International Journal of Information Security (2023) 22:833–867
https://doi.org/10.1007/s10207-023-00669-z

REGULAR CONTRIBUTION

Secure cloud-based mobile apps: attack taxonomy, requirements, mechanisms, tests and automation

Francisco T. Chimuco[1,2] · João B. F. Sequeiros[1] · Carolina Galvão Lopes[1] · Tiago M. C. Simões[1] · Mário M. Freire[1] · Pedro R. M. Inácio[1]

Published online: 17 February 2023

Abstract

The adoption and popularization of mobile devices, such as smartphones and tablets, accentuated after the second decade of this century, has been motivated by the growing number of mobile applications, which can solve problems in different areas of contemporary societies. Conversely, the software development industry is motivated by the increasing number and quality of resources that mobile devices possess nowadays (e.g., memory, sensors, processing power or battery). While powerful mobile devices do exist, one of the main driving factors behind the increase of resources is the usage of Cloud technology, which strongly complement mobile computing. As expected, the adoption of measures to mitigate security issues has not accompanied the growth and speed of development for Cloud and Mobile software, to ensure that these are resilient to attacks by design. Aiming to contribute to decrease the gap between software and security engineering, this paper presents a deep approach to attack taxonomy, security mechanisms, and security test specification for the Cloud and Mobile ecosystem of applications. This is also the first time an encompassing and conjoined approach is provided for attack taxonomy and specification of security tests automation tools for this ecosystem.

Keywords Security · Mobile computing · Cloud computing · Attack taxonomy · Security mechanisms · Security testing

1 Introduction

The intersection between Cloud computing and Mobile computing resulted in the emergence of the Cloud and Mobile ecosystem. The emergence of this ecosystem was motivated by the shortage of resources on mobile devices (smartphones and tablets), e.g., processing, low battery autonomy, storage, the ease of transport for mobile devices and the possibility to obtain additional resources to mobile devices (storage and execution of apps, data storage, processing, as well as from the need to process increasingly large amounts of data) from the Cloud.

Additionally, widespread adoption of mobile devices by end users, making them ubiquitous in their lives has been fueling the development of mobile applications. For example, the Statista website predicted that, in 2021, the number of users of mobile devices was approximately of 4.3 billion [161]. China, USA and India are the countries with the largest smartphone userbases, with an estimate of each having over 100 million users [161]. Cisco estimates that over 70% of the global population will have mobile connectivity by 2023 [30].

Cloud computing, also known simply by *the Cloud* [15], is based in having centralized computational resources that can be remotely accessed, while being virtually inexhaustible, capable of adapting computational power to the

✉ Francisco T. Chimuco
francisco.chimuco@ubi.pt

João B. F. Sequeiros
jbfs@ubi.pt

Carolina Galvão Lopes
carolina.lopes@ubi.pt

Tiago M. C. Simões
tiago.simoes@lx.it.pt

Mário M. Freire
mario@di.ubi.pt

Pedro R. M. Inácio
inacio@di.ubi.pt

[1] Universidade da Beira Interior and Instituto de Telecomunicações, Rua Marquês D'Ávila e Bolama, 6201-001 Covilhã, Castelo Branco, Portugal

[2] Instituto Superior de Ciências de Educação da Huíla, Rua Sarmento Rodrigues, 230, Lubango, Huíla, Angola

preferred dimension, storage and easily accessible, without user interaction. Resources are provided in the *pay-per-use* paradigm, an attractive business model, removing complexity and acquisition cost from the equation. Cloud resources are provided via different models, typically segregated into [12]: (i) *Software-as-a-Service* (SaaS), which is the use of the Cloud to execute software applications that can be accessed through a browser or web application; (ii) *Platform-as-a-Service* (PaaS), which relates to using the Cloud as a platform for developing, building and maintain applications and services, with PaaS providers offering a predefined combination of application servers, such as LAMP (Linux, Apache, MySQL and PHP) software stacks, J2EE and Ruby; and (iii), *Infrastructure-as-a-Service* (IaaS), a service model related to the provisioning of hardware resources, with the user controlling a virtualized operating system (OS), storage, network, etc. Cloud resources are typically provided through three main deployment models, whose characteristics are presented in [152]. One of the key drivers for using Cloud services is cost efficiency, as individual and business users have no need to significantly invest in Information & Technology (IT) infrastructure, given the inherent *elasticity* and *scalability* characteristics, enabling the scaling of resources as needed [152]. On the other hand, mobile computing is more focused on mobile consumer technology and ensuring its convenience to users, given its ubiquity characteristic. Considering that this technology has a strong focus on connectivity (e.g., to allow communication at a distance), increasing its attack surface, it is clearly necessary to have frameworks that generate a set of security engineering principles that allow software engineers and professionals in related areas to develop systems and applications that are secure by design or construction. This approach is the only allowing cost efficiency and trust in technology in the long run, since it allows avoiding vulnerabilities from the beginning, i.e., security issues are not left to the end and also allows decreasing the development and deployment time of the applications of the ecosystem considered in this study, which are key aspects of the technology market.

For the purpose of this paper, the *Cloud and Mobile ecosystem* is defined as being the one incorporating all mobile applications (i.e., those that run on mobile devices) that, in some way, make use of a remote computing resource, such as processing, network or storage, through the use of wireless access technologies (e.g., access points or satellite), as well as mobile networking technologies such as Third Generation (3 G), Fourth Generation (4 G) or Fifth Generation (5 G). Additionally, these applications depend on Cloud resources to offer a significant or important part of their functionalities.

Figure 1 schematizes the macro ecosystem resulting from the combination of the Cloud and Mobile ecosystems, emphasizing connection-related technologies [50]. A user accessing data stored or processed in one or several Clouds through a mobile application comprises a typical scenario in this ecosystem. Many of their characteristics are complementary, and communication between them for different purposes can occur [50]. Different communication patterns and technologies (e.g., Wi-Fi, Bluetooth, Near-Field Communication (NFC), General Packet Radio Service (GPRS), Radio-Frequency Identification (RFID), 3 G, 4 G, 5 G, ZigBee, etc.) can be used for communication [7], with different security measures needing to be ensured, according to each situation [156]. Finally, Mobile Device Manager (MDM), blank listed apps, untrusted apps and managed apps are other features or resources of a mobile device as described in [35].

Considering the peculiarities and different focuses of the technologies involving such a rich ecosystem, along with the fact that communications at a distance are involved, it is a huge challenge to develop systems or applications for the *Cloud and mobile ecosystem*, specially due to the heterogeneity and complexity of the various technologies (networks, programming languages, platforms, attack surfaces, taxonomy of attacks, Cloud, mobile devices) involved in the process of developing, deploying and using these applications, notably in terms of security and privacy [64,127,168]. This situation is aggravated by the short-time companies that have to deliver new products to the market, neglecting or leaving behind the security and privacy of the sensitive data of the systems and their end users. The immediate consequence of this neglect is the existence of a large number of vulnerabilities, resulting in security and privacy problems in mobile devices and services. Hence, the increased use of Cloud-based mobile applications has to be accompanied with the adoption of secure development paradigms, keeping security and privacy issues in mind from the beginning and not leaving them for a later stage. The main contributions of the work presented herein can be described as follows:

- A exhaustive attack taxonomy for the *Cloud and Mobile ecosystem* is proposed and discussed, sourced from over 150 references from the specialized literature and following a systematic well consolidated approach;
- The taxonomy is complemented by the inclusion of a mapping between mitigation techniques and the attacks;
- A deep approach on security requirements, mechanisms and tests for the *Cloud and Mobile ecosystem* is also included, to provide closure to security requirements engineering in this ecosystem.

In addition to the introduction, this article is organized as follows: Sect. 2 presents the taxonomy of attacks in Cloud and Mobile ecosystem. Section 3 presents the mechanisms and countermeasures to attacks on the Cloud and Mobile ecosystem. Security tests are covered in Sect. 4. Research challenges are discussed in Sect. 5. Finally, conclusions are presented in Sect. 6.

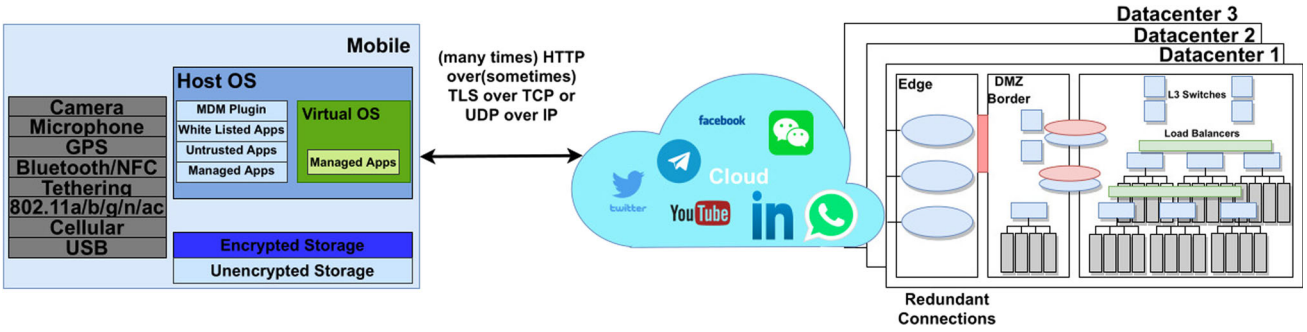

Fig. 1 Representation of the *Cloud and Mobile ecosystem* (based on [152])

2 Attack taxonomy

This section converges to an attack taxonomy for the Cloud and mobile ecosystem. To get to that output, it first briefly discusses the methodology used for this purpose; to then present a revision on works related with, or including taxonomies of attack to the sub-ecosystems under study; to finally presenting the proposed taxonomy.

2.1 Research methodology

In this subsection, the methodology of this research is presented, aiming to elaborate and present a new and more in-depth taxonomy of attacks or threats to Cloud-based mobile applications and their corresponding countermeasures. It aims to provide a common classification scheme that can be shared among security experts and mobile application developers.

The main drivers behind this work is to propose a very comprehensive taxonomy that addresses shortcomings of existing ones, while fully meeting the requirements of a classification scheme, typically defined as the properties of being *Unambiguous*, *Useful*, *Approved*, *Understandable*, *Exhaustive*, *Deterministic*, *Mutually exclusive*, *Reproducible*, *Conforming to standards*, and *Having well-defined terms/clear criteria* [62,70,103]. The first step of the method was thus to analyze the already existing taxonomies applicable to the Cloud or and mobile ecosystems.

At this stage, we resorted to the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) [120, 131] methodology to identify relevant publications. In the scope of this work, PRISMA was driven by the following Research Questions (RQs) and motives:

RQ1 What is a taxonomy and what are the requirements that need to be fulfilled in the specific cybersecurity domain, so as to be effective and useful tool?

RQ2 What are the current limitations of existing Cloud-based mobile application security attacks classification schemes?

The RQ1 guides the research in the direction of the aforementioned objectives and reflects the main motivation of this part of work.

The formulation of RQ2 provided an underlying assurance that existing taxonomies were analyzed while taking into account known satisfactory requirements. Only those specific to the Cloud and Mobile ecosystem were included in this analysis.

PRISMA methodology also requires the definition of the *Time Interval*, the *Keywords*, the *Results* and the *Selection Process* or *Inclusion/Exclusion Criteria*. Each of these phases is described in detail below.

2.1.1 Time interval

The search took place between November 2021 and March 2022. Articles published between 2007 and 2022 obtained from the most significant digital libraries on these subjects were included, namely IEEE Xplore, ACM Digital Library, Elsevier, Taylor & Francis and Springer Link.

2.1.2 Keywords

This phase was aimed at selecting the most appropriate keywords in terms filtering the relevant articles for review. Synonyms, alternative spellings of the main elements of the field of taxonomies of threats or attacks in the context of Cloud-based mobile application, i.e., Cloud and Mobile ecosystem were also included to find relevant articles. For this purpose, the key words used for this research process are mirrored in Table 1.

2.1.3 Results

Table 2 presents the results of the first search, and this is the data used for the research process.

As previously mentioned, the PRISMA methodology was used aiming at a refined, effective and efficient selection of the publications included in this study, given the fact that

Table 1 Keywords used in research process

Keywords	Data interval
Taxonomy of Cloud-based apps mobile threats	2007–2022
Attack taxonomy in mobile loud computing	2007–2022
Classification of attacks on Cloud-based mobile apps	2007–2022
Mobile vulnerabilities	2007–2022
Security in Cloud and Mobile ecosystem	2007–2022
Security issues in Cloud and Mobile ecosystem	2007–2022

Table 2 Publication retrieved on attack taxonomy in Cloud and Mobile ecosystem

Library	Date of first search	Results
ACM digital library	10/01/2022	98
IEEE Xplore	10/01/2022	126
Springer link	15/01/2022	64
Elsevier	25/01/2022	119
Taylor & Francis	20/01/2022	157
Total		**564**

this is widely accepted and quite preferred in the world of scientific research.

2.1.4 Selection process

Table 3 shows the inclusion/exclusion criteria used in selection process.

Using these criteria and based on the PRISMA methodology and its associated flow diagram, all its 164 articles were read thoroughly, from which resulted 90 publications that were included in the present review. Figure 2 illustrates the PRISMA flow diagram.

Table 3 Inclusion–exclusion criteria

Criterion	Rational
Inclusion 1. Every taxonomy that mentioned security issues related to Cloud-based mobile applications	Only those papers that are focused on directly proposing the classification of attacks for Cloud and Mobile ecosystem should be part of the study
Inclusion 2. Any published taxonomy originating from academia, industry or similar reputable organizations	The taxonomies for this study may be of academic or industrial origin
Exclusion 1. A paper that does not focus on threats classification in the mobile Cloud-based applications	Only papers that are focused on discussing the security issues of the Cloud and Mobile ecosystem will be accepted
Exclusion 2. Non-English papers	All articles not published in English will be excluded
Exclusion 3. Full-text papers	If the article does not focus on attacks on Cloud-based mobile applications, they will be excluded after a reading of the abstract, introduction and conclusions

2.2 Review of existing mobile cloud-based application security attacks taxonomies

This section focuses on the analysis of five publications related to the theme under study obtained from the 90 publications included in this review, as detailed in Sect. 2.1. All of them are taxonomies described in scientific papers sourcing from research in academia.

2.2.1 Wu et al. taxonomy

Wu et al. [180] published a survey in 2007 titled *A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks*, aimed at providing the state of the art in terms of attacks and countermeasures in Mobile Ad Hoc Networks (MANET). The classification of attacks was made according to the layers (application, transport, network, data link, physical and multi-layer) of the Open Systems Interconnection (OSI) model. This is a very interesting approach and comes as an asset for researchers and system engineers. However, the approach taken at that time does not take into account (and had no way of doing so) the technological advances that have occurred since the emergence of smartphones and tablets.

2.2.2 Zou et al. taxonomy

Zou et al. [193] published the paper titled *A Survey on Wireless Security: Technical Challenges, Recent Advances and Future Trends* in 2016. Their work was focused on examining security vulnerabilities and threats in wireless communications and devise efficient defense mechanisms toward

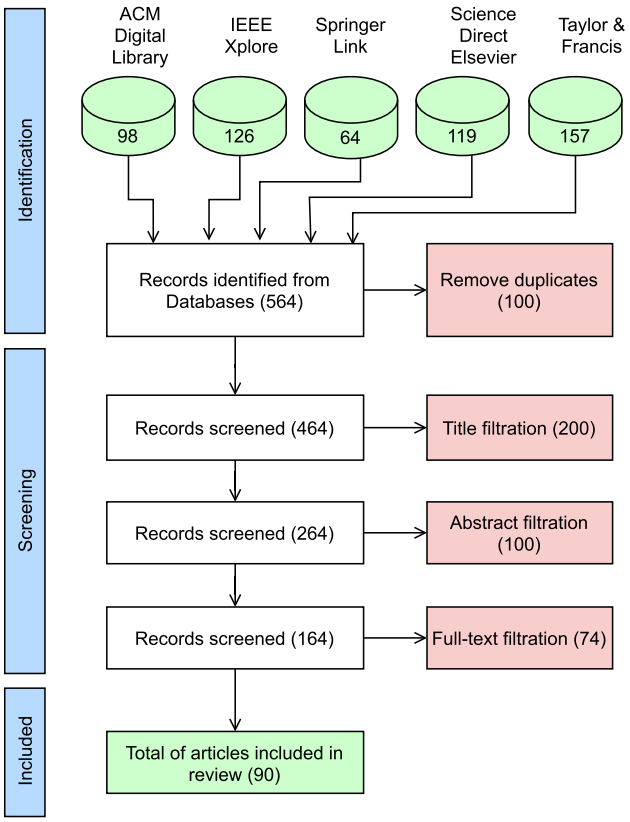

Fig. 2 PRISMA flow diagram

securing wireless networks. The classification of threats and attacks is made according to the layers of the OSI model of wireless networks and is a good source for researchers and developers of mobile applications that make use of mobile networks. However, it does not include a specific classification for Cloud-based mobile applications nor does it take into account threats to the back-end.

2.2.3 Bhatia and Verma taxonomy

Bhatia and Verma [21] presented their work in this area in a paper titled *Data security in the mobile cloud computing paradigm: a survey, taxonomy and open research questions* in 2017. Their main aim was to conduct a comprehensive state-of-the-art study on several related topics, among which are biometric and soft multifactor cipher solutions for data security and privacy assurance, as well as security issues and data security frameworks in the mobile cloud. Their study also presents a classification of Cloud and Mobile ecosystem security issues based on three categories, namely architecture, privacy and compliance.

This [21] taxonomy is very useful for obtaining information for those who wish to develop secure Cloud-based mobile applications and for researchers. Furthermore, it respects the requirements of a taxonomy. However, the classification still has gaps regarding some security problems, e.g., the ones that have the components of the Cloud and Mobile ecosystem, such as network, transport and communications, as the attack vector. The same happens in relation to security problems deriving from the human factor or social engineering.

