EXPERIMENTAL PENETRATION TESTING TEACHING AND LEARNING FOR HIGH SCHOOL STUDENTS USING CLOUD COMPUTING

UNIVERSITY OF TURKU

DEPARTMENT OF FUTURE TECHNOLOGIES
MASTER OF SCIENCE IN TECHNOLOGY THESIS
NETWORKED SYSTEMS SECURITY
JUNE 2019
BERIOSKA CONTRERAS VARGAS

SUPERVISOR:
JANI VOUTILAINEN (CITY OF TURKU).

EXAMINERS:
SEPPO VIRTANEN
ANTTI HAKKALA

UNIVERSITY OF TURKU
DEPARTMENT OF FUTURE TECHNOLOGIES

BERIOSKA CONTRERAS VARGAS:

EXPERIMENTAL PENETRATION TESTING TEACHING AND LEARNING FOR HIGH SCHOOL STUDENTS USING CLOUD COMPUTING

MASTER OF SCIENCE IN TECHNOLOGY THESIS, 75 P.
NETWORKED SYSTEMS SECURITY
JUNE 2019

The need for high school students trained in ICT to developing cybersecurity skills implies the understanding of threats on security. Considering that the aim of hacking is to circumvent restrictions, the goal of this experimental course is to train students in understanding hacking to improve security. Currently, the reality of hacking has alarmingly evolved and shaped an undeniable black market of information where talented teenagers are not exempt to partake. Despite the fact that the formal teaching and learning of hacking inside high schools can be seen as miseducation, that misunderstanding is faced in this work by addressing both the defensive and offensive security from the perspective of penetration testing.

By developing progressive challenges over an adaptive cloud environment, the students can be taught hacking from a constructive perspective. A cloud-based attack surface is implemented which consists of a set of systems gradually prepared by means of scripts. The theoretical and practical lessons are directed by a set of scaffolded and constructivist challenges. The discussion about ethics is confronted and remains present throughout the teaching and learning process. Finally, the results and empirical findings of the students are analyzed and measured demonstrating that high school students can acquire skills to protect information for the community, and for themselves.

Keywords: penetration testing, hacking, teaching and learning, progressive learning, high school, cloud computing.

ACKNOWLEDGMENTS

To my family, Marietta and Rodrigo, for supporting me unconditionally.

The Hack with Turku course is a collaborative effort of Kerttulin lukio, City of Turku and the University of Turku. The course was also supported by Lähitapiola, and speakers from CGI Suomi Oy, Helsinki Police Department, Microsoft Oy, Nixu Oyj, Second Nature Security Oy and TurkuSec Association.

TABLE OF CONTENTS

TABLE OF FIGURES

ABBREVIATIONS

API	Application Programming Interface
ARP	Address Resolution Protocol
CISSP	Certified Information Systems Security Professional
CORS	Cross-Origin Resource Sharing
CRLF	Carriage Return and Line Feed
CSRF	Cross Site Request Forgery
DOM	Document Object Model
FGPP	Fine-Grained Password Policy
HTTP	Hypertext Transfer Protocol
HTML	Hyper Text Markup Language
HSTS	Strict Transport Security
ICT	Information and Communication Technologies
LDAP	Lightweight Directory Access Protocol
NMAP	Network Mapper
OWASP	Open Web Application Security Project
PHP	Hypertext Preprocessor
RFC	Request for Comment
RSYNC	Remote Synchronization Protocol
SDLC	Software Development Life Cycle
SQL	Structure Query Language
SMB	Server Message Block
SSH	Secure Shell
SSL	Secure Socket Layer
TLS	Transport Layer Security
VNC	Virtual Network Computing
XSS	Cross Site Scripting

1. INTRODUCTION

This work originates in the hallways of Kerttulin lukio, an upper secondary school (high school) located in Turku Finland. Throughout this thesis, we will refer to this school simply as Kerttuli. The students at Kerttuli pursue a general context of learning that according to their motivations can be combined with specialized studies tracks, such as sports or information and communication technologies (ICT). The Kerttuli School of ICT track is a national program accepted by the Ministry of Education of Finland in the autumn 2000 [1][2].

In February 2019, the call for joining this experimental penetration testing course received forty applications from the students confirming the enthusiasm at Kerttuli. The course adopted the name Hack with Turku in recognition to the pioneer initiative of City of Espoo who organized the first call to general schools for Hack with Espoo in 2018. Despite the similar name, the teaching and learning of Hack with Turku differ from Espoo significantly. The majority of the students enrolled have an existing ICT background allowing them to adapt to the progressive learning proposed in this work. Consisting of a group of 20 participating students, 85% of them belong to the ICT track while 15% were general school students.

The need for high school students trained in ICT to developing cybersecurity skills implies the understanding of threats on security. Since a hacker's aim is to circumvent restrictions, then the goal of this experimental course is to train students to understand hacking to improve security. In addition, it is essential to practice analyzing possible solutions due to the cognitively demanding nature to understand all possible attacks and their causes and impacts. For this reason, a progressive learning and an adaptive cloud environment encouraging a constructive perspective based on a penetration testing methodology was created.

The work presented in this thesis proposes an experimental penetration testing teaching and learning method using cloud computing, and it is organized in five main chapters including this Introduction. The second chapter analyzes the student's curriculum, the penetration testing methodology and educational methods to define a suitable content scope framing the model of challenges. The third chapter introduces the experimental research influenced by progressive learning. The research process emphasizes the development of constructivist challenges that are gradually implemented to be measured. The fourth chapter introduces a scalable cloud infrastructure that serves multiple student instances to build progressive learning. A student instance is conceived as an adaptable environment that progressively mutates according to the logic stated in a set of scripts. The fifth chapter measures the students' results and empirical findings by verifying the challenges' achievement and timing, and it proposes a ranking that is announced at the end of the cycle. Finally, the sixth chapter summarizes the conclusions of this work and suggests what should be considered as future work.

2. STATE OF THE ART

2.1. STUDENTS CURRICULUM

The ICT special study track consists of multiple branches of knowledge, so the students have the flexibility to pursue elective courses according to their interest. Indeed, it is precisely the student's personal interests that motivates the most this new special course. About 40 ICT-specific courses are offered in the ICT track, and the courses also include humanities and natural sciences courses. The graduates of the special edition complete at least 12 special courses and do not participate up to 8 courses of the compulsory ones of the other subjects [2].

The following table represents a set of ICT courses that are considered in the content scope to support the development of challenges:

#	ICT Course	#	ICT Course
ICT1	Digital Citizen	ICT12	Game Development
ICT2	Tensor System	ICT13	Cisco CCNA I Part 1
ICT3	Communication Technologies	ICT14	Cisco CCNA I Part 2
ICT4	Internet	ICT15	Cisco CCNA II
ICT5	Basic Course in Programming	ICT16	Information Technology and Cybersecurity
ICT6	Data Structures	ICT17	Data Management
ICT7	Advanced Course in Programming	ICT18	Information Technology now and in the future
ICT8	HTML5	ICT19	Linux
ICT9	Computing Image Processing	ICT20	Robotics and Electronic
ICT10	Animation	ICT21	Special Course of the Year
ICT11	Game Design	ICT22	Project Course

Figure 1: ICT Track Curriculum Sample.

2.2. PENETRATION TESTING METHODOLOGY

A penetration test aims to exploit existing vulnerabilities to determine their nature and impact. The goal is to simulate an attack on a system or network to evaluate the risk of the environment. The penetration testing process is abbreviated as pen-test, an also referred by many names, such as, ethical hacking, red teaming and vulnerability testing [3]. The discussion about ethical hacking is controversial in high schools as the composite statement expresses two opposing meanings: a hacker who aims to circumvent restrictions while an ethical attribute should confront them. Such discrepancy derives in

complex decisions that any individual makes of free will. Conversely, from the affected position, any attempt to trespass a restriction is understood as a flagrant crime.

Any selected penetration testing framework proves by itself that hacking and ethics are independent concepts that require context and purpose. Therefore, a pen-test starts by defining clear objectives and scope seeking for agreement to avoid adverse effects on the confidentiality, integrity and availability of the information that we intend to protect. The purpose is to identify the vulnerabilities that may exist in operating systems, services and applications due to flaws, improper configurations or risky end-user behavior by simulating a break-in. Precursors in the prevention of intruders by using break-in examples were Farmer and Venema [4][5]. They examined the traces left by a set of intrusion techniques they performed on their own systems to demonstrate the impact of such threats. After the analysis, Farmer and Venema developed the Security Analysis Tool for Auditing Networks, or SATAN, an incipient network scanning tool developed in 1993. A few years later, Ross D. from Microsoft published in 1999 the first paper demonstrating a Script Injection into web content. The former publication extended the study about Cross Site Scripting conducted over Internet Explorer, as the issue proved the exploitation from the server side [6]. Hence, a penetration test is intended to validate the efficacy of defensive mechanisms over digital assets.

At present, the penetration testing methodologies are diverse as the testing is frequently tailored by the pen-tester according to the conditions of the environment. An intentionally abstract methodology is preferred over a specific framework in this work since it is more flexible to design the model of challenges. While a methodology introduces a set of processes following a predetermined sequence, the framework is useful to guide a penetration testing under a specific structure and techniques. The frameworks reviewed are the Technical Guide to Information Security Testing and Assessment SP-800-115 published by NIST in 2008 [7], the Open Source Security Methodology Manual OSSTMM v3 introduced by ISECOM in 2010 [8], and the Penetration test framework introduced and maintained by Kevin Orrey [9]. In addition, the Penetration Testing Execution Standard (PTES) proposed by PTES Group in 2009 [10], the ISO/IEC/IEEE 29119 Software and System Engineering - Software Testing released in 2013 [11], and the methodology described by A. Gordon [3] in the CISSP common body of knowledge released in 2015 were reviewed.

Consequently, the methodology adapted to this work considers a set of four stages as follows:

- **Reconnaissance and Discovery**: Searching for any available information on the target avoiding intrusive methods.
- **Enumeration and Vulnerability Analysis**: Intrusive data gathering. Mapping of known vulnerabilities.
- **Execution and Appropriation**: Attempt to gain temporary or permanent users and privileged access based on an attack plan and access strategy.
- **Document Findings**: Document the security state of the environment. It provides the test results by means of empirical evidences and remediations.

2.3. PROGRESSIVE LEARNING

Progressive education is a quest for non-traditional methods emphasizing a direct experience to motivate a continuous learning process. In this sense, the teaching and learning involves periodic experiences associated to a given context of interest. By experiencing active contexts, a student is stimulated to reflect, explore and inquire. Multiple progressive teaching methods are designed to raise motivation, a goal that is rarely achieved when students remain as passive actors in a traditional education environment.

It is difficult to pinpoint the birth of progressive learning on a global timeline. One point in time is the transition from the industrial society to the information society in the latest 19th century where we can notice changes in the skills exhibited by the labor force. To exemplify, the 20th century information society is characterized by analytical skills, problem solving, lifelong learning and team work that further extend the orientation of the first industrial era mostly based on discipline and efficiency. In the 21st century, we are transiting through the knowledge society where critical thinking, self-motivation, frustration tolerance, communication skills, information and technology skills, and self-esteem and confidence are distinguished as expected attributes [12].

Two methods introduced by progressive learning are Scaffolding and Constructivism. The scaffolding method consists of understanding the degree of support and mentoring the students require to develop the skills needed to solve a challenge. A challenge acts here as a brain teaser where students have minimal instructions and are awarded with either standard or bonus points. The challenge is influenced by the capture-the-flag hacking method [13], yet it differentiates in that a challenge aims to awake coding skills and the challenge can be solved in different creative ways. Also, a challenge suggests the idea that there is no dependence on any tool, but instead a hacker can become the maker of their own tools [14].

In the work of Barzilai and Blau [15], a scaffolding method is applied to integrate learning content within a game, enhancing the connections between enjoyment and formal knowledge. Pre and post problem-solving assessments are carried out on the students under three conditions: the students play only without scaffolded study, the scaffolding occurs previously to the play/game, and the play/game is scaffolded after. The results revealed that the students performed better when the play/game was scaffolded before. Otherwise the game based on itself is not a determinant in the problem-solving performance.

On the other hand, the constructivism method is rooted in the interactions or the collective experience to construct either individual or collaborative knowledge [16]. Throughout constructivist learning the students examine an experience from multiple views increasing their critical thinking and collaboration. Thus, the students construct the meaning rather than merely reproduce content. From this point of view, the teacher becomes a facilitator of the learning environment and provide the students opportunities to actively engage in self-regulated learning and collaborative learning practices [17]. This is the learning to

learn process that is driven by the challenges once scaffolded by the hands-on lessons and lectures.

2.4. DISCUSSION

In 2018, about 130,000 user accounts and plaintext passwords were revealed in Finland becoming the third largest data breach registered [18]. The security of a service was insufficient to prevent the attack against an open website. Currently, the global reality of hacking has evolved alarmingly shaping an undeniable black market where information is traded and where talented teenagers are not exempt to partake.

Approaching hacking inside high schools could be seen as miseducation. However, having students trained in ICT develop cybersecurity skills necessarily implies the realistic understanding of threats that affect security. The practice of penetration testing brings the benefit to discover earlier potential flaws in the information systems by addressing both the offensive and defensive perspectives together. Hence, the goal is to train students to understand hacking so as to improve the security. Students in ICT require the skills to protect information for the community and for themselves.

It is acknowledged that a penetration tester has strong programming and computer networking skills [19]. In addition, it is cognitively demanding to understand all possible attacks and their causes and impacts. Therefore, the teaching and learning process for penetration testing is impossible when it lacks a realistic environment to practice in, much more when such a scenario is not legal. Also, the teaching and learning can be diluted if a system is broken by mimicking a fun attack without any reasoning about the conditions causing such behavior that a student can also contribute to improve. Thus, the following experimental work is focused on answering the following questions:

- Why are progressive learning methods worth adopting to encourage penetration testing learning?

While the majority of the participating students have an ICT background, there is still a challenging diversity of knowledge derived from a flexible curriculum, and also there is a minor disparity of knowledge in the cases where ICT is not an integral part of the general study. Thus, by assuming that an ICT background is determinant in the learning process, the aim is to identify elements or instruments to reconcile the diversity of knowledge and the student's motivation.

- What are the implications in combining progressive learning over a cloud-based penetration testing environment?

Assuming that learning penetration testing is absolutely practical, the interest is to identify elements or instruments to balance curiosity and critical thinking.

3. EXPERIMENTAL RESEARCH METHOD

3.1. RESEARCH PROCESS

In this work, a progressive model of challenges is developed, applied and measured. Through the examination of the state of the art, a penetration testing methodology is considered abstract enough to be integrated with progressive learning and the student's curriculum. The whole period of this experimental initiative is about 20 sessions distributed in 10 weeks, including three special sessions with guest speakers. A normal session consists in 60 minutes of a formal theoretical lecture followed by 60 minutes of a hands-on lesson. The model of challenges is comprised of 10 constructivist experiences/experiments that are evaluated and measured based on accomplishment and timing.

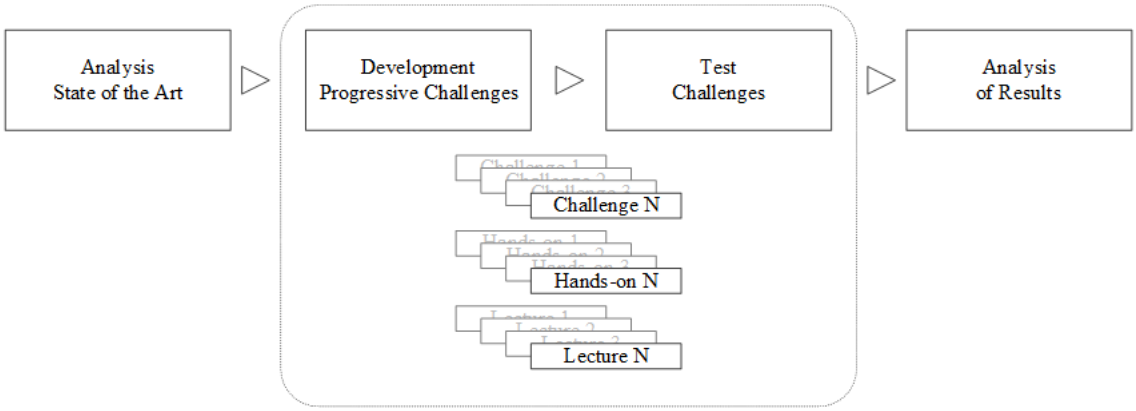

Figure 2: Research Process.

3.2. THE STUDENTS PARTICIPANTS

An inherent characteristic of the student audience is their diverse knowledge background. The selected number of students is twenty, 85% from the ICT track and 15% from general school students as guests. Also, the proportion of male students reached 85% while female students were 15%. Breaking down the figures of ICT students in levels, the students registered in 2016 were 20%, the students registered in 2017 became 20%, and the students registered in 2018 represented 45%.

3.3. TEST OF CHALLENGES

The challenges are driven by a constructivist learning method, designed as chained experiences that are gradually revealed to the students. The challenges are scaffolded by the study materials in the form of theoretical formal knowledge and hands-on lessons. Throughout the experimentation of the hands-on lessons, a student reasons about a security problem and some plausible solutions. A challenge states a concise tip and hint without instructions, keeping some level of ambiguity to induce collaborative constructions among students.

3.3.1. CHALLENGE 1

The first challenge focuses on the reconnaissance and discovery stage of the penetration testing methodology, and it is also an adaptation based on A. Kauramäki [13]. The techniques emphasized are active reconnaissance and brute force attack. The tools applied are NMAP [20] and Metasploit framework [21]. Under a non-blind test context, the targeted system performs as a Domain Controller over Microsoft [22], and the attacker system performs as a Linux system over a Kali distribution [23] (See Chapter Infrastructure Design and Deployment for further information).

The hint states "Break into the Domain. A lazy admin loves passwords easy to remember, the most common".

The challenge is scaffolded by the following lectures and hands-on series (See Annex A: Content Scope for further information):

- Lecture 1: Knowing your attack surface: SSH protocol
- Hands-on 1 - Knowing your attack surface: SSH and VNC
- Lecture 2: Penetration Testing Methodology.
- Hands-on 2: Active Reconnaissance: NMAP.

Once the credential is revealed by the student, he or she sends the credential to the mentor's email to get one standard point. The challenge considers a bonus point; it requires to use the credential over a brief adapted routine coded in python language [24][25], the syntax to run it is as follows:

```
$ python TimeLapsev2.py <user list> <pass list>
```

The student who completes the missing parts of the code and informs the mentor by email once done gets a bonus point. The code is evaluated by the mentor over the cloud-based student's instance.

3.3.2. CHALLENGE 2

The second challenge focuses on enumeration and vulnerability analysis of the penetration testing methodology, and it is also an adaptation based on A. Kauramäki [13]. The techniques emphasized are vulnerability scanning and obfuscation. The tools applied are OpenVAS [26] and Base64 encoding [27]. Under a non-blind test context, the targeted system performs as a Domain Controller over Microsoft, and the attacker system performs as a Linux system over a Kali distribution (See Chapter Infrastructure Design and Deployment for further information).

The hint is obfuscated using Base64 encoding, as follows:

"SGkuIEp1bXAgaW50byBPcGVuVmFzLiBTY2FuIG91ciBkb21haW4gd2l0aG91dCBjcmVkZW50aWFscywgdGhlbiBhcyBhIHJvb3QvXdlcnR5MTIzLiBXaGF0IGFyZSB0aGUgaGlnaCBzZXZlcml0eSB2dWxuZXJhYmlsaXRpZXMgb3ZlciB0aGVyZT8=".

