is vulnerable to client-side memory corruption issues may be exploited by dereferencing a malicious URI, possibly allowing arbitrary code execution under the application account. Other attacks can access local resources that may not stop returning data, possibly impacting application availability if too many threads or processes are not released. Note that the application does not need to explicitly return the response to the attacker for it to be vulnerable to information disclosures. An attacker can leverage DNS information to exfiltrate data through subdomain names to a DNS server that he/she controls.

3.2.5 Exploiting XXE to perform SSRF attacks

Aside from retrieval of sensitive data, the other main impact of XXE attacks is that they can be used to perform server-side request forgery (SSRF). This is a potentially serious vulnerability in which the server-side application can be induced to make HTTP requests to any URL that the server can access. To exploit an XXE vulnerability to perform an SSRF attack, you need to define an external XML entity using the URL that you want to target, and use the defined entity within a data value. If you can use the defined entity within a data value that is returned in the application's response, then you will be able to view the response from the URL within the application's response, and so gain two-way interaction with the backend system. If not, then you will only be able to perform blind SSRF attacks (which can still have critical consequences). In the following XXE example, the external entity will cause the server to make a back-end HTTP request to an internal system within the organization's infrastructure: <!DOCTYPE foo [<!ENTITY xxe SYSTEM "http://internal.vulnerable-website.com/">]>

3.2.6 Local/Remote File Inclusion

Local File Inclusion is quite simply the act of including files that are stored on the web server A file inclusion vulnerability is a type of web vulnerability that is most commonly found to affect web applications that rely on a scripting run time. This issue is caused when an application builds a path to executable code using an attacker-controlled variable in a way that allows the attacker to control which file is executed at run time. A file include vulnerability is distinct from a generic directory traversal attack, in that directory traversal is a way of gaining unauthorized file system access, and a file inclusion vulnerability subverts how an application loads code for execution. Successful exploitation of a file inclusion vulnerability will result in remote code execution on the web server that runs the affected web application. An attacker can use remote code execution to create a web shell on the web server, which can be used for website defacement. [81][36] 3.5

```
1  <?php
2  $file = $_GET['file'];
3  include($file);
4  ?>
5  //http://vuln/index.php?file=../../../../../../etc/passwd
6
```

Figure 3.5: Local file inclusion

Remote File Include (RFI) is an attack technique used to exploit ”dynamic file include” mechanisms in web applications. When web applications take user input (URL, parameter value, etc.) and pass them into file include commands, the web application might be tricked into including remote files with malicious code.Most of web application frameworks support file inclusion. File inclusion is mainly used for packaging common code into separate files that are later referenced by main application modules. When a web application references an include file, the code in this file may be executed implicitly or explicitly by calling specific procedures. If the choice of module to load is based on elements from the HTTP request, the web application might be vulnerable to RFI.[41][81]Next is an example of a vulnerable code that gives the permissions to an attacker to include files from remote hosts. 3.6

```php
<?php
$incfile = $_REQUEST["file"];
include($incfile.".php");
//http://www.target.com/vuln_page.php?file=http://www.attacker.com/malicous

```

Figure 3.6: Remote file inclusion

3.2.7 Remote Command Execution (RCE)

Remote Command Execution is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation.3.7

```c
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv) {
 char cat[] = "cat ";
 char *command;
 size_t commandLength;

 commandLength = strlen(cat) + strlen(argv[1]) + 1;
 command = (char *) malloc(commandLength);
 strncpy(command, cat, commandLength);
 strncat(command, argv[1], (commandLength - strlen(cat)) );

 system(command);
 return (0);
}
```

Figure 3.7: Command execution

PHP common dangerous functions that can execute commands: 26

- **eval();)**
- **assert();**

- **system();**
- **exec();**
- **passthru();**

Where may command injections exist?

1. Web Applications(i.e IBM, Sophos, Symantec, LanDesk, Cacti, SquirrelMail, ….)
2. ADSL SOHO routers(i.eD-Link, TP-Link, Linksys, ….)
3. IP Cameras(i.e TP-Link, D-Link, Vivotek, Zero-IP, …)
4. Network Printers(i.e Xerox, …)
5. IP PBX Applications(i.e Asterisk PBX, FreePBX, …)
6. Raspberry PI based Web Applications
7. Arduino based Web Applications

Remote command execution in very dangerous and critical vulnerability that gives the attacker the ability to do anything is possible running under the permissions of the current user.Use of functions that execute commands must be very carefully evaluated and make use of WAF's or Blacklists/Whitelists and also sanitize the input of the user.

3.2.8 Weak Authentication

"Authentication" refers to the process of proving an identity to an application or system.[27] That is, the task of demonstrating that you are who you claim to be. In software systems, this usually means providing a password for a corresponding user or account identifier. While this is the most common means of proving one's identity to a system, it is not the only one. In the interests of illuminating the larger landscape, we will introduce other means of authentication: Knowledge-based Authentication This approach relies on knowledge that only the genuine user would have. A password is an example of "something you know". Assuming that the password is kept confidential by the user, it can serve as a means of authentication. Secret questions also fall into this category. We explore potential weaknesses in this approach below. Possession-based Authentication This approach relies

on the fact that only the genuine user would be in possession of the artifact needed to authenticate. Assuming that the artifact is kept secure by the user, its possession can be used to vouch for the user. Key fobs that generate codes, and codes sent to your mobile-device are examples of possession-based authentication. Physical keys can also be used in this manner. The ability to access an email account registered with the application account is also an implementation of this technique. Identity-based Authentication This is about the aspects of the user that are unique and cannot be counterfeited. In contemporary systems, this typically means a biometric reading of some sort, such as a fingerprint, iris scan, voice-print, etc. that is compared to a base reference. Single, Two-Factor, and Multi-Factor Authentication These terms refer to the amount of evidence that must be presented to authenticate. Single Factor Authentication requires a solitary item of evidence, most typically a password in software systems. Note that an Id Badges, a driver's license, or a passport can serve a similar purpose in daily life. Two-Factor authentication is a combination of two "forms" of authentication, such as knowledge-based and possession-based. Multi-Factor authentication solutions combine three or more methods.

Risk-based Authentication This describes an adaptive approach to authentication that escalates the identity challenges to the user in response to: Their situational conformance to a "usage" profile that is developed and maintained by the application. This includes information used to identify the user such as typical usage timeframes, common operations, ip addresses utilized, geo-location, browser fingerprint etc. Attempted access to highly sensitive features or information. The application using risk-based authentication may demand additional authentication factors in response to deviations from the usage profile and/or to protect sensitive operations. What is Weak Authentication ? The more difficult an authentication mechanism is to defeat the stronger it is. Clearly the authentication strength of a system should correlate to the value of the assets it is protecting. Two-Factor and Multi-Factor Authentication solutions are appropriate for systems that deal with highly valued assets. Weak Authentication describes any scenario in which the strength of the authentication mechanism is relatively weak compared to the value of the assets being protected [30]. It also describes scenarios in which the authentication mechanism is flawed or vulnerable. Password Strength The "strength" of a password is related to the potential set of combinations that would need to be searched in order to guess it.

For example, a password scheme with a length of two characters and consisting only of digits would represent a a search space of 100 possible passwords (10 x 10),

whereas a 12 digit password would represent 1012 possible combinations. The larger the set of possible combinations, the harder it is to guess and the stronger the password. Thus, the following factors influence password strength: Length: The number of characters in the password. The greater the length, the greater the strength. Character Set: The range of possible characters that can be used in the password. The broader the range of characters, the greater the strength. It is typical for strong password schemes to require upper and lower case letters, digits, and punctuation characters. [37][49][52]

Password Policy describes the rules that are enforced regarding password strength, changes, and re-use. An effective password policy supports strong authentication [40]. It is generally accepted that the each of the following will increase the integrity of the authentication process: Periodically changing the password for an account makes it less likely that a password will be compromised, or that a compromised password will be used [88]. This is termed password expiration. Prohibiting the re-use of the same (or similar) password to the one being changed will prevent password expiration from being circumvented by users. Enforcing minimum strength rules for passwords will guarantee application compliance with Password Policy [76]. Prohibiting dictionary words and/or popular passwords will make password cracking less likely. The use of secret questions to further demonstrate identity. The more of these rules that are enforced, the stronger will be the authentication mechanism, Password Cracking [82]

There are countless hacking tools and frameworks available to help an attacker guess a password through an automated sequence of attempts. This is called “brute forcing” because such tools will attempt all possible password combinations given a set of constraints in an attempt to authenticate. [64] An application that does not protect itself against password cracking in some manner may be considered as having a Weak Authentication vulnerability depending the requirements and risk-level. Dictionary Attacks In addition to brute force attacks, password cracking tools also typically have the ability to test a file of candidate passwords. [97] This is called a dictionary attack because the file used may actually be a dictionary of words. Passwords that can be found in a dictionary are considered weak because they can eventually discovered using a dictionary attack. [85] An application that allows dictionary words as passwords may be considered as having a Weak Authentication vulnerability depending the application requirements and risk-level. Popular Passwords Since passwords are usually freely chosen and must be remembered, and given that humans are lazy, passwords that are easy to remember tend to be more popular than those that are not. In fact, some passwords become very popular and are

used far more frequently that might be expected. [39]

Although the most popular entries change over time, you can always find a “top-N” list somewhere, like here, or here, or here. Clearly it is in the user’s best interest to avoid the most popular passwords. An application that allows popular passwords may be considered as having a Weak Authentication vulnerability depending the application requirements and risk-level. The whole purpose of authentication is to ensure that only authorized users gain access to the application capabilities and the information it contains. It is essential therefore that the system verifies the “authentication status” of the user for every user action or request before it is carried out. The ability of a user to access any application feature or resource without having first authenticated represents a Weak Authentication vulnerability. A Bruteforce Attack uses several repetitive trial-and-error attempts to guess the password to break into a website or a service. These attempts are quick and vigorous and are carried out by bots [59]. A report by eSentire says that brute force attacks increased by 400 percent in 2017. While some of these attacks were blocked, a majority of them were able to gain unauthorized access to user accounts.How does a brute force attempt work? Most websites require a password of minimum eight characters. As you add another character to your password, its complexity increases and it becomes even more difficult for brute force hackers to break into the system.