2.2.4 Moorthy et al. taxonomy

In 2020, Moorthy et al. [122] published the results of their study in a survey titled *Security and privacy attacks during data communication in Software-Defined Mobile Clouds*. This study aimed at providing an in-depth analysis of possible attacks and threats to data privacy Software-Defined Mobile Clouds. Firstly, the research presents a new architecture for Cloud and mobile ecosystem. Secondly, it presents the various threats and vulnerabilities according to five categories, namely application, protocol, source, network and control-based security faced by Software-Defined Mobile Cloud in its different architectural layers. Finally, the study presents countermeasures to the threats and vulnerabilities in the Cloud and Mobile ecosystem. This research has the advantage of presenting one of the most in-depth taxonomies of attack for the Cloud and Mobile ecosystem, as well as the respective countermeasures. However, it presents a gap regarding the lack of reference to threats and vulnerabilities of mobile Web applications and other Cloud-related applications.

2.2.5 Almaiah et al. taxonomy

In the paper titled *Classification of Cyber Security Threats in Mobile Devices and Applications*, Almaiah et al. [9] presented, in 2021, a comprehensive framework of mobile device and application—cybersecurity threat classifications, being at that date the most extensive classification scheme of security threats for the mobile ecosystem. The main goal of the framework was to systematically identify cybersecurity threats, showing their potential impacts and to bring the attention of mobile users to these threats in order to take protective measures as appropriate. This work presents one of the most profound approaches regarding the taxonomy of attacks to the Cloud and Mobile ecosystem, constituting an added value in terms of a source of consultation for all those involved in the process of secure development of secure applications for this ecosystem. However, the classification does not take into account all threats on the server side or back-end. Furthermore, the framework presented does not provide information (nor is focused) on threat countermeasures.

2.2.6 Others

Although they are not part of the online libraries selected for the sample of this study, the publications of Taleby et. al [169] and Friedman, & Hoffman [54], are herein included for their number of citations and impact. The first publication analyses the state-of-the-art on security solutions, threats and vulnerabilities during the period between 2011 and 2017, focusing on software attacks, such as smartphone applications. It also proposes a set of some countermeasures against those attacks. The second publication presents a taxonomy of attacks in seven categories, namely malware, phishing and social engineering, direct attack by hackers, data communications interception and spoofing, loss and theft of devices, malicious insider actions and user policy violations. The aforementioned study also presents a set of countermeasures against those threats. However, the first article is not as deep in terms of approach and is limited to the mobile ecosystem, while the second one veers toward not being as specific in terms of the mobile ecosystem, as it considers all mobile devices (laptop and notebook personal computers, handheld computers, cell phones, PDAs, BlackBerry devices and iPod mobile digital devices) as mobile.

2.2.7 Comparison of related works

Table 4 compares the different works reviewed in the previous sections based on seven aspects, namely: topics addressed, Cloud and Mobile attacks landscape, mitigation landscape and research challenges. Several symbols are used for this comparison on the table: ✓ is used to denote that the description of a subject is addressed on the document, + or ++ are used to emphasize special attention given to a specific aspect (there is a difference with regard to the degree of emphasis, that is, the greater the emphasis, the symbol ++ is used), × are used to denote the absence of the subject.

The approach presented in the following section differs from the aforementioned ones in several aspects, starting with the depth and extension of the approach, since it aims at covering all sub-ecosystems that compose the Cloud and Mobile ecosystem as a whole, namely the Cloud, mobile Computing and the Internet in between. Also, the approach taken presents a taxonomy according to five domains of the Common Attack Pattern Enumerations and Classifications (CAPEC[1]) [119] classification, namely application, communication, hardware, physical security and social engineering. In addition, a set of mechanisms to mitigate each of the attacks is also presented. Furthermore, research challenges are presented, which can be used as a basis for future research.

[1] CAPEC is only a catalog of attacks organized according to different criteria (domain, mechanism, etc.).

2.3 The proposed taxonomy

As the Cloud and Mobile ecosystem results from the intersection of Cloud, mobile and Internet technologies, it became heir to many of the problems and threats of each of them. While Internet technologies, and therefore their related threats, do not fit directly in the scope of this study, which focuses on Cloud and mobile threats, they are nonetheless an integral part, due to, as previously mentioned, Internet technologies being closely intertwined with both Cloud and Mobile ecosystems. Figure 3 presents the taxonomy of attacks on the Cloud and Mobile ecosystem proposed herein, with many of those attacks steaming from the aforementioned three sub-ecosystems, also as a function of the CAPEC attack domains (note that the domain Supply Chain was omitted here; this was a deliberate decision, taking into consideration that many of the attacks that could be classified in that domain can also be classified as physical, hardware or social engineering attacks; therefore, this was a conscious decision, taken for the sake of simplifying the taxonomy). As discussed in Sect. 2.2, the classification of threats or attacks is done precisely according to the five domains of CAPEC. The next subsections focus on each one of the sub-types of attacks at a time.

2.3.1 Software-based attacks

The *Software-based attacks* category all have in common that they exploit application weaknesses resulting from vulnerabilities in the coding, construction or implementation of software applications, with the aim of causing a negative technical impact. Potential attacks on this category are described below.

Session fixation: this attack aims to achieve account hijacking, so as to subsequently and fraudulently enable access to areas restricted to legitimate users of a web app or a hybrid mobile app [98,172]. The success of this attack depends on prior login, as it requires the session ID of a legitimate user of the target application, impersonating the victim, which grants privileges and results in the violation of confidentiality, access control and authorization of sensitive user data, or theft of money from mobile banking app or mobile interbank application.

Cross-Site Request Forgery (CSRF): CSRF is an attack that forces the user to execute unintended actions in an application for which it is already authenticated in a given moment [39,102,138]. These attacks target unrequested status changes, and not directly data theft, as the attacker cannot see the answer to the forged request. According to [138], CSRF attacks occur when a website allows changes to be made through `GET` requests. There are two mains approaches to this type of attack, stored CSRF attacks and reflected CSRF attacks.

Table 4 Comparison of related works on attack taxonomies for the Cloud and Mobile ecosystem (✓ is used to denote that the description of a subject is addressed on the document, + or ++ are used to emphasize special attention given to a specific aspect, × are used to denote the absence of the subject)

Survey	Year	Topic focused	AttacksLandscape	Mitigation landscape	Cloud attacks	Mobile attacks	Taxonomy landscape	Research challenges
Wu et al. [180]	2007	Survey of attacks and counter-measures in MANET	✓	+	×	✓	+	✓
Zou et al. [193]	2016	Security vulnerabilities, threats and mitigate mechanisms on Wireless	✓	✓	×	✓	✓	✓
Bhatia and Verma [21]	2017	Security issues in mobile cloud, mitigation mechanisms	+	+	+	+	✓	++
Moorthy et al. [122]	2020	Security, Malicious attacks, Preventive solutions	++	++	+	++	++	++
Almaiah et al. [9]	2021	Mobile devices, Mobile applications, Cyber threats, Malicious attacks	++	✓	×	++	++	×
This article	**2023**	**Security, Mobile, Cloud, Attack Taxonomy, Security Mechanisms, Security Testing**	++	++	++	++	++	+

Cross-Site Scripting (XSS): this attack exploits vulnerabilities in Web applications in order to execute code in the victims end (most of the times in the browser or in a webkit of a mobile application). It takes advantage of improper data validation or the possibility to store contents that may become active when a web resource is visited [90,107]. Motivations for the attack vary, but are often related with compromising the user machine to obtain privileged access or exfiltrate data. The most well-known sub-types of XSS are: i) stored XSS (saving the malicious code ininterruptibly in a web application managed resource) and ii) reflected XSS (in which the malicious code or attack script are sent directly toward the user, e.g., in the URL, and not stored it in a lasting manner) [12,176].

SQL Injection (SQLi): SQLi is a command injection technique used to attack applications that use database management systems supporting Structured Query Language (SQL), in which the attacker attempts to introduce characters or keywords of a SQL instruction in an attempt of altering the original underlying programming logic [12,14,61,154]. The attack is often perpetrated to access, alter or remove information, inject accounts or bypass access control mechanisms. SQLi is one of the most commonplace and severe attacks, being classified as high risk, and an entry validation vulnerability [130].

Spoofing attacks: according to Bhatia et al. [21], spoofing attacks are a fraudulent act in which an entity fakes its identity to attempt to access resources and critical data. There are four variants of the spoofing attack, namely content spoofing, identity spoofing, resource location spoofing and action spoofing. There are several means to perform this type of attacks.

Buffer overflow attacks: a buffer overflow is an anomaly on a program occurring during a write operation to a buffer in memory, exceeding its limits and writing in adjacent or deliberate memory blocks. Their malicious effect can be maximized when non-validated inputs are projected to execute code after the overflow. Buffer overflows can cause irregular program behavior, including memory access errors, incorrect results or system security breaches [99].

Code inclusion attacks: in this type of attack, an adversary focuses on exploiting a weakness for the purpose of forcing arbitrary code to be retrieved (remotely or locally) and executed [18,73]. Inclusion differs from code injection by the mode of operation (in this case, the weakness is used to obtain the code that is to be executed).

Brute force attacks: This type of attack consists in trying a potentially large number of possibilities until one grants access or achieves an operation on a system [29,66]. The success of the attack depends on the domain of the variables and on attenuating factors that might make guessing easier. The motivations behind this attack are mainly to access, destroy, leak and steal sensitive information from the system and its legitimate users. There are several custom made and well-known tools available for different types of brute force attacks, namely hashcat,[2] Hashtopolis,[3] Aircrack-ng[4] or John the Ripper.[5]

Cache Poisoning Attacks: cache poisoning is the act of introducing false information into a Domain Name System (DNS) cache in order to cause DNS queries to return an incorrect response and, e.g., redirect users to malicious websites. This type of attack can target the cache of an application (e.g., a web browser cache) or a public cache (e.g., a DNS or Address Resolution Protocol (ARP) cache), exposing the application to a variety of attacks, such as redirection to malicious websites and malware injection [118].

Code injection attacks: This type of attack targets injecting malicious code into user inputs from the interface (web application forms or hybrid mobile applications) of applications written in HTML5. These types of applications are vulnerable to code injection attacks because of the possibility of merging the application's data and implementation code. In case of hybrid mobile applications based on HTLM5, it has increased the attack vectors or channels such as, Contacts, SMS, Wi-Fi, NFC, QR Code, etc [128]. This allows the attacker to inject malicious code to exhaust all the victim's resources. A clear example of such an attack is XSS, already described above. In addition to XSS, there are other techniques of carrying out a code injection attack.

Command injection attacks: this is a class of attacks to which web applications are susceptible, resulting from the semantic gap existing between database interpretation and web application interpretation, as well as from the inappropriate handling of user input [162]. The potential technical impact of this type of attack is the execution of unauthorized commands, thus affecting the confidentiality, integrity and availability of the system. Given the sheer amount of technologies composing web applications, there are several variants of this type of attack, namely LDAP Injection, IMAP/SMTP Command Injection, XML Injection, Manipulating Writeable Terminal Devices, JavaScript Command Injection, NoSQL Injection and OS Command Injection. Some SQLi are part of this type of attacks too.

Cryptanalysis attacks: Cryptanalysis focuses on finding vulnerabilities in cryptographic algorithms and using these weaknesses to decrypt the ciphertext without knowing the secret key. In addition, this can have other purposes such as Total Breach, Global Deduction, Information Deduction and Algorithm Distinguishing [119].

Audit log manipulation attacks (ALM): This type of attack targets log files for the purpose of manipulating (deleting,

[2] https://hashcat.net/hashcat/.

[3] https://hashtopolis.org.

[4] https://www.aircrack-ng.org.

[5] https://www.openwall.com/john/.

reading, and altering) them. In a log file audit manipulation attack scenario, an attacker injects, manipulates or deletes information in the log file in an attempt to deceive a potential audit of an attack. The success of this type of attack depends on the insufficiency of log file access controls mechanisms [119].

Session hijacking attacks: this type of attack involves an adversary exploiting weaknesses in the use of an application's authentication sessions. Put another way, this type of attack results from the successful exploitation of improper authentication [36,125]. Its main purpose is account hijacking. The attacker may be able to steal or manipulate an active session and use it to gain unauthorized access to the application. Finally, this type of attack can be executed through four techniques, namely Reusing Session IDs (also known as Session Replay), Session Fixation, Session Sidejacking and Cross-Site Tracing.

Side-channel attacks: a side-channel attack is typically defined [60] as the activities targeting the leakage of information from a physical system while exploiting timing, power consumption, and electromagnetic and acoustic emissions. Side-channel attacks can be carried out either locally, through physical access or at a short distance from the target device, or remotely, from malware hosted in the target cloud environment. Regarding smartphones/tablets, sophisticated side-channel attacks that target the built-in sensors of these devices have been developed, allowing their exploitation to infer keyboard input on touchscreens through sensor readings of native applications and websites, infer the location of the user by the power consumption available in the `proc` file system (`procfs`), and even infer the identity, location and diseases of the user through `procfs` [159].

Virtual machine (VM) escape attacks: a VM escape occurs whenever an application escapes the VM environment in which it is running and obtains control over the VMM, as it escalates its VM privileges to root level. Attackers usually try to break the guest operating system (OS) to access the hypervisor, or penetrate functionalities of another guest OS and its host OS. A single security breach on the hypervisor (e.g., a defective implementation of a virtualized resource) can lead to a compromise. By controlling the hypervisor, the attac-kers can do anything with the VMs it is hosting [1,41,149].

VM migration attacks: VM migration is one of the typical operations of virtualization, where VMs can be transferred between physical machines if needed. Therefore, during this process, VMs need to be protected against insecure networks or attacks. In such an attack scenario, an adversary can, among other things, initiate or redirect the migration process to a malicious network from which the VM can be accessed, cloned and generally compromised [1].

Malware-as-a-service attacks: Malicious code, or simply malware, are scripts or programs that can be injected in the Cloud and mobile ecosystem [74,163]. Malware are traditionally mainly classified based on two main factors: propagation strategy (e.g., virus or worm) and malicious activity (Trojan Horse, spyware, adware, rootkits, botnets, ransomware, backdoors and keyloggers) [142]. Malware injection allows attackers to exploit system (or human) vulnerabilities and manage authorizations, allowing unauthorized accesses to the system. For the Cloud and Mobile ecosystem, malware has a strong impact on user privacy and confidentiality, as they often provide system access to attackers. Additionally, invaders can execute actions such as stealing confidential user data [85,163].

Malicious QR code: in such an attack scenario, an attacker uses malicious QR codes to direct users to fraudulent websites, disguised as legitimate ones, with the objective of stealing sensitive personal information. This type of attack exploits human vulnerabilities and the fact that QR codes are becoming increasingly popular and often seen as trusted.

Server side request forgery (SSRF): a SSRF attack consists of abusing a functionality on the server in order to read or update internal resources. In an SSRF attack scenario, a malicious entity, among many actions, is able to provide or modify a URL that code running on the back-end will read or submit data to. Furthermore, the attacker can read the server configuration, connect to internal services such as http-enabled databases [24,75]. In case the vulnerable server is hosted on a remote server in the Cloud, SSRF attacks require less effort from attackers, causing a higher impact in terms of dangerousness in terms of data leakage or theft.

2.3.2 Communications-based attacks

Attacks in this major category target the exploitation of communications artifacts and related protocols for the purpose of blocking communication, manipulating and stealing data, causing a negative technical impact.

Man-in-the-middle (MitM) attacks: in MitM attacks, an attacker positions itself somewhere in between the sender and receiver of a communication, in this case between two users or clients of a Cloud-based mobile app (client and server). MiTMs can be either passive or active. If the attacker only eavesdrops or decrypts the message, it is a passive attack; if (s)he also has the means to change the message, violating its integrity, then it is an active attack. According to Ferrag et al. [51], in a MitM attack scenario, an attacker can intersect, modify and secretly relay the communication between two entities, without them realizing it. MitM are often an intermediate step toward more complex attacks or intrusions. An example of MitM attacks is presented in Kampourakis et al. [81], where the authors show how MitM attacks over HTTPS can be carried out in commonplace browsers.

Sniffing: Sniffing is the act of obtaining data in real time from data transmitted between smartphones (or tablets) and

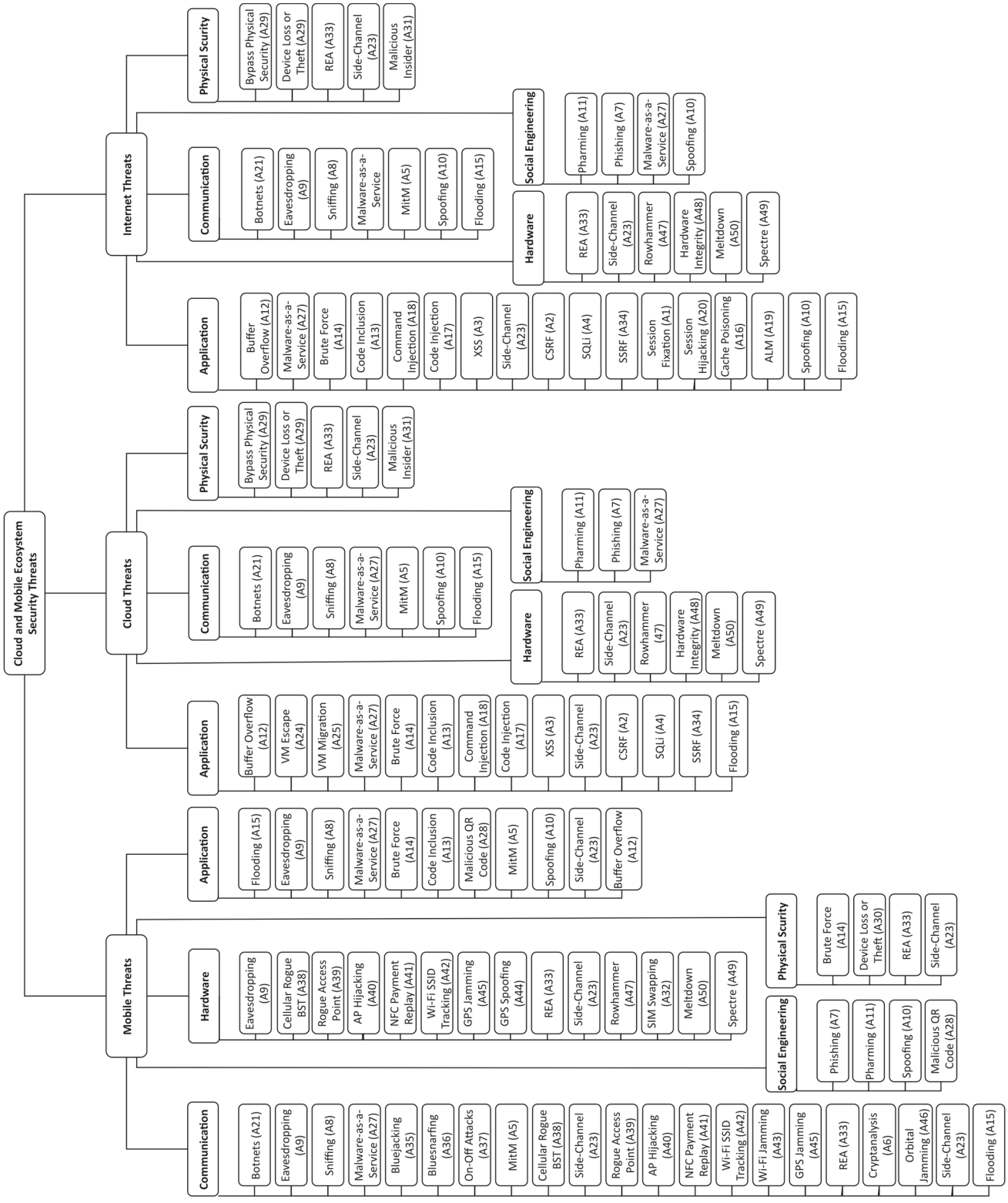

Fig. 3 Cloud and Mobile ecosystem security threats

the Cloud, through a network (Bluetooth, Wireless Sensor Network (WSN), Wi-Fi, 3 G/4 G/5 G, etc.). These attacks allow hackers to intercept and monitor user data, while it is in transit in the network, without interfering with the transmission process, to hinder detection. If strong cryptographic schemes are not used, data can be read or interpreted [20,51,85]. Also, an attacker can capture data from the different sensors available on a mobile device (accelerometer, Global Positioning System (GPS), compass, gyroscope, microphone, camera and headset) without user knowledge or consent [164].

