DOI: 10.2507/35th.daaam.proceedings.xxx

DATA MASKING AND TOKENISATION

Maristela Bosnjak, Jasmin Redzepagic* & Luka Zgrablic

This Publication has to be referred as: Bosnjak, M[aristela]; Redzepagic, J[asmin] & Zgrablic, L[uka] (2024). Data masking and tokenisation, Proceedings of the 35th DAAAM International Symposium, pp.xxxx-xxxx, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-xx-x, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/35th.daaam.proceedings.xxx

Abstract

This research paper examines mobile application security, focusing on data masking and tokenisation, which are essential for protecting sensitive data on Android and iOS platforms. It delves into platform-specific vulnerabilities and details techniques to safeguard data at rest and in transit. The study advocates for secure coding practices, advanced encryption, and regular security audits to enhance mobile app security. It emphasises the role of Mobile Device Management (MDM) solutions in strengthening security measures. To connect theory with practice, the paper includes real-world case studies that demonstrate the effectiveness of these security practices in various scenarios. These case studies provide actionable insights for developers, IT security practitioners, and professionals aiming to improve the development and management of secure mobile applications.

Keywords: mobile application security; data masking; secure coding practices; security audit.

1. Introduction

In the current era characterised by the widespread adoption of mobile applications, the overriding issue reverberating throughout the digital domain is the security of these pervasive technologies. As dependence on mobile apps grows, the urgency to address the increasing threats to data integrity and user privacy becomes paramount. This paper aims to dissect the complex structure of mobile application security, highlighting the critical importance of data masking and tokenisation. Amidst a constantly evolving technological environment, the vulnerabilities associated with Android and iOS platforms require detailed scrutiny.

This study investigates the substantial impacts of these vulnerabilities on user data and the possible consequences of security breakdowns, going beyond essential identification. This article examines many best practices, such as secure coding techniques, encryption tactics, and the critical function of regular security audits, considering that strong security measures are vital for safeguarding sensitive information.

Moreover, the discussion delves into Mobile Device Management (MDM) solutions, detailing their crucial contribution to bolstering mobile application security. Throughout this inquiry, real-world case studies and practical examples will shed light on the discussion, providing insights into practical implementations and emphasising the real-life impacts of embracing comprehensive security strategies. This exploration aims to identify and understand the security

challenges in mobile application development and to offer practical solutions, laying a solid groundwork for developers, practitioners, and security experts.

This research paper addresses the primary problem of mobile application security, explicitly protecting sensitive data on Android and iOS platforms. This includes identifying platform-specific vulnerabilities and finding effective techniques to safeguard data at rest and in transit.

2. Mobile application security challenges

Mobile applications have become integral to modern life, offering unprecedented convenience and introducing significant security challenges. This chapter comprehensively explores these challenges, detailing the general concerns and the more specific threats related to data breaches and privacy vulnerabilities [1].

As mobile applications become increasingly embedded in everyday activities, the ecosystem becomes more dynamic and susceptible to various security threats. These range from unauthorised access to sensitive data to exploiting existing software vulnerabilities. This section thoroughly examines the complex interaction between user behaviours, app design choices, and an ever-evolving threat landscape, highlighting the persistent security issues prevalent across the mobile application domain.

A significant area of concern within this landscape is the risk of data breaches, which significantly threaten user privacy. Mobile users often store personal and sensitive information within applications, which can become targets for unauthorised access and data compromise. In this research, we will dig deeper into the implications of such breaches, utilising real-world examples to emphasise the critical need for stringent security measures.

Recognising and understanding these security challenges is crucial for developing effective protective strategies. The subsequent sections of this text will delve into specific solutions to mitigate these risks. Discussions will focus on the critical roles of data masking, tokenisation, and mobile device management. By obscuring or anonymising sensitive information, data masking techniques help organisations comply with privacy regulations, mitigate the risk of data breaches, and protect customer confidentiality [1]. These strategies are essential for strengthening the security frameworks of mobile applications, addressing the identified challenges, and significantly enhancing the overall security posture of mobile environments. Furthermore, data anonymisation can strike an optimal balance between privacy protection and data utility, integrating techniques such as data masking [2]. This comprehensive approach ensures that mobile applications can continue to offer convenience without compromising the security and privacy of user data.

