ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
ΠΜΣ «ΑΣΦΑΛΕΙΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ»

«Δοκιμές Παρείσδυσης σε Εφαρμογές Android»
(ANDROID APPLICATION PENETRATION TESTING)

Συγγραφέας: Χονδρογιάννης Εμμανουήλ
Επιβλέπων Καθηγητής: Λαμπρινουδάκης Κωνσταντίνος

Πειραιάς, Φεβρουάριος 2020

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Λαμπρινουδάκη Κωνσταντίνο, καθηγητή μου και πρόεδρο του τμήματος «Ψηφιακών Συστημάτων» για την καθοδήγησή του του καθώς και τον Ταγματάρχη Έρευνας Πληροφορικής κ. Βάσιο Γεώργιο για την συνολική του υποστήριξη και συνεισφορά στην διεκπεραίωση της παρούσας εργασίας .

Τέλος, θα ήθελα να ευχαριστήσω το «Κέντρο Πληροφορικής Υποστήριξης Ελληνικού Στρατού» για την υποστήριξή του κατά την περίοδο εκπόνησης της διπλωματικής μου εργασίας.

Περίληψη

Το Android είναι ένα από τα πιο δημοφιλή λειτουργικά συστήματα για κινητές συσκευές στον κόσμο με συνεχή ανοδική πορεία. Δεδομένης της μεγάλης χρήσης του λογισμικού, ανοδική παρατηρείται η δημιουργία και χρήση των εφαρμογών του. Για τον λόγο αυτό αυξημένη είναι και ανάγκη για ασφάλεια στις εφαρμογές του Android.

Βάσει των παραπάνω, στην παρούσα διπλωματική γίνεται αρχικά λεπτομερής αναφορά στην δομή του λογισμικού ώστε να κατανοηθεί ο τρόπος λειτουργίας των εφαρμογών ενώ αναφέρονται και επεξηγούνται οι κίνδυνοι που αντιμετωπίζουν οι εφαρμογές όπως τους ορίζει ο OWASP. Για τον έλεγχο και εντοπισμό των κινδύνων αυτών έγινε επιλογή μίας ευάλωτης εφαρμογής (DIVA) προκειμένου να διεξαχθεί το penetration test ενώ σε κάθε βήμα υπάρχει επεξήγηση της πορείας που ακολουθήθηκε μέχρι τον εντοπισμό της ευπάθειας. Τέλος, παραθέτονται συμπεράσματα γύρω από την διεξαγωγή του penetration test.

Λέξεις Κλειδιά: Android, Mobile Application, Penetration Testing, OWASP, Vulnerability.

Abstract

Android is the most popular mobile operating system hence its applications are a common target to cyber-attacks. Therefore, the process of penetration testing is a fundamental step to recognize the impact of the vulnerabilities before the official launch of an application.

The purpose of this thesis is to present the penetration testing process and its results. At first, we analyzed the Android Operating System in order to understand the basics of Android applications and presented the top 10 threats mobiles face more frequently as OWASP reports. Afterwards, we prepared the penetration testing lab with some of the most important tools included and finally performed a penetration test on a known vulnerable application by describing every step of the process.

Keywords: Android, Mobile Application, Penetration Testing, OWASP, Vulnerability.

Περιεχόμενα

Ευχαριστίες 2
Περίληψη 3
Abstract 4
Κατάλογος Εικόνων 7
1. Περιγραφή Εργασίας 9
1.1 Γενικά 9
1.2 Στόχος Εργασίας 9
1.3 Δομή Εργασίας 9
2. Εισαγωγή 10
3. Λειτουργικό Σύστημα Android 11
3.1 Τι είναι το Android; 11
3.2 Η Ιστορία του Android 11
3.3 Android Versions 12
3.4 Αρχιτεκτονική Android 13
3.5 Application Components 16
3.6 Επικοινωνία μεταξύ Components 17
3.7 Android Application Package (APK) 17
3.8 Android Manifest Overview 19
4. Permission System 19
4.1 Application Sandboxing 19
4.2 Application Permissions 20
4.3 Protection Levels 20
5. Penetration Testing 21
5.1 Τι είναι το Penetration Testing; 21
5.2 Τύποι Penetration Testing 21
5.3 Γενική Μεθοδολογία 22
6. Open Web Application Security Project (OWASP) 23
6.1 Τι είναι το «OWASP»; 23
6.2 OWASP Top 10 Mobile Risks 24
6.3 OWASP Top 10 Mobile Risks – Γενικές Οδηγίες Εντοπισμού Ευπαθειών 27
7. Δημιουργία Penetration Testing Περιβάλλοντος 31
7.1 Εγκατάσταση Ubuntu 18.04.3 LTS 31
7.2 Εγκατάσταση Προσομοιωτή Κινητού - Genymotion 38
7.3 Εγκατάσταση Android Studio 40