Eavesdropping attacks: Eavesdropping[6] is a type of attack where the attacker tries to gain access to sensitive information of legitimate users from the messages (text, voice and video) exchanged between two or more users, e.g., of Instant Messaging (IM) applications. The same applies to recorded calls, call logs and multimedia stored in clear text in memory cards. Generally, fraudulently gaining access to the critical information of targets, either for political, financial and other gains, is the goal of the eavesdropping attack. The traditional eavesdropping attack affects mainly the communications layer of the mobile phone network and can be legitimate and legal (done by a government entity authorized by justice, to investigate suspicions of corruption or terrorist acts) or not. They can also be done through an out-of-band device (a voice recorder, a hidden micro-camera or by intercepting and listening to voice conversations (calls) or text conversations (SMS) from the mobile phone service provider). According [9], for this case, there is Eavesdropping on non-encrypted message content and Eavesdropping on calling.

Flooding attacks: when flooding, the attacker attempts to negatively impact of the service or resource availability from authorized users, by exploiting specific vulnerabilities or by sending overwhelming amounts of messages (e.g., SYN flooding attacks, User Datagram Protocol (UDP) flooding, Internet Control Message Protocol (ICMP) flooding) on the server. In a DoS scenario, the attacker often attempts to disable the network or services by uninterruptedly sending data packets to the target server from only one device. On the other hand, DDoS amplify that activity by using multiple nodes and even different networks that were compromised beforehand. DDoSs generate more traffic than a traditional DoS attacks, and from different sources, many of them are legitimate (infected) users [12,37,58]. As can easily be seen, this type of attack falls well into both the communications category and the software category, as it targets the network, transport and data connection and Cloud-based mobile applications whose architecture is client–server, the main consequence being the unavailability of software whose paradigm is SaaS.

Botnets attacks: a botnet is a set of devices (PC, smartphone, tablet, IoT) compromised by a malicious entity (i.e., the bot master) that can remotely control such devices for the purpose of conducting other types of attacks (e.g., denial of service, malware or SPAM distribution). Traynor et al. [173] are considered to be the pioneer in the study of the theoretical potential impact of mobile device botnets in telephone networks, in 2009. In a mobile botnet scenario, the objectives of an attacker are to execute commands, disseminate malware code and expand the bot network [84].

Byzantine attacks: Byzantine attacks target routing protocols, where two or more routers collude in order to download, fabricate, modify or divert packets, aiming to disrupt the routing service in a Mobile Ad Hoc Network [190]. Furthermore, given its nature as a wireless and decentralized transmission medium, most routing protocols are vulnerable to other types of attacks besides Byzantine, namely gray hole attack, sinkhole attack, wormhole attack, sleep deprivation attack and black hole attack [76]. Byzantine attacks are critical in the use of mobile devices in military operations scenarios and in clinical fields.

DoS (cellular) jamming attack: the purpose of the DoS Jamming attack is to disrupt mobile network communications by blocking, interrupting, or making unavailable communications between mobile devices and BST. This results in the unavailability of services provided by the mobile telephony and Internet service provider. In addition to DoS, it can also cause rapid discharge of the battery of mobile devices [122]. In the latter case, we would be in the presence of a resource depletion attack.

Bluejacking and bluesnarfing attacks: DDoS-type attacks that target a Bluetooth wireless network in order to render it useless or near useless. It usually occurs through an attack coming from a connection of malicious entities in a target network. If the attacker is able to send unsolicited messages to the target devices via the Bluetooth connection, then we are in the presence of Bluejacking. In a scenario where the attacker gains unauthorized access to information from Bluetooth-enabled mobile devices using the OBject EXchange (OBEX) protocol, we would be in the presence of the Bluesnarfing attack [134]. Through the Bluejacking attack, attackers can send unwanted sounds, videos to other Bluetooth-enabled devices. Bluesnarfing attack consists of using Bluetooth connection for the purpose of stealing sensitive information (contacts, e-mails, passwords, photographs and other useful data) from wireless devices such as smartphones, tablets and IoT devices.

On-off attacks: this type of attack targets a wireless sensor network (WSN), aiming to disrupt a trust redemption scheme, behaving alternately as a good or bad entity, in order

[6] What differs eavesdropping from a sniffing attack is the target. While espionage targets eavesdropping on the raw audio source of a conversation between two or more entities, sniffing involves listening in on conversations (written, audio or video) from a network-based communication channel, such as IP traffic.

to ensure immediate trust redemption before another attack occurs. The on-off attack can result in the disruption or DoS and the consumption of node power in a WSN by means of fake injection attacks, capable of injecting huge amounts of fake data packets [112]. Finally, since there is a vulnerability that results from aggressive attacker behavior, which when exploited makes the system unable to distinguish a bad behavior from a temporary error, malicious nodes have more opportunities to attack the WSN by virtue of them staying active most of the time [26,65,72,189,192].

Cellular rogue base station (CRBST) attacks: A CRBST attack is a security threat aiming exploit the radio interface between smartphones and base stations, potentially launching passive or active attacks against user equipment. Such attacks range from acquiring the International Mobile Subscriber Identifier (IMSI) of subscribers, DoS, leaking private information on 4 G networks and eavesdropping [83].

Rogue access points: The access points (AP) in a Wi-Fi networks might be subject to the attack of AP spoofing, usually called Rogue Access Point (RAP) [10]. This attack consists of cloning the Media Access Control (MAC) address and Service Set IDentifier (SSID) of an AP, leading to the appearance of a fake access point posing as a genuine one, leading users to connect to this new network as if they were connecting to the genuine network. If the attack is successful, an attacker has the ability to eavesdrop on communications and hijack a communication of a given target user, redirect them to malicious websites, and steal their login credentials.

Access point hijacking attacks: This type of attack is a variant of the session hijacking attack and targets the AP access credentials of legitimate administrators [80]. These credentials can be extracted, e.g., through a sniffing, brute force or MitM attack. After this, the attacker is able to carry out other types of attacks, such as DoS and RAP.

NFC payment replay attacks: This type of attack targets the exploitation of vulnerabilities in the Europay Mastercard Visa (EMV) wireless communication protocol between the smartcard and the payment terminal, namely the authenticity of the payment terminal is not guaranteed to the payment device of the costumer and the banking data exchanged between the payment device (smartcard) and the point of sale terminal are not encrypted [110,140]. Such an attack occurs when an attacker re-transmits authentication-related communication between a payment device, VISA smartcard or Mastercard (RFID) and an automated payment terminal. Since the relay attack is stealthy, both the smartcard and the payment terminal are unaware of it. In such scenario, provided the attacker has knowledge and skills in radio-electronics, (s)he might be able to use an NFC reader (or an NFC smartphone) equipped with a special antenna in order to steal the victim sensitive smartcard data within a safe distance of a few meters from the payment terminal, even if it is in a briefcase, by falsifying the distance between the victim and the automatic payment terminal.

Wi-Fi SSID tracking attacks: This type of attack aims to obtain sensitive data (location, routine, trajectory, etc.) of users of mobile devices using Wi-Fi networks to access the Internet. Furthermore, it consists of using sophisticated sniffing devices to bypass authentication (for closed networks), extract and identify the MAC address of the mobile device and establish a match with its potential owner. In practice, this type of attack (which can be passive or active) exploits the vulnerability of constant radio signal emission from smartphones, tablets and IoT devices [113].

Wi-Fi jamming attacks: This is a denial-of-service attack that blocks the radio frequency, making access to the Wi-Fi network and consequently to the Internet unavailable [27,136]. Generally, two techniques are used to carry out this type of attack, namely: (1) Wi-Fi AP Flooding with deauthentication frames; (2) transmit high levels of noise in the radio frequency band of the Wi-Fi network.

GPS spoofing attacks: With the adoption and widespread use of mobile devices and their applications, many emerging technologies have gained notoriety and became ubiquitous. One of these is GPS, providing positioning, navigation and timing services. However, such adoption has brought several security challenges, most notably, the existence, in the civilian version of GPS, of the possibility of GPS spoofing. According [101], GPS spoofing attack consists of breaking authentication by forging satellite signals in order to provide incorrect location and timing data to the user, putting the user security at risk for several reasons.

GPS jamming attacks: This attack aims to interrupt or obstruct the communication between the emitting satellite and the device (smartphone/tablet) receiving the GPS signal. Normally, the attack consists of blocking the signal to the receiver, since the receiving signal is weaker compared to the broadcasting signal, and can be carried out in two different ways, namely blanket jamming and deception jamming [71].

Orbital jamming attacks: Low-orbit satellites came into existence in order to increase the quality of communications, as they decrease latency during message exchanges between two or more entities. However, they are susceptible to interference attacks in particular situations. Since the radio frequency of the front-end of a low-orbit satellite can easily be saturated with the use of a powerful jammer, this would result in disabling the link in the entire frequency band [179].

2.3.3 Social engineering-based attacks

Attacks in this category target the manipulation and exploitation of the human factor, i.e., human weaknesses of behavioral nature, inducing them to perform actions or disclose confidential information that benefit the adversary. They

also have the particularity of not requiring physical contact between the adversary and the victim.

Phishing attacks: a phishing attack is performed via manipulation and dissemination of a web link to attempt to redirect users to a site controlled by the malicious entity, e.g., designed to capture user information and account access, with the final objective of stealing sensitive data. Main attacks vectors are e-mail keyloggers through Trojan horses and man-in-the-middle attack of data proxies. The most common targets are Internet banking and online payment (e.g., Paypal or VISA) users [56]. On mobile devices, attackers can also use, e.g., Short Message System (SMS) or Multimedia Message System (MMS) to steal one-time passwords or change DNS resolution on the device.

Pharming attacks: Pharming is a special type of phishing attack or DNS poisoning attack in which the user is redirected to a fake website by changing the IP address on the DNS server [57,86]. The objective of the attack is the same as the phishing attack, i.e., theft of sensitive data and money from legitimate users of the applications.

Malicious QR code: Although this type of attack also affects the category or application layer, it also falls under the category of Social Engineering attacks as it mainly relies on human behavior [89]. For example, the victim may be tricked into reading a malicious QR code advertising a fake promotional campaign for a product from a supermarket car park, being redirected to a malicious site.

Malware-as-a-service: Social engineering is also a form of delivery of malware as a service, as many users of mobile social networking and instant messaging applications are not cyber-secure and exhibit risky behaviors when using these platforms. They are then easily tricked into clicking on malicious, supposedly well-intentioned links that appear as an advertisement or game, resulting in the stealthy installation of malware on the victim's mobile device.

2.3.4 Hardware-based attacks

Attacks in this category focus on exploiting the physical hardware (the chips, circuit boards, device ports, etc.) used in computer systems, aiming to replace, destroy, modify and exploit the hardware components that make up a system.

Reverse engineering attacks: REA attacks were previously mentioned as software-based attacks, but they also can apply to hardware. The purpose of reverse engineering attacks is maintained, as the purpose of gaining access to sensitive information from systems and users, e.g., how the system is built hardware-wise, usage of fixed function hardware, or retrieval of embedded cipher keys [16]. The two main approaches, White Box Reverse Engineering and Black Box Reverse Engineering, are also kept in terms of hardware-based reverse engineering.

Mobile SIM swapping attack: This attack is typically performed via the swapping of the SIM card of a victim. In such scenario, a cybercriminal, with a few important details about the life of the target, can potentially and correctly answer security questions, impersonate the victim, and convince the mobile carrier to reassign your phone number to a new SIM card. Notice that SIM swapping can happen remotely. At that point, the criminal can get access to some of the data associated with that SIM card, change account passwords to lock the victim out of online banking profiles, e-mail, etc.

Side-channel attacks: This type of attack, extensively described in Sect. 2.3.1 also targets hardware and has the particularity of being difficult to stop, given its stealth characteristics, in terms of operationalisation. According to Montasary et al. [121], there are three variants for this type of attack, namely Prime+Probe Attacks, Time-Driven Attacks and Access-Driven Attacks. Moreover, these attacks are aimed at leaking sensitive data from encrypted digital devices and electronic circuits. Finally this type of attack can be operationalized locally (physical access to the device) or remotely.

Rowhammer attacks: This type of attack targets Dynamic Random-Access Memory (DRAM), as it is vulnerable to memory perturbation errors—a high rate of accesses to the same address in DRAM reverses bits in data stored at nearby addresses. Furthermore, mobile devices and the Cloud are also vulnerable to such attacks [175,184]. Rowhammer attacks generate adversarial workloads that exploit perturbation errors for the purpose of inverting the value of safety-critical bits. And considered an attack with a high degree of dangerousness, given the fact that it is easy to mount and easy to scale [31]. A Rowhammer attack can result in the pollution of the system memory, accessing and altering sensitive data and obtaining total control of the system [121].

Hardware integrity attacks: This type of attack results from the corruption, counterfeiting or cloning of one or more physical components of a computer system by a malicious entity, leading to other types of attacks or threats, such as denial of service and leakage of the victim's sensitive data [157,177].

Spectre attacks: Spectre attacks are part of a class of microarchitectural attacks. Spectre attacks trick the processor into speculative execution of sequences of instructions that should not have been executed under correct program execution [93]. Such instructions are called transient instructions, because their effects can be reversed. In a scenario of a successful Spectre attack, the attacker is able to leak information from within the address space of the victim's memory.

Meltdown attacks: The is a microarchitectural attack that exploits out-of-order execution to leak core memory. Differs from a Spectre attack in terms of targeting and modus operandi [93,104].

2.3.5 Physical security-based attacks

In this category, attacks target physical security, aiming to exploit weaknesses in the physical parts of a system in an attempt to compromise it.

Bypassing physical security (BPS): These attacks consist of techniques aiming at circumventing or avoiding detection by physical security and building surveillance systems or methods to bypass electronic or physical locks protecting entry points [143]. It may be part of more complex attacks aimed at accessing, altering or destroying sensitive user information, or making a service or resource unavailable [11,143].

Device theft or loss: The main distinctive characteristic of this attack is that it consists of trying access and steal a physical target device in order to perform malicious actions, such as altering, deleting, leaking, inserting and destroying data, as well as stealing money through banking transactions, posing as the rightful owner [32]. Some specific instances of some attacks, such as, e.g., Man-in-the-Disk attack [42], where the physical external storage can be physically taken from the device after being compromised or used to divert sensitive data, can also be categorized here. The attacker can also simply destroy the device, preventing the user from accessing their data and the services provided in the form of an application as a service.

Malicious insider: This type of attack occurs when a malicious entity (e.g., client, employee, hypervisor, Cloud provider/corrector, etc.) takes advantage of its legitimate inside privileges to secretly perform malicious activities, such as information theft and data or physical infrastructure destruction. This type of attack also occurs from client to server, when the person, employee or team with insider knowledge on how the system is built can implant malicious code to fully destroy the software solution [4,74].

The taxonomy proposed in this paper revolves around five main domains: application, communication, social engineering, physical security and hardware.

Figure 4 presents an illustration of it in the context in which each of the attacks occurs. It allows extracting a mental picture of the stage of security threats in the Cloud and Mobile ecosystem and the identification of the main attack vectors in the different layers that constitute the considered ecosystem. Moreover, it also addresses threats or attacks on mobile telecommunications infrastructures. To the best of our knowledge, the proposed taxonomy is the most comprehensive ever developed, as it covers all ecosystems or technologies that give rise to the Cloud and Mobile ecosystem considered as a whole, namely the Internet, the Cloud and mobile.

3 Mechanisms against attacks and vulnerabilities in the Cloud and Mobile ecosystem

This section presents algorithms and mechanisms that can be used for countering the various threats and attacks on the Cloud and Mobile ecosystem. A general perspective over the mapping is provided in Table 5, following the proposals of various researchers. The idea is that these mechanisms are guaranteed by design of the applications of this ecosystem, thus embedded right from the inception.

