УДК 004[056.53+413.4]

DOI: 10.18372/2073-4751.72.17455

[1]Антонішин М.В.,
orcid.org/0000-0002-2665-0066,
[1]Міснік О.І.,
orcid.org/0000-0002-4654-9125,
[1,2]Цуркан В.В., к.т.н.,
orcid.org/0000-0003-1352-042X

СПОСОБИ ТЕСТУВАННЯ УРАЗЛИВОСТЕЙ МОБІЛЬНИХ ПРОГРАМНИХ ЗАСТОСУНКІВ

[1]Інститут проблем моделювання в енергетиці імені Г.Є. Пухова НАН України
[2]Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

antonishin.mihail@gmail.com,
oleksii.misnik@gmail.com,
v.v.tsurkan@gmail.com

Вступ

Тестування на уразливості проводиться для перевірки мобільних програмних застосунків щодо збереженості властивостей інформації, яка ними обробляється. Серед них насамперед виокремлюються конфіденційність, цілісність та доступність. З огляду на це більшість організацій зі стандартизації та сертифікації орієнтовані на рекомендування і застосування власних способів тестування уразливостей. Однак, незважаючи на це, основою використання кожного з них є настанови відкритого проєкту забезпечення безпеки вебзастосунків (Open Worldwide Application Security Project, OWASP) [1-3], а саме [4,5]: керівництва з тестування безпеки мобільних програмних застосунків (Mobile Application Security Testing Guide, OWASP MSTG), стандарту верифікування безпеки мобільних програмних застосунків (Mobile Application Security Verification Standard, OWASP MASVS). За ними тестування уразливостей зводиться до застосування одного з способів (або їх поєднань) – статичного та динамічного. Статичне тестування проводиться шляхом сканування або мобільних програмних застосунків, або вивчення файлів, або після їх встановлення збережених параметрів рядків. Характерною особливістю даного способу є застосовність, по-перше, без виконання мобільних застосунків. По-друге, для попереднього їх аналізування та накопичення базової інформації. Це потрібно перш за все для визначення сценаріїв тестування уразливостей мобільних програмних застосунків. Тоді як на основі отриманої інформації динамічним способом верифікуються виявленні можливі загрози [4-6]. Крім того, виконується аналіз уразливостей, які можуть з'являтися при функціонуванні мобільних програмних застосунків у конкретному середовищі. Однак, виконання деяких сценаріїв тестування уразливостей на практиці обмежується неоднозначною належністю до статичного або динамічного способів (наприклад [4], експортованих компонентів, MSTG-STORAGE-6; нестандартних сертифікатів, сертифікату SSL-пінінгу, MSTG-NETWORK-4).

Отже, аналізування застосовності способів тестування уразливостей мобільних програмних застосунків є актуальним завданням.

Результати існуючих досліджень тестування уразливостей мобільних програмних застосунків або близьких до них розкрито в [7-15]. Забезпеченню конфіденційності інформації приділено увагу в [7]. Проаналізовано сценарії тестування безпеки мобільних програмних застосунків. Забезпечення їх повноти досягнуто поєднання настанов OWASP і Національного інституту стандартів і технологій (National

Institute of Standards and Technology, NIST). До того ж врахуванням положень Альянсу забезпечення безпеки хмари (Cloud Security Alliance, CSA). Це дозволило сформулювати рекомендації для зменшення витоку конфіденційної інформації.

Інтегрування гнучкості та забезпечення безпеки програмних застосунків досліджено в [8]. Це важливо перш за все для протидії атакам критично важливих активів. Серед способів інтегрування виокремлено узгодженість виявлення уразливостей на стадії упровадження програмних застосунків і використання методів рефакторингу. Тож для дослідження ефективності інструментальних засобів статичного тестування і їх важливості використано кількісні та якісні підходи. Як наслідок, показано існування обмежень щодо ефективності та сприйняття розробниками.

Проблему забезпечення безпеки програмних застосунків критичного значення описано в [9]. Результативність виконання даного завдання обумовлено дотриманням принципів і методів його розроблення. З огляду на це проаналізовано чотири категорії способів забезпечення безпеки програмних застосунків. Насамперед статичний і динамічний аналізи, формальні методи та адаптивні механізми. Встановлено переваги та недоліки їхнього використання на практиці. Продемонстровано застосовність кожного зі способів.

