ANALYSIS AND EVALUATION OF CAPTURE THE FLAG CHALLENGES IN SECURE MOBILE APPLICATION DEVELOPMENT

Stylianos Karagiannis[1, 2], Emmanouil Magkos[1],
George Chalavazis[1] and Maria Nefeli Nikiforos[1]

[1]Department of Informatics, Ionian University, Greece
[2]PDMFC, Portugal

ABSTRACT

Capture the Flag (CTF) challenges are frequently used as cybersecurity learning environments to engage students in cybersecurity education activities and learning, focusing on technical concepts. CTF challenges cover various learning topics. However, they do not always maintain a clear learning outcome. In this paper, we present a systematic approach to study and evaluate CTF challenges, then apply the evaluation methodology in two CTF challenges that relate to the development of secure mobile applications. For this proof of concept, we used the National Initiative for Cybersecurity Education (NICE) which is a cybersecurity educational framework published by the National Institute of Standards and Technology (NIST). Additional information was used for the evaluation process which included threat, vulnerability, and weakness taxonomies proposed by Open Web Application Security Project® (OWASP) and Mitre Corporation (MITRE). The evaluation methodology could be used to assess and determine the learning outcomes of other existing or upcoming CTF challenges, including though not limited to secure mobile application development.

KEYWORDS

Cybersecurity, Mobile Application Development, Capture the Flag, OWASP, MITRE, NICE Framework

1. INTRODUCTION

Cyber attacks are continuously evolving, while the skills shortage in cybersecurity remains a significant problem [1, 2, 3, 4, 5]. The skills shortage mainly concerns technical concepts of cybersecurity that are not usually covered in the existing curriculum [1]. Therefore, technical lab experience must be provided through realistic scenarios that can be used to educate students and get feedback from them [6, 7]. Capture the Flag (CTF) challenges is a novel approach to engage students in technical concepts and provide a digital environment for them to practice on various cybersecurity and IT topics [8, 9, 10, 11, 12, 13]. CTF challenges usually simulate systems that intentionally contain security flaws and software vulnerabilities for educational purposes [14]. It is therefore a major benefit that CTF challenges can successfully introduce students to technical concepts in a realistic setting by using CTF challenges [15, 16]. Furthermore, the scoring system these challenges maintain can significantly increase students' engagement [17, 18, 19].

There are various CTF challenges; however, their learning outcomes and results are not always clear [20]. Furthermore, not all cybersecurity topics are adequately covered in CTF challenges [21]. As a point of reference, a relevant topic that needs further exploration and is not adequately addressed by CTF challenges is the development of secure mobile applications, a topic that

DOI :10.5121/ijite.2022.11202

evolves due to technological growth and the continuous evolution of cyberattacks [22, 2]. Mobile security has been mentioned in the past research both at the very early stages of CTF development [23, 24, 25] and also recently [26, 27], justifying that this is an active research area. However, CTF challenges were developed to train mostly penetration testers and white-hat hackers, without adapting the concepts to teach IT security to software engineers regarding secure coding [28]. In addition, the total number of challenges that involve the Android operating system or the related applications is generally small, while they usually included on the taxonomy of miscellaneous categories or categories related to reverse engineering [28, 29]. Finally, the learning topics that relate to Android and mobile security holds many challenges, since mobile applications involve complex digital environments that typically include cloud services that extend to the Internet of Things (IOT) [30].

CTF challenges could tackle the aforementioned issues; however, their learning outcomes need to be clear and related to the academic curriculum and educational frameworks to maintain a consistent structure and meet the learning needs. Consequently, an evaluation process could be important to understand the context of CTF challenges and decide whether they provide realistic security scenarios with relation to the curriculum [21, 31, 2, 32]. As a result, a systematic evaluation methodology could be useful and capable to extract the learning outcomes from CTF challenges in a constructed way.

In this research paper, we develop and deploy a systematic evaluation methodology to determine the learning outcomes of CTF challenges. In order to provide a proof of concept, we evaluated CTF challenges focusing on secure mobile application development. The following Research Questions (RQ) have been drawn to conduct the research:

1. RQ1: Is it possible to determine the learning outcomes from existing CTF challenges?
2. RQ2: Do the existing CTF challenges cover adequately the learning topics of secure mobile application development?

To address RQ1, we developed an evaluation method using the Open Web Application Security Project (OWASP) Mobile Top 10 [33], the MITRE Common Weakness Enumeration (CWE) [34] and the National Initiative for Cybersecurity Education (NICE) framework [35, 36]. OWASP also publishes the Mobile Application Security Verification Standard (MASVS) [37], which outlines the security requirements of mobile applications. The above approaches were used to identify if it is possible to extract the learning outcomes from the selected CTF challenges. Finally, the NICE framework, published by the National Institute of Standards and Technology (NIST) [35, 36], was used to indicate the relation of cybersecurity exercises to Work Roles proposed by the framework and to define potential cybersecurity learning benefits. Two CTF challenges were selected, namely, (a) Damn Insecure and Vulnerable App (DIVA) [38], and (b) Extremely Vulnerable Android Labs (EVABS) [39]. Regarding RQ2, the evaluation of the CTF challenges aimed to prove whether DIVA and EVABS can provide their promised learning outcomes.

1.1. Contribution

This research paper presents the benefits of using a systematic evaluation methodology to determine the learning outcomes from CTF challenges. The paper contributes to the following: (a) Identification of the importance of educational frameworks in cybersecurity and popular open taxonomies, (b) Development an evaluation methodology as a systematic approach to determine the learning outcomes from CTF challenges, (c) Evaluation of CTF challenges for secure mobile application development as a proof of concept.

This research validates the possibility to enable existing knowledge to systematically determine the learning outcomes from CTF challenges. Educators can use the proposed methodology to extract the learning outcomes of existing or upcoming CTF challenges, including, though not limited to, secure mobile application development. Furthermore, this paper presents the evaluation results from two CTF challenges (DIVA and EVABS), as a proof of concept of the evaluation methodology. The selected CTF challenges are evaluated on whether they can cover the learning needs regarding secure mobile application development. For doing so, the OWASP Mobile Top 10 Risks, the MITRE Common Weakness Enumeration (CWE) and the NICE framework were used. Finally, a matching of DIVA and EVABS to the work role "Vulnerability Assessment Analyst (Work Role ID: PR-VAM-001)", proposed by the NICE framework, was established.

1.2. Related Work

To the best of the authors' knowledge, no similar research has been conducted that systematically investigates the learning outcomes of CTF challenges in the topics of secure mobile application development. However, researchers have used educational frameworks to evaluate security competitions in the past [40], e.g., by identifying the ten most important factors from the CTF competitions. In terms of research papers that mention DIVA or similar approaches, they use it as a vulnerable application for experimental purposes [41]. In another work [42], the researchers outlined twenty-five known vulnerabilities that cover four areas of Android applications and created a framework as an open-source repository [43]. Finally, a complete security penetration testing was conducted in another research regarding several Android applications, including DIVA and according to the OWASP Mobile 2016 [44].

