

**ВИСШЕ ВОЕННОМОРСКО УЧИЛИЩЕ
„НИКОЛА ЙОНКОВ ВАПЦАРОВ“**

Стоян Мечев



**МЕТОДИ ЗА ПОВИШАВАНЕ
НА СИГУРНОСТТА
НА ОПЕРАЦИОННИ СИСТЕМИ
ЗА МОБИЛНИ УСТРОЙСТВА**

Варна, 2025



**ВИСШЕ ВОЕННОМОРСКО УЧИЛИЩЕ
„НИКОЛА ЙОНКОВ ВАПЦАРОВ“**

Стоян Т. Мечев

**МЕТОДИ ЗА ПОВИШАВАНЕ
НА СИГУРНОСТТА НА ОПЕРАЦИОННИ
СИСТЕМИ ЗА МОБИЛНИ УСТРОЙСТВА**

Варна, 2025 г.

Стоян Т. Мечев

**МЕТОДИ ЗА ПОВИШАВАНЕ
НА СИГУРНОСТТА НА ОПЕРАЦИОННИ
СИСТЕМИ ЗА МОБИЛНИ УСТРОЙСТВА**

Варна, 2025 г.

**МЕТОДИ ЗА ПОВИШАВАНЕ НА СИГУРНОСТТА НА
ОПЕРАЦИОННИ СИСТЕМИ ЗА МОБИЛНИ УСТРОЙСТВА**
Книга по дисертация

ВСИЧКИ ПРАВА ЗАПАЗЕНИ

© Стоян Тодоров Мечев, автор, 2025

© Висше военноморско училище „Никола Й. Вапцаров“ – Варна

Научни рецензенти:

Проф. д-р Юлиян Иванов Цонев – Висше военноморско училище
„Никола Й. Вапцаров“

Проф. д-р Тодор Димитров Ганчев – Технически университет – Варна

Редактор и коректор: д-р Ваня Колева Колева

**Methods for improving the security
of operating systems for mobile devices**

Book based on Dissertation

© Stoyan Mechev, 2025

© Nikola Vaptsarov Naval Academy

Varna, Bulgaria

Scientific reviewers:

Julian Ivanov Tsonev, Full Professor, PhD

Todor Dimitrov Ganchev, Full Professor, PhD

Editor: Vanya Koleva Koleva, PhD

ISBN 978-619-7752-13-7 (e-book)

DOI <https://doi.org/10.63662/diss-books/mechev-2025>

Методи за повишаване на сигурността на операционни системи за мобилни устройства

Стоян Мечев

Web of Science Researcher ID: AAB-4270-2021

ORCID ID: 0000-0002-0809-8538

Висше военноморско училище „Н. Й. Вапцаров“ – Варна

e-mail: st.mechev@naval-acad.bg

Анотация

Целите, които се поставят в книгата, са изследване на методите за повишаване на сигурността на операционни системи за мобилни устройства чрез критичен обзор на научните публикации, свързани с проблемите на уязвимостите в операционните системи за мобилни устройства, анализ на публично достъпни бази от данни с уязвимости и техните системи за оценка на тежестта на щетите, които може да нанесе зловреден софтуер, предназначен за мобилни устройства, работещи под операционната система Android или iOS, откриване на слабости в сигурността на уеббраузъри за мобилни устройства.

За постигане на тези цели са използвани методите: сравнителен анализ, метод на балните скали и обобщение. Въз основа на резултатите от прилагането на сравнителния анализ се установява кои от националните бази данни имат по-добри възможности за филтриране на данните и получаване на адекватни резултати при търсене на уязвимости за мобилни устройства.

Ключови думи: Android; iOS; сигурност, уязвимост, база данни с уязвимости, браузър, рутинг, джейлбрейкинг.

Methods for improving the security of operating systems for mobile devices

Stoyan Mechev, Ph.D.

Web of Science Researcher ID: AAB-4270-2021

ORCID ID: 0000-0002-0809-8538

Nikola Vaptsarov Naval Academy

73, Vasil Drumev Str.

9002 Varna, Bulgaria

e-mail: st.mechev@naval-acad.bg

Abstract

The objectives of this book are to investigate methods for improving the security of mobile devices operating systems through a critical review of scientific publications related to vulnerabilities in mobile device operating systems, to analyze publicly available vulnerability databases and their systems to assess the severity of damage that can be caused by malware designed for mobile devices running the Android or iOS operating system, to detect security weaknesses in web browsers for mobile devices, and to identify vulnerabilities in mobile device operating systems.

The methods used to achieve this objective are: comparative analysis, scoring method and generalization. Based on the results of the application of the comparative analysis, it is determined which of the national databases have better capabilities to filter the data and obtain adequate results when searching for vulnerabilities for mobile devices.

Keywords: Android; iOS; security, vulnerability, vulnerability database, browser

Предговор

През второто десетилетие на XXI век употребата на мобилни устройства стана широка практика, което ги превърна в неотделима част от ежедневието на хората по целия свят. Например, по данни на Statista, за 2022 г. има над 4,7 милиарда потребители на смартфони (STATISTA 2023).

От една страна, използването на такова количество мобилни устройства предоставя огромен брой възможности за иновации и комуникация, но от друга страна, създава и множество предизвикателства в областта на сигурността.

Сигурността на операционните системи за мобилни устройства става фундаментален аспект, който изисква нашето внимание и изследване. Заплахите в онлайн средата, свързани с мобилните устройства, станали по-сложни и разпространени. От кражба на лични данни до злонамерени софтуерни атаки, нито една мобилна платформа не е в безопасност.

Предмет на изследването

Предмет на изследването са операционните системи за мобилни устройства (ОСМУ) Android и iOS.¹

Обект на изследването

Обект на изследването е сигурността на ОСМУ.

Цел на изследването

Да се анализират видовете уязвимости за ОСМУ и да се установи в каква степен откритите в изследванията уязвимости са отстранени в практиката.

Хипотеза

ОСМУ гарантират достатъчно ниво на сигурност при тяхната експлоатация.

¹ Навсякъде в изложението се подразбира iOS, която се използва в устройствата на Apple, освен ако изрично не е указано друго. Да се различава от Cisco IOS (Cisco Internet-work Operating System), която се използва за мрежовите устройства на Cisco.

Задачи на изследването

1. Обзор на научните публикации, свързани с проблемите на уязвимостите в операционните системи за мобилни устройства.
2. Анализ на публично достъпни национални бази данни, съдържащи описания на уязвимости за операционни системи за мобилни устройства.
3. Изследване на системи за определяне на степента на вредоносност на уязвимост.
4. Откриване на слабости в сигурността на уеббраузъри за мобилни устройства.

Методи на изследването

В изследването са приложени методите **анализ и обобщение**.

С оглед на спецификата на решаване на втората научноизследователска задача се прилага **методът на балните скали**, а при решаването на четвърта научноизследователска задача – **технически експеримент**.

Прогнозирани научни резултати

След направения литературен обзор, който се обособява в първа глава, се очаква:

- Да бъде открита област, която е слабо изследвана. Предполагаме, това е оценяването на възможностите за търсене на информация за уязвимости за ОСМУ в публично достъпни бази данни с уязвимости. По този начин да се обосноват втора и трета изследователска задача.
- Да се обобщят методите за откриване на нови неизследвани уязвимости.
- Да се установи наличието на уязвимости, които са известни, но не са отстранени, с което да се обоснове четвърта изследователска задача.

При решаването на втора и трета научноизследователска задача, текстът на които е обособен във втора глава, се очакват следните резултати:

- Систематизиране на национални публично достъпни БДУ и техните възможности за откриване на информация за уязвимости за ОСМУ.

- Прилагане на метода на балните скали за оценяване на възможностите за откриване на информация за уязвимости за ОСМУ.
- Разкриване на слаби страни при търсене на уязвимости за ОСМУ в БДУ.
- Даване на препоръки за отстраняване на посочените слаби страни.

При решаването на четвърта научноизследователска задача, текстът на която е обособен в трета глава, се очакват следните резултати:

- Разработване на софтуер, който да диагностицира наличието на уязвимости в браузърите за ОСМУ. Софтуерът да генерира протокол за направените проверки и ако са установени уязвимости, да извежда предупреждаващо съобщение за потребителя.
- Да се направят препоръки за справяне с откритите уязвимости в браузърите, които ще бъдат част от методите за повишаване на сигурността на операционни системи за мобилни устройства.

Разработеният софтуер може да послужи като средство за повишаване на сигурността, както следва:

- Пряко – да предупреди потребителя за уязвимости и да препоръча смяна на браузъра, ако е необходимо.
- Непряко – да насочи вниманието на потребителите към разрешенията, които се дават на приложенията, и по този начин да повиши осъзнатостта при даване на разрешения.

Ограничения на изследването

Потребители на информацията за уязвимостите

Специалисти по киберсигурност, разработчици на мобилни приложения и операционни системи, администратори на мрежи и сигурност, ръководители на организации, обикновени потребители на мобилни устройства.

В книгата не се изследват потребителите на уязвимостите – кой и с каква цел би могъл да се възползва от дадена уязвимост.

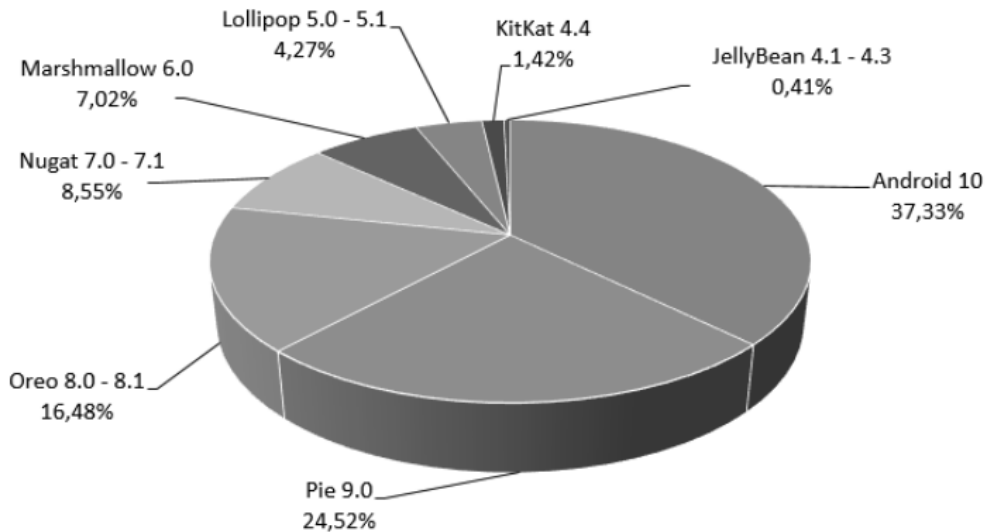
Изследвани мобилни операционни системи

Изследването на всички ОСМУ е нецелесъобразно, тъй като, по данни на (STATCOUNTER 2022), в края на 2021 г. пазарните дялове ОС Android и iOS са били съответно 70,01% и 29,24%, или общо 99,25% от

мобилните устройства на пазара са използвали една от двете операционни системи. Пазарният дял на всички останали мобилни операционни системи е под 1% и те би следвало да се изключат от изследването, което в случая е направено.

Версии на Android ОС

Като се отчете хронологията на внедряване на различните версии на ОС Android (Raphael 2023) и пазарните им дялове в края на 2021 г. (фиг. 1), може да се направи заключението, че изследването на уязвимости за версии, по-ранни от Android 5.0 Lollipop, всъщност не са продуктивни за конкретизираната цел на изследването, тъй като общият пазарен дял на всички по-ранни версии е под 2%.



Фигура 1. Пазарни дялове на ОС Android в края на 2021 г.

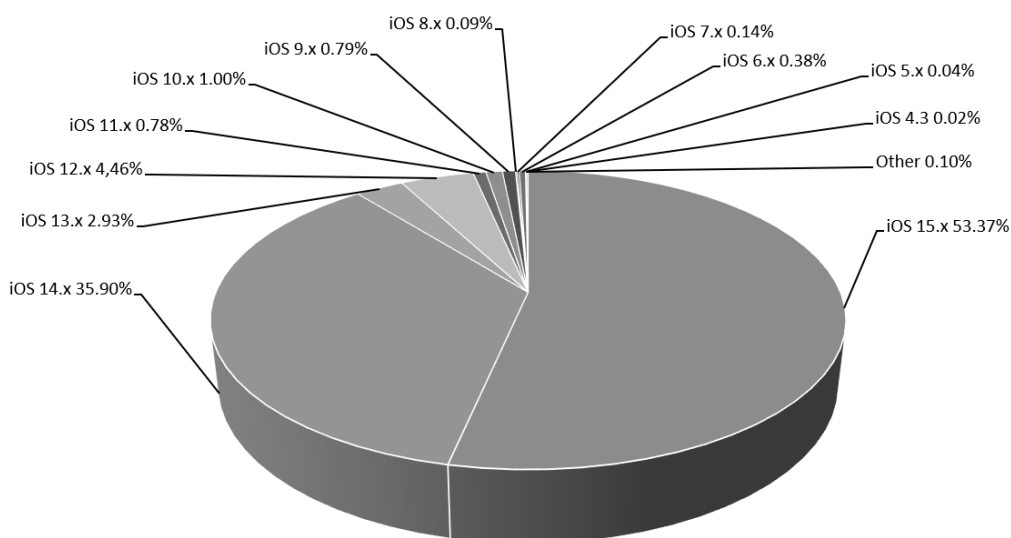
Забележка: *Фигурата е съставена по публикацията (Curry 2023).*

Версии на iOS

След прилагане на същите ограничения, а именно хронология на внедряване на iOS (Moreau 2021) и пазарен дял (фиг. 2), може да се направи заключението, че не е целесъобразно да се изследват версии на iOS, по-ранни от 10.0.

Изследвани устройства

Операционната система iOS се използва само на устройства на фирмата Apple, т.е. само мобилни телефони, планшети, часовници и телевизори. За сравнение, Android се използва в мобилни телефони, планшети, часовници, телевизори от различни производители, хладилници, приставки за телевизор (TV Box), машини за караоке, фотоапарати, автомобилна навигация, игрови конзоли. Изследването е ограничено до смартфони, защото при тях има най-голям брой сензори, и второ, това е общо и за двата вида операционни системи.



Фигура 2. Пазарни дялове на iOS в края на 2021 г.

Забележка: Фигурата е съставена по публикацията (Moreau 2021).

Темата на книгата кореспондира със следните стандарти:

1. БДС ISO/IEC 27005 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). ISO/IEC 27005:2022. *Сигурност на информацията, киберсигурност и защита на неприкосновеността. Указания за управление на рисковете за сигурността на информацията.*

2. БДС EN ISO/IEC 27002 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). ISO/IEC 27002:2022. *Сигурност на информацията, киберсигурност и защита на неприкосновеността. Механизми за контрол на сигурността на информацията.*

При оформлението на книгата е използван БДС ISO 7144:2011. ДОКУМЕНТАЦИЯ. Оформяне на дисертации и подобни документи, а за библиографията е приложен БДС ISO 690:2021 (E) INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). ISO 690: 2021(E), *Information and documentation – Guidelines for bibliographic references and citations to information resources*.

В литературния обзор са включени предимно статии, намиращи се в първи и втори квартал на Scopus или Web of Science, публикувани през последните шест години, за да се повишат надеждността и актуалността на резултатите, получени в изследването.

Съдържание

Анотация.....	5
Предговор	7
Предмет на изследването	7
Обект на изследването	7
Цел на изследването	7
Хипотеза	7
Задачи на изследването.....	8
Методи на изследването	8
Прогнозирани научни резултати.....	8
Ограничения на изследването.....	9
Потребители на информацията за уязвимостите	9
Изследвани мобилни операционни системи	9
Изследвани устройства.....	11
Съдържание	13
Списък на фигурите.....	16
Списък на таблиците	17
Списък на използваните съкращения.....	18
Тематичен тълковен речник	20
Глава първа. Оценка на съвременното състояние на проблема по сигурността на операционни системи за мобилни устройства	25
1.1. Методика на литературния обзор	25
1.2. Оценка на съвременното състояние на проблема по сигурността на операционни системи за мобилни устройства от групата на iOS	25
Изводи и обобщения по сигурността на iOS.....	31

1.3. Оценка на съвременното състояние на проблема по сигурността на операционни системи за мобилни устройства от групата на Android.....	32
Изводи и обобщения по сигурността на ОС Andorid	46
Изводи и обобщения по глава първа	47
Глава втора. Анализ на публично достъпни национални бази данни, съдържащи описания на уязвимости за операционни системи за мобилни устройства.....	49
2.1. Метод на балните скали и неговото приложение при БДУ.....	50
2.1.1. Теоретична постановка на метода на балните скали.....	50
2.1.2. Приложение на метода на балните скали в CVE	51
2.1.3. Процедура по класификация на уязвимостите.....	53
2.2. Изследвани бази данни с уязвимости	53
2.2.1. Критерии за оценка.....	54
2.2.2. Бази данни с уязвимости (БДУ).....	55
2.2.3. Обсъждане и резултати.....	59
Изводи:	63
Глава трета. Подход за откриване на пропуски в сигурността на уеббраузъри за мобилни устройства	66
3.1. Вградени защити за ОСМУ.....	66
3.1.1. Вградени защити на iOS.....	66
3.1.2. Вградени защити на Android.....	67
Изводи и обобщения по раздел 3.1.....	71
3.2. Експериментална част	71
3.2.1. Постановка на експеримента	73
3.2.2. Резултати от провеждането на експеримента.....	79
3.2.3. Анализ на получените резултати	82

Изводи и обобщения по раздел 3.2.....	83
Изводи и заключения по глава трета.....	84
Изводи и заключение	85
Научно-приложни приноси	87
Приложни приноси	87
Използвана литература	88
Приложения	107
Приложение 1. Опитни данни по експеримент към глава трета	107
Приложение 2. Изходен код към глава трета	112
Приложение 3. Пример за протокол от провежданите тестове, в които са открити уязвимости в браузъра	144
Приложение 4. Пример за протокол от провежданите тестове, в които не са открити уязвимости в браузъра	145

Списък на фигурите

Фигура 1. Пазарни дялове на ОС Android в края на 2021 г.....	10
Фигура 2. Пазарни дялове на iOS в края на 2021 г.	11
Фигура 3. Блок-схема на експеримента	60
Фигура 4. Пример за диалогов прозорец за даване на разрешение за достъп. ОС: iOS 17; устройство: iPhone 15; браузър: Safari; източник: авторът.	67
Фигура 5. Пример за диалогов прозорец за даване на разрешение на достъп. ОС: Android 13; устройство: Samsung A32 5G; браузър: Samsung Internet; източник: авторът.	70
Фигура 6. Изглед от началната страница на уебсайта, от който се провежда експериментът	74
Фигура 7. Блок-схема на експеримента	77
Фигура 8. Блок-схема на алгоритъм за опит за получаване на данни от конкретен сензор	78

Списък на таблиците

Таблица 1. БДУ, подредени по NCPI.....	54
Таблица 2. Скала за оценка на тежестта на уязвимост CVSS 3.1	56
Таблица 3. Таблица за оценка на тежестта на уязвимост	57
Таблица 4. Възможности за филтриране на информацията в БДУ	61
Таблица 5. Възможни резултати при опит за получаване на достъп до сензор	80
Таблица 6. Примери за фалшиво-положителни резултати	80
Таблица 7. Обобщени данни по изследваните сензори.....	81
Таблица 8. Обобщени резултати по браузъри за Android.....	82
Таблица 9. Обобщени резултати по браузъри за iOS.....	83

Списък на използваните съкращения

БДУ	– База данни с уязвимости
НЛКВ	– Национална лаборатория по компютърна вирусология
ОСМУ	– Операционна система за мобилни устройства
ПВП	– Потенциално вредно приложение
AUS CERT	– CERT на Австралия. Вж. CERT
AVD	– Android Virtual Device – виртуално Android устройство
BYOD	– Bring Your Own Device – вж. тематичен тълковен речник
CAPEC	– Common Attack Pattern Enumeration and Classification – общ списък и класификация на шаблоните за атака
CPE	– Common Platform Enumeration – общ списък на платформите. Вж. тематичен тълковен речник
CERT	– Computer Emergency Response Team – екип за бързо реагиране при компютърни инциденти
CISA	– Cybersecurity & Infrastructure Security Agency – Агенция по киберсигурност и сигурност на инфраструктурата на САЩ
CSAF	– Common Security Advisory Framework – обща консултативна работна рамка за сигурност
CWE	– Common Weakness Enumeration – общ списък на слабостите
CVSS	– Common Vulnerability Scoring System – система за оценка на критичността на уязвимостите
ICC	– Inter-Component Communication – комуникация между компонентите
iOS	– iPhone operating system – операционна система за мобилни устройства на компанията Apple. Разработена първоначално за iPhone, тя се използва и в мобилните устройства iPod Touch, iPad и Apple TV

IOS	– вж. Cisco IOS в тематичен тълковен речник
IPA	– Intelligent Personal Assistant – интелигентен личен асистент. Вж. тематичен тълковен речник
JSON	– JavaScript Object Notation – формат за съхраняване и пренасяне на данни
NBS	– Needs Based Security – сигурност, базирана на нуждите
NCPI	– National Cyber Power Index – национален индекс за киберсила
NVD	– National Vulnerability Database – националната база данни за уязвимости на САЩ
OS/OC	– Operating System – операционна система
US CISA	– вж. CISA
PHA	– Potentially Harmful Applications – потенциално вредни приложения (ПВП)
RD&P	– Runtime Detection and Prevention – откриване и защита по време на изпълнението на приложението
SVM	– Support Vector Machine – вж. тематичен тълковен речник

Тематичен тълковен речник

Експлойт – код, който се възползва от уязвимост на софтуера или пропуск в сигурността.

Експлоатиране, експлойтване – използване на експлойти.

Клониране на приложение – вж. препакетиране на приложение.

Обфускиране на кода – маскиран програмен код, който умишлено е направен нечетим от програмиста, който го е създал, с цел да не може лесно да бъде променян.

Препакетиране на приложение (repackaging) – вид атака, известна също и като „клониране на приложение“. При тази атака изпълнимият байткод на приложението се декомпилира до изходен код. След това атакуващият може да направи малки промени в изходния код (например пренасочване на приходите от реклама към друг потребител) или да добави зловреден код. След това приложението се компилира и се публикува отново в Google play от името на друг автор, което по същество представлява плагиатство. При повторното компилиране може да се използват техники за обфускиране на кода, така че плагиатството да не може да бъде установено чрез статичен анализ на кода.

Специални данни – терминът има по-широк обхват от приетия с GDPR. За специални данни по отношение на сигурността на смартфон се приемат:

- Лична информация на потребителя: телефонен номер, кратки съобщения (SMS и MMS), история на повикванията, история на посетените страници в интернет, кореспонденция по електронна поща, контакти, модел на телефона;

- Информация за идентификация на оборудването: сериен номер на устройството и идентификационен номер на оборудването (IMEI) и други подобни;

- Информация за текущото местоположение на устройството;

- Информация за приложен софтуер, удостоверяване на потребителя: отнася се главно за софтуер, използван за удостоверяване на потребителското име и парола;

- Информация за конфигурацията: лични предпочитания, настройки на софтуера (Yang et al. 2016).

Форензика (в информационните технологии) – изследване на цифрови следи и електронни улики с цел откриване, анализиране и предотвратяване на компютърни престъпления или инциденти, свързани с информационната сигурност.

Хибридно мобилно приложение – приложение, разработено с помощта на HTML, JavaScript и CSS, работещо на мобилен телефон.

.apk файл – пакет с файлове от приложение за Android.

Active Learning машинно обучение – активното учене е подмножество на машинното обучение, при което алгоритъмът за учене може интерактивно да поиска от потребителя да маркира данните с желаните резултати.

AndroidManifest.xml – манифест файлът описва съществена информация относно приложението за инструментите за компилиране на Android приложения, операционната система Android и Google Play. Наред с много други неща, файлът манифест трябва да декларира следното:

– Компонентите на приложението, включително всички дейности, услуги, приемници за излъчване и доставчици на съдържание. Той може да декларира възможности, като например с кои конфигурации на устройствата може да работи, и филтри за намерения, които описват как може да се стартира компонентът.

– Разрешенията, от които се нуждае приложението, за да получи достъп до защитени части на системата или до други приложения. Също така се декларира всички разрешения, които други приложения трябва да имат, ако искат да получат достъп до съдържание от това приложение.

– Хардуерните и софтуерните характеристики, които изисква приложението, което влияе на това, кои устройства могат да инсталират приложението от Google Play.

AT commands – инструкции, използвани за управление на модем. Наборът от команди се състои от поредица от кратки текстови низове, които могат да се комбинират, за да се създадат команди за операции, като набиране, управление на SMS функции и промяна на параметрите на връзката.

Baseband модем – базовият модем, вграден в смартфон.

Bricking – брикване (при Android устройства) – устройството става неизползваемо след неуспешен опит за рутване. Вж. rooting.

BYOD – Bring your own device – начин на работа, при който на служителите се позволява да използват свои собствени мобилни устройства, с които да осъществяват достъп до корпоративните данни и системи.

Cisco IOS – Cisco Internetwork Operating System – мрежова операционна система за устройствата на Cisco.

Code smells – аромати в изходния код – появяват се, когато кодът не е написан по основните стандарти, и дават възможност за кибератаки.

CPE – Common Platform Enumeration – структурирана схема за именуване на системи, софтуер и пакети за информационни технологии. Въз основа на общия синтаксис за унифицирани идентификатори на ресурси (URI), CPE включва формален формат на име, метод за проверка на имена спрямо система и формат на описание за обвързване на текст и тестове към име на платформа.

Decision Tree – дърво на решенията – непараметричен алгоритъм за обучение под наблюдение, който се използва както за задачи за класификация, така и за регресия. Има йерархична дървовидна структура, която се състои от коренов възел, разклонения, вътрешни възли и листни възли.

Deep learning – дълбоко обучение – метод за машинно обучение, използващ невронни мрежи.

DIFT – Dynamic Information Flow Tracking – динамичното проследяване на информационния поток – техника, която използва маркери за метаданни, за да проследява информационния поток между различни субекти.

False-positive result – фалшиво-положителни (фп) резултати при експеримент.

Fingerprint – пръстов отпечатък.

ICC – Inter-Component Communication – комуникация между компонентите.

Intent – намерение – в разработката на приложения за Android предоставя възможност за извършване на късно свързване по време на изпълнение между кода в различни приложения. Най-важната му употреба е при стартирането на дейности, където той може да се разглежда като свързващо звено между дейностите. По същество това е пасивна структура от данни, съхраняваща абстрактно описание на действието, което

трябва да бъде извършено.

IRA – интелигентен личен асистент – приложение, включващо форми на изкуствен интелект и машинно обучение. Например Siri, Google Assistant, Vixby.

Jailbreak – джейлбрейк, вж. Jailbreaking.

Jailbreaking – джейлбрейкване – процес на получаване на достъп до най-високите привилегии на iOS (по същество root права), които по подразбиране не са достъпни за смартфоните на Apple. Използването на Jailbreak отменя гаранцията на устройството.

JFC – Juice filming charging – вид side channel атака, осъществявана по време на зареждане на телефона с кабел. Тази атака може да улови или изведе лична информация на потребителите чрез автоматично записване на екрана на мобилните устройства по време на целия процес на зареждане.

Malware-Centric и Malware Creator-Centric методи за анализ – методологията, ориентирана към зловреден софтуер, се отнася до подход в киберсигурността, който се фокусира върху изучаването, анализа и предотвратяването на зловреден софтуер. Методологията, ориентирана към зловредния софтуер, включва разбиране на характеристиките, поведението и методите на разпространение на зловредния софтуер.

Man-In-The-Middle (MITM) – атака „човек по средата“ – общ термин за случаите, в които извършителят се позиционира в комуникацията между потребител и приложение – или за да подслушва, или за да се представя за една от страните, като създава впечатление, че се извършва нормален обмен на информация.

Marketplace (application marketplace) – пазарно място – официален източник на приложения, разработвани за съответната операционна система, или официален източник на приложения от производител на хардуер.

Privilege Escalation – повишаване на привилегиите – атака за получаване на незаконен достъп до повишени права, или привилегии, надхвърлящи определените за дадена идентичност, акаунт, потребител или машина.

Ransomware – рансъмвер – вид зловреден софтуер, който шифрова информацията на заразеното устройство и изнудва потребителя да му

плати откуп, за да получи ключ за дешифриране.

Repackaging – вж. препакетиране на приложение.

Rooting – рутване – за Android устройства – процес на придобиване на административни права за достъп (root достъп) до системните ресурси. Придобиването на административни права може да донесе следните рискове:

- Брикване на устройството (вж. bricking);
- Заплахи за киберсигурността;
- Загуба на гаранцията на устройството;
- Загуба на официални актуализации на софтуера.

Side Channel Attack – атака по странични канали – осъществява се чрез изтичане на информация от физическа криптосистема. Характеристиките, които могат да бъдат използвани при атака по странични канали, включват време, консумация на енергия и електромагнитни и акустични емисии.

SMiShing атака – вид атака, при която на потребителя се изпраща SMS или MMS. В краткото съобщение се съдържа URL адрес, който да доведе до инсталиране на зловреден софтуер.

Support Vector Machine (SVM) – алгоритъм за машинно обучение на софтуер, който се използва за класифициране на приложенията по показател „зловредни/добронамерени“.