На важливості потреби забезпечення безпеки мобільних програмних застосунків акцентовано увагу в [10]. Для її задоволення запропоновано використання відповідних фреймворків. Цьому передувало проведення систематичного огляду літератури. За його основу взято протокол дослідження з виокремленням і тлумаченням інформації за напрямом забезпечення безпеки мобільних програмних застосунків. Використання такого підходу дозволило встановити найбільш актуальні проблеми та серед них зосередитися на аутентифікації і авторизації.

Використання Burp Suite для тестування мобільних програмних застосунків викладено в [11]. За основу такого застосування взято його реалізування як HTTP-проксі. Цим обумовлюється універсальність Burp Suite щодо тестування безпеки програмних застосунків на будь-якому мобільному пристрої. Передумовами такої універсальності є їхня здатність до взаємодії через протокол HTTP або HTTPS. Завдяки цьому можливе направлення трафіку на Burp Suite.

Основні причини розроблення уразливого програмного забезпечення під керування операційної системи Android виокремлено в [12]. Насамперед неуважність до факторів забезпечення безпеки на етапі їхнього розроблення. Для подолання встановленої причини рекомендовано враховувати аспекти забезпечення безпеки мобільних програмних застосунків на кожній стадії життєвого циклу. Це досягнуто завдяки узагальненню поширених уразливостей і розробленням плагіну для Android Studio з підтриманням безпечного програмування.

Обмеження типових способів і засобів тестування уразливостей вебзастосунків встановлено в [13]. Їхнє подолання запропоновано досягти розробленням сканера вразливостей вебзастосунків. За його основу взято поєднання підходів "чорної" і "білої" скриньок. Завдяки такому поєднанню стало можливим зменшення тривалості сканування. Цим забезпечено підвищення ефективності даного процесу шляхом зменшення кількості хибних результатів [13,14]. Отримані результати продемонстровано на прикладі виявлення міжсайтових сценаріїв.

Складнощі тестування уразливостей програмних застосунків охарактеризовано в [15]. З цим пов'язується неефективність застосування відомих способів і засобів. Тому запропоновано приділити окрему увагу дослідженню зон уразливостей. Їх урахування дозволить підвищити рівень забезпеченості безпеки програмних застосунків. Насамперед на стадії інтелектуальної автоматизації прикладних завдань.

Мета

Метою роботи є встановлення особливостей застосування способів тестування уразливостей мобільних програмних застосунків.

Основна частина

Використання статичного та динамічного способів на практиці обмежується відсутністю відповідності сценаріїв тестування уразливостей мобільних програмних застосунків. Підтвердженням цьому є детальне їх ретроспективне аналізування. Насамперед вони реалізуються за декілька кроків. Ці кроки складно однозначно або визначити, або встановити залежність між ними.

Існування даних особливостей можна пояснити залежністю від інструментального засобу або дій фахівця. Здебільшого ним спочатку тестуються уразливості мобільного програмного застосунку статичним способом. Завдяки цьому формується вірогідний напрям виявлення уразливостей, наприклад, аналізування вихідного коду або конфігураційних файлів. Залежно від обраного напряму визначаються подальші дії. Насамперед розроблення власних модулів за умови їх відсутності серед відомих реалізацій окремих кроків статичного тестування уразливостей. Водночас така потреба може обумовлюватися орієнтованістю на забезпечення безпеки конкретного мобільного програмного застосунку.

До того ж прояв встановлених особливостей використання способів тестування уразливостей мобільних програмних застосунків спостерігається і для відповідних інструментальних засобів. Наприклад, Drozer позиціонується як програмна реалізація динамічного способу тестування уразливостей [16]. Однак, характерною особливістю його застосування на практиці є відсутність потреби в задоволенні вимог обов'язкового виконання мобільного програмного застосунку [5,6]. Натомість тестування його уразливостей інструментальним засобом MobSF може проводитися статичним і динамічним способами [5,17].

Тоді як практичне використання фреймворків Xposed та Frida зводиться до такої послідовності кроків [18,19]:

1. Сканувати код мобільного програмного застосунку статичним способом для виявлення потенційних уразливостей.

2. Сканувати код мобільного програмного застосунку динамічним способом для проведення дебагінгу, динамічного моніторингу виконання команд та запитів.

3. Розробити та запустити експлойт для тестування функціонування мобільного програмного застосунку.

4. Використати готовий модуль тестування уразливостей мобільного програмного застосунку.

Виокремлена послідовність кроків включає декілька різних способів тестування. При використанні Xposed застосовується модульна система. Вона активується даним фреймворком та запускається під час функціонування мобільного програмного застосунку. Фреймворком Frida реалізується набір різних сценаріїв тестування. Водночас використання деяких інструментальних засобів, наприклад, описаних в OWASP MSTG, характеризуються гнучкістю. Це дозволяє розробляти індивідуальні модулі тестування специфічного функціоналу мобільного програмного застосунку.