3.1 Application-based attacks countermeasures

The following are brief descriptions of countermeasures that can be applied to prevent the aforementioned application-based attacks:

- *Session Fixation Attacks Countermeasures:* preventing these type of attacks can be achieved by (1) correctly coding web application by following OWASP best practices [123]; (2) implementing server-side measures against Session Fixation; (3) implementing a multilayered *defense in depth* model; (4) use of a ModSecurity Web Application Firewall (WAF) tool [52];
- *CSRF Attacks Countermeasures:* A solution against CSRF attacks is to never allow GET requests, only allow the HTTP methods `POST`, `PUT` or `DELETE`. Other countermeasures include using multifactor authentication, defining a session expiry date, restricting attachment uploads, personalizing HTTP header validation, applying input/output filtering, or validating standard inputs;
- *XSS Attacks Countermeasures:* mitigating this attack is possible by adequately filtering all unreliable data, whitelisting or positive input validation, considering the use of auto-sanitization libraries such as OWASP AntiSamy or Java HTML Sanitizer Project for rich content, allowing only trusted websites at the browser end, considering the Content Security Policy (CSP) in all the website or web application to defend it against XSS, and codifying outputs to neutralize dangerous characters;
- *SQL Injection Attacks Countermeasures:* strong input validation, enforced usage of parameterized SQL commands, use of validation and encryption mechanisms that filter and/or transform malicious inputs into trusted inputs and implement access control mechanisms that ensure compliance with the principle of least privilege and use of custom error pages are all countermeasures that can be used to prevent SQL injection;
- *Buffer Overflow Attacks Countermeasures:* writing correct code, set up non-executable buffers, implement array limit verification and code pointer integrity verification are all measures to counter buffer overflow. Other recom-

mendations include the use of programming languages thar are immune to buffer overflow vulnerabilities, such as Java or C#, or the detection and elimination of buffer overflow vulnerabilities from the source code, usually by performing some sort of static analysis on either the source code or on the compiled binaries [99] and practice defensive programming;

- *Code Inclusion Attacks Countermeasures:* mitigation of Code Inclusion attacks includes using input validation, sandboxing, firewalls and input sanitization of any user data;
- *Brute Force Attacks Countermeasures:* recommended countermeasures to brute force attacks include the implementation of authentication and session management mechanisms and encryption of user access credentials during storage, rest and transit using secure hybrid encryption schemes. In other words, the application should have a multifactor authentication, account lock mechanisms, use of CAPTCHA and security questions and memorable words, strong and complex passwords (alphanumeric characters, one capital letter, one small letter and one special character), the enforcing of the usage of a password with a length of no less than 8 and no more than 64 characters and forcing change of password after 3 months by blocking access, and usage of the FRESCO framework;
- *Flooding (DDoS) Attacks Countermeasures:* potential solutions to attenuate the effects of Flooding attacks include adopting Network Intrusion Detection System (NIDS), Host-Based Intrusion Detection System (HIDS), Network Intrusion (NIPS), Host-based Intrusion Protection System (HIPS) and Extended Detection and Response (XDR) in computer networks, using depth defense, using the best authentication and authorization paradigms, such as multifactor authentication, detection, filtering and encryption. Additionally, the DCI protocol and Flow-Checker frameworks can also be used. Elastic Cloud deployments help handling the initial impact of flooding attacks or even completely prevent their effects if the Cloud has substantially more resources than the attacker;
- *Cache Poisoning Attacks Countermeasures:* an effective countermeasure to Cache Poisoning or DNS Spoofing Attacks is using the Domain Name System Security Extensions (DNSSEC) [94];
- *Code Injection Attacks Countermeasures:* to eliminate the vulnerability of code injection and prevent its corresponding attack, the following can be adopted as countermeasures: defensive coding, sanitation of all client entries (e.g., using XSS Auditor [17], Bek [68], CSAS [146]) and ScriptGard [148] frameworks/mechanisms), validation of all content entries, force regular application of software patches, use safe APIs and limit the damage caused by the code injection attack using CSP, ConScript [117] and Escudo [78] frameworks;
- *Command Injection Attacks Countermeasures:* countering command injection attacks and eliminate the respective vulnerability can be achieved by validating and filtering all input provided by the user. Additionally, all the entries must be parameterized;
- *ALM Attacks Countermeasures:* mitigation techniques for ALM include using input and output validation mechanisms, implement secure and strong access control mechanisms, use synchronization, use security testing by static analysis to identify log forging vulnerabilities, avoid viewing logs with command-line shells, use of Trusted Platform Module (TPM) [141] and Mobile Trusted Module (MTM) [170];
- *Session Hijacking Attacks Countermeasures:* one-time cookie stateless method can be used with HTTPS to prevent session hijacking attacks; additionally, the use of encryption, authentication, session management mechanisms and access control mechanism all contribute as attenuation measures. Moreover, for native mobile applications, the use of authentication protocols resistant to session hijacking attacks, such as chaotic hash-based biometrics, asymmetric encryption function, elliptic curve cryptosystem, bilinear pairings, hashing function-based techniques and self-certified public keys, mutual authentication, hash-based remote fingerprint authentication scheme, and pattern recognition approaches are very best practices to apply in this context;
- *Malware-as-a-Service Attacks Countermeasures:* implementing network defenses such as firewalls, IDS, IPS, VPN, AVs, Anti-malware system, Anti-Spam system, application signing, authenticated, authorized remote management, Cloud-based crowdsourcing [133] and use of TPM should help attenuate Malware-as-a-Service attacks;
- *REA Countermeasures:* obfuscation tools and techniques are designed specifically to prevent reverse engineering attacks, but their effectiveness should be tested using tools such as IDA Pro and Hopper. At the time of writing, the most recommended obfuscation tools/techniques are Proguard, Dexguard, Android NDK and Google Play Licensing Check, code obfuscation [38] and hybrid obfuscation [8]. Additionally, the use of TPM is also recommended in order to mitigate REA in the Cloud environment;
- SSRF Attacks Countermeasures: mitigating SSRF attacks requires setting up and deploying controls and technologies such as segmentation of remote resource access functionality into separate networks, firewalls, network access control mechanisms, sanitization and validation of all user input data, and disable HTTP redirects [123];

- *VM Escape Attacks Countermeasures:* correct guest machine configuration, correct configuration of the interaction rules between host and guest machines, keeping flexibility to a minimum [77], use of HyperSafe, adherence or subscription to Trusted Cloud Computing Platforms (TCCPs) (e.g., TPM) [1], use of virtual trusted data centers (TVDs) [63], use of security frameworks and architectures [165], and restriction of access to VM resources through accessing control policies like Discretionary Access Control (DAC), Mandatory Access Control, Role-Based Access Control (RBA) [187], Object-Based Access Control (OBAC) and Task-Based Access Control (TBAC) [181] are all countermeasures to VM escape;
- *Side-Channel Attacks Countermeasures:* to mitigate side-channel attacks, [182] proposes deploying elliptic curve cryptosystem as well as a public key infrastructure (PKI). Protecting cryptographic implementations, protecting user input, preventing network traffic analysis, permissions, keyboard layout randomization, limiting Access or sampling frequency, noise injection and preventing microarchitectural attacks [159] are also proposed counters for these attacks. For this purpose, the use of the App Guardian [191], Slogger [155], SEAndroid [158], SemaDroid [185], FlaskDroid [25], PINPOINT [144], AuDroid [135], AppVeto [129] frameworks are recommended. In addition, S-Box access and SGX & ARM TrustZone, TPM are highlighted countermeasures against cache-based side-channel attack [33].

3.2 Communication-based attacks countermeasures

The following are descriptions of countermeasures and mitigation techniques that can be applied to prevent the communication-based attacks:

- *MitM attacks countermeasures:* preventing MITM attacks [28,79,87,166,183] include using a myriad of techniques such as hash-key-based fingerprinting remote authentication scheme, using Diffie-Hellman (DH) with a co-location verification phase, using a correctly configured Transport Layer Security (e.g., TLS v1.2 or TLS v1.3), using cryptography tools, e.g., *Dsniff, Ettercap, Wsniff, Airjack* and End-to-End Encryption (E2EE), using strong cryptographic algorithms (e.g., AES, Rivest-Shamir-Adleman (RSA), Secure Hash Algorithm (SHA), Elliptic Curve Cryptography (ECC)) that can ensure data integrity and confidentiality, i.e., data origin authentication through digital signature and Hash-based Message Authentication Codes (HMACs), and using strong mutual authentication to always fully authenticate both ends of any communications channel;
- *Sniffing Attacks Countermeasures:* using sniffer detection techniques, e.g., based on Round-Trip Time (RTT), using access control and authorization and encryption mechanisms can aid in detecting (and mitigating) sniffing attacks;
- *Eavesdropping Attacks Countermeasures:* several technological or human centric techniques can help attenuate or mitigate the effects of eavesdropping attacks, and a panoply of those techniques should often be combined in order to be effective. They include secure authentication, authorization and access control mechanisms, applying patches on installed applications as soon as new updates become available, encrypt local and remote storage, use E2EE scheme for every message exchanged between two or more entities, do not disclose confidential data in conversations in public places, physically disable the microphone and camera on the devices if not in use, use of anti-virus and other security monitoring and detecting tools, and remain alert to unusual behavior or activity of installed applications;
- *Spoofing Attacks Countermeasures:* strong remote authentication and authorization mechanisms that must be applied to control the access to confidential information stored in the Cloud, such as those described to mitigate MitM attacks, can also be applied to prevent spoofing attacks;
- *Botnet Attacks Countermeasures:* using the best authentication and authorization paradigms, such as multifactor authentication, using a HIDS, NIDS and IPS integrated into a single Framework, and implementing the best cryptographic schemes that ensure authentication and authorization are countermeasures that can mitigate a botnet attack;
- *Byzantine Attacks Countermeasures:* ensuring that communications are encrypted, correct and authentic is an effective means to counter Byzantine Attacks. To this end, it is necessary to implement security mechanisms that guarantee these properties, namely secure encryption schemes and protocols, use of HMAC or digital signature to guarantee data integrity, and secure authentication schemes (e.g., hybrid multifactor authentication). Additionally, the use of technologies/techniques such as REST, PKI, Universally Unique Identifier (UUID) authentication, TLS and Security Enhanced (SE) -Floodlight [137] is also recommended;
- *VM Migration Attacks Countermeasures:* countermeasures include the application of network security mechanisms, such as remote attestation and security cryptography in the communication channel [108], or the use of

secure VM migration frameworks, such as PALM[7] and VNSS[8] [63];

- *DoS Jamming Attacks Countermeasures:* there are four countermeasures to Smart Jamming attacks against LTE CS-RS, LTE physical channel (PBCH, PCFICH, PUCCH, PRACH LTE), namely repeated game learning algorithm, pilot boosting, change of eNB Frequency and Timing and Research on smart jamming impact on a multi-cell configuration, having DoS as target. Monitoring of excess PUCCH energy, monitoring of eNB BER based on CQI value and dynamic PUCCH sizing are mitigations against correlated jamming attacks (Protocol aware Jamming), having LTE physical layer DoS as target [114]. Additionally, authentication, digital signature, cryptography and spread spectrum mechanisms should be implemented. Finally, more recently, three holistic solutions against jamming attacks have been proposed, namely jamming-resistant receivers for the massive MIMO uplink [46], Jamming detection in massive MIMO systems [5], Massive MIMO pilot re-transmission strategies for robustification against jamming [45].
- *Bluejacking and Bluesnarfing Attacks Countermeasures:* use of piconet system protection mechanisms at all protocol levels, enable the use of a PIN with embedded 128-bit keys, use long random PIN authentication, disable Bluetooth when it is not in use, particularly in public spaces, use newer versions of Bluetooth systems (from version 2.1 onwards), use Secure Simple Pairing (SSP) authentication protocols instead of the old PIN, and never accept transmissions of dubious or unknown origin are countermeasures for Bluejacking and Bluesnarfing attacks.
- *On-Off Attack Attacks Countermeasures:* the trust joint light probe-based defense (TLPD) scheme, proposed by Liu et al. [106], is a potential solution to counter On-Off attacks;
- *CRBS Attacks Countermeasures:* the best strategy to counter CRBS attacks is to implement a mechanism that identifies a fake BS from an authentic one. To this end, three approaches can be followed. Firstly, making the provision of a database of reliable base stations by mobile Internet service providers for reference. Next, the mobile device would determine the relative position of the BS through the received signal strength. Finally, the reliability of the BS would be assessed by comparison to others. On other hand, Shaik et al. [153] proposed a solution that fits into two categories, namely Fixing LTE protocols and Self-Organizing Network (SON) with intelligence.
- *RAP Attacks Countermeasures:* to counter this type of attack, a combination of several techniques or mechanisms can be used, such as hybrid RAP detection framework, Elimination [109], Multi-agent [160], Distributed Wireless Security Auditor (DWSA) [109] and RogueAP-Detector[9];
- *AP Hijacking Attacks Countermeasures:* strong encryption with authentication schemes or cipher modes (e.g., AES-GCM-256), secure encryption protocols, authentication (WPA2) and access control mechanisms are countermeasures against AP hijacking attacks;
- *NFC Payment Replay Attacks Countermeasures:* countermeasures to NFC payment replay attack consist of proximity verification and mutual authentication of the communication between the payment device (smartcard or smartphone) and the payment terminal through proximity tokens and key exchange protocol [126]. Furthermore, data shared between the smart card and the payment terminal and between the smartphone and the mobile phone operator must be encrypted using secure cryptographic schemes;
- *Wi-Fi SSID Tracking Attacks Countermeasures:* to counter this type of attack, privacy preservation must be guaranteed using hybrid cryptographic schemes, anonymous identity-based encryption, full 802.11 link-layer protocol that obfuscates all transmitted bits, including identifiers, mixing zones, silence periods and dynamic modification of signal strength [113];
- *Wi-Fi Jamming Attacks Countermeasures:* among the various solutions proposed to mitigate Wi-Fi Jamming Attacks, the channel hopping scheme is known as the most practical approach. Moreover, it can allow packet delivery even in the presence of jammers [44]. This scheme has been improved and its new version has been presented by [43];
- *GPS Spoofing Attacks Countermeasures:* countermeasures to GPS spoofing attacks are divided into two main categories, namely detection and mitigation. For detection, methods based on power monitoring of signals, spatial processing for spoofing detection, spoofing discrimination using time of arrival (TOA), spoofing discrimination using signal quality monitoring (SQM), Code and Phase Rates Consistency Check must be used. On the other hand, (i) detection of the Vestigial Signal, (ii) using Adaptive Filtering For Spoof Cancellation, (iii) Receiver Autonomous Integrity Monitoring (RAIM) and (iv) Multi-Antenna Null Steering and Beam Forming Techniques [3] are typically used as countermeasures;
- *GPS Jamming Attacks Countermeasures:* the usage of filtering mechanisms, such as spatial filtering, adaptive frequency-domain filtering, adaptive time-domain

[7] Protection aegis for live migration of VMs that keeps the integrity and privacy during and after migration.

[8] A security framework that changes the security policies for each VM, and continuously provides protection through VM migration.

[9] https://github.com/anotherik/RogueAP-Detector.

filtering, time–frequency ($t - -f$) filtering, multiple short-time Fourier transform (MSTFT) technique [145] and BEACON filter based on optimal bounding ellipsoid (OBE) criterion [111] are GPS Jamming mitigation techniques;
- *Orbital Jamming Attacks Countermeasures:* blind source separation (BSS)-based approach [188] and Satellite Diversity [179] are mitigation measures against Orbital Jamming attacks.

3.3 Hardware-based attacks countermeasures

Below are countermeasures against hardware-based attacks:

- *Mobile SIM Swapping Countermeasures:* several countermeasures can be applied to mitigate SIM Swapping, namely (i) establish two-factor authentication (2FA) using authentication applications instead of codes sent by e-mail or SMS; (ii) be aware of phishing attempts, being careful with messages from people and organizations you do not know, even if the sender seems familiar (e.g., there may be typos in the name of the sender, company logo and throughout the message that are a good warning that the message should be delete); (iv) instigate people to never click on links in suspicious messages; and (iv), endorsing the usage of a password manager;
- *Side-Channel Attacks Countermeasures:* Countermeasures to side-channel attacks consist of combining hardware partitioning and a secure and efficient implementation of the AES algorithm, the AES New Instruction (AES-NI);
- *Rowhammer Attacks Countermeasures:* To counter Rowhammer attacks, Rowhammer-induced bit reversals should be blocked by modifying DRAM memory controllers and by using a high-bandwidth oscilloscope to physically probe the memory bus [121].
- *Hardware Integrity Attack Countermeasures:* The countermeasure against hardware integrity attacks consists of the use of Hardware-assisted Trusted Computing (TC) technologies through the TPM (i.e., Intel Trusted eXecution Technology (TXT), AMD Platform Security coProcessor (PSP)) [69] and Trusted Execution Environment (TEE) (i.e., Intel Software Guard eXtension (SGX), AMD Secure Encrypted Virtualization (SEV), ARM TrustZone). Furthermore, TC guarantees [34,40]:
  - Trusted Boot;
  - Sealed Storage;
  - Curtained Memory;
  - Attestation;
  - Integrity Measurement.
- *Spectre Attacks Countermeasures:* To counter Spectre attacks, the use of the following mitigations or methods is recommended: Dynamically Allocated Way Guard (DAWG) [47,91], SafeSpec [88], InvisiSpec [186], ConTExT [151], SpecCFI [96] and run-time detection of Spectre attacks on Intel's architecture using hardware/software events and machine learning [2];
- *Meltdown Attack Countermeasures:* O The Meltdown attack can be mitigated by using the enhanced KAISER [104] defense mechanism, giving rise to

three implementations on Linux, Windows and MacOS called Kernel Page-table Isolation (KPTI), KVA Shadow and Double Map, respectively. Additionally, the use of MeltdownDetector [6] and Meltdown attack run-time detection on Intel architecture using hardware or software events and machine learning is also recommended.

3.4 Social engineering-based attacks countermeasures

The following are descriptions of countermeasures and mitigation techniques that can be applied to prevent the attacks based on social engineering:

- *Phishing Attacks Countermeasures:* utilizing authentication (e.g., using PhishPreventer [23] Authentication Protocol) and immunity by incorporating verification technologies (e.g., Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM)) [67], implementing access control mechanisms, implementing cryptography schemes, e.g., symmetric, asymmetric or hybrid ciphers that use standard encryption algorithms, and data sanitization are mitigations to this type of attacks. Additionally, good cyber hygiene practices should be taught and applied by users when using cyberspace;
- *Malicious QR Code Attacks Countermeasures:* according to [97], potential countermeasures against Malicious QR Code include using visual QR codes, standardizing DS in QR codes, masking (as the black and white QR code module distribution compatible with specifications follows a specific pattern determined by a mask), using a framework or security mechanisms that allow the detection of malign and benign URLs, blacklist malicious URLs susceptible to a phishing attack using Google Safe Browsing (GSB), OpenPhish (OP), and PhishTank (PT), as QR code URLs are usually shortened [19,48,49].