OWASP Mobile Top 10 [33] has also been used in other works for conducting vulnerability assessments in mobile applications [2, 45, 44]. In their work, the researchers performed a penetration test on DIVA and other vulnerable applications to identify the potential vulnerabilities, to extract a risk analysis, and provide security recommendations according to the risk matrix [44]. Other works analysed mobile security through using different approaches to provide security guidelines from a software development perspective (e.g., [46]). In Gil et al. [46], the NICE framework was used to determine learning outcomes and to identify whether the CTF challenges are capable of covering a specific Work Role. The NICE framework has been used previously to assess laboratories and CTF challenges [47, 48, 49]. For example, the researchers evaluated their exercises by matching them to the Tasks (T) of the NICE framework taxonomy [47]. In another work [50], it was studied whether NICE can be used as a baseline to create questionnaires that measure the level of knowledge improvement. In addition to the above, the NICE taxonomy has been used to evaluate Cyber Ranges and cybersecurity exercises [48, 49]. Although the aforementioned works share some concerns regarding secure mobile application development, they do not provide a structured methodology to examine vulnerable mobile applications by using a specific framework to extract the potential learning outcomes from the various approaches and challenges.

1.3. Structure

Section 2 presents the steps of the proposed methodology. Section 3 regards the execution, analysis and evaluation of the CTF challenges. The Section continues with the identification of vulnerabilities and learning outcomes for each of the challenges and their matching to the NICE framework. The paper concludes in Section 4.

2. METHODOLOGY

The methodology presents the three evaluation steps to systematically determine the learning outcomes from CTF challenges: (a) Lab and Exercise Execution, (b) Vulnerability and threat analysis, (c) Relation of the CTF challenge to an educational framework. In relation to the RQ1, the proposed methodology had to be tested to check its capability to determine the learning outcomes of the CTF Challenges. From this perspective, the evaluation methodology was applied on two CTF challenges that relate to secure mobile application development, specifically DIVA and EVABS.

The vulnerability and threat analysis were carried out to retrieve information on vulnerabilities and match it with common taxonomies from OWASP and MITRE. Then the matching with the NICE framework provided the relation to the academic curriculum and supported the investigation on the learning outcomes of the CTF challenges. The methodology steps of the proposed systematic evaluation are summarized in Figure 1.

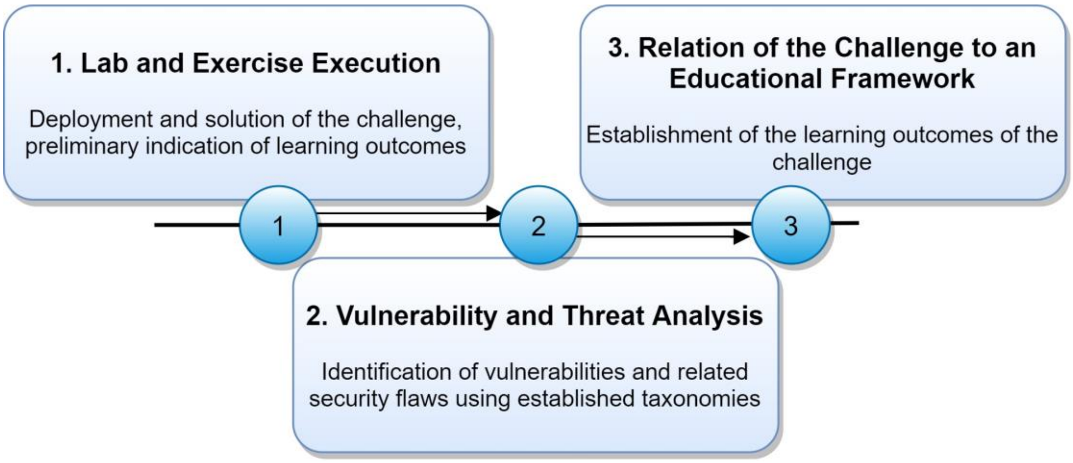
1. Lab and Exercise Execution
Deployment and solution of the challenge, preliminary indication of learning outcomes
3. Relation of the Challenge to an Educational Framework
Establishment of the learning outcomes of the challenge
1
2
3
2. Vulnerability and Threat Analysis
Identification of vulnerabilities and related security flaws using established taxonomies

Figure 1. Methodology steps followed in this research

Methodology Step 01 – Lab and Exercise Execution. This step involves choosing and executing a CTF challenge. It is usually considered a good practise to solve the challenges in order to understand completely the scope of the exercises. During the execution, a first indication of the learning outcomes of the challenge can be identified. As a proof of concept and to test the proposed methodology, the following CTF challenges were evaluated: (a) Damn Insecure and Vulnerable App (DIVA) [38], and (b) Extremely Vulnerable Android Labs (EVABS) [39]. DIVA and EVABS maintain two vulnerable Android applications in the form of Android Package Kits (APK) and CTF challenges to engage trainees. DIVA and EVABS are a combination of virtual laboratories and CTF challenges.

DIVA and EVABS are open-source Android applications that are intentionally vulnerable to act as learning platforms for security on Android applications. The main objective of DIVA and EVABS is to introduce beginners with very limited or zero knowledge to major and common real-world Android application vulnerabilities. The challenges follow a CTF-like approach, resulting in a gamified version of efficient learning. The selected challenges have been mentioned elsewhere [41] about their capability as learning tools that involve software and code vulnerabilities to cover topics such as insecure logging, insecure data storage, input validation errors, and access control issues. Along with other benefits, the aforementioned approaches have the capability to introduce students effectively to technical details and provide a virtual lab to engage to the Android ecosystem. The scope of the evaluation of DIVA and EVABS was to discover whether such approaches can procure the expected learning outcomes. The evaluation also presents whether such approaches can provide exercises realistic enough for students to familiarize with cybersecurity concepts in this topic [51]. Both DIVA and EVABS are some of the most popular in the literature [41], while their main idea is similar to other popular approaches, such as OWASP WebGoat and Damn Vulnerable Web Application (DVWA).

Methodology Step 02 – Vulnerability and threat analysis. A vulnerability assessment is performed to validate that the presented or hidden vulnerabilities in the challenge exist and that they can be determined in an automated way. This step also indicates whether the challenge is realistic enough. To instantiate the vulnerability assessment, MobSF [52] was used to enumerate and analyse the vulnerabilities of DIVA and EVABS. The evaluation provided information on the Top 10 Risks [33], OWASP Mobile Security Testing Guide (MSTG) [53] and MASVS [37], and a relation to MITRE CWE [30]. The above taxonomies and lists were selected since both OWASP and CWE maintain openness, high-maturity and are constantly updated by the community. A detailed description of the taxonomies that were used from the results of the vulnerability assessment is provided below.