Розглянемо тестування уразливостей на прикладі обходження механізмів забезпечення безпеки. Зокрема, перевіримо запускання програмного застосунку на віртуальній машині або скомпрометованому тестовому мобільному пристрої.

Для цього використаємо фреймворк Frida шляхом виконання таких дій [19]:

1. Отримати APK-файл мобільного програмного застосунку шляхом використання спеціальних програм, наприклад ADB або APK-екстрактор.

2. Завантажити APK-файл в інструментальний засіб статичного аналізу, наприклад, MobSF [17]. На даному кроці реалізується статичне сканування мобільного програмного застосунку. Воно орієнтоване на виявлення механізму забезпечення його безпеки (рис. 1).

3. Розробити та запустити експлойт уразливостей мобільного програмного застосунку. Ним тестуватиметься обходження функціоналу механізму забезпечення безпеки. Це дозволить перевірити існування можливості виконання мобільного програмного застосунку на скомпрометованому мобільному пристрої. Як наслідок, підтвердити уразливість програмного застосунку, що тестується (рис. 2).

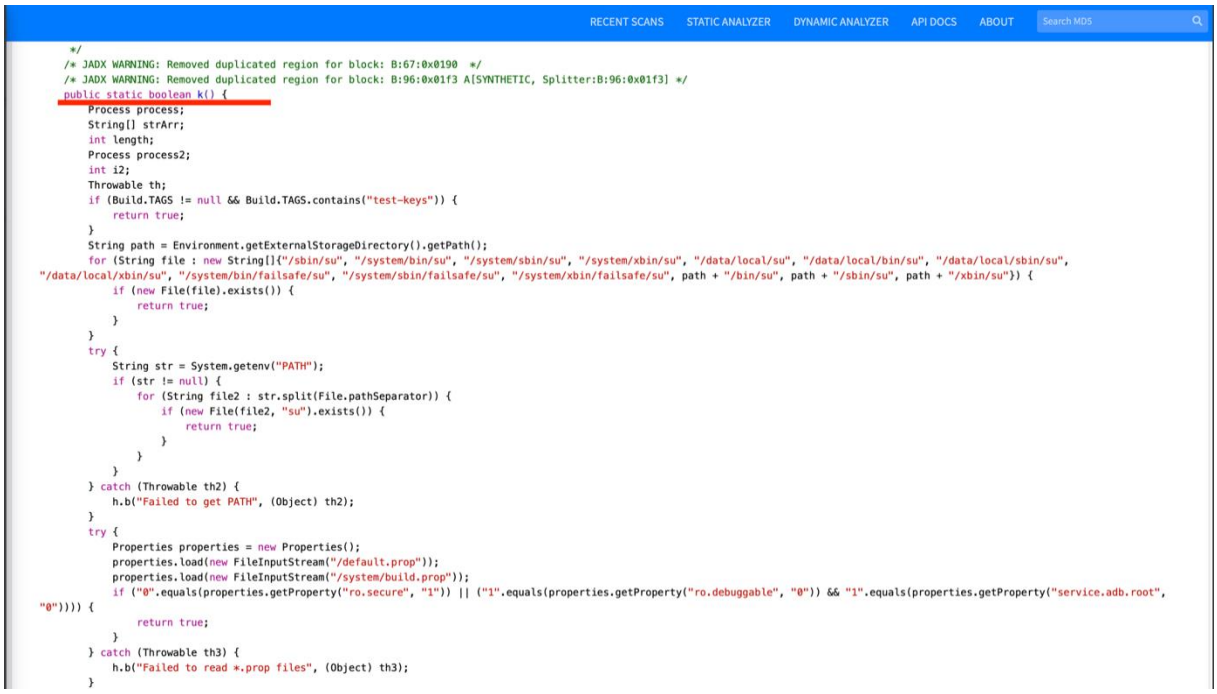

Рис. 1. Приклад отримання вихідної інформації для розроблення експлойту за результатами аналізування фрагменту коду тестового мобільного програмного застосунку