3. Platform-specific vulnerabilities

Let's now investigate platform-specific vulnerabilities that affect mobile operating systems, focusing on Android and iOS, which are predominant in the mobile industry. As mobile technology becomes increasingly widespread, it is imperative to understand the distinct security challenges each platform faces. We will delve into the unique vulnerabilities inherent to these systems and discuss how attackers can exploit them. By analysing the underlying architecture and security frameworks of Android and iOS, we aim to pinpoint their specific weaknesses to various attack vectors. Through comprehensive analysis, this chapter intends to provide developers, security professionals, and users with essential insights to better protect their devices from emerging threats. Gaining this knowledge is crucial for creating more secure applications and implementing practices that bolster the security of these widely used platforms.

3.1. Android vulnerabilities

While encouraging innovation, Android's open-source design poses specific concerns because of irregular security updates and fragmented software versions among various device makers. Comparing Android to iOS, which is an open-source operating system, iOS is typically less vulnerable to security flaws [3]. Common vulnerabilities in Android include:

- Permissions: Android applications are granted permission to access specific device resources. However, applications can request permissions they do not need or allow users to grant permissions to applications they do not trust accidentally.
- Insecure coding: Android applications are written in Java, a language known for its susceptibility to security vulnerabilities. Developers can make mistakes in their code that can allow attackers to exploit the application.
- Third-party libraries: Android applications often rely on third-party libraries to provide functionality. However, these libraries can also be vulnerable to security attacks.
- Rooted devices: Android devices can be rooted, meaning they have elevated privileges. This can make them more susceptible to attacks, as attackers can gain control of the device and install malicious software.

- Sideloading: Android devices allow users to install applications from sources other than the Google Play Store. This can introduce vulnerabilities, as users may not be able to verify the safety of these applications.

Some of the most significant attacks found in Android include [4]:

- Stagefright: Stagefright was a vulnerability in the Android multimedia framework that allowed attackers to execute code on devices remotely.
- Shellshock: Shellshock was a vulnerability in the Bash shell script interpreter that affected Android devices and other systems.
- Heartbleed: Heartbleed was a vulnerability in the OpenSSL cryptographic library that affected Android devices and other systems.
- Meltdown and Spectre: Meltdown and Spectre were vulnerabilities in Intel processors that affected Android devices and other systems.
- KRACK: KRACK was a vulnerability in the WPA2 wireless protocol that affected Android devices and other systems.
- QuadRooter: A set of four vulnerabilities affecting Qualcomm chipset software drivers that could give attackers root access to the device.
- Cloak and Dagger: A series of attacks exploiting Android's overlay and accessibility features, allowing attackers to hijack the user interface or steal information.
- Broadpwn: Broadcom's Wi-Fi chipset drivers could be vulnerable to remote code execution.
- StrandHogg: A vulnerability that allows malicious apps to masquerade as legitimate apps, enabling them to trick users into giving them sensitive permissions.
- Judy: A large-scale auto-clicking adware found in 41 apps that could potentially infect up to 36.5 million devices.

The vulnerabilities, such as Stagefright, BlueBorne, and QuadRooter, underscore significant security threats affecting millions of Android devices globally. These security flaws target everything from Bluetooth functionalities and multimedia processing to essential protocols like WPA2. Their discovery and exploitation highlight the ongoing struggle between system developers and cyber attackers.

In Q2 2023, the amount of malware, adware, or undesired software attacks on mobile devices increased. Throughout that time, 5,700,000 attacks were prevented. When a user attempted to pay for a transaction, the malware would steal their bank card credentials by inserting JavaScript code into the appearance of a well-known Asian online retailer. Among the other peculiar quarter finds was a movie-streaming app released on Google Play with a crypto miner [5].

Robust security mechanisms, regular system upgrades, and raising user knowledge are essential if Android continues leading the mobile operating system market. To safeguard their devices, users and developers must exercise caution, keep up with emerging security risks, and take preventative measures. A proactive approach is crucial to ensuring safe and dependable device management in our highly linked digital world.