7.4 Εγκατάσταση MobSF 41
7.5 Εγκατάσταση Drozer 42
7.6 Εγκατάσταση Jadx 45
8. Penetration Testing Case Study: Damn Insecure and Vulnerable Application (DIVA) 46
8.1 Insecure Logging 47
8.2 Hardcoding Issues 48
8.3 Insecure Data Storage – Part 1 50
8.4 Insecure Data Storage – Part 2 51
8.5 Insecure Data Storage – Part 3 52
8.6 Insecure Data Storage – Part 4 53
8.7 Input Validation Issues – Part 1 54
8.8 Access Control Issues – Part 1 56
8.9 Access Control Issues – Part 2 57
8.10 Access Control Issues – Part 3 58
8.11 Hardcoding Issues – Part 2 60
8.12 Input Validation Issues – Part 3 62
9. Συμπεράσματα 63
10. Βιβλιογραφία 64

Κατάλογος Εικόνων

Εικόνα 1. Statista - Number of mobile app downloads worldwide from 2014 to 2023, by region (in billions) 10
Εικόνα 2. Google Android logo over the years (2008/2014/2019) 11
Εικόνα 3. Wikipedia - Android (Πίνακας Android Version 2011-2019) 12
Εικόνα 4. Platform Architecture - The Android Software Stack 13
Εικόνα 5. Η εσωτερική δομή ενός APK αρχείου 18
Εικόνα 6. Χρήση του λογισμικού Rufus (1/2) 31
Εικόνα 7. Χρήση του λογισμικού Rufus (2/2) 32
Εικόνα 8. Ενεργοποίηση BIOS για διαφορετικό κατασκευαστή 32
Εικόνα 9. Εγκατάσταση Ubuntu - Βήμα 1ο 33
Εικόνα 10. Εγκατάσταση Ubuntu - Βήμα 2ο 33
Εικόνα 11. Εγκατάσταση Ubuntu - Βήμα 3ο 34
Εικόνα 12. Εγκατάσταση Ubuntu - Βήμα 4ο 34
Εικόνα 13. Εγκατάσταση Ubuntu - Βήμα 5ο 35
Εικόνα 14. Εγκατάσταση Ubuntu - Βήμα 6ο 35
Εικόνα 15. Εγκατάσταση Ubuntu - Βήμα 7ο 36
Εικόνα 16. Εγκατάσταση Ubuntu - Βήμα 8ο 36
Εικόνα 17. Εγκατάσταση Ubuntu - Βήμα 9ο 37
Εικόνα 18. Εγκατάσταση Ubuntu - Βήμα 10 37
Εικόνα 19. Εγκατάσταση Genymotion (1/3) 38
Εικόνα 20. Εγκατάσταση Genymotion (2/3) 39
Εικόνα 21. Εγκατάσταση Genymotion (3/3) 39
Εικόνα 22. Android Studio - Επιλογή Εικονικής Συσκευής 40
Εικόνα 23. Android Studio - Εγκατάσταση SDK 40
Εικόνα 24. Εγκατάσταση MobSF - Εκτέλεση μέσω τερματικού 41
Εικόνα 25. MobSF - Αρχική Οθόνη 42
Εικόνα 26. Η εφαρμογή Drozer στο κεντρικό μενού 43
Εικόνα 27. Αρχική οθόνη Drozer 44
Εικόνα 28. Σύνδεση στο Drozer μέσω τερματικού 44
Εικόνα 29. Εκτέλεσε Jadx μέσω τερματικού 45
Εικόνα 30. Γραφικό περιβάλλον Jadx 45
Εικόνα 31. Αρχική οθόνη εφαρμογής DIVA 46
Εικόνα 32. Insecure Logging - Αρχική Οθόνη 47
Εικόνα 33. Καταχώρηση της εισόδου ως plaintext στα Logs 47
Εικόνα 34. Jadx - Insecure Logging - Κώδικας Ευπάθειας 48
Εικόνα 35. Hardcoding Issues Part 1 - Αρχική Οθόνη 48
Εικόνα 36. Jadx - Hardcoding Issues Part1 - Κώδικας Ευπάθειας 49
Εικόνα 37. Hardcoding Issues Part 1 - Επιτυχής Είσοδος 49
Εικόνα 38. Insecure Data Storage Part 1 - Αρχική Οθόνη 50
Εικόνα 39. Jadx - Insecure Data Storage Part 1 - Κώδικας Ευπάθειας 50
Εικόνα 40. Insecure Data Storage Part1 - Εντοπισμός Ευπάθειας μέσω τερματικού 51
Εικόνα 41. Insecure Data Storage Part 2 - Αρχική Οθόνη 51
Εικόνα 42. Jadx – Insecure Data Storage Part 2 – Κώδικας Ευπάθειας 52
Εικόνα 43. Insecure Data Storage Part 2 - Αποτέλεσμα μέσω τερματικού 52
Εικόνα 44. Insecure Data Storage Part 3 - Αρχική Οθόνη 52
Εικόνα 45. Insecure Data Storage Part 3 - Κώδικας Ευπάθειας 53