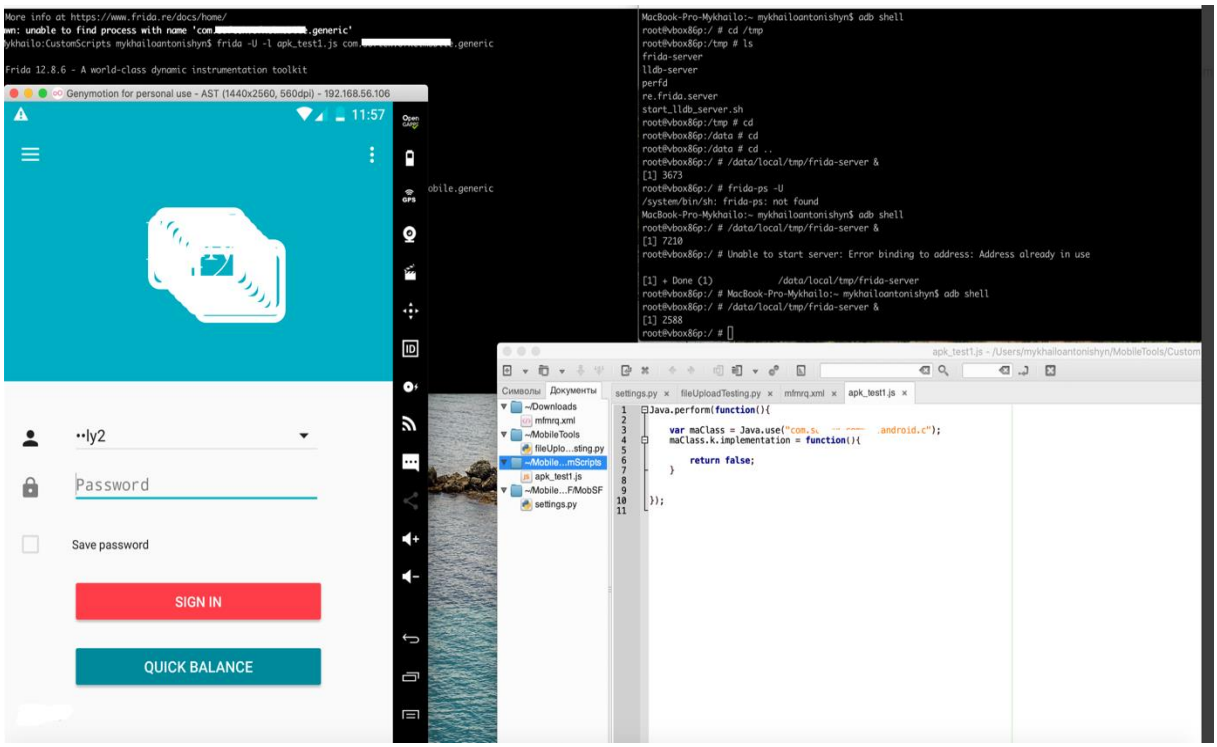

Рис. 2. Приклад розроблення та виконання експлойту вразливостей тестового мобільного програмного застосунку

Приділимо увагу тестуванню обходження механізму забезпечення безпеки мобільного програмного застосунку від перехоплення і дешифрування HTTP-трафіку, зокрема, SSL-пінінгу [20]. Для цього виокремлюється 4 варіанти сценаріїв коректності використання даного механізму. Реалізування останнього з них орієнтоване на застосування окремо розробленого програмного модулю, за допомогою якого тестується обходження стандартної реалізації механізму забезпечення безпеки з одним сертифікатом.

SSL-пінінг – це технологія, яка призначена для попередження атаки типу "людина посередині" підтвердженням довіреності SSL-сертифікату після початку з'єднання з сервером мобільного програмного застосунку. Відсутність такого механізму дозволить аналізувати та підроблювати HTTP-запити до серверу програмного застосунку. Тоді як перед перевірянням існування уразливостей необхідно налаштувати тестовий мобільний пристрій та проксі-сервер. Сконфігурувати їх так, щоб можливе було розшифровування запитів програмного застосунку. При цьому рекомендується дотримання такої послідовності кроків:

1. Згенерувати сертифікат на проксі-сервері для імпортування у тестовий мобільний пристрій або віртуальну машину.

2. Імпортувати сертифікат у тестовий мобільний пристрій. Залежно від версії платформи операційної системи Android це можна реалізувати двома різними шляхами:

по-перше, імпортування сертифікату безпосередньо в тестовий мобільний пристрій як користувацького. Однак, це можливо лише на версіях платформи до Android 7.0 включно. Внаслідок цього можливе компрометування мобільного програмного застосунку через його запускання на попередніх версіях зазначеної платформи;

по-друге, імпортування сертифікату в кореневі довірені сертифікати. Для цього генерується сертифікат та додається в мобільний пристрій скриптом

Для тестування уразливостей мобільних програмних застосунків необхідно виконати відповідну послідовність кроків:

1. Завантажити APK-файл в сканер мобільних програних застосунків MobSF через веб- або API-інтерфейс. Після цього в пункті "Файли" необхідно визначити наявність файлів з розширенням *.cer, *.der, *.pem та інших можливих сертифікатів.