- **OWASP Mobile Top 10 Risks.** OWASP Mobile Top 10 [33] is a collaborative effort between OWASP and the European Network and Information Security Agency (ENISA) to provide mobile security controls and identify the top risks. The exercises were matched to the Top 10 OWASP Mobile Risks to identify and document the security flaws that exist in each of the challenges.
- **OWASP MSTG and MASVS**. The relation to MSTG and MASVS can be helpful, as MSTG consists of a comprehensive manual for mobile application security development, testing, and reverse engineering, and MASVS is a standard for the security of mobile applications. These approaches can be used as reference guides during all phases of mobile application development and testing, providing complete information on security issues. Further educational material can also be retrieved by the aforementioned guides.
- **CWE by MITRE.** A relation to CWE [34] was considered important for the identification of weaknesses and extend the information that we already maintain from the above. CWE is a community-developed list of software and hardware weaknesses that can also be used to better identify and define security flaws in mobile applications.

Methodology Step 03 – Relation of the challenge to an Educational Framework. In this step, the learning outcomes are categorised by type or topic of knowledge according to the selected educational framework. More specifically, the learning outcomes of each of the exercises are identified using the relevance to the taxonomy provided by the selected educational framework. The level between the relationship per exercise and the established learning outcomes is evaluated by using the inputs from the previous methodological step.

The NICE framework from NIST was chosen as the reference educational framework, as it is largely considered as one of the most systematic approaches for structuring cybersecurity education and learning. The NICE framework consists of the following building blocks: (a) Categories, (b) Specialty Areas, and (c) Work Roles, including a taxonomy of capability indicators varying from Knowledge topics to Skills and Abilities that match to specific Work Roles. The KSA taxonomy involves the following capability indicators: (a) Knowledge (K), (b) Skills (S), and (c) Abilities (A), describing the work to be done by the trainees. The KSA taxonomy was later revised to KST - Knowledge (K), Skills (S), and Tasks (T) [54, 55], by merging Skills with Abilities and creating the capability indicator of Tasks (T). However, the KSA taxonomy was broadly known in recent years, and, therefore, the framework still maintains the old version. Consequently, in this paper we establish a relationship between the CTF challenges and NICE’s KSA taxonomy. Then we identify, the corresponding Work Role “Vulnerability Assessment Analyst (Work Role ID: PRVAM-001)” and evaluate the exercise using the capability indicators (KSA taxonomy). The level of relation indicates whether the challenges address directly or indirectly the capability indicators (low, average, and high relation). The low score on the relation is indicated on whether the challenges host the appropriate digital environment that could support additional capability indicators.

3. Analysis and Evaluation of DIVA and EVABS

In this section, the execution, analysis and evaluation of DIVA and EVABS are presented. According to the proposed methodology (described in Section 2) and after the execution of the challenges, the vulnerabilities and security flaws of the mobile applications provided by DIVA and EVABS were determined, then the challenges were related to the NICE framework.

3.1. Lab and Exercise Execution

DIVA consists of thirteen different challenges, included in five distinct categories (from DV01 to DV05), while EVABS consists of twelve different challenges. The purpose of DIVA and EVABS is to engage students to solve the challenges, use software toolkits to exploit vulnerabilities, and learn along the process. By doing so, students are supposed to better understand the risk of vulnerabilities and security flaws. After deploying and executing the CTF challenges, an Android Manifest Analysis (AMA) was conducted on the APK provided by DIVA and EVABS. The results from this analysis provide an initial understanding of the potential learning outcomes (Table 1).

Table 1. Results retrieved from the Android Manifest Analysis (AMA).

Security Flaws	**Learning outcomes**	**Risk**
Debugging is enabled	Engage into debugging and reverse engineering by hooking a debugger. Dump information and access the debugging helper classes [android:debuggable = true].	high
Activities are not protected	Check for unprotected activities that share data with other applications of the device leaving the data accessible to any other activity running on the device [android:exported = true].	high
Application data can be extracted	Extract and back up application data through ADB and USB connection [android:allowBackup = true].	warning
Content provider is not protected	Discover a content provider that is shared with other applications, leaving the data accessible to any other application [android:exported = true].	high

The aforementioned security flaws are frequent when developing a mobile application, as developers might forget to configure properly and secure their application. Students can engage and begin to understand fundamental concepts of mobile applications. Furthermore, the AMA provides information on the security fixes (e.g. change the android:debuggable to false). Finally, the security impact and the security risk of such configurations are also mentioned in the AMA report (Table 1).

3.2. Vulnerability and Threat Analysis

The vulnerability analysis with MobSF provided a better understanding on the security issues that DIVA and EVABS introduce. Students are invited to engage to security issues related to improper log storage, hard-coding issues, insufficient encryption, and issues regarding access rights, among others. EVABS provides an application as a CTF and presents twelve different challenges (Table 2).

EVABS manages to cover eight topics from the OWASP Top 10 (80 % coverage). This is a good result, considering that EVABS can address most of the OWASP Mobile Top 10 security risks. The vulnerabilities are not many; however, by using the details provided in Table 2, it is possible to introduce extra educational material or sub-tasks to better cover the related learning topics. For instance, more information can be provided on the mitigation actions or best practices on secure coding.

Table 2. Challenges presented in EVABS.

EVABS Challenges	Tasks provided from the Challenges
EV01/Debug Me	Debugging mode is enabled, resulting in storing unwanted log files [CWE532: Insertion of Sensitive Information into Log File, CWE-276: Incorrect Default Permissions, OWASP M10: Extraneous Functionality].
EV02/File Access	Privileges are not configured, and the malicious user can access extra files [OWASP M02: Insecure Data Storage].
EV03/Strings	Understand extraneous functionality and extract stored information from variables in the code [OWASP M10: Extraneous Functionality].
EV04/Resources	Discover the software stack of an Android application and how to access it [OWASP M02: Insecure Data Storage].
EV05/Shares and Prefs	Discover and change the common preferences of an application by changing the key values [OWASP M02: Insecure Data Storage].
EV06/DB Leak	Insecure data storage in a mini database allowing to extract credentials in plain-text [OWASP M01: Improper Platform Usage / OWASP MASVS: MSTG-STORAGE-14].
EV07/Export	Insecure data storage by triggering a process activity internally in code allows exporting data [OWASP M09: Reverse Engineering, M06: Insecure Authorization, M07: Poor Code Quality].
EV08/Decode	Encrypt and decrypt the hard-coded passwords. Hard-coded passwords, though encoded in BASE64 instead of plain-text [OWASP M05: Insufficient Cryptography].
EV09/Smali Injection	Use the Apktool a tool for reverse engineering to access Smali code and repackage the application [OWASP M09: Reverse Engineering, M07: Poor Code Quality].
EV10/Interception	Intercept the network traffic between the application and the server [OWASP M03: Insecure Communication].
EV11/Custom Access	Test various inputs to acquire access to restricted areas [OWASP M09: Reverse Engineering, OWASP MASVS: MSTGSTORAGE-14].
EV12/Instrument	Use Frida as a software to edit the files of any application [OWASP M10: Extraneous Functionality, M07: Poor Code Quality].