3.2. iOS vulnerabilities

Because iOS is a closed-source operating system, security risks are lower than Android's. However, iOS is not immune to intrusions. Common vulnerabilities found in iOS include [6]:

- Application sandboxing: iOS applications are sandboxed, which means they are isolated from each other and the rest of the operating system. This can help to prevent attacks that target the operating system itself. However, attackers can still exploit vulnerabilities in individual applications.
- Insecure coding: iOS applications are written in Objective-C, a language known for susceptibility to security vulnerabilities. Developers can make mistakes in their code that can allow attackers to exploit the application.
- Rooted devices: iOS devices can be jailbroken and modified to gain root privileges. This can make them more susceptible to attacks, as attackers can gain control of the device and install malicious software.
- Sideloading: iOS devices do not allow users to install applications from sources other than the App Store, which can help prevent users from installing malicious applications. However, there are ways to sideload applications which can introduce vulnerabilities.

Some of the most critical iOS vulnerabilities include [7]:

- XcodeGhost: A malware strain propagated through compromised Xcode versions, Apple's official tool for developing iOS apps. Infected apps collect device information and upload it to servers controlled by attackers.
- Masque Attack - This attack exploited a flaw that allowed malicious apps to replace legitimate apps installed through the App Store. It was executed by luring users to install an app from a phishing link, replacing a genuine app without the user's knowledge.

- WireLurker: Discovered primarily targeting Chinese iOS users, this malware infiltrated iOS devices through malicious software on Macs and moved onto iOS devices via USB connections. It could steal information and install further malicious applications.
- Pegasus Spyware: A highly sophisticated spyware that exploits multiple zero-day vulnerabilities to hack iOS devices and spy on victims. It could read text messages, track calls, collect passwords, track location, access the target device's camera and microphone, and gather information from applications.
- KeyRaider: This malware targeted jailbroken iOS devices to steal account names, passwords, GUIDs, and Apple push notification service tokens. It led to unauthorised purchases and ransom demands.
- AceDeceiver: This malware exploited flaws in Apple's DRM protection mechanism (FairPlay) to install malicious apps on iOS devices, regardless of whether they were jailbroken.
- Checkm8: A boot rom exploit affected all iOS devices running on chips from A5 to A11. It allows permanent unmatchable jailbreaking of these devices, posing significant security risks.
- Trident: Used in combination with Pegasus spyware, the Trident exploit chain utilised three zero-day vulnerabilities to compromise iOS devices and install the spyware for surveillance purposes
- Ikee Worm: The first worm to target iOS, specifically jailbroken devices, which changed the device's wallpaper and attempted to propagate to other devices on the network. It primarily aimed to create botnets.
- Fake Jailbreak: Scams promising to jailbreak newer versions of iOS would instead install malware or adware on the devices of unsuspecting users, leading to privacy breaches and unauthorised use of the device resources.

The array of iOS security breaches, ranging from malware distributed through compromised versions of Xcode to advanced spyware like Pegasus, underscores the critical need for stringent security measures. These incidents highlight users' need to adhere to safe practices, such as restricting app downloads to trusted sources like the App Store, refraining from jailbreaking devices, and consistently updating software to protect against emerging vulnerabilities. Apple has issued patches to mitigate critical zero-day WebKit vulnerabilities for the latest iOS, tvOS, and macOS platforms. The patches will cover over 15 vulnerabilities that expose Apple devices to data leaks, arbitrary code execution, and denial-of-service attacks. The zero-day is the company's first for 2024 [8].

Moreover, these attacks underline the importance of ongoing user education in cybersecurity, as many exploits leverage user actions, like downloading a malicious app or clicking on a deceptive link. The continuous development of these threats points to a relentless challenge between cybercriminals and security professionals, driving forward the enhancement of security technologies and strategies to safeguard users in a digital age that is perpetually connected. That's also why we must develop new methodologies to mitigate these problems. For example, Magen supports the dynamic masking of payloads and the static masking of large data sets, offering format-preserving encryption and tokenisation [9].