2. Встановити на тестовий мобільний пристрій або віртуальну машину фреймворк Xposed. Ним запускатимуться спеціальні модулі для виконання сценаріїв тестування.

3. Встановити модулі тестування SSLUnpinning та JustTrustMe (рис. 3). Ними виконуються однакові завдання, але вони мають різний функціонал виконання завдань щодо обходження механізму забезпечення безпеки SSL-пінінг. Тому для різних мобільних програмних застосунків можливе використання різних модулів.

4. На цьому кроці необхідно визначити, який модуль буде використовуватися для реалізування наступних сценаріїв. Вони будуть виконуватися для тестування уразливостей серверу з яким взаємодіє мобільний програмний застосунок.

5. Перевірити модуль JustTrustMe. Якщо його не можливо застосувати до мобільного програмного застосунку, то наступним кроком необхідно перевірити модуль SSLUnpinning.

6. Перейти в модуль SSLUnpinning та виконати відключення механізму забезпечення безпеки SSL-пінініг. Якщо така дія вдала, то далі можна тестувати взаємодію с сервером програмного застосунку.

7. Перейти в програмний застосунок, який встановлено на мобільному пристрій або віртуальній машині. Після цього проаналізувати HTTP-запити за умови попереднього впровадження сертифікату проксі-серверу.

8. Перейти в програмний застосунок, який встановлений на мобільному пристрої та в який імпортовано сторонній сертифікат. За допомогою проксі отримати доступ до запитів до серверу програмного застосунку (рис. 4).

Виокремленими та дослідженими прикладами показано некоректність окремого застосування динамічного способу тестування. Це обумовлено комплексністю дій, зокрема, статичного сканування мобільного програмного застосунку та мануального аналізування вихідного коду фахівцем. І, насамперед, за необхідності розроблення окремих модулів тестування уразливостей.

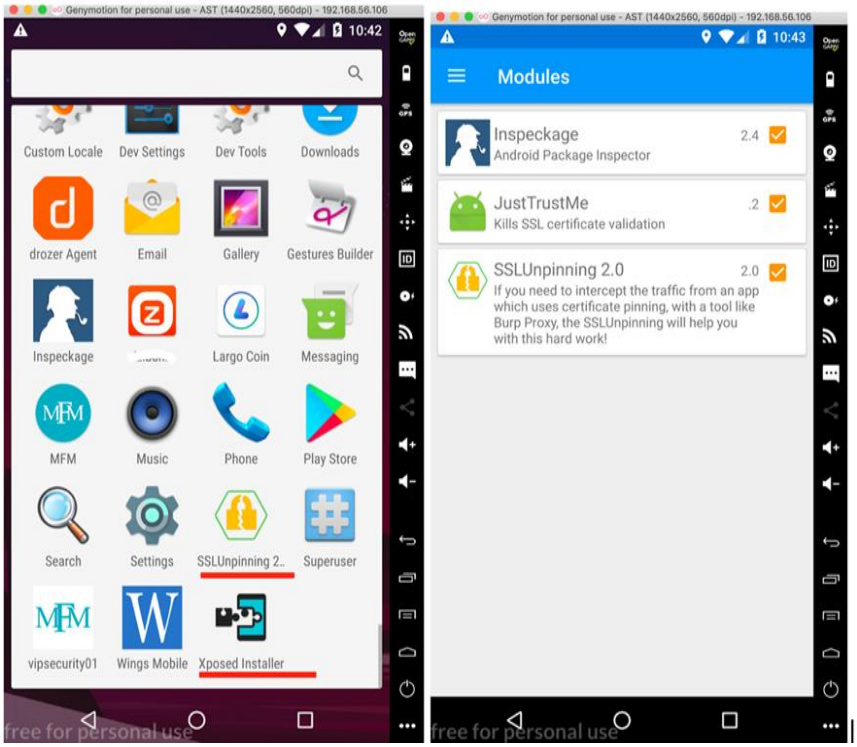

Рис. 3. Приклад розширення фреймворку Xposed мобільного пристрою модулями тестування