The challenge is scaffolded by the following lectures and hands-on series (See Annex A: Content Scope for further information):

- Lecture 3: Metasploit Framework.

- Hands-on 3: Metasploit Recon and Brute Force.
- Lecture 4: Vulnerability Analysis.
- Hands-on 4: Vulnerability Analysis: OpenVAS.

Once the hint is decoded by the student, he or she performs a vulnerability analysis applying differentiated credentials, and the higher severity vulnerabilities are notified to the mentor's email to get one standard point. The challenge considers a bonus point; it requires to reverse the programmed logic used to encode the hint to decode it. The routine is coded in python language, the syntax to run it is as follows:

```
$ python mybase64.py
```

The student builds a base64 decoding routine in python and informs the mentor by email once done to get a bonus point. The code is evaluated by the mentor over the cloud-based student's instance.

3.3.3. CHALLENGE 3

The third challenge focuses on the execution and appropriation stage of the penetration testing methodology. The techniques emphasized are Server Message Block (SMB) [22] exploitation and reverse shells [28]. The tools applied are Metasploit exploits and payloads. Under a non-blind test context, the targeted system performs as a Domain Controller over Microsoft, and the attacker system performs as a Linux system over a Kali distribution (See Chapter Infrastructure Design and Deployment for further information).

The hint states "Time for a post recon. Seize your new reverse sessions to dump the Domain users hash values".

The challenge is scaffolded by the following lectures and hands-on series (See Annex A: Content Scope for further information):

- Lecture 3: Metasploit Framework.
- Hands-on 3: Metasploit Recon and Brute Force.
- Lecture 5: Attack Methodology and System Appropriation
- Hands-on 5: Metasploit Payloads.

Once the hash values are revealed by the student, he or she sends the information to the mentor's email to get one standard point. The challenge considers a bonus point; it requires to push a brief adapted routine coded in python language [29][30] into the Domain, the syntax to run it is as follows:

```
PS> python c:\yourCookie.py
```

Besides completing the missing parts of the code, the student demonstrates a remote interaction by applying Netcat, a networking utility integrated in NMAP [20]. The following instruction is performed in the attacker machine:

```
$ nc -lnvp 5555
```

The student proves that it is possible to acquire a list of folders from the remote connection. The students inform their findings by email to get a bonus point. The code is evaluated by the mentor over the cloud-based student's instance.

3.3.4. CHALLENGE 4

The fourth challenge focuses on the execution stage of the penetration testing methodology. The techniques emphasized are eavesdropping and man in the middle attack through ARP poisoning technique [31][52]. The tools applied are Tshark [32] and arpspoof. Under a non-blind test context, a transmission from a Centos system against an Ubuntu system is registered in a dump file named 'ch4.pcap' located in the attacker machine (See Chapter Infrastructure Design and Deployment for further information).

The hint states "Sniff the traces left by Centos. The credential 'hwt' was disclosed in clear. Filter as you need. Search all the details and reveal it".

The challenge is scaffolded by the following lectures and hands-on series (See Annex A: Content Scope for further information):

- Lecture 6: OSI Model and Man in the Middle Attack
- Hands-on 6: TShark and ARP Poisoning.

Once the credential is revealed by the student, he or she sends the chosen filter to the mentor's email to get one standard point. The challenge considers a bonus point; it requires to complete the missing parts of an adapted routine coded in python language [33] that integrates Scapy libraries to handcraft an ARP frame. The syntax to run it is as follows:

```
$ sudo python myarpv2.py
```

The student performs the attack against the Ubuntu system, and then informs the mentor by email once it is done to get a bonus point. The code is evaluated by the mentor over the cloud-based student's instance.

3.3.5. CHALLENGE 5

The fifth challenge focuses on the execution stage of the penetration testing methodology. The techniques emphasized are eavesdropping and HTTP traffic interception. The tools applied are the python SimpleHTTPServer module [34] and OpenSSL [35]. Under a non-blind test context, the HTTP requests transmitted from a browser Firefox are intercepted through a minimalistic HTTP Server rendering a local web page in the attacker machine (See Chapter Infrastructure Design and Deployment for further information).

The hint states "Let's complete your own simple web server. Surf it and prove that you always see what you get there".

The challenge is scaffolded by the following lectures and hands-on series (See Annex A: Content Scope for further information):

- Lecture 7: Protocols SSL/TLS.
- Hands-on 7: SSL OpenSSL and SSLStrip attack.

- Lecture 8: OWASP 10 Overview.
- Hands-on 8: Webscarab and Burpsuite.

First, the student is required to complete the missing parts of the file 'myHttpv2.py' coded in python language [34]. Then, he or she demonstrates whether the interception is feasible. To obtain one standard point, the student must send an email for evaluation once gathering a finding by following the syntax:

```
$ python3 myHttpv2.py
```

The challenge considers a bonus point; it requires to complete the missing parts of a second file called 'myHttpsv2.py' coded in python language [34], the syntax to run it is as follows:

```
$ python3 myHttpsv2.py
```

Besides the full code, the second part of the challenge requires a self-signed certificate to enable a HTTPS communication. Once completed, the students inform the mentor by email about the results to get a bonus point. The code is evaluated by the mentor over the cloud-based student's instance.

3.3.6. CHALLENGE 6

The sixth challenge focuses on the execution stage of the penetration testing methodology. The technique utilized is command injection attack against a PHP web application [36]. The tools applied are the OWASP Webscarab [37] and Burpsuite [38]. Under a non-blind test context, a parameter is chosen that improperly validates the user inputs being exposed to multiple kinds of injections. The student performs the injection from the attacker system against a PHP web application supported by a Debian system (See Chapter Infrastructure Design and Deployment for further information).

The hint states "Strict validations are not in fashion. Prove that even the database credentials can be exposed to command injection".

The challenge is scaffolded by the following lectures and hands-on series (See Annex A: Content Scope for further information):

- Lecture 10: HTTP/HTML and Injections.
- Hands-on 10: Redirect and Command Injections.

Once the database credential is revealed by the student, he or she sends the payload used to perform the attack to the mentor's email to get one standard point. The payload is verified by the mentor over the cloud-based student's instance.

3.3.7. CHALLENGE 7

The seventh challenge focuses on the execution stage of the penetration testing methodology. The technique utilized is an injection attack against a PHP web application [36]. The tool applied is Burpsuite [38]. Under a non-blind test context, a specific PHP source code is chosen where multiple parameters are used to add a new user to a MySQL database. The student analyzes the code and identifies potential parameters exposed to injections. A payload is proposed by the student to prove the weakness in the parameter

who performs the attack from the Kali system against the PHP web application supported by a Debian system (See Chapter Infrastructure Design and Deployment for further information).

The hint states “Combine experiences and attempt injections over weak parameters. disprove that the page ‘Adduser.php’ is immune to injections”.

The challenge is scaffolded by the following lectures and hands-on series (See Annex for content scope):

- Lecture 11: HTML Elements and Injections.
- Hands-on 11: OS Command Injections.

Once the injection is proven by the student, he or she sends the payload used to perform the attack to the mentor’s email to get one standard point. The payload is verified by the mentor over the cloud-based student’s instance.

3.3.8. CHALLENGE 8

The eighth challenge focuses on the execution stage of the penetration testing methodology. The technique utilized is SQL injection attack [39] against a PHP web application. The tools applied are Burpsuite and W3af scanner [40]. Under a non-blind test context, a specific PHP source code is analyzed and adapted to alter the original behavior of the website. The student considers a condition prone to injection and demonstrates the case of exploitation. The attack is performed from the Kali system against the PHP web application supported by a Debian system (See Chapter Infrastructure Design and Deployment for further information).

The hint states “Concatenate and conquer! Demonstrate that an injection is also effective to change a password. Remember, a delimiter can make a difference”.

The challenge is scaffolded by the following lectures and hands-on series (See Annex for content scope):

- Lecture 12: SQL and Injections.
- Hands-on 12: SQL Injections and W3af.

A payload is used by the student to prove the artificial weakness. The student sends the payload used to perform the attack to the mentor’s email to get one standard point. The payload is verified by the mentor over the cloud-based student’s instance.

3.3.9. CHALLENGE 9

The ninth challenge also tackles the execution stage of the penetration testing methodology. The technique utilized is Cross Site Scripting (XSS) attacks [6] against a PHP web application. The tool applied is OWASP ZAP and its fuzzer function [41]. Under a non-blind test context, a specific parameter is chosen that seems to be prone to injections. The student analyzes the code and evaluates different variants of XSS attacks to prepare the attack. An attack strategy and payload are proposed by the student to demonstrate the flaw. The attack is performed from the Kali system against the PHP web

application supported by a Debian system (See Chapter Infrastructure Design and Deployment for further information).

The hint states “Surf through the code and reveal the flaw. Prove that the ‘coursename’ parameter is always exposed to a persistent injection”.

The challenge is scaffolded by the following lectures and hands-on series (See Annex for content scope):

- Lecture 13: OWASP Cross Site Scripting.
- Hands-on 13: XSS and OWASP ZAP I.
- Lecture 14: OWASP XSS and CORS.
- Hands-on 14: XSS and OWASP ZAP II.

Once the XSS injection is proven by the student, he or she sends the payload used to perform the attack to the mentor’s email to get one standard point. The payload is verified by the mentor over the cloud-based student’s instance.

3.3.10. CHALLENGE 10

The last challenge focuses on the whole penetration testing methodology where the students combine multiple techniques. Nonetheless, a starting point technique is suggested based on Directory Listing and Path Traversal. The tools applied ranged from NMAP to W3af scanner. Initially, the scenario is a black box and the students only know the URL of a WebSite replicated from a productive environment to be targeted (See Chapter Infrastructure Design and Deployment for further information).

A first hint states “Someone forgot the value of the information. Surf through our replicated WebSite and traverse it”.

A second hint states “Come back to the replicated WebSite! You might have already found clues. now it is time to defeat its controls. The last challenge is to hack it”.

The first stage of the challenge is scaffolded by the following lecture and hands-on (See Annex for content scope):

- Lecture 9: HTTP Web Server Hardening.
- Hands-on 9: Directory Listing and Path Traversal.

The last stage of the challenge is scaffolded by the following lecture and hands-on:

- Lecture 15: Reporting Findings.
- Hands-on 15: Protecting your Report: OpenSSL.
- Lecture 16: Software Development Life Cycle (SDLC).
- Hands-on 16: Python Unittest and Selenium.

The vulnerability analysis and execution stages are not directly scaffolded to enforce problem solving and critical thinking skills. Once completed, the students write their own reports of findings to get one point per each proven finding. The outcome is evaluated by the mentor.

3.4. ANALYSIS OF RESULTS

Throughout the evaluation of the challenges, a student receives feedback and corresponding points earned. All the student's findings are archived to track the timestamp included in their email headers that is used to analyze the results. By adopting a format in the form YYMMDD, according to the RFC-3339 [42], it is possible to calculate the time period between the moment when the challenge is announced and when the student's finding arrives. Such variation is measured for each student and challenge, allowing the estimation of an aggregated ranking based on points and time. The time is also used as a tiebreak. Consequently, a global top ten and top five rankings are measured and analyzed in the corresponding chapter.

3.5. DISCUSSION

From a technical viewpoint, progressive learning involves highly dynamic experiences, and the nature of penetration testing involves numerous tools that are able to exhaust significant computational resources. Furthermore, the environment must be realistic to motivate the students in developing multiple skills. Also, even when a virtualized infrastructure is deployed, the scalability and adaptability of the environment are still critical factors to promote successful progressive learning. It is acknowledged that by building systems over a virtualized platform costs are saved related to hardware, cooling and electricity [43]. However, such virtualization is still limited to the hardware of the host system, which is likely not sized to have frequent peaks of computer power consumption. Then, a cloud computing environment becomes a highly scalable option providing interfaces to build repeatable and dynamic settings, which is implemented in this work.

4. INFRASTRUCTURE DESIGN AND DEPLOYMENT

The selected infrastructure is based on a public cloud computing service provided by Microsoft Azure [44]. A private virtualization on the premises is discarded during this first experience owing to the restricted information relative to the assets and controls in place. Indeed, the evaluation of security controls on the premises is out of the scope of this project. Nonetheless, the following requirements are defined here as recommendations to reduce the risk of an eventual misuse of the technology and unintentional information disclosure. Furthermore, as the initial number of students is unknown, the size of the virtual systems was estimated per single student rather than multiple students accessing a shared environment at the time. The case of simultaneous sessions was studied by A. Kauramäki [13], where students performed penetration testing techniques concurrently and experimented reiterative interruptions over the planned infrastructure. As a result, the capacity and flexibility of the infrastructure became a critical success factor to run all the challenges.

4.1. REQUIREMENTS FOR MULTIPLE INSTANCES

The attack surface as a whole is an environment implemented over the cloud, and it consists of two elements: the multiple instances of virtual systems, and a replica of a productive system which is one instance. The number of multiple instances is equivalent to the number of students or participants, so it is one instance per student. One student instance embraces five virtual systems or testing targets where only one of them is published on the Internet to receive Secure Shell (SSH) [45][46] requests from remote students. The replica instance is a web application reachable from any student instance to be tested as well.

The requirements are organized in three categories: Hardware and Networking, Software and Applications, and Security.

The Hardware and Networking category is considered as an infrastructure and a service growing vertically and horizontally for learning purposes. This category derives from the following three general requirements (See Annex B: Requirements for Multiple Instances for more information):

- A virtual network must allow traffic between multiple segments.
- The five systems inside one student instance can exchange data from different subnets.
- There is a single public IP allowing inbound SSH/TCP requests, and it has limited access out to the Internet.

The Software and Applications category is considered as multiple services providing distinct levels of robustness. This category derives from the following two general requirements (See Annex B: Requirements for Multiple Instances for more information):

- The attacker system performs as a Virtual Network Computing (VNC) [47] server allowing graphical desktop sessions through a secure tunnel.

- The operating systems running inside the student instance have specific versions and patches for learning purposes avoiding any hardening criteria.

The Security category is considered as a set of countermeasures prioritizing access control and availability. This category derives from the following three general requirements (See Annex B: Requirements for Multiple Instances for more information):

- The credentials are differentiated from high and low privileges for mentor and student correspondingly.
- The passwords for students are personal and confidential.
- The network traffic is controlled by a network access control embedded on Microsoft Azure cloud.

4.1.1. PHYSICAL TOPOLOGY OF THE ATTACK SURFACE

The following topology illustrates the elements of the whole attack surface.

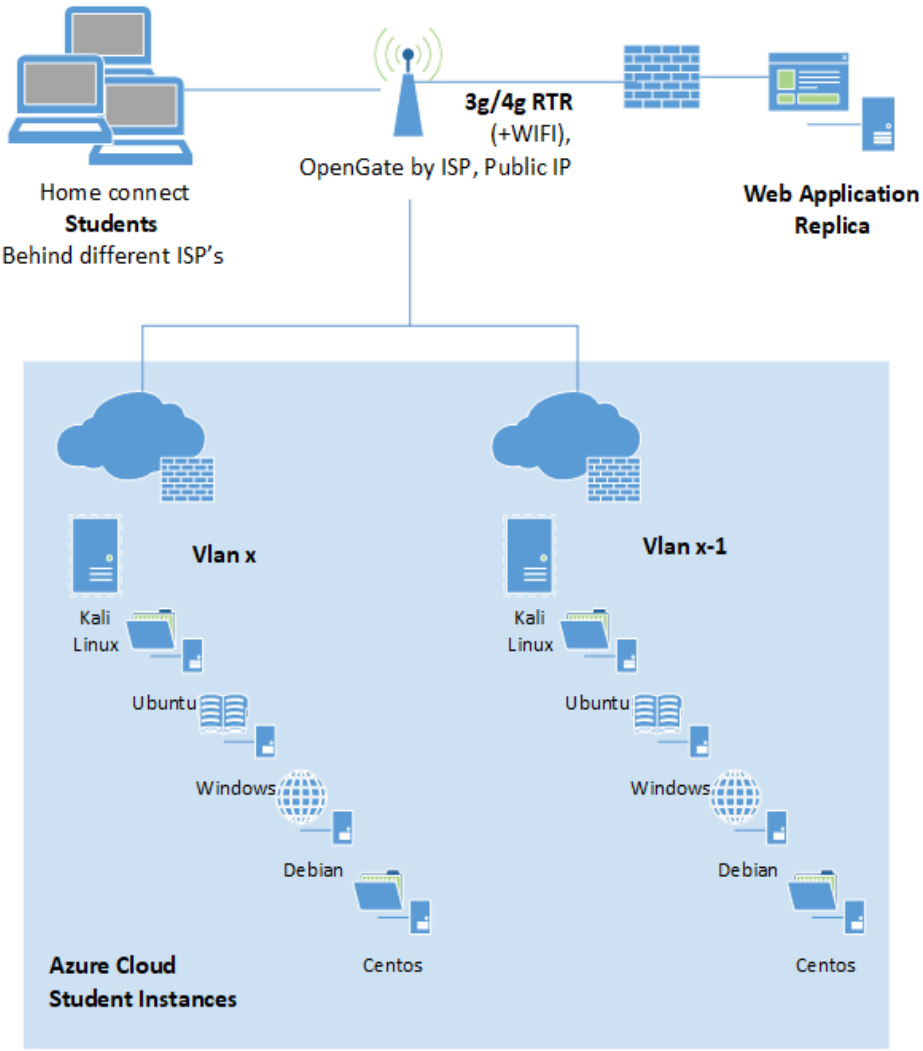

Figure 3: Physical Topology.

4.1.2. *LOGICAL TOPOLOGY OF ONE STUDENT INSTANCE*

The following logical topology illustrates the elements of an individual student instance.

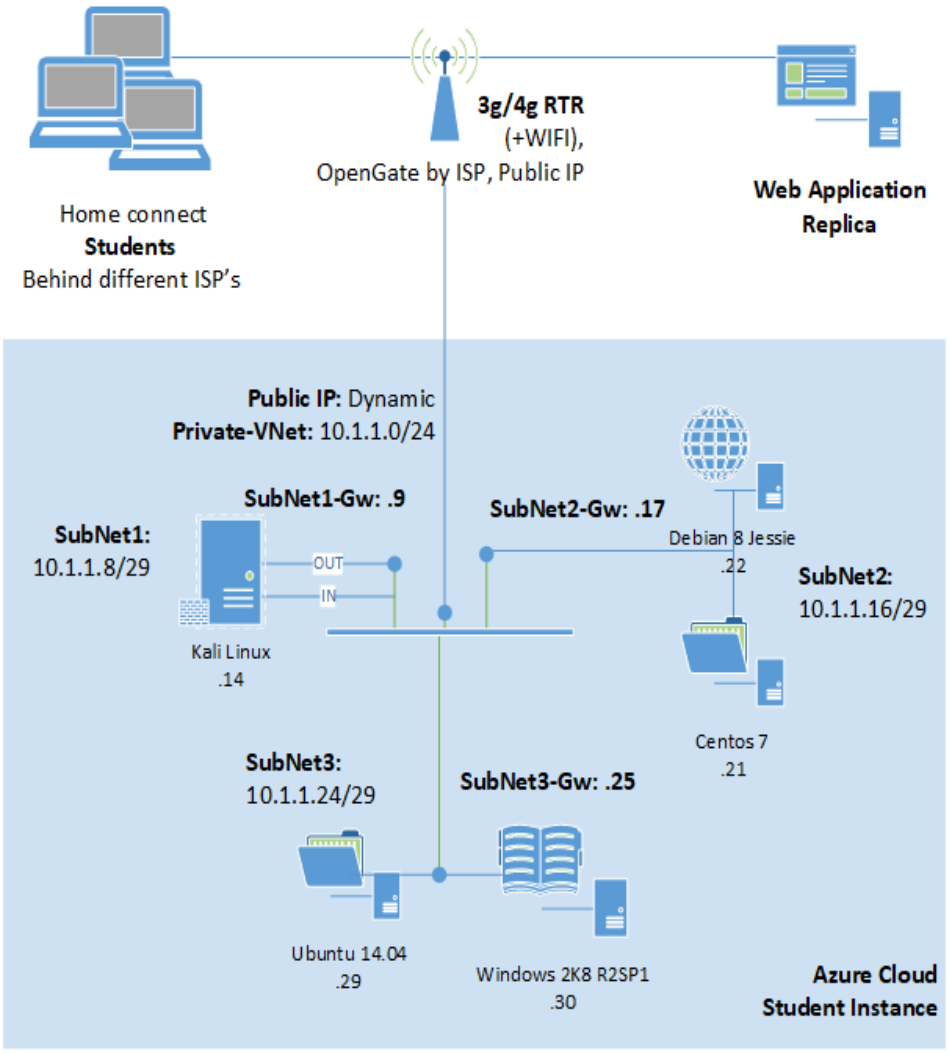

Figure 4: Logical Topology.