4. Approaches to solving mobile application security problems

Data masking and tokenisation are critical methods for securing sensitive data when full disclosure is unnecessary. Data masking hides specific elements within a dataset to block unauthorised viewing of valid values, which is especially useful in testing, development, or third-party engagements. Masking can be static (permanent changes) or dynamic (real-time during data access). Tokenisation replaces sensitive data with non-sensitive tokens, which only a secure system can revert to the original data, making them useless to intruders without access. Both techniques reduce data breach risks and ensure compliance with strict privacy laws like GDPR and HIPAA, thus bolstering organisational data security frameworks and protecting sensitive data even if other security measures fail. We must remember that breaches happen daily, even in most regulated environments, such as the healthcare industry, which is the industry with the highest cost of data breaches. This further proves how important it is to develop secure and secure environments in which we will host these applications. Having in mind that most modern environments for application deployments include containers and CI/CD (Continuous Integration/Continuous Deployment) pipelines, it is essential to point out that the security of CI/CD toolsets is paramount as they directly affect the entire development and deployment pipeline, presenting a unique set of security challenges that must be rigorously managed to protect against both internal and external threats [10].

4.1. Data masking

Data masking is an essential security approach that shields confidential information by hiding certain parts of a dataset, allowing it to be used safely in settings where accurate data exposure could be dangerous, such as development, testing, or training. Creating a version that looks architecturally like the original but contains non-sensitive data makes it possible to use genuine datasets without disclosing sensitive information. There are various types of data masking, including dynamic data masking, which momentarily obscures data in real-time during query execution without changing the underlying database, and static data masking, which modifies data permanently before it is moved from a secure production environment. This approach is crucial for protecting against outside threats and complying with stringent data protection regulations, such as GDPR, which require careful data processing.

Data masking can protect sensitive information in mobile applications, such as credit card numbers, Social Security numbers, and customer addresses. In data masking, actual data is substituted with artificial or scrambled data, making it meaningless and less appealing to potential attackers. This reduces the value of the data and its usefulness if compromised.

There are several techniques for masking data. A standard method involves replacing sensitive information with random characters; for example, a credit card number might be shown as "XXXX-XXXX-XXXX-XXXX." Alternatively, encryption can scramble the data, making it unreadable to anyone without the encryption key.

4.2. Tokenization

Tokenisation is a technique that replaces confidential information with distinct identifiers, or tokens, to strengthen application security. Tokenisation is necessary for protecting sensitive data in apps, such as credit card numbers, social security numbers, and personal health information. These tokens maintain essential details of the data without jeopardising its security. Applications handle less sensitive information directly by substituting tokens for sensitive data, which lowers the risk of data breaches and helps assure compliance with laws like PCI DSS and GDPR.

A real-world example would be a customer entering their credit card information on an online shopping application. Tokenisation substitutes a token for the actual credit card number. This token is kept in databases and used in the program procedures. The actual data is mapped back to the original data and is kept safe in a token vault concurrently. This vault's access is restricted to highly secure procedures, significantly reducing the methods by which a cybercriminal can obtain private information. As a result, even if an attacker gained access to the application's database, the extracted tokens would be useless without the tokenisation mechanism, guaranteeing the security of the sensitive data underneath.

5. Best practices for mobile app security

Mobile app security is increasingly critical in our technology-driven world, where mobile devices are ubiquitous in personal and professional settings. The challenge in securing mobile applications arises from the vast array of operating systems, various device types, and diverse user interactions. These apps are particularly vulnerable to security threats, leading to data theft, phishing schemes, and unauthorised device control. As reliance on mobile applications continues to surge for functions like financial transactions, shopping, and personal communication, it's imperative to safeguard them against emerging cyber threats.

Several best practices must be diligently followed to bolster mobile app security. Robust authentication processes, such as multi-factor authentication, should be implemented to minimise the risk of unauthorised access. Developers must continuously update and patch applications to address newly discovered security flaws. There are three main ways to achieve a better mobile app security level: secure coding, encryption, and audits, where appropriate.

Secure coding techniques are essential to prevent vulnerabilities like SQL injections and cross-site scripting. There are multiple principles involved when trying to write secure code:

- Employ defensive coding practices: Developers should craft code that can withstand typical security threats, such as SQL injections, cross-site scripting, and buffer overflows.
- Implement robust encryption: Developers must utilise strong encryption protocols to safeguard sensitive information, including passwords and credit card details.
- Refrain from embedding sensitive data: Developers should avoid hardcoding sensitive information directly into their applications. Instead, sensitive data should be kept in secure storage options like databases or keychains.
- Utilize secure authentication methods: Developers are advised to implement secure authentication strategies, such as two-factor authentication, to protect sensitive data.