Εικόνα 46. Insecure Data Storage Part 3 - Εμφάνιση Credentials μέσω τερματικού 53
Εικόνα 47. Insecure Data Storage Part 4 - Αρχική Οθόνη 53
Εικόνα 48. Insecure Data Storage Part 4 - Κώδικας Ευπάθειας 54
Εικόνα 49. Input Validation Issues Part 1 - Αρχική Οθόνη 54
Εικόνα 50. MobSF - Επιλογές Decompiled Κώδικα..... 55
Εικόνα 51. MobSF - Κώδικας Ευπάθειας..... 55
Εικόνα 52. Input Validation Issues - Επιτυχία SQL Injection 55
Εικόνα 53. Access Control Issues Part 1 - Αρχική Οθόνη και «VIEW API CREDENTIALS» οθόνη..... 56
Εικόνα 54. Εμφάνιση της ονομασίας του Activity μέσω τερματικού 56
Εικόνα 55. MobSF - Εμφάνιση των exported activities..... 56
Εικόνα 56. Drozer - Εμφάνιση των exported activities 57
Εικόνα 57. Drozer - Ενεργοποίηση του activity μέσω της παραπάνω εντολής 57
Εικόνα 58. Access Control Issues Part 2 - Αρχική Οθόνη 57
Εικόνα 59. Drozer - Εμφάνιση της ονομασίας του activity 57
Εικόνα 60. MobSF - Manifest File - Intent Filter 58
Εικόνα 61. Access Control Issues Part 3 - Αρχική Οθόνη και «GO TO PRIVATE NOTES» Οθόνη 58
Εικόνα 62. Access Control Issues Part 3 - Diva Private Notes Οθόνη 59
Εικόνα 63. Drozer - Έλεγχος Exported Activities..... 59
Εικόνα 64. Drozer - Scanning για εύρεση URIs..... 59
Εικόνα 65. Drozer - Query για τα περιεχόμενη του URI 60
Εικόνα 66. Hardcoding Issues Part 2 - Αρχική Οθόνη (αριστερά) και ο κώδικας του activity (δεξιά) . 60
Εικόνα 67. MobSF - Κώδικας DivaJni.java 60
Εικόνα 68. Εμφάνιση Strings που διαθέτει το αρχείο libdivajni.so 61
Εικόνα 69. Hardcoding Issues Part 2 - Επιτυχής πρόσβαση 61
Εικόνα 70. Input Validation Issues Part 3 - Αρχική Οθόνη 62
Εικόνα 71. Αποτυχία λειτουργίας της εφαρμογής λόγω buffer overflow 62

1. Περιγραφή Εργασίας

1.1 Γενικά

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο του Προγράμματος Μεταπτυχιακών Σπουδών «Ασφάλεια Ψηφιακών Συστημάτων» του Τμήματος «Ψηφιακών Συστημάτων» για το 3^{ο} εξάμηνο κατά το ακαδημαϊκό έτος 2019-2020 από τον μεταπτυχιακό φοιτητή Χονδρογιάννη Εμμανουήλ. Η εκπόνηση της εργασίας με θέμα «Δοκιμές Παρείσδυσης σε Εφαρμογές Android» πραγματοποιήθηκε σε συνεργασία με το «Κέντρο Πληροφορικής Υποστήριξης Ελληνικού Στρατού».