4.2. REQUIREMENTS FOR THE WEB APPLICATION REPLICA

The requirements for the Web Application Replica are also organized in three categories: Hardware and Networking, Software and Applications, and Security, as multiple instances.

The Hardware and Networking category is considered as an infrastructure and a service in a cloud isolated from production. This category derives from the following requirement:

- The website replicates productive functionalities in a parallel and isolated environment for testing. The implementation is published on the Internet without communication or any data exchanged from or to productive systems. The Replica WebSite must be reachable from the student instances using a dynamic public IP

and a fixed DNS named differently from the productive one. The website has restricted access for remote administration.

The Software and Applications category is considered as a layered standalone system composed of frontend, backend and database. This category derives from the following requirement:

- The system must represent the relevant functionalities of production. The functionalities are replicated from a productive system keeping the relevant functions only. The operating system is Ubuntu v16.04.5LTS. The frontend is designed in an Angular JS framework. The backend is designed with a Laravel PHP framework. The Database is PostgresSQL 9.4. The webserver is Apache 2.4.7 and PHPv5.5.9 Additional sensitive elements are disabled.

The Security category is considered as a set of provisions reducing the risk of productive data leakage. This category derives from the following three requirements:

- All user accounts must be different from the productive ones. There is not an exact match between the accounts of the productive system and the replicated one.
- The replicated data must be sanitized avoiding any data leakage. The database must contain dummy data as in a testing environment.
- The platform considers limited artificial vulnerabilities. At least one Apache directive should be lax enough to allow the traversing of a directory where a clue for a dummy account is. The RESTful backend has enabled the debugging parameter to snoop unhandled errors in the HTML format. Only successful authentication return/receive a valid token to gain access to protected resources. The account's lockout policy is set for 5 attempts and a 1-minute lockout.

4.3. HANDS-ON AND CHALLENGE PREPAREDNESS PROCESS

The preparedness of hands-on and challenges is a process that gradually enables the tools and components required. In other words, the student has limited clues until the end of the lesson about the hands-on and challenges corresponding to that day.

The process starts by narrowing the scope for the specific hands-on and challenge. The next step is the building of the scripts that are categorized as shell scripts and powershell scripts. Then, the scripts components are tested in one attack surface, and once the result is validated, they are finally deployed to all the student's instances.

Through a progressive preparedness, the current state of the attack surface is altered providing an adaptable environment for the students. An instance over Azure is a group of virtual resources that can be replicated and associated: For example, a firewall is a virtual component associated to a system.

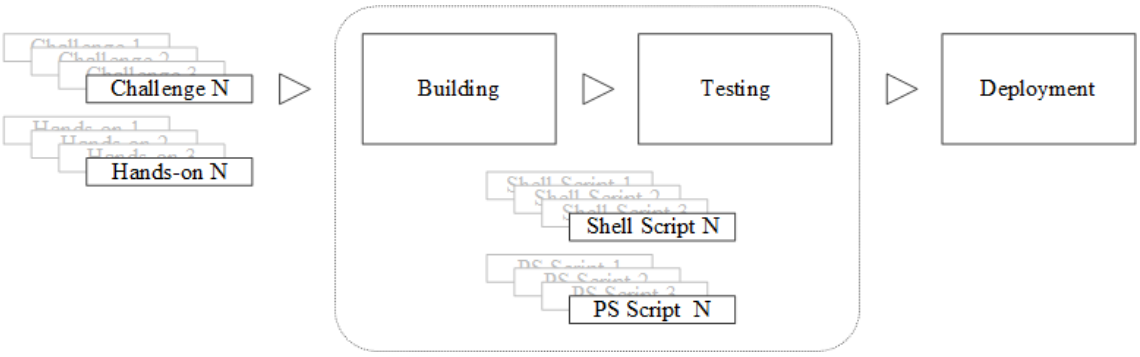

Figure 5: Preparedness Process.

Some scripts can be organized per hands-on lesson and challenge separately, although they are frequently blended for efficiency. A more accurate categorization is to group them as shell scripts running on Linux operating system, and a complementary group of powershell scripts running on Microsoft Windows system. The powershell scripts act as triggers to run the instructions in a finite loop of remote participants. A shell script file is built over a plain text encoded as Unicode Transformation Format UTF-8, the end of line sequence adopted is Line Feed (LF) character. The powershell script is built over a *.ps1 file supporting the powershell language and is encoded as UTF-16 using the variation Little Endian, and the end of line includes both characters Carriage Return and Line Feed (CRLF).

The following is a table organizing the scripts aforementioned:

#	Inputs	Scripts and Files
H1	Hands-on 1 - Knowing your attack surface: SSH and VNC	README smb.ps1 dcpromo.ps1 0hwt 1hwt 2hwt.ps1 3hwt.ps1 4hwt 5hwt 6hwt 7hwt.ps1
H2	Hands-on 2: Active Reconnaissance: NMAP.	admin.ps1 fgpp.ps1 11hwt.ps1 workspace.tar.gz
H3	Hands-on 3: Metasploit Recon and Brute Force. Challenge 1 announcement.	8hwt.ps1 9hwt 12hwt.ps1 TimeLapsev2.py
H4	Hands-on 4: Vulnerability Analysis: OpenVAS. Challenge 2 announcement.	13hwt.ps1 14hwt 15hwt mybase64.py
H5	Hands-on 5: Metasploit Payloads. Challenge 3 announcement.	17hwt.ps1 yourCookiev2.py python-2.7.15.amd64.msi

H6	Hands-on 6: TShark and ARP Poisoning. Challenge 4 announcement.	16hwt.ps1 19hwt.ps1 myarpv2.py ch4.pcap
H7	Hands-on 7: SSL OpenSSL and SSLstrip attack.	10hwt 18hwt.ps1 20hwt 21hwt.ps1
H8	Hands-on 8: Webscarab and Burpsuite. Challenge 5 announcement.	25hwt mygetHttpv2.py mygetHttpsv2.py webscarab-*.jar index1.html
H9	Hands-on 9: Directory Listing and Path Traversal. Challenge 10 first announcement.	22hwt 23hwt 24hwt
H10	Hands-on 10: Redirect and Command Injections. Challenge 6 announcement.	26hwt 27hwt
H11	Hands-on 11: OS Command Injections. Challenge 7 announcement.	28hwt index2.html my*.js my*.css
H12	Hands-on 12: SQL Injections and W3af. Challenge 8 announcement. Hands-on 13: XSS and OWASP ZAP I. Challenge 9 announcement. Hands-on 14: XSS and OWASP ZAP II. Challenge 10 second announcement.	29hwt 30hwt.ps1 ViewGrades.hwt*

Figure 6: Table of Scripts.

4.4. HANDS-ON 1 - KNOWING YOUR ATTACK SURFACE: SSH AND VNC

The preparedness for the first hands-on addresses the authentication of the multiple systems using differentiated credentials, the authorization to specific applications, and firewall rules.

4.4.1. AUTHENTICATION AND AUTHORIZATION SCRIPTS

A README file is created to accomplish a subset of tasks semi-automatically. The tasks are run once per student instance by either an ICT department or by the mentor. The subset of tasks consists of two requisites described in the following paragraphs.

A sudo security policy is modified locally for/by the systems hack-kali, hack-debian, hack-centos using the following command line:

```
1 $sudo sed -i 's|mentor ALL=(ALL) ALL|mentor ALL=(ALL) NOPASSWD:ALL|' /etc/sudoers.d/waagent
```

Alternatively, the command line statements can be run over/on Azure using Run Command Script or Custom Script Extension.

This first requisite is slightly modified in the case of hack-ubuntu, as follows:

```
$sudo sed -i 's|mentor ALL=(ALL) ALL|mentor ALL=(ALL) NOPASSWD:ALL|' /etc/sudoers.d/90-cloud-init-users
```

The second requisite covers a hack-win system that is configured to perform as a domain controller under a purposely over relaxed authentication setting. The next statement enables the Server Message Block (SMB) protocol version 1 for further testing [22].

```
#smb.ps1
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters" SMB1 -Type DWORD -Value 1 -Force
```

In the next lines is prepared Active Directory service over the hack-win system while the host firewall and updates are disabled to induce the artificial insecure system.

```
#dcpromo.ps1
$mycontent = @"
[DCInstall]
ReplicaOrNewDomain=Domain
NewDomain=Forest
NewDomainDNSName=turku.fi
ForestLevel=4
DomainNetbiosName=TURKU
DomainLevel=4
InstallDNS=Yes
ConfirmGc=Yes
CreateDNSDelegation=No
DatabasePath="C:\windows\NTDS"
LogPath="C:\windows\NTDS"
SYSVOLPath="C:\windows\SYSVOL"
SafeModeAdminPassword="secret"
RebootOnCompletion=Yes
"@
New-Item -Path "C:\" -Name "unattended.txt" -ItemType "file" -Value "$mycontent"
Get-Item C:\unattended.txt |findstr "turku"
dcpromo.exe /unattend:C:\unattended.txt
```

After completion of the requisites, the whitelist of applications is prepared by means of the shell script file “0hwt”. Notice that the symbol * is representing a possible application string. For instance; nmap, msfconsole, openvas-start, tshark, arpspoof, python, java, burpsuite, wfuzz, curl, zaproxy, among others are included in the model of challenges.

```
#0hwt
#!/bin/sh
cat <<_EOT_ | sudo -s
/usr/bin/cat > /etc/sudoers.d/0hwt << EOF
#sudo configuration for hwt
%turku ALL=(ALL) NOPASSWD:/usr/bin/*
%turku ALL=(ALL) NOPASSWD:/usr/sbin/*
EOF
/usr/bin/chown root:root /etc/sudoers.d/0hwt
/usr/bin/chmod 440 /etc/sudoers.d/0hwt
_EOT_
```

Complementary, the shell script file “1hwt”is built mainly authorizing the public key authentication of the user mentor and also creating the new user hack in the surface which is authorized to run the applications granted in the script “0hwt”. It is worth noting that a different password per student user is set each time.

```
#1hwt
#!/bin/sh
export SSHPASS='secret'
export WINEXE='mentor%secret'
export HACKPASSCLEAR='private1'
cat <<_EOT_ | sudo -s
/usr/sbin/groupmod -n turku mentor
/usr/sbin/useradd -u 1001 -g 1000 -d /home/hack -m hack
/usr/bin/openssl passwd -1 "${HACKPASSCLEAR}" > /home/mentor/.bashx
/usr/sbin/usermod -s /bin/bash -p "$(cat /home/mentor/.bashx)" hack
/usr/bin/mkdir /home/mentor/.ssh
/usr/bin/chmod 700 /home/mentor/.ssh
/usr/bin/touch /home/mentor/.ssh/authorized_keys
/usr/bin/touch /home/mentor/.ssh/known_hosts
/usr/bin/chmod 644 /home/mentor/.ssh/authorized_keys
/usr/bin/chmod 644 /home/mentor/.ssh/known_hosts
/usr/bin/chown mentor:mentor -R /home/mentor/.ssh
/usr/bin/echo "ssh-rsa \
Mentor’s pub key here created by OpenSSH_for_Windows\
>> /home/mentor/.ssh/authorized_keys
/usr/bin/echo "ssh-rsa \
Mentor’s pub key here created by Putty_for_Windows\
>> /home/mentor/.ssh/authorized_keys
/usr/bin/ssh-keyscan -t rsa 10.1.1.21 >> /home/mentor/.ssh/known_hosts
/usr/bin/ssh-keyscan -t rsa 10.1.1.22 >> /home/mentor/.ssh/known_hosts
/usr/bin/ssh-keyscan -t rsa 10.1.1.29 >> /home/mentor/.ssh/known_hosts
/usr/bin/sed -i 's/HISTSIZE=1000/HISTSIZE=1/' /home/mentor/.bashrc
/usr/bin/sed -i 's/HISTFILESIZE=2000/HISTFILESIZE=0/' /home/mentor/.bashrc
/usr/bin/sed -i 's/kali/swap/' /etc/hosts
/usr/bin/sed -i 's/swap/hack-kali/' /etc/hosts
/usr/bin/apt-get install sshpass -y
/usr/bin/apt-get autoremove -y
_EOT_
# hack-centos v7.5.1804
/usr/bin/sshpass -e scp /home/mentor/.bashx mentor@10.1.1.21:/home/mentor/.bashx
/usr/bin/sshpass -e ssh mentor@10.1.1.21 'sudo /usr/sbin/groupmod -n turku mentor'
/usr/bin/sshpass -e ssh mentor@10.1.1.21 'sudo /usr/sbin/useradd -u 1001 -g 1000 -d
/home/hack -m hack'
/usr/bin/sshpass -e ssh mentor@10.1.1.21 'sudo /usr/sbin/usermod -s /bin/bash -p "$(cat
/home/mentor/.bashx)" hack'
# hack-debian v8.11 jessie
/usr/bin/sshpass -e scp /home/mentor/.bashx mentor@10.1.1.22:/home/mentor/.bashx
/usr/bin/sshpass -e ssh mentor@10.1.1.22 'sudo /usr/sbin/groupmod -n turku mentor >>
/dev/null 2>&1'
/usr/bin/sshpass -e ssh mentor@10.1.1.22 'sudo /usr/sbin/useradd -u 1001 -g 1000 -d
/home/hack -m hack >> /dev/null 2>&1'
/usr/bin/sshpass -e ssh mentor@10.1.1.22 'sudo /usr/sbin/usermod -s /bin/bash -p "$(cat
/home/mentor/.bashx)" hack >> /dev/null 2>&1'
# hack-ubuntu v14.04.5 trusty
```

```
41 /usr/bin/sshpass -e scp /home/mentor/.bashx mentor@10.1.1.29:/home/mentor/.bashx
42 /usr/bin/sshpass -e ssh mentor@10.1.1.29 'sudo /usr/sbin/groupmod -n turku mentor >>
/dev/null 2>&1'
43 /usr/bin/sshpass -e ssh mentor@10.1.1.29 'sudo /usr/sbin/useradd -u 1001 -g 1000 -d
/home/hack -m -c "" hack >> /dev/null 2>&1'
44 /usr/bin/sshpass -e ssh mentor@10.1.1.29 'sudo /usr/sbin/usermod -s /bin/bash -p "$(cat
/home/mentor/.bashx)" hack >> /dev/null 2>&1'
45 # hack-win - v2008_R2_SP1 and winexe v1.1 on Kali.
46 /usr/bin/winexe -U "$WINEXE" //10.1.1.30 'cmd.exe /c net user hack .win1234. /add 2>&1>null'
47 /usr/bin/winexe -U "$WINEXE" //10.1.1.30 'cmd.exe /c net user hack '"${HACKPASSCLEAR}"
48 #Session exit
49 exit
```

The powershell script file “2hwt” is built and run over Microsoft Visual Studio Code from a Microsoft 10 operating system having enabled OpenSSHv7.7 client and the Putty suite to proceed with the next execution.

```
1 #2hwt.ps1
2 $myArray = ("op1","op2","op3",...,"op22")
3 New-Variable -Name "OPN" -Visibility Public -Value "secret"
4 foreach ($op in $myArray) {
5 #Set
6   .\pscp.exe -pw $OPN .\0hwt mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
7   .\pscp.exe -pw $OPN .\1hwt mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
8   .\plink.exe -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com '/usr/bin/chmod
700 /home/mentor/*hwt'
9   .\plink.exe -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'/home/mentor/0hwt'
10  .\plink.exe -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com '/usr/bin/sleep
5s'
11   .\plink.exe -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'/home/mentor/1hwt'
12   #Clean
13   #.\plink.exe -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo rm -rf
/home/mentor/*hwt'
14   #Check
15   .\plink.exe -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo echo
$SSH_CLIENT' + $op
16 }
17 Remove-Variable -Name "OPN" -Force
```

This last powershell script file is called “3hwt”, and its goal is to verify if the student password were assigned properly by looping a login attempt for the pool of systems.

```
1 #3hwt.ps1
2 $i = 1
3 $myArray = @("private1","private2","private3",...,"private22")
4 New-Variable -Name "OPN" -Visibility Public -Value "foo"
5 foreach ($element in $myArray) {
6     Set-Variable -Name "OPN" -Visibility Public -Value "$element"
7    # Check
8    .\plink.exe -pw $OPN hack@hack-kali-op$i-ip.westeurope.cloudapp.azure.com "pwd"
```

```
    $i++
}
Remove-Variable -Name "OPN" -Force
```

4.4.2. FIREWALL RULES SCRIPTS

The network security groups on Azure are associated with every machine following cross requirements. So, a host firewall based on netfilter was enabled on a hack-kali system for more specific rules as well (See Annex C: Host Firewall Rules for more information). A set of rules are defined and formatted in a plain text file named "4hwt" according to the specifications of iptables.

The following shell script file "5hwt" is preloaded to control the network interfaces and to enable the forwarding of packets and the rules's persistence.

```
#5hwt
#!/bin/sh
/usr/sbin/iptables -F
/usr/sbin/iptables -t nat -F
/usr/sbin/iptables-restore < /etc/network/if-pre-up.d/4hwt
echo -n '1' > /proc/sys/net/ipv4/ip_forward
echo -n '0' > /proc/sys/net/ipv4/conf/all/accept_source_route
echo -n '0' > /proc/sys/net/ipv4/conf/all/accept_redirects
echo -n '1' > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo -n '1' > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
```

The shell script file "6hwt" aims to push the script shell files "4hwt" and "5hwt" into the appropriate paths and enforces the file access mode and ownership required to control the firewall settings.

```
#6hwt
#!/bin/sh
cat <<_EOT_ | sudo -s
/usr/bin/mv /home/mentor/4hwt /etc/network/if-pre-up.d/4hwt
/usr/bin/chown root:root /etc/network/if-pre-up.d/4hwt
/usr/bin/chmod 600 /etc/network/if-pre-up.d/4hwt
/usr/bin/chown root:root /etc/network/if-pre-up.d/5hwt
/usr/bin/chmod 755 /etc/network/if-pre-up.d/5hwt
/usr/sbin/iptables-restore < /etc/network/if-pre-up.d/4hwt
_EOT_
```

The powershell script file "7hwt" uploads a set of shell script files to enable the firewall, and it then triggers the execution, as follows:

```
#7hwt.ps1
$myArray = ("op1","op2","op3",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
  .\pscp.exe -i id_rsa.ppk -pw $OPN .\4hwt mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\pscp.exe -i id_rsa.ppk -pw $OPN .\5hwt mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
```

```
7   .\pscp.exe -i id_rsa.ppk -pw $OPN .\6hwt mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
8   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'chmod 700 /home/mentor/6hwt'
9   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'/home/mentor/6hwt'
10   #Clean
11   #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/*hwt'
12   #Check
13   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'ping 8.8.8.8 -c 2'
14 }
15 Remove-Variable -Name "OPN" -Force
```

4.5. HANDS-ON 2: ACTIVE RECONNAISSANCE, NMAP

NMAP stands for Network Mapper and is an open source port scanner tool for network exploration and security auditing [20]. The NMAP Scripting Engine (NSE) allows to write simple scripts in Lua programming language to automate networking tasks [48].