Let’s say you choose eight alphanumeric characters for your password. This includes uppercase and lowercase letters, along with numbers. The possible character set you’ll use will be 26 x 2 alphabets (uppercase + lowercase) = 52 characters. Add 10 numbers to it and the possible character set will have 62 characters. So, an eight-character password that has uppercase and lowercase letters and numeric digits, it will take 628 attempts. This comes out to be about 218 trillion combinations. If a bot attempts one combination per second, it will take about 218 trillion seconds or 7 million years to crack that password. That means your eight-character password is safe, right? Not really. With say a computer can perform 1,000 combinations in a second. The total time for breaking your password will now be reduced to seven thousand years. Still safe, right? Not so fast. With a supercomputer that performs 109 combinations per second, it can test all your combinations in just 22 seconds![94]

An HTTP cookie is data sent from a website and stored on the user’s computer by the user’s web browser while the user is browsing. Links and Websites [46] Cookies were designed to be a reliable mechanism for websites to remember stateful information (such as items added in the shopping cart in an online store) or to record the user’s browsing

activity (including clicking particular buttons, logging in, or recording which pages were visited in the past) [18]. They can also be used to remember arbitrary pieces of information that the user previously entered into form fields such as names, addresses, passwords, and credit-card numbers.

Cookies perform essential functions in the modern web. Perhaps most importantly, authentication cookies are the most common method used by web servers to know whether the user is logged in or not, and which account they are logged in with. Without such a mechanism, the site would not know whether to send a page containing sensitive information, or require the user to authenticate themselves by logging in. The security of an authentication cookie generally depends on the security of the issuing website and the user's web browser, and on whether the cookie data is encrypted. Security vulnerabilities may allow a cookie's data to be read by a hacker, used to gain access to user data, or used to gain access (with the user's credentials) to the website to which the cookie belongs (see cross-site scripting and cross-site request forgery for examples) [84].

Opposed to cookies is the idea of tokens.Tokens were introduced for replacing cookies [44]. Tokens, is a set of characters and numbers and the user have to specifically add them to the header in his requests. So setting tokens you will have to do it yourself every time, while cookies are added automatically for you.The advantage of tokens is that we can send them to any domain we want. If for example you are on test.com domain and want to send an authenticated request to a totally different, not a related domain, you have to use a token [18]. Before sending a request,user simply adds the identifying token to the header and he will be authenticated on that domain. This is very useful if we build distributed systems, where servers are hosted on different domains but we want the same user to be authenticated across all of them.

3.3 Crypto

In this category user are introduced into basic substitution ciphers like Caesar and Vigenere, then XOR cipher, hashing functions and finally in symmetric and asymmetric encryption algorithms. They are learning how these algorithms are implemented and common vulnerabilities that have.They are given a set of challenges with encrypted data and the users have to decrypt them by understanding the encryption algorithm and exploit the vulnerability that exists.

3.3.1 Substitution Ciphers

In cryptography, a substitution cipher is a method of encrypting data by replacing units of plaintext with ciphertext, following rules of a fixed system.These ”units” can be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. In order the receiver to decrypt the ciphertext has to perform the inverse of the substitution.Substitution ciphers can be compared with transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.There are many different types of substitution cipher. If the cipher operates on single letters, it is termed a simple substitution cipher,a cipher that operates on larger groups of letters is termed polygraphic. A monoalphabetic cipher uses fixed substitution over the entire message, whereas a polyalphabetic cipher uses a number of substitutions at different positions in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice versa.28

The simplest and most famous substitution cipher is the Ceasar cipher where the alphabet used is shifted by a number.For example ROT-13 cipher is a Ceasar cipher where the number of the shifted letters is 13. Example of using ROT-13.Assume we have the sentence **The Secrets of UTH**.If we shift the alphabet by 13 then the result will be **Gur Frpergf bs HGU**.

On the other hand a polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The Vigenère cipher is probably the best-known example of a polyalphabetic cipher, though it is a simplified special case. The Enigma machine is more complex but is still fundamentally a polyalphabetic substitution cipher.Encryption with Vigenere uses a key made of letters (and an alphabet).Example of Vigenere cipher using the key **UTH**, alphabet **ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789** and the plaintext **What is happening with Vigenere** gives us **G0hd 1z 1tw9xu26n g101 Ep0xuyal**.

3.3.2 XOR

In cryptography, the simple XOR cipher is a type of additive cipher,25 an encryption algorithm that operates according to the principles: $A \oplus 0 = A$ $A \oplus A = 0$

$(A \oplus B) \oplus C = A \oplus (B \oplus C)$

$(B \oplus A) \oplus A = B \oplus 0 = B$

Where $\oplus$ denotes the exclusive OR (XOR) operation. This operation is sometimes called modulus 2 addition (or subtraction, which is identical).[2] With this logic, a string of text can be encrypted by applying the bitwise XOR operator to every character using a given key. To decrypt the output, merely reapplying the XOR function with the key will remove the cipher.3.8 A simple way to encrypt data with XOR with python is:

```
root@babayieka:~# ipython
Python 3.7.3rc1 (default, Mar 13 2019, 11:01:15)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.4.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: password='test_password'

In [2]: key='U'

In [3]: encryption = ''.join([chr(ord(x) ^ ord(key)) for x in password])

In [4]: decryption =''.join([chr(ord(x) ^ ord(key)) for x in encryption])

In [5]: encryption
Out[5]: '!0&!\n%4&&":\'1'

In [6]: decryption
Out[6]: 'test_password'
```

Figure 3.8: XOR

When it comes to cracking of XOR encryption there are plenty of exploits that an attacker could try.Single byte XOR encryption, where are only 255 key combinations where it's easy to bruteforce.In multibyte XOR encryption the attack is more difficult but not impossible.Using cryptographic techniques,such as frequency analysis,to analyze the encrypted data it's possible to retrieve the key and crack the encryption.Another attack in XOR encryption is known plaintext attack where the attacker knows a partial plaintext of the encrypted data an attacker can do repeatedly XOR through the encrypted data with the plaintext and get segments of the key.Let's don't forget the principal of the XOR. [38]

- $plaintext \oplus key = encrypted - text$
- $encrypted - data \oplus plaintext = key$

- $encrypted - text \oplus key = plaintext$

  If the key is smaller than the plaintext, the key is repeated.

3.3.3 AES/DES

AES and DES are both symmetric block ciphers.With the symmetric term we mean that the encryption and decryption happens with one shared key. DES algorithm is the first one that developed by federal government for encrypting the communications of the goverment.[1] DES uses a 56-bit bit key for encryption and the block size is 64 bits. Over the years there were many contests that proved the ineffectiveness of the algorithm and pushing users to use newly developed algorithms and more secure.After that triple DES was developed that does what the name says. Encrypt three times the data with the DES algorithm,which also proved ineffective to bruteforce attacks. AES data encryption is a more mathematically efficient and elegant cryptographic algorithm, but its main strength rests in the option for various key lengths. [47] AES allows you to choose a 128-bit, 192-bit or 256-bit key and block size of 128 bits, making it exponentially stronger than the 56-bit key of DES.[90][83]

3.3.4 Hashing functions

A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. The values are used to index a fixed-size table called a hash table. Use of a hash function to index a hash table is called hashing. In cryptography checksums are related to hashing functions and are one way functions that in theory they are constructing a unique value as output for every input.They way that the functions works is to take as input the data we want then pass them to the hashing functions and give an output of hash digest. [5] Most of the hash functions take an input string and produce a fix length of hash digest. Cryptographic hashing functions have many applications on information technology and information security such as:

- **File verification**: Comparing message digests (hash digests over the message) calculated before, and after, transmission can determine whether any changes have been made to the message or file. Common use is when we want to identify viruses or malware id's. [21][73]

- **Digital Signature**: This allows the signature calculation to be performed on the relatively small, statically sized hash digest. The message is considered authentic if the signature verification succeeds given the signature and recalculated hash digest over the message. [23][7]
- **Password hashing**: Password verification commonly relies on cryptographic hashes. Storing all user passwords as cleartext can result in a massive security breach if the database is compromised.[65] One way to reduce this danger is to only store the hash digest of each password. To authenticate a user, the password presented by the user is hashed and compared with the stored hash. A password reset method is required when password hashing is performed because original passwords cannot be recalculated from the stored hash value. [32] In most Unix and Unix-like operating systems passwords entered by the user is run through a key derivation function to create a hashed version of the password, which is saved. Only the hashed version is stored in the OS and the entered password is not saved for security reasons. [92]
- **Proof-of-Work Systems**: A proof-of-work system (or protocol, or function) is an economic measure to deter denial-of-service attacks [43] and other service abuses such as spam on a network by requiring some work from the service requester, usually meaning processing time by a computer. [95] A key feature of these schemes is their asymmetry: The work must be moderately hard (but feasible) on the requester side but easy to check for the service provider. One popular system – used in Bitcoin [29] mining and Hashcash – uses partial hash inversions to prove that work was done, to unlock a mining reward in Bitcoin and as a good-will token to send an e-mail in Hashcash. The sender is required to find a message whose hash value begins with a number of zero bits.
- **File data Identifier**: Most of source code management systems, like Git, Mercurial and Monotone, use the sha1sum of various types of content (file content, directory trees, ancestry information, etc.) to uniquely identify them. Hashes are used to identify files on peer-to-peer filesharing networks. For example, in an ed2k link, an MD4-variant hash is combined with the file size, providing sufficient information for locating file sources, downloading the file and verifying its contents. Magnet links are another example. Such file hashes are often the top hash of a hash list or a hash tree which allows for additional benefits.

Below there are some examples of hashing the string password in different hashes.

$root@babayieka\#\ echo$ -n "$password$" | $md5sum$
$5f4dcc3b5aa765d61d8327deb882cf99$
$root@babayieka\#echo - n"password"$ | $sha1sum$
$5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8$
$root@babayieka\#echo - n"password"$ | $sha256sum$
$5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8$
$root@babayieka\#echo - n"password"$ | $sha512sum$
$b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7785e5976ec049b46df5f13$
$af5a2ea6d103fd07c95385ffab0cacbc86$

3.3.5 RSA

When a user make a request to a website through secure and encrypted connection the symmetric encryption by its self will not work. So the problem here was how we share the secret key with the end connection without risking that someone whose intercepting the traffic will take the key.In November 1976, a paper published in the journal IEEE Transactions on Information Theory by Diffie and Hellman, titled "New Directions in Cryptography," addressed this problem and offered up a solution: public-key encryption.[22]Also known as asymmetric encryption, public key cryptography is used as a method of assuring the confidentiality, authenticity and non-repudiation of electronic communications and data storage. Public-key encryption uses two different keys at once, a combination of a private key and a public key. The private key must remain confidential to its respective owner, while the public key is made available to everyone via a publicly accessible repository or directory. To decode an encrypted message, a computer must use the public key, provided by the originating computer, and its own private key. Although a message sent from one computer to another won't be secure since the public key used for encryption is published and available to anyone, anyone who picks it up can't read it without the private key. RSA now is probably the most recognizable asymmetric algorithm.Originally developed in 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman. It is an asymmetric cryptographic algorithm.Asymmetric means that uses two different keys. One that is the private key (which is secret) for the decryption of the data and the other one,the public key for the encryption. Below are the steps that the algorithm of the RSA implements to encrypt data.