DIVA provides a vulnerable application as a CTF challenge and maintains five different challenge categories, including overall thirteen challenges. These challenges provide an opportunity to understand some of the most popular security flaws that exist in mobile applications. The relation of CTF challenges to OWASP Mobile Top 10, MASVS and MSTG (Table 3) provides a guide for students to learn more about these vulnerabilities. DIVA as a CTF does not provide related information; however, educators can combine and use data from documentation provided by MSTG [53], to expand their knowledge.

DIVA manages 40% coverage from the OWASP Mobile Top 10, which is quite low, meaning that the scope of the challenges could be limited. EVABS provided a better coverage than DIVA in terms of the OWASP Mobile Top 10 Risks. EVABS maintains 12 different challenges that extend to a further scope than DIVA. However, the scope can be extended in case more challenges or subtasks are provided along with supportive educational material. DIVA challenges might not cover the OWASP Mobile Top 10, though they can be good for students to practice and engage on fundamental topics of secure mobile application development.

Table 3. DIVA challenges and the relationship with OWASP to discover learning outcomes.

DIVA Challenges	**Tasks provided in the Challenges**
DV01/Insecure Logging	Discover files and logs that include private information entered by the users and placed in the device storage. Retrieve the history of input submissions on the device by retrieving the related logs. Debug mode is enabled, resulting in storing unwanted log files [OWASP MASVS: MSTG-STORAGE-3].
DV02/Hardcoded Issues	Extract credentials, stored keys or passwords that are hard-coded by the developers. Credentials for authentication processes of applications are stored within the code [OWASP M09: Reverse Engineering, OWASP MASVS: MSTG-STORAGE-14].
DV03/Insecure Data Storage	Discover whether confidential information is insecurely stored within the application or device. Private information is stored in plain-text [OWASP M02: Insecure Data Storage, OWASP MASVS: MSTG-STORAGE2, MSTG-STORAGE-3].
DV04/Input Validation Issues	Access sensitive information on the device without knowing the credentials, interrupting the regular functionality of the application. Topics including Structured Query Language (SQL) injection, or Cross Site Scripting (XSS) on both web services and mobile-enabled websites. Lack of input validation procedures, e.g., filtering, enabling SQL injections and buffer overflows [OWASP M07: Client Code Quality].
DV05/Access Control Issues	Bypass the authentication Application Programming Interface (API) without registering. Improperly configured procedure for access control, allowing access to restricted application functionalities [OWASP M04: Insecure Authentication].

To further validate the learning outcomes and scope of DIVA and EVABS, we determined their relation to the OWASP CWE (Table 4). The importance of this process was to examine whether DIVA and EVABS provide similar challenges. EVABS provides two extra vulnerabilities, namely CWE-89: Improper neutralization in SQL Commands (SQL Injection) and CWE-312: Cleartext storage of sensitive information. Considering that DIVA also includes an SQL injection vulnerability, it is clear that CWE was unable to identify this weakness in DIVA. Therefore, it is important to include reports from multiple taxonomies and enumeration lists to validate the results. Using the CWE, it is concluded that DIVA and EVABS provide the opportunity for students to engage to concepts such as SQL injection, log file retrieval, and cryptography, among others. DIVA and EVABS both present the following configuration flaws: (a) Debugging option for the mobile application is still enabled, (b) Activities are not always protected, (c) The application maintains backups, and (d) The content provider is not protected.

The security issues and configuration flaws are fundamental for students to understand the basic concepts of secure mobile development. EVABS maintains more security issues and, and more specifically, CWE-312 which is not covered by DIVA. Other than that, EVABS has a higher number of vulnerable files. In terms of the security score provided by MobSF, DIVA scores 38% and EVABS 45%. From the execution and analysis of the DIVA and EVABS, it seems that the challenges share similar content, except that EVABS tends to extend to more specific topics. The CWE showed that most of the weaknesses are related to the phase of “Architecture and Design”, while the likelihood of their exploitation is scored as high in most cases. The relation to OWASP and to CWE is of equivalent importance.

Table 4. Relation between MITRE CWE and the CTF challenges.

Identified Vulnerabilities	Description of the Vulnerabilities	Risk
CWE-276: Incorrect default permissions	Application can read/ write to external storage. Sensitive information should never be written into a temporary file and must be encrypted. Application creates temporary files (Number of Vulnerable Files: 5 from EVABS and 1 from DIVA).	info
CWE-89: Improper neutralization in SQL Commands (SQL Injection)	Application uses SQLite Database and execute raw SQL queries. Untrusted user input in raw SQL queries can cause SQL Injection. Sensitive information should be encrypted in the database (Number of Vulnerable Files: 1 from EVABS).	warning
CWE-312: Cleartext storage of sensitive information	Files may contain hard-coded sensitive information like usernames, passwords, keys, etc. (Number of Vulnerable Files: 2 from EVABS).	warning
CWE-532: Insertion of sensitive information into log file	The application logs information. Sensitive information should never be logged (Number of Vulnerable Files: 5 from EVABS and 5 from DIVA).	info
CWE-330: Use of insufficiently random values	The Application uses insecure random number generator (Vulnerable Files: 1 from EVABS and 2 from DIVA).	warning

CWE provides the following details: (a) Description of the weakness, (b) Internal CWE relationships, (c) Modes of introduction which present in which phase the vulnerability usually appears, (d) Applicable platforms, (e) Common consequences in terms of the Confidentiality, Integrity and Availability (CIA) triad, (f) Likelihood of the exploit to occur, (g) Demonstrative examples, (h) Relation to Common Vulnerabilities and Exposures (CVE) [56], (i) Potential mitigation actions, (j) Detection methods, and k) Relation to other taxonomies and attack patterns (Common Attack Pattern Enumeration and Classification (CAPEC) [57]). As a result, CWE introduces learning context significant for both educators and students. For students, it is important to read and analyse the information from CWE to better understand the security flaws. The taxonomy provides not only a description but also concrete coding examples and potential mitigation and detection actions. Therefore, the students can engage more to the CTF challenges, and educators can prepare better educational context or enhance it by adding more tasks.

3.3. Relation of the CTF challenges to the NICE Framework

In this Section, the matching between the KSA taxonomy and the virtual laboratories of DIVA and EVABS is performed. During the matching, three categories were formulated. The relationship between the challenges and the NICE framework regards the connection of the challenges with capability indicators of Knowledge (K), Skills (S) and Abilities (A), as presented in Tables 5, 6, and 7.

Regarding the indicator of Knowledge (K), 77 were addressed during the challenges from a total number of 630 capabilities. The indicator of Knowledge (K) is directly related to 28 Knowledge

(K) topics, 23 maintain moderate relation, and a minimum relationship was identified on 26 Knowledge (K) topics (Table 5).