1.2 Στόχος Εργασίας

Ο στόχος της παρούσας διπλωματικής εργασίας χωρίζεται σε δύο βασικά μέρη. Το πρώτο μέρος αφορά την κατανόηση του λειτουργικού συστήματος Android. Η ύπαρξη καλής γνώσης σχετικά με το λειτουργικό σύστημα που πρόκειται να εξεταστεί, αποτελεί βασικό κριτήριο για την πραγματοποίηση ενός penetration test. Το δεύτερο μέρος αφορά την πρακτική παρουσίαση του penetration testing σε μία ευάλωτη εφαρμογή, περιγράφοντας το σύνολο των εργαλείων που χρησιμοποιήσαμε αλλά και των βημάτων που ακολουθήσαμε.

1.3 Δομή Εργασίας

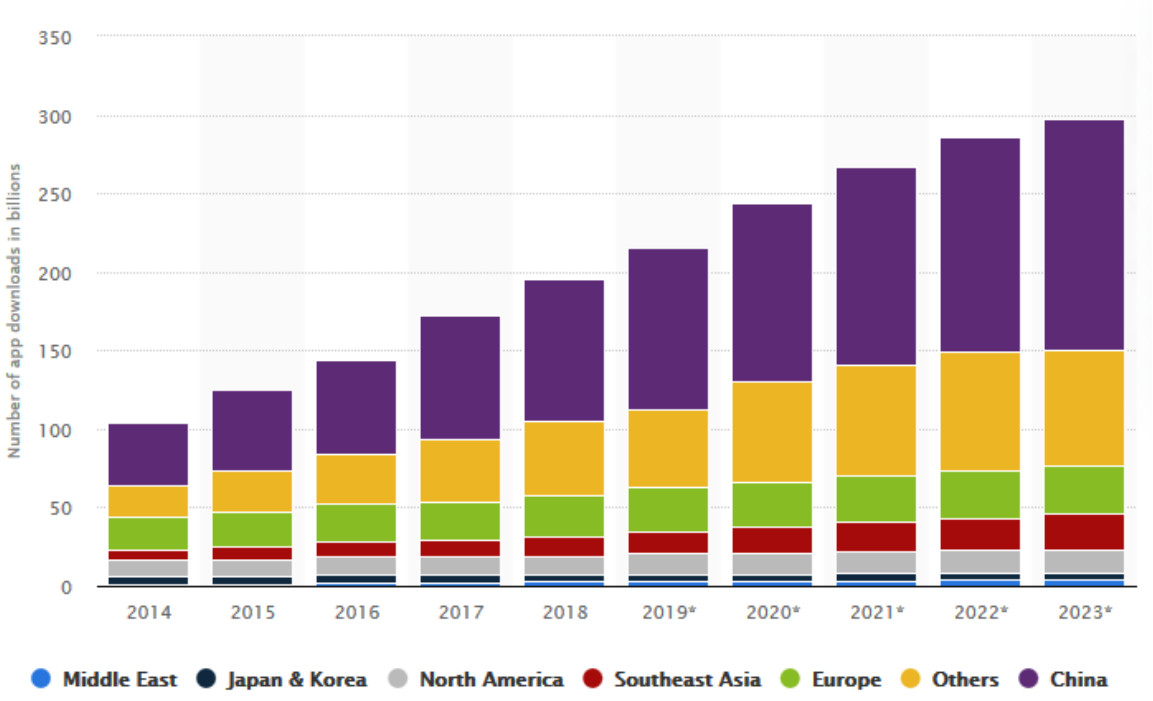
Η παρούσα διπλωματική εργασία ακολουθεί την παρακάτω δομή: Αρχικά, στην Ενότητα 2 γίνεται εισαγωγή στο λογισμικό των κινητών, το penetration testing και την ανάγκη για εντοπισμό των ευπαθειών. Στην Ενότητα 3 γίνεται ανάλυση του λογισμικού του Android ενώ στην Ενότητα 4 αναφέρονται οι μηχανισμοί ασφάλειας που διαθέτει το λογισμικό. Στην συνέχεια, στην Ενότητα 5 και 6 αντίστοιχα γίνεται μία σύντομη αναφορά στην έννοια του penetration testing και στο project OWASP και τους κινδύνους που αντιμετωπίζουν σε μεγάλο βαθμό οι κινητές συσκευές. Ακόμα, στην Ενότητα 7, παρουσιάζονται τα απαραίτητα εργαλεία που δομούν το pentesting lab και στην Ενότητα 8, υλοποιείται το penetration test σε μία ευάλωτη εφαρμογή. Στην Ενότητα 9 παραθέτονται τα συμπεράσματα - σκέψεις γύρω από την ασφάλεια των εφαρμογών και την αναγκαιότητα των penetration tests. Τέλος, στην Ενότητα 10, παρουσιάζονται όλοι οι σύνδεσμοι - πηγές που βοήθησαν στην εκπόνηση της εργασίας.