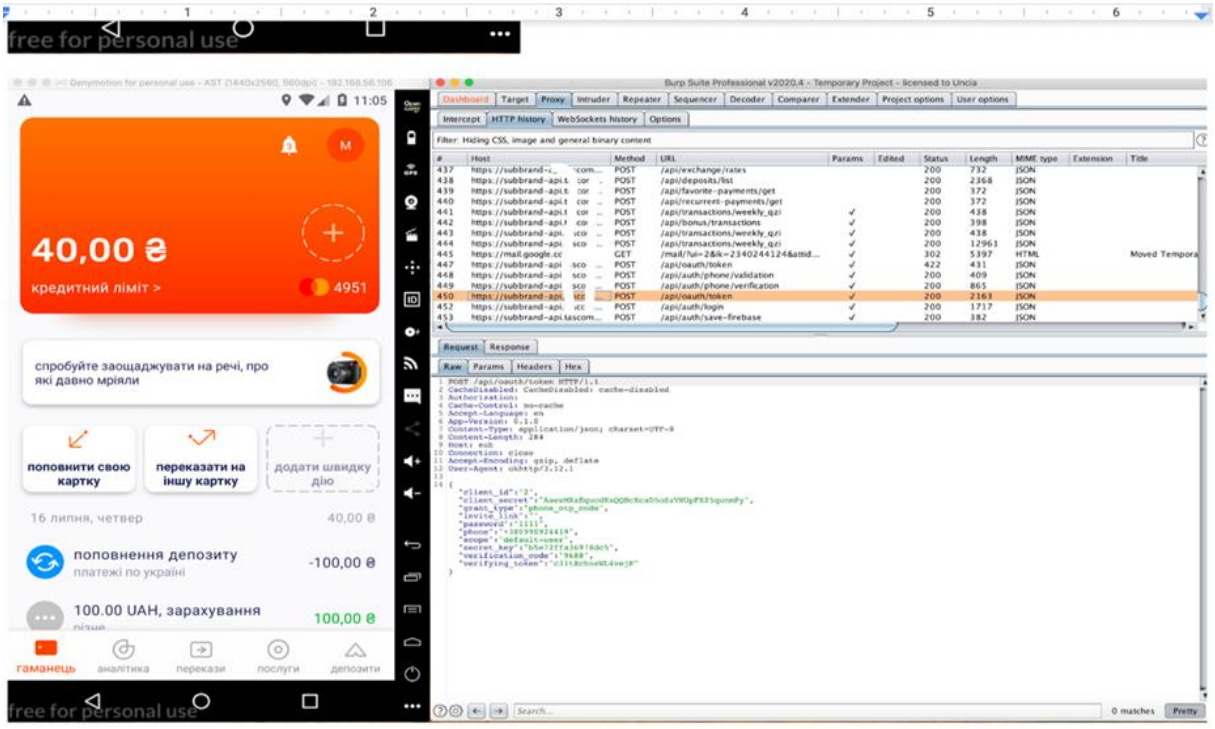

Рис. 4. Приклад виконання мобільного програмного застосунку при тестуванні обходження механізму забезпечення безпеки SSL-пінінг

Висновки

Отже, проаналізовано застосовність способів тестування уразливостей мобільних програмних застосунків. Показано орієнтованість даного процесу на настанови відкритого проєкту забезпечення безпеки вебзастосунків, OWASP. Серед способів тестування уразливостей виокремлено статичний і динамічний. Встановлено обмеженість їхнього застосування через неоднозначну належність сценаріїв до них, наприклад: експортованих компонентів, MSTG-STORAGE-6; нестандартних сертифікатів, сертифікату SSL-пінінгу, MSTG-NETWORK-4.

За результатами аналізування існуючих досліджень підтверджено їхню багатоаспектність. З одного боку, вони ґрунтуються на настановах відкритого проєкту забезпечення безпеки вебзастосунків, OWASP. З іншого – це дозволило сформулювати рекомендації для зменшення витоку конфіденційної інформації. Крім того зосереджено увагу на актуальності протидії атакам критично важливих активів завдяки інтегруванню гнучкості та безпечності розроблення програмних застосунків, а також використанню відповідних фреймворків. До того ж подоланням недоліків розроблених засобів тестування уразливостей.

Виокремленими та дослідженими прикладами продемонстровано обмеженість застосування статичного та динамічного способів. Перш за все через відсутність відповідності сценаріїв тестування уразливостей мобільних програмних застосунків. Це підтверджено складністю однозначного як визначення, так і виявлення залежності між ними. Водночас встановлено некоректність окремого застосування динамічного способу тестування. Це обумовлено комплексністю дій, зокрема, статичного сканування мобільного програмного застосунку та мануального аналізування вихідного коду фахівцем. І, насамперед, за необхідності, використання додаткових фреймворків, розроблення окремих модулів тестування уразливостей.