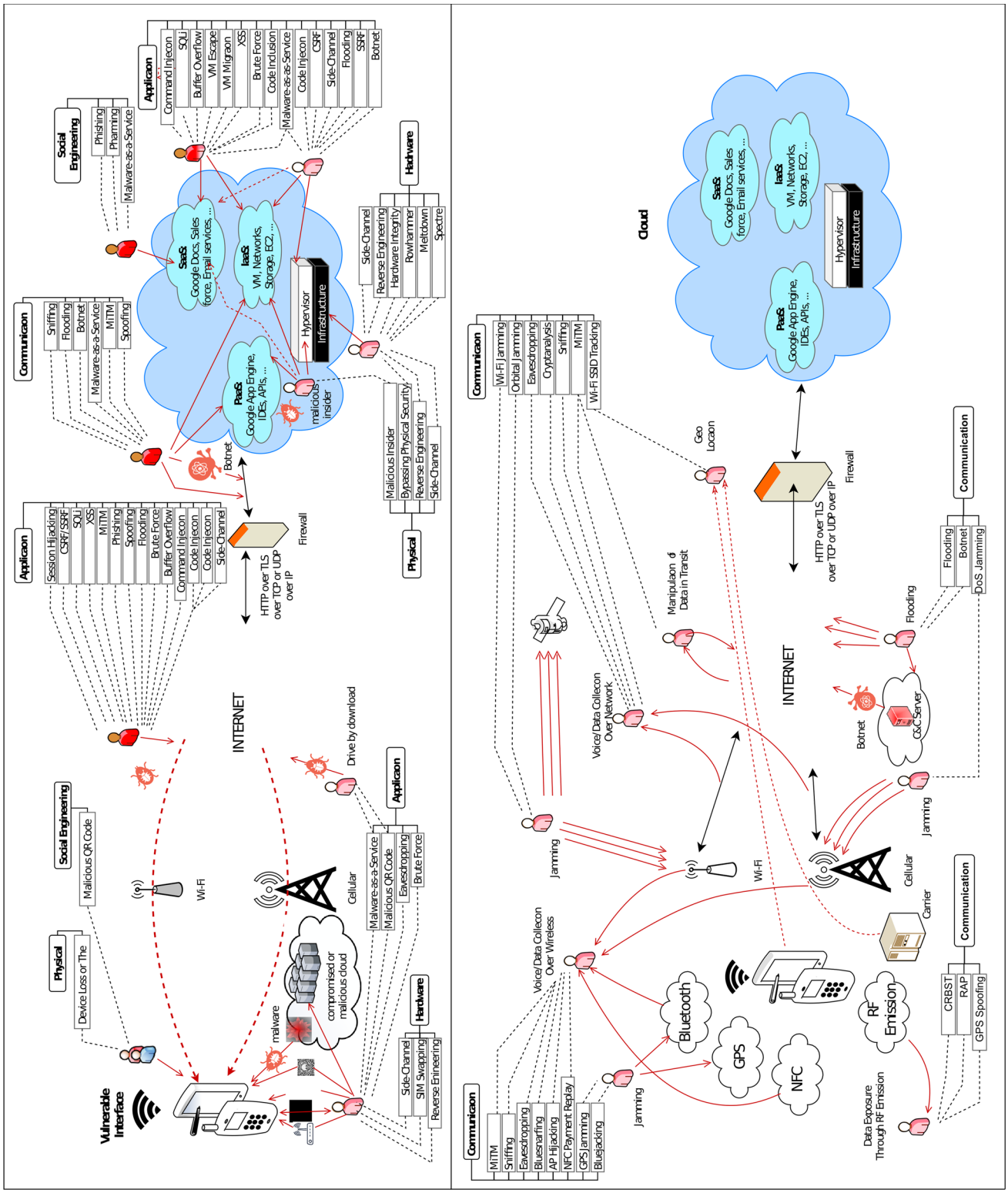

Fig. 4 Illustration of the taxonomy proposed in this article, showing the five main domains in context

3.5 Physical security-based attacks countermeasures

Brief descriptions of countermeasures and mitigation techniques that can be applied to prevent the attacks targeting physical security mentioned in the literature are as follows:

- *BPS:* implement intelligent access control mechanisms that are resistant to circumvention attempts, such as smart locks with a multifactor authentication system that combines two or more authentication paradigms, with biometrics being mandatory, are countermeasures against BPS;
- *Device Theft or Loss:* physical access control mechanisms, as well as intelligent monitoring and surveillance solutions are effective against this type of attack;
- *Malicious Insider Attacks Countermeasures:* mitigation techniques for this type of attack include implementing intelligent physical access and electronic surveillance mechanisms on data centers, implementing remote access control mechanisms to VMs, that ensure the minimal privilege principle, using cryptography and key management secure techniques that ensure sensitive data and key generation protection, and risk management and administration that guarantee policy execution and risk evaluation. Finally, it is recommended to adopt PSO, an ontological framework, proposed by Mavroeidis et al. [115] and TPM.

Finally, regardless of the countermeasures proposed by researchers and expert organizations in the form of best practices, we believe that security problems will only be effectively eradicated if Cloud and Mobile ecosystem applications are resistant to attacks or threats by design, and this will only occur when they are free from security vulnerabilities. This will only be possible when the security and privacy of the data of the users of the Cloud and system-based mobile applications are considered from the very beginning, even before the implementation of the applications, i.e., the data of the end users of the applications will only be secure as long as security mechanisms are incorporated during the entire engineering process of building systems and applications.

4 Tests and automation tools

This section presents the security tests for the Cloud and Mobile ecosystem, beginning with a general overview on security tests, and finishing with a specification of these tests for the specific ecosystem.

4.1 Security tests

With the widespread adoption of smartphones and tablets, software development has become more complex and certainly more distributed than ever, which in turn results in more software security issues. Embedding security into software means assuring that the software behaves correctly even if under a malicious attack, though software failures spontaneously occur in the real world—that is, without intentional prejudice [139]. There is a growing concern with security tests, given their preponderant role in software security.

Software security testing is the process used to identify if security resources implemented in software are consistent with their specification and construction. Software security tests can be divided in functional security tests and vulnerability security tests. This first ensures that security functions were correctly implemented, and are consistent with the security requirements. Software security requirements mainly include *confidentiality, integrity, availability, authentication, authorization, access control, auditing, privacy protection* or *security management*. Security vulnerability testing [171], on the other hand, attempts to discover vulnerabilities, often mimicking what an attacker would do. Vulnerabilities span from conception, implementation, operation and management faults or flows.

The difference between software security and secure software concerns thus the presence of an intelligent adversary that is attempting to attack the system [139]. Security is related to the information and services that are being protected, the abilities and resources of an attacker and the costs of potential solutions. It is an exercise in risk management. Risk analysis, particularly on the concept level, can help identify security problems and their impact. Once identified and classified, software risks can guide the software security testing process [139].

A vulnerability is an error or fault an attacker can exploit. There are different types of vulnerability, which lead to the creation of different taxonomies [100]. Security vulnerabilities in software systems can vary from local implementation errors (e.g., calling the `get()` function in C/C++) to much larger construction errors (e.g., a recovery system that, when failing, enters in an unsecure mode). Vulnerabilities can then be fitted into the implementation-level and project-level categories [116].

Project-level vulnerabilities are the hardest to address, while being the most prevalent and critical. The process of verifying if a given program has a vulnerability of this type is not an easy task neither for developers nor attackers, and requires experience, making their discovery harder to automate [139].

Table 5 Summary of the defense mechanisms for different types of attacks in the Cloud and Mobile ecosystem

ID	Attack	Defense mechanism
A1	Session fixation	Encryption, authentication and access control
A2	CSRF	Multifactor authentication, filtering, input validation, PUT, DELETE, POST
A3	XSS	Filtering, input/output validation, sanitization, CSP
A4	SQLi	Input validation, stored procedures, custom error pages
A5	MitM	Authentication, DH, HTTPS, Dsniff, Ettercarp, Wsnif, Airjack, E2EE, data origin authentication, TPM, MTM
A6	Cryptanalysis	Secure cryptographic schemes and cryptographic protocols, access control, secure backup
A7	Phishing	Authentication (e.g., PhishPreventer), immunity with SPF or DKIM, access control, encryption, cryptographic protocols, sanitization, awareness raising and training
A8	Sniffing	RTT-based sniffer detection, access control, encryption, HTTPS
A9	Eavesdropping	Authentication, access control, Up-to-date app and OS, secure backup, E2EE, camera and microphone off.
A10	Spoofing	Authentication, DH, TLS, Dsniff, Ettercarp, Wsnif, Airjack, E2EE, data origin authentication
A11	Pharming	Authentication, imunity with SPF or DKIM, access control, encryption, cryptographic protocols, sanitization
A12	Buffer overflow	Array limit and code pointer integrity verification, immunity, defensive coding, instruction set randomization
A13	Code inclusion	No passing user input, input validation, sanitization
A14	Brute force	Authentication, session management, encryption, secure backup using hybrid encryption (e.g., Blowfish and RSA)
A15	Flooding	NIDS, HIDS, NIPS, HIPS and XDR, authentication, authorization, filtering, encryption
A16	Cache poisoning	Using DNSSEC
A17	Code injection	Defensive coding, sanitization (XSS Auditor, Bek, CSAS, ScripGard), input validation, CSP, ConScript, Escudo
A18	Command injection	Input validation, filtering, parameterized inputs
A19	ALM	Input validation, authentication, access control, synchronization, static analysis, secure backup, encryption, TPM. MTM
A20	Session hijacking	one-time cookie stateless, encryption, authentication, access control
A21	Botnets	Authentication, access control, encryption, inspection (IPS, NIPS, IDS, NIDS, HIPS, HIDS, XDR)
A22	Byzantine attacks	Secure encryption schemes and protocols, MAC or DS, secure authentication schemes
A23	Side-channel	ECC, PKI, App Guardian, Slogger, SEAndroid, SemaDroid, FlaskDroid, PINPOINT, AuDroid, AppVeto, S-Box access and SGX & ARM TrustZone
A24	VM escape	VMM correct configuration, Hypersafe, TCCPs,, TVDs, DAC, RBAC, OBAC, TBAC, TPM
A25	VM migration	Remote attestation, encryption, access control, PALM, VNSS
A26	DoS jamming attack	Jamming-resistant receivers, Jamming detection in massive MIMO, Pilot re-transmission
A27	Malware-as-a-service	Firewalls, IDS, IPS, VPN, AVs, Anti-malware, Anti-Spam, application signing, secure remote management, TPM, MTM, Cloud-based crowdsourcing
A28	Malicious QR code	DS, QR code visual content, Machine learning with Naive Bayes algorithm, Phishing blacklists with PT, OP, GSB, masking

Table 5 continued

ID	Attack	Defense mechanism
A29	BPS	Access Control, smart locks, multifactor authentication (e.g., PIN, fingerprint), smartcards
A30	Device theft or loss	Locks doors, alarms, and monitoring
A31	Malicious insider	Access control, smart electronic surveillance, remote access control, encryption, PSO
A32	SIM swapping	2FA using authentication applications, be aware of phishing attempt, use a secure password manager
A33	REA	Obfuscation tools Proguard, Dexguard, Android NDK and Google Play Licensing Check, Obfuscation techniques, TPM, MTM
A34	SSRF	Segment remote resource access, access control, input validation, sanitization, URL schema and port, disable HTTP redirections
A35	Bluejacking	Authentication, Use long and random PIN codes, turn off Bluetooth when not using, secure Bluetooth, using the SSP protocols
A36	Bluesnarfing	Authentication, Use long and random PIN codes, turn off Bluetooth when not using, secure Bluetooth, using the SSP protocol
A37	On-off attack	TLPD
A38	CRBST	Fixing LTE protocols and SON with intelligence
A39	RAP	Hybrid RAP detection framework, Elimination, Multi-agent, DWSA [109] and RogueAP-Detector
A40	AP hijacking	Strong encryption schemes (AES-GCM-256), secure encryption protocols, authentication (WPA2) and access control mechanisms
A41	NFC payment replay	Proximity tokens by incorporating key exchange protocol, encryption, mutual authentication
A42	Wi-Fi SSID tracking	Anonymous identity-based encryption, full 802.11 link-layer protocol, encryption
A43	Wi-Fi jamming	Channel hopping scheme
A44	GPS spoofing	Detection (e.g., TOA, SQM) and mitigation (e.g., RAIM)
A45	GPS jamming	Filtering (e.g., MSTFT, BEACON)
A46	Orbital jamming	BSS-based approach and Satellite Diversity
A47	Rowhammer	Blocking of Rowhammer-induced bit reversals, using a high-bandwidth oscilloscope to physically probe the memory bus
A48	Hardware integrity	TPM (e.g., Intel TXT, AMD PSP), TEE (e.g., Intel SGX, AMD SEV, ARM TrustZone)
A49	Spectre	DAWG, SafeSpec, InvisiSpec, ConTExT, Real-time detection of Spectre attacks using Machine Learning
A50	Meltdown	KAISER, KPTI, KVA Shadow, Double Map, Run-time detection of Spectre attacks using Machine Learning, MeltdownDetector

4.2 Security tests techniques

Security tests for the Cloud and Mobile ecosystem have gained a strong traction in the academic and business communities, as the adoption of smartphones, tablets and Cloud computing has been growing in popularity. There are several works in this direction, with emphasis on Gao et al. [55], Murugesan et al. [124], Kong et al. [95], Knorr et al. [92] and Wang et al. [178]. An interesting characteristic of the majority of these papers is that they mainly refer to application security tests for the Android platform (explained by the prevalence of the mobile operating system). In terms of the type of analysis, tests are often segregated into *static* or *dynamic analysis*. *Static analysis* or testing does not require the execution of the program, contrary to *dynamic analysis* [178]. In terms of the knowledge of the system, tests can generally be of three types [95] namely: (i) *Black Box Testing*, which derives from an external description of the software, e.g., a description of its attack surface, and is used by individuals dedicated to finding vulnerabilities; (ii) *White Box Testing*, a more demanding approach derived from the source code of the software, which allows for a better coverage than *Black Box Testing*, typically used by application development companies; and (iii) *Grey Box Testing*, which results from combining *White* and *Black Box Testing*, trying to combine the best of both approaches, as some parts are derived from the general description, while others are based from the source code.

In terms of attacking, penetration testing or vulnerability testing techniques or method, tests can be classified as follows:

1. Attack Injection: a form of negative testing[10] that consists of the idea of purposely injecting faults in the security domain [53]. An intrusion results from combining an attack with one or more vulnerabilities. An attack is a failure that leads the system to an erroneous state or behavior and a security breach. So, the idea of injecting attacks consists in performing a large amount of attacks, and monitoring the status of the target software to detect if there are errors or faults. In case one of these conditions is detected, a vulnerability becomes known and tagged as the target for the injection attacks [53]. The project Attack Injection on Software Components (AJECT)[11] is an interesting work on the definition of an architecture of components for this type of injection. Its main components are *the attack injector* and *the monitor*. The first component performs the injection, while the second observes the target application so as to detect errors or faults [53].
2. Penetration Testing: experimental security tests done to identify invisible vulnerabilities in applications. Mainly used with permission to test cryptographic properties (e.g., authentication or confidentiality), to find vulnerabilities in the test environment and examine the performance of a system under extreme conditions [22]. There are two approaches for this type of test. The first relates to tests done by a team which is often external to the corporation—known as red team—and specializes in this type of tests, and has as its main purpose to demonstrate that the system has vulnerabilities. In this case, the team behaves as an attacker, as it does not possess knowledge of the system. This is known as *ethical hacking* [132]. On the other hand, when this task is performed systematically in an attempt to find the highest possible amount of vulnerabilities, usually with knowledge of the system, we are in the presence of the second approach. There are several tools that aid in performing penetration testing, most of which are specific for each category and objective of the test, platform or attack surface, as illustrated in Table 6.
3. Vulnerability Sweepers: applications or scripts that search for known vulnerabilities, e.g., a set of CVE[12] or OSVDB[13] vulnerabilities. The difference between a sweeper and an injector or fuzzer is tenuous, with the main difference being that the latter also search for unknown vulnerabilities. They are then used for different purposes. Sweepers are used to check that known vulnerabilities are not present, while fuzzers and injectors search for known and new vulnerabilities, either by developers, bounty hunters, or hackers. Sweepers require the existence of a vulnerability database, where the attacks are clearly defined; injectors and fuzzers insert semi-random inputs into interfaces, and contain heuristics to help guide the injection process.
4. Proxies: contrary to the previously mentioned tools, proxies allow performing security tests without any knowledge of the system or application. The modus operandi is based on modifying communications in and out of the application, instead of directly injecting inputs in the target. A simple example of this type of a tool is the Interactive TCP Relay (ITR), which enables the interception and modification of TCP/IP communications in a client–server setting [13]. WebScarab,[14] ZAP[15] and Paros[16] are examples of sophisticated open-source proxies. They are, however, directed toward web applications, contrary to

[10] *Negative Tests* verify if the software does not do what it should not do through requirements in the shape of *"The system should not..."*; security tests are negative tests if they are specified with this type of requirement.

[11] http://aject.di.fc.ul.pt.

[12] https://cve.mitre.org.

[13] https://blog.osvdb.org.

[14] https://github.com/OWASP/OWASP-WebScarab.

[15] https://www.zaproxy.org.

[16] http://www.parosproxy.org.

ITR, which works over any TCP/IP communication. Also, several functionalities, such as modifying a form field in a web page, are available in common web browsers (e.g., DevTools in Chrome or the Firebug plugin on Firefox). In the mobile applications context, other tools such as Wireshark,[17] tPacketCapturepro,[18] NMAP,[19] Nessus[20] or Metasploit Framework[21] can also be mentioned.