Глава първа

ОЦЕНКА НА СЪВРЕМЕНОТО СЪСТОЯНИЕ НА ПРОБЛЕМА ПО СИГУРНОСТТА НА ОПЕРАЦИОННИ СИСТЕМИ ЗА МОБИЛНИ УСТРОЙСТВА

1.1. Методика на литературния обзор

В първа глава се решава първата научноизследователска задача на книгата, както следва: обзор на научните публикации, свързани с проблемите на уязвимостите в операционните системи за мобилни устройства.

В литературния обзор са включени предимно статии, намиращи се в квартална зона на Scopus или Web of Science, за да се спазят препоръките на специалисти в областта (Jennex 2015).

При търсенето на публикации в Scopus или Web of Science са използвани ключовите думи: [Android], [iOS], [malware], [security], [jailbreaking], [rooting] и допълнителен филтър за намиране на съответната публикация в един от четирите квартали. Разглеждани са основно статии, публикувани в списания, намиращи се в Q1 или Q2 за съответната година.

Състоянието на проблема може да бъде установено чрез ретроспективен обзор на научните публикации. Наистина първият вирус за ОСМУ е разработен през 2004 г. за ОС Symbian (Enck et al. 2009), но доколкото в уводната част бяха обосновани и въведени ограничения на предмета на изследването, по-надолу ще бъдат обследвани публикациите, свързани с Android и iOS. И по-конкретно, публикации, свързани със сигурността на Android 5.0 или по-нови версии, т.е. внедрени след октомври 2014 г. (GOOGLE 2022), и версии на iOS 10.0 или по-нови, т.е. внедрени след септември 2016 г. (Moreau 2021). Счита се, че така ще бъде решена първата научноизследователска задача, формулирана в увода на книгата.

1.2. Оценка на съвременното състояние на проблема по сигурността на операционни системи за мобилни устройства от групата на iOS

Първата версия на ОСМУ за устройства на компанията Apple е представена на 9 януари 2007 г., като мобилна версия на операционната

система за настолни компютри OS X (Moreau 2021). Apple получава право да използва името iOS за ОС на своите мобилни устройства след съдебен спор с компанията Cisco, която е собственик на търговската марка IOS (Marsal 2010).

Ще бъдат разгледани проблеми в научните публикации относно уязвимости в iOS до този момент.

Meng и колектив разширяват предишни изследвания върху Juice filming charging (JFC) (Meng et al. 2015; 2016), като разработват метод за установяване на посочената атака срещу смартфони, работещи под iOS и Android. За целта използват SVM класификатор, за да анализират натоварването на процесора на смартфона при наличие и отсъствие на атака (Meng et al. 2019).

Продължителността на изследванията в тази насока показва, че производителите не са се справили с тази специфична за смартфоните заплаха за сигурността.

Wang и колектив споделят опита си от прилагането на обфускация на кода на комерсиални приложения за iOS като метод за противодействие на заплахата от злонамерено реверсивно инженерство. Авторите оценяват ефективността от прилагането на метода по два показателя:

- Устойчивост – доколко обфускирането на кода го прави устойчив на реверсивно инженерство;
- Натоварване – в каква степен обфускирането се отразява на размера на изпълнимия код и на забавянето на изпълнението на приложението (Wang et al. 2019).

Horsman изследва възможността за извличане на информацията от IPA Siri на Apple, когато се взаимодейства със заключено устройство, работещо под управлението на iOS 11.2.5. Авторът доказва, че когато телефонът е конфигуриран да позволява взаимодействия със Siri и когато е заключен, е възможно частично възстановяване на информацията относно:

- Историята на обажданията;
- SMS;
- Телефонни контакти;
- Карти на Apple;
- Календар;
- Информация за устройството (Horsman 2019).

La Cour и колектив за първи път изследват съвременните безжични интерфейси за зареждане за възможност за реализация на *side channel attack*. Демонстрират атака за заснемане на *пръстов отпечатък* на уебсайт чрез безжично зарядно устройство за смартфони, работещи както под iOS, така и под Android, съответно Apple iPhone 11 и Google Pixel 4. Телефоните се поставят върху предавател за безжично зареждане, след което чрез скрипт се посещават уебстраници от предварително подготвен списък. Докато всяка от уебстраниците се зарежда, се записва количеството ток, което се консумира от предавателя за безжично зареждане. Събраните данни се използват за обучаване на невронна мрежа, чрез която впоследствие да се класифицира токовата следа, снета от телефона на жертвата.

При 10-секундни токови следи от iPhone 11 и Google Pixel 4 е постигната точност от над 90%, а когато следите са съкратени до 2,5 секунди, постигат точност, не по-малко от 80%.

Проучването показва също, че тази атака може да се осъществи, без да се разчита на скъпо или обемисто измервателно оборудване, като например високопроизводителен осцилоскоп. В експерименталната конфигурация е използван микроконтролер за измерване на тока, подаван към безжично зарядно устройство. Авторите считат, че е възможно да се конструира безжично зарядно устройство, в корпуса на което да се постави атакуващата схема. По този начин собствениците на смартфони няма да могат да идентифицират злонамерено или компрометирано зарядно устройство, поставено на публично място за зареждане на телефони (La Cour et al. 2021).

Markert и колектив за първи път провеждат обстойно изследване на сигурността на избрани от потребителите четири- и шестцифрени ПИН кодове за заключване на смартфон. Авторите установяват, че при ограничен брой опити, съответстващи на настройките за отключване на смартфона, използването на шестцифрени ПИН кодове, вместо четирицифрени, предоставя малко или никакво увеличение на сигурността и изненадващо, дори може да я намали. Също е изследвана ефективността от черните списъци, използвани в различни версии на iOS. В iOS черният списък се използва за съхраняване на определен брой най-често употребявани ПИН кодове. При опит за задаване на такъв ПИН, системата

извежда съобщение, което предупреждава потребителя и му препоръчва да избере друг код. Авторите отчитат, че след определен брой предупреждаващи съобщения, раздразнението на потребителите нараства и следователно размерът на черния списък трябва да се ограничи. Според техния анализ, оптимално съотношение между приложимост и сигурност на черния списък се постига при списък с дължина точно 1000 записа за четирицифрени ПИН кодове и списък с дължина точно 2000 записа за шестцифрени кодове (Markert et al. 2021).

Bhatt и колектив използват метода за *машинно обучение* Active Learning, в дефинирана от тях работна рамка iABC-AL (iOS Application Analyzer and Behavior Classifier using Active Learning). С помощта на работната рамка приложенията за iOS могат да бъдат класифицирани на базата на разрешенията, които са получили, и поведението им в следните групи:

- Полезни приложения – те са безопасни за използване и не довеждат до изтичане на специалните данни на потребителя.

- Подозрителни приложения – те използват излишни/допълнителни разрешения спрямо категорията си и може да доведат до изтичане на специални данни.

- Зловредни приложения – те споделят специални данни с домейни от трета страна, рекламни компании или аналитични фирми, без знанието и разрешението на потребителя (Bhatt et al. 2021).

Schultz и колектив тестват триизмерен модел на пръстов отпечатък за отключване на смартфони, работещи под Android и iOS. Пръстовият отпечатък е изработен от нанокompозитна смес от етилен-винил ацетат и графен. Авторите докладват най-малко 90% успеваемост при отключване на мобилни устройства на Apple. Успеваемостта е показана по-надолу, както следва:

- iPhone 6 (95%);
- iPhone 7(90%);
- iPhone SE (100%);
- iPad Pro (90%) (Schultz et al. 2021).

Трябва да се отбележи, че въпреки голямата му успеваемост, използваният метод не представлява сериозна заплаха за сигурността поради сложността на процеса на изработване на триизмерния модел.

Ciaramella и колектив предлагат метод за автоматично откриване на зловреден софтуер. Предложеният метод се състои от следните стъпки:

1. Чрез реверсивно инженерство се извлича асемблерният код на изследваното приложение.

2. Създаване на формален модел на приложението с помощта на описателен език за спецификация на автомати. В случая – Milner's Calculus of Communicating Systems (CCS).

3. Приложението се класифицира като полезно или зловредно след проверка на така създадения модел (Ciaramella et al. 2022).

Classen и колектив за първи път изследват заплахите за сигурността, свързани с безжичните чипове на iPhone, които продължават да работят, когато устройството е в режим на изтощена батерия. Поддръжката на този режим е на хардуерно ниво и всеки от чиповете може да бъде управляван индивидуално. Авторите анализират iOS 15 и доказват, че са възможни злонамерени действия, дори когато устройството не е в състояние да изпълнява основните си функции (Classen et al. 2022).

Salah и колектив използват алгоритми за *машинно обучение*, за да разпознаят операционната система чрез анализ на IPv6 мрежов трафик. Изследването обхваща както операционни системи за настолни компютри, така и Android и iOS. Авторите определят Decision Tree като най-ефективен алгоритъм, с постигната точност от 99% при най-кратко време за обучение (Salah et al. 2022).

Cooke фокусира вниманието върху метаданните на потребителите на мобилни устройства и стратегията на Apple да насърчи генерирането на тези данни от самите потребители. Тези метаданни, след като бъдат събрани, се анализират от компанията и се вписват в нейния бизнесмодел. Авторът разглежда тази дейност на Apple като посегателство върху сигурността на личните данни на потребителите. Cooke оценява настройката ProtectMyPrivacy (прев.: „Защити моята поверителност“), която може да бъде инсталирана само на устройства с *джейлбрейк*. Тази настройка ограничава възможностите на Apple да събира метаданни по два начина:

1. Като повишава осведомеността на потребителите относно събираните метаданни.

2. Като позволява на потребителите да контролират кои метаданни

се предоставят на Apple.

В този аспект авторът оценява прилагането на *джейлбрейк* и инсталирането на ProtectMyPrivacy като действие, допринасящо за защитата на личните данни на потребителите (Cooke 2020).

Laayu и колектив използват инструмента за *форензика* – MOBILedit Forensic Express, за да изследват iOS 14.8.1. При изследването е използвана процедурата на NIST в определена последователност: 1) съхранение; 2) придобиване; 3) изследване и анализ; 4) докладване на резултатите. Авторите сравняват възможностите за извличане на информация от устройство със и без *джейлбрейк*. Резултатите показват, че:

– С помощта на използвания софтуер е възможно да се направи копие на всички данни и настройки на телефона, включително изтрити данни, заснети снимки, данни от акаунти, контакти, съобщения, имейли, обаждания, информация от органайзера и инсталирани приложения.

– Повече информация се извлича от устройство, на което е приложен *джелбрейк* (Laayu et al. 2022).

Sardana и Bhatt изследват 331 приложения за iOS от три категории за наличие на излишни разрешения. За изследването авторите прилагат очертана от тях работна рамка CiAFP (Category-based Classification of iOS Apps by mining frequent permissions, или в превод: „Класифициране на приложения за iOS по категории чрез извличане на често срещани разрешения“). Експериментът се състои от два етапа:

1) Извличане на общо десет разрешения за всяка категория приложения чрез алгоритъма Apriori, който е една от наложилите се техники за извличане на асоциативни правила за намиране на често давани разрешения.

2) Прилагане на пет различни алгоритъма за машинно обучение, за да класифицират приложенията. Установено е, че в категориите „Образование“, „Финанси“ и „Здраве и фитнес“ има съответно 40%, 26% и 24% приложения с излишни разрешения (Sardana, Bhatt 2022).

При работата по книгата са констатирани два особени случая, които ще бъдат разгледани обособено.

Въпреки че изследването на Mehrnezhad и колектив излиза извън времето ограничение на литературния обзор за iOS, за отбелязване е, че те за първи път показват как сигурността на потребителите може да бъде компрометирана чрез JavaScript код, изпълнен в браузър за мобилен

телефон. Авторите използват факта, че в съответствие със спецификациите на W3C, мобилните уеббраузъри разрешават JavaScript код в уебстраница да получава достъп до данни от сензорите за движение и ориентация без разрешението на потребителя. Авторите разработват собствен код – TouchSignatures, който е в състояние да разграничи действията на потребителя при докосване (т.е. докосване, превъртане, задържане и мащабиране) и въвежданите ПИН кодове, което позволява на отдалечен уебсайт да научи дейностите на потребителя от страна на клиента. Атаката е приложима както за Android, така и за iOS. Авторите отбелязват, че са го докладвали на основните разработчици на енджини за браузър и са получили отговори, че ще отстранят уязвимостта (Mehrnezhad et al. 2016).

Mehrnezhad и Toreini изследват сензорите за мобилни устройства и политиките за разрешения, които Android, iOS и мобилните браузъри предлагат за тях. Фокусът на изследването е насочен към повишаване на осъзнатостта на потребителите относно рисковете за сигурността, които могат да възникнат при разрешаване на достъп до мобилните сензори. Резултатите показват, че познаването на сензорите не дава бързо отражение върху поведението на потребителите. Въпреки това натрупването на знания в тази област има силно влияние върху възприятията на потребителите в това отношение (Mehrnezhad, Toreini 2019).

Изводи и обобщения по сигурността на iOS

Извършеният преглед на научните публикации, свързани с повишаването на сигурността на iOS, способства да се формулират следните изводи:

- ОСМУ iOS е уязвима за *side channel* атаки.
- Алгоритмите за *машинно обучение* намират приложение в повишаването на сигурността на iOS, като се прилагат при класифициране на приложенията на базата на анализ на разрешенията, които получават; за откриване на *side channel* атаки; за разпознаване на операционната система чрез анализ на IPv6 мрежов трафик.
- Преобладаващото мнение е, че *джелбрейкът* е опасен за сигурността, но също така може да бъде разглеждан и като средство за защита на личните данни на потребителя.

Последните две публикации логично повдигат въпроса, дали

към момента на настоящото изследване проблемът със сигурността на уеббраузърите е решен, което обосновава научноизследователската задача в последната глава на книгата.

1.3. Оценка на съвременното състояние на проблема по сигурността на операционни системи за мобилни устройства от групата на Android

Първата версия на ОСМУ Android се нарича Astro и е внедрена през септември 2008 г. на мобилен телефон HTC Dream (Krajci, Cummings 2013). Още през 2009 г. се появява публикация, свързана със сигурността на Android ОС. Изследвани са моделът за сигурност на ОС, механизмът на комуникация между приложенията и системата за получаване на разрешения за достъп. Посочени са слабостите в модела за сигурност и е предложена работна рамка, която надгражда този модел, и по този начин би подобрила системата за сигурност на изследваната версия на Android (Ongtang et al. 2009).

Предложена е услуга по сигурността, наречена Kirin. Тя сертифицира приложенията за Android, за да намали риска от инсталиране на злонамерени приложения. Това се постига чрез шаблони с правила за сигурност, които се прилагат при инсталиране на ново приложение (Enck et al. 2009).

В съответствие с въведените ограничения, е редно да се посочат следните публикации.

Подобен подход на Kirin е описан и в доклад на Chenkai Guo, където е предложен прототип на система за анализиране на разрешенията. Описаният софтуер е наречен MalDetector (Guo et al. 2015).

Също, анализ на декомпилиран код, с цел да се установи наличието на разрешения за достъп до чувствителна информация се използва от Wang и колектив (Wang et al. 2015).

Munoz и колектив изследват набор от приблизително 25 хил. приложения от Google Play Store. Използват се статистически методи, за да се анализират 48 показателя от метаданните за приложенията. В резултат на изследването са определени метаданни, чрез които може да се прогнозира дали дадено приложение е зловредно, или не (Munoz et al. 2015).

Ко и колектив прилагат метод за анализиране на декомпилирания код, за да установят наличие на многослойна атака. При този вид атака се използват три приложения с различни права за достъп. Авторите установяват, че всяко от отделните приложения преминава проверките с антивирусен софтуер, а зловредните действия се осъществяват в резултат от координираното действие на трите приложения (Ko et al. 2015).

Khandelwal и Mohapatra разглеждат процесите на публикуване на приложения за Android и iOS, като отбелязват, че за изследвания период приложенията за iOS имат по-висока сигурност поради анализирането и одобряването на кода на приложенията, преди да бъдат публикувани в App Store (Khandelwal, Mohapatra 2015).

Същото проучване посочва следните уязвимости за устройства, работещи под Android:

- Зловреден софтуер – вируси, червеи, троянски коне.
- Ботнет – мрежа от компрометирани устройства, която може да бъде контролирана чрез отдалечен достъп.
- Разширена постоянна заплаха – кибератака, изпълнявана от група висококвалифицирани хакери, които систематично компрометират мрежата на специфична цел за продължителен период.
- Уязвимости с повишаване на правата за достъп.
- Rootkit – злонамерено или опасно приложение, което получава право да работи в привилегирован режим. Такива вредни приложения обикновено крият съществуването си от потребителя, като променят стандартните функционалности на операционната система.

И предлага механизми за защита, приложими за Android:

- Антивирусен софтуер (Antimalware);
- Системи за откриване/предотвратяване на проникване;
- Linux контрол на достъпа до системните ресурси;
- Контрол на достъпа чрез разрешения за Android;
- Шифроване на данни/Шифроване на телефонни разговори;
- Проверка за сертификат за приложение;
- Проверка на интегритета;
- Използването на заключване на устройството с трудни за отгатване пароли;
- Филтри за спам;

– Дистанционно управление, в смисъл – защита на устройството при кражба или загуба;

– Управление на ресурсите, в смисъл на избягване на DoS атаки.

Xenakis и Ntantogian доказват възможността за разработване на злонамерен софтуер, който да атакува *baseband* модема на Android или iOS мобилни телефони. Разработеното приложение може напълно да компрометира сигурността на мрежовата връзка, осъществена по клетъчната мрежа, като използва команди за управление на модема, известни като AT commands (Xenakis, Ntantogian 2015). Този подход е използван при клетъчни мрежи с 2G и 3G технологии.

Canfora и колектив доказват нов начин за конструиране на зловреден софтуер, който се изгражда динамично. При този софтуер злонамереният код се композира по време на изпълнение на приложението. Така злонамереният код не може да бъде открит при статичен анализ (Canfora et al. 2015).

Buhov и колектив разглеждат мрежовата сигурност на мобилни телефони и по-специално използването на протоколите HTTP и HTTPS в приложения за Android (Buhov et al. 2015).

Vecchiato и колектив разглеждат политиката bring your own device (BYOD), при която на служителите се позволява с техни собствени мобилни устройства да осъществяват достъп до корпоративните данни и системи. Изследвани са новите проблеми със сигурността и поверителността, които възникват при този начин на работа. Предложен е инструмент, анализиращ потребителските настройки за сигурност на мобилното устройство (Vecchiato et al. 2015).

Alothali и колектив анализират разрешенията и свързаните с тях рискове за сигурността при инсталиране на безплатни игри в Google Play (Alothali et al. 2015).

Kotkar и Game разработват приложение, което работи във фонов режим и следи за опити за изпращане на кратки текстови съобщения без знанието на потребителя. При констатиране на такъв опит се появява предупреждаващо съобщение за потребителя (Kotkar, Game 2016).

Soh и колектив предлагат подход за установяване на „*препакуване*“ на приложения чрез анализиране на потребителския интерфейс на приложенията по време на изпълнение (Soh et al. 2015).

Aysan и Sen изследват възможността за добавяне на зловреден код при обновяване на приложения, които вече са инсталирани. Направен е статичен и динамичен анализ на приблизително 30 хил. приложения, които използват техниките за обновление: надграждане, тихо инсталиране (възможно само при устройства с root права) и динамично зареждане на класове. В резултат на изследването са открити 70 нови зловредни приложения, които преди това са успели да издържат проверките на VirusTotal (Aysan, Sen 2016).

Andriatsimandefitra и Tong използват Dynamic Information Flow Tracking (DIFT), за да установят зловредно действие на приложения по време на тяхното изпълнение (Andriatsimandefitra, Tong 2016).

Liang и колектив анализират механизма за защита на поверителността на Android и описват различни заплахи за различните видове специални потребителски данни (Liang, Wu, Xu, et al. 2016).

Li и колектив предлагат схема за откриване на зловреден софтуер за Android, използвайки SVM базиран подход (Li et al. 2016).

Kurniawan и колектив използват *машинно обучение* за установяване на наличието на зловреден софтуер в Android. В изследването се анализират аномалиите в консумацията на енергия, температурата на батерията и данните за мрежовия трафик, като се използва алгоритъм за класифициране на злонамерен и добронамерен софтуер (Kurniawan et al. 2015).

Westyarian и колектив използват *машинно обучение*, за да установят наличието на злонамерен софтуер. Те анализират системните обръщания към API, имащи отношение към разрешения, които са декларирани в AndroidManifest.xml файла (Westyarian et al. 2015).

Park и колектив предлагат система за установяване на зловредни приложения за Android, реализиращи атаката SMiShing. Използват комбинация от клиентска програма, инсталирана на телефона на потребителя, и клъстеризиран сървър. Клиентската програма проверява приложенията, инсталирани на смартфона на потребителя, и изпраща анонимна информация към сървъра за входящия трафик от кратки съобщения (SMS или MMS). На сървъра се съхраняват два черни списъка:

- 1) С източници на зловредни съобщения и зловредни линкове.
- 2) Със злонамерени .apk файлове.

Ако информацията от клиентската програма съвпада с някой от черните списъци, съобщението се определя като зловредно.

Ако информацията от клиентската програма липсва в черните списъци, но има линк за изтегляне на приложение, то приложението се изтегля на сървъра и се проверява и класифицира като полезно или зловредно, преди да се инсталира на телефона на потребителя (Park et al. 2015).

Liang и колектив предлагат внедряването на допълнителни услуги към Android, с цел да се осигури поверителност и персонализация за потребителите. Прототипът, който внедряват, се нарича PAPDroid и се състои от две системни услуги:

- Услуга за поддръжка на персонализация (PSS).
- Услуга за защита на поверителността (PPS), за укрепване на сигурността на поверителността на системата Android.

PPS първо оценява нивото на риска на всяко приложение, отново чрез *машинно обучение*, и след това съответно налага различна политика за защита на поверителността (Liang, Wu, Song, et al. 2016).

Zaman и колектив анализират мрежовия трафик, за да установят наличието на зловреден софтуер, инсталиран на мобилното устройство. За целта се проследяват връзките на всички приложения към отдалечени локации за определен период от време, след което се сравняват с база от данни с известни злонамерени домейни (Zaman et al. 2015).

Yang и колектив предлагат софтуер, който извършва статичен анализ на байткода на приложенията, с цел да установи пасивно изтичане на специални данни (Yang et al. 2016).

При активното изтичане на данни потребителят инсталира зловредното приложение, като при това му дава необходимите разрешения за достъп до специална информация.

Пасивното изтичане на данни може да се случи по време на следните етапи от жизнения цикъл на приложението:

- По време на етапа на проектиране на приложението;
- При разработване на приложението;
- При използване на приложението на база на съществуващи недостатъци в сигурността на системата, при условие че изтичането на специални данни е без разрешението на потребителя (Yang et al. 2016).

Bezobrazov и колектив предлагат система, която се основава на интеграцията на изкуствена имунна система и изкуствени невронни мрежи и има за цел да открива и блокира нежелани и злонамерени приложения за Android. Предложената система е част от разработваната от екипа „интелигентна система за киберотбрана“, която е насочена към откриване на злонамерени приложения в Android. Предложената система се състои от набор от „интелигентни“ имунни детектори (всеки детектор, базиран на невронна мрежа). Всеки детектор има период от време, наречен живот, през който той преминава през етапи, като създаване, обучение, селекция и откриване.

Модулът за генериране на детектори произвежда набор от „незрели“ детектори. След това, по време на етапа на обучение, имунните детектори се научават да коригират класификацията на обекти в средата на операционната система и да откриват злонамерен код.

След обучение всички имунни детектори преминават през етапа на подбор, където се проверява тяхната коректност. Ако детектор класифицира тестови доброкачествени обекти като злонамерени, той се унищожава и се заменя с нов детектор. Модулът за избор позволява намаляване на честотата на фалшиви аларми и повишаване на нивото на защита на системата.

Всяко новоинсталирано приложение се проверява от набор от детектори, които го анализират. Ако детекторите класифицират новото приложение като доброкачествено, то може да бъде инсталирано на устройството. Ако някой детектор класифицира приложение като злонамерено, то се блокира.

Когато бъде открито ново злонамерено приложение, системата извлича данни от това приложение и ги добавя към данните за обучение. Междувременно детекторът, който е открил нов злонамерен код, се трансформира в детектор за имунна памет. Механизмът за актуализиране на обучителните данни позволява да се развие цялата защитна система и осигурява възможност за адаптиране към новите, непознати злонамерени приложения. Механизмът на имунната памет осигурява високо ниво на реакция при повтарящи се опити за известни атаки (Bezobrazov et al. 2016).

Chen и Zhang предлагат приложение за Android, което да оценява мрежовата сигурност на мобилни устройства, свързани към WiFi мрежи. Предложеното приложение функционира само на устройство с root достъп (Chen, Zhang 2016).

Ji и колектив предлагат работна рамка за статичен анализ на изпълнимия код на приложения за Android, с използване на SVM и облачни технологии. Данните за новоинсталирано приложение се изпращат в облака, където приложението се класифицира като зловредно или полезно. За класифицирането се използва невронна мрежа и k-Nearest Neighbor алгоритъм (Ji et al. 2016).

Takahashi и Van използват *машинно обучение* за анализ на метаданните на приложенията и по този начин да ги класифицират като зловредни или полезни (Takahashi, Van 2016).

Kanwal и колектив разглеждат опасността от използването на *рансомуеър* за мобилни телефони, работещи под Android. Също така проучват наличните приложения за преодоляване на този вид заплаха (Kanwal et al. 2016).

Malik и Khatter използват самостоятелно разработения инструмент Androdata за извличане на две функции на приложенията за Android – искани разрешения и извиквани системни повиквания (по отношение на честотата). В изследването са разгледани два набора от по над 500 приложения, като единият набор е с доброкачествени приложения, а другият – със зловредни. Направено е сравнение между видовете и количеството разрешения, които изискват доброкачествените и зловредните приложения (Malik, Khatter 2016a).

Sehgal и Kumar също насочват вниманието към разрешенията, които дава потребителят при инсталиране на приложения за Android. Авторите предлагат уебплатформа, която да дава информация за риска, който се поема от потребителя, преди приложението да бъде инсталирано (Sehgal, Kumar 2016).

Harris и Chin изследват разрешенията, които се изискват от приложения по време на инсталация. Дефинират 13 най-рискови разрешения, искани от приложенията, и проучват количеството рискови разрешения за приложения, разработени от доверен разработчик. Потребителите са накарани да вярват, че значката за най-добрите разработчици ги защи-

тава с безопасни приложения. Авторите доказват, че това е заблуда и потребителите трябва разумно да ограничат изтеглянето на приложения, които изискват рискови разрешения, независимо от техния печат за доверие (Harris, Chin 2016).

Dar и Parvez посочват следните недостатъци на рамката за сигурност на Android:

– Потребителят, който инсталира приложение за Android, е принуден да приеме всички разрешения, за да може успешно да инсталира приложението. Потребителят не може да приеме едно и да отхвърли друго разрешение, тъй като това може да доведе до прекъсване на инсталацията.

– Програмистите имат свободата да пишат всякакъв вид код в Android. Те могат да поискат n на брой разрешения, за да постигнат определена функционалност, която би могла да бъде постигната и с по-малко разрешения.

– Начинаещите потребители не са в състояние да разберат целесъобразността на разрешенията, които им се искат, и почти не се интересуват от тези разрешения. В резултат на това начинаещите потребители са в постоянна заплата да бъдат компрометирани личната им информация и сигурността им (Dar, Parvez 2016).

Съответно предлагат следните решения:

Техника 1. Сигурност, базирана на нуждите (NBS). При този метод се използва реверсивно инженерство, за да се извлекат всички файлове от *.apk* файла. Процесът включва извличане или декомпилиране, модифициране на изходния код и повторно компилиране. Целта е да се премахнат ненужните разрешения. Това се постига чрез модифициране на *AndroidManifest.xml*, в който са изброени всички разрешения, искани от програмиста при създаването на приложението. След анализ на разрешенията се премахват допълнителните или ненужните разрешения, които не са в областта на приложението и може да представляват опасност за сигурността. След това се дава допълнителна свобода на потребителите, като им се предоставя екран за разрешение по време на изпълнението, който казва на потребителя, че това приложение се опитва да получи достъп до ресурса. Потребителят може да приеме или отхвърли тази заявка по време на изпълнението.

Dar и Parvez разработват оригинална техника, но не предлагат вариант за автоматизиране на този процес и в същото време посягат на авторските права на разработчика на софтуера.

Техника 2. Проверка на сигурността чрез API. Целта на тази техника е да даде на потребителя допълнителна свобода да приеме всички разрешения при инсталиране на приложението. Потребителят не е обезпечен по никакъв забележим начин, а API за сигурност, работейки във фонов режим, се грижи за всички въпроси, свързани с поверителността на потребителя.

Техника 3. RD&P система (Runtime Detection and Prevention). В тази техника авторите създават приложение, наречено controller, което ще се грижи за активизирането на всички приложения по време на изпълнение. Ако приложение се опита да осъществи достъп до някой от ресурсите във фонов режим, без да информира потребителя, контролиращото приложение незабавно информира потребителя за тази злонамерена дейност.

Jha и Lee изследват базираната на разрешения сигурност в Android от гледна точка на експертите по политики за сигурност, разработчиците и крайните потребители (Jha, Lee 2016).