The preparedness for the second hands-on involves a set of scripts to override the password security policies of the hack-win domain controller, and lastly it organizes a subset of NSE files into the workspace folder located in hack-kali system.

4.5.1. PASSWORD SECURITY POLICY

The following powershell script file creates a new organization unit based on Lightweight Directory Access Protocol (LDAP) [22] integrated in the domain controller hack-win. An existing admin user is associated to the new organization unit as well.

```
1 #admin.ps1
2 Import-Module activedirectory
3 New-ADOrganizationalUnit -Name "admins" -Path "DC=turku,DC=fi" -
ProtectedFromAccidentalDeletion $False
4 New-ADUser -Name "admin" -PasswordNotRequired $True -Path "OU=admins,DC=turku,DC=fi"
```

Besides the admin credential, a complementary powershell script file needs to override the current password policy by means of a Fine-Grained Password Policy (FGPP) based on LDAP entities. The goal is to purposely lower the security level of the authentication for the upcoming challenge.

```
1 #fgpp.ps1
2 Import-Module activedirectory
3 New-ADFineGrainedPasswordPolicy -Name "AdminPolicy" -ComplexityEnabled $false -
MinPasswordLength 4 -Precedence 100
4 Set-ADFineGrainedPasswordPolicy AdminPolicy -ReversibleEncryptionEnabled $false
5 Set-ADFineGrainedPasswordPolicy AdminPolicy -MinPasswordAge "1.00:00:00" -MaxPasswordAge
"365.00:00:00"
6 Add-ADFineGrainedPasswordPolicySubject AdminPolicy -Subjects admin
7 Add-ADFineGrainedPasswordPolicySubject AdminPolicy -Subjects Guest
gpupdate /Force
```

```
Set-ADAccountPassword -Identity "CN=Guest,CN=Users,DC=turku,DC=fi" -Reset -NewPassword
(ConvertTo-SecureString -AsPlainText "welcome" -Force)
Set-ADAccountPassword -Identity "CN=admin,OU=admins,DC=turku,DC=fi" -Reset -NewPassword
(ConvertTo-SecureString -AsPlainText "football" -Force)
Set-ADUser -Identity 'CN=admin,OU=admins,DC=turku,DC=fi' -Enabled $True -
PasswordNeverExpires $True
Add-ADGroupMember -Identity "CN=Administrators,CN=Builtin,DC=turku,DC=fi" -Members
"CN=admin,OU=admins,DC=turku,DC=fi"
```

4.5.2. NMAP NSE FILES

The following powershell script file "11hwt.ps1" creates the folder workspace where a group of Nmap Scripting Engine (NSE) files will be uploaded such as whois-domain.nse, smb-os-discovery.nse, http-apache-server-status, among others [48]. The NSE files are archived in the file workspace.tar.gz.

```
#11hwt.ps1
$myArray = ("op1","op2",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
.\pscp.exe -i id_rsa.ppk -pw $OPN .\workspace.tar.gz mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo mkdir /home/hack/workspace'
.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo cp /home/mentor/workspace.tar.gz /home/hack'
.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo tar -xzvf /home/hack/workspace.tar.gz -C /home/hack/'
.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chown hack:turku -R /home/hack/workspace'
.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/hack/workspace.tar.gz'
.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/workspace.tar.gz'
  #Check
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo ls -la /home/hack/workspace'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo cat /etc/sudoers.d/0hwt |grep nmap'
}
Remove-Variable -Name "OPN" -Force
```

4.6. HANDS-ON 3: METASPLOIT RECON AND BRUTE FORCE

The preparedness for the third hands-on enables Metasploit framework and includes a file programmed in python language as part of the challenge 1. Metasploit was developed by Moore H.D. in 2003 with the aim to provide an exploitation framework for testing and development [49]. In 2009 Metasploit was acquired by Rapid7 [21] which still provides the framework as an open source yet adding commercial editions. The project originally was written in Perl scripting language, and then it was rebuilt into Ruby. The framework is based on six modules; auxiliary, exploit, payload, encoder, nop and post.

4.6.1. METASPLOIT INITIALIZATION

The shell script file “8hwt” aims to initialize the Metasploit’s database.

```
#8hwt
#!/bin/sh
cat <<_EOT_ | sudo -s
/usr/bin/msfdb init
/usr/bin/systemctl enable postgresql
/usr/bin/systemctl start postgresql
_EOT_
```

The powershell script file “9hwt” uploads and runs the shell script file “8hwt”and checks if the binary msfconsole has been authorized for the Turku student group.

```
#9hwt.ps1
$myArray = ("op1","op2","op3",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
  .\pscp.exe -pw $OPN .\9hwt
mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com:/home/mentor
  .\plink.exe -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'chmod 700
/home/mentor/9hwt'
  .\plink.exe -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'/home/mentor/9hwt'
  #Clean
  #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/*hwt'
  #Check
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/systemctl status postgresql'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo cat /etc/sudoers.d/0hwt |grep openvas'
}
Remove-Variable -Name "OPN" -Force
```

4.6.2. CHALLENGE 1 PREPAREDNESS

Multiple tools are available to perform a Brute Force attack, and the challenge approaches the analysis of a code developed in python, integrating a module from Metasploit. The adaptation is based on the project TimeLapse, a SMB timed bruteforcer using Metasploit smb_login [25], which is published on Github. Consequently, the student needs to do some research to complete the challenge while analyzing the timeout constraint.

```
        #Snippet code - Challenge 1
        if counttries == (tries + 1):
            print "Hit Maximum Attempts, sleeping for " + str(reset+1) + " minutes."
            time.sleep((reset+1)*60)
            counttries = tries
            os.system('rm pass_file.tmp')
        with open("pass_file.tmp", "a") as tmpfile:
            tmpfile.writelines(line)
        tmpfile.close()
        if counttries > 0:
          # <<<<< COMPLETE HERE THE MISSING PARTS >>>>>
```

```
37          #
38           os.system('------------ "---------------------------------; \
             --------- '+target+'; ------------ '+domain+'; \
             ------------ '+user_file+'; \
             set PASS_FILE pass_file.tmp;spool TimeLapse.log; run;   exit"')
39       counttries = counttries - 1
```

The following powershell script file “12hwt.ps1” uploads the file TimeLapsev2.py and ensures the rights and correct path for the file.

```
1 #12hwt.ps1
2 $myArray = ("op1","op2","op3",...,"op22")
3 New-Variable -Name "OPN" -Visibility Public -Value "secret"
4 foreach ($op in $myArray) {
5   .\pscp.exe -i id_rsa.ppk -pw $OPN .\TimeLapsev2.py   mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
6   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo mv /home/mentor/TimeLapsev2.py /home/hack/workspace'
7   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chown hack:turku /home/hack/workspace/TimeLapsev2.py'
8   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chmod 755 /home/hack/workspace/TimeLapsev2.py'
9   #Clean
10   #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/*hwt'
11   #Check
12   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo ls -la /home/hack/workspace |grep *.py'
13 }
14 Remove-Variable -Name "OPN" -Force
```

4.7. HANDS-ON 4: VULNERABILITY ANALYSIS: OPENVAS

The preparedness for the fourth hands-on embraces the implementation of the vulnerability scanner Greenbone OpenVAS [26]. The routine assures the latest Greenbone feeds, such as the Network Vulnerability Tests (NVT), where the acceptance of RSYNC outbound traffic is required. Also, a complementary set of instructions in powershell is given to reinforce the credentials over hack-win to leverage the scope of the hands-on. Lastly, a file programmed in python language as part of the challenge 2 is included.

The project OpenVAS stands for Open Vulnerability Assessment System, and it started as a code branch of the GNessus project in 2005. It was originally developed by communities of pen-testers. Since 2008, it has been maintained by Greenbone Network GmbH. OpenVAS is a framework for scanning and managing of vulnerabilities [50][51], and it remains as an open source project.

4.7.1. *OPENVAS SETTINGS*

The shell script file “15hwt” aims to implement the OpenVAS framework for scanning and managing vulnerabilities.

```
#15hwt
#!/bin/sh
cat <<_EOT_ | sudo -s
/usr/sbin/openvasmd --create-user=hack
/usr/sbin/openvasmd --user=hack --new-password=hwt2019
/usr/bin/openvas-setup
/usr/bin/setfacl -m g:turku:rw /var/lib/openvas/mgr/tasks.db
/usr/bin/setfacl -m g:turku:rw /var/run/openvassd.pid
/usr/bin/setfacl -m g:turku:rw /var/run/openvasmd.pid
/usr/bin/setfacl -m g:turku:rw /var/run/openvassd.sock
/usr/sbin/openvasmd --user=admin --new-password=hwt2019
/usr/sbin/openvasmd --rebuild
/usr/sbin/greenbone-nvt-sync --wget
/usr/sbin/greenbone-certdata-sync --wget
/usr/sbin/greenbone-scapdata-sync --wget
/usr/sbin/shutdown -r now
_EOT_
```

The next set of instructions in powershell language prepares the credentials on hack-win system, and they can be run on Azure using Run Command Script or Custom Script Extension, so it is a semi-automatic requisite [44]. The Fine-Grained Password Policy (FGPP) [22] based on LDAP entities is used again to leverage the hands-on.

```
#13hwt.ps1
Import-Module activedirectory
New-ADUser -Name "root" -PasswordNotRequired $True -Path "OU=admins,DC=turku,DC=fi"
Add-ADFineGrainedPasswordPolicySubject AdminPolicy -Subjects root
Add-ADGroupMember -Identity "CN=Administrators,CN=Builtin,DC=turku,DC=fi" -Members
"CN=root,OU=admins,DC=turku,DC=fi"
Set-ADUser -Identity 'CN=root,OU=admins,DC=turku,DC=fi' -Enabled $True -PasswordNeverExpires
$True
Set-ADAccountPassword -Identity "CN=root,OU=admins,DC=turku,DC=fi" -Reset -NewPassword
(ConvertTo-SecureString -AsPlainText "qwerty123" -Force)
Set-ADAccountPassword -Identity "CN=Guest,CN=Users,DC=turku,DC=fi" -Reset -NewPassword
(ConvertTo-SecureString -AsPlainText "welcome" -Force)
Set-ADAccountPassword -Identity "CN=admin,OU=admins,DC=turku,DC=fi" -Reset -NewPassword
(ConvertTo-SecureString -AsPlainText "football" -Force)
```

4.7.2. *CHALLENGE 2 PREPAREDNESS*

There are multiple ways to encode a string into Base64 [27], even by using an existing online encoder/decoder until coding a creativity routine. The goal is to get familiarized with Base64 by programming a decoder considering the following encoder developed in python as starting point.

```
#Challenge 2
#!/usr/bin/python
import base64
m = "Tervetuloa"
f = m.encode('utf-8')
```

```
c = base64.b64encode(f)
print(c)
```

The following powershell script file “14hwt.ps1” uploads the file mybase64.py and ensures the rights and correct path for the file. Additionally, the shell script file “15hwt” is uploaded to implement OpenVAS, as follows:

```
#14hwt.ps1
$myArray = ("op1","op2","op3",...,"op21")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
    .\pscp.exe -i id_rsa.ppk -pw $OPN .\15hwt  mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
    .\pscp.exe -i id_rsa.ppk -pw $OPN .\mybase64.py mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo mv /home/mentor/mybase64.py /home/hack/workspace'
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chown hack:turku /home/hack/workspace/mybase64.py'
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chmod 755 /home/hack/workspace/mybase64.py'
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chmod 700 /home/mentor/15hwt'
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /home/mentor/15hwt'
    #Clean
    #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/*hwt'
    #Check
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo ls -la /home/hack/workspace/mybase*'
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo cat /etc/sudoers.d/0hwt |grep arpspoof'
}
Remove-Variable -Name "OPN" -Force
```

4.8. HANDS-ON 5: METASPLOIT PAYLOADS

The fifth hands-on extends the practice over shellcodes by applying Metasploit payloads classified in three categories: single, stagers and stages. By understanding shellcodes, it is possible to generate a reverse shell leveraged by a Metasploit command execution, such as the msfvenom tool [21]. The student reflects about how a targeted system triggers the outcome and forwards the shell to an attacker [28]. As the Metasploit framework is already deployed, the preparedness for this case is focused on the Challenge 3. The goal is to prove the consequences of a remote shell by proposing a python code to be run in the hack-win system called yourCookiev2.py.

The package python-2.7.15.amd64.msi is preinstalled in the hack-win system as a requisite. As usual, the following deployment can be run on Azure using Run Command Script or Custom Script Extension:

```
dism /online /Disable-Feature /FeatureName:Internet-Explorer-Optional-amd64
```

```
msiexec.exe /i c:\python-2.7.15.amd64.msi /quiet /norestart TARGETDIR=C:\python27
ADDLOCAL=ALL ALLUSER=1
sleep 2'
move c:\python-2.7.15.amd64.msi C:\python27
```

4.8.1. CHALLENGE 3 PREPAREDNESS

A shellcode depends on the operating system and processor to be assembled at the machine level and then compiled and encoded. Shellcoding follows a complex procedure that is simplified when it is leveraged by Metasploit extensions. Nevertheless, the aim is slightly different as it encourages the deep understanding of how dangerous the careless execution of unauthorized software is. Thus, a code written in python language called yourCookiev2.py is provided to prove that a standard output can be forwarded through a new connection. The adaptation is based on a couple of Github projects related to payloads and reverse shells [29][30]. Consequently, the student needs to do some research to complete the challenge and trigger a command from the attacking side to confirm the data breach.

```
#Snippet code - Challenge 3
#<<< Complete the Missing Parts denoted by dashes>>>
---- - ----
---- - ----
client.-------((----,----))
#<<< End of the Missing Parts Zone >>>
while True:
  data = client.recv(1024)
  if len(data) > 0:
   cmd = subprocess.Popen(data[:].decode("utf-8"), \
   shell=True, stdout=subprocess.PIPE, \
   stderr=subprocess.STDOUT, stdin=subprocess.PIPE)
   output_bytes = cmd.stdout.read()
   output_str = str(output_bytes).encode("utf-8")
   client.send(str.encode(output_str + '.'))
   print(output_str)
```

The following powershell script file “17hwt.ps1” uploads the file yourCookiev2.py and ensures the rights and correct path for the file. Additionally, the python MSI package is uploaded as follows:

```
#17hwt.ps1
$myArray = ("op1","op2",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
  .\pscp.exe -i id_rsa.ppk -pw $OPN .\yourCookiev2.py mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\pscp.exe -i id_rsa.ppk -pw $OPN .\python-2.7.15.amd64.msi mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo mv /home/mentor/yourCookiev2.py /home/hack/workspace'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chown hack:turku /home/hack/workspace/yourCookiev2.py'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chmod 755 /home/hack/workspace/yourCookiev2.py'
  #Clean
```

```
   #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/*hwt'
   #Check
   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo ls -la /home/hack/workspace/yourCookiev2.py'
}
Remove-Variable -Name "OPN" -Force
```

4.9. HANDS-ON 6: TSHARK AND ARP POISONING

The preparedness for the third hands-on enables Wireshark, an open source network protocol analyzer [32]. The CLI version of Wireshark is Tshark, which is used in this hands-on and challenge 4. A sample of traffic between Centos and Ubuntu systems is captured in the file ch4.pcap that conveys credentials in clear text. An additional interface is set into the hack-kali system to reach a middle position and execute an Address Resolution Protocol (ARP) poisoning attack [31][52].

4.9.1. FILTERING OF CAPTURED TRAFFIC

The Tshark network analyzer uses the same syntax for capturing filters such as tcpdump and any other sniffing program that uses libpcap/Winpcap libraries to capture traffic from a given interface. Therefore, once the traffic is captured from a live network, it is possible to read the packets saved in a file. The student needs to differentiate capture filters from display filters and propose his or her own filter to narrow the search for the credential.

The next set of instructions in powershell language preloads the dump file of the traffic in the attacker system to leverage the hands-on. Also, the python code myarpv2.py detailed in the next section is uploaded.

```
#19hwt.ps1
$myArray = ("op1","op2",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\myarpv2.py  mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor/
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\ch4.pcap mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor/
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/mv /home/mentor/myarpv2.py /home/hack/workspace/'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/mv /home/mentor/ch4.pcap /home/hack/workspace/'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chown hack:turku -R /home/hack/workspace/'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chmod 755 /home/hack/workspace/myarpv2.py'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chmod 644 /home/hack/workspace/ch4.pcap'
   #Check
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo ls -la /home/hack/workspace/myarpv2.py'
```

```
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo cat /etc/sudoers.d/0hwt |grep python' +$op
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo cat /etc/sudoers.d/0hwt |grep shark' +$op
}
Remove-Variable -Name "OPN" -Force
```

4.9.2. CHALLENGE 4 PREPAREDNESS

It is acknowledged/well known that an Ethernet network interface is identified by a Medium Access Control (MAC) address. Also, a MAC acts as a unique identifier at the data link layer of the Open System Interconnection (OSI). Therefore, the scenario requires the attacker can inject forged ARP frames in a common network. An additional interface is enabled in the attacker system as follows:

The following network settings are stated in the new interface file named if-eth1:

```
auto eth1
iface eth1 inet static
address 10.1.1.28
netmask 255.255.255.248
```

The following powershell script file “16hwt.ps1” uploads the file if-eth1 and ensures the rights and correct path for the file as follows:

```
#16hwt.ps1
$myArray = ("op1","op2","op3",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\if-eth1  mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/mv /home/mentor/if-eth1 /etc/network/interfaces.d/'
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chown root:root /etc/network/interfaces.d/if-eth1'
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chmod 644 /etc/network/interfaces.d/if-eth1'
    #Clean
    #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-op22-
ip.westeurope.cloudapp.azure.com 'sudo rm -rf /etc/network/interfaces.d/if-eth1'
    #Check
    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo ls -la /etc/network/interfaces.d/if-eth1' +$op
}
Remove-Variable -Name "OPN" -Force
```

An ARP query aims to deliver a system by broadcasting a message to the expected IP, and the requested system replies back with its correct MAC address. However, there are two mechanisms to add a new entry in the ARP table, the ARP request-reply and Gratuitous ARP. Since the replies or Gratuitous ARP frames are declarative protocols, there is not a default authentication. Thus, an attacker in the same network could forge a

frame to act on behalf of a host, such as a gateway. Then, the goal is to get some familiarity with Scapy library over a python code called myarpv2.py. This code is an adaptation based on an open source Python project [33]. The next set of instructions exhibit the main logic:

```
#snippet code - challenge 4
def send_arpspoof(IFACE_MAC,IFACE_IP,V_IP,GW_IP):
  V_MAC = get_MAC(IFACE_MAC,V_IP)
  GW_MAC = get_MAC(IFACE_MAC,GW_IP)
  #<<< Complete the Missing Parts denoted by dashes>>>
  # victim poisoning, sends ARP packets to victim by faking gateway
  send(ARP(hwdst=---,pdst=---,hwsrc=--_MAC,psrc=--_IP,op=2),iface='eth1')
  # gateway poisoning, sends ARP packets to the gateway by faking victim
  send(ARP(hwdst=---,pdst=---,hwsrc=--_MAC,psrc=--_IP,op=2),iface='eth1')
  #<<< End of the Missing Parts
IFACE_MAC = get_if_hwaddr('eth1')
IFACE_IP = get_if_addr('eth1')
```

4.10. HANDS-ON 7: SSL OPENSSL AND SSLSTRIP ATTACK

The preparedness for hands-on 7 is relevant for the upcoming challenges even though there is no announcement of a challenge in this hands-on. The goal of this hands-on is to enable the project Schoolmate [39][53][54], which is a purposely vulnerable web application to leverage the understanding of OpenSSL [35] and the SSLStrip attack [55]. Also, this website becomes the main target in the next challenges.