2. Εισαγωγή

Στην εποχή μας οι άνθρωποι χρησιμοποιούν τα smartphones σε σχεδόν όλες τις πτυχές της ζωής τους, όπως είναι η κοινωνική δικτύωση, η αναζήτηση του επόμενου σταθμού της καριέρας τους, τα οικονομικά, η μάθηση και η υγεία. Σύμφωνα με το Statista, όλοι οι δημιουργοί (εκτός της gaming κατηγορίας) εφαρμογών παρατήρησαν αυξανόμενη τάση στο κατέβασμα των εφαρμογών τους μέσω του Google Play Store χωρίς καμία ένδειξη για μείωση του ρυθμού αυτού (Εικόνα 1). Ωστόσο, η τάση αυτή προσέλκυσε όλο και περισσότερους hackers γεγονός που όχι μόνο αυξάνει την πιθανότητα για επίθεση στις εφαρμογές των κινητών αλλά δίνει την δυνατότητα για επίθεση σε οποιοδήποτε σύστημα ή συσκευή που βρίσκεται στο ίδιο δίκτυο. Για τον λόγο αυτό η διεξαγωγή του penetration testing αποτελεί ένα από τα πιο σημαντικά σημεία της ασφάλειας καθώς είναι δυνατός ο εντοπισμός αδυναμιών, ο τρόπος εκμετάλλευσής τους αλλά και το μέγεθος της ζημιάς που μπορεί να προκληθεί.

Γενικότερα, ως «penetration testing» ή αλλιώς «ethical hacking» ορίζεται η διαδικασία κατά την οποία πραγματοποιείται εξουσιοδοτημένη κυβερνοεπίθεση σε κάποιο υπολογιστικό σύστημα προκειμένου να εκτιμηθεί το επίπεδο ασφάλειας. Με άλλα λόγια εντοπίζονται ταυτόχρονα οι ευπάθειες αλλά και τα δυνατά σημεία του συστήματος που εξετάζεται.

Το Android λειτουργικό, μεταξύ των Mobile OS, είναι ένα από τα πιο δημοφιλή και συνεχώς αναπτυσσόμενα λειτουργικά συστήματα. Όπως συμβαίνει σε κάθε λογισμικό, κατά το στάδιο της ανάπτυξης, οι developers άθελά τους δημιουργούν κενά ασφαλείας ή αγνοούν τις καλές πρακτικές ανάπτυξης ενός ασφαλούς συστήματος. Τέτοια λάθη θα πρέπει να εντοπίζονται πολύ πριν την επίσημη έκδοση της εφαρμογής. Ειδικότερα, αν μία εφαρμογή θέλουμε να έχει ένα σημαντικό επίπεδο ασφάλειας, θα πρέπει να πραγματοποιείται ανά τακτά χρονικά διαστήματα το penetration testing προκειμένου να μειωθούν οι ευπάθειες και οι επιτυχείς προσπάθειες επίθεσης .



Εικόνα 1.Statista - Number of mobile app downloads worldwide from 2014 to 2023, by region (in billions)

3. Λειτουργικό Σύστημα Android

3.1 Τι είναι το Android;

Το Android είναι ένα λειτουργικό σύστημα για κινητές συσκευές το οποίο βασίστηκε σε μία τροποποιημένη έκδοση του Linux Kernel και άλλων λογισμικών ανοιχτού κώδικα. Η ανάπτυξη του πραγματοποιήθηκε από μία κοινοπραξία εταιρειών, γνωστή ως Open Handset Alliance ενώ την προώθηση του ανέλαβε η Google.

Η Open Handset Alliance είναι ένα σύνολο είναι ένα σύνολο εταιρειών με μεγάλη εμπειρία γύρω από το hardware και το software. Στην ομάδα αυτή συμπεριλαμβάνονται εταιρείες όπως η Google, Intel, NVIDIA, Qualcomm, Motorola, HTC και T-Mobile με βασικό τους άξονα το Android Software. Ο κύριος στόχος τους είναι η ανάπτυξη νέων τεχνολογιών που θα μειώσουν σημαντικά τον χρόνο - κόστος ανάπτυξης των κινητών συσκευών αλλά και των υπηρεσιών.