Mercaldo и колектив предлагат методология, базирана на формални методи, която е в състояние да открие *рансъмуеър*. В същото време и да идентифицира в кода на зловредния софтуер инструкциите, които изпълняват характерните действия на *рансъмуеър* (Mercaldo et al. 2016).

Malik и Khatter използват динамичен анализ на системните обръщения за констатиране на семейства от зловредни приложения (Malik, Khatter 2016b).

Song и колектив също насочват вниманието към улавянето на *рансъмуеър*, но използват динамичен анализ на натоварването на процесора, използването на паметта и входно-изходните операции, необходими за всеки процес (Song et al. 2016).

Somarriba и колектив предлагат работна рамка, която се състои от четири компонента: вграден клиент, филтър, уебслужба и визуализация. Работната рамка се използва за проследяване и визуализация на аномалии в извикванията на функции в приложения за Android. Мониторингът се осъществява по време на изпълнение на приложенията. По този начин

всяко приложение може да бъде наблюдавано без необходимост от root права (Somarriba et al. 2016).

Yuan и колектив използват технологията за изкуствен интелект Deep Learning (невронна мрежа с три или повече слоя), за да създадат онлайн енджин (DroidDetector), който може автоматично да установи дали дадено приложение е зловредно, или не (Yuan et al. 2016).

Mehrnezhad и колектив за първи път показват как сигурността на потребителите може да бъде компрометирана чрез JavaScript код, изпълнен в браузър за мобилен телефон. Атаката е приложима както за Android, така и за iOS (Mehrnezhad et al. 2016).

Allix и колектив насочват вниманието си към разликата в производителността на класификаторите на зловреден софтуер, използващи алгоритми за *машинно обучение*. Изследването показва, че в реални условия производителността на класификаторите е много по-ниска.

С помощта на собствен инструмент съставят дейтасет (набор от образци на предполагаемо добри/добронамерени приложения) от 52 000 приложения, изтеглени от Google Play. Оказва се, че в дейтасета се намират приложения, които VirusTotal определя като зловредни и преди това не са били известни (Allix et al. 2016).

Мао и колектив разглеждат сигурността на *хибридни мобилни приложения*. Предимство на тези приложения е, че могат да бъдат изпълнени на различни платформи – Android, iOS или друга. Предлагат динамично решение за идентифициране на свръхзаявени разрешения от *мобилни хибридни приложения*. За целта използват работна рамка, която динамично извлича необходимите разрешения по време на изпълнение на приложението. След това извлечените разрешения се сравняват с разрешенията, декларирани от разработчика, и по този начин идентифицират свръхзаявените разрешения (Mao et al. 2016).

Eze и колектив отчитат факта, че когато потребителите дават разрешение за инсталиране на дадено приложение, те обикновено имат слаба представа за чувствителната информация и ресурсите, до които има достъп този софтуер. Авторите проучват възприятието и реакцията на потребителите, когато им бъде представен визуален анализ на дейностите на приложенията за Android и последствията за сигурността. За целта се използват интерактивни визуални схеми, изобразяващи въздействието

от дейностите на приложенията. Eze и колектив демонстрират, че когато на потребителите се представят визуални схеми за дейностите на приложенията, те стават по-осъзнати и чувствителни към посегателството върху личните им данни (Eze et al. 2016).

Andriotis и колектив изследват физическата защита на устройството и по-конкретно шаблоните за заключване на екрана (Andriotis et al. 2016).

Piao и колектив разглеждат замаскирането на кода на *.apk* файловете с помощта на DexGuard, като средство за защита от атаката „*препакетиране*“. В изследването се посочват някои слабости на DexGuard и авторите предлагат техника на *обфускиране*, базирана на модела „клиент – сървър“.

При предложената техника оригиналният *.apk* файл се разделя на две части. Едната част е подпрограма, а другата е основна процедура, съхранена на сървър. И двете процедури се *обфускират* с помощта на типични методи за *обфускация*, като преименуване на идентификатори, рандомизация на контролния поток и обфускация на низове. Основната процедура добавя процедури за откриване на подправяне. В резултат на предложената техника се получава нов *.apk* файл с по-висока степен на защита, който се инсталира на смартфона на потребителя (Piao et al. 2016).

Caviglione и колектив изследват откриването на две зловредни приложения, които действат съвместно на едно устройство и комуникират помежду си чрез скрит канал. Предлагат подход за установяване на наличието на такива приложения, като използват невронна мрежа и дърво на решенията, и с тези инструменти анализират потреблението на енергия в мобилното устройство. Чрез експеримент анализират седем скрити канала за комуникация, два от които са предложени от авторите (Caviglione et al. 2016).

Xiao и колектив използват динамичен анализ, за да установят наличието на зловреден софтуер чрез анализ на последователност от системни обръщания (Xiao et al. 2016).

Farina и колектив разработват ботнет инфраструктура за мобилни телефони и доказват възможността тя да бъде използвана за осъществяване на бавни DoS атаки (Farina et al. 2016).

Jang и колектив предлагат система за точно откриване и класификация на зловреден софтуер, наречена Andro-Dumpsys. За откриването на зловредния софтуер се използва комбиниран подход от malware-centric и malware creator-centric методи за анализ. Използват Volatile memory acquisition (придобиване/прехващане на променливата памет) за анализ на кода на приложенията (Jang et al. 2016).

Meng и колектив разработват атака juice filming. При тази атака информацията от дисплея на мобилното устройство може да изтече през стандартен микро USB конектор, чрез стандарта Mobile High-Definition Link (MHL) за Android устройства. Атаката е приложима и за iOS устройства. Особеност на предложената атака е, че не изисква инсталирането на никакъв софтуер на мобилното устройство (Meng et al. 2015). Същият основен автор и колектив доказват възможността за разработване на зловредно „зарядно“ устройство, което да записва съдържанието на дисплея на смартфона, докато той се зарежда (Meng et al. 2016). По този начин може да бъде извлечена специална информация.

Xu и колектив предлагат метод, наречен ICCDetector. Методът се базира на следене на шаблоните в комуникацията между отделните компоненти на приложенията – inter-component communication (ICC). Използва машинно обучение, SVM и Decision Tree (Xu et al. 2016).

Емпирично проучване от 2017 г. обследва 660 уязвимости, свързани с Android. Установява се, че за периода на изследването най-често засегнатият слой е слойът на Linux ядрото (Linares-Vasquez et al. 2017).

Изследването е разширено от почти същия екип от учени, като обхваща период от 10 години (2008 – 2017). Изследвани са 1235 уязвимости за Android, извлечени от CVE Details, NVD и официалните бюлетини с уязвимости на Google. Предложена е детайлна таксономия на уязвимостите за Android (Mazuera-Rozo et al. 2019).

В публикация от 2018 г. се докладват резултатите от изследванията на комуникацията между отделни компоненти на различни приложения, работещи под Android. Разгледани са уязвимостите и съответните им code smells. Обсъдени са варианти за тяхното смекчаване по време на разработването на приложенията. Предложен е софтуер, който извършва статичен анализ по време на разработване на изходния код на приложенията и осигурява обратна връзка в средата за разработка с цел избягва-

не на code smells (Gadient et al. 2018).

Kotterel и колектив анализират състоянието на системата от разрешения за приложения, работещи под ОС Android, като отчитат слабостите на внедрените към 2015 г. версии. Посочват методи за защита на приложения, а именно: Google Bouncer – софтуер, който анализира приложенията за зловреден код, преди да бъдат публикувани в Google Playstore; антивирусен софтуер; анализ на видовете разрешения, преди приложението да бъде инсталирано; следене на приложенията по време на изпълнение с помощта на допълнителен софтуер. Резултатът е предложено специализирано приложение, с помощта на което се модифицират приложенията преди инсталиране. С помощта на приложението се пренаписват точките на уязвимост по време на изпълнение (Cotterell et al. 2015).

Bernardi и колектив използват езика Declare и съпътстващите го инструменти (плъгина Declare Miner за ProM 6.1) съответно за разпознаване и представяне на поведенческия модел на зловреден софтуер (дефиниран като набор от системни повиквания и техните взаимоотношения). Този подход позволява не само разпознаването на зловредните приложения, но и тяхното класифициране (Bernardi et al. 2019).

Ananya и колектив експериментално доказват значението на правилния подбор на характеристики на приложенията, използвани за трениране на алгоритми за *машинно обучение* при откриване на зловредни приложения за Android. Предлагат собствен метод SAILS (Selection of relevant Attributes for Improving Locally extracted features using classical feature Selectors) за подбор на подходящи характеристики (Ananya et al. 2020).

Diamantaris и колектив проучват възможността за атаки на Android ОС чрез WebAPI на HTML 5. В изследването се използва динамична система за анализ на приложения в реално време. Проучени са 183 571 от най-популярните уебсайтове през периода март – септември 2018 г.

Установени са 5313 уникални домейна, които достъпват поне едно мобилно WebAPI повикване; 35,89% от тях също така водят до достъп до сензори от страна на скриптове от трети лица, които идват от 11 домейна на второ ниво. Авторите предлагат таксономия на атаките през WebAPI (Diamantaris et al. 2020).

Casolare и колектив използват динамичен анализ на системните повиквания, за да създадат изображение в PNG формат, чрез което да представят приложенията за Android. След това използват различни алгоритми за *машинно обучение*, за да класифицират изображенията и по този начин да разграничат зловредните от стандартните приложения (Casolare et al. 2021).

Surendran и Thomas анализират системните повиквания на приложенията за Android и на тази основа създават подреден граф, чрез който се описва поведението на приложението. След това така създаденият граф се подава към класификатор, използващ алгоритъм за *машинно обучение*. Авторите посочват 0,99 точност при установяване на вредоносни приложения по скалата от 0 до 1 (Surendran, Thomas 2022).

Chimuso и колектив разглеждат сигурността на мобилни приложения, използващи облачни технологии. Представят актуална таксономия на атаките към облачни и мобилни екосистеми и предлагат защитни механизми (Chimuso et al. 2023).

Shao и колектив разглеждат предимствата и недостатъците на придобиването на root права за Android устройства от страна на потребителя. Авторите предлагат RootGuard, разработен от тях инструмент за защита на устройство след прилагане на *rooting*. Софтуерът позволява на потребителите да предоставят на дадено приложение необходимите оперативни привилегии, в съответствие с използваните от него системни извиквания и параметри. Същевременно RootGuard поддържа политики по подразбиране, за да предпази ОС Android от атаки със зловреден софтуер (Shao et al. 2014).

Jin и Zhang доказват възможността чрез анализ на системните журнали (log) файлове на ОС Android да разпознаят дали устройството е рутнато (Jin, Zhang 2017). Като недостатък на този метод може да се посочи, че той не е автоматизиран.

Съществуват приложения с високи изисквания за сигурност, които имат вградена защита от изпълнение на устройство с root права. В своята публикация Soewito и Suwandaru разглеждат ситуация, в която потребителят съзнателно заобикаля вградената защита на приложението с помощта на прехващане на системното повикване, установяващо наличието на рутване. Авторите предлагат решение, което разработчиците да

прилагат, за да избегнат този вид злоупотреба (Soewito, Suwandaru 2022).

Elsersy и колектив разработват метод, наречен Rootector, за установяване на рутване на Android устройства чрез техники за класификация чрез машинно обучение. В изследването се оценява производителността на десет машинни класификатора, за да се идентифицира най-добрият модел за класификация. Резултатите от експерименталната оценка показват над 97% точност и определят Deep Learning като най-подходящ класификатор за предложената работна рамка (Elsersy et al. 2023).

Изводи и обобщения по сигурността на ОС Android

Ако обобщим така извършения преглед на научните публикации, свързани с повишаването на сигурността на ОС Android, то се очертава комплекс от ясно изразени мерки за справяне с проблема, както следва:

1. Анализ на потребителските настройки на устройството за евентуални пропуски в сигурността.
2. Анализ на разрешенията за достъп до ресурсите на устройството, при инсталиране на дадено мобилно приложение.
3. Анализ на изходния код на приложението, преди да бъде компилирано.
4. Статичен анализ на изпълнимия байткод на приложението.
5. Динамичен анализ на поведението на приложението по време на неговата работа. Такъв анализ може да се извърши в сигурна среда, преди приложението да бъде инсталирано на реално устройство на краен потребител. Анализът може да се извърши от приложение, работещо във фонов режим, следящо за подозрителни дейности.
6. Анализ на системните повиквания (system calls).
7. Анализ на поведението на приложенията за установяване на наличието на атака, осъществена от съвместното действие на няколко приложения. Примери – *privilege escalation*; динамично създаване на вирус.
8. Анализ на мрежовия трафик с цел установяване на зловредни действия.
9. Използване на код на JavaScript, който се изпълнява в раздел на браузър, работещ във фонов режим.
10. Откриване на слабости в хибридни мобилни приложения.
11. Повишаване на осъзнатостта на потребителя при приемане на

исканията за достъп до системните ресурси по време на инсталиране на приложението.

12. Използване на алгоритми за машинно обучение за констатиране на *rooting* на устройството.

13. Използване на алгоритми за машинно обучение за откриване на зловредни приложения. Алгоритмите за машинно обучение са многоаспектен инструмент в борбата за повишаване на сигурността на ОСМУ, ясно кореспондиращ с темата на книгата.

14. Разкриване на скрити комуникации на мобилен зловреден софтуер чрез следене на консумация на енергия с помощта на изкуствен интелект.

Изводи и обобщения по глава първа

Историческата ретроспекция показва, че още с внедряването на първите версии на ОСМУ се появяват уязвимости.

При прегледа не са открити публикации, оценяващи базите данни, в които се съхраняват и класифицират уязвимостите за мобилни устройства. Логично е да се направи изследване, което да оцени как се съхраняват уязвимостите и как след това тези уязвимости могат да бъдат открити. Така направеният частен извод обосновава необходимостта от решаване на втората научноизследователска задача към втора глава.

Несъмнено, един от откритите въпроси за сигурността на ОСМУ произтича от разрешенията, които се дават от потребителите при инсталиране и използване на приложенията.

От извършения преглед следва, че проблемите на уязвимостите се намират във фокуса на вниманието на изследователите на сигурността на ОСМУ на практика от внедряването в експлоатация на тези операционни системи.

По тематика научните публикации по проблема условно могат да се разделят на четири кръга:

1. Публикации с принос към изграждане на терминологичния речник.

2. Публикации, свързани с недостатъчната техническа квалификация на масовия потребител на мобилни устройства, което води до заплахи за сигурността.

3. Процедури за откриване и борба със зловредния софтуер.

4. Особено място като инструмент за изследване заемат алгоритмите за машинно обучение. Те имат двойно приложение. От една страна, да установят уязвимост, например чрез класифициране на зловредни приложения. От друга страна, да демонстрират определена атака. Могат да се дадат следните примери:

– Сваляне на отпечатък на посещавани сайтове чрез проследяване на енергийното потребление на безжично зарядно устройство.

– Разпознаване на ОС чрез анализ на мрежов трафик.

Следва да се подчертае, че алгоритмите за *машинно обучение* са едно от основните средства, използвани в методите за повишаване на сигурността на ОСМУ, пряко свързани с темата на книгата.

Откриват се публикации относно уязвимости, общи и за двете операционни системи, например достъп до сензорите на мобилното устройство с помощта на JavaScript код, вграден в HTML страница. Възниква въпросът, дали тези уязвимости са отстранени? Така направеният частен извод обосновава необходимостта от решаване на четвъртата научноизследователска задача към трета глава на книгата. Но с оглед на факта, че тези проблеми не са решени, а в същото време заемат ключово място, процесът на изследване ще продължи и в бъдеще.

Отделно следва да се подчертае, че независимо от усилията за борба със зловредния софтуер, на официалните *marketplace* (пазарно място) на приложения, като Google Play Store и Samsung Galaxy Store, продължават да се намират зловредни приложения (DOCTOR WEB 2022).

Глава втора

АНАЛИЗ НА ПУБЛИЧНО ДОСТЪПНИ НАЦИОНАЛНИ БАЗИ ДАННИ, СЪДЪРЖАЩИ ОПИСАНИЯ НА УЯЗВИМОСТИ ЗА ОПЕРАЦИОННИ СИСТЕМИ ЗА МОБИЛНИ УСТРОЙСТВА

В съответствие със структурата на книгата, в настоящата глава ще бъдат решени втора и трета научноизследователска задача.

В един съвременен мобилен телефон има възможност за плащания онлайн, двуфакторно удостоверяване при парични преводи, получаване и изпращане на електронна поща, проследяване на местоположението на устройството, записване на видео- и звукови файлове, заснемане на фотографии. Тази голяма функционалност предоставя възможност за много злоупотреби.

По данни на сайта CVE details, за последните три календарни години са открити 2329 уязвимости за различни версии на операционната система Android, а за 2023 г. вече има открити над 27 уязвимости (CVE Details 2023a). Уязвимостите за различни версии на iOS, открити за същия период, са 621 (CVE Details 2023b).

Ето защо точното идентифициране на уязвимостите в ОСМУ и тяхното откриване в БДУ е от голямо значение за практиката.

За първи път през изследвания период NVD се почва от Umasankar, като източник на информация за уязвимости за Android (Umasankar 2017).

Tiwari и Velayutham посочват NVD и CVE като основни източници на информация за уязвимости за Android. Разработват приложение, което автоматично да извлича описания на уязвимости от NVD, и споменават за трудности при извличане на детайлната информация относно конкретни уязвимости (Tiwari, Velayutham 2019).

Уязвимостите на Android за периода 2008 – 2017 г. са задълбочено изследвани и класифицирани чрез „проучване на CVE, NVD и официалните бюлетини за сигурност на Google“ (Mazuera-Rozo et al. 2019). В същото изследване се споменава, че има уязвимости в огледалната БДУ

на NVD – CVE details, които са „неправилно класифицирани“ (Mazuera-Rozo et al. 2019). Подобна констатация е едно от основанията, за да се задълбочи анализът на възможностите на БДУ за коректно извличане на информацията относно уязвимости за ОСМУ.

Националните БДУ, NVD, CNVD, CNNVD, JVNDB, наред с други публично достъпни БДУ, са изследвани относно техните възможности за предоставяне на конкретна информация за уязвимости за IoT устройства (Rytel et al. 2020).

И последно, NVD, CNVD, CNNVD са изследвани по отношение на информацията, която предоставят (Forain et al. 2022), но не разглеждат възможностите за филтриране и извличане на данни за група уязвимости по определен признак, например версия на ОСМУ.

Прави впечатление фактът, че за изследвания период по темата не се откриват публикации на български език или от България, индексирани в Scopus или Web of science.

Следователно може да се твърди, че по посочените БДУ е направен опит за комплексно изследване, но в крайна сметка – с пропуски по отношение на филтрирането на данни за уязвимости за ОСМУ. И това е допълнително основание за решаване на втората научноизследователска задача.

2.1. Метод на балните скали и неговото приложение при БДУ

2.1.1. Теоретична постановка на метода на балните скали

Методът на балните скали е разработен през 1928 г. от американските учени L. L. Thurstone и E. J. Chave за нуждите на експерименталната психология, но впоследствие намира приложение като метод за оценка на качеството изобщо. При използването на научнообоснован подход за разработване на балните скали, при ангажиране на оценители с нужната квалификация и при спазване на всички методологични изисквания, прилагането на този метод дава възможност да се получат обективни, надеждни и достоверни резултати при определяне на оценките.

Същността на метода се състои в това, че изследваният признак (свойство) се разделя на няколко нива, подредени във възходящ или низходящ ред. На всяко от нивата съответстват определени числени стой-

ности, наречени балове. Първото и последното от нивата съответстват на крайните степени на проява на свойството. Оценките, присъждани по метода на балните скали, са безразмерни величини.

Когато се прилага методът на балните скали, се въвеждат две основни понятия:

– Градация на балната скала – това е разликата между две съседни бални оценки в скалата.

– Размах на балната скала – това е броят на градациите в скалата.

Съществуват различни видове бални скали. В зависимост от броя и мястото на опорните нива, различаваме: дискретни бални скали, непрекъснати и скали от смесен тип (Златева 2013, с. 117 – 118).

За целите на изследването ще бъде използвана само *дискретната бална скала*. Тя се характеризира с наличието на ограничен брой опорни нива по всеки показател, разположени на точно фиксирани места. Присъдената оценка трябва да съвпада с някое от опорните нива. В практиката се приема, че броят на опорните нива по отделните показатели не трябва да надвишава 5 – 7.

В практиката на сензорния анализ се използват различни видове бални скали: 5-бални, 10-бални, 20-бални или 100-бални. Независимо от вида и размаха ѝ, всяка бална скала трябва да е разработена на основата на проста зависимост между качеството на продукта по сензорни показатели и съответстващите му бални оценки (Златева 2013).

Очевидно отсъства актуален терминологичен (или англо-български) речник по информатика. Последният излиза от печат през 1995 г. (Каменова 1995), т.е. две години след появата на интернет. От друга страна, методът на балните скали стихийно се използва в изследванията в областта на информатиката, без обаче да се нарича с коректен термин на български език. Заобикалянето на вербалното обозначение на термина свидетелства за занемаряването на българската терминология и е един от примерите за препятствията пред теоретичното осмисляне на този клон на човешкото знание в българския език.

2.1.2. Приложение на метода на балните скали в CVE

В БДУ CVE *методът на балните скали* се използва за определяне на степента на критичност на уязвимостите. Различават се три групи

метрики за оценка: базова, времева и метрика на обкръжението.

В групата на базовите метрики се оценяват вътрешните характеристики на уязвимостта, които са постоянни във времето и в различните потребителски среди. Тя се състои от две подгрупи: метрики на експлоайта и метрики на въздействието.

Метриките на експлоайта отразяват лекотата и техническите средства, чрез които уязвимостта може да бъде използвана.

Метриките за въздействие отразяват пряката последица от успешна експлоатация и представляват последиците за обекта, който търпи въздействието. Разграничаването между уязвимост и обект на въздействието е ключова характеристика, въведена с CVSS 3.0.

Метричната група Temporal (Време) отразява характеристиките на дадена уязвимост, които могат да се променят с течение на времето, но не и в различните потребителски среди. Например наличието на лесен за използване комплект за експлоатиране би увеличило CVSS оценката, докато създаването на официална кръпка би я намалило.

Метричната група „Околна среда“ представя характеристиките на уязвимостта, които са релевантни и уникални за конкретната потребителска среда. Съображенията включват наличието на механизми за контрол на сигурността, които могат да смекчат някои или всички последици от успешна атака, и относителната важност на уязвимата система в рамките на технологичната инфраструктура (FIRST 2022).

В БДУ уязвимостите са представени чрез вектор, който съдържа само базовите метрики. Векторът има следната структура:

Версия на системата за оценяване: стойност/**кодово обозначение на метриката:** стойност/**кодово обозначение на метриката:** стойност/...

Елементите на вектора са подредени, както следва:

Метрики на експлоайта:

- AV – вектор на атаката;
 - AC – комплексност на атаката;
 - PR – необходими привилегии (права) на атакуващия, преди успешно да експлоатира уязвимостта;
 - UI – взаимодействие с потребител, различен от атакуващия.
- Обхват – S* (има само един елемент).

Метрики на въздействието (impact metrics):

- С – въздействие върху поверителността;
- I – въздействие върху целостта (интегритета);
- А – измерва въздействието върху достъпността на засегнатия компонент, произтичащо от успешно експлоатирана уязвимост (FIRST 2022).

Пример за описание на вектор: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:H

2.1.3. Процедура по класификация на уязвимостите

Класифицирането на уязвимост от CVE преминава през следните етапи:

- Анализатор (експерт) от NVD преглежда всички референтни материали, предоставени със запис на CVE, и присвоява подходящи референтни тагове.

- Уязвимостта се категоризира, като ѝ се присвоява CWE номер.

- Определя се степента на критичност на уязвимостта по скалата CVSS 3.1.

- Уязвимостта се категоризира съгласно CPE.

- Резултатите от анализа се проверяват за гарантиране на качеството от друг, по-старши анализатор, преди да бъдат публикувани на уебсайта и в каналите за данни (NIST 2022).

Използваният алгоритъм за класифициране е добър, но е универсален и невинаги е подходящ при класифициране на уязвимости ОСМУ. Например iOS не е описана в CPE, докато Android е описана.

С това е решена третата научноизследователска задача, поставена в книгата.

2.2. Изследвани бази данни с уязвимости

Дори и беглият поглед върху разнообразието на БДУ за ОСМУ подсказва, че те са десетки или стотици и очевидно се различават по своята достъпност, ефективност и степен на пълнота.

В изследването са включени националните БДУ на държавите с най-голям National Cyber Power Index (NCPI), или, преведено на български – *Национален индекс за киберсила*, за 2020 г., според доклада на Belfer Center for Science and International Affairs Harvard Kennedy School

(Voo et al. 2020). Също така е проучено наличието на БДУ на ниво Европейски съюз и на национално ниво, за България.

2.2.1. Критерии за оценка

Те са свързани с възможностите за търсене на уязвимости за конкретна версия на ОСМУ.

Критерий 1. Възможности на филтъра.

Критерий 2. Релевантност на откритата информация.

Въз основа на резултатите от извършеното проучване по така зададените критерии и ограничения на разкритите информационни ресурси е съставена таблица 1.

Таблица 1. БДУ, подредени по NCPI

1	2	3	4	5	6	7
САЩ	1	NVD	3	CVE	CVSS	182974
КНР	2	CNNVD	1	CNNVD	Собствена и CVSS	194735
КНР	2	CNVD	4	CNVD	Собствена и CVSS	178134
Русия	4	БДУБИ	5	BDU	CVSS	41919
Япония	9	JVN iPedia	2	JVNDB	CVSS	146748
Япония	9	JVN	0	JVN, JVNVU	CVSS	n/a

Забележки:

– Обозначения: 1 – държава; 2 – NCPI; 3 – име на БДУ; 4 – оценка на възможностите за търсене на уязвимости; 5 – конвенция за именуване на уязвимостите; 6 – методика за оценка на уязвимостите; 7 – общ брой записи към датата на проучването.

– Таблицата е съставена по данни, представени по-надолу в изложението.

– В хода на изследването не бяха открити публично достъпни БДУ за ОСМУ на Великобритания (класирана на 3-то място от NCPI),

Нидерландия (5-о място), Франция (6-о място), Германия (7-о място), Канада (8-о място) и Австралия (10-о място). БДУ на Европейския съюз и България не са включени в класацията на NCPI.

2.2.2. Бази данни с уязвимости (БДУ)

Информация относно уязвимостите за ОСМУ може да бъде получена от публично достъпни БДУ, от закрити БДУ и чрез собствени проучвания.

2.2.2.1. База данни на САЩ

Официалното название на базата е National Vulnerability Database (NIST 2022), което може да се преведе като „Национална база данни за уязвимости“. NVD се поддържа от *Националния институт за стандарти и технологии* (NIST) към Министерството на търговията на САЩ и работи в тясна връзка с програмата CVE (Common Vulnerabilities and Exposures). Базата данни CVE е изключена от таблицата с базите данни, тъй като, макар и финансирана от правителството на САЩ, се управлява от корпорация MITRE, а не от държавна структура. Освен това е по-скоро глобална, отколкото национална база данни и има ограничени възможности за търсене по версия на операционна система (MITRE 2022a).

NVD има за задача да анализира всяка уязвимост, след като ѝ бъде присвоен уникален идентификатор (CVE ID), и да я класифицира по следните критерии: CWE, CVSS v2.0, CVSS v3.1 и CPE (NIST 2022).

CVE ID е уникален буквено-цифров идентификатор, зададен от програмата CVE. Всеки идентификатор препраща към конкретна уязвимост. CVE ID позволява на множество страни да обсъждат, споделят и съпоставят информация за конкретна уязвимост, знаейки, че се отнасят за едно и също нещо (MITRE 2022b).

В основата си Common Weakness Enumeration (CWE) е списък с различни видове софтуерни и хардуерни слабости, като в списъка са включени конкретни и кратки дефиниции за всеки често срещан тип слабост (MITRE 2022b).

Общата система за оценяване на уязвимостта (Common Vulnerability Scoring System, CVSS) предоставя начин за представяне на основните характеристики на дадена уязвимост и създаване на числена оценка, отразяваща нейната тежест. След това численият резултат може да бъде

преведен в качествено представяне (като ниско, средно, високо и критично), за да помогне на организациите да оценят правилно и да приоритизират своите процеси за управление на уязвимости (FIRST 2022).

Действащият стандарт за оценка на CVSS е CVSS 3.1. При него се определят видовете оценки на уязвимости (таблица 2).

Таблица 2. Скала за оценка на тежестта на уязвимост CVSS 3.1

Рейтинг (Rating)	CVSS резултат (Score)
Без рейтинг (None)	0.0
Нисък (Low)	0.1 – 3.9
Среден (Medium)	4.0 – 6.9
Висок (High)	7.0 – 8.9
Критичен (Critical)	9.0 – 10.0

Забележка: Таблицата е съставена по цитираната публикация (FIRST 2022).

В практиката все още се използва и CVSS 2.0, с което могат да се обяснят незадоволителната популяризация на новата система от критерии и консерватизмът на част от администраторите, които я използват. От друга страна, ако съдим по прилагането на CVSS 2.0 и CVSS 3.1, то следва да направим заключението, че това са добре работещи инструменти, към които правят референции националните бази и на други държави.

Слабост на NVD е, че се използва само базовата метрика за оценка на уязвимостите. Не по-малко важно е да се отчете времевата метрика, където се отразява сложността за разработването на експлойт.