Table 5. Relation between CTF challenges and capability indicator of Knowledge (K).

Relation	Knowledge (K) covered by the Challenges
High	**Fundamental knowledge on operating systems, networks and databases** (K0007, K0397, K0129, K0130, K0609, K0373, K0417, K0023, K0210, K0224), **Vulnerability assessment and offensive security** (K0005, K0070, K0481, K0603, K0342, K0362, K0367, K0536), **Log analysis, forensic research and data extraction** (K0447, K0449, K0535, K0132, K0021).
Moderate	**Network forensics, network analysis** (K0011, K0046, K0058, K0269, K0334, K0339, K0301), **Vulnerability assessment and ethical hacking** (K0009, K0013, K0040, K0147, K0119, K0206, K0624), **Malware analysis** (K0188, K0191, K0392, K0480, K0604), S**ecurity fundamentals, encryption and access management** (K0308, K0019, K0037, K0336), **Intelligence frameworks and platform connectivity** (K0283, K0577), **Application security, auditing and logging** (K0229, K0343, K0363).
Low	**Risk management and risk assessment** (K0002, K0012, K0115, K0165, K0375), Security and Privacy principles (K0004, K0044, K0157), **Malware analysis, reverse engineering and programming structure** (K0259, K0106, K0183, K0068), **Secure coding and encryption schemes** (K0190, K0016, K0018, K0020, K0140), **Fundamental networks, protocols and network access control** (K0001, K0033, K0034, K0061, K0202, K0393, K0418, K0471, K0089).

The challenges directly cover the Knowledge (K) topics related to fundamental concepts of Android OS, the maintained databases that mobile applications might use and engage to vulnerability analysis, log and forensic analysis overall. Other topics are also presented, including a large aspect of Knowledge (K) topics, as shown in Table 5. The challenges provide an opportunity to present the aforementioned matching (Table 5). However, supplementary instructional material could improve the learning curve for students to build their knowledge. It is indicated from the analysis that the Work Role that is mostly related to the challenges is “Vulnerability Assessment Analyst (Work Role ID: PR-VAM-001)” with a coverage of 51%. This means that the challenges managed to cover half the Work Role in terms of the capability indicator of Knowledge (K).

Table 6. Relation between CTF challenges and capability indicator of Skills (S).

Relation	Skills (S) covered by the Challenges
High	**Vulnerability assessment and system analysis** (S0001, S0066, S0078, S0167, S0335, S0006, S0205, S009), **Network packet analysis** (S0081), **Risk assessment and applicability of security principles** (S0171, S0367), **Offensive security** (S0051, S0270, S0293, S0044).
Moderate	**Malware analysis and reporting, application code analysis and auditing** (S0003, S0120, S0131, S0137), **Forensics, log collection and security controls configuration** (S0221, S0292, S0034, S0332, S0340, S0063, S0183).
Low	**Apply security and privacy principles** (S0006, S0205), **Vulnerability scanning and system analysis** (S0242, S0248).

Of the Skills (S) capacity indicator, 30 were addressed from a total number of 374. Of these, 15 maintain a high relationship, 11 maintain a moderate relationship, and 4 were identified to have a low relationship with the challenges (Table 6). In relation to the Work Role Vulnerability

Assessment Analyst (Work Role ID: PR-VAM-001) the coverage is sufficient (75%). More types of vulnerabilities could be used according to the OWASP Top 10 and take the NICE framework as a reference to upgrade the approaches if necessary to involve other Work Roles. This result indicates that the challenges provide a sufficient introduction to the tasks, thus enabling the acquisition of Skills (S) on the specific Work Role, while including other topics as well.

Regarding the capability indicator of Abilities (A), 19 Abilities (A) of a total of 176 were identified, 8 Abilities (A) with a moderate relation and 4 Abilities (A) with a low relation (Table 7).

Table 7. Relation between CTF challenges and capability indicator of Abilities (A).

Relation	Abilities (A) covered by the Challenges
High	Vulnerability assessment and code auditing (A0001, A0015, A0036, A0092, A0093), Malware Analysis (A0010), Command Line Interface (CLI) and system observation (A0058, A0097).
Moderate	Critical thinking and code logic (A0044, A0106, A0107).
Low	Evaluation and understanding of objectives (A0040, A0108, A0155), Networks and traffic analysis (A0055, A0065).

The challenges succeeded in addressing the capability indicator of Abilities (A) at 100% covering the Work Role of Vulnerability Assessment Analyst (Work Role ID: PR-VAM-001). However, the number of Abilities (A) of the NICE framework with respect to this Work Role were only 4, which is not sufficient, while sometimes the Abilities (A) overlap with the capability indicator of Skills (S). Therefore, even if the challenges fulfil all the Abilities (A) of the specific Work Role, the result is still not convincing enough. NIST already addressed this issue by publishing updates on the framework [54, 55] and revising the KSA taxonomy to Knowledge, Skills and Tasks (KST), through combining the capability indicators of Abilities (A) and Skills (S).

3.4. Evaluation Remarks and Coverage of OWASP, CWE and NICE

As presented in the aforementioned subsections, it is possible to maintain a systematic approach and use the proposed evaluation methodology to determine the learning outcomes of the CTF challenges (RQ1). This subsection analyses the results from the evaluation in relation to RQ2. More specifically, the evaluation shows that DIVA and EVABS were able to cover the learning topics of secure mobile application development. Furthermore, the results from the relation between the CTF challenges and the OWASP Mobile Top 10 Risks indicates that the challenges are presented realistically.

Figure 2 presents the results of the evaluation in terms of coverage among challenges, the OWASP Mobile 10 Risks, MITRE CWE and the NICE framework. DIVA and EVABS provide the opportunity for students to engage to secure mobile application development, mainly regarding the Work Role of “Vulnerability Assessment Analyst (Work Role ID: PR-VAM-001)” proposed by the NICE framework. Additional subtasks on the challenges are required for the challenges to cover the Work Role in a complete manner. Students might also need supplementary educational material to understand better the capability indicators proposed by the framework.