3.2 Η Ιστορία του Android

Τον Οκτώβρη του 2003 δημιουργήθηκε η Android Inc. στο Πάλο Άλτο της Καλιφόρνια από τους Andy Rubin, Rich Miner, Nick Sears, and Chris White. Οι πρωταρχικές βλέψεις των δημιουργών ήταν η δημιουργία ενός εξελιγμένου λογισμικού για ψηφιακές κάμερες το οποίο αποτέλεσε την αρχή των επενδύσεων. Ωστόσο, η αγορά που σχετιζόταν με το αντικείμενό τους δεν ανταποκρινόταν στους στόχους της Android γεγονός που οδήγησε στην αλλαγή της επαγγελματικής τους πορείας. Τον Ιούλιο του 2005 η Google εξαγόρασε την Android Inc. για 50 εκατομμύρια δολάρια ενώ απορρόφησε και τους προαναφερθέντες συντελεστές.

Η πρώτη αναφορά στο λογισμικό Android καταγράφηκε το 2007 ενώ τον Σεπτέμβρη του 2008 τέθηκε προς πώληση η πρώτη Android συσκευή.

Εικόνα 2.Google Android logo over the years (2008/2014/2019)

3.3 Android Versions

Η Google από το 2011 και μετά ξεκίνησε να αναπτύσσει τις πιο στιβαρές εκδόσεις του Android. Στο παρακάτω στιγμιότυπο παρουσιάζονται όλες οι εκδόσεις του λογισμικού από τον Φεβρουάριο του 2011 με την έκδοση «Gingerbread 2.3» έως τον Σεπτέμβρη του 2019 με την έκδοση «Android 10». Από τα αριστερά στα δεξιά φαίνεται η έκδοση του Android, η κωδική ονομασία, η ημερομηνία έκδοσης και το API Level για την κάθε έκδοση. Οι εκδόσεις με ροζ απόχρωση είναι παλαιές και δεν υποστηρίζονται πλέον, με κίτρινο είναι παλαιές αλλά υποστηρίζονται ενώ με πράσινο ορίζεται η πιο πρόσφατη.

Σημαντικό είναι να αναφερθεί ότι κατά την περίοδο εκπόνησης της διπλωματικής, η έκδοση του «Android 10» βρίσκεται κυρίως στα κινητά «Pixel» της Google (π.χ. Pixel, Pixel XL, Pixel 2, Pixel 2 XL, Pixel 3, Pixel 3 XL, Pixel 3a, και Pixel 3a XL)..

Version	Code name	Release date	API level
10	10	September 3, 2019	29
9	Pie	August 6, 2018	28
8.1	Oreo	December 5, 2017	27
8.0		August 21, 2017	26
7.1	Nougat	October 4, 2016	25
7.0		August 22, 2016	24
6.0	Marshmallow	October 5, 2015	23
5.1	Lollipop	March 9, 2015	22
5.0		November 3, 2014	21
4.4	KitKat	October 31, 2013	19
4.3	Jelly Bean	July 24, 2013	18
4.2		November 13, 2012	17
4.1		July 9, 2012	16
4.0	Ice Cream Sandwich	October 19, 2011	15
2.3	Gingerbread	February 9, 2011	10