The Schoolmate website is designed as a main point of interaction between students, parents, teachers and administrative staff at a standard school. The website is a LAMP-alike project, so it consists of a Linux Debianv8.11, Apachev2.4.10, MySQLv5.5.62 and PHPv5.6.40.

4.10.1. SCHOOLMATE PROJECT AND OPENSSL

The Hypertext Transfer Protocol (HTTP) is a request and response-oriented protocol, the interest is to understand the exchanging of messages. Since the HTTP messages are exchanged in cleartext, the cryptographic toolkit provided by OpenSSL is helpful to improve the security of the messages. The Secure Socket Layer (SSL) and Transport Layer Security (TLS) are introduced and also applied by using the Apache security module [58]. The student analyzes and practices digital certificates to enable a secure website applying HTTPS.

The following shell script file “10hwt” is implemented to deploy the Schoolmate website over HTTP and HTTPS protocols.

```
#10hwt
#!/bin/bash
export SSHPASS='secret'
cat <<_EOF_ | /usr/bin/sudo -s
sudo /usr/bin/chmod 600 /home/mentor/SchoolMateHwt_v1.5.4.tar.gz
sudo /usr/bin/chown mentor:turku /home/mentor/SchoolMateHwt_v1.5.4.tar.gz
sudo /usr/bin/echo "10.1.1.22 hack-debian.turku.fi" >> /etc/hosts
_EOF_
```

```
 9 /usr/bin/sshpass -e scp /home/mentor/SchoolMateHwt_v1.5.4.tar.gz
mentor@10.1.1.22:/home/mentor/.
 10 # hack-debian v8.11 jessie
 11 cat <<_EOT_ | /usr/bin/sshpass -e ssh mentor@10.1.1.22
 12 /usr/bin/sudo -s
 13 /bin/echo "10.1.1.22 hack-debian.turku.fi" >> /etc/hosts
 14 /bin/sed -i 's/HISTSIZE=1000/HISTSIZE=1/' /home/mentor/.bashrc
 15 /bin/sed -i 's/HISTFILESIZE=2000/HISTFILESIZE=0/' /home/mentor/.bashrc
 16 /usr/bin/apt-get -y update
 17 /usr/bin/apt-get -y install debconf-utils
 18 /usr/bin/apt-get -y install apache2
 19 /bin/systemctl enable apache2
 20 /bin/systemctl start apache2
 21 /bin/mv /home/mentor/SchoolMateHwt_v1.5.4.tar.gz /var/www/html/.
 22 /bin/sed -i 's|#ServerName www.example.com|ServerName hack-debian.turku.fi|' \
/etc/apache2/sites-available/000-default.conf
 23 /bin/sed -i 's|DocumentRoot /var/www/html|DocumentRoot /var/www/html/schoolmate|' \
/etc/apache2/sites-available/000-default.conf
 24 /usr/sbin/a2ensite default-ssl
 25 /bin/systemctl reload apache2
 26 /usr/sbin/a2enmod ssl
 27 /bin/systemctl restart apache2
 28 /usr/bin/openssl genrsa -out /etc/ssl/private/myca.key 2048
 29 /bin/mkdir /etc/apache2/ssl.crt
 30 /usr/bin/openssl req -new -x509 -days 365 -key /etc/ssl/private/myca.key \
-out /etc/apache2/ssl.crt/myca.crt -subj '/C=FI/ST=Turku/L=Turku/O=HWT/OU=HWT/CN=myca.turku.fi'
 31 /usr/bin/openssl genrsa -out /etc/ssl/private/mydebian.key 2048
 32 /usr/bin/openssl req -new -sha256 -key /etc/ssl/private/mydebian.key -out \
/etc/ssl/certs/mydebian.csr \
-subj '/C=FI/ST=Turku/L=Turku/O=HWT/OU=HWT/CN=hack-debian.turku.fi'
 33 /usr/bin/openssl x509 -req -days 365 -in /etc/ssl/certs/mydebian.csr -CA \
/etc/apache2/ssl.crt/myca.crt \
-CAkey /etc/ssl/private/myca.key -CAcreateserial -out /etc/ssl/certs/mydebian.pem
 34 /bin/sed -i 's/ssl-cert-snakeoil.key/mydebian.key/' \
/etc/apache2/sites-available/default-ssl.conf
 35 /bin/sed -i 's/ssl-cert-snakeoil.pem/mydebian.pem/' \
/etc/apache2/sites-available/default-ssl.conf
 36 /bin/sed -i 's|#SSLCACertificateFile /etc/apache2/ssl.crt/ca-bundle.crt|SSLCACertificateFile
/etc/apache2/ssl.crt/myca.crt|' \
/etc/apache2/sites-available/default-ssl.conf
 37 /bin/sed -i 's|DocumentRoot /var/www/html|DocumentRoot /var/www/html/schoolmate|' \
/etc/apache2/sites-available/default-ssl.conf
 38 /bin/sed -i 's/index.html/index.tmp/' /etc/apache2/mods-available/dir.conf
 39 /bin/sed -i 's/index.php/index.html/' /etc/apache2/mods-available/dir.conf
 40 /bin/sed -i 's/index.tmp/index.php/' /etc/apache2/mods-available/dir.conf
 41 /bin/sed -i 's|DocumentRoot /var/www/html/schoolmate|Redirect permanent / https://hack-
debian.turku.fi/ |' \
/etc/apache2/sites-available/000-default.conf
 42 /bin/chown root:ssl-cert /etc/ssl/private/my*.key
 43 /bin/chmod 640 /etc/ssl/private/my*.key
 44 /bin/systemctl restart apache2
 45/usr/bin/debconf-set-selections <<< 'mysql-server-5.5 mysql-server/root_password password
secret'
 46 /usr/bin/debconf-set-selections <<< 'mysql-server-5.5 mysql-server/root_password_again
password secret'
```

```
47 /usr/bin/apt-get -y install mysql-server
48 /usr/bin/apt-get -y install php5 libapache2-mod-php5 php5-mcrypt php5-mysql
49 /bin/tar -xzvf /var/www/html/SchoolMateHwt_v1.5.4.tar.gz -C /var/www/html
50 /bin/chmod 600 -R /var/www/html/SchoolMateHwt_v1.5.4.tar.gz
51 /bin/chmod 755 -R /var/www/html/schoolmate
52 /bin/chmod 700 -R /var/www/html/schoolmate/*.sql
53 /usr/bin/mysql -u root -p"secret" < /var/www/html/schoolmate/dumpSchoolMate.sql
54 /usr/bin/mysql -u root -p"secret" < /var/www/html/schoolmate/authSchoolMate.sql
55 /bin/systemctl restart apache2
56 _EOT_
```

The following powershell script file “18hwt.ps1” uploads the shell script file “10hwt” to deploy the Schoolmate website in the student’s instances:

```
1 #18hwt.ps1
2 $myArray = ("op1","op2","op3",...,"op22")
3 New-Variable -Name "OPN" -Visibility Public -Value "secret"
4 foreach ($op in $myArray) {
5    .\pscp.exe -i id_rsa.ppk -pw $OPN .\10hwt mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor/
6    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chown mentor:turku /home/mentor/10hwt'
7    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chmod 700 /home/mentor/10hwt'
8    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /home/mentor/10hwt'
9    #Clean
10   #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-op22-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/*hwt'
11    #Check
12    .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'ping 10.1.1.22 -c 2' + $op
13 }
14 Remove-Variable -Name "OPN" -Force
```

4.10.2. SSLSTRIP ATTACK

The SSLStrip attack is introduced by the developer Moxie Marlinspike at the Black Hat DC in 2009 [55]. The attack has the potential to hijack HTTP traffic on a network against a victim while it maintains the original HTTPS communication with the server. SSLStrip is mounted over a middle position with the aim to strip out any HTTPS link from an HTTP connection. Thus, the SSLStrip attack intercepts HTTP traffic rather than HTTPS. A browser can mitigate the attack by forcing the use of HTTP Strict Transport Security (HSTS).

The SSLStrip attack can be implemented in different ways such as ARP poisoning, rogue hotspot and by setting a proxy to derive the browser traffic. In this hands-on the focus is the understanding of the proxy. The attacker system hack-kali has enabled the tool at /usr/bin/sslstrip that it is used to intercept the traffic against Schoolmate. For instance:

```
1 $sudo sslstrip -w /home/hack/workspace/stripLog -a -l 8080
```

The following shell script file “20hwt” adjusts some settings over hack-ubuntu and hack-win. As the ARP poisoning attack alters the ARP table, some additional commands are authorized at hack-ubuntu. Additionally, the domain controller includes a domain name service (DNS) register to bind hack-debian for further tests.

```
#20hwt
#!/bin/sh
export SSHPASS='secret'
export WINEXE='mentor%secret'
# hack-ubuntu v14.04.5 trusty
/usr/bin/sshpass -e ssh mentor@10.1.1.29 'sudo /bin/touch /home/mentor/0hwt'
/usr/bin/sshpass -e ssh mentor@10.1.1.29 'sudo /bin/echo "%turku ALL=(ALL) NOPASSWD:/usr/sbin/arp" >> /home/mentor/0hwt'
/usr/bin/sshpass -e ssh mentor@10.1.1.29 'sudo /bin/echo "%turku ALL=(ALL) NOPASSWD:/sbin/ip" >> /home/mentor/0hwt'
/usr/bin/sshpass -e ssh mentor@10.1.1.29 'sudo /bin/echo "%turku ALL=(ALL) NOPASSWD:/bin/ip" >> /home/mentor/0hwt'
/usr/bin/sshpass -e ssh mentor@10.1.1.29 'sudo /bin/cp /home/mentor/0hwt /etc/sudoers.d'
/usr/bin/sshpass -e ssh mentor@10.1.1.29 'sudo /bin/chmod 440 /etc/sudoers.d/0hwt'
# hack-win - v2008_R2_SP1 and winexe v1.1 en Kali.
/usr/bin/winexe -U "$WINEXE" //10.1.1.30 'cmd.exe /c dnscmd.exe /Config turku.fi /AllowUpdate 1'
/usr/bin/winexe -U "$WINEXE" //10.1.1.30 'cmd.exe /c dnscmd.exe hack-win.turku.fi /RecordAdd turku.fi hack-debian 10 A 10.1.1.22'
/usr/bin/winexe -U "$WINEXE" //10.1.1.30 'cmd.exe /c ipconfig /flushdns'
exit
```

The following powershell script file “21hwt.ps1” uploads the shell script file “20hwt” to enable the additional settings over hack-ubuntu and hack-win systems.

```
#21hwt.ps1
$myArray = ("op1","op2","op3",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\20hwt mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com:/home/mentor/
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/chown mentor:turku /home/mentor/20hwt'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/chmod 700 /home/mentor/20hwt'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /home/mentor/20hwt'
  #Clean
  #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-op22-ip.westeurope.cloudapp.azure.com 'sudo rm -rf /home/mentor/workspace/*.pcap'
  #Check
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo cat /etc/sudoers.d/0hwt |grep sslstrip' + $op
}
Remove-Variable -Name "OPN" -Force
```

4.11. HANDS-ON 8: WEBSCARAB AND BURPSUITE

The eighth hands-on extends the practice over a proxy service which is understood as the element located in the middle of the communication between a user and an application. From the penetration testing perspective, a proxy aims to test web applications for their security since a proxy may control the access to a web-based service via HTTP. Some applications in this scope are Webscarab, Burpsuite, OWASP Zed Attack Proxy (ZAP), and Twisted Project, among others.

The OWASP WebScarab is a project proposed by Rogan Dawes which was active from 2002 to 2011 [37]. WebScarab is written in Java and introduces a framework for analyzing applications using HTTP and HTTPS for communication. On the other hand, the Burpsuite is a platform for security testing of web applications. Burpsuite is written in Java and developed by PortSwigger Web Security having three editions, Enterprise, Professional and the Community [38].

In this context, the next powershell script file aims to enable the appropriate Java version for the aforementioned security tools. Challenge 5 is covered in this hands-on, and multiple files are uploaded into the attacker system to analyze the behavior of HTTP through a simple HTTP server leveraged by python. The goal is to analyze the GET requests further by solving two incomplete codes written in python language [34], named mygetHttpv1.py and mygetHttpsv2.py.

4.11.1. CHALLENGE 5 PREPAREDNESS

The challenge requires the preloading of a static "index.html" file including the Cascading Style Sheets definitions (CSS) as well as enabling a form HTML element to get a user and a password.

The first part of the challenge focuses on the code mygetHttpv2.py that is served from the attacker system and is rendered from the client side using the browser Firefox. The following extract exhibit the main logic:

```
# Snippet code - challenge 5
class myHandler(http.server.SimpleHTTPRequestHandler):
  def go_GET(self):
          self.path == '/home/hack/workspace/html/index.html'
          return http.server.SimpleHTTPRequestHandler.go_GET(self)
#<<< Complete the Missing Parts denoted by dashes>>>
httpd = socketserver.TCPServer((----,----),----)
```

The second part of the challenge focuses on the code mygetHttpsv2.py that extends the existing logic by including cryptographic elements to enable HTTPS. The HTTPS requires a signed certificate and the correspondent key. The following extract exhibits the main logic:

```
# Snippet code - challenge 5
myHandler = SimpleHTTPRequestHandler
httpd = HTTPServer((----,----),----)
httpd.socket = ssl.wrap_socket(httpd.socket,keyfile=----,certfile=----,server_side=True)
```

Lastly, the following powershell script file “25hwt.ps1” uploads the challenge’s files and ensures the rights and correct path for the files as follows:

```
#25hwt.ps1
$myArray = ("op1","op2","op3",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
.\pscp.exe -i id_rsa.ppk  -pw $OPN .\mygetHttpv2.py mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com:/home/mentor
.\pscp.exe -i id_rsa.ppk  -pw $OPN .\mygetHttpsv2.py mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com:/home/mentor
.\pscp.exe -i id_rsa.ppk  -pw $OPN .\webscarab-selfcontained-20070504-1631.jar mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com:/home/mentor
.\pscp.exe -i id_rsa.ppk  -pw $OPN .\index.html mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com:/home/mentor
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/mkdir /home/hack/workspace/html'
 .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/chmod 755 /home/hack/workspace/html'
 .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/mv /home/mentor/mygetHttp* /home/hack/workspace/html'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/mv /home/mentor/webscarab* /home/hack/workspace/html'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/mv /home/mentor/index.html /home/hack/workspace/html'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/chown hack:turku -R /home/hack/workspace/html'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/chmod 644 /home/hack/workspace/html/*'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/update-alternatives --set java /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/chmod 700 /home/mentor/0hwt'
  #Clean
    #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-op22-ip.westeurope.cloudapp.azure.com 'sudo rm -rf /home/mentor/*hwt*'
  #Check
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'sudo /usr/bin/update-alternatives --get-selections |grep java-8'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com 'ls -la /home/hack/workspace/html/mygetHttp*' + $op
}
Remove-Variable -Name "OPN" -Force
```

4.12. HANDS-ON 9: DIRECTORY LISTING AND PATH TRAVERSAL

The HTTP protocol was created by Sir Timothy John Berners-Lee in 1989, so the students reflect about the evolution of the protocol that has been used by the World Wide Web since 1990 [56][57]. HTTP is a stateless and object-oriented protocol, so the goal is to understand the semantics used. In addition, the open source HTTP server Apache is studied to have a broader view about its functionalities and modules related to security [58]. Assuming there is a lack of hardening or a lax pathname construction, a document

resource can be exposed to a data leakage. Thus, the motivation is to explore the directory listing issue based on over relaxed directives and a path traversal attack to seize improper neutralize elements.

The attacker system hack-kali has enabled the tools curl and wfuzz, that can be used to spot an eventual path traversal flaw. For instance:

```
$ curl 10.1.1.14:5555/?='../myarp.py%00'&sbt='submit
$ wfuzz -c -z file,/usr/share/wfuzz/wordlist/Injections/Traversal.txt
http://10.1.1.14:5555/index.html?name=FUZZ
```

4.12.1. CHALLENGE 10 FIRST ANNOUNCEMENT PREPAREDNESS

According to the specific requirements (See Annex B: Requirements for Multiple Instances for more information), some artificial vulnerabilities has been induced in the WebSite replicated. Then, the students need to identify the flaw and from it plan an attack strategy to break into the system. The replicated WebSite is available from a complementary cloud on the Internet, and the adjustments are based merely in the DNS resolution of the Uniform Resource Locator (URL) of the WebSite.

The following shell script file “22hwt” is implemented to update the firewall service with additional firewall rules to allow the traffic between the student’s instance and the replicated WebSite.

```
#22hwt
#!/bin/sh
export SSHPASS='secret'
cat <<_EOT_ | sudo -s
/usr/bin/mv /home/mentor/4hwt /etc/network/if-pre-up.d
/usr/bin/chown root:root /etc/network/if-pre-up.d/4hwt
/usr/bin/chmod 600 /etc/network/if-pre-up.d/4hwt
/usr/sbin/iptables-restore < /etc/network/if-pre-up.d/4hwt
/usr/bin/sed -i 's|#supersede domain-name "fugue.com home.vix.com";|supersede domain-name
"domain.fi";|' /etc/dhcp/dhclient.conf
/usr/bin/sed -i 's|#prepend domain-name-servers 127.0.0.1;|prepend domain-name-servers
xxx.xxx.xx.x;|' /etc/dhcp/dhclient.conf
/usr/bin/systemctl restart networking
_EOT_
exit
```

Lastly, the powershell script files “23hwt.ps1” and “24hwt.ps1” are used to trigger and check the DNS binding, so it is also possible to unify them, as follows:

```
#23hwt.ps1
$myArray = ("op1","op2","op3",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\4hwt  mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\22hwt mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chown mentor:turku /home/mentor/22hwt'
```

```
 8   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo chmod 700 /home/mentor/22hwt'
 9   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /home/mentor/22hwt'
 10   #Clean
 11   #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-op22-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/22hwt'
 12   #Check
 13   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'curl WebSite.domain.fi |grep HACK' + $op
 14 }
 15 Remove-Variable -Name "OPN" -Force
```

4.13. HANDS-ON 10: REDIRECT AND COMMAND INJECTIONS

The hands-on number 10 focuses on the code analysis to identify the conditions raising an unauthorized redirection or command execution. Consequently, the Apache document root path allows the modification of the developed resources by the hack user. Thus, the student explores the Hyper Text Markup Language (HTML) document scheme [59] that shapes the front-end of the website, yet the server-side logic is viciously embedded into the front-end being prone to injections. Since the Schoolmate is a dynamic website having direct access to a MySQL database, the interest of the 6th challenge is the unauthorized disclosure of the database credentials. The goal is to perform a command injection to reveal sensitive information [41].