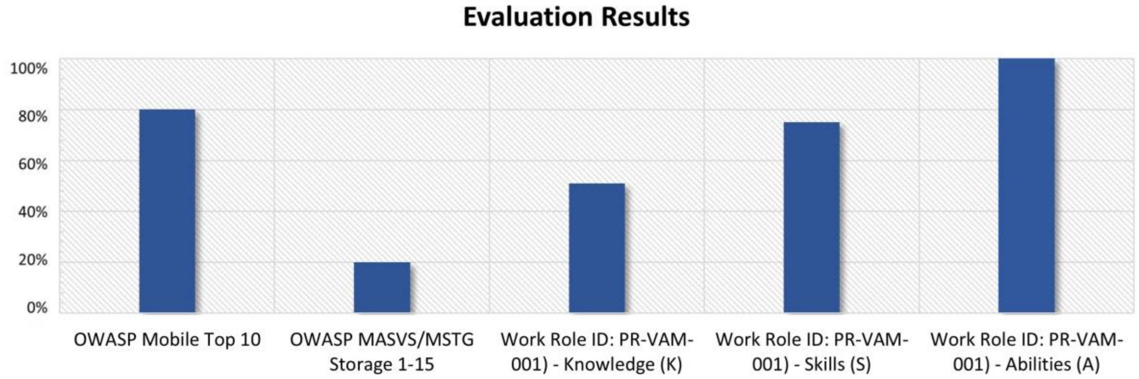

Figure 2. Evaluation results for DIVA and EVABS in relation to OWASP Mobile Top 10, MASVS/MSTG and the NICE framework

As presented in Figure 2, the CTF challenges managed to cover 80% of the OWASP Mobile Top 10 Risks and 20% of the MSTG specific Category (MSTG Storage) that consists of 15 different types of vulnerabilities. In terms of the NICE framework, the Work Role "Vulnerability Assessment Analyst (Work Role ID: PR-VAM-001)" was covered adequately.

More specifically, the capability indicators of Knowledge (K) were addressed by 51%, Skills (S) by 75% and Abilities (A) by 100%. The capability indicator of Knowledge (K) could be covered better, while the number of Abilities (A) proposed by the framework is rather low (4 Abilities overall and all of them covered by the CTF challenges). This may validate the reason for NIST merging Skills (S) and Abilities (A) in its later versions. The results (Figure 2) validate that DIVA and EVABS, as well as similar implementations, can be realistic enough and present the required learning context. In terms of MASVS and MSTG, the CTF challenges focused mostly on the MSTG Data Storage, meaning that the exercises provided information mostly on the management of local data storage on Android applications, an outcome that is validated by the relation to the other taxonomies as well. Other than that, in terms of the provided learning outcomes, DIVA and EVABS covered sufficiently most of the selected taxonomies and the related Work Role proposed by the NICE framework. Finally, the matching among the CTF challenges, the selected taxonomies and the NICE framework was successful in relation to RQ1. The results are summarized in Figure 2.

In addition to the educational and learning outcomes, DIVA and EVABS present the opportunity to use software tools and engage to the following technical concepts:

Logcat: Dumps logs and system messages, including error reporting and messages created by the application. This command-line tool was used in most challenges [58].

JD-GUI: Displays and analyses Java source codes (".class" files). The reconstructed source code with the JD-GUI [55] allow for the direct access to methods and fields.

Android Debug Bridge (ADB): Allows direct communication with the device to debug Android applications. This was a core component in solving challenges [60].

SQLite: The challenges introduce SQLite [61] which helps to engage with SQLite databases on Android devices.

Android Manifest: The challenges required accessing the Android manifest XML file, which contains important metadata about the Android application [62].

Drozer: Searches and evaluates applications for security vulnerabilities and cross-validates the results [63].

MobSF: Automated way for identifying security vulnerabilities of the mobile applications by conducting malware analysis, security assessment and performing static and dynamic analysis [52].

Android Package Kit (APK): Extended from the Java Archive (JAR), the Android Package Kit (APK) is the file format of the Android application package used by the Android operating system. During the challenges, the students were often invited to access APK files and get familiar with the structure.

Dex2jar: Tool to covert Java files (e.g., dex, class datatypes) [64].

Frida: Dynamic toolkit for developers, reverse engineers, and security researchers. The challenges required dynamic instruments of this tool to process files of the application and to monitor the device at any given time [65].

Apktool: Reverse engineering Android APK files. It was used in more than five challenges [66].

Android Studio Profiler: Analysis of network traffic between the server and the application [67].

Not all the software mentioned above was proposed by the CTF challenges or was necessary to complete the challenges. However, such software can be used as additional action points to cover better the learning topics. Therefore, challenges provide the opportunity to introduce other software tools to perform vulnerability assessment or code analysis. Then, students can use the aforementioned software (e.g., MobSF or Drozer) to construct and extend their knowledge further.

The selected CTF challenges not only engage technically the trainees to security aspects of mobile applications but can also provide fundamental tasks related to debugging tools and the Android architecture. DIVA and EVABS provide the assets for students to engage to the learning topic, however the challenges need to extend according to the results of this research and include information related to mitigation actions and best practices for secure coding. With the appropriate additions, DIVA and EVABS can be used efficiently as cybersecurity learning environments.

4. CONCLUSIONS

Regarding *RQ1 –* “Is *it possible to determine the learning outcomes from existing CTF challenges?”*, an evaluation methodology was constructed as a systematic approach to determine the learning outcomes of CTF challenges. The evaluation methodology was based on the relationship among the CTF challenges, OWASP Mobile Top 10 Risks, CWE by MITRE, and the NICE framework by NIST. MITRE OWASP Mobile Top 10 Risks and the CWE were used as open repositories to provide an automated way to extract the vulnerabilities and determine the learning outcomes of the CTF challenges. On the other hand, the NICE framework was successfully integrated in the evaluation methodology and was used as a proof of concept to match two CTF challenges, to a specific Work Role. The results show that the evaluation methodology proved to be useful and the learning outcomes from the CTF challenges were determined sufficiently. Using the proposed evaluation methodology, educators can identify potential improvements in existing CTF challenges and cover better the indicators of learning capacity.

Regarding the *RQ2 – "Do the existing CTF challenges cover adequately the learning topics to secure mobile application development?"*, the research investigated two CTF challenges that relate to this learning topic. The results show that DIVA and EVABS adequately address the technical aspects related to mobile cybersecurity, and specifically the development of secure mobile applications. The learning outcomes from DIVA and EVABS are similar and cover a very specific learning topic. Therefore, existing CTF challenges can cover the technical aspects of mobile applications, although with a limited scope. Research identified that additional educational and instructive material should be provided, and indicative security fixes must be presented. Furthermore, additional sub-scenarios must be provided to support and cover fully the learning topic. DIVA and EVABS present vulnerabilities in a gamified approach to be easily embedded and understood by the students. As a result, students or security professionals can engage in topics related to security vulnerabilities in mobile applications.

From the results of this research, quantitative information was retrieved justifying that the evaluation methodology can successfully determine the learning outcomes. DIVA and EVABS mainly address the Work Role of "Vulnerability Assessment Analyst (ID: PR-VAM-001)", at a rate of 65%, as presented in subsection 3.4. The NICE framework also provides information related to capability indicators that are not addressed by the CTF challenges and trigger ideas for extension. The challenges managed to cover an overall 80% of the OWASP Mobile Top 10 Risks, which means that they are realistic enough and address the main security issues in the development of secure mobile applications. During the challenges, twelve different software tools were required, which is an important aspect of the implementations.