2.2.2.2. Бази данни на КНР

В Китай се поддържат две национални бази от данни с уязвимости – CNNVD и CNVD.

Първата база данни е Китайската национална база данни за уязвимости на информационната сигурност (China National Vulnerability Database of Information Security – CNNVD). Достъпна е само на китайски език. Анализ на уязвимостите и оценка на риска се прави от Китайския център за оценка на информационната сигурност (China Information Security Evaluation Center – CNITSEC) (CNITSEC 2022). В описанията

на уязвимостите се използват референции към CVE и CVSS.

Скалата за оценка на уязвимостите на CNNVD е подобна на CVSS 3.1 (таблица 3).

Таблица 3. Таблица за оценка на тежестта на уязвимост

Ниво на уязвимост	Оценка
Суперкритично (super critical)	9,0 – 10
Висок риск (high risk)	7,0 – 8,9
Среден риск (medium risk)	4,0 – 6,9
Нисък риск (low risk)	0 – 3,9

Забележки:

– Таблицата е съставена по цитираната публикация (CNNVD 2022).

– Местата на колоните и градацията на оценките са разменени с цел по-добро съпоставяне с БДУ на САЩ.

Втората БДУ е Китайската национална база данни с уязвимости (China National Vulnerability Database – CNVD). CNVD се поддържа от Националния координационен център за работа с компютърни мрежи в извънредни ситуации (CNCERT). В създаването на CNVD участват национални държавни ведомства, важни потребители на информационни системи, оператори, големи доставчици на средства за сигурност, доставчици на софтуерно осигуряване, научноизследователски институции и публични потребители на интернет (CNVD 2022). В CNVD се публикуват както конкретни уязвимости, така и седмични, и месечни бюлетени с обобщена информация. Особеност на китайските бази данни с уязвимости е, че използват цветна кодировка за степените на опасност.

2.2.2.3. База данни на РФ

Официалното название на базата е *Банк данных угроз безопасности информации* (БДУБИ), което може да се преведе като „Банка от данни със заплахи за информационната сигурност“. Поддържа се от *Государственный научно-исследовательский испытательный институт проблем технической защиты информации Федеральной службы по*

техническому и экспортному контролю, или „Държавен научноизследователски изпитателен институт по проблемите на техническата защита на информацията към Федералната служба по технически и експортен контрол“.

БДУБИ съдържа информация за основните заплахи и уязвимости за информационната сигурност, предимно специфични за държавните информационни системи и автоматизираните системи за управление на производствените и технологичните процеси на критични съоръжения (БДУБИ 2022).

На този етап БДУБИ все още прави референции към NVD и в една или друга степен може да се смята за неин аналог.

2.2.2.4. Бази данни с уязвимости на Япония

В Япония се поддържат две БДУ – JVN и JVN iPedia.

Официалното англоезично название на първата БДУ е Japan Vulnerability Notes (JVN)(JVN 2022), което може да се преведе като „Японски записи за уязвимости“. Това е портален сайт за информация за уязвимости, предназначен да помогне за гарантиране на интернет сигурността чрез предоставяне на информация за уязвимости и техните решения за софтуерни продукти, използвани в Япония. JVN се управлява съвместно от Координационния център JPCERT и Агенцията за насърчаване на информационните технологии (IPA).

Официалното англоезично название на втората БДУ е JVN iPedia. По същество JVN iPedia се разглежда като разширение на JVN. В допълнение към информацията за противодействие на уязвимостите, публикувана на JVN, това е база данни с информация за противодействие на уязвимости, която се публикува ежедневно както в Япония, така и в чужбина (JVN iPedia 2022). Особеност на JVN iPedia е, че макар самият сайт да има интерфейс на английски език, само малка част от описанията на уязвимостите са преведени от японски.

Официално и двете японски БДУ са съвместими с глобалната БДУ CVE, препратките към която са разположени на заглавните им страници.

2.2.2.5. Източници, изключени от изследването

В хода на изследването беше установено, че във Великобритания, Нидерландия, Франция, Германия, Канада и Австралия, а така също и в

Европейския съюз, функционираат организации, които предоставят информация за уязвимости на ОСМУ. Но информацията, предоставяна от съответните организации, или не е публично достъпна, или не е реализирана като БДУ. По тази причина тези източници не са предмет на настоящото изследване (CERT-EU 2022; AUSCERT 2022; CCTX 2022)(BSI 2022; CERT-FR 2022; NL NCSC 2022; UK NCSC 2022).

В България функционира Национална лаборатория по компютърна вирусология (НЛКВ) към БАН. Тя не поддържа собствена база от данни за злонамерен софтуер и злонамерени атаки.²

Другата българска организация, свързана с проблемите на киберсигурността, е Националният екип за реагиране при инциденти в компютърната сигурност, или CERT България (CERT Bulgaria 2022). На сайта на CERT България се публикува информация за уязвимости, която е на английски език и няма никаква възможност за търсене и филтриране на информацията. Предоставяната информация се отнася само за 2022 г. и не е реализирана като БДУ. Ето защо и двете български структури са неподходящи източници на информация по приетите ограничения за целите на изследването.

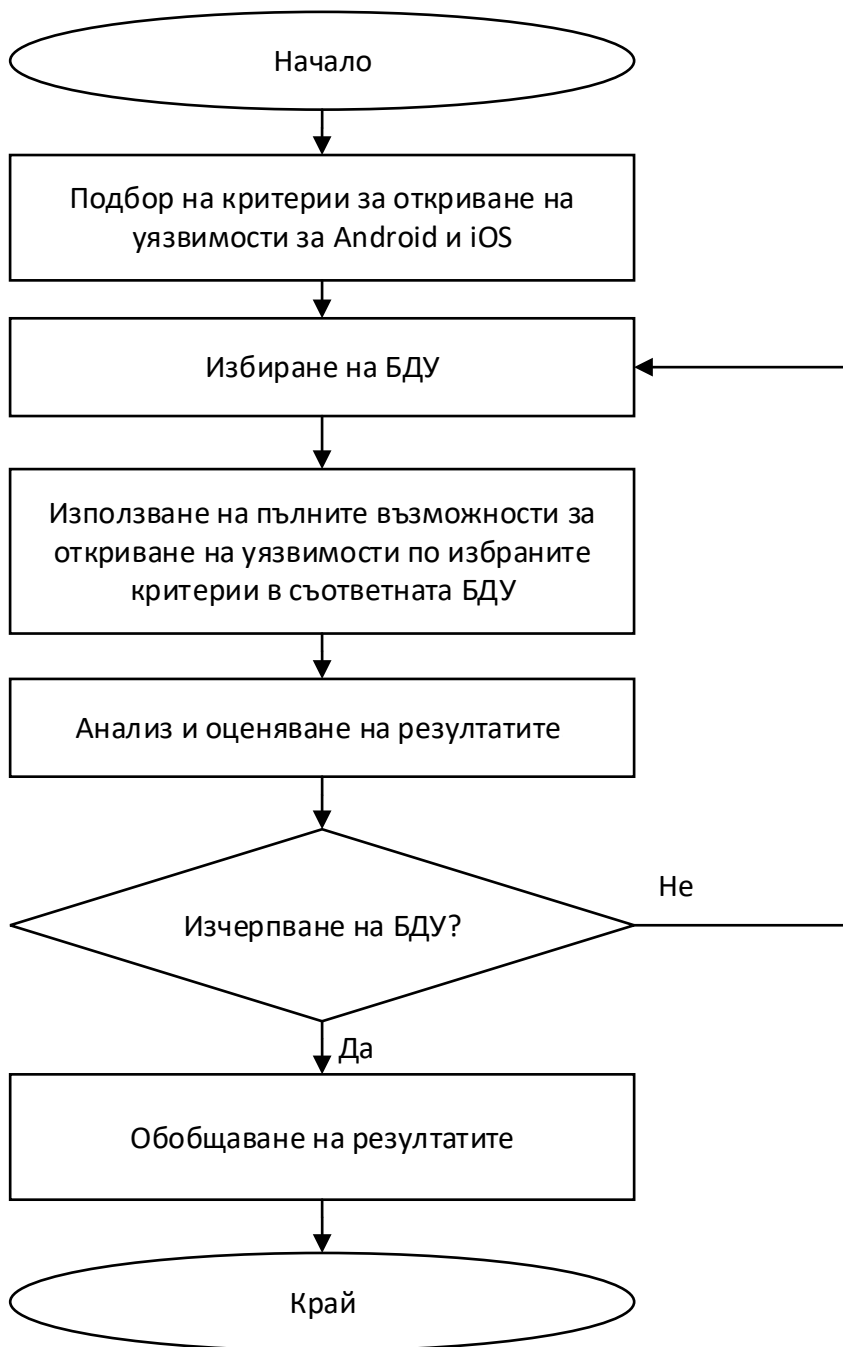
2.2.3. Обсъждане и резултати

Всяка от анализиранияте БДУ има вградени възможности за търсене на уязвимости със задаване на допълнителни условия за филтриране на резултатите. Създава се впечатлението, че откриването на уязвимости за конкретна версия на ОСМУ е лесна за решаване задача. Опитът показва, че това всъщност не е така. Оказва се, че в резултатите от търсенето, освен уязвимости, свързани с конкретна версия на ОСМУ, се появява и информация, която в една или друга степен не е релевантна спрямо зададените условия.

Използвани са вградените възможности за търсене и филтриране на всяка една от анализиранияте БДУ. Направени са опити да се открият уязвимости за конкретни версии на Android и iOS, отговарящи на ограниченията на изследването.

Обследването на всяка БДУ преминава през следните стъпки, отразени на фигура 3.

² POLIMIROVA, D. 2022-07-12. Nalichie na natsionalna baza danni s uyazvimosti. Email [personal communication].



Фигура 3. Блок-схема на експеримента

В резултат на извършеното обследване на информационните ресурси са очертани следните възможности за филтриране на информацията при изследваните БДУ. Те са представени в таблица 4.

Таблица 4. Възможности за филтриране на информацията в БДУ

Възможности на филтъра	Оценка	БДУ с такива възможности
Няма възможност за търсене. Информацията е подредена в раздели, по хронологичен ред на публикациите.	0	JVN
Търсене по идентификатор на уязвимост (CNVD или CVE), ниво на риск, период на публикуване на уязвимостта и период на обновление на информацията за уязвимостта. Няма възможност за търсене по операционна система или производител на хардуер.	1	CNNVD
Търсене по ключова дума, разработчик (производител), продукт, дата на публикуване, дата на последно обновление, CVSSv3, CVSSv2 и CWE.	2	JVN iPedia
Търсене по ключова дума, CVE номер на уязвимост, продукт, дата на публикуване, дата на последно обновление, CVSSv3, CVSSv2, CWE, CPE. CPE по същество позволява задаване на разработчик на продукта, наименование и версия на продукта.	3	NVD
Търсене по ключова дума, CNVD номер на уязвимост, дата на публикуване, наличие на референтна информация, разработчик (производител), продукт, версия на продукта, причина за уязвимостта, заплахи, причинени от уязвимостта, тежест на заплахата, разположение на експлойта спрямо атакуваното устройство.	4	CNVD

<p>Търсене по ключова дума, производител на програмното осигуряване, тип на програмното осигуряване, в което е открита уязвимостта, наименование на програмното осигуряване, апаратна платформа, версия на програмното осигуряване, няколко подверсии на основната версия на ОСМУ, статус на уязвимостта. Допълнителни параметри – времеви период, в който е установена уязвимостта, година на добавяне на уязвимостта, клас на уязвимостта, ниво на опасност, базов CVSS вектор, идентификатор на типа на грешката по CWE, идентификатор на грешката в друга БДУ, наличие на експлоит, способ за експлоатиране на уязвимостта, способ за отстраняване на уязвимостта, операционна система.</p>	5	БДУБИ
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---	-------

Забележки:

- Присвоените стойности за оценка са въз основа на експертния опит на автора.
- Всяка неанглоезична БДУ, при автоматичен превод на английски език, загубва от своята функционалност.
- На JVN се присъжда оценка 0, защото няма възможност за търсене.

Конкретните резултати от обследването очертават следните особености на анализираниите БДУ:

CNNVD предоставя възможности за търсене само по идентификатор на уязвимост (CNNVD или CVE), което не позволява да се търсят група уязвимости, свързани с конкретна версия на ОСМУ.

При **JVN iPedia** и **NVD** се наблюдават фалшиви положителни резултати при търсене по ключова дума (False-positive result). Например тази слабост се проследява при Android 5.0 в случая: <https://web.archive.org/web/20210620220718/https://jvndb.jvn.jp/en/contents/2020/JVNDB-2020-000081.html>

Частен случай на посочения недостатък се установява при **NVD**. При него, при задаване на условието „точно съвпадение“ по отношение на ключовата дума, тя се открива дори когато е част от по-дълъг текстов низ. Пример: при търсене на **iOS 15.0**, се откриват уязвимости, свързани с **iOS 15.0.1**, **iOS 15.0.2**, и т.н.

NVD дава много добри резултати за откриване на конкретна версия за Android с помощта на **CPE**. За съжаление **CPE** не може да се използва при търсене на уязвимости за **iOS**.

Това следва да се счита за значим недостатък. По данни на Apple (APPLE 2023), **iOS 16** се поддържа от 23 различни модела на iPhone, най-старият от които е от 2016 г. (Jones 2023). Това може да доведе до подвеждащи резултати относно уязвимостите за конкретно устройство. Например при търсене по **CPE** за модел на iPhone от 2019 г., ще бъдат открити уязвимости за устройството, а не уязвимости за **iOS 16**, която е реализирана през 2022 г.

Филтърът на **CNVD** дава възможност за избор на разработчик, но при избиране на фирмите, разработващи Android и Apple **iOS**, се наблюдават повторения на стойностите, макар че се избират от падащ списък. Например Google може да се открие три пъти, което подвежда потребителя.

Въпреки че **БДУБИ** има най-малък брой записи (табл. 1), то от изложеното в таблица 4 става ясно, че тя има значителни предимства по отношение на филтрирането пред останалите **БДУ**, с които се извършва сравнението.

Препоръка:

Да се разширят възможностите на **NVD** за филтриране на данни, като се добави възможност за филтриране по операционна система.

Изводи:

След решаването на втора и трета научноизследователска задача, могат да се формулират следните заключения:

– Установяват се две основни тенденции относно предоставянето на информация за уязвимости за **ОСМУ**: от една страна, поддръжка на публична база, а от друга – публично-частни партньорства без предоставяне на публична информация за уязвимостите.

– Търсенето на уязвимости, когато за критерий се използва версията на ОСМУ, в отделни случаи дава нерелевантни резултати. Установяването на тяхната честота може да бъде предмет на отделно изследване.

– Към 2022 г. наложеният стандарт за оценка на тежестта на дадена уязвимост е CVSS (табл. 1).

– Направената оценка на нивото на системност на базите данни с уязвимости с прилагане на *метод на балните скали* дава положителни резултати. Използването на този метод в проучването демонстрира неговата приложимост при оценката на вградените филтри на БДУ за откриване на уязвимости в областта на ОСМУ.

– Нито една от изследваните БДУ не позволява извличане на реални статистически данни за брой на уязвимости за ОСМУ.

– На практика CVE е БДУ, на която се позовават националните БДУ на държавите с най-висок NCPI. Също така, където е възможно, на CVE се позовават и компаниите – производителки на Android и iOS.

– Не всички открити уязвимости са опасни. За уязвимостите, които са много опасни, няма разработени експлойти или е твърде сложно да се разработят.

– По същество по-голяма заплаха представляват уязвимостите, за които има разработени експлойти (Spivakovsky 2022). Затова се препоръчва друг метод за справяне със заплахите за сигурността (Goldstein 2022), при който се използва CSAF.

– Съществува стройна система за оценяване, класифициране и съхраняване на уязвимости за ОСМУ.

В хода на изследването се установи, че в отделните държави съществуват организации на национално ниво, като AUS CERT и US CISA, които издават публично достъпни ежедневни бюлетини по киберсигурност. Бюлетините обединяват различни източници, включително бюлетините на Microsoft, Apple и Google, и съдържат информация за:

- Нови киберинциденти;
- Обновления за отстраняване на уязвимости в приложенията;
- Обновления за отстраняване на уязвимости в ОСМУ;
- Известни експлойти.

Информацията в бюлетините може да е от полза както на обикновените потребители, така и на системните администратори, за своевре-

менно предприемане на превантивни мерки с цел повишаване на сигурността на ОСМУ.

Изследването генерира следната препоръка за повишаване на сигурността:

– Всеки заинтересован следва да се абонира за тези бюлетини и своевременно да инсталира обновленията.

Дадената организационна препоръка е част от комплекса методи за повишаване на сигурността на ОСМУ.

Глава трета

ПОДХОД ЗА ОТКРИВАНЕ НА ПРОПУСКИ В СИГУРНОСТТА НА УЕББРАУЗЪРИ ЗА МОБИЛНИ УСТРОЙСТВА

В съответствие със структурата на книгата, в настоящата глава ще бъде решена четвърта научноизследователска задача.

3.1. Вградени защити за ОСМУ

3.1.1. Вградени защити на iOS

Сигурността на операционните системи от семейството на iOS е разгледана заедно със сигурността на другите операционни системи от екосистемата на Apple в подробен доклад от 2022 г. В доклада са разгледани следните аспекти на сигурността:

- Сигурност на хардуера;
- Системна сигурност;
- Криптиране и защита на данните;
- Сигурност на приложенията;
- Сигурност на услугите;
- Мрежова сигурност;
- Сигурност на средата за разработка;
- Сигурност на устройството като част от информационната инфраструктура на организация (APPLE 2022).

3.1.1.1. Защита от зловредни приложения в iOS

Важен елемент от защитата от зловреден код в приложенията за iOS заема политиката на Apple за подписване на приложенията.

За да разработват и инсталират приложения в iOS или iPadOS устройства, разработчиците трябва да се регистрират в Apple и да се присъединят към програма за разработчици на Apple. Реалната самоличност на всеки разработчик, независимо дали е физическо лице, или организация, се проверява от Apple преди издаването на сертификата му. Този сертификат позволява на разработчиците да подписват приложения и да ги изпращат в App Store за разпространение. В резултат на това всички приложения в App Store се изпращат от лице или организация, което може да бъде идентифицирано. Това служи като възпиращ фактор за създаването на злонамерени приложения.

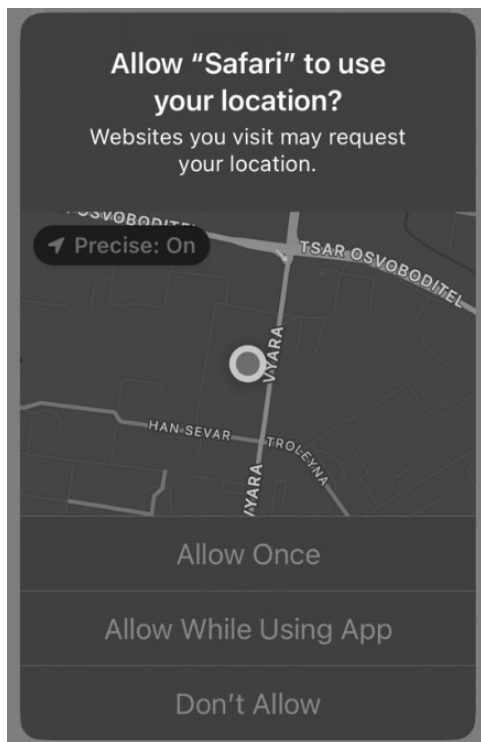
Преди публикуването на приложенията, те също преминават про-

верка от Apple, за да се гарантира, че като цяло работят, както е описано, и не съдържат очевидни грешки или други проблеми. Този процес на куриране дава на потребителите увереност в качеството на приложенията, които купуват.

Проверка на кодovия подпис: iOS и iPadOS позволяват на разработчиците да вграждат рамки в своите приложения, които могат да се използват от самото приложение или чрез разширения, вградени в приложението. За да предпази системата и другите приложения от зареждане на код на трета страна в тяхното адресно пространство, системата извършва валидиране на кодов подпис на всички динамични библиотеки, към които даден процес се свързва по време на стартиране. Тази проверка се извършва чрез идентификатора на екипа (ID на екипа), който се извлича от сертификат, издаден от Apple (APPLE 2022).

3.1.1.2. Разрешения в iOS

Този въпрос в iOS е решен по следния начин: операционните системи на Apple по подразбиране ограничават достъпа до защитени данни и ресурси. На разработчиците на приложения е предоставена възможност да заявят достъп до определени ресурси (APPLE DEVELOPER 2023a). Потребителят решава дали да одобри, или да откаже заявката (фиг. 4).



Фигура 4. Пример за диалогов прозорец за даване на разрешение за достъп. ОС: iOS 17; устройство: iPhone 15; браузър: Safari; източник: авторът.

Заявяването на разрешенията се осъществява във вид на двойки, състоящи се от „ключ“ и „стойност“, които се задават в средата за разработка Xcode (APPLE DEVELOPER 2023b).

3.1.2. Вградени защиты на Android

3.1.2.1. Защита от злонамерени приложения в Android

Вграденият инструмент за проверка на приложенията за Android се нарича Google Play Protect. Проверката се извършва както преди инсталирането на приложенията, така и на вече инсталирани в устройството приложения.

Проверка преди инсталиране

Тя се извършва от автоматизирани алгоритми за откриване на ПВП. Алгоритмите разглеждат стотици сигнали и сравняват поведението в цялата екосистема на Android, за да оценят дали някое приложение показва подозрително поведение. Например:

- Взаимодействие по необичаен начин с други приложения в устройството;
- Достъп или споделяне на лични данни без разрешение;
- Агресивно инсталиране на приложения (включително ПВП);
- Достъп до злонамерени уебсайтове или заобикаляне на вградените функции за сигурност.

Извършват се статичен и динамичен анализ, след което приложението се класифицира. Ако приложението е класифицирано като потенциално вредно, то се препраща за ръчна проверка от специалист анализатор (GOOGLE DEVELOPERS 2023a).

Споразумение за разпространение на Google Play Developer

Това е допълнително ниво за осигуряване. Този договор е ръководство за поведението на разработчиците, които публикуват приложения в Google Play.

Освен това Google Play използва различни методи, за да проверява дали разработчиците спазват тези правила. Вътрешният механизъм за оценка на риска на Google Play анализира информация от акаунта на разработчика в Google, действията, историята, данните за фактуриране, информацията за устройството и др. При възникнало съмнение за нарушение се преглеждат ръчно транзакциите, за да се установи дали разработчикът спазва правилата (GOOGLE DEVELOPERS 2023a).

Проверка на приложения в устройството

Тя се извършва веднъж на ден и се отнася за всички приложения, независимо откъде са инсталирани. Ако бъде открито ПВП, системата предлага на потребителя да го премахне. Приложенията, които са класифицирани като особено опасни, се премахват автоматично. Ежедневното сканиране позволява на Google Play Protect да реагира бързо на открита заплаха, съкращавайки времето, за което потребителите могат да бъдат изложени на заплахата. Едновременно се намалява и количеството на устройствата, които могат да бъдат засегнати. Въпреки че Google Play Protect работи във фонов режим, потребителите могат да проверят кога за последен път устройството им е сканирано, и да прегледат списъка със сканирани приложения в секцията Google Play Protect на своето приложение Google Play (GOOGLE DEVELOPERS 2023b).

Въпреки положените усилия се намират начини за заобикаляне на защитите (Hutchinson et al. 2019; Muhammad et al. 2023).

3.1.2.2. Разрешения в Android

Всяко приложение за Android заявява набор от разрешения, описани във файла `AndroidManifest.xml`. Този файл е задължителен елемент от софтуерния пакет на всяко приложение за Android.

Различават се следните типове разрешения:

– Разрешения по време на инсталиране (Install-time permissions). Те предоставят ограничен достъп до забранени данни или позволяват на приложението да извършва забранени действия, които минимално влияят върху системата или други приложения. Системата автоматично предоставя разрешенията, когато потребителят инсталира приложението (GOOGLE ANDROID 2023).

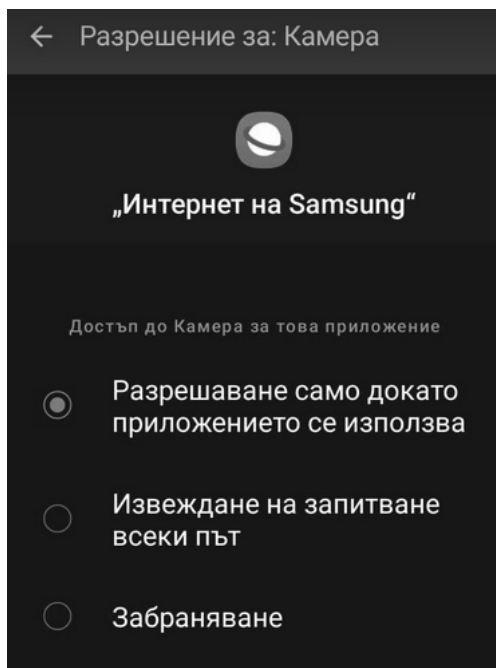
– Нормални разрешения. Те са с нисък риск и се предоставят автоматично на заявяващото приложение. Сами по себе си не представляват истинска заплаха за крайния потребител. Например: `SET_WALLPAPER` (промяна на тапета) (Malik, Khatter 2016a).

– Подписани разрешения. Те се предоставят автоматично на приложенията от същия софтуерен пакет, които декларират разрешението. Приложението, което декларира разрешението, и приложението, което го изисква, трябва да имат еднакъв сертификат или подпис (GOOGLE ANDROID 2023).

– Разрешения, давани по време на изпълнение на приложението (Runtime permissions). Определяни са също като „опасни“. Това са разрешения с висок риск и изискват изрично съгласие от потребителите по време на изпълнението на приложението. Посочените разрешения позволяват достъп до потенциално опасни обръщания към API, като изпращане на SMS, достъп до лични данни на потребителя и взимане на контрол над устройството (фиг. 5).

Например: SEND_SMS (изпращане на SMS), CALL_PHONE (телефониране), BRICK (изтриване на данни) (Malik, Khatter 2016a).

– Специални разрешения (Special permissions). Само разработчиците на ОС и производителите на устройства могат да дефинират специални разрешения (GOOGLE ANDROID 2023).



Фигура 5. Пример за диалогов прозорец за даване на разрешение на достъп. ОС: Android 13; устройство: Samsung A32 5G; браузър: Samsung Internet; източник: авторът.

Ullah и колектив поддържат тезата, че повечето приложения за Android съдържат излишни разрешения за основните функции на приложенията и събират данни от потребителите, което може да предизвика изтичане на личната информация на потребителите и може да им причини вреда. Приложенията, които имат достъп до данните на потребителя и чувствителни разрешения, трябва да включват политика за поверител-

ност в самото приложение и линк в обявата в магазина на приложенията. Това предпазва от възможни заплахи. Независимо от предоставените възможности, основният проблем е, че потребителят не полага усилия да прочете или да вникне в политиката за поверителност на приложението и да разбере целта на такива чувствителни разрешения (Ullah et al. 2022).

В така представените системи за разрешения на приложенията особено място заемат браузърите, тъй като те автоматично получават разрешения за достъп до някои от сензорите на мобилните устройства, без потребителят да е информиран за това.

Изводи и обобщения по раздел 3.1.

Всъщност вградените защити на ОСМУ дават задоволително ниво на сигурност благодарение на системата от разрешения, която използват. Слабата страна са потребителите, които не подхождат достатъчно отговорно при инсталиране на приложения за ОСМУ.

Засега една от слабостите остават браузърите за ОСМУ, чрез които може да бъде получена специална информация, без да се изисква разрешение от потребителя.

3.2. Експериментална част

Съгласно класификацията на MITRE – CAPEC, версия 3.9, съществуват шест области на атака: софтуер, хардуер, комуникации, веригата за доставки (последователността от процеси, участващи в производството и разпространението на атакувания обект), социално инженерство, физическа сигурност (MITRE 2023). При провеждането на планирания експеримент фокусът е само върху потенциално уязвим софтуер и по-точно върху браузъри за мобилни устройства.

Тестовете се извършват на принципа на черната кутия, при който не се изисква проектиране и разработване на изследвания обект (Kong et al. 2019).

Софтуерната поддръжка на мобилен телефон обикновено е с продължителност от две до три години. След това производителят не публикува обновления за операционната система. Такава е ситуацията с различните марки смартфони, работещи под Android. При iPhone този период е по-дълъг. Например може на сравнително стар хардуер да се

инсталира нова версия на операционната система. В хода на изследването е констатирано, че на iPhone X (пуснат на пазара през ноември 2017 г.) може да се инсталира iOS 16 (официално внедрена през юни 2022 г.).

Допълнително объркване внасят и различните реализации на съответния приложен програмен интерфейс към ОСМУ.

Ето защо голямото разнообразие от комбинации на софтуер и хардуер предполага, че изпитанията с реални устройства са за предпочитане пред изпитанията с виртуални устройства.

Съгласно препоръките на World Wide Web Consortium (W3C), съвременните мобилни браузъри трябва да поддържат интерфейси за следните сензори: API за универсален сензор, акселерометър, сензор за геолокация, сензор за близост, жirosкоп, сензор за ориентация и сензор за осветеност (W3C 2023).

Предназначенията на приложния програмен интерфейс (API) за универсален сензор са, както следва:

- Да насърчи съгласуваността на приложните програмни интерфейси за различни сензори.