4.13.1. CHALLENGE 6 PREPAREDNESS

PHP stands for Hypertext Preprocessor that interprets the scripting routines in the server side [36]. The targeted document is ManageSchoolInfo.php, that is adapted to include a new parameter associated with a PHP function called “shell_exec” that aims to execute commands via shell and returns the complete output as a string. When the parameter is controlled by a user-input and having an improper validation of it, the injection condition can then be proven. The tool applied is Burpsuite over Firefox browser. To exemplify, the next one is a tested case:

```
 1 $ vi /var/www/html/schoolmate/ManageSchoolInfo.php
 35 $redirect_file = $_POST[file];
 36 $output = shell_exec('/bin/cat '.$redirect_file);
 139 <textarea name='sitemessage' cols=40 rows=10>$sitemessage . $output </textarea>
 155 <input type='hidden' name='file' value='$redirect_file'>
```

Lastly, the students provide their own payloads to seize the flaw and gather the credentials, proving the injection issue.

The following shell script file “26hwt” is deployed to grant additional rights to the student user over the Schoolmate document root. An access control list is enabled to avoid functional restrictions since the website requires higher privileges to run smoothly.

```
 1 #22hwt
 2 #!/bin/sh
 3 export SSHPASS='secret'
```

```
4 /usr/bin/echo "10.1.1.14 hack-kali.turku.fi" >> /etc/hosts
5 # hack-debian v8.11 jessie
6 /usr/bin/sshpass -e ssh mentor@10.1.1.22 'sudo setfacl -m g:turku:rwx -R
/var/www/html/schoolmate/'
7 sudo echo "hack-debian done"
8 exit
```

The following powershell script file “27hwt.ps1” uploads the shell script file “26hwt” to enable the additional settings over hack-debian system.

```
1 #27hwt.ps1
2 $myArray = ("op1","op2","op3",...,"op22")
3 New-Variable -Name "OPN" -Visibility Public -Value "secret"
4 foreach ($op in $myArray) {
5   .\pscp.exe -i id_rsa.ppk -pw $OPN .\26hwt mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
6   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chmod 700  /home/mentor/26hwt'
7   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /home/mentor/26hwt'
8   #Clean
9   #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-op22-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/*hwt*'
10   #Check
11   .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'/usr/bin/cat /etc/hosts |grep ".14"' + $op
12 }
13 Remove-Variable -Name "OPN" -Force
```

4.14. HANDS-ON 11: OS COMMAND INJECTIONS

The hands-on number 11 stays tied to code analysis to identify the conditions raising injections, which is a security risk identified by the Open Web Application Security Project Top Ten [60]. The OWASP foundation started in 2001 and is a global community encouraged to improve software safety and security [41]. The Top Ten ranks the most critical web application security risks based on a Risk Rating Methodology to prioritize the vulnerabilities. The injections occur when untrusted data is sent to an interpreter as part of a command or query. The attacker tricks the interpreter into executing unintended commands or accessing data without proper authorization.

Meanwhile, the focus remains in the code analysis, the HTML is studied further to understand the specification of the Document Object Model originally defined by the World Wide Web Consortium (W3C) [61]. The students explore the DOM elements that allow the programs and scripts to dynamically access and update the content, structure and style of documents. Thus, the exploration over JavaScript, invented in 1995 by Brendan Eich who is also the co-founder of Mozilla, is inevitable. JavaScript resides in the client-side, and it is interpreted by the browsers [62], including extensions for the user’s dynamic interactions. By controlling the HTML document content and appearance, JavaScript becomes relevant to enhance the user experience and usability of the websites. Then, JavaScript can change the behavior of the documents, which are also attractive for attackers seeking to force unexpected behaviors on the webpage.

4.14.1. PASSWORD VALIDATION PREPAREDNESS

Through hands-on 11, an HTML form is altered by embedding a script to enhance the complexity of a gathered password. One case is performed over the python simple HTTP Server module, so the scenario is static without interactions with a database, for example:

```
$ python -m SimpleHTTPServer 5555
$ vi /home/hack/workspace/www/index.html
function validate(){
  var passwd = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}$/;
   if(document.getElementById("id2").value.match(passwd)){
      document.getElementById("f1").submit();}
  else{
      alert('Password Weak!');}
 }
<input id="id3" type="submit" name="sbt" value="submit" onclick="validate();" />
```

The following powershell script file “28hwt.ps1” uploads all the required files for the static site, and ensures the rights and proper location of them in the attacker machine.

```
#28hwt.ps1
$myArray = ("op1","op2","op3",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\index.html mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\my*.js mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\my*.css mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/mkdir /home/hack/workspace/www'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chmod 755 /home/hack/workspace/www'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/mv /home/mentor/my*.css /home/hack/workspace/www'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/mv /home/mentor/my*.js /home/hack/workspace/www'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/mv /home/mentor/index.html /home/hack/workspace/www'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chown hack:turku -R /home/hack/workspace/www'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chmod 644 /home/hack/workspace/www/*'
  #Clean
  #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-op22-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/*hwt*'
  #Check
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'ls -la /home/hack/workspace/www/my*' + $op
}
Remove-Variable -Name "OPN" -Force
```

4.14.2. CHALLENGE 7 PREPAREDNESS

The targeted document is Adduser.php and is part of the Schoolmate website, which is analyzed by the students to identify parameters exposed to an injection. The goal is to find the parameter and perform an injection, which is eventually stored into the database being a first approximation in the direction of Cross Site Scripting problematic [6][63]. Also, it is likely that the payloads provided by the students demonstrate variations depending on their assumptions and creativity. Lastly, this challenge inherits the preparedness carried out by this hands-on and the previous challenge 6, so there are no additional adjustments.

4.15. HANDS-ON 12: SQL INJECTIONS AND W3AF

The preparedness for the hands-on 12 covers the Structure Query Language (SQL) injection attack [39][63] and the implementation of the Web Application Attack and Audit Framework W3af [40].

The SQL is a declarative query language, designed by Donald Chamberlin and Raymond Boyce in the early 1970s [64]. Through SQL we can access and manipulate relational database systems. A relational model is based on set theory and predicate logic allowing the data to be structured in a series of tables. The SQL predicate logic can identify relations between different entities or variables by means of SQL statements. In 1987 SQL became the standard ISO 9075, being updated in 2016 [3].

From another stance, the object-oriented database model stores the data as objects being an alternative model. A variation is the hybrid relational database that is characterized by building an object-oriented interface on top of the original system. Lastly, as a relational database ensures the entity integrity and referential integrity through the use of a primary and foreign key management, the security concern is the disregarding of the semantics of the data stored [3][63].

Furthermore, since the source code analysis increases the precision in detecting a SQL injection attack, frequently a pen-tester faces blind scenarios having handy only the front-end of a web application. Thus, the open-source project W3af contributes to the automation of a first sight detection to direct the exploitation effort. The W3af was developed by Andrés Riancho in 2007 [40], and the code is written in Python. Also, the W3af is being shaped as a framework similar to metasploit, so the usage of its multiple plugins is intuitive.

4.15.1. NUMERIC SQL INJECTION

The SQL injection is also referred to as a Failure to Preserve SQL Query Structure according to the Common Weakness Enumeration community [65]. An attacker inserts malformed data into parameters controlled by the user, and by processing such insertion the application changes the semantics of a coded SQL statement. A user-input is used to alter a SQL statement taking advantage of any insecure string concatenation or insecure construction. The software constructs all or part of a SQL statement using external user-inputs without neutralizing special elements that are sent to the backend. A SQL numeric injection overwrites constructions based on concatenated string that expects a certain id parameter. The next case is an adaptation of the page ViewGrades.php:

```
#Snippet code
$mystatement = "SELECT * FROM users WHERE userid = $myuid ";
$myquery = mysqli_query($mysqli,$mystatement);
while ($mydata = mysqli_fetch_array($myquery)){
```

The sanitization of the aforementioned case is approached by means of a prepared SQL statement that is also referred to as a parameterized query. The next example is used to validate the correct form of the data to be processed by the statements coded in ViewGrades.php:

```
#Snippet code
 $stmt = mysqli_prepare($mysqli,"SELECT username, type FROM users WHERE userid = ?");
 mysqli_stmt_bind_param($stmt, 'i', $myuid);
 mysqli_stmt_execute($stmt);
 mysqli_stmt_bind_result($stmt, $myname, $mytype);
 while (mysqli_stmt_fetch($stmt))
 {
  print("<tr class='".( $row%2==0 ? "even" : "odd" )."'>
  <td align='left'>$myname</td>
  <td align='left'>$mytype</td>
```

The following shell script file “29hwt” covers the implementation of W3af, uploads the required files to be included in the Schoolmate’s document root, and ensures the rights and proper location of them in the hack-debian system.

```
#29hwt
#!/bin/sh
# hack-kali vkali-Rolling and sshpass v1.06
export SSHPASS='secret'
cat <<_EOT_ | sudo -s
/usr/bin/sed -i 's/# deb-src/deb-src/' /etc/apt/sources.list
/usr/bin/apt-get update
/usr/bin/apt-get build-dep -y lxml
/usr/bin/apt-get -y install seclists
/usr/bin/apt-get -y install npm
/usr/bin/pip install lxml==3.4.4
/usr/bin/pip install xdot==0.6
/usr/bin/pip install npm
/usr/bin/git clone https://github.com/andresriancho/w3af.git /home/hack/workspace/w3af
/home/hack/workspace/w3af/w3af_console
/tmp/w3af_dependency_install.sh
/usr/bin/setfacl -m g:turku:rwx -R /home/hack/workspace/w3af
_EOT_
```

```
# hack-debian v8.11 jessie
/usr/bin/sshpass -e scp /home/mentor/ViewGrades.hwt* mentor@10.1.1.22:/home/mentor/
cat <<_EOT_ | /usr/bin/sshpass -e ssh -t -t mentor@10.1.1.22
sudo /bin/chown root:root /home/mentor/ViewGrades.hwt*
sudo /bin/mv /home/mentor/ViewGrades.hwt* /var/www/html/schoolmate/
sudo /usr/bin/setfacl -m g:turku:rwx -R /var/www/html/schoolmate/
sudo /usr/bin/getfacl /var/www/html/schoolmate/ViewGrades.*
_EOT_
exit
```

As usual, the following powershell script file "30hwt.ps1" deploys the previous shell script file for all the student's instances.

```
#30hwt.ps1
$myArray = ("op1","op2","op3",...,"op22")
New-Variable -Name "OPN" -Visibility Public -Value "secret"
foreach ($op in $myArray) {
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\ViewGrades.hwt* mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\pscp.exe -i id_rsa.ppk  -pw $OPN .\29hwt mentor@hack-kali-$op-
ip.westeurope.cloudapp.azure.com:/home/mentor
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /usr/bin/chmod 700 /home/mentor/29hwt'
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo /home/mentor/29hwt'
  #Clean
  #.\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-op22-ip.westeurope.cloudapp.azure.com
'sudo rm -rf /home/mentor/*hwt*'
  #Check
  .\plink.exe -i id_rsa.ppk -pw $OPN mentor@hack-kali-$op-ip.westeurope.cloudapp.azure.com
'sudo echo $SSH_CLIENT' + $op
}
Remove-Variable -Name "OPN" -Force
```

4.16. CHALLENGE 8 PREPAREDNESS

Challenge 8 aims to modify the password of any user by means of a SQL injection [39][63] rather than addressing a targeted document or parameter. As a result, the student proposes certain conditions understood either as a SQL backdoor or unvalidated parameter to prove the injection. So, it is likely that the payloads provided by the students demonstrate variations depending on their assumptions and creativity. Lastly, this challenge inherits the preparedness carried out by this hands-on and challenge 6, so there is not additional adjustments.

4.17. HANDS-ON 13: XSS AND OWASP ZAP I

Hands-on 13 approaches the first order of Cross Site Scripting (XSS) attacks [6], particularly the reflected ones. In addition, the project OWASP Zed Attack Proxy is introduced to intercept HTTP messages and also to fuzz parameters with multiple payloads [41][60].

At present, the WWW involves multiple interactions between many sites, and the scripting languages are the glue for those interactions. The scripts can be included in the

<body> and <header> HTML sections, and any function call has the potential to alter the expected behavior of the website. Indeed, a XSS attack is a type of injection where malicious scripts are injected in trusted websites. An incorrect neutralization allows for the processing of a user-input that is treated as a valid output in the web page served to the user. XSS is a well-known thread, registered in the 4th position of the OWASP top ten of 2003 and prevails in the 7th position of the ranking released in 2017. Thus, XSS evolves exhibiting at least three variations: reflected XSS (or Non-persistent), stored XSS (or persistent) and DOM-based XSS.

Mitigations like the same origin policy limits the resources accessible to scripts running on a given website or origin. The policy prevents the access to resources from other sites or origin at the client-side, or the browser. Nevertheless, the policy is not infallible considering highly dynamic websites leveraged by Asynchronous JavaScript and XML (AJAX), JQuery and Fetch application programming interface (API), among others.

As the natural successor of OWASP Webscarab, the OWASP Zed Attack Proxy is an open source penetration testing tool for finding vulnerabilities in web applications. ZAP is developed in Java and it is maintained by an active community that reached the project Flagship category at OWASP Foundation [41].

4.17.1. REFLECTED XSS (NON-PERSISTENT)

The XSS injection is also referred to as a Failure to Preserve the Web Page Structure according to the Common Weakness Enumeration community [66]. Any language or technology used to build applications can be affected. The student targets a PHP page inside the Schoolmate website, called StudentMain.php. Once the code is reviewed, we can distinguish an unvalidated parameter named $page, as follows:

```
79 <input type='hidden' name='page' value='$page’>
```

By analyzing the code, we can determine the aim of the parameter that derives the users to the contents associated to the user’s profile. The associated HTTP request conveys the following parameters:

```
1 page2=0&logout=&page=4&selectclass=
```

Then, a payload is injected to prove the improper validation of the user-input, for instance:

```
1 page2=0&logout=&page=4'><a href=“http://10.1.1.28:5555">ClickMe</a><br'&selectclass=
```

The $page parameter is requested from different pages, yet it is originally initialized at the index page, as follows:

```
1 $page = $_POST["page"];
```

Hence, from the server-side web we can propose a remediation to neutralize the flaw by forcing only numeric values as the next instruction. Other recommendations can be the refinement of HTML attributes and the use of additional functions to control the input.

```
1  $page = intval($_POST["page"]);
```

Lastly, OWASP ZAP is allowed at the sudoers list set by “0hwt”, and the interest is the Fuzzer function to automate the testing by either injecting their own payloads or those suggested by the project.

4.17.2. CHALLENGE 9 PREPAREDNESS

Challenge 9 is a targeted attack over a known parameter providing nonadditional clues about the usage and location in the documents. The goal is to prove a second order XSS injection where the student proposes his or her strategy and payload. So, it is likely that the payloads provided by the students demonstrate variations depending on their assumptions and creativity. Lastly, this challenge inherits the preparedness carried out by this hands-on and challenge 6, so there are no additional adjustments.

4.18. HANDS-ON 14: XSS AND OWASP ZAP II

The number 14 hands-on extends the discussion about the same origin restriction and the underestimated prevalence of Cross Site Request Forgery (CSRF) [41].

Since a strict security origin policy is inflexible for highly interactive websites, the cross-origin request authorization [67] becomes relevant to access the expected resources from the server-side. Indeed, cross origin requests mitigates the exposition to CSRF although it depends on the settings selected. For instance, the Cross-Origin Resource Sharing (CORS) can include a flexible header to tolerate any or * origin. Also, even when the credentials are not passed by default, those are still set at the discretion of the origin. Moreover, we can define different modes to fetch a request, such as same-origin, cors, cors-with-forced-preflight and even no-cors.

Therefore, it is worth noting that ad-hoc responses allow the retrieval of serialized data, and the concerns are the insecure deserialization, a security risk ranked in the 8th position in OWASP Top Ten 2017 [60].

4.18.1. DOM XSS

The document object model XSS attack is also referred as type 0 XSS, and it allows a client to perform an injection onto the page rather than the server [66]. DOM-based XSS generally involves server-controlled script such as Javascript that performs sanity checks on a form before the user submits it. Eventually, when a server-supplied script processes user-inputs, the response back to the web page can enable a dynamic HTML.

The hands-on targets a specific parameter over Schoolmate website called sitetext, which is also exposed to second order or stored XSS. The following payload is injected into the parameter which has a poor validation:

```
1 <script>function ft(){a=document.cookie;alert(a);};ft()</script>
```

Once the injection of the script is proven, then it becomes relevant to verify if the function call can trigger an interaction with other origins as follows:

```
1 <script>function ft(){a=document.getElementById("if1");
a.src="http://10.1.1.28:5555";};</script><iframe id="if1" frameborder="0" marginwidth="0"
marginheight="0" height="100%" width="100%" onload="ft();"></iframe>
```

An <iframe> or inline frame is used in this case to include an additional document inside the existing document. Such context is referencing our simple HTTP server to prove the injection. The iframe element has been chosen over other elements able to induce an equivalent effect to suggest the idea of multiple iframes and calls. Multiple inline frames enhance usability and increase interactions for dynamic websites as the work of Provos et al. analyzes further [68]. Conceiving the CSRF attack as an opportunistic attempt to forge or tamper a legitimate request, the aforesaid condition is attractive for an attacker to change the state of a request, even having no clue about the response.

The last step in this testing is about OWASP ZAP. Since the tool is allowed in the sudoers list, then the goal is automating the injection of a set of handcrafted payloads by using a fuzzer function.

4.18.2. CHALLENGE 10 PREPAREDNESS

The Challenge 10 is the last one, and it is prepared according to the specified requirements (See Annex B: Requirements for Multiple Instances for more information). The goal is to break-into the system considering the first announcement where the aim was to traverse the replicated WebSite and acquire information to proceed further. So, the students might have already found clues, which can be used to defeat the token-based authentication implemented. The replicated WebSite authorize a user to fetch resources by means of a token. A user login into the website by sending their username and password, so a request is sent to the backend application to validate the credential against to the database. Once the backend finds a match then responds returning a valid token. Hence, the students' needs to analyze first the scenario to plan the strategy and demonstrate the exploitation. Finally, the students consolidate their work in a report of findings to be scored as well.

4.19. HANDS-ON 15: PROTECTING YOUR REPORT: OPENSSL

A report of findings is the final outcome of the whole penetration testing process and declares the security state of the environment. As the methodology is flexible the final selection of findings can be based on different criteria, such as, detailed tactics and tools, proof of concepts or unit tests, risk assessment, etc.

Detailed tactics and tools describe the test scenario and strategy to prove a breach. A proof of concept (PoC) or unit test is a repeatable set of instructions coded to prove a breach. A risk assessment estimates the likelihood and impact of a threat given certain scenarios. Consequently, the report in itself is a sensitive asset that should be processed, stored and transmitted in a secure way.

To avoid the unintentional disclosure of the report content the goal of the hands-on lesson is to protect the file by means of encryption. There are multiple open source communities improving libraries to boost encryption, in this case is chosen OpenSSL once again. Besides implementing SSL and TLS protocols, the OpenSSL toolkit provides a general-purpose cryptography library. The student reflects about the usage of RSA algorithm by generating a key pair to protect the confidentiality and integrity of his or her work [69].

The hands-on lesson aims to encrypt a file by using the public key cryptography standard 8 (PKCS8) as a container. The next sentence is generating a private key following recommendation stated by NIST SP 800-131A [69]. The Privacy Enhanced Mail (PEM) format is used due to it can contain additional elements besides the private key. The output is a Base64 encoding of the original X509 specification.

```
$openssl genpkey –algorithm RSA \
        -pkeyopt rsa_keygen_bits:4096 \
        -pkeyopt rsa_keygen_pubexp:65537 | \
        openssl pkcs8 –topk8 –nocrypt –outform pem > mykey.p8
```

PKCS8 was introduced by RSA and specified by the RFC 5208:2010. Also, updated by RFC 5958 where is declared as a media subtype [70]. Thus, the public key can be extracted from the PKCS8 container, as follows:

```
$ openssl pkey -pubout -inform pem -outform pem \
        -in mykey.p8 -out mykey.pub
```

The RSA provides an asymmetric algorithm allowing the encryption of files by using the rsautl application, for instance:

```
$ openssl rsautl –encrypt –inkey mykey.pub –pubin \
        -in myreport.txt –out myreport.rsa
```

Lastly, the hands-on encourage the student to dig deeper in signing and verification of outcomes. Since any report should be digitally signed as proof of integrity of the original content.