1. First generate the keys:
    - Choose p,q which is prime numbers.
    - Calculate $n = pxq$ (Where n is the Modulus)
    - Calculate $phi(n) = (p-1)x(q-1)$
    - Choose integer e
    - Calculate $d = m^{-}1(mod(phi(n)))$
    - Create the public key = (n,e)
    - Create the private key = (d,e)
2. Encrypt m where m is the plaintext and $m < n$
    - $C = M^{e}mod(n)$
3. Decrypt the ciphertext C
    - $M = C^{d}mod(n)$

RSA attacks

- Weak public key factorization
- Wiener's attack
- Hastad's attack (Small public exponent attack)
- Small q (q < 100,000)
- Common factor between ciphertext and modulus attack
- Fermat's factorisation for close p and q
- Gimmicky Primes method
- Past CTF Primes method
- Self-Initializing Quadratic Sieve (SIQS) using Yafu
- Common factor attacks across multiple keys
- Small fractions method when p/q is close to a small fraction

- Boneh Durfee Method when the private exponent d is too small compared to the modulus (i.e d < n0.292)
- Elliptic Curve Method
- Pollards p-1 for relatively smooth numbers
- Mersenne primes factorization

A very common tool mostly used in CTF's is 27

In many cases developers use and combine asymmetric and symmetric encryption.A use case of them is HTTPS.Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP). [25] It is used for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, its predecessor, Secure Sockets Layer (SSL). HTTPS is a TCP/IP application layer protocol, which is actually the SSL/TLS security protocol running on top of HTTP. [8] An HTTPS connection between a client and a server, employs both types of encryption. Asymmetric encryption is used first to establish the connection, which is then replaced with symmetric encryption (called the session) for the duration of the connection. A session key is a one-time use symmetric key which is used for encryption and decryption. Session keys are randomly created and are used only for any particular session. [69]

Below are the steps that need to be so the established connection will be secure. [68]

1. The server and the client in order to engage in a secure conversation, a TLS certificate needs to be created and verified by the Certificate Authority (CA).
2. The browser sends a ClientHello message 3.3.5 and indicates that it would like to start a conversation with a secure server. The ClientHello message contains all the information the server needs in order to connect to the client via TLS, including the various cipher suites and maximum TLS version that it supports.
3. Then a ServerHello 30 message which includes the TLS version to be used, the server TLS certificate, and the server's asymmetric public key.The server will send this message in response to a ClientHello message when it was able to find an acceptable set of algorithms. If it cannot find such a match, it will respond with a handshake failure alert.

4. The browser verifies the server certificate, and creates a random session key.
5. The session key is encrypted using the server's public key and is sent back to the server.
6. The server decrypts the session key with its own private key.
7. Now both parties have the session key. The public key encryption is terminated and replaced with symmetric encryption. The session with the server continues using only symmetric encryption.

ClientHello describes a Step within the TLS Handshake process. First message of a TLS handshake is when the Protocol Client initiates a connections to the Protocol Server using a ClientHello. The message by which the client states its intention to do some SSL/TLS. Note that ”client” is a symbolic role; it means ”the party which speaks first”. It so happens that in the HTTPS context, which is HTTP-within-SSL-within-TCP, all three layers have a notion of ”client” and ”server”, and they all agree (the TCP client is also the SSL client and the HTTP client), but that's kind of a coincidence.29

The ClientHello message contains:

- the maximum protocol version that the client wishes to support;
- the ClientHello.random (32 bytes, out of which 28 are suppose to be generated with a cryptographically strong number generator);
- the ”session ID” (in case the client wants to resume a session in an abbreviated handshake, see below);
- the list of ”Cipher Suites” that the client knows of, ordered by client preference;
- the list of compression algorithms that the client knows of, ordered by client preference;
- some optional extensions.

3.4 Forensics

In this category we are focusing in network forensics and the users are dive into Wireshark and learn to read pcap files, sniff network traffic, analyze the packets and protocols. This

category also introduce students in firewalls and iptables. They will have the opportunity to defend the system writing their own rules to respond in deployed attacks like bruteforce attacks and flood attacks. Forensics is the art of recovering the digital trail left on a computer. There are plenty of methods to find data which is seemingly deleted, not stored, or worse, covertly recorded. In Computer Science we have many types of forensics. In this section we'll be concentrate Network Forensics. Network forensics is a sub-category of digital forensics which is related to computer network traffic monitoring and analysis. [17] [62]

3.4.1 Network forensics

- The first, relating to security, involves monitoring a network for anomalous traffic and identifying intrusions.An attacker might be able to erase all log files on a compromised host;network-based evidence might therefore be the only evidence available for forensic analysis.

- The second form relates to law enforcement. In this case analysis of captured network traffic can include tasks such as reassembling transferred files, searching for keywords and parsing human communication such as emails or chat sessions.

The most common tool used in network forensics is Wireshark 33 which does live packet sniffing and analysis of them. These can be in real time dynamically or in a saved pcap file where the analysis is happening afterwords to find any abnormal communications on the network or even files.Incident response is an organized approach to addressing and managing the aftermath of a security breach or cyberattack. The goal is to handle the situation in a way that limits damage and reduces recovery time and costs.The SOC analysts perform the analysis and try to understand the phases of the attack how it started,which was the entry point if the attackers took any information from the systems if they moved any sensitive data and find the source of the cyberattack. Common research questions in forensic cases are known as the five or six W's: What, Why, Who, When, Where and How. With capturing and analyzing the network the defenders have the opportunity to identify threat, defend properly and respond to external attacks. Also they can learn known attack patterns and defend against script kiddies.

3.4.2 IPtables

In computing firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules.[33]

A firewall typically establishes a barrier between a trusted internal network and untrusted external network, such as the Internet. So we can say that the firewall is our first line of defence.

First generation: packet filters

The first reported type of network firewall is called a packet filter. Packet filters act by inspecting packets transferred between computers. When a packet does not match the packet filter's set of filtering rules, the packet filter either drops (silently discards) the packet, or rejects the packet (discards it and generates an Internet Control Message Protocol notification for the sender) else it is allowed to pass.[6] Packets may be filtered by source and destination network addresses, protocol, source and destination port numbers. The bulk of Internet communication in 20th and early 21st century used either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) in conjunction with well-known ports, enabling firewalls of that era to distinguish between, and thus control, specific types of traffic (such as web browsing, remote printing, email transmission, file transfer), unless the machines on each side of the packet filter used the same non-standard ports.

The first paper published on firewall technology was in 1988, when engineers from Digital Equipment Corporation (DEC) developed filter systems known as packet filter firewalls. At AT&T Bell Labs, Bill Cheswick and Steve Bellovin continued their research in packet filtering and developed a working model for their own company based on their original first generation architecture.

The Linux kernel comes with a packet filtering framework named netfilter. It allows you to allow, drop and modify traffic leaving in and out of a system. A tool, iptables builds upon this functionality to provide a powerful firewall, which you can configure by adding rules. In addition, other programs such as fail2ban also use iptables to block attackers.[75] 3.9

The packet filtering mechanism provided by iptables is organized into three different kinds of structures: tables, chains and targets. Simply put, a table is something that allows you to process packets in specific ways. The default table is the filter table, although there are other tables too.Again, these tables have chains attached to them. These chains

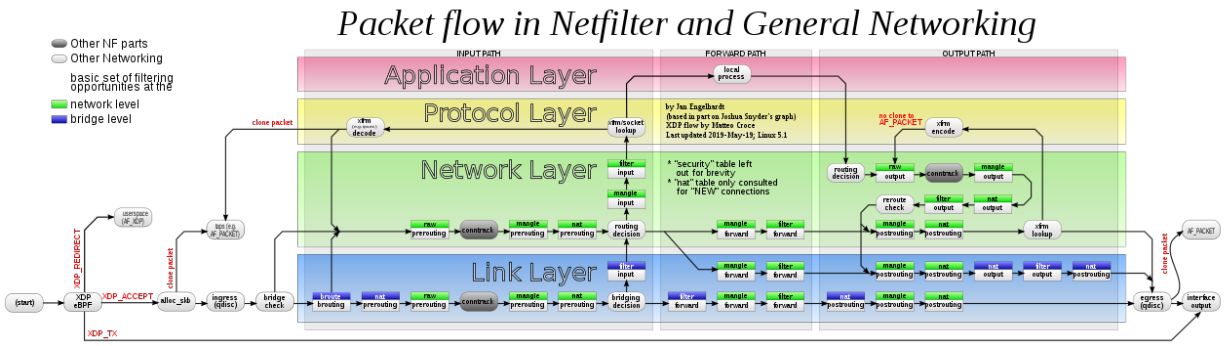

Figure 3.9: Netfilter packet flow

allow you to inspect traffic at various points, such as when they just arrive on the network interface or just before they're handed over to a process. You can add rules to them match specific packets — such as TCP packets going to port 80 — and associate it with a target. A target decides the fate of a packet, such as allowing or rejecting it. When a packet arrives (or leaves, depending on the chain), iptables matches it against rules in these chains one-by-one. When it finds a match, it jumps onto the target and performs the action associated with it. If it doesn't find a match with any of the rules, it simply does what the default policy of the chain tells it to. The default policy is also a target. By default, all chains have a default policy of allowing packets.

Tables

Tables gives the right to do various thing with packets.

- The filter table: This is the default and perhaps the most widely used table. It is used to make decisions about whether a packet should be allowed to reach its destination.
- The mangle table: This table allows you to alter packet headers in various ways, such as changing TTL values.
- The nat table: This table allows you to route packets to different hosts on NAT (Network Address Translation) networks by changing the source and destination addresses of packets. It is often used to allow access to services that can't be accessed directly, because they're on a NAT network.
- The raw table: iptables is a stateful firewall, which means that packets are inspected with respect to their "state". (For example, a packet could be part of a new connection, or it could be part of an existing connection.) The raw table allows you to work

with packets before the kernel starts tracking its state. In addition, you can also exempt certain packets from the state-tracking machinery.

Chains

Chains are composing the tables and allow user to filter the packets in different points.

- PREROUTING: Packets will enter this chain before a routing decision is made.
- INPUT: Rules in this chain apply to packets before they're given to a local process.
- OUTPUT: Packets sent from the machine itself will be visiting this chain.
- FORWARD: Rules in this chain apply to every packets that are routed through the current host.
- POSTROUTING: Routing decision has been made. Packets enter this chain are leaving the network interface and going to hardware.

Example rules

This command is to accept connection from localhost.

- iptables -A INPUT -p tcp -s localhost –dport 8080 -j ACCEPT

This command it to drop connection from any other hosts

- iptables -A INPUT -p tcp –dport 8080 -j DROP

This will allow echo requests (standard ICMP pings), but it won't explicitly allow any other ICMP traffic to pass through the firewall.