- Да даде възможност за усъвършенствани случаи на употреба благодарение на ефикасни ППИ на ниско ниво.

- Да увеличи темпото, с което нови сензори могат да бъдат въведени в употреба, като опрости процесите на спецификация и реализация (Waldron 2023).

С помощта на интерфейса за акселерометъра може да се получат данни за ускорението, приложено към осите X, Y и Z на устройството в локална координатна система, дефинирана от самото устройство (Kostiainen 2023a).

ППИ Geolocation Sensor предоставя информация за географското местоположение на хостащото устройство (Kostiainen et al. 2023).

Интерфейсът Magnetometer предоставя информация за магнитното поле, засечено от основния магнитометричен сензор на устройството. Магнитометричният сензор измерва магнитното поле за трите физически оси (x, y, z), в μT (микротесла) (Kostiainen, Bhaumik 2023a).

Предназначението на сензора за близост е да предоставя информация за нивото на близост. Нивото на близост се отчита като разстоянието (в сантиметри) от сензора до най-близката видима повърхност. Точната

стойност на разстоянието, отчетена от различните устройства, може да се различава поради разлики в метода на откриване, конструкцията на сензора и др. Освен това някои сензори за близост може да са в състояние да предоставят само булева стойност, за да покажат дали има физически обект, който е близо, по-скоро като откриване на присъствие, отколкото абсолютна стойност за разстоянието (Kostiainen, Bhaumik 2023b).

С помощта на ППИ за жирокоп се предоставя информация за ъгловата скорост около локалните оси X, Y и Z на устройството, в единици радиан за секунда (Kostiainen 2023b).

Интерфейсът Orientation Sensor предоставя обща информация, описваща физическата ориентация на устройството спрямо триизмерна декартова координатна система. Този ППИ дава възможност да се получат координати както спрямо референтната координатна система на Земята (клас Absolute Orientation Sensor), така и спрямо други ориентири, като истинския север (Christiansen, Kostiainen 2023).

Чрез Ambient Light Sensor интерфейса се получават данни за нивата на околна светлина, открити от основния светлинен детектор на устройството (Kostiainen, Bhaumik 2023c).

3.2.1. Постановка на експеримента

За провеждането на експеримента е създаден уебсайт, достъпен на адрес: <https://www.learning-is.fun/mdsati> (вж. фиг. 6).

На сайта е присвоено работно название: Mobile Device Sensors Access Testing via Internet (MDSATI). Преведено на български език: „Тестване на достъпа до сензорите на мобилно устройство през интернет“. При разработването на сайта са използвани скриптовите езици JavaScript (JS) и PHP.

Предимства на тестването през уебсайт:

- Провеждането на теста не изисква инсталиране на специализиран софтуер и специфични умения за работа с него.
- Провеждането на теста не зависи от вида или версията на изследваната операционна система.
- Резултатите от конкретен тест се получават за по-малко от една минута.

Mobile Device Sensor Access Testing via Internet

Този уеб сайт не обработва и не съхранява специални данни по смисъла на GDPR.

Няколко секунди след натискането на бутона "Започни теста" ще получите кратък отчет за сензорите, до които има достъп браузъра, който използвате.

This web site does not process or store special data within the meaning of the GDPR.

A few seconds after pressing the "Start test" button, you will receive a short report about the sensors that the browser you are using has access to.

Започни теста
Start test

Фигура 6. Изглед от началната страница на уебсайта, от който се провежда експериментът

Началната страница на сайта (index.php) съдържа PHP скрипт, който предотвратява многократното изпълнение на един и същи тест. За целта се стартира PHP сесия с продължителност 24 часа и се проверява за наличието на файл с резултати. Ако съществува файл с резултати, останалата част от страницата се модифицира, така че да съдържа съобщение, че тестът вече е изпълнен, и активна хипервръзка към протокола от теста. Бутонът за стартиране на теста е деактивиран.

Ако не съществува файл с резултати, се извиква PHP скрипт createReportFiles.php, който създава файл за резултатите и файл за протокола от теста. Файлът за протокол е необходим, за да представи резултатите от теста във вид, достъпен за масовия потребител.

Бутонът за изпълнение на теста е активен, а хипервръзката към протокола е скрита.

След натискането на бутона „Започни теста“, се изпълнява функцията `init()`.

Предвидена е възможност за автоматично стартиране на функцията

init(), в случай че е необходимо последователно изпълнение на няколко теста чрез скрипт.

Например:

<https://www.learning-is.fun/mdsati/index.php?testMode=auto>

Функцията init() извиква последователно функциите doTest(), testDone() и generateHTMLReport(). След това визуализира хипервръзката за изтегляне на отчета.

Функцията doTest() осигурява извикването на процедури, всяка от които се опитва да осъществи достъп до конкретен сензор на мобилното устройство. Преди извикването на всяка процедура се извежда кратко съобщение за съответния опит. Първо се прави опит за задействане на вибрацията – функция probeVibrate(). След това се прави опит за достъп до следните сензори:

- Сензор за ориентация в пространството – функция getOrientation().
- Сензор за ускорение – функция getAcceleration().
- Сензор за честота на завъртане – функция getRotationRate().
- Сензор за близост до обект – функция getProximity().
- Сензор за ниво на осветеност на околната среда – функция getLight().
- Магнитен сензор. Този сензор има две приложения – измерване на нивото на магнитното поле (функция getMagnetometer()) и компас (функция getCompass()).

- Сензор за батерията – функция getBattery().
- Сензор за географско позициониране – функция getGeolocation().
- Камера – функция getPic().
- Микрофон – функция getAudio().

По време на теста се извеждат кратки информативни съобщения за изпълнение на опит за достъп до всеки от сензорите.

Всяка от извиканите процедури добавя резултата от опита в общ файл с помощта на функцията addToReport(). Функцията addToReport() приема като параметър JSON обект, с общ вид:

{тип:стойност, тип:стойност, тип:стойност}

Първият елемент е типът на теста. Вторият елемент съдържа данни от съответния сензор или съобщение за грешка, ако опитът за достъп до сензора е неуспешен. Третият елемент съдържа логическа стойност

„истина“ или „лъжа“, в зависимост от това, дали опитът е бил успешен, или е завършил с грешка.

Пример за запис след успешен опит:

```
{„type“:“geolocation“,„data“:“Вашите координати са: Ширина 43.2114197 Дължина 27.9322682“,„logical“:true}
```

Пример за запис след неуспешен опит:

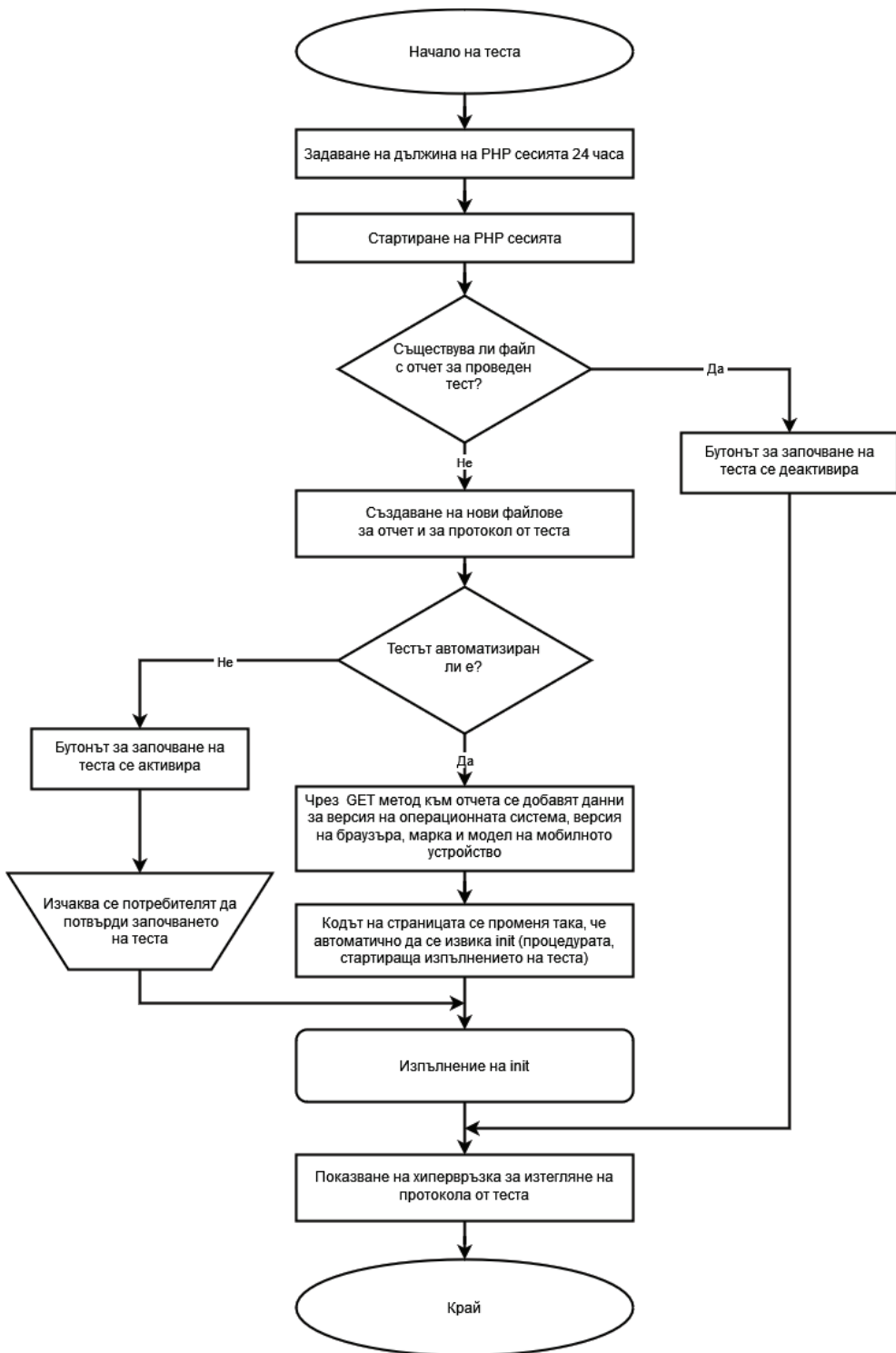
```
{„type“:“light“,„data“:“Error: Timeout: Няма данни за осветеност“,„logical“:false}
```

Генерирането на протокола става с изпълнението на JS процедурата `generateHTMLReport()`.

В протокола се добавя подходящо съобщение, в зависимост от резултата от проверката за достъп до сензорите (прил. 3 и прил. 4).

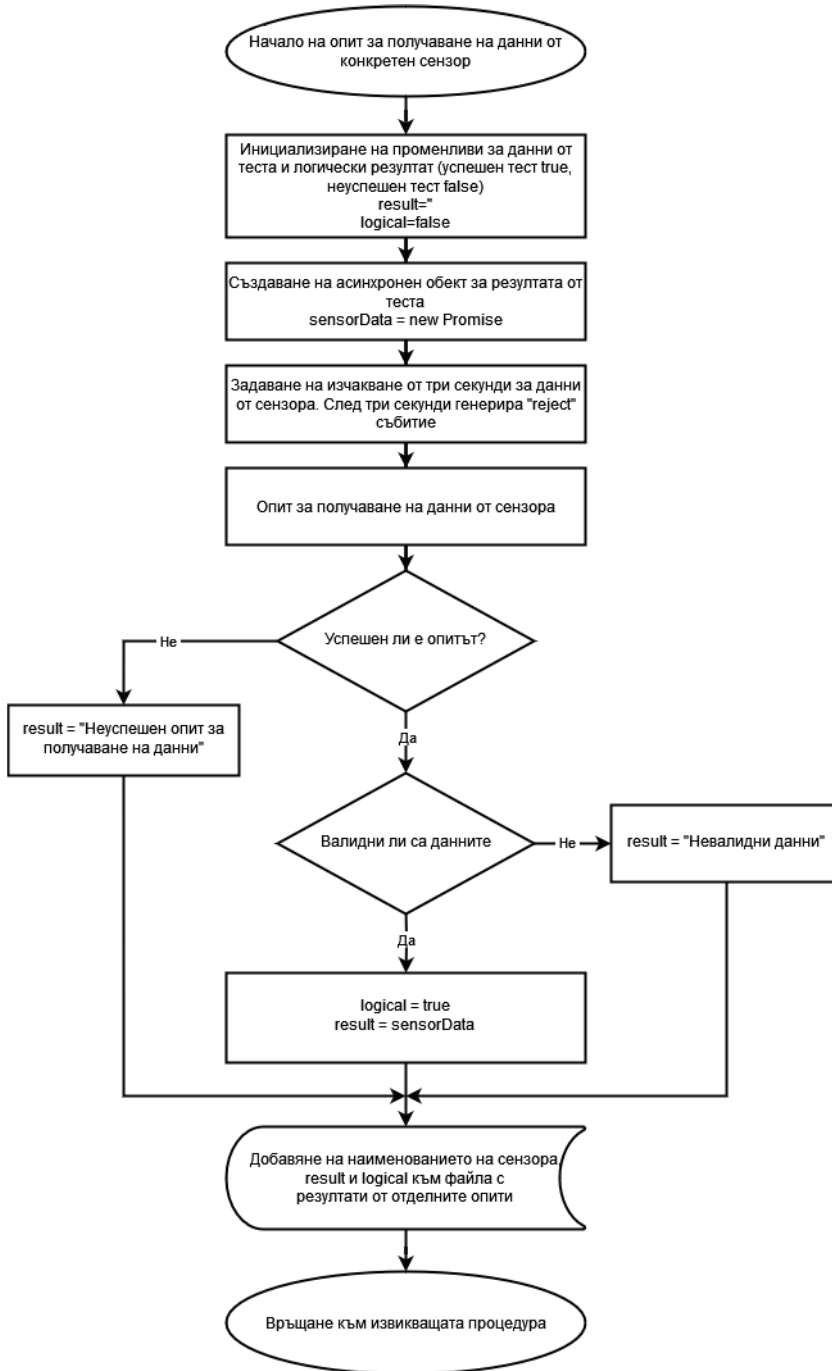
Работата с файлове от страна на сървъра се осъществява с помощта на PHP скриптове, които имат имена, аналогични на процедурите на JavaScript, които ги извикват.

Алгоритъмът на провеждане на експеримента е отразен в блок-схема (фиг. 7).



Фигура 7. Блок-схема на експеримента

Алгоритъмът на опит за получаване на данни от конкретен сензор е отразен в блок-схема (фиг. 8).



Фигура 8. Блок-схема на алгоритъм за опит за получаване на данни от конкретен сензор

Възниква законният въпрос за поверителността на резултатите.

По време на теста (ако потребителят даде разрешение) се прави снимка с камерата на устройството и се записва звук. Така създадените файлове не се съхраняват на сървъра и са достъпни само докато страницата е отворена.

Имената на файловете за отделните протоколи се генерират автоматично. Следователно съществува възможност за опит за налучкване на името на файла с протокол, генериран за друг потребител. За да се избегне тази ситуация, са предприети следните мерки:

- За всеки потребител се създава РНР сесия. След това към името на файла, съдържащ отчета, се добавя уникалният идентификатор на РНР сесията. Така името на файла не може да бъде налучкано.

- Допълнително са зададени права за достъп до директорията, в която се съхраняват протоколите, така че съдържанието ѝ да не може да бъде разгледано.

3.2.2. Резултати от провеждането на експеримента

По време на опитите се оказва, че при опит за задействане на вибрацията, FireFox не дава съобщение за грешка. В същото време устройството не вибрира. Получаването на реална информация относно успешността на опита предполага анкетиране на потребителя. С цел да се облекчат потребителите при провеждане на експеримента, е взето решение данните от този сензор да не се подлагат на анализ.

Също така няма да бъдат анализирани данните за достъп до геолокация, камера, микрофон, тъй като при опит за достъп до тези сензори потребителят получава винаги предупреждение (освен ако не е дал дългосрочно разрешение). Следователно може да се приеме, че въпросът относно сигурността на тези сензори е решен. Включването на тези сензори в експеримента цели да насочи вниманието на потребителите към това, какви разрешения дават, и да сравнят с това, че за другите данни не се дават разрешения.

Както се вижда от блок-схемата на алгоритъма за достъп до конкретен сензор (фиг. 8) – има логическа променлива, именувана *logical*. Възможните стойности на променливата са: състояние „Истина“ (отговаря на цифрата едно) или състояние „Лъжа“ (отговаря на цифрата нула). В

началото на всеки от опитите за достъп до конкретен сензор променливата *logical* се инициализира със стойност „Лъжа“. Ако сензорът върне съобщение за грешка, опитът за достъп се счита за неуспешен и *logical* остава без промяна. Ако сензорът върне данни, опитът се счита за успешен и стойността на *logical* се променя на „Истина“. По време на експеримента е установено, че някои браузъри връщат стойност NULL като данни от сензор. В резултат на това променливата *logical* получава стойност „Истина“, без да са получени реални данни от сензора. Такъв резултат е фалшиво-положителен (табл. 5). Това налага ръчна обработка на данните за получаване на реални резултати (табл. 6).

Таблица 5. Възможни резултати при опит за получаване на достъп до сензор

	Отговор от сензор		
	Получено съобщение за грешка при опит за достъп до сензор	Получени реални данни от сензор	Получени NULL данни от сензор
Резултат	Отрицателен	Положителен	Фалшиво-положителен

Таблица 6. Примери за фалшиво-положителни резултати

Прот. №	OS	Browser	Device	Orientation	Acceleration
1	Android 14	Chrome 117.0.5938.61	Google Pixel 8	1	1
				alpha: undefined, ...	accelerationX: 0, ...
2	Android 13	Firefox 119.1.1	Samsung Galaxy A32 5G	1	1
				alpha: 13.504 ...	accelerationX: null, ...

Забележка: Таблицата е съставена по данни от описания експеримент.

И в двата случая очевидно това е въпрос на програмна реализация от страна на разработчиците на браузъра.

Опитни данни

Опитните данни са извлечени от прил. 1. Данните са получени след проведени експерименти с реални мобилни устройства, тествани с програмния код, представен в прил. 2.

Брой изследвани устройства: 32.

Брой изследвани версии на операционни системи: 13.

Брой изследвани сензори: 8.

Брой изследвани браузъри: 5 (вж. прил. 1).

Обобщените данни от опитите за достъп до сензорите на мобилните устройства са представени в таблица 7.

Таблица 7. Обобщени данни по изследваните сензори

	О	А	R	Р	L	М	С	В
Брой успешни опити	3	39	37	0	0	0	35	34
Брой неуспешни	60	57	57	97	97	97	62	63
Брой ФП опити	34	1	3	0	0	0	0	0
% на успеваемост	3.09	40.21	38.14	0.00	0.00	0.00	36.08	35.05

Забележки:

1. Таблицата е обобщение на прил. 1.

2. Използвани съкращения:

O – Orientation;

A – Acceleration;

R – RotationRate;

P – Proximity;

L – Light;

M – Magnetometer;

C – Compass;

B – Battery.

3.2.3. Анализ на получените резултати

Показателно е, че всички опити за достъп до сензорите Proximity, Light, Magnetometer са неуспешни. Това свидетелства за стремежа на разработчиците на софтуер да повишат сигурността на браузърите. Например спирането на поддръжката на Ambient Light Sensor (в експеримента Light) предпазва от атаката, демонстрирана от Olejnik през 2017 г. (Olejnik 2017).

Наблюдава се еднотипно поведение на браузърите, независимо от вида на устройството и версията на ОСМУ (вж. прил. 1).

Chrome и Edge, работещи под Android, почти винаги имат достъп до четири от осемте изследвани сензора.

Firefox, работещ под Android, само веднъж има достъп до един сензор.

Samsung Internet, работещ под Android, винаги има достъп до три сензора, а в някои случаи и до четири.

Посочените по-горе резултати са обобщени в таблица 8.

Таблица 8. Обобщени резултати по браузъри за Android

Браузър	Chrome	При 17 от 18 опита има достъп до четири сензора. При 1 от 18 опита има достъп до два сензора.
	Firefox	При 1 от 13 опита има достъп до един сензор. При 12 от 13 опита няма достъп до нито един от сензорите.
	Edge	При 11 от 12 опита има достъп до четири сензора. При 1 от 12 опита има достъп до два сензора. При 1 от 13 опита няма достъп.
	Samsung Internet	При 4 от 7 опита има достъп до четири сензора. При 3 от 7 опита има достъп до три сензора.

Забележка: Таблицата е съставена по данни от прил. 1.

От една страна, може да се създаде впечатлението, че софтуерът не работи правилно под iOS, тъй като почти всички опити за достъп до сензорите са неуспешни. От друга страна, при iPhone XS с iOS 12, Chrome и Safari имат достъп до три от сензорите. Едновременно с това, при същото устройство, но с iOS 13 и iOS 14 и двата браузъра нямат достъп до сензорите (табл. 9).

Таблица 9. Обобщени резултати по браузъри за iOS

Браузър	Chrome	При 1 от 21 опита има достъп до три сензора. При 20 от 21 опита няма достъп до нито един сензор.
	Safari	При 1 от 25 опита има достъп до три сензора. При 24 от 25 опита няма достъп до нито един сензор.

Забележка: Таблицата е съставена по данни от прил. 1.

Ползи:

Повишаване на осъзнатостта на потребителите относно рисковете за сигурността, които има в браузърите.

Ще се определи кой от браузърите е по-устойчив на атаки.

Въпреки че поддръжката на интерфейса за следене на нивото на батерията е прекратена още през 2017 г. (Olejnik et al. 2017), експерименталните резултати показват, че този интерфейс се поддържа от Google Chrome (протокол 1 от прил. 1).

Изводи и обобщения по раздел 3.2.

Получените опитни данни за изследваната група уязвимости показват, че част от тях не са отстранени.

Разработеният софтуер MDSATI може да послужи за повишаване на осъзнатостта на потребителите за рисковете за сигурността по отношение на разрешенията, които дават, и по отношение на сайтовете, които посещават.

Следва да се отчете, че при опита за достъп до сензора за ориентация има значително количество фалшиво-положителни резултати. На даденото ниво на разработка на MDSATI не е предвидено автоматично

отчитане на фалшиво-положителните резултати при опит за достъп до сензор. Това налага допълнителна ръчна обработка на опитните данни за получаване на достоверни резултати. За в бъдеще следва да се подобри алгоритъмът по такъв начин, че този пропуск да бъде отстранен. Друга посока на разширение на функционалността на софтуера е да се добави към протокола по-подробна информация относно това, кой сензор на каква атака може да бъде подложен.

Става ясно, че има сензори, за които потребителят е задължително известен, когато има опит за достъп до тях. Такива са камерата, микрофонът и геолокацията. С оглед на описаните уязвимости за останалите сензори, като сензорите за ускорение, ориентация, фоновото осветление, компас (Diamantaris et al. 2020), е логично да се препоръча този подход за известяване на потребителя. Така направената препоръка може и трябва да се приеме като част от методите за повишаване на сигурността на ОСМУ.

Изводи и заключения по глава трета

Проверката на откритата още през 2016 г. група от уязвимости показва, че част от тях все още не са отстранени, въпреки че са минали повече от шест години. В областта на ИТ това е парадоксален срок.

Ето защо могат да бъдат направени следните препоръки за повишаване на сигурността:

- Да се инсталират приложения само от официалните marketplaces.
- Винаги да се използват ОСМУ, на които са инсталирани последните обновления от разработчика.

Посочените препоръки не са новост, но всъщност в практиката само частично се изпълняват, поради което е целесъобразно да бъдат повторени.

По този начин се осигурява прилагането на вградените мерки за сигурност на ОСМУ, което кореспондира с темата на книгата.

Изводи и заключение

С оглед на издигнатата в началото на книгата хипотеза се установява, че всъщност ОСМУ са сигурни. За това допринасят:

– Системата от разрешения, която се използва при инсталирането на софтуера.

– Google Play Protect – система на Google за анализ на софтуера, преди да бъде пуснат на пазара.

– Политиката на Apple за анализ на софтуера, преди да бъде пуснат на пазара.

Слабата страна са потребителите, които инсталират софтуер, без да обръщат достатъчно внимание на предупрежденията за исканите разрешения по време на инсталиране и използване на приложенията за мобилни устройства. Друга слабост продължават да бъдат браузърите за ОСМУ, чрез които може да бъде получена специална информация, без да се изисква разрешение от потребителя.

С това частично се доказва издигнатата хипотеза.

Отсъствието на категоричност при защитата на издигнатата теза се обуславя от неразрешения проблем с уязвимостта на двете ОСМУ към *side channel* атаки. Показателни примери за такива атаки са:

Достъпът до жирокопа за устройства, работещи под Android. Браузърите Chrome, Edge и Samsung Internet имат достъп до този сензор, а той подлежи на атака, описана още през 2014 г. (Michalevsky et al. 2014) и преповторена като уязвимост в обзорна статия от 2020 г. (Diamantaridis et al. 2020).

Атаката Juice filming charging (Meng et al. 2015; 2016), приложима и за двете ОСМУ.

Направен е литературен обзор, с поставяне на високи критерии за надеждност на сведенията в научните публикации, който позволи да се обосноват коректно научноизследователските задачи във втора и трета глава.

Група от уязвимости, открити и публикувани през 2016 г., са подложени на експериментално обследване със специално разработен в хода на изследването софтуер. Експериментът потвърди, че към края на проучването част от уязвимостите не са отстранени, въпреки тяхната важ-

ност за сигурността на ОСМУ.

Разработеният за експеримента софтуер е на работещо ниво и с перспективи за по-нататъшно усъвършенстване.

В книгата се установява, че методите за повишаване на сигурността на ОСМУ могат да се разделят на следните групи:

1. Диагностика на уязвимости.

Установяването на уязвимостите може да се извършва или ръчно (например анализ на журнални файлове), или автоматизирано. В диагностиката широко приложение намират алгоритмите за *машинно обучение*, които се използват за класифициране на уязвимости, изброени в първа глава.

2. Отстраняване на уязвимости.

Действията по отстраняване на уязвимости могат да бъдат:

- Инсталиране на обновления на ОСМУ.
- Следене на бюлетини за уязвимости и своевременно прилагане на мерки за намаляване на риска.
- Работа с човешкия фактор. Тя включва както просвещаването на потребителите с оглед на повишаването на осъзнатостта им, така и взаимодействието с разработчиците на ОСМУ и приложения за прилагане на добри практики за повишаване на сигурността при разработка на софтуер.

Тези мерки могат да се прилагат както постфактум, така и профилактично, преди да са нанесени щети върху ОСМУ.

Така изброените по-горе действия формират комплекса от методи за повишаване на сигурността на ОСМУ.

В книгата се разкриват възможности за бъдещо развитие и изследване на сигурността на мобилни устройства, които се надяваме да допринесат за по-безопасно и по-надеждно използване на тези технологии.

Научно-приложни приноси

1. Направени са специализиран, тематично ориентиран литературен обзор и класификационен анализ на съвременни литературни източници, отразяващи резултати от актуални изследвания на уязвимостите и методите за повишаване на сигурността на мобилните устройства с операционни системи iOS и Android.

2. Дефинирани са актуалните методи за повишаване на сигурността на операционните системи iOS и Android на мобилни устройства.

3. Приложен е методът на балните скали при оценката на вградените филтри на БДУ за откриване на уязвимости в областта на ОСМУ. Въведен е терминът „метод на балните скали“ (Scoring System) и е предложено той да бъде въведен в българския терминологичен речник по информатика.

4. Разработен е оригинален подход за тестване на достъпа до сензорите на мобилно устройство през Internet – Mobile Device Sensors Access Testing via Internet (MDSATI).

Приложни приноси

1. Експериментално е установено, че някои от уязвимостите за мобилни браузъри, открити през 2016 г., все още не са отстранени, въпреки че са минали повече от шест години.

2. Формулирани са препоръки за повишаване на сигурността на операционни системи за мобилни устройства.

3. Разработен е алгоритъм за експериментално изследване на сигурността на операционните системи на мобилни устройства.

4. Разработен е алгоритъм за получаване и обработка на данни от сензор на мобилно устройство.

5. Разработен е програмен код за тестване на уязвимостите на операционните системи на мобилни устройства.

ИЗПОЛЗВАНА ЛИТЕРАТУРА

ALLIX, Kevin et al., 2016. Empirical assessment of machine learning-based malware detectors for Android: Measuring the gap between in-the-lab and in-the-wild validation scenarios. *Empirical Software Engineering*, vol. 21, no. 1, pp. 183–211. [Online] Available from: <https://doi.org/10.1007/S10664-014-9352-6>.

ALOTHALI, Eiman et al., 2015. Identification and analysis of free games' permissions in Google Play. In: *2015 – 6th International Conference on Information and Communication Systems (ICICS 2015)*. Institute of Electrical and Electronics Engineers Inc. 6 May, pp. 83–88. ISBN 9781479973491. [Online] Available from: <https://doi.org/10.1109/IACS.2015.7103207>.

ANANYA, A. et al., 2020. SysDroid: a dynamic ML-based android malware analyzer using system call traces. *Cluster Computing*, vol. 23, no. 4, pp. 2789–2808. [Online] [Accessed 18 July 2023]. Available from: <https://doi.org/10.1007/s10586-019-03045-6>.

ANDRIATSIMANDEFITRA, Radoniaina; TONG, Valerie Viet Triem, 2016. Detection and Identification of Android Malware Based on Information Flow Monitoring. In: *Proceedings – 2015 IEEE – 2nd International Conference on Cyber Security and Cloud Computing (CSCloud)*. Institute of Electrical and Electronics Engineers Inc. 4 January, pp. 200–203. ISBN 9781467392990. [Online] Available from: <https://doi.org/10.1109/CSCLOUD.2015.27>.