4.20. HANDS-ON 16: PYTHON UNITTEST AND SELENIUM

The last hands-on preparedness, as in the previous hands-on, only requires the use of existing tools pre-installed in the attacker system. So, the main requirement is to enable the authorization at the sudoers file, as in the first script “0hwt”. As usual, the hands-on is scaffolded by a theoretical lecture that introduces the context and installs the discussion derived from the challenges. So, this time is slightly different as the closing hands-on provides the final ideas about improving findings outcomes.

From the perspective of the Software Development Life Cycle (SDLC), there is a sequence of stages addressing user requirements. Refactoring of software is common at different stages and before reaching production or a go-live point. In this sense, the security testing extends the functional-centered tests to identify flaws earlier to reduce costs. At the planning and designing stage, it is feasible to integrate the review of the security architecture and threat modeling. Through the development stage, a static application security testing (SAST) increases the accuracy of earlier flaw detections, and the analysis of binary code brings insights even over the compiled code. At the Testing stage, a dynamic application security testing (DAST), penetration testing and vulnerability scanning contribute with faster tests [3].

From another stance, this work has been unfolding multiple experiences based on the Python language to boost coding skills. The Python language is an interpreted and general-purpose programming language created by Guido van Rossum in 1991 [24]. So,

the motivation of this hands-on is to explore the unit testing framework of Python, which is inspired by JUnit to support test automation.

A unittest has mainly three sections: the initial preparedness of a unit test, the central unit test coded to verify a response for a specific input, and finally the test executed by a runner to provide a result. The reporting result is a Boolean value, confirming or not an assertion of the unit test. The next simple case is tested in the attacker machine:

```
#!/usr/bin/python
import unittest
def add(x,y):
    return x + y
class mytest(unittest.TestCase):
   def test(self):
       self.assertEqual(add(3,5), 8)
if __name__ == '__main__':
   unittest.main()
```

The instance TestCase is also the base class providing different groups of methods tackling the running of the test, implementing and checking the test and inquiring methods to return results.

On the other hand, the module urllib2 is reviewed so as to fetch Internet resources from URLs. By supporting different schemes, the module is able to support request-response oriented protocols, representing a URL request and response in the form of objects. The next code explores this scenario:

```
#!/usr/bin/python
import urllib2
req = urllib2.Request('http://10.1.1.28:5555/?name=name&pass=pass&sbt=submit’)
res = urllib2.urlopen(req)
page = res.read()
If len(page) >0 :
     print « » + page
```

Under this context, Selenium is introduced as a python API to write functional/acceptable tests using WebDrivers. Such API requires a driver to interface with a chosen browser such as Chrome, Edge, Firefox, and Safari. Thus, this hands-on encourages the students to explore both Selenium and Unittest to build proof of concepts for security testing.

4.21. DISCUSSION

The infrastructure supporting the teaching and learning process is based on cloud computing to provide the capacity and flexibility needed in progressive learning. An individual attack surface brings to the student the opportunity to practice at his or her own pace from home with minimal impact on their own computers. The mobility gained is beneficial for students located in distant areas that have Internet access. Furthermore, an individual attack surface implies responsibilities, so the students understand that they are liable citizens and they agreed upon the usage of the technology. Thus, any attack or analysis outside the scope of the attack is considered unauthorized.

From another stance, the infrastructure is adaptive, and the settings can be altered remotely by means of scripts. The scripts are tested in the mentor's instance supporting the continual refactoring of such scripts. There are multiple techniques and tools, and the most relevant ones for the methodology were chosen. However, the purpose is also to awaken a maker mindset to create tools. For such endeavor, the open source communities are preferred in this work as they are the main booster in making free software. Indeed, from the view of constructivist methods by exploring the environment is how the students can connect ideas and came up with new ones.

Through the testing and refactoring of scripts we can advert that the challenges are essential to direct the effort of the ICT operation. A model of challenges defines the purpose, scope, the techniques and tools of the learning experiences. From this viewpoint, it is feasible to reconcile the curiosity and critical thinking of the students under a legit environment prepared for penetration testing.

5. ANALYSIS OF RESULTS

5.1. METHOD

Challenges are announced gradually, so we can measure a timing period δTS between the moment when the challenge is released (T_0) and the shifted time when the student notifies the challenge's solution (T_1). By recording the timestamps in the form YYMMDD according to the RFC-3339 [42], we can estimate a distance using the decimal system. Consequently, the time units vary where an hour is associated to 10^2 units, a day 10^4 units and so on, as a logarithmic base 10 scale.

$$\delta TS = T_1 - T_0$$

The aim of the timing measurement is to break a tie of any exact match of points earned by the students or teams. Therefore, the points represent the number of challenges solved or hacked, and the δTS represents a complementary tiebreaker value useful to calculate a fair ranking.

5.2. CHALLENGE RESULTS ANALYSIS

The following table displays the results in the same order of the progressive challenges:

ID	Description	Hack Rate	MIN δTS	MAX δTS	AVG. δTS	SDV. δTS
CH1	Brute Force	0.8	316	99616	13325	23183
CH1	TimeLapsev2.py	0.7	406	90442	15787	22670
CH2	OpenVAS	0.6	81	197118	187868	563970
CH2	mybase64.py	0.7	36	2010748	180241	528993
CH3	hashdump	0.6	30	1960748	271107	591752
CH3	yourCookie.py	0.5	30	900118	112390	279294
CH4	tshark	0.4	4	1890748	562905	644080
CH4	myarpv2.py	0.2	636	829635	612678	353717
CH5	mygetHttpv2.py	0.4	156	1060748	161855	367290
CH5	mygetHttpsv2.py	0.3	156	39541	12055	16525
CH6	cmd injection	0.2	49916	110537	70127	28574
CH7	free injection	0.2	10921	20946	18410	4324
CH8	SQL injection	0.1	709824	769911	739868	30044
CH9	XSS injection	0.2	501	8635	3323	3759
CH10	Path Traversal	0.2	30211	150755	85530	51443
CH10	Report	0.3	59420	79515	66040	7623

Figure 7: Table of Challenge Results.

At first glance, the hack rate results steadily decreased while the challenges are shifted from system and network security towards web application security. From this perspective, the complexity increases unintendedly due to the shift in the scope rather than by the design. Accordingly, 80% of the students were able to execute a brute force attack applying Metasploit framework, whilst 10% of the participants proposed an attack vector for the SQL injection case. The fastest solution for the brute force attack arrived about 3 hours after the announcement of the challenge, and the average performance was about one day.

The challenge 5 denotes a transition between scopes and it focuses on the study of HTTP message exchanging. At this moment, there is a whole view of the penetration testing methodology and the students start to combine techniques. The fastest answer for the customized HTTP secure server challenge arrived in about 2 hours, and on average the answers were solved the next day.

In terms of timing, multiple answers arrived in less than 60 minutes being the fastest one achieved in just 4 minutes. The faster student proposed a custom filter to reveal accurately the credentials exposed in a traffic sample using the Tshark analyzer.

A relevant insight is exhibited in the challenge 7, where students proposed different alternatives to prove an injection flaw. The results arrived between one and two days after the announcement. Then, the performance was ratified in the challenge 9 where XSS injection achieved the fastest answer in roughly 5 hours, and the average response period remained in less than one day.

The final challenge started by traversing directories to analyze the composition of the targeted web site. The students selected different techniques to discover and enumerate the components providing their answers between 3 days and until 2 weeks after the announcement. The answers were properly organized in reports to be extended further by the next findings. In this experience the students spontaneously started to collaborate and share insights between them since the case was emulating a blind scenario with limited information. The students were able to understand the authentication logic and the interactions involved. Certainly, the backend provided debugging data intentionally that is avoided in a productive environment. Nonetheless, a student necessarily requires a thorough analysis and critical thinking about all components to prove a break-in. The students proposed their own attack strategy, altering the existing conditions of the website to induce a forged token from the client side. The findings were provided on average one week after the announcement and according to the form and deadline defined.

5.3. FINAL RANKING ANALYSIS

The final ranking sorts the performance of the students based on the number of challenges solved and the average response period, also referred as Tiebreak.

The following table displays the final ranking of the students:

RK	Name	Points	Tiebreak	RK	Name	Points	Tiebreak
1	Student 1	20	75117	11	Student 11	4	10396
2	Student 2	18	107946	12	Student 12	4	50700
3	Student 3	18	111500	13	Student 13	3	320109
4	Student 4	17	21176	14	Student 14	2	500
5	Student 5	16	91795	15	Student 15	2	10849
6	Student 6	9	121421	16	Student 16	2	19630
7	Student 7	6	23563	17	Student 17	0	0
8	Student 8	6	314998	18	Student 18	0	0
9	Student 9	6	1483994	19	Student 19	0	0
10	Student 10	5	31995	20	Student 20	0	0

Figure 8: Table of Final Ranking.

By breaking down the number of student participants, the proportion of female students reached 15%, and male students 85%. Equivalently, the ICT track participation obtained 85%, and the general school participation represented 15%. In terms of levels, the students registered in 2016 were 20%, those from 2017 were 20%, and those from 2018 represented 45%.

Considering a top ten sample, the students registered in 2018 were 60% while 20% belonged to the previous year, 2017 and the same for 2016. Consequently, the ranking revealed that the most enthusiastic students were those who started their journey in ICT school one year before the debut of Hack with Turku, and that it is an unexpected insight as the preliminary assumption was that the most experienced students would reach the higher scores.

In terms of attendance, there were 3 formal requests for dropping the course. The main reason expressed was the time involved, an aspect that is variable considering the flexible curriculum and the different course load of a student in the period. It is worth noting, that the attendance was not measured, yet the average attendance per session was roughly 70%. Also, some students decided to join the course to understand how to protect themselves from hacking: They were highly committed with their attendance yet less interested in the competition.

Finally, during the closing day of the first edition of Hack with Turku, all the students were recognized for their committed participation, and the top five students were rewarded by the sponsor Lähitapiola to attend the Hack Day 2019 event to keep sharpening their skills in a fun and constructive way.

6. CONCLUSIONS AND FUTURE WORK

Cybersecurity is a multidisciplinary domain that approaches penetration testing as one countermeasure to prevent an attack and is understood as offensive security. A defensive security under ubiquitous environments is challenging and requires a deep understanding of the adversarial mindset to design effective controls.

To reconcile the diversity of knowledge and the student's motivation, the progressive learning philosophy is adopted. In particular, the scaffolding and constructivist methods directly influenced the designing of the challenges, hands-on lessons and lectures. In the analysis, the results demonstrated that more determinant than the background was the motivation of the students. By being immersed in relevant cybersecurity challenges, the students have a more realistic view about the problems that we face and how they can help to solve them.

The infrastructure supporting the teaching and learning process is based on cloud computing to provide the capacity and flexibility needed in progressive learning. A cloud-based attack surface brings to the students the opportunity to practice at his or her own pace granting them mobility and having minimal impact on their own computers. The infrastructure is adaptive, and the settings can be altered remotely by means of scripts. Through the progressive testing and refactoring of the scripts, we can realize that the challenges are essential to direct the effort of the ICT operation. A model of challenges defines the purpose, scope, the techniques and tools of the learning experiences. From this viewpoint, it is feasible to reconcile the curiosity and critical thinking of the students under a legit environment prepared for penetration testing.

As a future work should be interesting to explore and compare a recursive logic to deploy the scripts against the iterative loops. Also, it should be of great interest to apply this progressive teaching and learning methods in different contexts such as the Internet of Things emphasizing C programming language.

7. REFERENCES

[1] Kerttulin lukio, *Kerttulin Lukion Opetussuunnitelma 1.8.2016 Alkaen Versio 11.9.2017*. Accessed on: Dec. 14, 2018. [Online]. Available: http://www.turku.fi/sites/default/files/atoms/files//kerttulin_lukion_ops_2016_11092017.pdf

[2] Kerttulin lukio, *Kerttulin lukio Opinto-Opas 2018-2019*. Accessed on: Jan. 14, 2019. [Online]. Available: http://www.turku.fi/kerttulin-lukio/opiskelijalle/opinto-opas-ja-opetussuunnitelma

[3] A. Gordon, *Official (ISC)2 Guide to the CISSP CBK, 4th edition*. Boca Raton, Florida, United States of America: CRC Press, Taylor & Francis Group, 2015.

[4] D. Farmer and W. Venema, *Improving the Security of Your Site by Breaking into It.* Usenet posting, 1993. Accessed on: Jan. 14, 2019. [Online]. Available: http://www.porcupine.org/satan/admin-guide-to-cracking.html

[5] D. Farmer and W. Venema, *Being Prepared for Intrusion*. Dr. Dobb's Journal, vol.26, issue 4, pp.78-85, 2001. Accessed on: Jan. 14, 2019. [Online]. Available: http://www.drdobbs.com/being-prepared-for-intrusion/184404565

[6] J. Grossman, R. Hansen, P. Petkov, A. Rager, and S. Fogie. *XSS Attacks: Cross Site Scripting Exploits and Defense*. Massachusetts, United States of America: Syngress, Elsevier, 2007.

[7] K. Scarfone, M. Souppaya, A.Cody, and A. Orebaugh. *Technical Guide to Information Security Testing and Assessment*. The National Institute of Standards and Technology, Special Publication 800-115. Maryland, United States of America: NIST, 2008.

[8] P. Herzog, *The Open Source Security Testing Methodology Manual: OSSTMM 3*. Institute for Security and Open Methodologies, ISECOM, 2010. Accessed on: Feb. 11, 2019. [Online]. Available: http://www.isecom.org/mirror/OSSTMM.3.pdf

[9] K. Orrey, M. Byrne, A. Doraiswamy, L. Lawson, and N. Ouchn. *Penetration Test Framework (PTF) v0.59, 2014*. Accessed on: Feb. 11, 2019. [Online]. Available: http://www.vulnerabilityassessment.co.uk/Penetration%20Test.html

[10] PTES Group, *Penetration Testing Execution Standard*, 2009. Accessed on: Feb. 11, 2019. [Online]. Available: http://www.pentest-standard.org/index.php/FAQ

[11] International Organization for Standardization, *Software and Systems Engineering - Software Testing*. ISO/IEC/IEEE 29119, 2013.

[12] Primer Congreso Internacional DuocUC. *La Educación Técnico Profesional al Servicio de Chile - Rol y Responsabilidad Social*. Fundación Santillana. Santiago, Chile, 2016.

[13] A. Kauramäki, *Building Virtual Penetration Testing Environment.* Master's thesis, University of Turku, Turku, Finland, 2017.

[14] K. Seong-Won and L. Youngjun, *A Study of Educational Method Using APP Inventor for Elementary Computing Education*. Journal of Theoretical and Applied Information Technology, JATIT vol.95, n. 18, 2005.

[15] S. Barzilai and I. Blau, *Scaffolding game-based learning: Impact on learning achievements, perceived learning, and game experiences*. Science Direct, Computers & Education Journal, vol. 70, p.65, 2014.

[16] W. J. Matthews, *Constructivism in the classroom: Epistemology, history, and empirical evidence*. Teacher Education Quarterly Vol.30, n. 3 p.51-64, 2003. Accessed on: Feb. 18, 2019. [Online]. Available: Education Resources Information Center, U.S. Department, https://eric.ed.gov

[17] D. Alt, *Constructivist Learning and Openness to Diversity and Challenge in Higher Education Environments*. Learning Environments Research Vol.20, n. 1,p. 99-119, 2017. Accessed on: Feb. 18, 2019. [Online]. Available: Education Resources Information Center, U.S. Department, https://eric.ed.gov

[18] National Cyber Security Centre of Finland, *Finnish users' passwords in clear text revealed: TRAFICOM Alert, 2018*. Accessed on: Feb. 18, 2019. [Online]. Available: https://www.kyberturvallisuuskeskus.fi/en/finnish-users-passwords-clear-text-revealed

[19] N. Langley, *Ethical Hacking is Challenging and Lucrative but Training is Expensive*. Computer Weekly, p.44, 2005.

[20] G. Lyon, NMAP: The Network Mapper Security Scanner. Accessed on: Jan. 14, 2019. [Online]. Available: https://insecure.org/

[21] Rapid7, *Metasploit Knowledge Base*. Accessed on: Jan. 14, 2019. [Online]. Available: https://kb.help.rapid7.com/docs

[22] Microsoft, *Microsoft Documentation*. Accessed on: Jan. 14, 2019. [Online]. Available: https://docs.microsoft.com/

[23] Offensive Security, *Kali Linux*, 2018. Accessed on: Jan. 14, 2019. [Online]. Available: https://www.offensive-security.com/

[24] Python Software Foundation, *Python Documentation*. Accessed on: Jan. 14, 2019. [Online]. Available: https://docs.python.org/

[25] Binary1985, *TimeLapse, SMB Timed bruteforcer using metasploit smb_login.* Accessed on: Feb. 18, 2019. [Online]. Available: https://github.com/binary1985/TimeLapse

[26] Greenbone Networks, *Open Vulnerability System OpenVas*, 2017. Accessed on: Jan. 14, 2019. [Online]. Available: http://www.openvas.org/

[27] Internet Engineering Task Force, *The Base16, Base32, and Base64 Data Encodings*. IETF RFC: 4648, 2006.

[28] J.C. Foster, *Sockets, Shellcode, Porting, and Coding: Reverse Engineering Exploits and Tool Coding for Security Professionals.* Syngress, Elsevier, 2005.

[29] Iteong, *Reverse-Shell: Reverse Shell with Python 3*. Accessed on: Feb. 18, 2019. [Online]. Available: https://github.com/iteong/reverse-shell

[30] Swisskyrepo, *Payloads All the Things: A list of useful payloads.* Accessed on: Feb. 18, 2019. [Online]. Available: https://github.com/swisskyrepo/PayloadsAllTheThings

[31] Internet Engineering Task Force. *An Ethernet Address Resolution Protocol*. IETF RFC: 826, 1982.

[32] G. Combs. *Wireshark: Open Source Network Protocol Analyzer*. Accessed on: Feb. 18, 2019. [Online]. Available: https://www.wireshark.org

[33] Programcreek, *Python scapy.all.ARP Examples*. Accessed on: Feb. 18, 2019. [Online]. Available: https://www.programcreek.com/python/example/103599/scapy.all.ARP

[34] Python Software Foundation, *Python Standard Library: Module Simple HTTP Server*. Accessed on: Feb. 18, 2019. [Online]. Available: https://docs.python.org

[35] OpenSSL Software Foundation, *Cryptography and SSL/TLS Toolkit*, 2018. Accessed on: Feb. 18, 2019. [Online]. Available: https://www.openssl.org/

[36] The PHP Group, *Hypertext Preprocessor (PHP) Documentation*. Accessed on: Feb. 18, 2019. [Online]. Available: https://www.php.net/docs.php

[37] R. Dawes, *OWASP WebScarab*. Accessed on: Feb. 18, 2019. [Online]. Available: https://github.com/OWASP/OWASP-WebScarab

[38] Portswigger Web Security, *Burp Suite Editions, 2019*. Accessed on: Feb. 18, 2019. [Online]. Available: https://portswigger.net/burp/

[39] A. Kiezun, P.J. Guo, K. Jayaraman and M. D. Ernst, *Automatic Creation of SQL Injection and Cross-Site Scripting Attacks*. ICSE'09, Proceedings of the 31st International Conference on Software Engineering. IEEE Computer Society Washington, DC, United States of America, 2009.