- iptables -A INPUT -p icmp -m icmp –icmp-type 8 -j ACCEPT

This will block the traffic from the specific IP address.

- iptables -I INPUT -s 192.168.1.100 -j DROP

Using UTH PENTESTBOX users have the opportunity to write their own rules with the iptables to defend against a various attacks.The goal in this process is to make the users respond to a deployed atttack.

3.5 Reverse Engineering

In this category we are introducing the users in the basic of reverse engineering of ELF files, assembly code, registers, find strings inside registers, understand their flow and use gdb-peda to dissasemple functions. According to Institute of Electrical and Electronics Engineers (IEEE) reverse engineering is ”the process of analyzing a subject system to identify the system’s components and their interrelationships and to create representations of the system in another form or at a higher level of abstraction”, where the ”subject system” is the end product of software development. [14] Reverse engineering is a process of examination only: the software system under consideration is not modified (which would make it re-engineering or restructuring). Reverse engineering can be performed from any stage of the product cycle, not necessarily from the functional end product. [15]Reverse engineering begin from the need that many times the source code of a program does not exists and to understand how is the flow of the program,what is the purpose, what it does and how it does something made researchers to evolve this field,develop tools to make reversing more easy for them.In information security in most cases of a malware attack the source code does not exist so the road to understand the program and it’s flow is to reverse it through reading the assembly. [42]

Reverse engineering of software can be accomplished by various methods. [74]

Disassembly is the process turning the raw machine language of the program in machine code format.This needs some experience for someone to be comfortable. The GNU debugger(GDB) is a popular tool used in Linux and Ollydbg for Windows.

Decompilation is the process that tries to recreate the source code in some high-level language for a program only available in machine code or bytecode. The Hex-Rays Decompiler is very popular and handy extension of the IDA Pro disassembler.

Below an executable file is reversed with gdb-peda which is a GNU debugger with python extention for giving more information.The goal is to find the right password and run the program with it.To find it with gdb-peda we setting a breakpoint where the program compares the given from user password and the real password.3.10

[63]

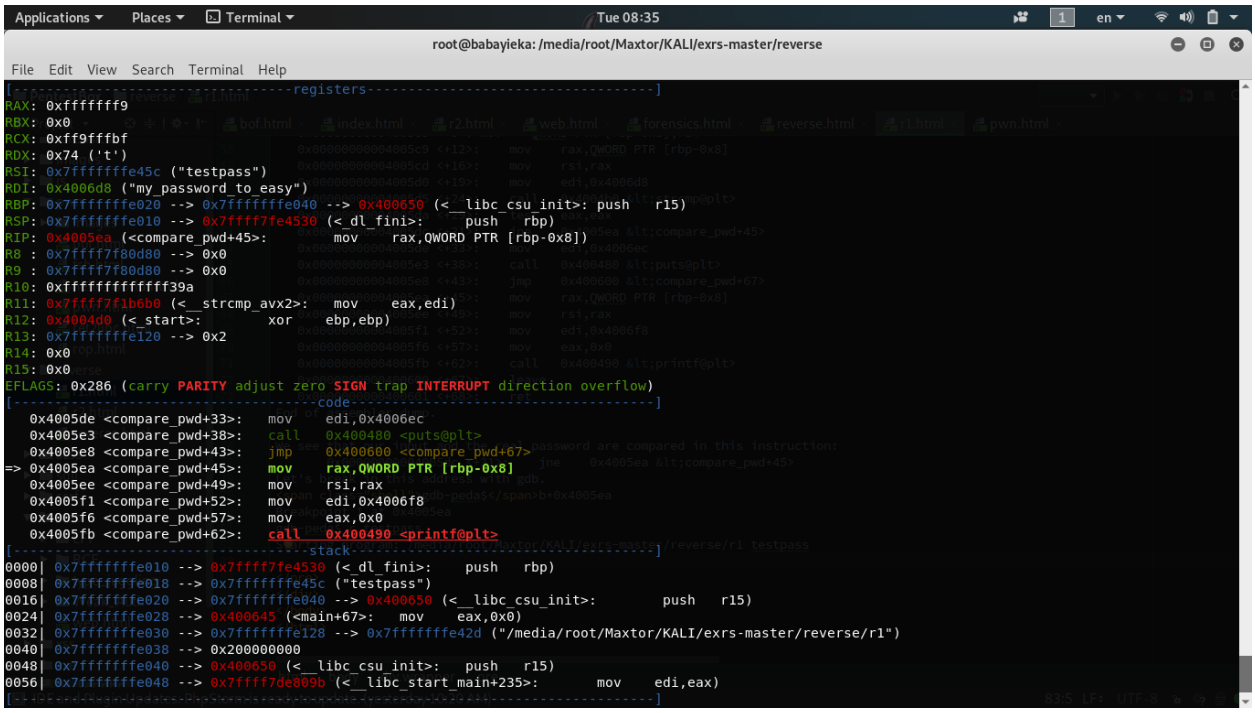

Figure 3.10: Reverse engineering using gdb-peda

3.6 Binary exploitation

In this category users are introduced in common vulnerabilities and exploitation techniques.A very strong skill for binary exploitation is reverse engineering.In binary exploitation users have to reverse the binary using any dissasembler read how it works find the vulnerability and learn pwntools, write a program or create a payload to exploit this vulnerability.

Binary exploitation or as known in CTF's PWN is the process where taking advantage of a bug/vulnerability from an executable program and cause it run unintendedly. Mostly the binary files are **ELF** binaries. In order someone exploit a bug in a binary file or application he must have a solid understanding of assembly, reverse engineering and basic knowledge of programming. Exploit means generally the advantage of the bug that we take to control the program.In pwn we write programs in programming languages (python,ruby) that we call them exploits. [48] Our target when we write exploits is to get a shell(execute /bin/sh) or modify any functions. These vulnerabilities comes from bad design, logical errors or lack of checking. Binaries have their own security measures to defend from various attacks. Security measures are:

- Canary: A canary is some (usually random) value that is used to verify that nothing has been overwritten. Programs may place canaries in memory, and check that they still have the exact same value after running potentially dangerous code, verifying the integrity of that memory.
- ASLR (Address Space Layout Randomization):** Security measure in modern OSes to randomize stack and libc addresses on each program execution.
- NX (Non-Executable): Security measure in modern OSes to separate processor instructions (code) and data (everything that's not code.) This prevents memory from being both executable and writable.
- PIE (Position Independent Executable): Essentially ASLR, but for the binary itself. When this protection is enabled, locations of actual code in the binary are randomized.

3.6.1 Buffer Overflow

A buffer overflow attack is a situation where a program uses some low level C function to write a variable or a string into a segment of memory which has a specific length and we try to write something longer than the allocated memory.That causes to overwrite addresses other than the allocated ones.In this section we will be talking about stack based buffer overflow [67]. The stack is simply an area in RAM that was chosen to be the stack - there is no special hardware to store stack contents. The esp/rsp register holds the address in memory where the bottom of the stack resides. When something is pushed to the stack, esp decrements by 4 (or 8 on 64-bit x86), and the value that was pushed is stored at that location in memory. Likewise, when a pop instruction is executed, the value at esp is retrieved (i.e. esp is dereferenced), and esp is then incremented by 4 (or 8) [71][51]. Below is a screenshot from ghidra tool where we imported a vulnerable binary.3.11

3.6.2 ROP

ROP stands for Return Oriented Programming. As an exploitment method is reusing tiny bits of code throughout the binary to construct commands we want to execute.[35][9] As we saw in buffer overflows, having stack control can be very powerful since it allows us to overwrite saved instruction pointers($eip $rip), giving us control over what the program

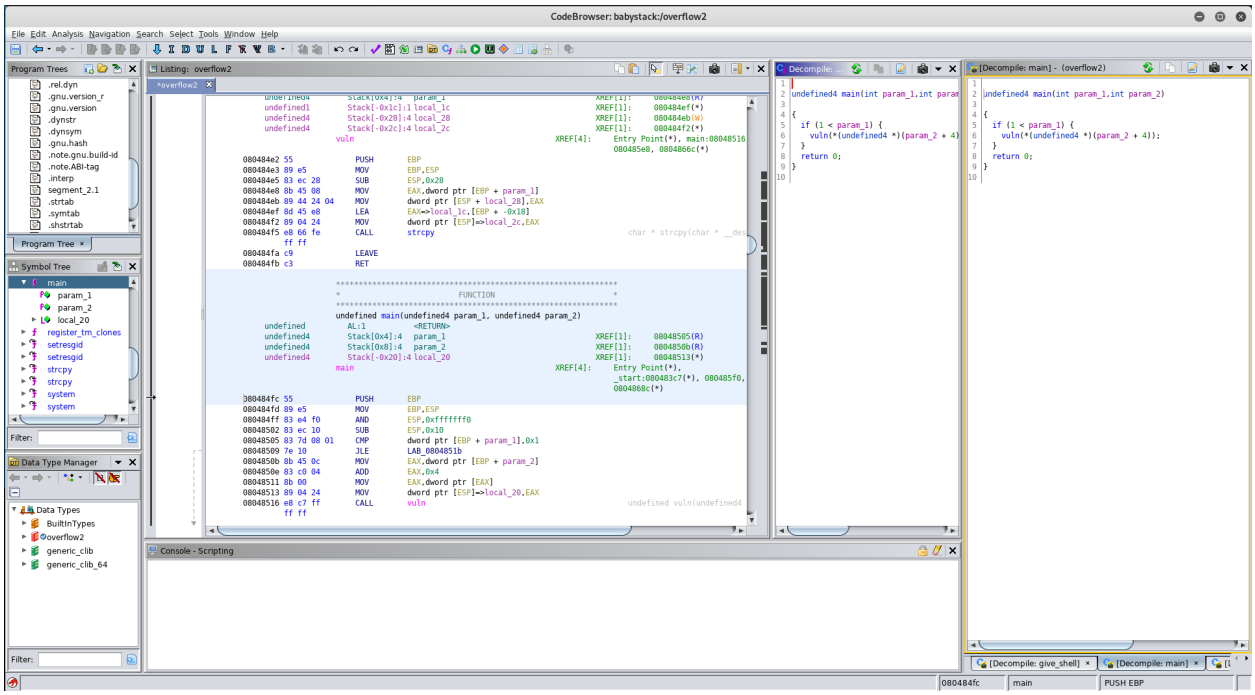

Figure 3.11: Ghidra

does next. Most programs don't have a convenient function to run system("/bin/sh"), so we need to find a way to manually invoke system or another exec function to get us our shell.Instead of returning to functions, return to instruction sequences followed by a return instruction, can return into middle of existing instructions to simulate different instructions. All we need are usable byte sequences anywhere in executable memory pages [80]. Various instruction sequences can be combined to form gadgets. Gadgets perform higher-level actions [19]