ANDRIOTIS, Panagiotis et al., 2016. A study on usability and security features of the Android pattern lock screen. *Information and Computer Security*, vol. 24, no. 1, pp. 53–72. [Online] Available from: <https://doi.org/10.1108/ICS-01-2015-0001>.

APPLE DEVELOPER, 2023a. Requesting access to protected resources. Web site. *Wayback Machine. Internet archive*. 18 November. [Accessed 18 November 2023]. Available from: http://web.archive.org/web/20231118174630/http://web.archive.org/screenshot/https://developer.apple.com/documentation/uikit/protecting_the_user_s_privacy/requesting_access_to_protected_resources/.

APPLE DEVELOPER, 2023b. Entitlements. Web site. *Wayback Machine. Internet archive*. 19 November. [Accessed 19 November 2023]. Available from: <http://web.archive.org/web/20231119090807/http://web.archive.org/screenshot/https://developer.apple.com/documentation/bundleresources/entitlements>.

APPLE, 2022. Apple Platform Security. May, pp. 1–219. Web site. *Wayback*

Machine. Internet archive. [Accessed 14 November 2023]. Available from: http://web.archive.org/web/20231105161629/https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf.

APPLE, 2023. iPhone models compatible with iOS 16 – Apple Support. Web site. *Wayback Machine. Internet archive.* 30 January. [Accessed 11 February 2023]. Available from: <https://web.archive.org/web/20230130120403/https://support.apple.com/guide/iphone/supported-models-iph3fa5df43/ios>.

AUSCERT, 2022. AusCERT: Australia’s LEADING cyber emergency response team. Safeguard your information. Web site. *Wayback Machine. Internet archive.* 19 September. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220919000852/https://auscert.org.au/>.

AYSAN, Ahmet Ilhan; SEN, Sevil, 2016. “Do You Want to Install an Update of This Application?” A Rigorous Analysis of Updated Android Applications. In: *Proceedings – 2015 IEEE – 2nd International Conference on Cyber Security and Cloud Computing (CSCloud)*. Institute of Electrical and Electronics Engineers Inc. 4 January, pp. 181–186. ISBN 9781467392990. [Online] Available from: <https://doi.org/10.1109/CSCLOUD.2015.97>.

BERNARDI, Mario Luca et al., 2019. Dynamic malware detection and phylogeny analysis using process mining. *International Journal of Information Security*, vol. 18, no. 3, pp. 257–284. [Online] Available from: <https://doi.org/10.1007/S10207-018-0415-3>.

BEZOBRAZOV, Sergei et al., 2016. The methods of artificial intelligence for malicious applications detection in android OS. *International Journal of Computing*, vol. 15, no. 3, pp. 184–190. [Online] Available from: <https://doi.org/10.47839/ijc.15.3.851>.

BHATT, Arpita Jadhav et al., 2021. iABC-AL: Active learning-based privacy leaks threat detection for iOS applications. *Journal of King Saud University – Computer and Information Sciences*, vol. 33, no. 7, pp. 769–786. [Online] Available from: <https://doi.org/10.1016/J.JKSUCI.2018.05.008>.

BSI, 2022. BSI – UP KRITIS. Web site. *Wayback Machine. Internet archive: Bundesamt für Sicherheit in der Informationstechnik (BSI)*. 11 August. [Accessed 10 November 2023]. Available from: https://web.archive.org/web/20220811085322/https://www.bsi.bund.de/DE/Themen/KRITIS-und-regulierte-Unternehmen/Kritische-Infrastrukturen/UP-KRITIS/up-kritis_node.html.

BUHOV, Damjan et al., 2015. Network security challenges in android applications. In: *Proceedings – 2015 – 10th International Conference on Availability, Reliability and Security (ARES)*. Institute of Electrical and Electronics Engineers Inc. 16 October, pp. 327–332. ISBN 9781467365901. [Online] Available from: <https://doi.org/10.1109/ARES.2015.59>.

CANFORA, Gerardo et al., 2015. Composition-malware: Building android malware at run time. In: *Proceedings – 2015 – 10th International Conference on Availability, Reliability and Security (ARES)*. Institute of Electrical and Electronics Engineers Inc. 16 October, pp. 318–326. ISBN 9781467365901. [Online] Available from: <https://doi.org/10.1109/ARES.2015.64>.

CASOLARE, Rosangela et al., 2021. Dynamic mobile malware detection through system call-based image representation. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 12, no. 1, pp. 44–63. [Online] [Accessed 28 July 2023]. Available from: <https://doi.org/10.22667/JOWUA.2021.03.31.044>.

CAVIGLIONE, Luca et al., 2016. Seeing the unseen: Revealing mobile malware hidden communications via energy consumption and artificial intelligence. *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 799–810. [Online] Available from: <https://doi.org/10.1109/TIFS.2015.2510825>.

CCTX, 2022. ABOUT CCTX – Canadian Cyber Threat Exchange (CCTX). Informing Canadian Business: Canadian Cyber Threat Exchange (CCTX). Web site. *Wayback Machine. Internet archive*. 27 March. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220327174736/https://cctx.ca/about-cctx/>.

CERT BULGARIA, 2022. Национален екип за реагиране при инциденти с компютърната сигурност: Добре дошли в CERT Bulgaria! Web site. *Wayback Machine. Internet archive*. 26 July. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220726101323/https://www.govcert.bg/BG/Pages/default.aspx>.

CERT-EU, 2022. CERT-EU – Publications. Web site. *Wayback Machine. Internet archive*. 15 August. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220815090149/https://cert.europa.eu/publications>.

CERT-FR, 2022. CERT-FR – Centre gouvernemental de veille, d’alerte et de réponse aux attaques informatiques. Web site. *Wayback Machine. Internet archive*.

2 October. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20221002202558/https://www.cert.ssi.gouv.fr/>.

CHEN, Guanlin; ZHANG, Lidong, 2016. A Network Security Scanning System for Mobile Internet Based on Android. *International Journal of Security and Its Applications*, vol. 10, no. 11, pp. 99–108. [Online] Available from: <https://doi.org/10.14257/ijisia.2016.10.11.09>.

CHIMUCO, Francisco T. et al., 2023. Secure cloud-based mobile apps: attack taxonomy, requirements, mechanisms, tests and automation. *International Journal of Information Security*, vol. 22, no. 1, 17 February, pp. 833–867. [Online] Available from: <https://doi.org/10.1007/S10207-023-00669-Z>.

CHRISTIANSEN, Kenneth Rohde; KOSTIAINEN, Anssi, 2023. Orientation Sensor. Web site. *Wayback Machine. Internet archive*. 1 August. [Accessed 6 November 2023]. Available from: <https://web.archive.org/web/20231022174350/https://www.w3.org/TR/orientation-sensor/>.

CIARAMELLA, Giovanni et al., 2022. A Model Checking-based Approach to Malicious Family Detection in iOS Environment. *Procedia Computer Science*, vol. 207, pp. 1981–1991. [Online] Available from: <https://doi.org/10.1016/J.PROCS.2022.09.257>.

CLASSEN, Jiska et al., 2022. Evil Never Sleeps: When Wireless Malware Stays On After Turning Off iPhones. In: *WiSec 2022 – Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. Association for Computing Machinery, Inc. 16 May, pp. 146–156. ISBN 9781450392167. [Online] Available from: <https://doi.org/10.1145/3507657.3528547>.

CNITSEC, 2022. 国家信息安全漏洞库 [National Information Security Vulnerability Database]. Web site. *Wayback Machine. Internet archive: China Information Technology Security Evaluation Center (CNITSEC)*. 1 May. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220501130155/http://www.cnnvd.org.cn/web/xxk/gyCnnvdJs.tag>.

CNNVD, 2022. 国家信息安全漏洞库 [National Information Security Vulnerability Database]. Web site. *Wayback Machine. Internet archive: China National Vulnerability Database (CNNVD)*. 1 May. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220501130153/http://www.cnnvd.org.cn/web/wz/bzxqById.tag?id=2&mkid=2>.

CNVD, 2022. 国家信息安全漏洞共享平台 [National Information Security Vulnerability Sharing Platform]. Web site. *China National Vulnerability Database (CNVD)*. 15 October. [Accessed 10 November 2023]. Available from: <https://www.cnvd.org.cn/webinfo/list?type=7>.

COOKE, Thomas N., 2020. Metadata, jailbreaking, and the cybernetic governmentality of ios: Or, the need to distinguish digital privacy from digital privacy. *Surveillance and Society*, vol. 18, no. 1, pp. 90–103. [Online] Available from: <https://doi.org/10.24908/SS.V18I1.13118>.

COTTERELL, Kathryn et al., 2015. An Android Security Policy Enforcement Tool. *International Journal of Electronics and Telecommunications*, vol. 61, no. 4, pp. 311–320. [Online] Available from: <https://doi.org/10.1515/ELETEL-2015-0040>.

CURRY, David, 2023. Android Statistics (2023) – Business of Apps. Web site. *Wayback Machine. Internet archive: Business of Apps*. 30 January. [Accessed 13 February 2023]. Available from: <https://web.archive.org/web/20230130073257/https://www.businessofapps.com/data/android-statistics/>.

CVE DETAILS, 2023a. Google Android: CVE security vulnerabilities, versions and detailed reports. Web site. *CVE Details: The ultimate security vulnerability datasource*. 14 February. [Accessed 15 February 2023]. Available from: <https://web.archive.org/web/20230215063933/https://www.cvedetails.com/product/19997/Google-Android.html>.

CVE DETAILS, 2023b. Apple Iphone Os: CVE security vulnerabilities, versions and detailed reports. Web site. *CVE Details: The ultimate security vulnerability datasource*. 15 February. [Accessed 15 February 2023]. Available from: https://web.archive.org/web/20230215065258/https://www.cvedetails.com/product/15556/Apple-Iphone-Os.html?vendor_id=49.

DAR, Muneer Ahmad; PARVEZ, Javed, 2016. Novel techniques to enhance the security of smartphone applications. *International Journal of Interactive Mobile Technologies*, vol. 10, no. 4, pp. 32–36. [Online] Available from: <https://doi.org/10.3991/IJIM.V10I4.5869>.

DIAMANTARIS, Michalis et al., 2020. The Seven Deadly Sins of the HTML5 WebAPI. *ACM Transactions on Privacy and Security*, vol. 23, no. 4. [Online] Available from: <https://doi.org/10.1145/3403947>.

DOCTOR WEB, 2022. Dr.Web – Doctor Web’s October 2022 review of

virus activity on mobile devices. Web site. *Doctor Web*. 2 December. [Accessed 6 December 2022]. Available from: <https://web.archive.org/web/20221205142833/https://news.drweb.com/show/review/?i=14617&lng=en#googleplay>.

ELSERSY, Wael F. et al., 2023. ROOTECTOR: Robust Android Rooting Detection Framework Using Machine Learning Algorithms. *Arabian Journal for Science and Engineering*, vol. 48, no. 2, pp. 1771–1791. [Online] Available from: <https://doi.org/10.1007/S13369-022-06949-5>.

ENCK, William et al., 2009. On Lightweight Mobile Phone Application Certification. In: *Chicago: Association for Computing Machinery*. November, pp. 235–245. ISBN 978-1-60558-352-5.

EZE, Chika et al., 2016. Using visualizations to enhance users' understanding of app activities on android devices. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 7, no. 1, pp. 39–57.

FARINA, Paolo et al., 2016. Are mobile botnets a possible threat? the case of SlowBot Net. *Computers and Security*, vol. 58, pp. 268–283. [Online] Available from: <https://doi.org/10.1016/J.COSE.2016.02.005>.

FIRST, 2022. CVSS v3.1 Specification Document. Web site. *Forum of Incident Response and Security Teams (FIRST), Inc.: Improving Security Together*. 1 November. [Accessed 9 November 2022]. Available from: <https://web.archive.org/web/20221101080431/https://www.first.org/cvss/v3.1/specification-document>.

FORAIN, Igor et al., 2022. Towards System Security: What a Comparison of National Vulnerability Databases Reveals. In: *Iberian Conference on Information Systems and Technologies (CISTI)*, vol. 2022 – June. [Online] Available from: <https://doi.org/10.23919/CISTI54924.2022.9820232>.

GADIANT, Pascal et al., 2018. Security code smells in Android ICC. *Empirical Software Engineering*, vol. 24, no. 5, pp. 3046–3076. [Online] [Accessed 28 November 2022]. Available from: <https://doi.org/10.1007/S10664-018-9673-Y>.

GOLDSTEIN, Eric, 2022. Transforming the Vulnerability Management Landscape – CISA. Web site. *Certified Information Systems Auditor (CISA): An official website of the United States government*. 10 November. [Accessed 16 November 2022]. Available from: <https://web.archive.org/web/20221114172229/https://www.cisa.gov/blog/2022/11/10/transforming-vulnerability-management-landscape#>.

GOOGLE ANDROID, 2023. Permissions on Android. Web site. *Developers*. 7 November. [Accessed 12 November 2023]. Available from: <https://web.archive.org/web/20231107000733/https://developer.android.com/guide/topics/permissions/overview>.

GOOGLE DEVELOPERS, 2023a. Cloud-based protections: Analysis and review. Web site. *Google Play Protect*. 4 November. [Accessed 14 November 2023]. Available from: <https://web.archive.org/web/20231104180701/https://developers.google.com/android/play-protect/cloud-based-protections>.

GOOGLE DEVELOPERS, 2023b. On-device protections. Web site. *Google Play Protect*. 23 October. [Accessed 14 November 2023]. Available from: <https://web.archive.org/web/20231023230239/https://developers.google.com/android/play-protect/client-protections>.

GOOGLE, 2022. SDK Platform release notes. Web site. *Developers: Android Studio*. 24 November. [Accessed 25 November 2022]. Available from: <https://web.archive.org/web/20221124143604/https://developer.android.com/studio/releases/platforms>.

GUO, Chenkai et al., 2015. MalDetector – Using Permission Combinations to Evaluate Malicious Features of Android App. In: *2015 – 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. Beijing, China, 23–25 Sept., pp. 157–160. [Online] Available from: <https://doi.org/10.1109/ICSESS.2015.7339027>.

HARRIS, Mark A.; CHIN, Amita G., 2016. Consumer trust in Google’s top developers’ apps: An exploratory study. *Information and Computer Security*, vol. 24, no. 5, pp. 474–495. Available from: <https://doi.org/10.1108/ICS-11-2015-0044>.

HORSMAN, Graeme, 2019. Loose-Lipped Mobile Device Intelligent Personal Assistants: A Discussion of Information Gleaned from Siri on Locked iOS Devices. *Journal of Forensic Sciences*, vol. 64, no. 1, pp. 231–235. Available from: <https://doi.org/10.1111/1556-4029.13804>.

HUTCHINSON, Shinelle et al., 2019. Are We Really Protected? An Investigation into the Play Protect Service. In: *Proceedings – 2019 IEEE International Conference on Big Data (Big Data)*. Institute of Electrical and Electronics Engineers Inc. 1 December, pp. 4997–5004. ISBN 978-1-7281-0858-2. [Online] Available from: <https://doi.org/10.1109/BIGDATA47090.2019.9006100>.

JANG, Jae-Wook et al., 2016. Andro-Dumpsys: Anti-malware system based on the similarity of malware creator and malware centric information. *Computers & Security*, vol. 58, May, pp. 125–138. [Online] Available from: <https://doi.org/10.1016/J.COSE.2015.12.005>.

JENNEX, Murray E., 2015. Literature reviews and the review process: An editor-in-chiefs perspective. *Communications of the Association for Information Systems*, vol. 36, pp. 139–146. Available from: <https://doi.org/10.17705/1CAIS.03608>.

JHA, Ajay Kumar; LEE, Woo Jin, 2016. Analysis of permission-based security in android through policy expert, developer, and end user perspectives. *Journal of Universal Computer Science*, vol. 22, no. 4, pp. 459–474.

JIN, Junjie; ZHANG, Wei, 2017. System log-based android root state detection. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag, pp. 793–798. ISBN 9783319685410. Available from: https://doi.org/10.1007/978-3-319-68542-7_69.

Ji, Xiaobo et al., 2016. Static Anomaly Detection Framework for Android-based Mobile Phones. *International Journal of Security and Its Applications*, vol. 10, no. 12, pp. 251–260. [Online] Available from: <https://doi.org/10.14257/ijisia.2016.10.12.20>.

JONES, Matthew, 2023. iPhone History: Every Generation in Timeline Order 2007 – 2022. Web site. *History Cooperative*. January 2023. [Accessed 11 February 2023]. Available from: <https://web.archive.org/web/20230112023356/https://historycooperative.org/the-history-of-the-iphone/>.

JVN IPEDIA, 2022. *JVN iPedia. Vulnerability Countermeasure Information Database*. Web site. 1 October. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20221001042457/https://jvndb.jvn.jp/en/>.

JVN, 2022. *JVN–Japan Vulnerability Notes*. Web site. 3 September. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220903085032/http://jvn.jp/>.

KANWAL, Mahpara et al., 2016. Survey of detection tools for android ransomware. *International Journal of Control Theory and Applications*, vol. 9, no. 20, pp. 19–25. [Online] Available from: https://www.researchgate.net/publication/312060132_Survey_of_detection_tools_for_android_ransomware.

KHANDELWAL, Ankita; MOHAPATRA, A. K., 2015. An insight into the security issues and their solutions for android phones. In: *2015 – 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. 11–13 March. New Delhi, India: Institute of Electrical and Electronics Engineers Inc., pp. 106–109. ISBN 9789380544168. [Online] Available from: <https://ieeexplore.ieee.org/document/7100230/metrics#metrics>.

KONG, Pingfan et al., 2019. Automated testing of Android Apps: A systematic literature review. *IEEE Transactions on Reliability*, vol. 68, no. 1, March, pp. 45–66. Available from: <https://doi.org/10.1109/TR.2018.2865733>.

KO, Ru et al., 2015. Vulnerability detection of multiple layer colluding application through intent privilege checking. In: *Proceedings – 2015 – 5th International Conference on IT Convergence and Security (ICITCS)*. Kuala Lumpur, Malaysia: Institute of Electrical and Electronics Engineers Inc., 24–27 August. ISBN 9781467365376. [Online] Available from: <https://doi.org/10.1109/ICITCS.2015.7293036>.

KOSTIAINEN, Anssi, 2023a. Accelerometer. Web site. *Wayback Machine. Internet archive*. 30 January. [Accessed 6 November 2023]. Available from: <https://web.archive.org/web/20231019143008/https://www.w3.org/TR/accelerometer/>.

KOSTIAINEN, Anssi, 2023b. Gyroscope. W3C Candidate Recommendation Draft. Web site. *Wayback Machine. Internet archive*. 20 October. [Accessed 6 November 2023]. Available from: <https://web.archive.org/web/20231021081847/https://www.w3.org/TR/gyroscope/>.

KOSTIAINEN, Anssi; BHAUMIK, Rijubrata, 2023a. Magnetometer. Web site. *Magnetometer. W3C Working Draft*. 30 January. [Accessed 6 November 2023]. Available from: <https://web.archive.org/web/20231021075339/https://www.w3.org/TR/magnetometer/>.

KOSTIAINEN, Anssi; BHAUMIK, Rijubrata, 2023b. Proximity Sensor. Web site. *Wayback Machine. Internet archive*. 30 January. [Accessed 6 November 2023]. Available from: <https://web.archive.org/web/20231017015516/https://www.w3.org/TR/proximity/>.

KOSTIAINEN, Anssi; BHAUMIK, Rijubrata, 2023c. Ambient Light Sensor. Web site. *Wayback Machine. Internet archive*. 20 October. [Accessed 6 November 2023]. Available from: <https://web.archive.org/web/20231021081858/https://www.w3.org/TR/ambient-light/>.

KOSTIAINEN, Anssi et al., 2023. Geolocation Sensor. W3C Working Draft. Web site. *Wayback Machine. Internet archive*. 26 October. [Accessed 6 November 2023]. Available from: <https://web.archive.org/web/20231106071049/https://www.w3.org/TR/geolocation-sensor/>.

KOTKAR, Chetan; GAME, Pravin, 2016. Prevention mechanism for prohibiting SMS malware attack on android smartphone. In: *12th IEEE International Conference Electronics, Energy, Environment, Communication, Computer, Control: (E3 – C3), INDICON 2015*. Institute of Electrical and Electronics Engineers Inc. 29 March 2016. ISBN 9781467373999. Available from: <https://doi.org/10.1109/INDICON.2015.7443579>.

KRAJCI, Iggy; CUMMINGS, Darren, 2013. History and Evolution of the Android OS. In: ИВАНОВ, Иван; ПЕТРОВ, Петър (Eds.). *Android on x86: An Introduction to Optimizing for Intel Architecture*. Berkeley, CA: Apress, pp. 1–8. [Online] [Accessed 13 November 2022]. Available from: https://doi.org/10.1007/978-1-4302-6131-5_1.

KURNIAWAN, Harry et al., 2015. Android anomaly detection system using machine learning classification. In: *Proceedings – 2015 – 5th International Conference on Electrical Engineering and Informatics: Bridging the Knowledge between Academic, Industry, and Community (ICEEI)*. Institute of Electrical and Electronics Engineers Inc. 10 December, pp. 288–293. ISBN 9781467373197. [Online] Available from: <https://doi.org/10.1109/ICEEI.2015.7352512>.

LAAYU, Muhammad Rakha et al., 2022. Comparison of Acquisition Results on iPhone 7 Plus (iOS 14.8.1) between Jailbreaking vs Non-Jailbreaking Device. In: *2022 – 10th International Conference on Information and Communication Technology (ICoICT)*. Institute of Electrical and Electronics Engineers Inc., pp. 402–406. ISBN 9781665481656. [Online] Available from: <https://doi.org/10.1109/ICoICT55009.2022.9914862>.

LA COUR, Alexander S. et al., 2021. Wireless Charging Power Side-Channel Attacks. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 651–665. [Online] Available from: <https://doi.org/10.1145/3460120.3484733>.

LIANG, Hongliang et al., 2016a. PAPDroid: Personalization Awareness Privacy Protection in Android. In: *Proceedings – 2015 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)*. Institute

of Electrical and Electronics Engineers Inc., 7 March 2016, pp. 122–125. ISBN 9781467386371. [Online] Available from: <https://doi.org/10.1109/IIKI.2015.34>.

LIANG, Hongliang et al., 2016b. Survey on Privacy Protection of Android Devices. In: *Proceedings – 2015 – 2nd IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*. Institute of Electrical and Electronics Engineers Inc., 4 January 2016, pp. 241–246. ISBN 9781467392990. [Online] Available from: <https://doi.org/10.1109/CSCLOUD.2015.21>.

LINARES-VASQUEZ, Mario et al., 2017. An empirical study on android-related vulnerabilities. In: *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 20–21 May 2017, pp. 2–13. [Online] Available from: <https://doi.org/10.1109/MSR.2017.60>.

LI, Wenjia et al., 2016. Detecting Malware for Android Platform: An SVM-Based Approach. In: *Proceedings – 2015 – 2nd IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*. Institute of Electrical and Electronics Engineers Inc., 4 January, pp. 464–469. ISBN 9781467392990. [Online] Available from: <https://doi.org/10.1109/CSCLOUD.2015.50>.

MALIK, Sapna; KHATTER, Kiran, 2016a. Behaviour analysis of android application. *International Journal of Control Theory and Applications*, vol. 9, no. 11, pp. 5307–5322. [Online] Available from: https://www.researchgate.net/publication/311795421_Behaviour_analysis_of_android_application.

MALIK, Sapna; KHATTER, Kiran, 2016b. System call analysis of Android Malware families. *Indian Journal of Science and Technology*, vol. 9, no. 21, pp. 1–13. School of Engineering and Technology. [Online] Available from: <https://doi.org/10.17485/IJST/2016/V9I21/90273>.

MAO, Jian et al., 2016. Automatic permission inference for hybrid mobile apps. *Journal of High Speed Networks*, vol. 22, no. 1, pp. 55–64. [Online] Available from: <https://doi.org/10.3233/JHS-160538>.

MARKERT, Philipp et al., 2021. On the Security of Smartphone Unlock PINs. *ACM Transactions on Privacy and Security (TOPS)*, vol. 24, issue 4, article no. 30, pp. 1–36. [Online] Available from: <https://doi.org/10.1145/3473040>.

MARSAL, Katie, 2010. Cisco licenses iOS name to Apple, screenshot shows iWork on iPhone. Web site. *Wayback Machine. Internet archive*. 8 June. [Accessed 28 November 2022]. Available from: https://web.archive.org/web/20220524150811/https://appleinsider.com/articles/10/06/08/cisco_licenses_ios_name_to_apple_

screenshot_shows_iwork_on_iphone#.

MAZUERA-ROZO, Alejandro et al., 2019. The Android OS stack and its vulnerabilities: an empirical study. *Empirical Software Engineering. An International Journal*, vol. 24, no. 4, pp. 2056–2101. [Online] [Accessed 25 October 2022]. Available from: <https://doi.org/10.1007/S10664-019-09689-7>.

MEHRNEZHAD, Maryam; TOREINI, Ehsan, 2019. What is This Sensor and Does This App Need Access to It? *Informatics*, vol. 6, no. 1, p. 7. [Online] Available from: <https://doi.org/10.3390/INFORMATICS6010007>.

MEHRNEZHAD, Maryam et al., 2016. TouchSignatures: Identification of user touch actions and PINs based on mobile sensor data via JavaScript. *Journal of Information Security and Applications*, vol. 26, February, pp. 23–38. [Online] Available from: <https://doi.org/10.1016/J.JISA.2015.11.007>.

MENG, Weizhi et al., 2015. Charging me and i know your secrets!: Towards juice filming attacks on smartphones. In: *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security (CPSS). Part of ASIA CCS – 2015*, pp. 89–98. [Online] Available from: <https://doi.org/10.1145/2732198.2732205>.

MENG, Weizhi et al., 2016. JuiceCaster: Towards automatic juice filming attacks on smartphones. *Journal of Network and Computer Applications*, vol. 68, June, pp. 201–212. [Online] Available from: <https://doi.org/10.1016/J.JNCA.2016.04.015>.

MENG, Weizhi et al., 2019. Towards detection of juice filming charging attacks via supervised CPU usage analysis on smartphones. *Computers and Electrical Engineering*, vol. 78, pp. 230–241. [Online] Available from: <https://doi.org/10.1016/J.COMPELECENG.2019.07.008>.

MERCALDO, Francesco et al., 2016. Ransomware steals your phone. Formal methods rescue it. In: ALBERT, Elvira; LANESE, Ivan (Eds.), 2016. Formal Techniques for Distributed Objects, Components, and Systems (FORTE). – Part of the book series: *Lecture Notes in Computer Science (LNCS)*, vol. 9688, pp. 212–221. [Online] Available from: https://doi.org/10.1007/978-3-319-39570-8_14.

MICHALEVSKY, Yan et al., 2014. Gyrophone: Recognizing speech from gyroscope signals. In: *Proceedings of the 23rd USENIX Security Symposium*. USENIX Association, 20–22 August, pp. 1053-1067. ISBN 978-1-931971-15-7. [Online] Available from: <https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-michalevsky.pdf>.

MITRE, 2022a. CVE – CVE List Search Tips. Web site. *Wayback Machine. Internet archive*. [Accessed 3 November 2022]. Available from: https://web.archive.org/web/20221007022347/https://cve.mitre.org/find/search_tips.html.

MITRE, 2022b. CWE – CWE List Version 4.8. Web site. *Common Weakness Enumeration (CWE)*. 14 September 2022. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220914235250/https://cwe.mitre.org/data/index.html>.

MITRE, 2023. CAPEC – CAPEC-3000: Domains of Attack (Version 3.9). Web site. *Common Attack Pattern Enumeration and Classification (CAPEC)*, 19 May 2023. [Accessed 8 August 2023]. Available from: <https://web.archive.org/web/20230519025214/https://capec.mitre.org/data/definitions/3000.html>.

MOREAU, Seán, 2021. The evolution of iOS. A look at the evolution of Apple’s mobile operating system from 2007 to today. Web site. *Computerworld*. [Accessed 25 November 2022]. Available from: <https://web.archive.org/web/20220412035433/https://www.computerworld.com/article/2975868/the-evolution-of-ios.html#>.

MUHAMMAD, Zia et al., 2023. Smartphone Security and Privacy: A Survey on APTs, Sensor-Based Attacks, Side-Channel Attacks, Google Play Attacks, and Defenses. *Technologies*, vol. 11, no. 3. [Online] Available from: <https://doi.org/10.3390/TECHNOLOGIES11030076>.

MUNOZ, Alfonso et al., 2015. Android malware detection from Google Play meta-data: Selection of important features. In: *2015 – IEEE Conference on Communications and Network Security (CNS)*. Florence: Institute of Electrical and Electronics Engineers Inc., 3 December, pp. 701–702. ISBN 9781467378765. [Online] Available from: <https://doi.org/10.1109/CNS.2015.7346893>.

NIST, 2022. NVD (National Vulnerability Database) – CVEs and the NVD Process. Web site. *National Institute of Standards and Technology. U. S. Department of Commerce (NIST)*. 9 November 2022. [Accessed 9 November 2022]. Available from: <https://web.archive.org/web/20221109123234/https://nvd.nist.gov/general/cve-process>.