The evaluation methodology could be used extensively as a basis to evaluate existing or upcoming CTF challenges. Information that was suggested to accompany the CTF challenges include the content presented in Section 3. The software tools for conducting the vulnerability analysis could be changed according to the CTF challenge and to select an educational framework similar to the NICE framework. The OWASP Mobile Top 10 can be selected depending on the topic and establish the relationship with MITRE CWE. Finally, the evaluation methodology can retrieve information on potential updates, which can be enabled to extend the CTF challenges, enhance the learning outcomes, and align with popular educational frameworks and taxonomies.

ACKNOWLEDGEMENTS

This work has been supported by the European Union's Horizon 2020 Project "CyberSANE" under grant agreement No. 833683.

REFERENCES

[1] S. Furnell, "The cybersecurity workforce and skills," Computers & Security, vol. 100, p. 102080, 2021.

[2] I. Ul Haq and T. A. Khan, "Penetration frameworks and development issues in secure mobile application development: A systematic literature review," IEEE Access, 2021.

[3] O. Nock, J. Starkey, and C. M. Angelopoulos, "Addressing the security gap in iot: towards an iot cyber range," Sensors, vol. 20, no. 18, p. 5439, 2020.

[4] J. Payne, "Secure mobile application development," IT Professional, vol. 15, no. 3, pp. 6–9, 2013.

[5] R. Vogel, "Closing the cybersecurity skills gap," Salus Journal, vol. 4, no. 2, pp. 32–46, 2016.

[6] S. A. Shonola and M. S. Joy, "Enhancing mobile learning security," ArXiv, vol. abs/1610.06046, 2016.

[7] S. M. Dakka, “Using socrative to enhance in-class student engagement and collaboration,” arXiv preprint arXiv:1510.02500, 2015.

[8] T. J. Burns, S. C. Rios, T. K. Jordan, Q. Gu, and T. Underwood, “Analysis and exercises for engaging beginners in online {CTF} competitions for security education,” in 2017 USENIX Workshop on Advances in Security Education (ASE 17), 2017.

[9] Borja, T., Benalcázar, M.E., Valdivieso Caraguay, Á.L., Barona López, L.I. (2021). Risk Analysis and Android Application Penetration Testing Based on OWASP 2016. In: Rocha, Á., Ferrás, C., López-López, P.C., Guarda, T. (eds) Information Technology and Systems. ICITS 2021. Advances in Intelligent Systems and Computing, vol 1330. Springer, Cham. https://doi.org/10.1007/978-3-030-68285-9_44

[10] M. M. Yamin and B. Katt, “Modeling and executing cyber security exercise scenarios in cyber ranges,” Computers & Security, p. 102635, 2022.

[11] D. Votipka, E. Zhang, and M. L. Mazurek, “Hacked: A pedagogical analysis of online vulnerability discovery exercises,” in 2021 IEEE Symposium on Security and Privacy (SP), pp. 1268–1285, IEEE, 2021.

[12] Vykopal, Jan, Valdemar Švábenský, and Ee-Chien Chang. "Benefits and pitfalls of using capture the flag games in university courses.” in Proceedings of the 51st ACM Technical Symposium on Computer Science Education, pp. 752–758, 2020.

[13] D. Burley, M. Bishop, S. Kaza, D. S. Gibson, S. Buck, A. Parrish, and H. Mattord, “Special Session: Joint Task Force on Cybersecurity Education,” in Proceedings of the 49th ACM Technical Symposium on Computer Science Education, pp. 918–919, 2018.

[14] M. Bhatt, I. Ahmed, and Z. Lin, “Using virtual machine introspection for operating systems security education,” in Proceedings of the 49th ACM Technical Symposium on Computer Science Education, pp. 396–401, 2018.

[15] Taylor, C., Arias, P., Klopchic, J., Matarazzo, C., & Dube, E. (2017). {CTF}:{State-of-the-Art} and Building the Next Generation. In 2017 USENIX Workshop on Advances in Security Education (ASE 17).

[16] V. Ford, A. Siraj, A. Haynes, and E. Brown, “Capture the flag unplugged: an offline cyber competition,” in Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, pp. 225–230, 2017.

[17] M. Lehrfeld and P. Guest, “Building an ethical hacking site for learning and student engagement,” in SoutheastCon 2016, pp. 1–6, IEEE, 2016.

[18] K. Leune and S. J. Petrilli Jr, “Using capture-the-flag to enhance the effectiveness of cybersecurity education,” in Proceedings of the 18th Annual Conference on Information Technology Education, pp. 47–52, 2017.

[19] J. Diakoumakos, E. Chaskos, N. Kolokotronis, and G. Lepouras, “Cyber-range federation and cyber-security games: A gamification scoring model,” in 2021 IEEE International Conference on Cyber Security and Resilience (CSR), pp. 186–191, IEEE, 2021.

[20] K. Chung and J. Cohen, “Learning obstacles in the capture the flag model,” in 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14), 2014.

[21] Švábenský, V., Čeleda, P., Vykopal, J., & Brišáková, S. (2021), “Cybersecurity knowledge and skills taught in capture the flag challenges.”, Computers & Security, 102, 102154, 2021.

[22] X. Meng, K. Qian, D. Lo, P. Bhattacharya, and F. Wu, “Secure mobile software development with vulnerability detectors in static code analysis,” in 2018 International Symposium on Net- works, Computers and Communications (ISNCC), pp. 1–4, IEEE, 2018.

[23] Davis, Andy, et al. "The Fun and Future of {CTF}." 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14). 2014.

[24] M. Guo, P. Bhattacharya, M. Yang, K. Qian, and L. Yang, “Learning mobile security with android security labware,” in Proceeding of the 44th ACM technical symposium on Computer science education, pp. 675–680, 2013.

[25] A. K. Jain and D. Shanbhag, “Addressing security and privacy risks in mobile applications,” IT Professional, vol. 14, no. 5, pp. 28–33, 2012.

[26] L. M. Podila, J. P. Bandreddi, J. I. Campos, Q. Niyaz, X. Yang, A. Trekles, C. Czerniak, and A. Y. Javaid, “Practice-oriented smartphone security exercises for developing cybersecurity mindset in high school students,” in 2020 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), pp. 303–310, IEEE, 2020.

[27] E. O. Mkpojiogu, A. Hussain, and M. O. Agbudu, “Security issues in the use of mobile educational apps: A review.,” International Journal of Interactive Mobile Technologies, vol. 15, no. 6, 2021.

[28] T. E. Gasiba, K. Beckers, S. Suppan, and F. Rezabek, “On the requirements for serious games geared towards software developers in the industry,” in 2019 IEEE 27th International Requirements Engineering Conference (RE), pp. 286–296, IEEE, 2019.

[29] N. Team, “Apk,” in Handbook for CTFers, pp. 269–293, Springer, 2022.

[30] J. Gao, L. Li, P. Kong, T. F. Bissyand ́e, and J. Klein, “Understanding the evolution of android app vulnerabilities,” IEEE Transactions on Reliability, vol. 70, no. 1, pp. 212–230, 2019.