- Write specific 32-bit value to specific memory location
- Add/sub/and/or/xor value at memory location with immediate value
- Call function in shared library

So the idea of ROP is to rather than use a single libc function to run our shellcode, string together small pieces of code that are called gadgets. Most of the time the difficult part in ROP is to find the apt gadgets. [6]

3.6.3 ret2libc

Buffer overflow as we discussed previously is a vulnerability that occurs from programming errors and allows the attacker to overwrite other data.Return to libc is a exploitation technique that exploits the buffer overflow. A binary is dynamically linked and has a libc file. [78] This means that the whole set of standard library functions are located somewhere in the memory used by the program.In normal situations there is no functions that can call directly system(”/bin/sh”). Also with DEP/NX protection the attacker cannot inject and execute shellcode. As we said in the previous section most binaries are using dynamic linking and most of them are using libc. Inside libc there are a some functions that can be used to spawn a shell such as:

- system(”/bin/sh”)
- execve(”bin/sh”,null,null)

Until now in every binary we chase to use functions and code that can give us a shell like system().The general concept to take shell access is to find the base address of the libc then the address of the function we want to call and the push the string ”/bin/sh” to the stack. As we discussed in the Buffer Overflow ASLR is a security measure that randomizes stack and libc addresses on each program execution.If we want to randomize everything that would be very difficult in computation that’s why ASLR randomizes only the base address of libc.[35]

3.6.4 FSB

The Format String exploit occurs when the submitted data of an input string is evaluated as a command by the application. In this way, the attacker could execute code, read the stack, or cause a segmentation fault in the running application, causing new behaviors that could compromise the security or the stability of the system.3.6.439

To understand the attack, it’s necessary to understand the components that constitute it.

- The Format Function is an ANSI C conversion function, like printf, fprintf, which converts a primitive variable of the programming language into a human-readable string representation.

- The Format String is the argument of the Format Function and is an ASCII Z string which contains text and format parameters, like: printf (“Number value : %d”, variable);
- The Format String Parameter, like %x %s defines the type of conversion of the format function.

The attack could be executed when the application doesn’t properly validate the submitted input. In this case, if a Format String parameter, like %x, is inserted into the posted data, the string is parsed by the Format Function, and the conversion specified in the parameters is executed. However, the Format Function is expecting more arguments as input, and if these arguments are not supplied, the function could read or write the stack.In this way, it is possible to define a well-crafted input that could change the behavior of the format function, permitting the attacker to cause denial of service or to execute arbitrary commands.If the application uses Format Functions in the source-code, which is able to interpret formatting characters, the attacker could explore the vulnerability by inserting formatting characters in a form of the website. For example, if the printf function is used to print the value inserted in some fields of the page, the website could be vulnerable to this kind of attack, as showed below:

printf (Value); For example, if we give as input the format argument %x %x %x %x, printf will pop off four stack values and print them in hexadecimal, potentially leaking sensitive information.If we send the format argument %p %p %p %p, printf will pop off four stack addresses.This could give us an address that we will use it for chain it with another exploitation technique. printf can also index to an arbitrary ”argument” with the following syntax: ”%n$x” (where n is the decimal index of the argument you want).

4. EVALUATION

For the evaluation process we use a simple assessment test with general questions that will be covered in the lab where the students answer it two times. The first before the lab exploration and education begins and the other after they finish the lab. We don't expect from users to know everything after they finish the program as some of the exercises are advanced and require complex exploits, but we want to see how they respond in these series, their impressions and score themselves in knowledge.

4.1 Questions

1. Nmap by default scans; A.65535 B.100 C.1000 D.1024
2. If you were to see the following in a packet capture, what would you expect was happening? id=' or 1<3; A.Cross-site scripting B.Command injection C.SQL injection D.XML external entity injection
3. Cross-site scripting is an attack on the _____ that is based on the _____ trusting the _____.? A.user, user, website B.user, website, user C.website, website, user D.user, website, website
4. What protocol is used to take a destination IP address and get a packet to a destination on the local network? A.DHCP B.ARP C.DNS D.RARP
5. What would be the result of sending the string AAAAAAAAAAAAAAAA into a variable that has been allocated space for 8 bytes? A.Heap spraying B.SQL injection C.Buffer overflow D.Slowloris attack
6. If you want to scan all the live hosts in your local network, what CIDR notation (prefix) would you use? A./23 B./22 C./24 D./20
7. If you were to see the following in a packet capture, what would you expect was happening? ?page=../../../../../etc/passwd A.Unauthenticated file read B.Local file inclusion C.Directory traversal D.Local file disclosure
8. What attack would you expect to happen, we the following? ?page=php://filter/convert.base64-encode/resource=index.php A.Local file inclusion B.Php wrapper injection C.Php file read D.Unauthenticated file write

9. The prime difference between HTTP and HTTPS is: A.HTTP is encrypting the network traffic B.HTTPS uses AES algorithm to encrypt traffic and HTTP DES algorithm C.HTTPS uses TLS (SSL) to encrypt normal HTTP D.HTTP is using base64 encryption

10. The main difference between wget and curl is: A.curl can't do post requests B.wget is a network downloader C.curl is a tool to transfer data from or to a server D.B and C

11. What is brute-force password attack? A.Read passwords from file and check if it's the right one B.Trying every possible combination of chars,numbers,symbols,special chars C.B and A D.Make use of very big passwords

12. What has been done to the following string? %3Cscript%3Ealert(document.cookie);%3C/script%3E A.Base64 encoding B.URL encoding C.Encryption D.Cryptographic hashing

13. What would you get from running the command dig ns domain.com? A.Mail exchanger records for domain.com B.Name server records for domain.com C.Caching name server for domain.com D.IP address for the hostname ns

14. With checksec tool we are able to see? A.If a binary file has security vulnerabilities B.What architecture is the binary C.What security measurements are enable/disabled on the binary D.B and C

15. If you were to notice operating system commands inside a DNS request while looking at a packet capture, what might you be looking at? A.Tunneling attack B.DNS amplification C.DNS recursion D.XML entity injection

16. What would be the purpose of running a ping sweep? A.You want to identify responsive hosts without a port scan. B.You want to use something that is light on network traffic. C.You want to use a protocol that may be allowed through the firewall. D.All of the above.

17. 3-way handshake is? A.syn,ack,syn B.syn,syn,ack C.ack,syn,ack D.syn,syn-ack,ack

18. What would be one reason not to write malware in Python? A.Python interpreter is slow. B.Python interpreter may not be available. C.There is inadequate library support. D.Python is a hard language to learn.

19. If you saw the following command line, what would you be capturing?tcpdump -i eth0 host 192.168.1.2 A.Traffic just from 192.168.1.2 B.Traffic to and from 192.168.1.2 C.Traffic just to 192.168.1.2 D.All traffic other than from 192.168.86.5

20. What is Diffie-Hellman used for? A.Key management B.Key isolation C.Key exchange D.Key revocation

21. XML external entity injection allows an attacker to: A.Execute system commands B.Read files C.Perform DoS attack D.Perform Server Side Request Forgery

22. When we are hashing a password we can reverse it by? A.Decrypt the hash with the private key B.Hash it again C.We cannot unhash it D.Hashing is one way

23. SQLmap is a tool to test for? A.SQL injection B.NOSQL injection C.Cross Site Scripting D.Remote command execution

24. Cross site scripting is? A.Server side attack B.Client side attack C.Attack to steal users cookies D.Attack to inject malicious code

25. IPtables is a tool to? A.See our ip B.See the ip's of the network C.Set firewall rules D.See the protocols used in the network

26. Which of these is a reason to use an exploit against a local vulnerability? A.Pivoting B.Log manipulation C.Privilege escalation D.Password collection

27. Why we are hashing the passwords when they are stored in the database? A.It is quick and easy to store B.Because SQL accepts only hashed passwords C.Because if an attacker compromise the database is not easy to find the password D.So it's easy to reset the passwords

28. How to protect against an XSS? A.Use best practices of programming B.Sanitize the input of the user C.Use a WAF to protect against this attack D.Make a Whitelist of accepted inputs

29. Why the password "password" is a bad password? A.Because it will crash the server B.Because it can be guessed easily C.Because it contains only characters D.Because is small

30. Hydra is a tool to attack? A.Login forms B.Databases C.Mobile applications D.Binary files

31. If you see in a packet the following what are you suspecting? ?ps=sytem(”id”) A.Cross site scripting B.Cross site tracing C.Remote command execution D.Denial of Service

32. If you see in a packet the following what are you suspecting? delete.php?filename=logs.txt;id A.Command Injection B.SQL injection C.LDAP injection D.XML injection

33. What type of attack depends on the attacker entering JavaScript into a text area that is intended for users to enter text that will be viewed by other users? A.SQL injection B.Clickjacking C.Cross-site scripting D.Bluejacking

34. How an attacker can take advantage of a buffer overflow? A.Make a Denial of Service attack B.Read files C.Control the program D.Run system commands

35. In a Linux system, where is the password file stored? A./etc/passwd B./etc/shadow C./etc/user/password D./shadow/etc

36. What is the difference between eip and rip? A.rip is a register B.eip can store more bits C.rip is a general purpose register D.rip is in 64-bit mode

37. Which encryption algorithm is a symmetric stream cipher? A.AES B.ECC C.RC4 D.PGP

38. If you see in a packet the following what are you suspecting? http://example.domain.com/index.php A.Local file inclusion B.Server side request forgery C.Remote file inclusion D.Local file disclosure

39. If you see in a packet the following what are you suspecting? http://example.domain.com/index.php A.Local file disclosure B.Server side request forgery C.Remote file inclusion D.Open redirect

40. If you see in a packet the following what are you suspecting? http://example.domain.com/index.php A.Local file disclosure B.Server side request forgery C.Remote file inclusion D.Open redirect

41. If you see in a packet the following what are you suspecting? http://example.domain.com/index.php A.Local file disclosure B.Server side request forgery C.Remote file inclusion D.Open redirect

42. Which of the following is the correct XOR output? A. $0 \oplus 0 = 1$ B. $1 \oplus 0 = 0$ C. $1 \oplus 0 = 0$ D. $1 \oplus 1 = 0$

43. What happens in the synflood attack? A.The attacker leaks data from the server B.The attacker sends crafted data to make the server leak data C.The attacker sends syn-ack and the server response with syn D.The attacker sends many syn requests to perform Dos attack
44. Which input value would you utilize in order to evaluate and test for SQL injection vulnerabilities? A.SQL test B.admin and password C.|| or |! D.1'or'1'='1
45. What default port does SSH utilize? A.Port 22 B.Port 21 C.Port 443 D.Port 25
46. Which of the statements are True? A.Stack is LIFO B.Stack is FIFO C.Stack can be randomized D.Stack is the same for different binaries
47. In format string bug, a malicious user can: A.Leak the addresses of the stack B.Print in any format(hex,bin,etc) the values of the stack C.Overwrite the instruction pointer D.Reflect in which address the input is stored
48. Security measurements in binaries are? A.Canary B.ASLR C.NX D.PIE
49. GNU debugger is ? A.Only used for debugging B.Used to modify the code C.Used to modify registers D.Used to modify data structures
50. Every binary: A.Has function system inside that we can call B.Accepts shellcode with DEP/NX C.Has a standard libc address D.Use libc