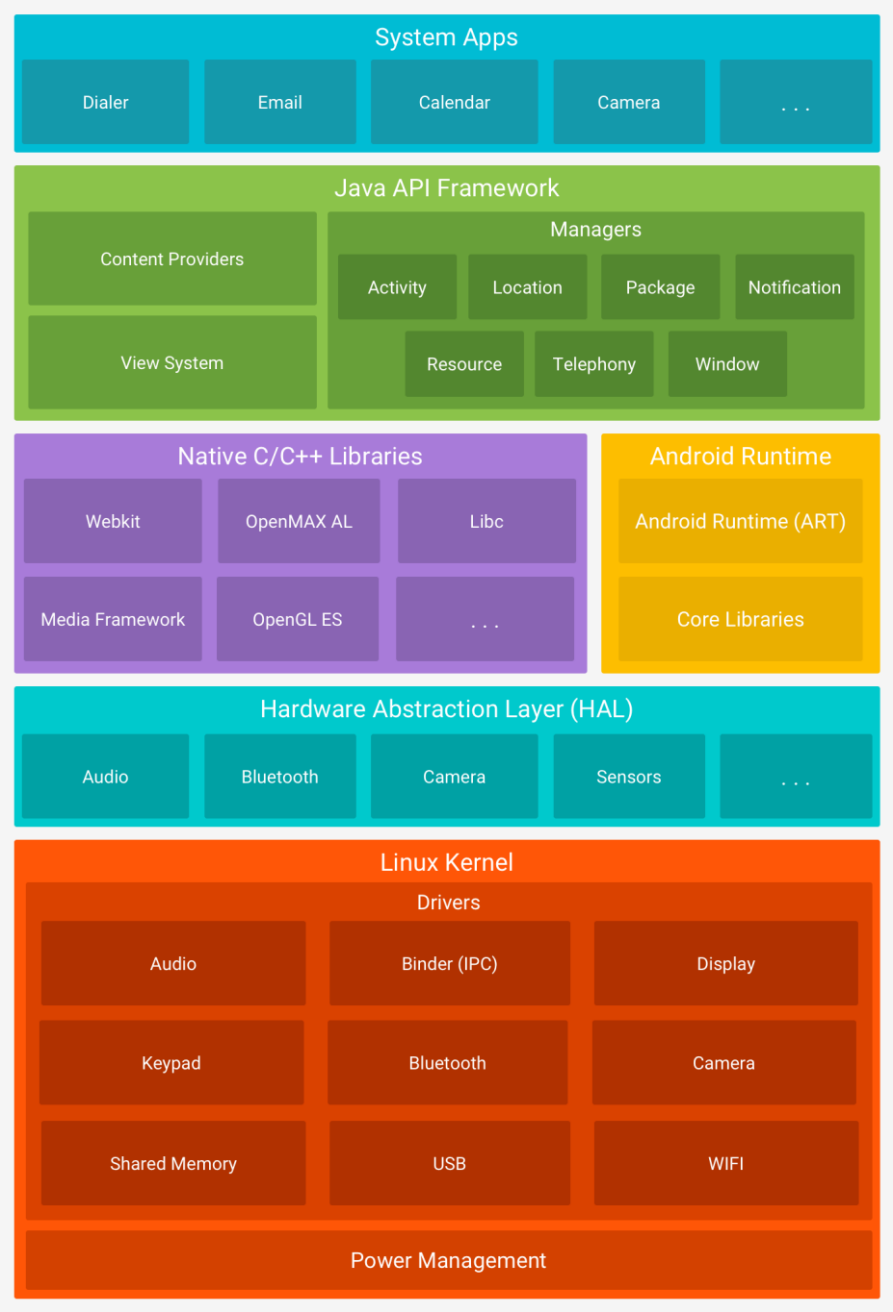
Old version Older version, still supported **Latest version**

Εικόνα 3. Wikipedia - Android (Πίνακας Android Version 2011-2019)

3.4 Αρχιτεκτονική Android

Για να υπάρξει μία αρχική εικόνα γύρω από τον τρόπο λειτουργίας του λογισμικού θα πρέπει να εξεταστεί η αρχιτεκτονική του. Το Android χωρίζεται σε 5 διαφορετικά επίπεδα όπως φαίνεται στο παρακάτω σχήμα και θα αναλυθούν στην συνέχεια. Επιγραμματικά τα επίπεδα αυτά είναι τα εξής:

1. Linux Kernel
2. Hardware Abstraction Layer (HAL)
3. Native Libraries – Android Runtime (ART)
4. Java API Framework
5. System Applications



Εικόνα 4. Platform Architecture - The Android Software Stack

3.4.1 Linux Kernel

Στο χαμηλότερο επίπεδο της στοίβας του Android βρίσκεται το Linux Kernel το οποίο βασίζεται στην Linux 4.14 έκδοση. Αποτελεί την βάση του λειτουργικού συστήματος καθώς παρέχει υπηρεσίες όπως είναι η διαχείριση της μνήμης, το multitasking, η διαχείριση ενέργειας, και εν γένει drivers σημαντικά για την συσκευή (Εικόνα 4). Στο επίπεδο αυτό δεν υπάρχει άμεση αλληλεπίδραση χρήστη – συστήματος.

3.4.2 Hardware Abstraction Layer (HAL)

Στο επόμενο επίπεδο βρίσκεται το Hardware Abstraction Layer το οποίο λειτουργεί ως διεπαφή μεταξύ του hardware με το υπόλοιπο Java API Framework και διαθέτει πολλά διαφορετικά library modules. Κάθε ένα από αυτά τα modules δημιουργεί μια διεπαφή για κάθε ξεχωριστό κομμάτι hardware. (π.χ. Κάμερα, Bluetooth κ.α.). Με άλλα λόγια, κάθε φορά που το API ζητά πρόσβαση σε κάποιο hardware κομμάτι του συστήματος, το Android καλεί το αντίστοιχο library module.