5. Mobile forensics: according to [178], mobile forensics gives companies and users legitimate tools for data analysis and recovery in a mobile device. Tools such as XRY, from Micro Systemation, or Celebrite UFED Touch Ultimate, enable both logical and physical data extraction from a wide variety of devices, even if the data was erased. This type of tools scans for a large amount of data, such as SMS/MMS, e-mails, phone registries, calendar, web traffic, bookmarks, pictures, voicemail, location information, application data, etc. [147,174].
6. Fuzzers: as previously mentioned, fuzzers have similarities with attack injections, as both based on injecting inputs. Nonetheless, according to Sutton et al. [167], fuzzing consists in *"(...) making vulnerability discoveries through brute force,"* while attack injection attempts to be a scientific methodology where the degree of vulnerability is discovered by searching specific vulnerabilities with already known characteristics. Moreover, attack injection includes a monitoring component, usually not present in fuzzers. Finally, fuzzers usually inject malformed inputs until they crash the target application [105]. Additional information on fuzzing can be found in [59]. Mangle and FileFuzz are examples of tools that try to fuzz file formats, WSFuzzer is used for diverse network protocols, and SPIKE and Sulley are generic fuzzers. Finally, sqlmap[22] is an advanced fuzzer, used to discover SQL injection vulnerabilities in web applications. Sqlmap also allows the user to attack the discovered vulnerabilities, being considered an automatic SQL injection and database takeover tool.

4.3 Security testing tools

In this survey, security tests on the Cloud and Mobile ecosystem are organized according to parameter, approach, method and test tool, and the two main mobile device platforms. Table 6 presents the main security test tools taxonomy for attacks on the Cloud and Mobile ecosystem. It also contains the vulnerabilities or parameters to be tested, the potential attacks in case the vulnerability is not eliminated, the type of test, the type of analysis, the test method and the mobile platform to which it applies, serving as a wrap up for the above discussion.

For an application of the Android platform, the testing process is simpler to implement, given the nature of the security paradigm of this platform. Thus, there are some assumptions that must be taken into account, namely *Host Device* (Windows, Linux, MacOS), *Android Studio* (which comes with the Android SDK), *Android NDK*, *Scrcpy* (to display or control devices from the computer) and at least one *Android mobile device*. In the case of the iOS platform, the test can only be carried out from a MacOS, while considering the following five assumptions, namely *MacOS host computer with admin rights, Xcode and Xcode Command Line Tools installed, Wi-Fi network that permits client-to-client traffic*, at least one *jailbroken iOS device* (of the desired iOS version) and *Burp Suite* or *other interception proxy tool*.

It is increasingly important to emphasize security test automation in this context, mainly when trying to analyze complex projects (which is typically the case in the Cloud and Mobile ecosystem). This process is usually done resorting to static or dynamic analysis, in which case there are some parameters that need to be taken into account and should first answer the *"what to test?"* question. The target is defined as the security vulnerabilities in the application. Furthermore, it is a good practice to structure the test in parts, e.g., as proposed by OWASP [123], which argues that a penetration test is composed of five phases, namely *Preparation, Intelligence Gathering, Mapping the Application, Exploitation and Reporting*. According to [92], static analysis is based in information contained in system files (e.g., in the Android platform, these are contained in the `APK` file, including manifest and compiled code). Tests are usually performed in a personal computer or in the mobile device. In general, testing should be performed for the following vulnerabilities: V1—Proper SSL usage and Insecure TLS Protection, V2—Dynamic binary analysis: debugging, tracing, V3—Content providers, V4—Use of encryption, V5—Poor use of certificate parameters, V6—Code quality, 7—Add-ons, V8—Input Validation, V9—Malware and Privacy Scanners, V10—Data leakage, V11—Secure backup, logging and Insecure Data Storage, V12—Web Server connection, V13—Web Server Authentication, V14—Interception of network, V15—Authentication and Authorization, V16—Access Control, V17—Exploit Database Vulnerabilities, V18—Database frangibility scanner, V19—Find Bugs, V20—Input validation of user SID, V21—Mobile decryption, unpacking & conversion, V22—Data leakage and Breach, V23—DoS and DDoS Attacks, V24—Run-time manipulation: code injection, patching, V25—Static binary analysis: disassembly, decompilation. These vulnerabilities were chosen taking into consideration

[17] https://www.wireshark.org.
[18] https://tpacketcapture-pro.soft112.com.
[19] https://nmap.org.
[20] http://www.nessus.org.
[21] https://www.metasploit.com.
[22] http://sqlmap.org.

Table 6 Taxonomy of the tools for Cloud and Mobile ecosystem applications security testing

Test parameter	Attack	Test types	Test analysis	Method	Tools		
					Both	Android	iOS
V1, V4	A5, A9, A23, A24, A42, A39, A38, A8, A6, A19, A22, A37, A14, A49, A50, A43	White Box	Static analysis	Mobile forensics	XRY, UFED Touch, OpenSSL	AndroGuard, MalloDroid, apktool, Amandroid	
V14		Grey Box	Hybrid	Penetration Testing	Burp Suite, Wireshark, bettercap		
V14		Black Box	Dynamic Analysis	Proxy	mitm-relay, Kali Linux, Burp Suite, Wi-Fi Framework [150], WPAxFuzz [82]		
V5		Black Box	Dynamic analysis	Proxies	NMAP, Nessus, SuperScan, Metasploit Framework		
V10		Grey Box	Dynamic analysis	Proxies	Wireshark	tPacketCapturepro, AFWall+	
V11		Grey Box	Dynamic Analysis	Proxies, Penetration Testing	Frida	adb	PassFab iPhone Backup Unlocker
V12		Black Box	Manual Dynamic Analysis	Proxies	ITR, OWASP WebScarab, OWASP ZAP, Paros		
V13		Black Box	Dynamic Analysis	Proxies	Wireshark, CERT Tapioca	tPacketCapturepro	
V15	A1-A4, A7, A10, A11, A16-A18, A20, A21, A27, A34, A38-A40, A44	Grey Box	Dynamic analysis	Vulnerability Scanner	OWASP WebScarab, Nikto, Wikto, Paros Proxy, Spike Proxy, OWASP ZAP		

Table 6 continued

Test parameter	Attack	Test types	Test analysis	Method	Tools		
					Both	Android	iOS
V15, V16	A31, A14, A1, A20	Grey Box	Dynamic Analysis	Penetration Testing	NMAP, Kali Linux		
V17	A4, A18	White Box	Manual Dynamic Analysis	Penetration Testing	SQLite browser		Xcode, Xcode Command Line Tools
V1, V4		Grey Box	Dynamic Analysis	Proxies	OWASP WebScarab, OWASP ZAP	API monitor	
V18		Grey Box	Dynamic Analysis	Vulnerability Scanner	Rapid7 Nexpose, Vulnerability Manager Plus		
V19		White Box	Static Analysis	Bytecode Scanner	Bytecode Scanner, QARK		
		White Box	Static Analysis	Source code analyzer	PARASOFT C/C++ TEST, RATS, Clang Code Analyze, Androbugs	Angr	
		White Box	Static Analysis	Binary code Scanner	BlackBerry Jarvis, CodeSonar for Binaries, BugScam, SAST, BugScam		
V20		Grey Box	Manual Dynamic Analysis	Checking input fields in GUI			
V22	A29, A30, A25	White Box	Static Analysis	Mobile forensics	BlackBag Blacklight, Encase Forensics, Oxygen Forensic Suite	AndroGuard, Drozer, SpotBugs, Andriller	Elcomsoft iOS Forensic Toolkit
V4, V11	A9, A27, A29, A23, A21, A49, A50	White Box	Static Analysis	Mobile forensics	Slueth Kit, Autopsy Browser	AndroGuard, Drozer, apktool, Amandroid	iOSbackup

Table 6 continued

Test parameter	Attack	Test types	Test analysis	Method	Tools		
					Both	Android	iOS
V6	A12	White Box	Dynamic Analysis	Penetration Testing	W3af		
		White Box	Static Analysis	Bytecode Scanner	SpotBugs		
		White Box	Static Analysis	Source code analyzer	PARASOFT C/C++ TEST, RATS, Clang Code Analyze	Angr	
		White Box	Static Analysis	Binary code Scanner	SpotBugs, FxCop, BugScam		class-dump-z, frida-ios-dump, Damn Vulnerable iOS App
V21	A10, A9, A8, A21, A5, A15, A16, A33	White Box	Static Analysis	Penetration Testing	Ghidra	Dex2jar, JD-GUI, Dextra	Clutch
V21		Black Box	Static Analysis	Penetration Testing	MobSF	APKEnum	Damn Vulnerable iOS App
V25		Grey Box	Static Analysis	Manual (Reversed) Code Review	r2ghidra-dec, r2frida, Radare2		Hooper
V11, V22	A31, A8, A5, A9	Grey Box	Dynamic analysis	Proxies	Wireshark	tPacketCapturepro, AFWall+, adb	
		Grey Box	Dynamic Analysis	Penetration Testing	VASTO		
		White Box	Dynamic Analysis	Stressing Testing (fuzzing)	Webfuzz, Wfuzz		
		Grey Box	Dynamic analysis	Vulnerability Scanner	Acunetix, Web3af, Nikto, HP WebInspect		
		Grey Box	Dynamic Analysis	Penetration Testing	TCPDump, Wireshark	idb tool	
V12	A2-A4, A34, A18, A17	Black Box	Manual Dynamic Analysis	Proxies	ITR, OWASP WebScarab, OWASP ZAP, Paros		
V7	A7, A11, A21, A27	White Box	Static Analysis	Mobile forensics		Add-ons Detector	

Table 6 continued

Test parameter	Attack	Test types	Test analysis	Method	Tools		
					Both	Android	iOS
V23	A26, A43, A46, A45, A15	Black Box	Dynamic Analysis	Penetration Testing	NMAP, SlowBot Net, MetaSploit, LOIC, Kali Linux		
V8	A2-A4, A17-A19, A34, A13	Grey Box	Static Analysis	Mobile forensics	Bitdefender, Norton, McAfee, Kaspersky	SandDroid	
V13, V23	A9, A10, A16, A21	Grey Box	Hybrid Analysis	Proxies, Penetration Testing	Wireshark, Cydia Substrate	tPacketCapturepro	Cycript
V9	A27, A28, A21, A9, A10, A16, A41	Grey Box	Static Analysis	Mobile forensics	Bitdefender, Norton, McAfee, Kaspersky	Recap vulnerability scanner	
	A27	Grey Box	Dynamic Analysis	Malware Analysis and Detection		DroidScope, Androtomist	
V10		Black Box	Dynamic Analysis	Proxies	Wireshark	tPacketCapturepro	
V4, V15, V13, V16	A36, A35, A41, A22	Black Box	Dynamic Analysis	Penetration Testing, Proxies	Kali Linux, hcitool	NFCSpy	
V2	A2, A4-A6, A8, A10-A12, A14, A16, A18, A20, A21, A25, A40	Black Box	Dynamic Analysis	Penetration Testing		Introspy-Android	Introspy-iOS
V3	A27, A9, A21	White Box	Hybrid Analysis	Vulnerability Scanner	RMS	Drozer, Sieve	
V19, V24, V25	A47	Grey Box	Dynamic Analysis	Vulnerability Scanner		Drammer Test App	
V10-V16	A48	Grey Box	Dynamic Analysis	Vulnerability Scanner or Configuration Compliance of the Platform	Nessus		

both the literature [22,92] and the OWASP Mobile Security Application Testing Guide [123], from where this list of 25 vulnerabilities was compiled and adapted toward the purpose of aiding the definition of the taxonomy proposed herein.

5 Research challenges

History has been proving that security related problems and challenges are here to stay for many years, and several can be identified or deduced from the discussion in this survey. Most challenges concern the creation of mobile applications that are both secure and resilient to attacks and security threats. Tackling this challenge requires many others to be addressed in the meanwhile, namely the following two:

1. *Lack of a specific tool that allows software developers to incorporate security by construction for the Cloud and Mobile ecosystem*—in the digital realm, utility has been the main driver for progress, and security is typically left for the second plane. The consequence is that, nowadays, developers and practitioners might have a set of good, yet generic security practices, and some frameworks that focus on each of the subset ecosystems that compose the Cloud and Mobile ecosystem, but no specific frameworks that contemplates it as a whole, so as to eliminate the gap between security engineering and software engineering, to guarantee user security and data privacy by construction, i.e., during the software development process, given the component complexity of this ecosystem, and the heterogeneity of the technologies that compose it, such as platform (OS, data structure, programming language), hardware, characteristics (detection tools, multimedia tools, interaction means, network technology) or APIs. The same is applicable to the Cloud and its service delivery models (SaaS, PaaS, and Iaas).
2. *Heterogeneity of attack vectors*—as the Cloud and Mobile ecosystem lives of the combination of a wide panoply of different and complex technologies, it motivates the search for, and aggravates the existence of, several attack vectors, which makes the development of applicable and scalable security mechanisms difficult for a given architecture. The development of countermeasures is also conditioned by OS, platform, communication protocols, network technology, hardware resources and even secondary support services.

A future research direction includes another in-depth approach to attack modeling in the Cloud and Mobile ecosystem, given its importance in the secure application development process, and the construction and presentation of a prototype of a framework that guarantees the building security of mobile applications based on the Cloud. It is intended that the framework is made up of five main modules, namely security requirements, good security practices, security mechanisms, attack models, tests and automation tools. The framework should be targeted at developers which are not necessarily specialized in cybersecurity, guiding them via a series of questions into providing inputs regarding the functional and security requirements and specifications of the system being developed or to be developed. The purpose of the framework is to automate and guarantee that security modeling and engineering are fully integrated into the software engineering processes. It will also enable developing domain-specific knowledge that allows for more reliable environments.

6 Conclusion

The exponential growth of mobile devices and applications has not been accompanied by significant achievements concerning security issues or challenges. The lack of single standardization for mobile and Cloud platforms is one such problem. Furthermore, gaps between software engineering and security engineering continue to exist, with immediate consequences in the inability of incorporating security mechanisms in the development process of software specifically designed for the Cloud and Mobile ecosystem. This issue is further aggravated by the fact that developers involved in the development processes are not necessarily computer security experts. A survey over more than one hundred references from the specialized literature was used to build up a concise taxonomy for the security issues affecting this ecosystem, presented herein as a main contribution. The taxonomy is then the basis for the comprehensive description of practical security issues and for the extensive identification of tools or approaches to detect or mitigate them.

Supported by the contributions of these survey, we believe that the aforementioned gaps, or open issues, can be overcome through the adoption of automated tools (frameworks) that are easy to use in the sense that security skills are not essential or mandatory in the process of using and interpreting the results obtained through the framework. Considering each phase of the software development process, these frameworks should generate security requirements, attack models, specification of security tests and mapping of security mechanisms to ensure that Cloud-based mobile applications are secure by design.

Acknowledgements The authors wish to thank the Instituto de Telecomunicações. This work was performed under the scope of Project SECURIoTESIGN, with funding from FCT/COMPETE/ FEDER (Projects with reference numbers UIDB/50008/2020 and POCI-01-0145-FEDER-030657) and FCT research and doctoral grants BIM/n°32/2018-B00582 and SFRH/BD/133838/2017, respectively, and also supported by operation Centro-01-0145-FEDER-000019 - C4 - Centro de Competências

em Cloud Computing, co-financed by the European Regional Development Fund (ERDF) through the Programa Operacional Regional do Centro (Centro 2020), in the scope of the Sistema de Apoio à Investigação Científica e Tecnológica - Programas Integrados de IC&DT.

Funding Open access funding provided by FCT|FCCN (b-on).

Data availability Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest All the authors of this paper declare that he/she has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. AbdElRahem, O., et al.: Virtualization security: a survey. In: 2016 11th International Conference on Computer Engineering Systems (ICCES), pp. 32–40. IEEE, Cairo, Egypt (2016). https://doi.org/10.1109/ICCES.2016.7821971
2. Ahmad, B.A.: Real time detection of spectre and meltdown attacks using machine learning (2020). https://doi.org/10.48550/ARXIV.2006.01442
3. Ahmad, M., et al.: Impact and detection of GPS spoofing and countermeasures against spoofing. In: 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1–8. IEEE, Sukkur, Pakistan (2019). https://doi.org/10.1109/ICOMET.2019.8673518
4. Ahmed, A., et al.: Malicious insiders attack in iot based multi-cloud e-healthcare environment: a systematic literature review. Multimedia Tools Appl. **77**(17), 21947–21965 (2018)
5. Akhlaghpasand, H., et al.: Jamming detection in massive mimo systems. IEEE Wirel. Commun. Lett. **7**(2), 242–245 (2018). https://doi.org/10.1109/LWC.2017.2769650
6. Akyildiz, T.A., et al.: Meltdowndetector: a runtime approach for detecting meltdown attacks. Future Gener. Comput. Syst. **112**, 136–147 (2020). https://doi.org/10.1016/j.future.2020.05.017
7. Al-Fuqaha, A., et al.: Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun. Surv. Tutor. **17**(4), 2347–2376 (2015)
8. Al-Hakimi, A.M.H., et al.: Hybrid obfuscation technique to protect source code from prohibited software reverse engineering. IEEE Access **8**, 187326–187342 (2020). https://doi.org/10.1109/ACCESS.2020.3028428
9. Almaiah, M.A., et al.: Classification of Cyber Security Threats on Mobile Devices and Applications, pp. 107–123. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-74575-2_6
10. Alotaibi, B., Elleithy, K.: Rogue access point detection: taxonomy, challenges, and future directions. Wirel. Pers. Commun. **90**(3), 1261–1290 (2016). https://doi.org/10.1007/s11277-016-3390-x
11. Alsunaidi, S.J., Almuhaideb, A.M.: Security methods against potential physical attacks on smartphones. In: 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), pp. 1–6. IEEE, Riyadh, Saudi Arabia (2019). https://doi.org/10.1109/CAIS.2019.8769458
12. Amara, N., et al.: Cloud computing security threats and attacks with their mitigation techniques. In: 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp. 244–251. IEEE, Nanjing, China (2017). https://doi.org/10.1109/CyberC.2017.37
13. Appelt, D., Alshahwan, N., Briand, L.: Assessing the impact of firewalls and database proxies on SQL injection testing. In: International Workshop on Future Internet Testing, pp. 32–47. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-07785-7_2
14. Appelt, D., et al.: Behind an application firewall, are we safe from sql injection attacks? In: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), pp. 1–10. IEEE, Graz, Austria (2015). https://doi.org/10.1109/ICST.2015.7102581
15. Armbrust, M., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
16. Basile, C., et al.: A meta-model for software protections and reverse engineering attacks. J. Syst. Softw. **150**, 3–21 (2019). https://doi.org/10.1016/j.jss.2018.12.025
17. Bates, D., et al.: Regular expressions considered harmful in client-side xss filters. In: Proceedings of the 19th International Conference on World Wide Web, WWW'10, pp. 91–100. Association for Computing Machinery, New York, NY (2010). https://doi.org/10.1145/1772690.1772701
18. Begum, A., et al.: RFI and SQLI based local file inclusion vulnerabilities in web applications of Bangladesh. In: 2016 International Workshop on Computational Intelligence (IWCI), pp. 21–25 (2016). https://doi.org/10.1109/IWCI.2016.7860332
19. Bell, S., Komisarczuk, P.: An analysis of phishing blacklists: google safe browsing, openphish, and phishtank. In: Proceedings of the Australasian Computer Science Week Multiconference, ACSW'20. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3373017.3373020
20. Bhadauria, R., Sanyal, S.: Survey on security issues in cloud computing and associated mitigation techniques. Int. J. Comput. Appl. **47**(18), 47–66 (2012). https://doi.org/10.5120/7292-0578
21. Bhatia, T., Verma, A.: Data security in mobile cloud computing paradigm: a survey, taxonomy and open research issues. J. Supercomput. **73**(6), 2558–2631 (2017)
22. Bojjagani, S., et al.: Vaptai: A threat model for vulnerability assessment and penetration testing of android and ios mobile banking apps. In: 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), vol. 00, pp. 77–86. IEEE, San Jose, CA, USA (2018). https://doi.org/10.1109/CIC.2017.00022
23. Bojjagani, S., et al.: Phishpreventer: a secure authentication protocol for prevention of phishing attacks in mobile environment with formal verification. Procedia Comput. Sci. **171**, 1110–1119 (2020). https://doi.org/10.1016/j.procs.2020.04.119
24. Brian, Others: Owasp top 10 (2021). https://owasp.org/Top10
25. Bugiel, S., et al.: Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In: 22nd USENIX Security Symposium (USENIX Security 13), pp. 131–146. USENIX Association, Washington, D.C. (2013).