Below are some results from the evaluation test before students start the lab course. Then number of the questions are fifty and we tried to have equal number of questions for each category. So for five categories and fifty questions, ten questions for each category. Some of the questions are more general and they cover more than one category or they test the knowledge of using tools for different purposes.4.1 4.2 4.3 4.4

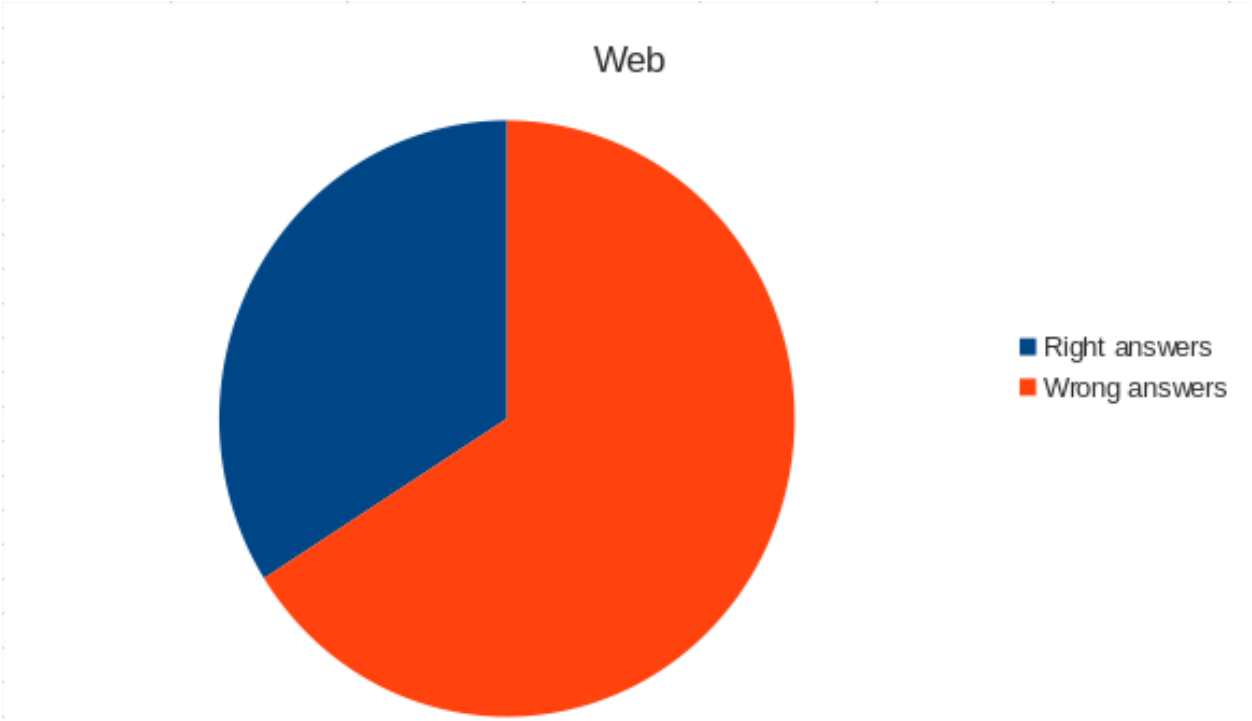

Figure 4.1: Web category

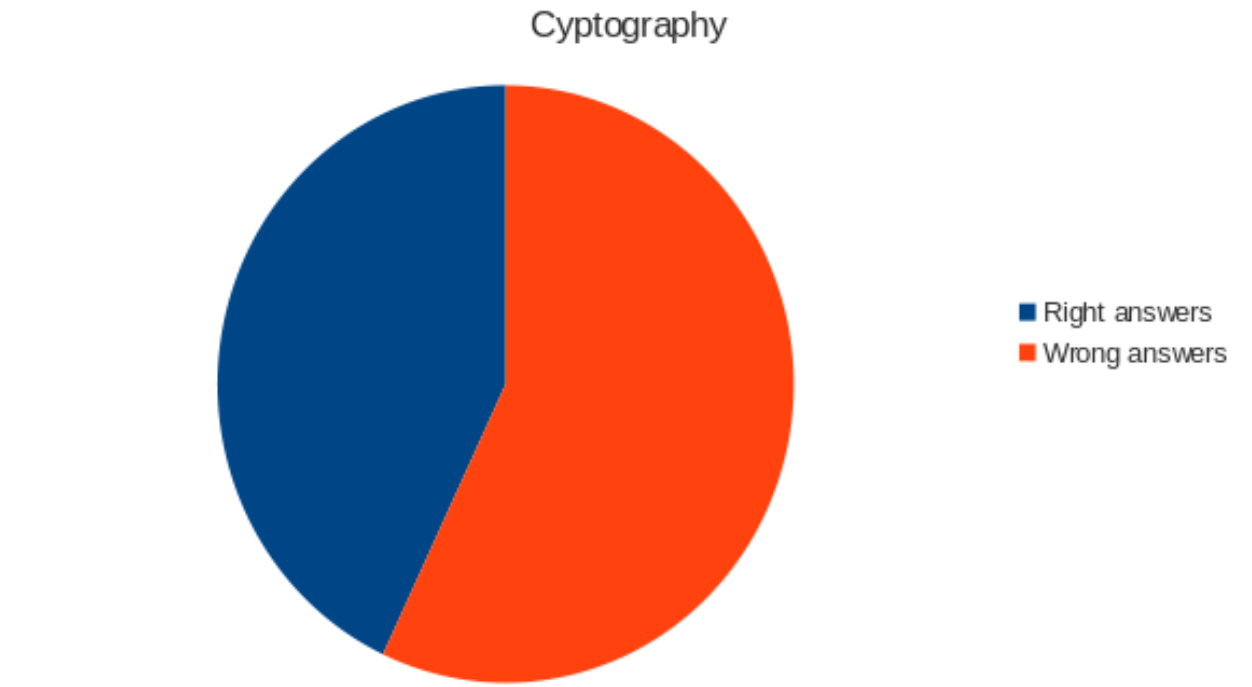

Figure 4.2: Cryptography

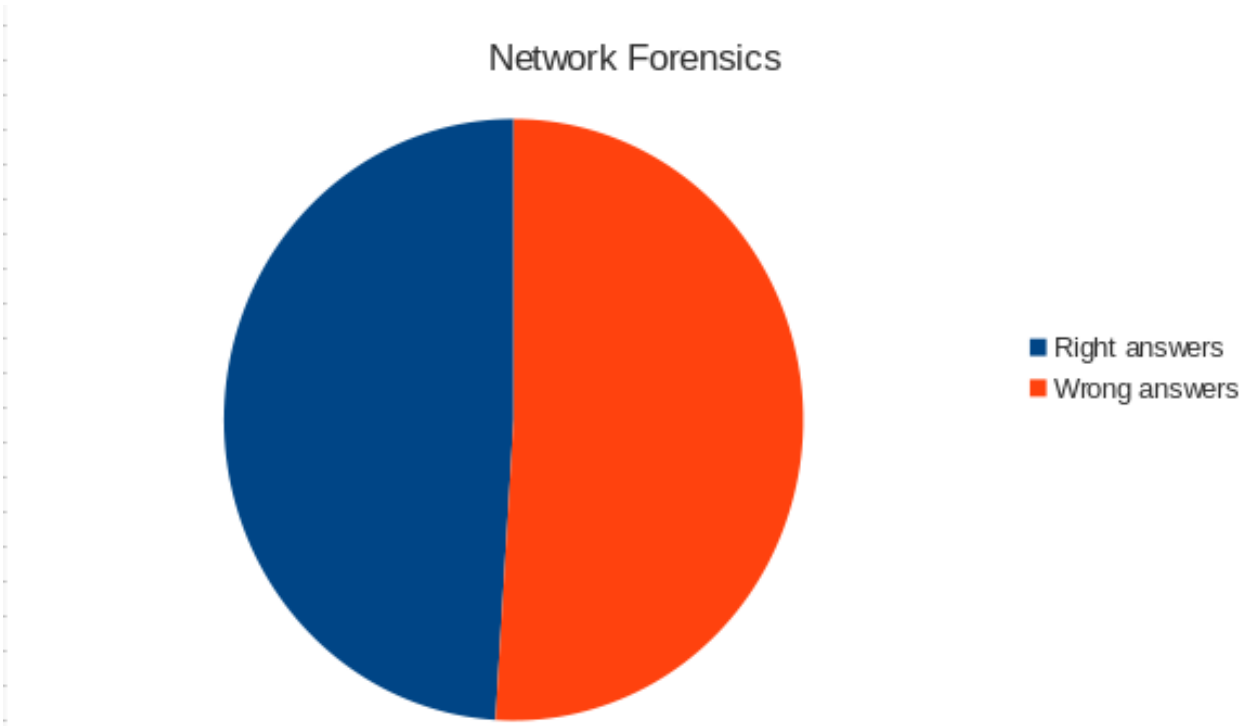

Figure 4.3: Forensics

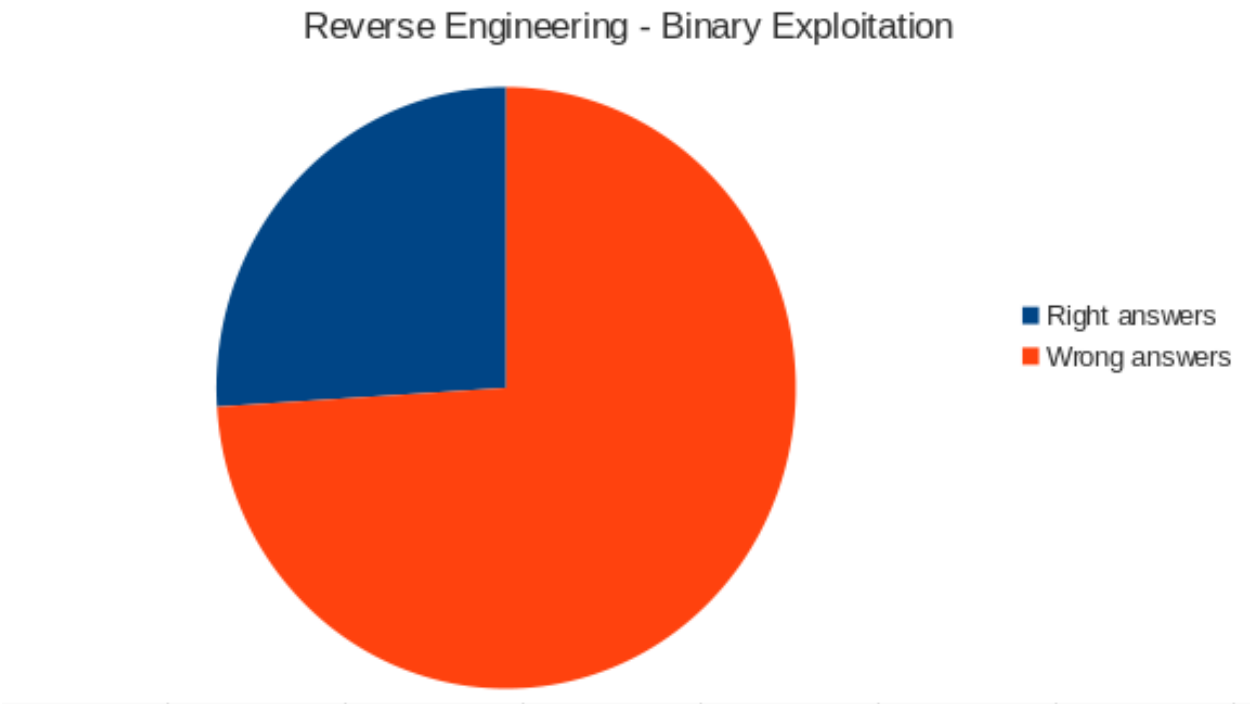

Figure 4.4: Reverse Engineering - Binary Exploitation

5. CONCLUSIONS AND FUTURE WORK

We have developed a general laboratory environment for cyber security education. Our UTHPentestBox environment is built upon a production OS (Linux). The environment can be setup on students personal computers or public computers with zero software cost and without the need of special hardware equipment regarding the Docker containers implementation. Based on the UTHPentestBox environment . The labs tested in our computer security courses in the last semester the evaluation results are quite encouraging.In our future work, we plan to further improve the existing labs, as well as develop more labs to cover a broader scope of computer security principles, more complex exploitation and based in real world examples.As evidenced by the completion data, we are still in the midst of executing this project.Thus far we have found the reception from the students and other users are quite positive.The users seem interested in the material and report that they have learned a considerable amount from this labs.The next logical step following completion of this work is the creation of an online program in a way that users as learning and solving the challenges of each category they will be submitting flags like CTF games and the process will be more gamified and interesting. There are countless difficulties in attempting to provide cyber security training via the Internet, though, and these difficulties will have to be overcome if meaningful online training is to be made available to transitioning users. An alternate direction for our future efforts might be to redesign the course with a focus on preparing participants for one of the industry accepted certification exams. Obviously this would be of great help to the students who believe they wish to pursue a career in cyber security or software development, as it would undoubtedly increase, their solid understanding of how things work, their experience on identifying and exploiting vulnerabilities, and be more ready for the workforce.

REFERENCES

[1] Mehdi-Laurent Akkar and Christophe Giraud. An implementation of des and aes, secure against some attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 309–318. Springer, 2001.

[2] Hussain Aldawood and Geoffrey Skinner. Educating and raising awareness on cyber security social engineering: A literature review. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 62–68. IEEE, 2018.

[3] Ross Anderson, Chris Barton, Rainer Böhme, Richard Clayton, Michel JG Van Eeten, Michael Levi, Tyler Moore, and Stefan Savage. Measuring the cost of cybercrime. In *The economics of information security and privacy*, pages 265–300. Springer, 2013.

[4] Daniel Bates, Adam Barth, and Collin Jackson. Regular expressions considered harmful in client-side xss filters. In *Proceedings of the 19th international conference on World wide web*, pages 91–100, 2010.

[5] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. Spongent: The design space of lightweight cryptographic hashing. *IEEE Transactions on Computers*, 62(10):2041–2053, 2012.

[6] Erik Buchanan, Ryan Roemer, Stefan Savage, and Hovav Shacham. Return-oriented programming: Exploitation without code injection. *Black Hat*, 8, 2008.

[7] Johannes Buchmann, Erik Dahmen, and Michael Szydlo. Hash-based digital signature schemes. In *Post-Quantum Cryptography*, pages 35–93. Springer, 2009.

[8] Franco Callegati, Walter Cerroni, and Marco Ramilli. Man-in-the-middle attack to the https protocol. *IEEE Security & Privacy*, 7(1):78–81, 2009.

[9] Nicholas Carlini and David Wagner. {ROP} is still dangerous: Breaking modern defenses. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 385–399, 2014.

[10] V Sharath Chandra and S Selvakumar. Bixsan: Browser independent xss sanitizer for prevention of xss attacks. *ACM SIGSOFT Software Engineering Notes*, 36(5):1–7, 2011.

[11] Li-Chiou Chen. Secure web development teaching modules. 2010.

[12] Li-Chiou Chen and Chienting Lin. Combining theory with practice in information security education. In *Proceedings of the 11th Colloquium for Information Systems Security Education*, pages 28–35, 2007.

[13] Ronald S Cheung, Joseph P Cohen, Henry Z Lo, and Fabio Elia. Challenge based learning in cybersecurity education. In *Proceedings of the International Conference on Security and Management (SAM)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer …, 2011.

[14] E. J. Chikofsky and J. H. Cross. Reverse engineering and design recovery: a taxonomy. *IEEE Software*, 7(1):13–17, 1990.