Encrypting data when transmitted and stored ensures that sensitive information remains protected. We can employ different techniques:

- Symmetric encryption: This method employs a single data encryption key. It's commonly used for file encryption and facilitating secure device communications.
- Asymmetric encryption: This method utilises a pair of keys—public and private—for data encryption and decryption. The public key is openly shareable, while the private key remains confidential. This encryption is often used in secure communications, such as online banking and email security.
- Hashing: A one-way encryption that transforms data into a distinct fingerprint. Hashes are primarily used to confirm data integrity, helping to ensure that files have not been altered during download.

Additionally, rigorous security auditing, testing, and vulnerability assessments should be conducted before app releases. These proactive measures help identify and rectify potential security weaknesses, enhancing the overall security posture of mobile applications and providing a safer environment for users. Again, we can do this in multiple different ways:

- Static security analysis: This process involves analysing an application's source code without executing it. It helps detect various vulnerabilities, such as buffer overflows, SQL injections, and cross-site scripting.
- Dynamic security analysis: While the application is active, this analysis can pinpoint issues that static security analysis might miss, including memory leaks and race conditions.
- Penetration testing: This method simulates attacks on an application to identify significant vulnerabilities that could be exploited.

If we look at mobile app security from the opposite direction, from the direction of end users, some of whom might be our employees, one of the main ways to protect them from malicious mobile applications is by using MDM (Mobile Device Management). These are some of the operational procedures that are possible if we use MDM:

- Enrolling, configuring, and deploying devices
- Managing users and groups
- Controlling device settings and security policies
- Remotely locking, erasing, or wiping devices
- Monitoring device activity and collecting data
- Protecting data at rest, in transit, and in use.

In terms of mobile device security, using MDM allows us to do the following:

- Device enrolment: MDM solutions can help to automate the enrolment process for devices, which helps to ensure that all devices are correctly configured and compliant with security policies
- Profile deployment: MDM solutions can be used to deploy profiles to devices, which can be used to configure the device settings, such as passcode requirements, app restrictions, and web browsing restrictions
- Remote management: MDM solutions can be used to remotely manage devices, which allows IT administrators to lock devices, wipe data, and install software updates
- Content filtering: MDM solutions can be used to filter content on devices, which helps to prevent users from accessing inappropriate or malicious content
- Data encryption: MDM solutions can be used to encrypt data on devices, which helps to protect data from unauthorised access
- Security auditing: MDM solutions can be used to audit device activity, which helps to identify and investigate security incidents.

From the aspect of corporate security policies, all these capabilities are a very welcome addition to corporate security. They've also become somewhat of a norm, especially in larger enterprises, where MDM has taken a significant foothold in the past decade. For example, we can count on the fact that there will always be employees who will lose their mobile devices somewhere, and we must have some remote management capability to deal with these situations.

6. Case-study: Data masking tool

Most companies use numerous databases and applications in their daily operations. These (production) databases are frequently copied to different environments for additional purposes, such as development, testing, acceptance, training, outsourcing, etc. Nevertheless, many of these systems include corporate vital and privacy-sensitive data and personally identifiable information (PII), which you are not permitted to utilise for such purposes.

Data masking, also known as anonymisation, is the process of modifying data so that it can still be used for testing and development purposes. Still, it makes it nearly difficult to identify an individual.

For a case study, the data masking tool DATPROF [11] was used on a student information database containing real names, usernames, student classes, and other faculty information. The tool generated synthetic data to replace existing data.

The results showed how well the user-friendly interface of the Data Masking DATAPROF tool managed complex data relationships. This user-friendliness made complex data linkages easier to integrate and manage and drastically decreased the time and effort users need to put data masking methods into place. The tool's user-friendly interface ensured that confidential data was safeguarded without sacrificing the accuracy or functioning of the underlying data structures,

even for users with differing degrees of technical proficiency. The tool's robust features also helped maintain data accuracy and consistency across various datasets, demonstrating its capacity to handle complex data masking problems in different application settings.

Its features include support for XML and CSV files, integrated synthetic data generators, HTML audit and GDPR reporting, REST API testing for data automation, and an intuitive online interface.