https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/bugiel
26. Chae, Y., et al.: Trust management for defending on-off attacks. IEEE Trans. Parallel Distrib. Syst. **26**(4), 1178–1191 (2015). https://doi.org/10.1109/TPDS.2014.2317719
27. Chatzoglou, E., et al.: How is your wi-fi connection today? dos attacks on wpa3-sae. J. Inf. Secur. Appl. **64**, 103058 (2022). https://doi.org/10.1016/j.jisa.2021.103058
28. Chen, C., et al.: A scalable transitive human-verifiable authentication protocol for mobile devices. IEEE Trans. Inf. Forensics Secur. **8**(8), 1318–1330 (2013)
29. Cho, J.S., Yeo, S.S., Kim, S.K.: Securing against brute-force attack: a hash-based RFID mutual authentication protocol using a secret value. Comput. Commun. **34**(3), 391–397 (2011). https://doi.org/10.1016/j.comcom.2010.02.029
30. Cisco: Cisco annual internet report. https://dl.acm.org/doi/pdf/10.5555/2206209 (2020)
31. Cojocar, L., et al.: Are we susceptible to rowhammer? an end-to-end methodology for cloud providers. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 712–728. IEEE, San Francisco, CA, USA (2020). https://doi.org/10.1109/SP40000.2020.00085
32. Colp, P., et al.: Protecting data on smartphones and tablets from memory attacks. SIGARCH Comput. Archit. News **43**(1), 177–189 (2015). https://doi.org/10.1145/2786763.2694380
33. Coppolino, L., et al.: Cloud security: emerging threats and current solutions. Comput. Electr. Eng. **59**, 126–140 (2017). https://doi.org/10.1016/j.compeleceng.2016.03.004
34. Coppolino, L., et al.: A comprehensive survey of hardware-assisted security: from the edge to the cloud. Internet Things **6**, 100055 (2019). https://doi.org/10.1016/j.iot.2019.100055
35. Council, F.C., DHS: Mobile security reference architecture (2013). https://s3.amazonaws.com/sitesusa/wp-content/uploads/sites/1151/2016/10/Mobile-Security-Reference-Architecture.pdf
36. Dacosta, I., et al.: One-time cookies: preventing session hijacking attacks with stateless authentication tokens. ACM Trans. Internet Technol. (2012). https://doi.org/10.1145/2220352.2220353
37. Daffu, P., Kaur, A.: Mitigation of DDOS attacks in cloud computing. In: 2016 5th International Conference on Wireless Networks and Embedded Systems (WECON), pp. 1–5. IEEE, Rajpura, India (2016). https://doi.org/10.1109/WECON.2016.7993478
38. Dalai, A.K., et al.: A code obfuscation technique to prevent reverse engineering. In: 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 828–832. IEEE, Chennai, India (2017). https://doi.org/10.1109/WiSPNET.2017.8299877
39. De Ryck, P., et al.: Automatic and precise client-side protection against CSRF attacks. In: Atluri, V., Diaz, C. (eds.) Computer Security - ESORICS 2011, pp. 100–116. Springer, Berlin (2011)
40. Demigha, O., Larguet, R.: Hardware-based solutions for trusted cloud computing. Comput. Secur. **103**, 102117 (2021). https://doi.org/10.1016/j.cose.2020.102117
41. Ding, B., et al.: Return-oriented programming attack on the xen hypervisor. In: 2012 7th International Conference on Availability, Reliability and Security, pp. 479–484. IEEE, Prague, Czech Republic (2012)
42. Diogenes, Y., Ozkaya, E.: Cybersecurity–Attack and defense strategies: counter modern threats and employ state-of-the-art tools and techniques to protect your organization against cybercriminals. Packt Publishing Ltd (2019)
43. Djuraev, S., Nam, S.Y.: Channel-hopping-based jamming mitigation in wireless lan considering throughput and fairness. Electronics (2020). https://doi.org/10.3390/electronics9111749
44. Djuraev, S., et al.: Channel hopping scheme to mitigate jamming attacks in wireless lans. EURASIP J. Wirel. Commun. Netw. **2017**(1), 1–12 (2017). https://doi.org/10.1186/s13638-016-0785-z
45. Do, T.T., Ngo, H.Q., Duong, T.Q., Oechtering, T.J., Skoglund, M.: Massive mimo pilot retransmission strategies for robustification against jamming. IEEE Wirel. Commun. Lett. **6**(1), 58–61 (2017). https://doi.org/10.1109/LWC.2016.2631163
46. Do, T.T., et al.: Jamming-resistant receivers for the massive mimo uplink. IEEE Trans. Inf. Forensics Secur. **13**(1), 210–223 (2018). https://doi.org/10.1109/TIFS.2017.2746007
47. Domnitser, L., et al.: Non-monopolizable caches: low-complexity mitigation of cache side channel attacks. ACM Trans. Archit. Code Optim. (2012). https://doi.org/10.1145/2086696.2086714
48. Downs, J.S., et al.: Behavioral response to phishing risk. In: Proceedings of the Anti-Phishing Working Groups 2nd Annual ECrime Researchers Summit, eCrime'07, p. 37–44. Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1299015.1299019
49. Egelman, S., et al.: You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI'08, p. 1065–1074. Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1357054.1357219
50. Elazhary, H.: Internet of things (iot), mobile cloud, cloudlet, mobile iot, iot cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. J. Netw. Comput. Appl. **128**, 105–140 (2019). https://doi.org/10.1016/j.jnca.2018.10.021
51. Ferrag, M.A., et al.: Authentication schemes for smart mobile devices: threat models, countermeasures, and open research issues. Telecommun. Syst. **73**(2), 317–348 (2020). https://doi.org/10.1007/s11235-019-00612-5
52. Folini, C., Ristić, I.: Open source web application firewall (2021). https://github.com/SpiderLabs/ModSecurity
53. Fonseca, J., et al.: Vulnerability attack injection for web applications. In: 2009 IEEE/IFIP International Conference on Dependable Systems Networks, pp. 93–102. IEEE, Lisbon, Portugal (2009)
54. Friedman, J., Hoffman, D.V.: Protecting data on mobile devices: a taxonomy of security threats to mobile computing and review of applicable defenses. Inf. Knowl. Syst. Manag. **7**(1–2), 159–180 (2008)
55. Gao, J., et al.: Mobile application testing: a tutorial. Computer **47**(2), 46–55 (2014)
56. Garera, S., Provos, N., Chew, M., Rubin, A.D.: A framework for detection and measurement of phishing attacks. In: Proceedings of the 2007 ACM Workshop on Recurring Malcode, WORM'07, pp. 1–8. ACM, New York, NY, USA (2007). https://doi.org/10.1145/1314389.1314391
57. Gastellier-Prevost, S., Laurent, M.: Defeating pharming attacks at the client-side. In: 2011 5th International Conference on Network and System Security, pp. 33–40. IEEE, Milan, Italy (2011). https://doi.org/10.1109/ICNSS.2011.6059957
58. Girma, A., et al.: Analysis of DDOS attacks and an introduction of a hybrid statistical model to detect DDOS attacks on cloud computing environment. In: 2015 12th International Conference on Information Technology—New Generations, pp. 212–217. IEEE, Las Vegas, NV, USA (2015). https://doi.org/10.1109/ITNG.2015.40
59. Godefroid, P., et al.: Learn fuzz: Machine learning for input fuzzing. In: 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 50–59. IEEE, Las Vegas, NV, USA (2017). https://doi.org/10.1109/ASE.2017.8115618

60. Grassi, P.A., et al.: Digital identity guidelines (2017). https://doi.org/10.6028/NIST.SP.800-63-3. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf
61. Halfond, W.G., et al.: A classification of sql-injection attacks and countermeasures. In: Proceedings of the IEEE International Symposium on Secure Software Engineering, vol. 1, pp. 13–15. IEEE, IEEE, Georgia, USA (2006)
62. Hansman, S., Hunt, R.: A taxonomy of network and computer attacks. Comput. Secur. **24**(1), 31–43 (2005). https://doi.org/10.1016/j.cose.2004.06.011
63. Hashizume, K., et al.: An analysis of security issues for cloud computing. J. Internet Serv. Appl. **4**(1), 5 (2013). https://doi.org/10.1186/1869-0238-4-5
64. Heer, T., et al.: Security challenges in the ip-based internet of things. Wirel. Pers. Commun. **61**(3), 527–542 (2011)
65. Heise, P., et al.: Self-configuring real-time communication network based on openflow. In: 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), pp. 1–6. IEEE, Rome, Italy (2016). https://doi.org/10.1109/LANMAN.2016.7548851
66. Herley, C., Florêncio, D.: Protecting financial institutions from brute-force attacks. In: S. Jajodia, P. Samarati, S. Cimato (eds.) Proceedings of The Ifip Tc 11 23rd International Information Security Conference, pp. 681–685. Springer US, Boston, MA (2008)
67. Hong, J.: The state of phishing attacks. Commun. ACM **55**(1), 74–81 (2012). https://doi.org/10.1145/2063176.2063197
68. Hooimeijer, P., Livshits, B., Molnar, D., Saxena, P., Veanes, M.: Fast and precise sanitizer analysis with BEK. In: 20th USENIX Security Symposium (USENIX Security 11). USENIX Association, San Francisco, CA (2011). https://www.usenix.org/conference/usenix-security-11/fast-and-precise-sanitizer-analysis-bek
69. Hosseinzadeh, S., et al.: Recent trends in applying TPM to cloud computing. Secur. Privacy **3**(1), e93 (2020). https://doi.org/10.1002/spy2.93
70. Howard, J.D., Longstaff, T.A.: A common language for computer security incidents (1998). https://doi.org/10.2172/751004. https://www.osti.gov/biblio/751004
71. Hu, H., Wei, N.: A study of GPS jamming and anti-jamming. In: 2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS), vol. 1, pp. 388–391. IEEE, Shenzhen (2009). https://doi.org/10.1109/PEITS.2009.5406988
72. Hu, Y., Dong, M., Ota, K., Liu, A., Guo, M.: Mobile target detection in wireless sensor networks with adjustable sensing frequency. IEEE Syst. J. **10**(3), 1160–1171 (2016). https://doi.org/10.1109/JSYST.2014.2308391
73. Hubczyk, M., et al.: Local and Remote File Inclusion, pp. 189–200. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-25355-3_17
74. Iqbal, S., Kiah, M.L.M., Dhaghighi, B., Hussain, M., Khan, S., Khan, M.K., Choo, K.K.R.: On cloud security attacks: a taxonomy and intrusion detection and prevention as a service. J. Netw. Comput. Appl. **74**, 98–120 (2016)
75. Jabiyev, B., et al.: Preventing Server-Side Request Forgery Attacks, pp. 1626–1635. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3412841.3442036
76. Jamal, T., Butt, S.A.: Malicious node analysis in manets. Int. J. Inf. Technol. **11**(4), 859–867 (2019). https://doi.org/10.1007/s41870-018-0168-2
77. Jasti, A., et al.: Security in multi-tenancy cloud. In: 44th Annual 2010 IEEE International Carnahan Conference on Security Technology, pp. 35–41. IEEE, San Jose, CA, USA (2010)
78. Jayaraman, K., et al.: Escudo: a fine-grained protection model for web browsers. In: 2010 IEEE 30th International Conference on Distributed Computing Systems, pp. 231–240. IEEE, Genoa, Italy (2010). https://doi.org/10.1109/ICDCS.2010.71
79. Jeong, Y.S., et al.: An efficient authentication system of smart device using multi factors in mobile cloud service architecture. Int. J. Commun Syst **28**(4), 659–674 (2015)
80. Jha, S., Ali, S.: Mobile agent based architecture to prevent session hijacking attacks in IEEE 802.11 wlan. In: 2014 International Conference on Computer and Communication Technology (ICCCT), pp. 227–232. IEEE, Allahabad, India (2014). https://doi.org/10.1109/ICCCT.2014.7001497
81. Kampourakis, V., et al.: Revisiting man-in-the-middle attacks against https. Netw. Secur. (2022). https://doi.org/10.12968/S1353-4858(22)70028-1
82. Kampourakis, V., et al.: Wpaxfuzz: sniffing out vulnerabilities in wi-fi implementations. Cryptography **6**(4), 53 (2022)
83. Karaçay, L., Bilgin, Z., Gündüz, A.B., Çomak, P., Tomur, E., Soykan, E.U., Gülen, U., Karakoç, F.: A network-based positioning method to locate false base stations. IEEE Access **9**, 111368–111382 (2021). https://doi.org/10.1109/ACCESS.2021.3103673
84. Karim, A., et al.: Smartbot: a behavioral analysis framework augmented with machine learning to identify mobile botnet applications. PLoS ONE **11**(3), 1–35 (2016). https://doi.org/10.1371/journal.pone.0150077
85. Karimi, K., et al.: Smart home-smartphone systems: threats, security requirements and open research challenges. In: 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), pp. 1–5. IEEE, Agadir, Morocco (2019)
86. Karlof, C., et al.: Dynamic pharming attacks and locked same-origin policies for web browsers. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS'07, pp. 58–71. Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1315245.1315254
87. Khan, M.K., et al.: More efficient key-hash based fingerprint remote authentication scheme using mobile device. Computing **96**(9), 793–816 (2014)
88. Khasawneh, K.N., Koruyeh, E.M., Song, C., Evtyushkin, D., Ponomarev, D., Abu-Ghazaleh, N.: Safespec: Banishing the spectre of a meltdown with leakage-free speculation. In: 2019 56th ACM/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE, Las Vegas, NV, USA (2019)
89. Kieseberg, P., et al.: QR code security. In: Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia, MoMM'10, p. 430–435. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1971519.1971593
90. Kieyzun, A., et al.: Automatic creation of SQL injection and cross-site scripting attacks. In: Proceedings of the 31st International Conference on Software Engineering, ICSE'09, pp. 199–209. IEEE Computer Society, Washington, DC, USA (2009). https://doi.org/10.1109/ICSE.2009.5070521
91. Kiriansky, V., et al.: Dawg: a defense against cache timing attacks in speculative execution processors. In: 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 974–987. IEEE, Fukuoka, Japan (2018). https://doi.org/10.1109/MICRO.2018.00083
92. Knorr, K., Aspinall, D.: Security testing for android mhealth apps. In: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 1–8. IEEE, Graz, Austria (2015). https://doi.org/10.1109/ICSTW.2015.7107459
93. Kocher, P., et al.: Spectre attacks: exploiting speculative execution. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 1–19.