NL NCSC, 2022. Become an NCSC partner and receive relevant information. Web site. *Netherlands: National Cyber Security Centre (NCSC)*. 21 April 2022. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220421172916/https://english.ncsc.nl/get-to-work/become-a-partner>.

OLEJNIK, Lukasz, 2017. Stealing sensitive browser data with the W3C Ambient Light Sensor API. Web site. *Security, Privacy & Tech Inquiries*. 19 April 2017. [Accessed 7 November 2023]. Available from: <https://web.archive.org/web/20230526110009/https://blog.lukaszolejnik.com/stealing-sensitive-browser-data-with-the-w3c-ambient-light-sensor-api/>.

OLEJNIK, Lukasz et al., 2017. Battery status not included: Assessing privacy in web standards. In: *CEUR Workshop Proceedings*. CEUR-WS. 2017. pp. 17–24. [Online] Available from: <https://lukaszolejnik.com/AssessingPrivacyWebStandardsIWPE17.pdf>.

ONGTANG, Machigar et al., 2009. Semantically rich application-centric security in android. In: *Proceedings – 2009 – Annual Computer Security Applications Conference (ACSAC)*. Honolulu, HI, USA, 7–11 December, pp. 340–349. ISSN 1063-9527. [Online] Available from: <https://doi.org/10.1109/ACSAC.2009.39>.

PARK, Wonjoo et al., 2015. Detecting malware with similarity to Android applications. In: *2015 – International Conference on Information and Communication Technology Convergence (ICTC): Innovations Toward the IoT, 5G, and Smart Media Era*. Institute of Electrical and Electronics Engineers Inc., Jeju, Korea (South), 28–30 October, pp. 1249–1251. ISBN 9781467371155. [Online] Available from: <https://doi.org/10.1109/ICTC.2015.7354788>.

PIAO, Yuxue et al., 2016. Server-based code obfuscation scheme for APK tamper detection. *Security and Communication Networks*, vol. 9, no. 6, pp. 457–467. [Online] Available from: <https://doi.org/10.1002/SEC.936>.

RAPHAEL, JR (Editor), 2023. Android versions: A living history from 1.0 to 16. Web site. *Computerworld*. 10 Jun. [Accessed 20 November 2023]. Available from: <https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html>.

RYTEL, Marcin et al., 2020. Towards a Safer Internet of Things – A Survey of IoT Vulnerability Data Sources. *Sensors 2020*, vol. 20, no. 21, p. 5969. [Online] [Accessed 25 October 2022]. Available from: <https://doi.org/10.3390/S20215969>.

SALAH, Saeed et al., 2022. Desktop and mobile operating system fingerprinting based on IPv6 protocol using machine learning algorithms. *International Journal of Security and Networks*, vol. 17, no. 1, pp. 1–12. [Online] Available from: <https://doi.org/10.1504/IJSN.2022.122543>.

SARDANA, Neetu; BHATT, Arpita Jadhav, 2022. CiAFP: Category-based Classification of iOS Apps by mining frequent permissions. In: *IC3 – 2022: Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing*. Association for Computing Machinery. Noida, India, 4–6 August, pp. 22–26. ISBN 978-1-4503-9675-2. [Online] Available from: <https://doi.org/10.1145/3549206.3549211>.

SCHULTZ, Clayton W. et al., 2021. Three-Dimensional Conductive Fingerprint Phantoms Made of Ethylene-Vinyl Acetate/Graphene Nanocomposite for Evaluating Smartphone Scanners. *ACS Applied Electronic Materials*, vol. 3, no. 5, pp. 2097–2105. [Online] Available from: <https://doi.org/10.1021/ACSAELM.1C00119>.

SEHGAL, Garima; KUMAR, R., 2016. Risk information analysis for android applications. *International Journal of Control Theory and Applications*, vol. 9, no. 13, pp. 6323–6328.

SHAO, Yuru et al., 2014. RootGuard: Protecting rooted android phones. *Computer*, vol. 47, no. 6, pp. 32–40. [Online] Available from: <https://doi.org/10.1109/MC.2014.163>.

SOEWITO, Benfano; SUWANDARU, Agung, 2022. Android sensitive data leakage prevention with rooting detection using Java function hooking. *Journal of King Saud University – Computer and Information Sciences*, vol. 34, no. 5, pp. 1950–1957. [Online] Available from: <https://doi.org/10.1016/J.JKSUCI.2020.07.006>.

SOH, Charlie et al., 2015. Detecting Clones in Android Applications through Analyzing User Interfaces. In: *2015 – IEEE 23rd International Conference on Program Comprehension*. IEEE Computer Society. 18–19 May, pp. 163–173. ISBN 9781467381598. [Online] Available from: <https://doi.org/10.1109/ICPC.2015.25>.

SOMARRIBA, Oscar et al., 2016. Detection and Visualization of Android Malware Behavior. *Journal of Electrical and Computer Engineering*, 14 March 2016. ESSN 2090-0155. ISSN 2090-0147 (Print). [Online] Available from: <https://doi.org/10.1155/2016/8034967>.

SONG, Sanggeun et al., 2016. The Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform. *Mobile Information Systems*, 15 May 2016. [Online] Available from: <https://doi.org/10.1155/2016/2946735>.

SPIVAKOVSKY, Alex, 2022. Why CVE Management as a Primary Strategy Doesn't Work. Web site. *Dark Reading*. 11 November 2022. [Accessed 15 November 2022]. Available from: <https://www.darkreading.com/vulnerabilities-threats/why-cve-management-as-a-primary-strategy-doesn-t-work>.

STATCOUNTER, 2022. Mobile Operating System Market Share Worldwide. Web site. *Statcounter Global Stats*. 10 January 2022. [Accessed 25 November 2022]. Available from: <https://web.archive.org/web/20220110031504/https://gs.statcounter.com/os-market-share/mobile/worldwide#>.

STATISTA, 2023. Global: Number of smartphone users 2013 – 2028. Web site. *Statista*. [Accessed 7 November 2023]. Available from: <https://www.statista.com/forecasts/1143723/smartphone-users-in-the-world>.

SURENDRAN, Roopak; THOMAS, Tony, 2022. Detection of malware applications from centrality measures of syscall graph. *Concurrency and Computation: Practice and Experience*, vol. 34, no. 10, 1 May, e6835. [Online] Available from: <https://doi.org/10.1002/CPE.6835>.

TAKAHASHI, Takeshi; BAN, Tao, 2016. 6 – 3 risk analysis system for android applications. *Journal of the National Institute of Information and Communications Technology*, vol. 63, no. 2, pp. 155–159. [Online] Available from: <https://www.nict.go.jp/publication/shuppan/kihou-journal/journal-vol63no2/journal-vol63no2-06-03.pdf>

TIWARI, Pradeep Kumar; VELAYUTHAM, T., 2019. Android Vulnerabilities: Taxonomy and nextGen Ecosystem. In: *2019 – IEEE Bombay Section Signature Conference (IBSSC)*. Mumbai, India, 26–28 July. [Online] Available from: <https://doi.org/10.1109/IBSSC47189.2019.8973083>.

UK NCSC, 2022. CISP – Cyber Security Information Sharing Partnership – NCSC.GOV.UK. Web site. *National Cyber Security Centre (NCSC)*. 2 September. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20220902073419/https://www.ncsc.gov.uk/section/keep-up-to-date/cisp>.

ULLAH, Salim et al., 2022. Understanding Users' Behavior towards Applications Privacy Policies. *Electronics*, vol. 11, no. 2, pp. 246. EISSN 2079-9292. Multidisciplinary Digital Publishing Institute (MDPI), Basel, Switzerland. [Online] [Accessed 11 November 2023]. Available from: <https://doi.org/10.3390/ELECTRONICS11020246>.

UMASANKAR, 2017. Analysis of latest vulnerabilities in android. In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, vol. 2017, January, pp. 1236–1241. Udupi, India, 13–16 September. [Online] Available from: <https://doi.org/10.1109/ICACCI.2017.8126011>.

VECCHIATO, Daniel et al., 2015. A security configuration assessment for android devices. In: *Symposium on Applied Computing (SAC) '15: Proceedings of the 30th Annual ACM Symposium on Applied Computing*. Association for Computing Machinery. 13 April, pp. 2299–2304. ISBN 9781450331968. [Online] Available from: <https://doi.org/10.1145/2695664.2695679>.

VOO, Julia et al., 2020. National Cyber Power Index 2020 Methodology and Analytical Considerations. Web site. *Harvard Kennedy School: Belfer Center for Science and International Affairs*. [Online] [Accessed 25 October 2022]. Available from: www.belfercenter.org/CCPI.

W3C, 2023. W3C standards and drafts. Web site. *W3C (World Wide Web Consortium)*. 1 November. [Accessed 2 November 2023]. Available from: <https://web.archive.org/web/20231101072221/https://www.w3.org/TR/>.

WALDRON, Rick, 2023. Generic Sensor API: W3C Candidate Recommendation Draft, 6 October. Web site. *Wayback Machine. Internet archive*. [Accessed 6 November 2023]. Available from: <https://web.archive.org/web/20231017015536/https://www.w3.org/TR/generic-sensor/>.

WANG, Haoyu et al., 2015. Using text mining to infer the purpose of permission use in mobile apps. In: *UbiComp 2015 – Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. Osaka, Japan, 7–11 September, pp. 1107–1118. ISBN 978-1-4503-3574-4. [Online] Available from: <https://doi.org/10.1145/2750858.2805833>.

WANG, Pei et al., 2019. Field experience with obfuscating million-user iOS apps in large enterprise mobile development. *Journal of Software – Practice and Experience*, vol. 49, no. 2, Special Issue: Software Engineering in Practice, pp. 252–273. [Online] Available from: <https://doi.org/10.1002/SPE.2648>.

WESTYARIAN et al., 2015. Malware detection on Android smartphones using API class and machine learning. In: *Proceedings – 2015 – 5th International Conference on Electrical Engineering and Informatics (ICEEI): Bridging the Knowledge between Academic, Industry, and Community*. Institute of Electrical and Electronics Engineers Inc. Denpasar, Indonesia, 10–11 August, pp. 294–297.

EISBN 978-1-4673-7319-7. ISBN 978-1-4673-6778-3 (Print). [Online] Available from: <https://doi.org/10.1109/ICEEL.2015.7352513>.

XENAKIS, Christos; NTANTOGIAN, Christoforos, 2015. Attacking the baseband modem of mobile phones to breach the users' privacy and network security. In: *2015 – 7th International Conference on Cyber Conflict: Architectures in Cyberspace (CYCON)*. Tallinn, Estonia, 26–29 May, pp. 231–244. EISBN 978-9-9499-5443-8. ISBN 978-9-9499-5442-1 (Print). [Online] Available from: <https://doi.org/10.1109/CYCON.2015.7158480>.

XIAO, Xi et al., 2016. Identifying Android malware with system call co-occurrence matrices. *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 5, pp. 675–684. [Online] Available from: <https://doi.org/10.1002/ETT.3016>.

XU, Ke et al., 2016. ICC Detector: ICC – Based Malware Detection on Android. *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, June, pp. 1252–1264. EISSN 1556-6021. ISSN 1556-6013 (Print). [Online] Available from: <https://doi.org/10.1109/TIFS.2016.2523912>.

YANG, Tianchang et al., 2016. An analysis on sensitive data passive leakage in Android applications. In: *Proceedings – 2015 IEEE – 16th International Conference on Communication Technology (ICCT)*. Hangzhou, China, 18–20 October 2015, pp. 125–131. *IEEE Xplore*, vol. February 2016. [Online] Available from: <https://doi.org/10.1109/ICCT.2015.7399807>.

YUAN, Zhenlong et al., 2016. Droiddetector: Android malware characterization and detection using deep learning. *Tsinghua Science and Technology*, vol. 21, no. 1, February, pp. 114–123. EISSN 1007-0214. [Online] Available from: <https://doi.org/10.1109/TST.2016.7399288>.

ZAMAN, Mehede et al., 2015. Malware detection in Android by network traffic analysis. In: *Proceedings of 2015 International Conference on Networking Systems and Security (NSysS)*. Dhaka, Bangladesh, 5–7 January. Institute of Electrical and Electronics Engineers Inc. EISBN 978-1-4799-8126-7. [Online] Available from: <https://doi.org/10.1109/NSYSS.2015.7043530>.

БДУБИ, 2022. БДУ – Вход. Web site. *Банк данных угроз безопасности информации (БДУБИ)*. 2 October 2022. [Accessed 10 November 2023]. Available from: <https://web.archive.org/web/20221006232715/https://bdu.fstec.ru/>.

ЗЛАТЕВА, Денка, 2012. *Сензорен анализ*. Икономически университет – Варна. Варна: Издателство „Наука и икономика“, с. 1–188. ISBN 978-954-21-0640-1.

КАМЕНОВА, Боряна, 1995. *Английско-български речник по информатика*. София: „Литера Прима“. ISBN 954-8163-33-0.

Приложения

Приложение 1. Опитни данни по експеримент към глава трета

UID	OS	Browser	Device	Orientation	Acceleration	RotationRate	Proximity	Light	Magnetometer	Compass	Battery
1	Android 14	Chrome	Google Pixel 8	фп	1	1	0	0	0	1	1
2	Android 13	FireFox	Samsung Galaxy A32 5G	1	фп	фп	0	0	0	0	0
3	Android 13	Samsung Internet	Samsung Galaxy A32 5G	фп	1	1	0	0	0	1	0
4	Android 13	Chrome	Samsung Galaxy A32 5G	фп	1	1	0	0	0	1	1
12	iOS 10	Safari	iPhone 7	0	0	0	0	0	0	0	0
13	iOS 15	Safari	iPhone 15	0	0	0	0	0	0	0	0
16	iOS 17	Safari	iPhone 15	0	0	0	0	0	0	0	0
17	iOS 17	Chrome	iPhone 15	0	0	0	0	0	0	0	0
18	iOS 16	Safari	iPhone 14Pro	0	0	0	0	0	0	0	0
19	iOS 16	Chrome	iPhone 14Pro	0	0	0	0	0	0	0	0
20	iOS 17	Safari	iPhone 13	0	0	0	0	0	0	0	0
21	iOS 17	Chrome	iPhone 13	0	0	0	0	0	0	0	0
22	iOS 15	Safari	iPhone 13	0	0	0	0	0	0	0	0
23	iOS 15	Chrome	iPhone 13	0	0	0	0	0	0	0	0
25	iOS 16	Safari	iPhone 13	0	0	0	0	0	0	0	0
26	iOS 14	Safari	iPhone 12	0	0	0	0	0	0	0	0
27	iOS 14	Chrome	iPhone 12	0	0	0	0	0	0	0	0
28	iOS 17	Safari	iPhone 12	0	0	0	0	0	0	0	0
31	iOS 13	Safari	iPhone 11	0	0	0	0	0	0	0	0
32	iOS 13	Chrome	iPhone 11	0	0	0	0	0	0	0	0
33	iOS 14	Safari	iPhone 11	0	0	0	0	0	0	0	0
34	iOS 14	Chrome	iPhone 11	0	0	0	0	0	0	0	0
35	iOS 15	Safari	iPhone 11	0	0	0	0	0	0	0	0

36	iOS 15	Chrome	iPhone 11	0	0	0	0	0	0	0	0
37	iOS 15	Safari	iPhone 8	0	0	0	0	0	0	0	0
38	iOS 15	Chrome	iPhone 8	0	0	0	0	0	0	0	0
39	iOS 13	Safari	iPhone 8	0	0	0	0	0	0	0	0
40	iOS 13	Chrome	iPhone 8	0	0	0	0	0	0	0	0
41	iOS 15	Safari	iPhone XS	0	0	0	0	0	0	0	0
42	iOS 15	Chrome	iPhone XS	0	0	0	0	0	0	0	0
43	iOS 14	Safari	iPhone XS	0	0	0	0	0	0	0	0
44	iOS 14	Chrome	iPhone XS	0	0	0	0	0	0	0	0
45	iOS 13	Safari	iPhone XS	0	0	0	0	0	0	0	0
46	iOS 13	Chrome	iPhone XS	0	0	0	0	0	0	0	0
47	iOS 12	Safari	iPhone XS	1	1	1	0	0	0	0	0
48	iOS 12	Chrome	iPhone XS	1	1	1	0	0	0	0	0
49	iOS 12	Safari	iPhone 15 Pro Max	0	0	0	0	0	0	0	0
50	iOS 17	Chrome	iPhone 15 Pro Max	0	0	0	0	0	0	0	0
51	iOS 16	Safari	iPhone 14 Pro Max	0	0	0	0	0	0	0	0
52	iOS 16	Chrome	iPhone 14 Pro Max	0	0	0	0	0	0	0	0
53	iOS 15	Safari	iPhone 13 Pro Max	0	0	0	0	0	0	0	0
54	iOS 15	Chrome	iPhone 13 Pro	0	0	0	0	0	0	0	0
55	iOS 14	Safari	iPhone 12 Pro Max	0	0	0	0	0	0	0	0
56	iOS 14	Chrome	iPhone 12 Pro Max	0	0	0	0	0	0	0	0
57	iOS 13	Chrome	iPhone 11 Pro Max	0	0	0	0	0	0	0	0
58	iOS 13	Safari	iPhone 11 Pro Max	0	0	0	0	0	0	0	0
59	iOS 14	Safari	iPhone 11 Pro Max	0	0	0	0	0	0	0	0
60	iOS 14	Chrome	iPhone 11 Pro Max	0	0	0	0	0	0	0	0

61	iOS 16	Safari	iPhone 11 Pro Max	0	0	0	0	0	0	0	0
62	iOS 16	Chrome	iPhone 11 Pro Max	0	0	0	0	0	0	0	0
63	Android 13	Chrome	Samsung Galaxy S23	φπ	1	1	0	0	0	1	1
64	Android 13	Edge	Samsung Galaxy S23	φπ	1	1	0	0	0	1	1
66	Android 13	Samsung Internet	Samsung Galaxy S23	φπ	1	1	0	0	0	1	0
67	Android 13	Chrome	Samsung Galaxy Z Fold 5	φπ	1	1	0	0	0	1	1
68	Android 13	Edge	Samsung Galaxy Z Fold 5	φπ	1	1	0	0	0	1	1
69	Android 13	Samsung Internet	Samsung Galaxy Z Fold 5	φπ	1	1	0	0	0	1	0
70	Android 13	Firefox	Samsung Galaxy Z Fold 5	0	0	0	0	0	0	0	0
71	Android 12	Chrome	Samsung Galaxy S22	φπ	1	1	0	0	0	1	1
72	Android 12	Edge	Samsung Galaxy S22	φπ	1	1	0	0	0	1	1
73	Android 12	Firefox	Samsung Galaxy S22	0	0	0	0	0	0	0	0
74	Android 12	Samsung Internet	Samsung Galaxy S22	φπ	1	1	0	0	0	1	1
75	Android 11	Chrome	Samsung Galaxy S21	φπ	1	1	0	0	0	1	1
76	Android 11	Edge	Samsung Galaxy S21	φπ	1	1	0	0	0	1	1
77	Android 11	Firefox	Samsung Galaxy S21	0	0	0	0	0	0	0	0
78	Android 11	Samsung Internet	Samsung Galaxy S21	φπ	1	1	0	0	0	1	1
79	Android 10	Chrome	Samsung Galaxy S20	φπ	1	1	0	0	0	1	1

80	Android 10	Edge	Samsung Galaxy S20	φπ	1	1	0	0	0	1	1
81	Android 10	Firefox	Samsung Galaxy S20	0	0	0	0	0	0	0	0
82	Android 10	Samsung Internet	Samsung Galaxy S20	φπ	1	1	0	0	0	1	1
83	Android 11	Chrome	Samsung Galaxy A52	φπ	1	1	0	0	0	1	1
84	Android 11	Edge	Samsung Galaxy A52	φπ	1	1	0	0	0	1	1
85	Android 11	Firefox	Samsung Galaxy A52	0	0	0	0	0	0	0	0
86	Android 11	Samsung Internet	Samsung Galaxy A52	φπ	1	1	0	0	0	1	1
87	Android 11	Chrome	Samsung Galaxy A11	0	1	φπ	0	0	0	0	1
88	Android 10	Edge	Samsung Galaxy A11	0	1	φπ	0	0	0	0	1
89	Android 10	Firefox	Samsung Galaxy A11	0	0	0	0	0	0	0	0
90	Android 13	Chrome	Google Pixel 7	φπ	1	1	0	0	0	1	1
91	Android 13	Edge	Google Pixel 7	φπ	1	1	0	0	0	1	1
92	Android 13	Edge	Google Pixel 7	0	0	0	0	0	0	0	0
94	Android 13	Chrome	Google Pixel 7-Real	φπ	1	1	0	0	0	1	1
95	Android 12	Chrome	Google Pixel 6	φπ	1	1	0	0	0	1	1
96	Android 12	Edge	Google Pixel 6	φπ	1	1	0	0	0	1	1
97	Android 12	Firefox	Google Pixel 6	0	0	0	0	0	0	0	0
98	Android 12	Chrome	Google Pixel 5	φπ	1	1	0	0	0	1	1
99	Android 12	Edge	Google Pixel 5	φπ	1	1	0	0	0	1	1
100	Android 12	Firefox	Google Pixel 5	0	0	0	0	0	0	0	0
102	Android 13	Chrome	Xiaomi Redmi Note 12 4G	0	1	1	0	0	0	1	1
103	Android 13	Firefox	Xiaomi Redmi Note 12 4G	0	0	0	0	0	0	0	0

104	Android 12	Chrome	Xiaomi Redmi Note 12 Pro	φπ	1	1	0	0	0	1	1
105	Android 12	Edge	Xiaomi Redmi Note 12 Pro	φπ	1	1	0	0	0	1	1
106	Android 12	Firefox	Xiaomi Redmi Note 12 Pro	0	0	0	0	0	0	0	0
107	Android 11	Chrome	Xiaomi Redmi Note 11	φπ	1	1	0	0	0	1	1
108	Android 11	Chrome	Xiaomi Redmi Note 11	φπ	1	1	0	0	0	1	1
109	Android 11	Firefox	Xiaomi Redmi Note 11	0	0	0	0	0	0	0	0
110	Android 9	Chrome	Xiaomi Redmi Note 8	φπ	1	1	0	0	0	1	1
111	Android 9	Edge	Xiaomi Redmi Note 8	φπ	1	1	0	0	0	1	1
112	Android 9	Firefox	Xiaomi Redmi Note 8	0	0	0	0	0	0	0	0

Приложение 2. Изходен код към глава трета

```
Index.php
<?php
    // Настройка на време за изтичане на сесията. 24 часа
    = 86400 секунди
    ini_set('session.gc_maxlifetime', 86400);
    session_start();

    // Създаване на файлове за резултатите от теста и
html отчета
    if (isset($_SESSION['reportFile'])) {
        require('testDone.php');
    } else {
        $_SESSION=array();
        $_SESSION['testsDone']=false;
        require('createReportFiles.php');
    }

    // Проверка, дали тестът е автоматизиран, или е
потребител с реално устройство
        if (isset($_GET['testMode'])&&$_
GET['testMode']=== 'auto') {

        // Прочитане на версията на ОС, браузър и устройство
        $OS='{ "type": "OS", "data": "' . $_GET['OS'] . "' }';
        $browser='{ "type": "Browser", "data": "' . $_
GET['browser'] . "' }';
        $device='{ "type": "Device", "data": "' . $_
GET['device'] . "' }';
    } else {
        //
        $OS='{ "type": "OS", "data": "" }';
        $browser='{ "type": "Browser", "data": "" }';
        $device='{ "type": "Device", "data": "" }';
    }
}
```

```

        // Добавяне на версията на ОС, браузър и устройство
към доклада
        $records = array($OS, $browser, $device);
        foreach ($records as $record) {
            file_put_contents($_SESSION["reportFile"], $record
. "\n", FILE_APPEND | LOCK_EX);
        }

?>

<!DOCTYPE html>
<html>
<head>
    <!-- Set the character encoding to UTF-8 -->
    <meta charset="UTF-8">
    <title>Mobile Device Sensor Access Testing via
Internet</title>
    <link rel="stylesheet" type="text/css" href="styles/
stylesheet.css">

    <script src="scripts/init_test.js"></script>

</head>

<?php
    // Осигуряване на автоматично зареждане на функцията
init(), ако тестът е автоматичен
            if (isset($_GET['testMode'])&&$_
GET['testMode']=='auto'){
                echo '<body class="body" onload="init()">';
            } else {
                echo '<body class="body">';
            }
?>

    <h3 align=center>Mobile Device Sensor Access Testing
via Internet</h3>

```

```

<div class="container">
    <p class="p1">Този уебсайт не обработва и не
    съхранява специални данни по смисъла на GDPR.<br>Няколко
    секунди след натискането на бутона „Започни теста“ ще
    получите кратък отчет за сензорите, до които има достъп
    браузърът, който използвате.</p>
    <p class="p1">This web site does not process or
    store special data within the meaning of the GDPR.<br>A
    few seconds after pressing the "Start test" button, you
    will receive a short report about the sensors that the
    browser you are using has access to.</p>
    <p id="line1"></p>

    <?php
        if ($_SESSION['testsDone']){
            echo '<button id="doTestButton" class="button1"
disabled">Този тест вече е изпълняван</button><br>';
        } else {
            echo '<button id="doTestButton" class="button1"
onclick="init()">Започни        теста<br>Start        test</
button><br>';
        }
    ?>
    <p id="start"></p>
    <p id="end"></p>
    <p id="info"></p>

    <?php
        // Генериране на линк към отчета
        if ($_SESSION['testsDone']){
            $flink = '<a href="" . $_SESSION["reportFileHTML"]
. "" id="report">Изтеглете отчета</a>';
        } else {
            $flink = '<a href="" . $_SESSION["reportFileHTML"]
. "" id="report" style="display: none;" >Изтеглете
отчета</a>';

```

```

    }
    // Извеждане на линка
    echo $flink;
    //
    ?>
        <img alt="Captured Image" id="capturedImage"
style="display: none;">
        <audio id="capturedAudio" style="display: none;"
controls></audio>
    </div>

</body>
</html>

```

generateHTMLReport.php

```

<?php
    session_start();
    // Зареждане на номера на протокола
    $fname="reportCount.txt";
    // Прочитане на текущата стойност за брой на отчетите
(ако съществува)
    if (file_exists($fname)) {
        $num = intval(file_get_contents($fname));
    } else {
        $num = 0; // Инициализация с нула, ако файлът не
съществува
    }

    $resultData=array();
    $dataArray=array();
    $dataArray["UID"]=$num;
    $resultData["UID"]=$num;
    if (isset($_SESSION["reportFile"])) {
        $fname=$_SESSION["reportFile"];
        $fnameHTML=$_SESSION["reportFileHTML"];
        $file = fopen($fname, 'r');

```

```

if ($file) {
    while (($line = fgets($file)) !== false) {
        $decodedData = json_decode($line, true); //
Превръщаме JSON запис в асоциативен масив
        print_r($line);
        if ($decodedData !== null) {
            $dataArray[$decodedData['type']] =
$decodedData['data']; // Добавяме асоциативния масив към
основния масив
            // Това е за втория csv файл с логическите
резултати
            $resultData[$decodedData['type']] =
(int)$decodedData['logical'];
        }
    }
    fclose($file);
} else {
    echo "Грешка при отварянето на файла.";
}

} else {
    echo "No report file name set";
}

// Изчистване на тялото на отчета
$reportBody='';

// Отваряне на файла и добавяне на html тагове
$file=fopen($fnameHTML,'a');

fwrite($file, `
<!DOCTYPE html>
<html>
<head>
            <meta charset="UTF-8">
</head>

```

```

<body>
  <p style="text-align: center; font-weight:
bold;">Протокол № \.$dataArray["UID"].'</p>' .
  \<strong>Начало на теста:</strong>
  \.$dataArray["startTime"].'<br>' .
  \<strong>Край на теста:</strong>
  \.$dataArray["endTime"].'<br>'
  );

  if (isset( $dataArray["Device"]))
    $reportBody=$reportBody.'<strong>Изследвано
устройство:</strong> \.$dataArray["Device"].'<br>';
  if (isset( $dataArray["OS"]))
    $reportBody=$reportBody.'<strong>Операционна
система:</strong> \.$dataArray["OS"].'<br>';
  if (isset( $dataArray["Browser"]))
    $reportBody=$reportBody.'<strong>Браузър:</
strong> \.$dataArray["Browser"].'<br>';

  $reportBody=$reportBody.
  \<strong>Информация за устройството:</strong>
  \.$dataArray["devInfo"].'<br>' .
  \<strong>Достъп до вибрацията<sup>1</sup></sup>:</
strong> \.$dataArray["vibration"].'<br>' .
  \<strong>Достъп до сензора за ориентацията
на устройството в пространството:</strong>
  \.$dataArray["orientation"].'<br>' .
  \<strong>Достъп до сензора за ускорение:</strong>:
  \.$dataArray["acceleration"].'<br>' .
  \<strong>Достъп до честота на завъртане:</strong>
  \.$dataArray["rotationRate"].'<br>' .
  \<strong>Достъп до сензора за близост:</strong>
  \.$dataArray["proximity"].'<br>' .
  \<strong>Достъп до сензора за осветеност:</strong>
  \.$dataArray["light"].'<br>' .
  \<strong>Достъп до магнитометър:</strong>