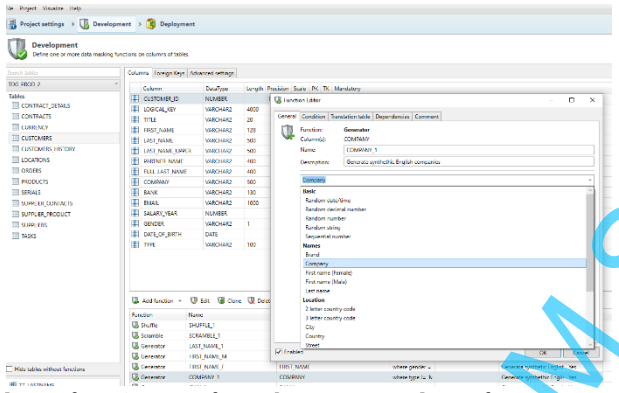

Figure 1. DATPROF tool interface (picture from: https://www.datprof.com/. Accessed on 07.07.2024.)

Figure 1 shows the general layout of the DATPROF tool interface. Because of GPDR restrictions, the database used for the case study cannot be displayed.

7. Future research

Multiple promising areas for future research should be investigated. Investigating memory safety in mobile app development entails refining current programming languages or crafting new ones that automatically address prevalent issues such as buffer overflows and memory leaks. Compiler technology advancements might concentrate on sophisticated static analysis tools explicitly designed to identify memory-related vulnerabilities. This strategy allows developers to produce inherently safer code, thereby minimising the risks of exploitable vulnerabilities.

In automated security testing, the emphasis is on creating cutting-edge tools that meld smoothly with mobile development workflows. Enhancing dynamic analysis tools to replicate real-world attack scenarios more accurately and employing machine learning to foresee and adjust to emerging threats are critical aspects of this focus. Furthermore, masked token models introduce a sophisticated method of maintaining data privacy during the machine learning training phase [12]. Such improvements can drastically lower the time and resources required to pinpoint vulnerabilities, thus bolstering the security of apps before they are launched. New approaches are also being developed for masking private data using Named Entity Recognition and Deep Learning concepts to achieve data anonymity [13]. Furthermore, the NextGen Data Masking Engine introduces a versatile approach to handling complex data structures for adequate data security and compliance [14]. This topic is being actively researched as it can potentially change how we implement different security layers.

Moreover, customising threat modelling for the mobile sector involves adapting existing methods to the unique challenges the mobile environment poses, including aspects related to mobile operating systems, hardware, and communication networks [15][16]. By proactively identifying and addressing potential threats during the development phase, developers can craft safer mobile applications, significantly lowering the risks for users and developers in a world increasingly dominated by mobile technology. In the current era characterised by the widespread adoption of mobile applications, the overriding issue reverberating throughout the digital domain is the security of these pervasive technologies. As dependence on mobile apps grows, the urgency to address the increasing threats to data integrity and user privacy becomes paramount [17].

8. Conclusion

The paper's in-depth analysis of mobile application security—which mainly focuses on data masking and tokenisation—has shed light on the critical steps required to safeguard sensitive data in the context of the widespread use of mobile apps in today's world. The best methods for enhancing mobile app security were highlighted in this study, which also carefully investigated the security flaws and problems of the iOS and Android platforms. Data masking and tokenisation have become essential tactics to make sensitive data unreadable and provide a substantial layer of protection against unwanted access.

Our investigation also covered the difficulties that Android and iOS present, highlighting the significance of implementing platform-specific security controls. We also found that novel tactics, including data masking, are required to guarantee complete data protection in the cloud because traditional data security measures are insufficient. Data masking is an essential security measure that protects sensitive data by replacing it with realistic but not actual data, allowing organisations to use and share data without exposing personal or sensitive information.

The study examined a wide range of recommended procedures for protecting mobile applications, including doing in-depth security audits, utilising encryption, and using secure coding techniques. It highlighted how Mobile Device Management (MDM) solutions may improve security, demonstrating their importance in a comprehensive security plan. The case study showed how simple it is to perform data masking with a specialist tool for data masking like DATPROF.

In conclusion, the study not only determined the critical security controls for mobile applications but also offered developers and IT security professionals useful information and workable answers. The results highlight the importance of incorporating thorough security measures and employing cutting-edge security techniques to safeguard sensitive data in mobile applications.