IEEE, San Francisco, CA, USA (2019). https://doi.org/10.1109/SP.2019.00002
94. Kolkman, O., Gieben, R.: Dnssec operational practices. Tech. rep., RFC 4641, September (2006)
95. Kong, P., et al.: Automated testing of android apps: a systematic literature review. IEEE Trans. Reliab. **68**(1), 45–66 (2019). https://doi.org/10.1109/TR.2018.2865733
96. Koruyeh, E.M., Haji Amin Shirazi, S., Khasawneh, K.N., Song, C., Abu-Ghazaleh, N.: Speccfi: mitigating spectre attacks using CFI informed speculation. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 39–53. IEEE, San Francisco, CA, USA (2020). https://doi.org/10.1109/SP40000.2020.00033
97. Krombholz, K., et al.: Qr code security - how secure and usable apps can protect users against malicious QR codes. In: 2015 10th International Conference on Availability. Reliability and Security, pp. 230–237. IEEE, Toulouse, France (2015)
98. Kumar, R., K, I., Goel, A.K.: Automated session fixation vulnerability detection in web applications using the set-cookie http response header in cookies. In: Proceedings of the 7th International Conference on Security of Information and Networks, SIN'14, p. 351–354. Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2659651.2659718
99. Kuperman, B.A., et al.: Detection and prevention of stack buffer overflow attacks. Commun. ACM **48**(11), 50–56 (2005). https://doi.org/10.1145/1096000.1096004
100. Landwehr, C.E., et al.: A taxonomy of computer program security flaws. ACM Comput. Surv. **26**(3), 211–254 (1994). https://doi.org/10.1145/185403.185412
101. Larcom, J.A., Liu, H.: Modeling and characterization of GPS spoofing. In: 2013 IEEE International Conference on Technologies for Homeland Security (HST), pp. 729–734. IEEE, Waltham, MA, USA (2013). https://doi.org/10.1109/THS.2013.6699094
102. Lin, X., et al.: Threat modeling for CSRF attacks. In: 2009 International Conference on Computational Science and Engineering, vol. 3, pp. 486–491. IEEE, Vancouver, BC, Canada (2009)
103. Lindqvist, U., Jonsson, E.: How to systematically classify computer security intrusions. In: Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No.97CB36097), pp. 154–163 (1997). https://doi.org/10.1109/SECPRI.1997.601330
104. Lipp, M., et al.: Meltdown: reading kernel memory from user space. In: 27th USENIX Security Symposium (USENIX Security 18), pp. 973–990. USENIX Association, Baltimore, MD (2018). https://www.usenix.org/conference/usenixsecurity18/presentation/lipp
105. Liu, B., et al.: Software vulnerability discovery techniques: a survey. In: 2012 4th International Conference on Multimedia Information Networking and Security, pp. 152–156. IEEE, Nanjing, China (2012). https://doi.org/10.1109/MINES.2012.202
106. Liu, X., et al.: Defending on-off attacks using light probing messages in smart sensors for industrial communication systems. IEEE Trans. Industr. Inf. **14**(9), 3801–3811 (2018). https://doi.org/10.1109/TII.2018.2836150
107. Louw, M.T., Venkatakrishnan, V.N.: Blueprint: robust prevention of cross-site scripting attacks for existing browsers. In: 2009 30th IEEE Symposium on Security and Privacy, pp. 331–346. IEEE, Oakland, CA, USA, USA (2009). https://doi.org/10.1109/SP.2009.33
108. Luo, S., et al.: Virtualization security for cloud computing service. In: 2011 International Conference on Cloud and Service Computing, pp. 174–179. IEEE, Hong Kong, China (2011)
109. Ma, L., et al.: A hybrid rogue access point protection framework for commodity wi-fi networks. In: IEEE INFOCOM 2008—The 27th Conference on Computer Communications, pp. 1220–1228. IEEE, Phoenix, AZ, USA (2008). https://doi.org/10.1109/INFOCOM.2008.178
110. Madhoun, N.E., Pujolle, G.: Security enhancements in EMV protocol for NFC mobile payment. In: 2016 IEEE Trustcom/BigDataSE/ISPA, pp. 1889–1895. IEEE, Tianjin, China (2016). https://doi.org/10.1109/TrustCom.2016.0289
111. Mao, W.L.: Robust set-membership filtering techniques on GPS sensor jamming mitigation. IEEE Sens. J. **17**(6), 1810–1818 (2017). https://doi.org/10.1109/JSEN.2016.2558192
112. Masini, B.M., et al.: Vehicular networking for mobile crowd sensing. Ad Hoc Netw. **36**, 407–408 (2016). https://doi.org/10.1016/j.adhoc.2015.10.002
113. Matte, C.: Wi-fi tracking: fingerprinting attacks and countermeasures. Ph.D. thesis, Université de Lyon (2017)
114. Mavoungou, S., et al.: Survey on threats and attacks on mobile networks. IEEE Access **4**, 4543–4572 (2016). https://doi.org/10.1109/ACCESS.2016.2601009
115. Mavroeidis, V., et al.: A framework for data-driven physical security and insider threat detection. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1108–1115. IEEE, Barcelona, Spain (2018). https://doi.org/10.1109/ASONAM.2018.8508599
116. McGraw, G.: Software security. IEEE Secur. Privacy **2**(2), 80–83 (2004). https://doi.org/10.1109/MSECP.2004.1281254
117. Meyerovich, L.A., Livshits, B.: Conscript: specifying and enforcing fine-grained security policies for javascript in the browser. In: 2010 IEEE Symposium on Security and Privacy, pp. 481–496. IEEE, Oakland, CA, USA (2010). https://doi.org/10.1109/SP.2010.36
118. MITRE: Capec-141: Cache poisoning (2021). https://capec.mitre.org/data/definitions/141.html
119. MITRE: Common attack pattern enumeration and classification (2021). https://capec.mitre.org/data/definitions/3000.html
120. Moher, D., et al.: Preferred reporting items for systematic review and meta-analysis protocols (prisma-p) 2015 statement. Syst. Rev. **4**(1), 1–9 (2015)
121. Montasari, R., et al.: Cloud Computing Security: Hardware-Based Attacks and Countermeasures, pp. 155–167. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-60425-7_6
122. Moorthy, V., et al.: Security and privacy attacks during data communication in software defined mobile clouds. Comput. Commun. **153**, 515–526 (2020). https://doi.org/10.1016/j.comcom.2020.02.030
123. Muller, B., et al.: Owasp mobile security testing guide (2022). https://github.com/OWASP/owasp-mstg/releases/tag/v1.4.0
124. Murugesan, L., Balasubramanian, P.: Cloud based mobile application testing. In: 2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS), pp. 287–289. IEEE, Taiyuan, China (2014)
125. Nikiforakis, N., et al.: Sessionshield: lightweight protection against session hijacking. In: Erlingsson, Ú., Wieringa, R., Zannone, N. (eds.) Engineering Secure Software and Systems, pp. 87–100. Springer, Berlin (2011)
126. Njebiu, V., Kimwele, M., Rimiru, R.: Secure contactless mobile payment system. In: 2021 IEEE Latin-American Conference on Communications (LATINCOM), pp. 1–6. IEEE, Santo Domingo, Dominican Republic (2021). https://doi.org/10.1109/LATINCOM53176.2021.9647831
127. Oberheide, J., Jahanian, F.: When mobile is harder than fixed (and vice versa): demystifying security challenges in mobile environments. In: Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, HotMobile'10, p. 43–48. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1734583.1734595
128. O.S., J.N., Mary Saira Bhanu, S.: A survey on code injection attacks in mobile cloud computing environment. In: 2018 8th International Conference on Cloud Computing, Data Science

Engineering (Confluence), pp. 1–6. IEEE, Noida, India (2018). https://doi.org/10.1109/CONFLUENCE.2018.8443032
129. Osman, T., et al.: Securing applications against side-channel attacks through resource access veto. Dig. Threats Res. Practice (2020). https://doi.org/10.1145/3416124
130. OWASP, O.W.A.S.P.: Application threat modeling (2017). https://www.owasp.org/index.php/Application_Threat_Modeling
131. Page, M.J., et al.: The prisma 2020 statement: an updated guideline for reporting systematic reviews. Syst. Rev. **10**(1), 1–11 (2021)
132. Palmer, C.C.: Ethical hacking. IBM Syst. J. **40**(3), 769–780 (2001). https://doi.org/10.1147/sj.403.0769
133. Papamartzivanos, D., et al.: A cloud-based architecture to crowdsource mobile app privacy leaks. In: Proceedings of the 18th Panhellenic Conference on Informatics, PCI'14, p. 1–6. Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2645791.2645799
134. Patel, N., et al.: Investigating bluetooth vulnerabilities to defend from attacks. In: 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), pp. 549–554. IEEE, Ankara, Turkey (2021). https://doi.org/10.1109/ISMSIT52890.2021.9604655
135. Petracca, G., et al.: Audroid: Preventing attacks on audio channels in mobile devices. In: Proceedings of the 31st Annual Computer Security Applications Conference, ACSAC 2015, p. 181–190. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2818000.2818005
136. Pirayesh, H., Zeng, H.: Jamming attacks and anti-jamming strategies in wireless networks: a comprehensive survey. IEEE Commun. Surv. Tutor. **24**(2), 767–809 (2022). https://doi.org/10.1109/COMST.2022.3159185
137. Porras, P.A., et al.: Securing the software defined network control layer. In: NDSS. Network and Distributed System Security (NDSS) Symposium, San Diego, California (2015). https://www.ndss-symposium.org/ndss2015/ndss-2015-programme/securing-software-defined-network-control-layer/
138. Portela, F., Queirós, R.: Introdução ao Desenvolvimento Moderno para à Web. FCA Editora de Informática, Lda, Lisboa (2018)
139. Potter, B., McGraw, G.: Software security testing. IEEE Secur. Privacy **2**(5), 81–85 (2004). https://doi.org/10.1109/MSP.2004.84
140. Pourghomi, P., et al.: A proposed NFC payment application (2013). https://doi.org/10.48550/ARXIV.1312.2828
141. Proudler, G.: Introduction to trusted computing concepts and trusted platform module 2.0 (2016). https://www.trustedcomputinggroup.org/wp-content/uploads/Introduction-to-Trusted-Computing-Concepts-and-TPM-.pdf
142. Qamar, A., et al.: Mobile malware attacks: review, taxonomy & future directions. Future Gener. Comput. Syst. **97**, 887–909 (2019). https://doi.org/10.1016/j.future.2019.03.007
143. Rahman, A.F.A., et al.: Securing sensor to cloud ecosystem using internet of things (iot) security framework. In: Proceedings of the International Conference on Internet of Things and Cloud Computing, ICC'16. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2896387.2906198
144. Ratazzi, P., et al.: Pinpoint: efficient and effective resource isolation for mobile security and privacy (2019)
145. Rezaei, M.J.O.: New GPS anti-jamming system based on multiple short-time Fourier transform. IET Radar Sonar Navig. **10**(4), 807–815 (2016). https://doi.org/10.1049/iet-rsn.2015.0417
146. Samuel, M., et al.: Context-sensitive auto-sanitization in web templating languages using type qualifiers. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS '11, p. 587–600. Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2046707.2046775
147. Satrya, G.B., et al.: Android forensics analysis: private chat on social messenger. In: 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 430–435. IEEE, Vienna, Austria (2016)
148. Saxena, P., et al.: Scriptgard: automatic context-sensitive sanitization for large-scale legacy web applications. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS'11, p. 601–614. Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2046707.2046776
149. Scarfone, K.A., et al.: Sp 800-125. guide to security for full virtualization technologies. https://dl.acm.org/doi/pdf/10.5555/2206209 (2011). Accessed: 25 Sept 2020
150. Schepers, D., et al.: A framework to test and fuzz wi-fi devices. In: Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec'21, pp. 368–370. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3448300.3468261
151. Schwarz, M., et al.: Context: leakage-free transient execution (2019). https://doi.org/10.48550/ARXIV.1905.09100
152. Sequeiros, J.A.B.F., et al.: Attack and system modeling applied to iot, cloud, and mobile ecosystems: embedding security by design. ACM Comput. Surv. (2020). https://doi.org/10.1145/3376123
153. Shaik, A., et al.: On the impact of rogue base stations in 4g/lte self organizing networks. In: Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WiSec'18, pp. 75–86. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3212480.3212497
154. Shar, L.K., Tan, H.B.K.: Defeating sql injection. Computer **46**(3), 69–77 (2013). https://doi.org/10.1109/MC.2012.283
155. Shrestha, P., et al.: Slogger: smashing motion-based touchstroke logging with transparent system noise. In: Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WiSec'16, pp. 67–77. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939918.2939924
156. Sicari, S., et al.: Security, privacy and trust in internet of things: the road ahead. Comput. Netw. **76**, 146–164 (2015)
157. Sisejkovic, D., et al.: Deceptive logic locking for hardware integrity protection against machine learning attacks. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **41**(6), 1716–1729 (2022). https://doi.org/10.1109/TCAD.2021.3100275
158. Smalley, S., Craig, R.: Security enhanced (se) android: bringing flexible mac to android. In: NDSS, vol. 310, pp. 20–38. San Diego, CA, USA (2013)
159. Spreitzer, R., et al.: Systematic classification of side-channel attacks: a case study for mobile devices. IEEE Commun. Surv. Tutor. **20**(1), 465–488 (2018). https://doi.org/10.1109/COMST.2017.2779824
160. Sriram, V.S.S., Sahoo, G., Agrawal, K.K.: Detecting and eliminating rogue access points in IEEE-802.11 wlan—a multi-agent sourcing methodology. In: 2010 IEEE 2nd International Advance Computing Conference (IACC), pp. 256–260. IEEE, Patiala, India (2010). https://doi.org/10.1109/IADCC.2010.5422999
161. Statista: Number of mobile phone users worldwide from 2015 to 2020. https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/ (2020). Accessed: 27 Aug 2020
162. Su, Z., Wassermann, G.: The essence of command injection attacks in web applications. In: Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'06, pp. 372–382. Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1111037.1111070

163. Suarez-Tangil, G., et al.: Evolution, detection and analysis of malware for smart devices. IEEE Commun. Surv. Tutor. **16**(2), 961–987 (2014). https://doi.org/10.1109/SURV.2013.101613.00077
164. Suarez-Tangil, G., et al.: Evolution, detection and analysis of malware for smart devices. IEEE Commun. Surv. Tutor. **16**(2), 961–987 (2014). https://doi.org/10.1109/SURV.2013.101613.00077
165. Subashini, S., et al.: A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl. **34**(1), 1–11 (2011)
166. Sun, H., et al.: An efficient authentication scheme for access control in mobile pay-tv systems. IEEE Trans. Multimedia **11**(5), 947–959 (2009)
167. Sutton, M., et al.: Fuzzing: Brute Force Vulnerability Discovery. Pearson Education, Crawfordsville (2007)
168. Takabi, H., Joshi, J.B.D., Ahn, G.: Security and privacy challenges in cloud computing environments. IEEE Secur. Privacy **8**(6), 24–31 (2010). https://doi.org/10.1109/MSP.2010.186
169. Taleby, M., et al.: A survey on smartphones security: software vulnerabilities, malware, and attacks. Int. J. Adv. Comput. Sci. Appl. (2017). https://doi.org/10.14569/ijacsa.2017.081005
170. TCG: Tcg mobile trusted module specification (2010). https://trustedcomputinggroup.org/resource/mobile-phone-work-group-mobile-trusted-module-specification/
171. Tian-yang, G., et al.: Research on software security testing. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.190.4771&rep=rep1&type=pdf (2010). Accessed: 15 Aug 2020
172. Tommasi, F., et al.: Mobile session fixation attack in micropayment systems. IEEE Access **7**, 41576–41583 (2019). https://doi.org/10.1109/ACCESS.2019.2905219
173. Traynor, P., et al.: On cellular botnets: measuring the impact of malicious devices on a cellular network core. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09, pp. 223–234. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1653662.1653690
174. Umar, R., Riadi, I., Zamroni, G.M., et al.: Mobile forensic tools evaluation for digital crime investigation. Int. J. Adv. Sci. Eng. Inf. Technol. **8**(3), 949 (2018)
175. van der Veen, V., et al.: Drammer: deterministic rowhammer attacks on mobile platforms. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS'16, pp. 1675–1689. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2976749.2978406
176. Vogt, P., Nentwich, F., Jovanovic, N., Kirda, E., Kruegel, C., Vigna, G.: Cross site scripting prevention with dynamic data tainting and static analysis. In: NDSS, vol. 2007, pp. 12. Internet Society, San Diego, CA, USA (2007)
177. Wang, J., et al.: Hypercheck: a hardware-assisted integrity monitor. In: Jha, S., Sommer, R., Kreibich, C. (eds.) Recent Advances in Intrusion Detection, pp. 158–177. Springer, Berlin (2010)
178. Wang, Y., Alshboul, Y.: Mobile security testing approaches and challenges. In: 2015 First Conference on Mobile and Secure Services (MOBISECSERV), pp. 1–5. IEEE, Gainesville, FL, USA (2015)
179. Weerackody, V.: Satellite diversity to mitigate jamming in leo satellite mega-constellations. In: 2021 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6. IEEE, Montreal, QC, Canada (2021). https://doi.org/10.1109/ICCWorkshops50388.2021.9473519
180. Wu, B., et al.: A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks, pp. 103–135. Springer, Boston (2007). https://doi.org/10.1007/978-0-387-33112-6_5
181. Wu, J., et al.: An access control model for preventing virtual machine escape attack. Future Internet **9**(2), 20 (2017). https://doi.org/10.3390/fi9020020
182. Xi, K., et al.: A fingerprint based bio-cryptographic security protocol designed for client/server authentication in mobile computing environment. Secur. Commun. Netw. **4**(5), 487–499 (2011). https://doi.org/10.1002/sec.225
183. Xi, K., et al.: A fingerprint based bio-cryptographic security protocol designed for client/server authentication in mobile computing environment. Secur. Commun. Netw. **4**(5), 487–499 (2011)
184. Xiao, Y., et al.: One bit flips, one cloud flops: Cross-VM row hammer attacks and privilege escalation. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 19–35. USENIX Association, Austin, TX (2016). https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/xiao
185. Xu, Z., Zhu, S.: Semadroid: a privacy-aware sensor management framework for smartphones. In: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, CODASPY'15, p. 61–72. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2699026.2699114. https://doi.org/10.1145/2699026.2699114
186. Yan, M., et al.: Invisispec: Making speculative execution invisible in the cache hierarchy. In: 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 428–441. IEEE, Fukuoka, Japan (2018). https://doi.org/10.1109/MICRO.2018.00042
187. Yan, Z., et al.: Flexible data access control based on trust and reputation in cloud computing. IEEE Trans. Cloud Comput. **5**(3), 485–498 (2017)
188. Yang, H., Zhang, H., Zhang, J., Yang, L.: An anti-repeater-jamming approach based on blind source separation for the downlink of satellite communication systems. Int. J. Satellite Commun. Network. **37**(6), 527–535 (2019). https://doi.org/10.1002/sat.1294
189. Yang, X., et al.: A novel en-route filtering scheme against false data injection attacks in cyber-physical networked systems. IEEE Trans. Comput. **64**(1), 4–18 (2015). https://doi.org/10.1109/TC.2013.177
190. Yu, M., et al.: A secure routing protocol against byzantine attacks for manets in adversarial environments. IEEE Trans. Veh. Technol. **58**(1), 449–460 (2009). https://doi.org/10.1109/TVT.2008.923683
191. Zhang, N., et al.: Leave me alone: App-level protection against runtime information gathering on android. In: 2015 IEEE Symposium on Security and Privacy, pp. 915–930. IEEE, San Jose, CA, USA (2015). https://doi.org/10.1109/SP.2015.61
192. Zhang, Y., He, S., Chen, J.: Data gathering optimization by dynamic sensing and routing in rechargeable sensor networks. IEEE/ACM Trans. Network. **24**(3), 1632–1646 (2016). https://doi.org/10.1109/TNET.2015.2425146
193. Zou, Y., et al.: A survey on wireless security: technical challenges, recent advances, and future trends. Proc. IEEE **104**(9), 1727–1765 (2016). https://doi.org/10.1109/JPROC.2016.2558521