```

```

`. $dataArray["magnetometer"].' <br>' .
    `<strong>Достъп до компас:</strong>
`. $dataArray["compass"].' <br>' .
    `<strong>Достъп до батерията: </strong>
`. $dataArray["battery"].' <br>' .
    `<strong>Достъп до сензора за геолокация:</strong>
`. $dataArray["geolocation"].' <br>' .
    `<strong>Достъп до камерата<sup>2</sup>:</strong>
`. $dataArray["image"].' <br>' .
    `<strong>Достъп до микрофона<sup>3</sup>:</strong>
`. $dataArray["audio"].' <br>' .
    `<br>';

```

```

// Добавяне на тялото част от отчета
fwrite($file, $reportBody);

```

```

// Добавяне на бележките под линия
fwrite($file, `
    <i>1. Тестът за вибрация дава фалшиво положителни
резултати. Достъпът е успешен само в случай, че устройството
е започнало да вибрира по време на теста.</i><br>
    <i>2. Риск за сигурността има, ако достъпът до
камерата е успешен без браузърът да е поискал разрешение
за достъп.</i><br>
    <i>3. Риск за сигурността има, ако достъпът до
микрофона е успешен, без браузърът да е поискал разрешение
за достъп.<br>Прехващането на звук продължава дори когато
браузърът работи във фонов режим.</i><br>'
);

```

```

// Изчисляване на ниво на сигурност
$security_risk = $resultData["orientation"]
    + $resultData["acceleration"]
    + $resultData["rotationRate"]

```

```

+ $resultData["proximity"]
+ $resultData["light"]
+ $resultData["magnetometer"]
+ $resultData["compass"]
+ $resultData["battery"];
if ($security_risk < 1){
$msg = '<p style="font-weight:bold;">ПОЗДРАВЛЕНИЯ!<br>
  Вие използвате сигурен браузър.</p>';
} else {
  $msg = '<p style="font-weight: bold; color:
red;">ВНИМАНИЕ!<br>
  Установени са рискове за сигурността!<br>
  Обмислете смяна на браузъра, който използвате!</
p>';
}
// Добавяне на съобщението за нивото на сигурност към
протокола
fwrite($file, $msg);

// Затваряне на HTML документа
fwrite($file, `
  </body>
  </html>
`);

// Затваряне на файла
fclose($file);

// Добавяне на данни към файла с текстовите резултати
// $keys = array_keys($dataArray);
$values = array_values($dataArray);
$fp = fopen('reports/results.csv', 'a');
$delimiter = '|';
// fputcsv($fp, $keys, $delimiter);
fputcsv($fp, $values, $delimiter);
fclose($fp);

```

```
// За по-добра четимост на файла с логическите  
резултати
```

```
$resultData["Device"]=$dataArray["Device"];  
$resultData["OS"]=$dataArray["OS"];  
$resultData["Browser"]=$dataArray["Browser"];  
$resultData["devInfo"]=$dataArray["devInfo"];  
$resultData["startTime"]=$dataArray["startTime"];  
$resultData["endTime"]=$dataArray["endTime"];
```

```
// Добавяне на данни към файла с логическите резултати
```

```
// $keys = array_keys($resultData);  
$values = array_values($resultData);  
$fp = fopen('reports/logical_results.csv', 'a');  
$delimiter = '|';  
// fputcsv($fp, $keys, $delimiter);  
fputcsv($fp, $values, $delimiter);  
fclose($fp);
```

```
echo 'Html report done';
```

```
?>
```

```
-----
```

```
testDone.php
```

```
<?php
```

```
    session_start();  
    $_SESSION['testsDone']=true;
```

```
?>
```

```
sessionClose.php
```

```
<?php
```

```
// Стартиране на сесията  
session_start();
```

```
// Изчистване на данните в сесията
```

```
$_SESSION = array();
```

```
// Затваряне на сесията (прекратяване на текущата сесия)
session_destroy();
?>
```

dataCollector.php

```
<?php
// PHP script for storing data in a report file
session_start();
if (isset($_SESSION["reportFile"])) {
    $reportName = $_SESSION["reportFile"];
} else {
    $reportName="report.txt";
}
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $data = file_get_contents('php://input');
    if ($data !== null){
        file_put_contents($reportName, $data . "\n", FILE_
APPEND);
    }
    else {
        file_put_contents($reportName, "Null Data" .date("Y-
m-d-H-i-s"). "\n", FILE_APPEND);
    }
}
?>
```

createReportFiles.php

```
<?php
session_start();
$SID=session_id();
$reportsDir="reports";
$htmlReports="htmlreports";
$fname="reportCount.txt";
$leadingZeros = 5; // Брой водещи нули
// Прочитане на текущата стойност за брой на отчетите
(ако съществува)
```

```

    if (file_exists($fname)) {
        $num = intval(file_get_contents($fname));
    } else {
        $num = 0; // Инициализация с нула, ако файлът не
съществува
    }
    // Увеличаване на стойността с единица за да се
получи ново име на файл
    $num++;
    // Запис на новата стойност във файла, презаписвайки
старата
    file_put_contents($fname, $num);

    // Генериране на име на файла за конкретния отчет
    $dateYmd=date("Y-m-d-").str_pad($num, $leadingZeros,
'0', STR_PAD_LEFT);
    $_SESSION["reportFile"]="./".$reportsDir."/report-
'".$SID.'_'. $dateYmd.'.txt';
    $_SESSION["reportFileHTML"]="./".$htmlReports."/
report-'".$SID.'_'. $dateYmd.'.html';
    // Създаване на празни файлове с така генерираните
имена
    file_put_contents($_SESSION["reportFile"], '');
    file_put_contents($_SESSION["reportFileHTML"], '');

?>

```

```

async function init() {
    // Изключване на бутона
    const button=document.getElementById("doTestButton");
    button.disabled = true;

    // Извеждана на елемента userAgent от navigator
    document.getElementById("line1").innerHTML =
'<span>Вашето устройство: <br>' + navigator.userAgent +
'</span>';

```

```

// Стартиране на doTest
await doTest();

// Отбелязване, че тестовете са изпълнени и генериране
на HTML протокол
await testDone();
await generateHTMLReport();

// Показване на линка за изтегляне на отчета
const link = document.getElementById("report");
link.style.display = "inline"; // Променя стила на
линка и по този начин го прави видим `;

}

async function doTest() {
    var blink='<span style="animation: blink 1s
infinite;">.</span>'
    // Добавяне на начален час на теста към страницата и
към файла
    let startTime = new Date();
    await addToReport({"type":"startTime", "data":
startTime.toLocaleString()});
    document.getElementById("start").textContent = "Test
started at: " + startTime.toLocaleString();

    // Изпълняване на отделните стъпки от теста
await deviceInfo();

    infoMsg('Стартиран опит за достъп до вибрацията..');
await probeVibrate();

    infoMsg('Стартиран опит за достъп до сензора за
ориентация..');

```

```
await getOrientation();

    infoMsg('Стартиран опит за достъп до сензора за
ускорение..');
    await getAcceleration();

    infoMsg('Стартиран опит за достъп до сензора за
честота на завъртане..');
    await getRotationRate();

    infoMsg('Стартиран опит за достъп до сензора за
близост..');
    await getProximity();

    infoMsg('Стартиран опит за достъп до сензора за
осветеност..');
    await getLight();

    infoMsg('Стартиран опит за достъп до
магнитометър..');
    await getMagnetometer();

    infoMsg('Стартиран опит за достъп до компас..');
    await getCompass();

    infoMsg('Стартиран опит за достъп до батерията..');
    await getBatteryLevel();

    infoMsg('Стартиран опит за достъп до GPS..');
    await getGeolocation();

    infoMsg('Стартиран опит за достъп до камера..');
    await getPic();

    infoMsg('Стартиран опит за достъп до микрофон..');
    await getAudio();
```

```

    infoMsg('');
    document.getElementById("info").textContent = "";
    // Добавяне на крайни дата и час към страницата и към
    файла с резултатите
    let endTime = new Date();
    await addToReport({"type":"endTime", "data": endTime.
toLocaleDateString()});
    document.getElementById("end").textContent = "Test
ended at: " + endTime.toLocaleDateString();
}

async function deviceInfo() {
    let data=navigator.userAgent;
    await addToReport({"type":"devInfo", "data": data});
}

async function generateHTMLReport() {
    try {
        const response = await fetch('generateHTMLReport.
php');

        // Проверка на HTTP статус
        if (response.ok) {
            const data = await response.text(); // Извличане
на текстовата информация от отговора
            console.log('generateHTMLReport.php върна следните
данни:', data);
        } else {
            throw new Error('Грешка при изпращане на заявката');
        }
    } catch (error) {
        console.error('Грешка при изпълнение на
generateHTMLReport:', response.toString());
    }
}

```

```

async function getAcceleration() {
    var logical=false;
    var result='';
    try {
        const accelerationData = await new Promise((resolve,
reject) => {
            setTimeout(() => {
                reject(new Error('Timeout: Няма данни за
ускорение'));
            }, 3000); // Изчаква до 3 секунди за данни

            window.addEventListener('devicemotion', (event)
=> {
                resolve(event);
            }, { once: true });
        });

        // Проверка за наличието на данни и добавяне на
резултата към отчета
        if (accelerationData !== undefined){
            const accelerationX = accelerationData.
acceleration.x;
            const accelerationY = accelerationData.
acceleration.y;
            const accelerationZ = accelerationData.
acceleration.z;
            result="accelerationX: "+accelerationX+",
accelerationY: "+accelerationY+", accelerationZ:
"+accelerationZ;
            logical=true;
        } else {
            result="Няма данни от сензора за ускорение";
        }
    } catch (error) {
        result = error.toString();
    } finally {

```

```

    await addToReport({ "type": "acceleration", "data":
result, "logical": logical });
    }
}

async function getAudio() {
    var logical=false;
    var result='';
    try {
        // Вземаме достъп до микрофона
        const stream = await navigator.mediaDevices.
getUserMedia({ audio: true });

        // Създаваме AudioContext и MediaRecorder
        const audioContext = new AudioContext();
        const mediaRecorder = new MediaRecorder(stream);
        const chunks = [];

        mediaRecorder.ondataavailable = (e) => {
            if (e.data.size > 0) {
                chunks.push(e.data);
            }
        };

        // Начало на записа
        mediaRecorder.start();

        // Изчакваме 3 секунди, след което спираме записа
        await new Promise(resolve => setTimeout(() => {
            mediaRecorder.stop();
            stream.getTracks().forEach(track => track.
stop()));
            resolve();
        }, 3000));

        mediaRecorder.onstop = async () => {

```

```

// Изчакваме успешното спиране на записа
await new Promise(resolve => {
    result = "Успешен достъп до микрофона";
    logical = true;
    resolve();
});

// Създаваме аудио файл от записаните данни
const audioBlob = new Blob(chunks, { type:
'audio/wav' });
const audioUrl = URL.createObjectURL(audioBlob);

// Зареждаме аудио файла в тага с указаното id
const audioElement = document.
getElementById('capturedAudio');
audioElement.src = audioUrl;
audioElement.style.display = 'block'; //
Показване на аудиото
// Добавяме резултата към report
await addToReport({ "type": "audio", "data":
result, "logical":logical });

};

} catch (error) {
    logical=false;
    result = `Грешка при запис на аудио: ` + error.
toString();
    await addToReport({ "type": "audio", "data":
result, "logical":logical });
}
}

async function getBatteryLevel() {
    var logical = false;
    var result = '';

```

```

    try {
        const batteryLevel = await new Promise(async
(resolve, reject) => {
            setTimeout(() => {
                reject(new Error('Timeout: Няма данни за
батерията'));
            }, 3000); // Изчаква до 3 секунди за данни
за батерията
            try {
                const battery = await navigator.getBattery();
                resolve(battery.level * 100);
            } catch (error) {
                reject(error, 'Грешка при опит за получаване на
данни за батерията');
            }
        });

        if (batteryLevel !== undefined) {
            result = `Ниво на батерията: ` + batteryLevel +
` %`;
            logical = true;
        } else {
            result = "Няма данни за батерията";
        }
    } catch (error) {
        result = error.toString();
    } finally {
        await addToReport({ "type": "battery", "data":
result, "logical": logical });
    }
}

async function getCompass() {
    var logical = false;
    var result = '';
    try {

```

```

const sensor = new AbsoluteOrientationSensor();

const heading = await new Promise(async (resolve,
reject) => {
  setTimeout(() => {
    reject(new Error('Timeout: Липсват данни за
компаса.'));
  }, 3000); //Изчакваме 3 секунди и след това reject
при липса на данни
  sensor.onreading = () => {
    // Получаваме кватерниони от сензора
    const q = sensor.quaternion;

    // Изчисляваме heading използвайки формулата
    const heading = Math.atan2(2 * q[0] * q[1] + 2
* q[2] * q[3], 1 - 2 * q[1] * q[1] - 2 * q[2] * q[2]) *
(180 / Math.PI);

    // Коригираме heading, ако е необходимо
    const headingAdjusted = 270 + heading;
    resolve(headingAdjusted);
  };

  sensor.onerror = event => {
    reject(event.error.name);
  };

  sensor.start();
});

// След получаване на посоката
if (heading !== undefined) {
  result = `Посоката на компаса е: ` + heading + `
градуса`;
  logical = true;
} else {

```

```

        result = "Няма данни от сензора за компас";
    }
} catch (error) {
    result = error.toString();
} finally {
    await addToReport({ "type": "compass", "data":
result, "logical": logical });
}
}

```

```

async function getGeolocation() {
    var logical=false;
    var result='';
    try {
        const position = await new Promise((resolve, reject)
=> {
            setTimeout(() => {
                reject(new Error('Timeout: Няма данни за
геолокацията'));
            }, 3000); // Изчаква до 3 секунди за данни
            navigator.geolocation.getCurrentPosition(resolve,
reject);
        });

        if (position !== undefined) {
            const latitude = position.coords.latitude;
            const longitude = position.coords.longitude;
            result = `Вашиятe координати са: Ширинa `+ latitude
+ ` Дължина ` + longitude;
            logical=true;
        } else {
            result = "Няма данни за геолокацията";
        }
    } catch (error) {
        result = error.toString();
    } finally {

```

```

        await addToReport({ "type": "geolocation", "data":
result, "logical": logical });
    }
}

async function getLight() {
    var logical = false;
    var result = '';
    try {
        const lightData = await new Promise((resolve,
reject) => {
            setTimeout(() => {
                reject(new Error('Timeout: Няма данни за
осветеност'));
            }, 3000); // Изчаква до 3 секунди за данни за
осветеност
            // Works with new interface
            if ('AmbientLightSensor' in window) {
                try {
                    var sensor = new AmbientLightSensor();
                    sensor.addEventListener('reading', (event)
=> {
                        resolve(event);
                    });
                    sensor.start();
                } catch (error) {
                    reject(error, 'AmbientLightSensor: Грешка
при опит за получаване на данни за осветеност');
                } // new interface end
            } else { // Try with old interface
                try {
                    window.addEventListener('devicelight', (event)
=> {
                        resolve(event);
                    }, { once: true });
                } catch {

```

```

        reject(error, `devicelight: Грешка при опит
за получаване на данни за осветеност`);
    }
}
});

```

```

    if (lightData !== undefined) {
        const illuminance = lightData.value;
        result = `Интензитет на осветеност: `+ illuminance
+' lux`;
        logical = true;
    } else {
        result = `Няма данни за осветеност, lightData
undefined`;
    }
} catch (error) {
    result = error.toString();
} finally {
    await addToReport({ `type`: `light`, `data`:
result, `logical`: logical });
}
}

```

```

async function getMagnetometer() {
    var logical = false;
    var result = ``;
    try {
        const magnetometerData = await new Promise((resolve,
reject) => {
            setTimeout(() => {
                reject(new Error(`Time out: Няма данни от
магнитометъра`));
            }, 3000); // Изчаква до 3 секунди за данни

            if (`Magnetometer` in window) {
                try {

```

```

        var sensor = new Magnetometer({ frequency: 60
    });
    sensor.addEventListener('reading', (event) => {
        resolve(event);
    });
    sensor.start();
    } catch (error) {
        reject(error, 'Magnetometer: Грешка при опит
за получаване на данни от магнитрометъра');
    } // new interface end
    } else {
        try {
            const sensor = new AbsoluteOrientationSensor();

            sensor.addEventListener('reading', () => {
                // Access magnetometer data
                const magnetometerData = {
                    x: sensor.magnetometer.x,
                    y: sensor.magnetometer.y,
                    z: sensor.magnetometer.z
                };
                resolve(magnetometerData);
            });
            sensor.start();
        } catch (error) {
            reject(error, 'Magnetometer(A): Грешка при опит
за получаване на данни от магнитрометъра');
        }
    }
    });
    if (magnetometerData !== undefined) {
        const x = magnetometerData.x;
        const y = magnetometerData.y;
        const z = magnetometerData.z;
        result = `Магнитрометър X: ` + x + `, Y: ` + y +
`, Z: ` + z;

```

```

        logical = true;
    } else {
        result = "Няма да данни от магнитометъра";
    }
} catch (error) {
    result = error.toString();
} finally {
    await addToReport({ "type": "magnetometer", "data":
result, "logical": logical });
}
}

```

```

async function getOrientation() {
    var logical=false;
    var result='';
    try {
        // Listen for deviceorientation event
        const orientationData = await new Promise((resolve,
reject) => {
            setTimeout(() => {
                reject(new Error('Няма данни за ориентация.'));
            }, 3000); // Изчаква до 3 секунди за данни
            if ('AbsoluteOrientationSensor' in window) {
                try {
                    var sensor = new AbsoluteOrientationSensor();
                    sensor.addEventListener('reading', (event)
=> {
                        resolve(event);
                    });
                    sensor.start();
                } catch (error) {
                    reject(error, 'AbsoluteOrientationSensor:
Грешка при опит за получаване на данни за ориентация');
                } // new interface end
            } else { // Try with old interface
                try {

```

```

        window.addEventListener('deviceorientation',
(event) => {
            resolve(event);
            }, { once: true });
        } catch (error) {
            reject (error, 'Грешка при опит за получаване
на данни за ориентация');
        }
    }
});
// Проверка за наличието на данни и добавяне на
резултата към отчета
if (orientationData !== undefined){
    const alpha = orientationData.alpha;
    const beta = orientationData.beta;
    const gamma = orientationData.gamma;
    result="alpha: "+alpha+", beta: "+beta+ ", gamma:
"+gamma;
    logical=true;
} else {
    result="Няма данни за ориентация, orientationData
undefined";
}
} catch (error) {
    result = error.toString();
} finally {
    await addToReport({ "type": "orientation", "data":
result, "logical": logical });
}
}

async function getPic() {
    var logical=false;
    var result='';
    try {
        // Вземаме достъп до устройствата

```

```

        const stream = await navigator.mediaDevices.
getUserMedia({ video: true });

        // Връщаме потоковата информация към елемента
<video>
const videoElement = document.createElement('video');
document.body.appendChild(videoElement);
videoElement.srcObject = stream;
videoElement.play();

        // Изчакваме малко време, за да видим изображението
await new Promise(resolve => setTimeout(resolve,
3000));

        // Създаваме елемент canvas и копираме изображението
в него
const canvas = document.createElement('canvas');
const context = canvas.getContext('2d');
canvas.width = videoElement.videoWidth;
canvas.height = videoElement.videoHeight;
context.drawImage(videoElement, 0, 0, canvas.width,
canvas.height);

        // Край на заснемането
stream.getTracks().forEach(track => track.stop());
document.body.removeChild(videoElement);

        // Преобразуваме изображението в Data URL
const imageDataUrl = canvas.toDataURL('image/jpeg');
// Проверка за успешно създаване на изображение
if (imageDataUrl !== undefined) {

                const imageElement = document.
getElementById('capturedImage');

        // Задаваме на src на елемента и показване на

```

изображението

```
        imageElement.src = imageDataUrl;
        imageElement.style.display = 'block';
        result = "Успешен достъп до камерата";
        logical = true;
    } else {
        result = "Неуспешен опит за достъп до камерата,
imageDataUrl undefined";
    }
} catch (error) {
    result = `Грешка при заснемане на изображение: `
+ error.toString();
} finally {
    await addToReport({ "type": "image", "data":
result, "logical": logical });
}
}
```

```
async function getProximity() {
    var logical = false;
    var result = '';
    try {
        const proximityData = await new Promise((resolve,
reject) => {
            setTimeout(() => {
                reject(new Error(`Няма данни за близост`));
            }, 3000); // Изчаква до 3 секунди за данни за
близост
            if ('ProximitySensor' in window) {
                try {
                    var sensor = new ProximitySensor();
                    sensor.addEventListener('reading', (event)
=> {
                        resolve(event);
                    });
                    sensor.start();
                }
            }
        });
    }
}
```

```

        } catch (error) {
            reject(error, 'ProximitySensor: Грешка при
опит за получаване на данни за близост');
        } // new interface end
    } else {
        try {
            window.addEventListener('deviceproximity',
(event) => {
                resolve(event);
            }, { once: true });
        } catch {
            reject(error, 'Грешка при опит за получаване
на данни за близост');
        }
    }
});

    if (proximityData !== undefined) {
        const minDistance = proximityData.min;
        const maxDistance = proximityData.max;
        const value = proximityData.value;
        result = 'Мин. разстояние: ' + minDistance + '
Макс. разстояние: ' + maxDistance + ', Текущо разстояние:
' + value;
        logical = true;
    } else {
        result = "Няма данни за близост, proximityData
undefined";
    }
} catch (error) {
    result = error.toString();
} finally {
    await addToReport({ "type": "proximity", "data":
result, "logical": logical });
}
}

```

```

async function getRotationRate() {
  var logical = false;
  var result='';
  try {
    const rotationData = await new Promise((resolve,
reject) => {
      setTimeout(() => {
        reject(new Error('Timeout: Няма данни за
rotationRate'));
      }, 3000); // Изчаква до 3 секунди за данни за
rotationRate

      try {
        window.addEventListener('devicemotion', (event)
=> {
          if ('rotationRate' in event) {
            resolve(event.rotationRate);
          } else {
            reject(new Error('Няма данни за rotationRate'));
          }
        }, { once: true });
      } catch {
        reject(error, 'Грешка при опит за получаване на
данни за rotationRate');
      }
    });

    if (rotationData!== undefined) {
      const alpha = rotationData.alpha;
      const beta = rotationData.beta;
      const gamma = rotationData.gamma;
      result = "alpha: " + alpha + ", beta: " + beta +
", gamma: " + gamma;
      logical = true;
    } else {

```

```

        result = "Няма данни за rotationRate, rotationData
undefined";
    }
    } catch (error) {
        result = error.toString();
    } finally {
        await addToReport({ "type": "rotationRate",
"data": result, "logical": logical });
    }
}

function infoMsg(msg) {
    if (msg){
        document.getElementById("info").innerHTML =
'<span>'+msg+'</span><span class="blink">.</span>';
    } else {
        document.getElementById("info").innerHTML = '';
    }
}

// Function to vibrate the device
async function probeVibrate() {
    var logical=false;
    var result='';
    if (/Mobi|Android|iOS/i.test(navigator.userAgent))
{
    if ('vibrate' in navigator) {
        try {
            await navigator.vibrate(300); // Опитваме се
да вибрираме устройството за кратко време.
            logical = true; // Вибрацията беше успешна,
значи имаме достъп.
            result = "Успешен достъп до вибрацията";
        } catch (error) {
            logical = false; // Вибрацията не може да
бъде стартирана.

```

```

        result = "Вибрацията не може да бъде
стартирана";
    }
    } else {
        logical = false; // Устройството не поддържа
вибрация.
        result = "Устройството не поддържа вибрация";
    }
    } else {
        logical = false;
        result = "Устройството не е мобилно";
    }
    await addToReport({"type":"vibration", "data":
result, "logical":logical});
}

```

```

async function testDone() {
    try {
        const response = await fetch('testDone.php');

        // Проверка на успешния HTTP статус (например, 200
за успешно)
        if (response.ok) {
            const data = await response.text(); // Извличане
на текстовата информация от отговора
            console.log('testDone.php върна следните данни:',
data);
        } else {
            throw new Error('Грешка при изпращане на заявката');
        }
    } catch (error) {
        console.error('Грешка при изпълнение на testDone:',
error);
    }
}

```

```

}

// Функция addToReport(data) приема като параметър JSON
обект
async function addToReport(data) {
  console.log(data);
  try {
    const response = await fetch('dataCollector.php', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(data),
    });
    /* За диагностични съобщения
    if (response.ok) {
      console.log('Данните бяха успешно изпратени.');
```

```

    } else {
      console.error('Грешка при изпращането на данните.');
```

```

    }
    */
  } catch (error) {
    console.error('Грешка: ', error);
  }
}

```

Приложение 3. Пример за протокол от провежданите тестове, в които са открити уязвимости в браузъра

Протокол № 142

Начало на теста: 11/26/2023, 8:38:32 PM

Край на теста: 11/26/2023, 8:38:50 PM

Изследвано устройство:

Операционна система:

Браузър:

Информация за устройството: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Mobile Safari/537.36

Достъп до вибрацията¹: Успешен достъп до вибрацията

Достъп до сензора за ориентацията на устройството в пространството: alpha: undefined, beta: undefined, gamma: undefined

Достъп до сензора за ускорение: accelerationX: 0, accelerationY: 0, accelerationZ: 0

Достъп до честота на завъртане: alpha: 0, beta: 0, gamma: 0.1

Достъп до сензора за близост: Еггор: Няма данни за близост

Достъп до сензора за осветеност: Еггор: Timeout: Няма данни за осветеност

Достъп до магнитометър: Еггор: Time out: Няма данни от магнитометъра

Достъп до компас: Посоката на компаса е: 315.78707556835076 градуса

Достъп до батерията: Ниво на батерията: 100 %

Достъп до сензора за геолокация: Еггор: Timeout: Няма данни за геолокацията

Достъп до камерата²: Грешка при заснемане на изображение: NotAllowedError: Permission denied

Достъп до микрофона³: Грешка при запис на аудио: NotAllowedError: Permission denied

1. Тестът за вибрация дава фалшиво положителни резултати. Достъпът е успешен само в случай, че устройството е започнало да вибрира по време на теста.

2. Риск за сигурността има, ако достъпът до камерата е успешен без браузърът да е поискал разрешение за достъп.

3. Риск за сигурността има, ако достъпът до микрофона е успешен, без браузърът да е поискал разрешение за достъп.

Прехващането на звук продължава дори когато браузърът работи във фонов режим.

ВНИМАНИЕ!

Установени са рискове за сигурността!

Обмислете смяна на браузъра, който използвате!

Приложение 4. Пример за протокол от провежданите тестове, в които не са открити уязвимости в брауъра

Протокол № 143

Начало на теста: 26/11/2023, 19:41:46

Край на теста: 26/11/2023, 19:42:17

Изследвано устройство:

Операционна система:

Брауър:

Информация за устройството: Mozilla/5.0 (iPhone; CPU iPhone OS 14_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0 Mobile/15E148 Safari/604.1

Достъп до вибрацията¹: Устройството не поддържа вибрация

Достъп до сензора за ориентацията на устройството в пространството: Error: Няма данни за ориентация.

Достъп до сензора за ускорение: Error: Timeout: Няма данни за ускорение

Достъп до честота на завъртане: Error: Timeout: Няма данни за rotationRate

Достъп до сензора за близост: Error: Няма данни за близост

Достъп до сензора за осветеност: Error: Timeout: Няма данни за осветеност

Достъп до магнитометър: ReferenceError: Can't find variable: AbsoluteOrientationSensor

Достъп до компас: ReferenceError: Can't find variable: AbsoluteOrientationSensor

Достъп до батерията: TypeError: navigator['getBattery'] is not a function. (In 'navigator['getBattery']()', 'navigator['getBattery']' is undefined)

Достъп до сензора за геолокация: [object GeolocationPositionError]

Достъп до камерата²: Успешен достъп до камерата

Достъп до микрофона³: Грешка при запис на аудио: NotAllowedError: The request is not allowed by the user agent or the platform in the current context, possibly because the user denied permission.

1. Тестът за вибрация дава фалшиво положителни резултати. Достъпът е успешен само в случай, че устройството е започнало да вибрира по време на теста.

2. Риск за сигурността има, ако достъпът до камерата е успешен без брауърът да е поискал разрешение за достъп.

3. Риск за сигурността има, ако достъпът до микрофона е успешен, без брауърът да е поискал разрешение за достъп.

Прехващането на звук продължава дори когато брауърът работи във фонов режим.

ПОЗДРАВЛЕНИЯ!

Вие използвате сигурен брауър.

Стоян Т. Мечев

**МЕТОДИ ЗА ПОВИШАВАНЕ НА СИГУРНОСТТА
НА ОПЕРАЦИОННИ СИСТЕМИ
ЗА МОБИЛНИ УСТРОЙСТВА**

Книга по дисертация

Българска
Първо издание

Научни рецензенти:

Проф. д-р **Юлиян Иванов Цонев**,
Висше военноморско училище „Никола Й. Вапцаров“
Проф. д-р **Тодор Димитров Ганчев**,
Технически университет – Варна

Редактор и коректор: д-р Ваня Колева Колева

Подготвена за печат: 8.09.2025 г.
Предпечатна подготовка: „Етикет принт“ ЕООД

ISBN 978-619-7752-13-7 (e-book)
DOI <https://doi.org/10.63662/diss-books/mechev-2025>

Висше военноморско училище „Никола Й. Вапцаров“, 9002, Варна