9. References

[1] Badgujar, P. (2021). Implementing Data Masking Techniques for Privacy Protection. Journal of Technological Innovation, Vol.2 Issue 4, October-December 2021
[2] ŞAHİN, Y.; DOGRU, İ. (2023). An Enterprise Data Privacy Governance Model: Security-Centric Multi-Model Data Anonymization, Proceedings of Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi, April 15, 2023, Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi, DOI 10.29137/umagd.1272085
[3] Garg, S., & Baliyan, N. (2021). Comparative analysis of Android and iOS from security viewpoint. In Computer Science Review (Vol. 40, p. 100372). Elsevier BV. DOI 10.1016/j.cosrev.2021.100372
[4] Android Mobile Security Threats, Available from: https://www.kaspersky.com/resource-center/threats/android-mobile-threats Access: 2024-04-08
[5] IT threat evolution in Q2 2023. Mobile statistics, Available from: https://securelist.com/it-threat-evolution-q2-2023-mobile-statistics/110427/ Access: 2024-04-08
[6] What Are the iOS Security Vulnerabilities? Available from: https://www.preemptive.com/blog/what-are-the-ios-security-vulnerabilities/ Access: 2024-04-08
[7] Iphone Os : Security Vulnerabilities, CVEs. Available from: https://www.cvedetails.com/vulnerability-list/vendor_id-49/product_id-15556/Apple-Iphone-Os.html Access: 2024-04-08
[8] Apple Patches Critical Zero-Day Vulnerability for Macs and iPhones. Available from: https://www.spiceworks.com/it-security/vulnerability-management/news/apple-patches-critical-zero-day-vulnerability-macs-iphones/ Access: 2024-04-12
[9] Moffie, M.; Mor, D.; Asaf, S.; Farkash, A. (2022). Next Generation Data Masking Engine, Proceedings of Lecture Notes in Computer Science, 2022, pp. 152–160, Springer International Publishing, DOI 10.1007/978-3-030-93944-1_10
[10] Dakic, V.; Redzepagic, J.; Basic, M. (2022). CI/CD Toolset Security, Proceedings of DAAAM Proceedings, DAAAM International Vienna, 2022, pp. 0161–0164, DAAAM International Vienna, DOI 10.2507/33rd.daaam.proceedings.022
[11] DATAPROF tool for data masking. Available from: https://www.datprof.com/products/datprof-privacy/ Access: 2024-02-10
[12] Hou, L.; Geng, Y.; Han, L.; Yang, H.; Zheng, K.; Wang, X. (2022). Masked Token Enabled Pre-training: A Task-Agnostic Approach for Understanding Complex Traffic Flow, Proceedings of the Institute of Electrical and Electronics Engineers, February 10, 2022, Institute of Electrical and Electronics Engineers (IEEE), DOI 10.36227/techrxiv.19134854.v1
[13] Kodandaram, Satwik R.; Honnappa, K. & Soni K. (2021). Masking private user information using Natural Language Processing, International Journal of Advance Research, Ideas and Innovations in Technology, Volume 7, Issue 3 - V7I3-2048, ISSN: 2454-132X
[14] R. A.; Hegadi, R. S.; M. T. N. (2018). A Study on Big Data Privacy Protection Models using Data Masking Methods, Proceedings of the International Journal of Electrical and Computer Engineering (IJECE), Institute of Advanced Engineering and Science, October 01, 2018, p. 3976, DOI 10.11591/ijece.v8i5.pp3976-3983
[15] Aeri, M. (2020). Secure Data Sharing in Cloud Computing Systems: Techniques and Applications, Proceedings of the Turkish Journal of Computer and Mathematics Education (TURCOMAT), Ninety Nine Publication, December 15, 2020, pp. 2087–2094, DOI 10.17762/turcomat.v11i3.13606
[16] Stefanovic, D.; Nikolic, D.; Dakic, D.; Spasojevic, I.; Ristic, S. (2020). Static Code Analysis Tools: A Systematic Literature Review, Proceedings of DAAAM Proceedings, DAAAM International Vienna, 2020, pp. 0565–0573, DOI 10.2507/31st.daaam.proceedings.078
[17] Dakic, V.; Jakobovic, K.; Zgrablic, L. (2022). Linux Security in Physical, Virtual, and Cloud Environments, Proceedings of DAAAM Proceedings, DAAAM International Vienna, 2022, pp. 0151–0160, DOI 10.2507/33rd.daaam.proceedings.021