[15] Elliot J. Chikofsky and James H Cross. Reverse engineering and design recovery: A taxonomy. *IEEE software*, 7(1):13–17, 1990.

[16] Joseph L Christian. *Bug Bounty Programs: Analyzing the Future of Vulnerability Research*. PhD thesis, Utica College, 2018.

[17] Vicka Corey, Charles Peterman, Sybil Shearin, Michael S Greenberg, and James Van Bokkelen. Network forensics analysis. *IEEE Internet Computing*, 6(6):60–66, 2002.

[18] Italo Dacosta, Saurabh Chakradeo, Mustaque Ahamad, and Patrick Traynor. One-time cookies: Preventing session hijacking attacks with stateless authentication tokens. *ACM Transactions on Internet Technology (TOIT)*, 12(1):1–24, 2012.

[19] Dino Dai Zovi. Practical return-oriented programming. *SOURCE Boston*, 2010.

[20] Henry Dalziel. *Introduction to US Cybersecurity Careers*. Syngress, 2014.

[21] Giovanni Di Crescenzo and Faramak Vakil. Cryptographic hashing for virus localization. In *Proceedings of the 4th ACM workshop on Recurring malcode*, pages 41–48, 2006.

[22] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[23] Chris Dods, Nigel P Smart, and Martijn Stam. Hash based digital signature schemes. In *IMA International Conference on Cryptography and Coding*, pages 96–115. Springer, 2005.

[24] Wenliang Du. Seed: hands-on lab exercises for computer security education. *IEEE Security & Privacy*, 9(5):70–73, 2011.

[25] Zakir Durumeric, James Kasten, Michael Bailey, and J Alex Halderman. Analysis of the https certificate ecosystem. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 291–304, 2013.

[26] Jon Erickson. *Hacking: the art of exploitation*. No starch press, 2008.

[27] John Franks, Phillip Hallam-Baker, Jeffrey Hostetler, Scott Lawrence, Paul Leach, Ari Luotonen, and Lawrence Stewart. Http authentication: Basic and digest access authentication, 1999.

[28] Eric Fulton, Cameron Lawrence, and Shawn Clouse. White hats chasing black hats: Careers in it and the skills required to get there. *Journal of Information Systems Education*, 24(1):8, 2019.

[29] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[30] Md Maruf Hassan, Shamima Sultana Nipa, Marjan Akter, Rafita Haque, Fabiha Nawar Deepa, Mostafijur Rahman, Md Asif Siddiqui, Md Hasan Sharif, et al. Broken authentication and session management vulnerability: A case study of web application. *International Journal of Simulation Systems, Science & Technology*, 19(2):6–1, 2018.

[31] Hideaki Hata, Mingyu Guo, and M Ali Babar. Understanding the heterogeneity of contributors in bug bounty programs. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 223–228. IEEE, 2017.

[32] George Hatzivasilis, Ioannis Papaefstathiou, and Charalampos Manifavas. Password hashing competition-survey and benchmark. *IACR Cryptology ePrint Archive*, 2015:265, 2015.

[33] Daniel Hoffman, Durga Prabhakar, and Paul Strooper. Testing iptables. In *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*, pages 80–91. IBM Press, 2003.

[34] Ji Hu, Christoph Meinel, and Michael Schmitt. Tele-lab it security: an architecture for interactive lessons for security education. *ACM SIGCSE Bulletin*, 36(1):412–416, 2004.

[35] Zi-Shun Huang and Ian G Harris. Return-oriented vulnerabilities in arm executables. In *2012 IEEE Conference on Technologies for Homeland Security (HST)*, pages 1–6. IEEE, 2012.

[36] Michal Hubczyk, Adam Domanski, and Joanna Domanska. Local and remote file inclusion. In *Internet-Technical Developments and Applications 2*, pages 189–200. Springer, 2012.

[37] Jiang Huiping. Strong password authentication protocols. In *2010 4th International Conference on Distance Learning and Education*, pages 50–52. IEEE, 2010.

[38] Fei Huo and Guang Gong. Xor encryption versus phase encryption, an in-depth analysis. *IEEE Transactions on Electromagnetic Compatibility*, 57(4):903–911, 2015.

[39] Alexa Huth, Michael Orlando, and Linda Pesante. Password security, protection, and management. *United States Computer Emergency Readiness Team*, 2012.

[40] Philip G Inglesant and M Angela Sasse. The true cost of unusable password policies: password use in the wild. In *Proceedings of the sigchi conference on human factors in computing systems*, pages 383–392, 2010.

[41] Gordon Johnson. Remote and local file inclusion explained. *Hacking9*, 2008.

[42] Andrew Johnson-Laird. Software reverse engineering in the real world. *U. Dayton l. rev.*, 19:843, 1993.

[43] Ed Kaiser and Wu chang Feng. mod_kapow: Mitigating dos with transparent proof-of-work. In *Proceedings of the 2007 ACM CoNEXT conference*, pages 1–2, 2007.

[44] Charles W Kaufman, Radia J Pearlman, and Morrie Gasser. System for increasing the difficulty of password guessing attacks in a distributed authentication scheme employing authentication tokens, February 13 1996. US Patent 5,491,752.

[45] Sara Kraemer and Pascale Carayon. Human errors and violations in computer and information security: The viewpoint of network administrators and security specialists. *Applied ergonomics*, 38(2):143–154, 2007.

[46] David M Kristol. Http cookies: Standards, privacy, and politics. *ACM Transactions on Internet Technology (TOIT)*, 1(2):151–198, 2001.