Література

1. *Антонішин М.В., Міснік О.І., Цуркан В.В.* Оцінювання стану захищеності програмних застосунків операційної системи Android за методологією OWASP Mobile TOP 10. Моделювання та інформаційні технології. – 2018. – Вип. 82. – С. 94-101.

2. *Antonishyn M., Misnik O.* Analysis of testing approaches to Android mobile application vulnerabilities. Information Technologies and Security. – Vol. 2577. – Aachen, Germany, 2019. – P. 270-280.

3. *Antonishyn M.* Mobile applications vulnerabilities testing model. Information Technology and Security. January - June 2020. – Vol. 8. – Iss. 1 (14). – P. 49-57.

4. OWASP Mobile Application Security Testing Guide. URL: https://github.com/OWASP/owasp-mstg/.

5. OWASP Mobile security verification standard. URL: https://github.com/OWASP/owasp-masvs/.

6. NIST 800-163. Vetting the Security of Mobile application. [Valid from 2019-04-19]. DOI: https://doi.org/10.6028/NIST.SP.800-163r1.

7. *Lin HY., Chang HC., Su YC.* The Study of Improvement and Risk Evaluation for Mobile Application Security Testing. Security with Intelligent Computing and Big-data Services / SL. Peng, SJ. Wang, V. Balas, M. Zhao. Cham: Springer, 2018. – Vol. 733. – P. 248-256.

8. *Oyetoyan T., Milosheska B., Grini M., Soares Cruzes D.* Myths and Facts About Static Application Security Testing Tools: An Action Research at Telenor Digital. Agile Processes in Software Engineering and Extreme Programming / J. Garbajosa, X. Wang, A. Aguiar. Cham: Springer, 2018. – Vol. 314. – P. 86-103.

9. *Malek S., Bagheri H., Garcia J., Sadeghi A.* Security and Software Engineering. Handbook of Software Engineering / S. Cha, R. Taylor, K. Kang. Cham: Springer, 2019. – P. 445-489.

10. *Mejía J., Maciel P., Muñoz M., Quiñonez Y.* Frameworks to Develop Secure Mobile Applications: A Systematic Literature

Review. Trends and Innovations in Information Systems and Technologies / Á. Rocha, H. Adeli, L. Reis, S. Costanzo, I. Orovic, F. Moreira. Cham: Springer, 2020. – Vol. 1160. - P. 137-146.

11. *Rahalkar S.* Testing Mobile Apps and APIs with Burp Suite. A Complete Guide to Burp Suite. Berkeley: Apress, 2021. – P. 147-164.

12. *Tran AD., Nguyen MQ., Phan GH., Tran MT.* Security Issues in Android Application Development and Plug-in for Android Studio to Support Secure Programming. Future Data and Security Engineering / T.K. Dang, J. Küng, T.M. Chung, M. Takizawa. Cham: Springer, 2021. – Vol. 1500. – P. 105-122.

13. *Ogundokun R.O., Misra S., Segun-Owolabi T., Gulanikar A.A., Agrawal A., Damasevicius R.* A Web Application Vulnerability Testing System. Recent Innovations in Computing / P.K. Singh, Y. Singh, J.K. Chhabra, Z. Illés, C. Verma. Cham: Springer, 2022. – Vol. 855. – P. 741-751.

14. *Міснік О.І., Антонішин М.В., Цуркан В.В.* Аналіз якості роботи сканерів уразливостей веб-застосунків. Моделювання та інформаційні технології. – 2018. – Вип. 83. – С. 77-86.

15. *Luo Y., Wan J., She S.* Software Security Vulnerability Mining Based on Deep Learning. Application of Intelligent Systems in Multi-modal Information Analytics / V. Sugumaran, A.G. Sreedevi, Z. Xu. Cham: Springer, 2022. – Vol. 136. – P. 536-543.

16. Drozer user guide. URL: https://labs. f-secure.com/assets/BlogFiles/ mwri-drozer-user- guide-2015-03-23.pdf.

17. MobSF Documentation. URL: https://mobsf.github.io/docs/#/.

18. Xposed Framework implementation. URL: https://github.com/topics/xposed-framework.

19. Frida. URL: https://github.com/ frida.

20. *Antonishyn M.* Four ways to bypass Android SSL. Verification and Certificate Pinning. International Scientific Journal "Transfer of Innovative Technologies". – 2020. – Vol. 3. – No. 1. – P. 96-99.