[31] M. Ellis, L. Baum, K. Filer, and S. H. Edwards, “Experience report: Exploring the use of ctf-based co-curricular instruction to increase student comfort and success in computing,” in Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, pp. 303–309, 2021.

[32] T. Espinha Gasiba, U. Lechner, and M. Pinto-Albuquerque, “Cybersecurity challenges in industry: Measuring the challenge solve time to inform future challenges,” Information, vol. 11, no. 11, p. 533, 2020.

[33] “Open Web Application Security Project (OWASP) Mobile TOP 10 Risks.” https://owasp. org/www-project-mobile-top-10/. (accessed: 02.04.2022).

[34] “CWE, Common Weakness Enumeration.” https://cwe.mitre.org/. (accessed: 16.03.2022).

[35] W. Newhouse, S. Keith, B. Scribner, and G. Witte, “National initiative for cybersecurity edu- cation (nice) cybersecurity workforce framework,” NIST special publication, vol. 800, no. 2017, p. 181, 2017.

[36] C. Paulsen, E. McDuffie, W. Newhouse, and P. Toth, “Nice: Creating a cybersecurity workforce and aware public,” IEEE Security & Privacy, vol. 10, no. 3, pp. 76–79, 2012.

[37] “OWASP MASVS - Mobile Application Security Verification Standard.” https://github.com/ OWASP/owasp-masvs. (accessed: 16.03.2022).

[38] “DIVA Android - Damn Insecure and vulnerable App for Android).” https://www.nist.gov/itl/applied-cybersecurity/nice/nice-framework-resource-center/latest-updates. (accessed: 16.03.2022).

[39] “EVABS - Extremely Vulnerable Android Labs.” https://github.com/abhi-r3v0/EVABS. (ac- cessed: 16.03.2022).

[40] R. Raman, S. Sunny, V. Pavithran, and K. Achuthan, “Framework for evaluating capture the flag (ctf) security competitions,” in International Conference for Convergence for Technology- 2014, pp. 1–5, IEEE, 2014.

[41] R. B. Joseph, M. F. Zibran, and F. Z. Eishita, “Choosing the weapon: A comparative study of security analyzers for android applications,” in 2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications (SERA), pp. 51–57, IEEE, 2021.

[42] J. Mitra and V.-P. Ranganath, “Ghera: A repository of android app vulnerability benchmarks,” in Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering, pp. 43–52, 2017.

[43] “Ghera - Repository of verifiable Android app vulnerability benchmarks.” https://bitbucket.org/secure-it-i/android-app-vulnerability-benchmarks (accessed: 16.03.2022).

[44] Borja, T., Benalcázar, M. E., Valdivieso Caraguay, Á. L., & Barona López, L. I, “Risk analysis and android application penetration testing based on owasp 2016,” in International Conference on Information Technology & Systems, pp. 461–478, Springer, 2021.

[45] A. Alanda, D. Satria, H. Mooduto, and B. Kurniawan, “Mobile application security penetration testing based on owasp,” in IOP Conference Series: Materials Science and Engineering, vol. 846, p. 012036, IOP Publishing, 2020.

[46] Gil, Celio, et al. "A conceptual exploration for the safe development of mobile devices software based on OWASP." Int. J. Appl. Eng. Res 13.18 (2018): 13603-13609.

[47] M. Dawson, P. Taveras, and D. Taylor, “Applying software assurance and cybersecurity nice job tasks through secure software engineering labs,” Procedia Computer Science, vol. 164, pp. 301– 312, 2019.

[48] C. Pham, D. Tang, K.-i. Chinen, and R. Beuran, “Cyris: A cyber range instantiation system for facilitating security training,” in Proceedings of the Seventh Symposium on Information and Communication Technology, pp. 251–258, 2016.

[49] R. Beuran, C. Pham, D. Tang, K.-i. Chinen, Y. Tan, and Y. Shinoda, “Cytrone: An integrated cybersecurity training framework,” in Proceedings of the 3rd International Conference on In-

formation Systems Security and Privacy (ICISSP 2017), pp. 157–166, SCITEPRESS–Science and Technology Publications, 2017.

[50] M. Karjalainen, S. Puuska, and T. Kokkonen, “Measuring Learning in a Cyber Security Exercise,” in 2020 12th International Conference on Education Technology and Computers, pp. 205–209, 2020.

[51] A. Furfaro, L. Argento, A. Parise, and A. Piccolo, “Using virtual environments for the assessment of cybersecurity issues in iot scenarios,” Simulation Modelling Practice and Theory, vol. 73, pp. 43–54, 2017.

[52] “MobSF - Mobile Security Framework.” https://github.com/MobSF/Mobile-Security-Framework-MobSF (accessed: 16.03.2022).

[53] “OWASP Mobile Security Testing Guide (MSTG)).” https://github.com/MobSF/owasp-mstg. (accessed: 01.04.2022).

[54] “Current Version of The Workforce Framework for Cybersecurity (NICE Framework - Current Version).” https://www.nist.gov/itl/applied-cybersecurity/nice/nice-framework-resource-center/workforce-framework-cybersecurity-nice (accessed: 01.04.2022).

[55] “Latest Updates on The Workforce Framework for Cybersecurity (NICE Framework).” https://www.nist.gov/itl/applied-cybersecurity/nice/nice-framework-resource-center/latest-updates (accessed: 30.10.2021).

[56] “MITRE CVE - Common Vulnerabilities and Exposures.” https://cve.mitre.org/ (accessed: 16.03.2022).

[57] “Common Attack Pattern Enumerations and Classifications (CAPEC™))).” https://capec.mitre.org/ (accessed: 01.04.2022).

[58] “Logcat command-line tool)).” https://developer.android.com/studio/command-line/logcat (accessed: 01.04.2022).

[59] “JD-UI: Standalone graphical utility that displays Java source codes)).” https://java-decompiler.github.io/ (accessed: 01.04.2022).

[60] “Android Debug Bridge (adb))).” https://developer.android.com/studio/command-line/adb (accessed: 30.10.2021).

[61] “SQLite)).” https://www.sqlite.org/ (accessed: 01.04.2022).

[62] “Android Manifest - App Manifest Overview)).” https://developer.android.com/guide/topics/manifest/manifest-intro (accessed: 01.04.2022).

[63] “Drozer - Security Assessment Framework for Android)).” https://github.com/FSecureLABS/drozer (accessed: 01.04.2022).

[64] “Dex2Jar - Tools to work with android files (e.g., .dex and .class files)).” https://github.com/pxb1988/dex2jar (accessed: 01.04.2022).

[65] “Frida - Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers.)).” https://frida.re/ (accessed: 01.04.2022).

[66] “APKTool - A tool for reverse engineering Android APK files)).” https://github.com/iBotPeaches/Apktool (accessed: 01.04.2022).

[67] “The Android Profiler)).” https://developer.android.com/studio/profile/android-profiler (accessed: 01.04.2022).