[47] Yogesh Kumar, Rajiv Munjal, and Harsh Sharma. Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures. *International Journal of Computer Science and Management Studies*, 11(03):60–63, 2011.

[48] Giovanni Lagorio. Introduction to binary exploitation on linux. 2017.

[49] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.

[50] Aron Laszka, Mingyi Zhao, and Jens Grossklags. Banishing misaligned incentives for validating reports in bug-bounty platforms. In *European Symposium on Research in Computer Security*, pages 161–178. Springer, 2016.

[51] F Lauria and A Oliveri. Istituto di informatica e telematica. 2018.

[52] I-En Liao, Cheng-Chi Lee, and Min-Shiang Hwang. A password authentication scheme over insecure networks. *Journal of Computer and System Sciences*, 72(4):727–740, 2006.

[53] Virginia Loh-Hagan. *Ethical Hacker*. Cherry Lake, 2015.

[54] Carlos A Lozano and Shahmeer Amir. *Bug Bounty Hunting Essentials: Quick-paced guide to help white-hat hackers get through bug bounty programs*. Packt Publishing Ltd, 2018.

[55] Haibo Luo. Ssrf vulnerability attack and prevention based on php. In *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pages 469–472. IEEE, 2019.

[56] Akash Mahajan. *Burp Suite Essentials*. Packt Publishing Ltd, 2014.

[57] Suresh S Malladi and Hemang C Subramanian. Bug bounty programs for cybersecurity: Practices, issues, and recommendations. *IEEE Software*, 37(1):31–39, 2019.

[58] Steve Mansfield-Devine. Significant rise in cybercrime against public sector organisations, 2012.

[59] Simon Marechal. Advances in password cracking. *Journal in computer virology*, 4(1):73–81, 2008.

[60] Wojciech Mazurczyk and Luca Caviglione. Information hiding as a challenge for malware detection. *arXiv preprint arXiv:1504.04867*, 2015.

[61] Charlie McMurdie. The cybercrime landscape and our policing response. *Journal of Cyber Policy*, 1(1):85–93, 2016.

[62] Natarajan Meghanathan, Sumanth Reddy Allam, and Loretta A Moore. Tools and techniques for network forensics. *arXiv preprint arXiv:1004.0570*, 2010.

[63] Xiaozhu Meng and Barton P Miller. Binary code is not easy. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, pages 24–35, 2016.

[64] Preda Mihailescu. The fuzzy vault for fingerprints is vulnerable to brute force attack. *arXiv preprint arXiv:0708.2974*, 2007.

[65] Dennis Mirante and Justin Cappos. Understanding password database compromises. *Dept. of Computer Science and Engineering Polytechnic Inst. of NYU, Tech. Rep. TR-CSE-2013-02*, 2013.

[66] Steve Moyle. The blackhat's toolbox: Sql injections. *Network security*, 2007(11):12–14, 2007.

[67] Praveen Murthy. Reducing buffer overflow, December 12 2019. US Patent App. 16/005,512.

[68] Siva G Narendra, Prabhakar Tadepalli, and Thomas N Spitzer. Hybrid symmetric/asymmetric cryptography with user authentication, May 29 2012. US Patent 8,189,788.

[69] David Naylor, Alessandro Finamore, Ilias Leontiadis, Yan Grunenberger, Marco Mellia, Maurizio Munafò, Konstantina Papagiannaki, and Peter Steenkiste. The cost of

the” s” in https. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 133–140, 2014.

[70] William Newhouse, Stephanie Keith, Benjamin Scribner, and Greg Witte. National initiative for cybersecurity education (nice) cybersecurity workforce framework. *NIST Special Publication*, 800:181, 2017.

[71] tefan Nicula and Răzvan Daniel Zota. Exploiting stack-based buffer overflow using modern day techniques. *Procedia Computer Science*, 160:9–14, 2019.

[72] K Nirmal, B Janet, and R Kumar. Web application vulnerabilities-the hacker’s treasure. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 58–62. IEEE, 2018.

[73] Gregor Overney. Secure file verification station for ensuring data integrity, October 2 2003. US Patent App. 10/112,473.

[74] Marius Popa. Binary code disassembly for reverse engineering. *Journal of Mobile, Embedded and Distributed Systems*, 4(4):233–248, 2012.

[75] Gregor N Purdy. *Linux iptables Pocket Reference: Firewalls, NAT & Accounting*. ” O’Reilly Media, Inc.”, 2004.

[76] Amit Raikar and Guruprasad Ramarao. Method and system for establishing a consistent password policy, December 7 2010. US Patent 7,849,320.

[77] Tim Ring. Why bug hunters are coming in from the wild. *Computer Fraud & Security*, 2014(2):16–20, 2014.

[78] AliAkbar Sadeghi, Farzane Aminmansour, and Hamid Reza Shahriari. Tiny jump-oriented programming attack (a class of code reuse attacks). In *2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, pages 52–57. IEEE, 2015.

[79] Fred B Schneider. Cybersecurity education in universities. *IEEE Security & Privacy*, 11(4):3–4, 2013.

[80] Hovav Shacham. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 552–561, 2007.

[81] Hossain Shahriar, Md Arabin Islam Talukder, Mohammad Rahman, Hongmei Chi, Sheikh Ahamed, and Fan Wu. Hands-on file inclusion vulnerablity and proactive control for secure software development. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 604–609. IEEE, 2019.

[82] Richard Shay, Saranga Komanduri, Patrick Gage Kelley, Pedro Giovanni Leon, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Encountering stronger password requirements: user attitudes and behaviors. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, pages 1–20, 2010.

[83] Gurpreet Singh. A study of encryption algorithms (rsa, des, 3des and aes) for information security. *International Journal of Computer Applications*, 67(19), 2013.

[84] Tarun Soin, Vineet Dixit, and Yixin Sun. Detecting stolen authentication cookie attacks, December 13 2011. US Patent 8,079,076.

[85] Eugene H Spafford. Opus: Preventing weak password choices. *Computers & Security*, 11(3):273–278, 1992.

[86] Clare Stevens. Assembling cybersecurity: The politics and materiality of technical malware reports and the case of stuxnet. *Contemporary Security Policy*, 41(1):129–152, 2020.

[87] Nataliya B Sukhai. Hacking and cybercrime. In *Proceedings of the 1st annual conference on Information security curriculum development*, pages 128–132, 2004.

[88] Wayne C Summers and Edward Bosworth. Password policy: the good, the bad, and the ugly. In *Proceedings of the winter international synposium on Information and communication technologies*, pages 1–6. Trinity College Dublin, 2004.

[89] Blair Taylor and Siddharth Kaza. Security injections@ towson: Integrating secure coding into introductory computer science courses. *ACM Transactions on Computing Education (TOCE)*, 16(4):1–20, 2016.

[90] Jawahar Thakur and Nagesh Kumar. Des, aes and blowfish: Symmetric key cryptography algorithms simulation based performance analysis. *International journal of emerging technology and advanced engineering*, 1(2):6–12, 2011.

[91] Alexander Tse, Reuben Heredia, Mohammed Taher, and Jordan Chiou. ibibliography sql injection solutions. *University of British Columbia*, 2008.

[92] David Wagner and Ian Goldberg. Proofs of security for the unix password hashing algorithm. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 560–572. Springer, 2000.

[93] Xinli Wang, Yan Bai, and Guy C Hembroff. Hands-on exercises for it security education. In *Proceedings of the 16th Annual Conference on Information Technology Education*, pages 161–166, 2015.

[94] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 162–175, 2010.

[95] Andrew White, Alan Tickle, and Andrew Clark. Overcoming reputation and proof-of-work systems in botnets. In *2010 Fourth International Conference on Network and System Security*, pages 120–127. IEEE, 2010.

[96] Dave Wichers and Jeff Williams. Owasp top-10 2017. *OWASP Foundation*, 2017.

[97] Jianxin Jeff Yan. A note on proactive password checking. In *Proceedings of the 2001 workshop on New security paradigms*, pages 127–135, 2001.

[98] Mingyi Zhao, Aron Laszka, and Jens Grossklags. Devising effective policies for bug-bounty platforms and security vulnerability discovery. *Journal of Information Policy*, 7:372–418, 2017.

5.1 Links and Websites

1. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6573305`
2. `https://www.josephpcohen.com/papers/cbl.pdf`
3. `https://cyber-peace.org/wp-content/uploads/2017/07/Hackers-Are-Targeting-Nuclear-and-F.B.I.pdf`
4. `https://ieeexplore.ieee.org/abstract/document/7085972`
5. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1392709`
6. `https://ieeexplore.ieee.org/abstract/document/4402456`

7. https://ohud.in.net/18-34-48.pdf
8. https://dl.acm.org/doi/abs/10.1145/2047594.2047628
9. https://www.sciencedirect.com/science/article/abs/pii/S000368700600041X
10. https://www.sciencedirect.com/science/article/abs/pii/S0260691718305975
11. https://www.usenix.org/conference/3gse14/summit-program/presentation/chapman
12. https://journal.unnes.ac.id/sju/index.php/ujmer/article/view/24217
13. https://www.cyberdegrees.org/jobs/penetration-tester/
14. https://www.cyberdegrees.org/jobs/security-specialist/
15. https://academic.oup.com/cybersecurity/article/3/2/81/4524054
16. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2418812
17. https://link.springer.com/chapter/10.1007/978-3-662-58387-6_8
18. http://web.mit.edu/smadnick/www/wp/2016-23.pdf
19. http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A723516&dswid=3348
20. https://www.hackerone.com/services
21. https://bugcrowd.com
22. https://www.intigriti.com
23. http://brutelogic.com.br
24. https://en.wikipedia.org/wiki/HTTP_cookie
25. https://en.wikipedia.org/wiki/XOR_cipher
26. https://www.blackhat.com/docs/eu-15/materials/eu-15-Stasinopoulos-Commix-Detecting-
pdf
27. https://github.com/Ganapati/RsaCtfTool
28. https://en.wikipedia.org/wiki/Substitution_cipher

29. https://ldapwiki.com/wiki/ClientHello

30. https://ldapwiki.com/wiki/ServerHello

31. https://en.wikipedia.org/wiki/Cryptographic_hash_function

32. https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material/documents/introduction-to-network-forensics-handbook.pdf

33. https://wiki.wireshark.org/

34. https://www.booleanworld.com/depth-guide-iptables-linux-firewall

35. https://www.yeswehack.com/

36. https://www.infoworld.com/article/2624136/security-lessons-still-lacking-for-com
html

37. https://www.offensive-security.com

38. https://www.kali.org/

39. https://owasp.org/www-community/attacks/Format_string_attack

40. https://github.com/Gallopsled/pwntools

41. https://github.com/JonathanSalwan/ROPgadget