Антонішин М.В., Міснік О.І., Цуркан В.В.

СПОСОБИ ТЕСТУВАННЯ УРАЗЛИВОСТЕЙ МОБІЛЬНИХ ПРОГРАМНИХ ЗАСТОСУНКІВ

Проаналізовано застосовність способів тестування уразливостей мобільних програмних застосунків. Показано їхню орієнтованість на задоволення настанов відкритого проєкту забезпечення безпеки вебзастосунків. Це зведено до застосування керівництва з тестування і стандарту верифікування безпеки мобільних програмних застосунків. За ними виокремлено статичний і динамічний способи тестування уразливостей. Водночас встановлено обмеженість практичного застосування даних способів у зв'язку з неоднозначною належністю сценаріїв до статичного або динамічного тестування, наприклад: експортованих компонентів, MSTG-STORAGE-6, нестандартних сертифікатів, сертифікату SSL-пінінгу, MSTG-NETWORK-4. Цим обумовлено актуальність аналізування застосовності способів тестування уразливостей мобільних програмних застосунків. Результатами існуючих досліджень підтверджено узагальнену орієнтованість на задоволення настанов відкритого проєкту забезпечення безпеки вебзастосунків. На їхній основі забезпечено повноту сценаріїв тестування і, як наслідок, сформульовано рекомендації для зменшення витоку конфіденційної інформації. Зосереджено увагу на протидії атакам критично важливих програмних застосунків. Окрему увагу приділено розширенню можливостей розроблених засобів тестування уразливостей використанням відповідних фреймворків. Виокремленими та дослідженими прикладами продемонстровано обмеженість застосування статичного та динамічного способів. Для цього продемонстровано, по-перше, використання відповідних інтрументальних засобів

Drozer, MobSF, Xposed, Frida. По-друге, тестування обходженості механізмів забезпечення безпеки від перехоплення і дешифрування HTTP-трафіку, зокрема, SSL-пінінгу. Перш за все через складність однозначного як визначення, так і виявлення залежності між сценаріями тестування уразливостей мобільних програмних застосунків. Водночас встановлено некоректність окремого застосування динамічного способу тестування. Це обумовлено необхідністю комплексного виконання дій, зокрема, статичного сканування мобільного програмного застосунку, мануального аналізування вихідного коду фахівцем. Крім того використання додаткових фреймворків і розроблення окремих модулів тестування уразливостей.

Ключові слова: *мобільний програмний застосунок, тестування уразливостей, OWASP MSTG, OWASP MASVS, статичний спосіб тестування уразливостей, динамічний спосіб тестування уразливостей.*

Antonishyn M.V., Misnik O.I., Tsurkan V.V.

WAYS OF TESTING VULNERABILITIES IN MOBILE APPLICATIONS

It has been analyzed the applicability ways of vulnerability testing for mobile software applications. After doing this, we have further demonstrated their orientation towards meeting the guidelines of the open source security project for web applications, OWASP. This has led to the identification of static and dynamic vulnerability testing methods. However, the practical application of these ways is limited due to the ambiguous classification of scenarios as either static or dynamic testing, such as exported components, MSTG-STORAGE-6, non-standard certificates, SSL pinning certificates, and MSTG-NETWORK-4. This highlights the relevance of analyzing the applicability ways of vulnerability testing for mobile software applications. Existing research has confirmed the generalized orientation towards meeting OWASP guidelines and ensured the completeness of testing scenarios, resulting in recommendations for reducing the leakage of confidential information. Attention has been focused on preventing attacks on critical software applications, and efforts have been made to expand the capabilities of vulnerability testing tools using appropriate frameworks. The limitations of static and dynamic testing ways have been demonstrated through examples of the use of corresponding tools such as Drozer, MobSF, Xposed, and Frida. Additionally, testing has been conducted on the bypassing of security mechanisms for intercepting and decrypting HTTP traffic, including SSL pinning. We have identified the complexity of determining and establishing dependencies between vulnerability testing scenarios for mobile software applications as the primary challenge in their separate application. Furthermore, the incorrectness of using only dynamic testing has been established. This is because the comprehensive execution of actions is necessary, including static scanning of the mobile software application and manual analysis of the source code by an expert. Additionally, the use of additional frameworks and the development of separate vulnerability testing modules have been explored.

Keywords*: mobile application, vulnerability testing, OWASP MSTG, OWASP MASVS, vulnerabilities testing static way, vulnerabilities testing dynamic way.*