[40] A. Riancho, *W3af: Web Application Attack and Audit Framework*. Feb. 18, 2019. [Online]. Available: https://github.com/andresriancho/w3af

[41] OWASP Foundation, The Open Web Application Security Project Top Ten, 2019. Accessed on: Feb. 18, 2019. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project/

[42] Internet Engineering Task Force, *Date and Time on the Internet: Timestamps.* IETF RFC: 3339, 2002.

[43] J. Faircloth, *Penetration Tester's Open Source Toolkit - 4th Edition*. Massachusetts, United States of America: Syngress, Elsevier, 2016.

[44] Microsoft Corporation, *Microsoft Azure*. Accessed on: Feb. 18, 2019. [Online]. Available: https://azure.microsoft.com/

[45] Internet Engineering Task Force, *The Secure Shell (SSH) Public Key File Format*. IETF RFC: 4716, 2006.

[46] T. Ylonen, P. Turner, K. Scarfone and M. Souppaya, *Security of Interactive and Automated Access Management Using Secure Shell (SSH)*. The National Institute of Standards and Technology Internal Report 7966, p. 50. Maryland, United States of America: NIST, 2015.

[47] GlavSoft, TightVNC Software: Remote Desktop Software. Accessed on: Feb. 18, 2019. [Online]. Available: https://www.tightvnc.com

[48] G. Lyon and D. Fifield, *Mastering the Nmap Scripting Engine.* Presentation given at Defcon and Black Hat USA, 2010. Accessed on: Feb. 18, 2019. [Online]. Available: https://nmap.org/presentations/BHDC10/

[49] H.D. Moore, *Metasploit and Money*. Black Hat DC 2010. Accessed on: Feb. 18, 2019. [Online]. Available: https://hdm.io/

[50] Joint Task Force, *Security and Privacy Controls for Information Systems and Organizations*. The National Institute of Standards and Technology, Special Publication 800-53, Rev.5. Maryland, United States of America: NIST, 2017.

[51] G. Vache Marconato, M. Kaâniche and V. Nicomette, *A Vulnerability Life Cycle-Based Security Modeling and Evaluation Approach*. IEEE, The Computer Journal, vol. 56, issue 4, 2013.

[52] B. Crispo, Network Security course. Department of Information Engineering and Computer Science, DISI. University of Trento, Trento, Italy, 2017.

[53] R. Hauge, Schoolmate, 2004. Accessed on: Feb. 18, 2019. [Online]. Available: https://sourceforge.net/projects/schooolmate

[54] M. Ceccato and P. Tonella, Security Testing course. Department of Information Engineering and Computer Science, DISI. University of Trento, Trento, Italy, 2017.

[55] Moxie Marlinspike, *SSLStrip*. Accessed on: Feb. 18, 2019. [Online]. Available: https://github.com/moxie0/sslstrip

[56] Internet Engineering Task Force, *Hypertext Transfer Protocol -- HTTP/1.1*. IETF RFC: 2616, 1999.

[57] Internet Engineering Task Force, *Hypertext Transfer Protocol -- HTTP/2*. IETF RFC: 7540, 2015.

[58] National Checklist Program, *CIS Apache HTTP Server 2.4 Benchmark 1.3.1 Checklist Details*. The National Institute of Standards and Technology. Maryland, United States of America: NIST, 2017.

[59] Internet Engineering Task Force, *The 'text/html' Media Type*. IETF RFC: 2854, 2000.

[60] OWASP Foundation, The Official OWASP Top 10 Document Repository, 2017. Accessed on: Feb. 18, 2019. [Online]. Available: https://github.com/OWASP/Top10

[61] World Wide Web Consortium, *W3C: Document Object Model (DOM) Level 3 Core Specification*, 2004.

[62] W3schools, JavaScript Tutorial. Accessed on: Feb. 18, 2019. [Online]. Available: https://www.w3schools.com/js/default.asp

[63] M. Howard, D. LeBlanc and J. Viega, *The 24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them*. McGraw-Hill, 2010.

[64] Computer History Museum, *Donald Chamberlin Biography*. Accessed on: Feb. 18, 2019. [Online]. Available: https://www.computerhistory.org/fellowawards/hall/donald-chamberlin/

[65] Common Weakness Enumeration, *CWE-89, Improper Neutralization of Special Elements used in an SQL Command*. Accessed on: Feb. 18, 2019. [Online]. Available: https://cwe.mitre.org

[66] Common Weakness Enumeration, *CWE-79, Failure to Preserve Web Page Structure*. Accessed on: Feb. 18, 2019. [Online]. Available: https://cwe.mitre.org.

[67] Internet Engineering Task Force, *The Web Origin Concept*. IETF RFC: 6454, 2011.

[68] N. Provos, P. Mavrommatis, M. Abu Rajab and F. Monrose, *All Your iFRAMEs Point to Us*. SS'08, Proceedings of the 17th Conference on Security Symposium, pp. 1-15. USENIX Association, Berkeley, CA, United States of America, 2008.

[69] E. Barker and A. Roginsky, *Transitioning the Use of Cryptographic Algorithms and Key Lengths*. The National Institute of Standards and Technology, Special Publication SP 800-131A Rev.2. Maryland, United States of America: NIST, 2018.

[70] Internet Engineering Task Force, *Asymmetric Key Packages*. IETF RFC: 5958, 2010.

8. ANNEX A: CONTENT SCOPE

The following requirements are defined to enable the attack surface on Azure Cloud:

Outcome Name	Content Scope
Lecture: What is all about?	Who am I? What is HWK? When and Where? Who are the speakers? How is scored? Rule of Thumb The attack Surface - Demo
Lecture 1: Knowing your attack surface: SSH protocol	How do we get here? What is SSH? Why SSHv2? What are the encryption options? How is exchanged a shared key? Is a hash value equal to a signature? Signing and Pub key cipher are equal? Authentication methods. References
Hands-on 1 - Knowing your attack surface: SSH and VNC	What is SSH? How SSH works? Authorizing your own public key RSA. Authorizing your own public key PEM. What is VNC? How VNC works? Set a VNC Tunnel over SSH References
Lecture 2: Penetration Testing Methodology.	What is penetration testing? The CIA triad Pen-Test scope Pen-Test methodology Reconnaissance Passive vs Active Recon Vulnerability Analysis Execution Document Findings References
Hands-on 2: Active Reconnaissance: NMAP.	What is NMAP? How NMAP works? Targeted Scanning The Nmap NSE Integrating NMAP and Metasploit. How to prevent scanning? Tips and Hints References
Lecture 3: Metasploit Framework.	What is Kali Linux? Metasploit Framework What is an exploit? Common exploitation process Module Auxiliary Module Exploit Payloads Module Post References
Hands-on 3: Metasploit Recon and Brute Force.	What is Metasploit? How it works? MSF shortcuts What is SMB? A brute force attack. Testing weak passwords. How to prevent fingerprinting and weak passwords?

	Tips and Hints References
Lecture 4: Vulnerability Analysis.	What is a vulnerability? Zero day attack What is CVE? CVE Sample What is Openvas? Openvas elements How it works? GSA interactions Reports Quality of detection References
Hands-on 4: Vulnerability Analysis: OpenVas.	What is Openvas? How it works? Tasks – Port Lists Tasks - Targets Tasks - ScanConfig Tasks – New Task Tasks – Credentials How to prevent illegit assessments? Tips and Hints References
Lecture 5: Attack Methodology and System Appropriation	Attack methodology overview Threat agent Attack method cycle Target acquisition and analysis controls Target access and appropriation controls Bots and Botnets A Shellcode I A Shellcode II References
Hands-on 5: Metasploit Payloads.	What is a payload? How it works? Enumeration recap. A reverse shell Using meterpreter Using msfvenom Target Appropriation How to prevent SMB abuse? Tips and Hints References
Lecture 6: OSI Model and Man in the Middle Attack	What is a MIM attack? OSI Model Physical Layer Data Link Layer Network Layer Transport Layer I/II Session Layer Presentation Layer Application Layer I/II Hijacking and MITM References
Hands-on 6: TShark and ARP Poisoning.	What is Tshark? Capture Filters Display Filters What is a Mac address? What is ARP? ARP Poisoning I ARP Poisoning II How to prevent? Tips and Hints References
Lecture 7: Protocols SSL/TLS.	What is SSL/TLS? SSL Architecture

	How it works? Handshake protocol Certificate content Certificate Authority Certificate hierarchy Public Key infrastructure References
Hands-on 7: SSL OpenSSL and SSLstrip attack.	What is OpenSSL? Our own CA Webserver CRT Apache SSL GET and POST What is SSLStrip? How it works? I/II How to prevent? References
Lecture 8: OWASP 10 Overview.	What is OWASP? I/III A1. Injection A2. B. Authentication A3. S. Data Exposure A4. XXE A5. B. Access Control A6. S. Misconfiguration A7. XSS A8. I. Deserialization A9. C. Known Vulns. A10.I. Logging & M. References
Hands-on 8: Webscarab and Burpsuite.	What is a proxy? What is Webscarab? Using Webscarab I/III What is Burpsuite? Using Burpsuite How to prevent? Tips and Hints References
Lecture 9: HTTP Web Server Hardening.	HTTP Protocol HTTP URI Parameter HTTP Request Msg. HTTP Response Msg. HTTP Methods and Status Codes HTTP Cookies Session Hijacking attack CRIME attack What is a hardening? Apache Checklist Sample References
Hands-on 9: Directory Listing and Path Traversal.	Directory Listing What is Apache? Apache Directive What is Burpsuite? Using Burpsuite How to prevent? Tips and Hints References
Lecture 10: HTTP/HTML and Injections.	HTTP Connection Internet Message Syntax HTTP Message Format HTTP Content Negotiation HTTP Transfer Coding HTML HTML Evolution HTML Elements Security Concerns Redirect Injection Command Injection

	References
Hands-on 10: Redirect and Command Injections.	URL Redirection Redirect Injection I Redirect Injection II Command Injection I Command Injection II Command Injection III Command Injection IV Command Injection V How to prevent? Tips and Hints References
Lecture 11: HTML Elements and Injections.	HTML Summary HTML Principal Elements HTML DOM HTML General Elements HTML Hyperlinks HTML Image & Table HTML Form JavaScript Overview JavaScript Capability OS command Injection References
Hands-on 11: OS Command Injections.	HTML DOM Usage JavaScript Function Password Validation External Function Adding Elements I Adding Elements II How to control JS? Tips and Hints References
Lecture 12: SQL and Injections.	What is SQL? Database relations Database relational elements SQL DB components SQL statements samples Object-Oriented model Security Concerns References
Hands-on 12: SQL Injections and W3af.	What is SQL Injection? String Concatenation Unvalidated Parameter Numeric SQL Injection I Sanitization Sample I Numeric SQL injection II Sanitization Sample II What is W3af? Using W3af Scanner Tips and Hints References
Lecture 13: OWASP Cross Site Scripting	Verizon Trends What is XSS? Reflected XSS Attack Same Origin Policy Persistent XSS Attack AJAX Request DOM XSS Attack Potential Mitigations References
Hands-on 13: XSS and OWASP ZAP I.	Reflected XSS Attack Sanitization Sample Persistent XSS Attack Sanitization Sample What is OWASP ZAP?

	Using OWASP ZAP Tips and Hints References
Lecture 14: OWASP XSS and CORS	Cross Origin Request CORS Resource Sharing CORS Headers CORS Headers Sample XHR request Sample Fetch request Sample Security Concerns References
Hands-on 14: XSS and OWASP ZAP II.	DOM XSS Attack I DOM XSS Attack II What is a CSRF attack? XSS vs. CSRF Script Obfuscation OWASP ZAP and Seclists Tips and Hints References
Lecture 15: Documenting Findings.	What is a Report of Findings? Privacy of Findings Report Structure Report Delivery Report Remediations Findings Analysis Proof of Concepts References
Hands-on 15: Protecting your Report: OpenSSL.	OpenSSL Elements OpenSSL GenpKey I OpenSSL GenpKey II OpenSSL Encrypting File OpenSSL Signing File Tips and Hints References
Lecture 16: Software Development Life Cycle (SDLC).	SDLC Overview SDLC Phases I SDLC Phases II SDLC Phases III Testing Techniques Techniques Comparison What is Unit Testing? References
Hands-on 16: Python Unittest and Selenium.	Python Unit Testing Unittest Sample Unittest Main Methods UrlLib2 Sample Unittest Main Assertions Selenium Sample I Selenium Sample II References

9. ANNEX B: REQUIREMENTS FOR MULTIPLE INSTANCES

The infrastructure design is leveraged by the following specific requirements:

Ref.	Category	General Requirements
HN	Hardware and Networking	An infrastructure as a service growing vertically for learning purposes.
SA	Software and Applications	Multiple services providing distinct levels of robustness.
SE	Security	Set of countermeasures prioritizing access control and availability.

Ref.	Specific Requirements
HN.1.1	A virtual network must allow traffic between multiple segments. One student instance has associated one virtual network covering the segment 10.1.1.0/24, that also enable three subnets; subnet1:10.1.1.8/29, subnet2:10.1.1.16/29 and subnet3:10.1.1.24/29.
HN.1.2	The five systems inside one student instance can exchange data from different subnets. All the subnets traffic is forwarded between each other according to the following settings: The system hack-kali has size std. B2s (2vcpus,4GB Ram), IP 10.1.1.14 at subnet1 10.1.1.8/29. The system hack-win has size std. B1ms (1vcpu,2GB Ram), IP 10.1.1.30 at subnet3 10.1.1.24/29. The system hack-debian has size std. B1s (1vcpu,1GB Ram), IP 10.1.1.22 at subnet2 10.1.1.16/29. The system hack-centos has size std. B1s (1vcpu,1GB Ram), IP 10.1.1.21 at subnet2 10.1.1.16/29. The system hack-ubuntu has size std. B1s (1vcpu,1GB Ram), IP 10.1.1.29 at subnet3 10.1.1.24/29
HN.1.3	There is a single public IP allowing inbound SSH requests and it has limited access out to Internet. Only the system hack-kali has associated a virtual public IP in their single interface. There is a DNS name associated to the public IP e.g. hack-kali-op1-ip.westeurope.cloudapp.azure.com. The hack-kali machine is reachable only for secure remote login through SSH protocol and only accesses Internet temporarily for preparedness tasks. Then, the student must access only the replica application system on Internet, any other traffic should be dropped.
SA.1.1	The hack-kali system performs as VNC server allowing graphical desktop sessions through a secure tunnel. The system hack-kali version Kali-Rolling performs the package tightvncserver v1.3.9-9. The service Xtightvnc allows multiple graphical connections that are always authenticated. The first connection uses by default the port TCP/5901 and so on. The VNC connection is tunnelled through SSH locally in the student workstation to protect the transmission started by the free version of VncViewer. To exemplify, the tunnel follows the syntax, $ssh user@<hostname> -L <source-port>:localhost:<destination-port>
SA.1.2	The operating systems running inside the student instance have specific versions and patches for learning purpose avoiding any hardening criteria. The systems adhere to different levels of robustness for learning purposes. The system hack-win is a Domain Controller on v2k8R2SP1 with SMBv1 enabled and lax security password policies. The procedure is supported by scripts. The system hack-debian v8.11 Jessie is a weak Web site running on top of PHPv5x, Apache v2x and MySqlv5x. The system hack-ubuntu v14.04 Trusty is a server exchanging permanent clear data with hack-centos v7.5
SE.1.1	The credentials are differentiated from high and low privileges for mentor and student correspondingly. The Azure platform considers a mentor account with contributor role able to restart and rebuild systems. The systems hack-kali, hack-win, hack-debian, hack-centos and hack-ubuntu have two accounts mentor and hack. A sudo security policy

	is enabled for mentor to avoid password requests (e.g. mentor ALL=(ALL) NOPASSWD:ALL). The sudo security policy for the hack account brings specific access to the binaries required (e.g. %turku ALL=(ALL) NOPASSWD:/usr/bin/nmap). The creation of hack accounts will be supported by scripts.
SE.1.2	The passwords for students are personal and confidential. The password will be sent through secure email to each student, having each of them a different password. The total number of student instances is 22.
SE.1.3	The network traffic is controlled by a network access control embedded on Azure cloud. The cloud provides network security groups that perform similarly to a host firewalls associated to each network interface. All systems are associated to a common network security group criterion. The rules are set for inbound and outbound traffic, as follows: The basic rules applied for inbound traffic are the following: -AllowInSSH port 22 protocol TCP source any destination hack-kali action allow. -AllowVnetInBound port any protocol any source VirtualNetwork destination VirtualNetwork action allow. -DenyAllInBound port any protocol any source any destination any action deny. The basic rules applied for outbound traffic are the following: -AllowInternetOutBound port any protocol any source hack-kali destination any action allow (preparedness). -AllowInternetOutBound port any protocol any source hack-kali destination any action deny (go live). -AllowREPLICAOutBound port any protocol any source hack-kali destination REPLICA action allow (go live). -AllowVnetOutBound port any protocol any source VirtualNetwork destination VirtualNetwork action allow. -DenyAllOutBound port any protocol any source any destination any action deny.

10. ANNEX C: HOST FIREWALL RULES

The following set of rules belongs to netfilter host firewall running on hack-kali system.

```
# security table
*security
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
# nat table
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 10.1.1.8/29 -o eth0 -j MASQUERADE
-A PREROUTING -i eth1 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 8080
COMMIT
# filter table
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -m conntrack --ctstate INVALID -j DROP
-A INPUT -i eth0 -p tcp -m tcp -m multiport --dports 22,4444,5555 -m conntrack --ctstate NEW -
j ACCEPT
-A INPUT -i eth1 -p tcp -m tcp -m multiport --dports 80,443,8080 -m conntrack --ctstate NEW -j
ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A FORWARD -m conntrack --ctstate NEW,RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -m conntrack --ctstate INVALID -j DROP
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -m conntrack --ctstate INVALID -j DROP
-A OUTPUT -d 10.1.1.16/29,10.1.1.24/29 -m conntrack --ctstate NEW,RELATED,ESTABLISHED -j
ACCEPT
-A OUTPUT -o eth0 -d xxx.xxx.xx,xx.xxx.xxx.xx,xx.xx.xx.xx,xxx.xx.xx.xx -p udp -m udp --dport
53 -m conntrack --ctstate NEW -j ACCEPT
-A OUTPUT -o eth0 -d xxx.xxx.xx,xx.xxx.xxx.xx,xx.xx.xx.xx,xxx.xx.xx.xx-p tcp -m tcp --dport 53
-m conntrack --ctstate NEW -j ACCEPT
-A OUTPUT -o eth0 -d xxx.xxx.xx.xxx -p tcp -m tcp -m multiport --dports 80,443 -m conntrack --
ctstate NEW -j ACCEPT
-A OUTPUT -o eth1 -p tcp -m tcp -m multiport --dports 80,443,8080 -m conntrack --ctstate NEW -
j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp -m multiport --dports 22,53,80,443,873,8080 -m conntrack --
ctstate NEW -j DROP
-A OUTPUT -o eth0 -p udp -m udp -m multiport --dports 53,873 -m conntrack --ctstate NEW -j
DROP
-A OUTPUT -d 10.1.1.16/29,10.1.1.24/29 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A OUTPUT -d 10.1.1.16/29,10.1.1.24/29 -p icmp -m icmp --icmp-type 0 -j ACCEPT
COMMIT
# Completed
```