3.4.3 Android Runtime – Native Libraries

Στο τρίτο κατά σειρά επίπεδο βρίσκονται οι Native Libraries οι οποίες περιλαμβάνουν C/C++ core libraries αλλά και Libraries υλοποιημένες σε Java:

- Media: είναι βιβλιοθήκη η οποία παρέχει υποστήριξη για την αναπαραγωγή-καταγραφή αρχείων ήχου και βίντεο.
- Surface Manager: είναι βιβλιοθήκη υπεύθυνη για την διαχείριση πρόσβασης στο υποσύστημα οθονών.
- SGL και OpenGL: και τα δύο είναι cross-platform, cross-language APIs που χρησιμοποιούνται για 2D και 3D γραφικά.
- SQLite: σύστημα για βάσεις δεδομένων
- FreeType: σύστημα γραμματοσειρών
- Web-Kit: open-source browser
- SSL: είναι μία τεχνολογία υπεύθυνη για την κρυπτογραφημένη σύνδεση μεταξύ ενός web browser και ενός web server.

Παράλληλα, στο επίπεδο αυτό βρίσκεται το Android Runtime το οποίο αποτελεί το runtime environment του Android. Στις παλαιότερες εκδόσεις Android οι εφαρμογές έτρεχαν εσωτερικά σε Virtual Machines. Η πρώτη έκδοση ήταν το Dalvik Virtual Machine (DVM) το οποίο χαρακτηρίζεται από την «Just in Time» (JIT) τεχνολογία, η οποία μετέτρεπε τον κώδικα της κάθε εφαρμογής σε γλώσσα μηχανής κάθε φορά που ο χρήστης επιθυμούσε να την εκκινήσει. Από την έκδοση 4.4 KitKat και ύστερα εμφανίστηκε πειραματικά το ART (Android Runtime) το οποίο σε αντίθεση με τον προκάτοχό του, παρέχει την δυνατότητα μετατροπής του κώδικα της εφαρμογής σε γλώσσα μηχανής κατά την διαδικασία εγκατάστασης, γεγονός που καθυστερεί την εγκατάσταση μίας εφαρμογής αλλά προσφέρει μεγαλύτερη ταχύτητα πλοήγησης.

Η τεχνολογία αυτή ονομάζεται «Ahead of Time» compilation (AOT) ενώ πλέον λειτουργεί ως προκαθορισμένος τρόπος για το compile των εφαρμογών. Ωστόσο, κάθε χρήστης έχει την δυνατότητα να επιλέξει ανάμεσα στις δύο προαναφερθείσες τεχνολογίες.

3.4.4 Java API Framework

Στο 4ο επίπεδο της στοίβας βρίσκεται το Java API Framework από το οποίο οι εφαρμογές αντλούν πολλά διαφορετικά services με την μορφή Java κλάσεων. Κάποια από τα πιο σημαντικά services είναι:

- Activity Manager: διαχειρίζεται τον κύκλο ζωής της εφαρμογής.
- Content Providers: χρησιμοποιούνται για την διαχείριση των δεδομένων που μοιράζονται μεταξύ των εφαρμογών.
- Resource Manager: παρέχει πρόσβαση σε resources όπως είναι τα strings, color settings και user interface layouts.
- Notifications Manager: παρέχει στις εφαρμογές την δυνατότητα εμφάνισης ειδοποιήσεων στον χρήστη.
- Package Manager: παρέχει πληροφορίες σε μία εφαρμογή για κάποια άλλη εφαρμογή που είναι εγκαταστημένο στην συσκευή.
- Telephony Manager: παρέχει πληροφορίες σχετικά με το κινητό π.χ. sim serial number, κατάσταση κινητού κ.α.
- Location Manager: παρέχει την δυνατότητα σε μία εφαρμογή να ενημερώνεται για την τρέχουσα τοποθεσία.

3.4.5 System Applications

Στο 5ο και τελευταίο επίπεδο της στοίβας βρίσκονται οι εφαρμογές του συστήματος με τις οποίες έχει άμεση επαφή ο απλός χρήστης. Τέτοιες εφαρμογές είναι το email, τα SMS, το ημερολόγιο, ο περιηγητής δικτύου, οι επαφές και άλλες οι οποίες είναι υλοποιημένες σε Java.

Στο επίπεδο αυτό οι εφαρμογές έχουν την δυνατότητα να τρέχουν σε sandboxed περιβάλλον, γεγονός που απομονώνει κάθε εφαρμογή από τις υπόλοιπες από τυχόν αστοχίες – λάθη του λογισμικού. Περισσότερα για τον μηχανισμό του sandboxing αναφέρονται σε επόμενο κεφάλαιο της εργασίας.

3.5 Application Components

Τα application components αποτελούν τα θεμέλια της κάθε εφαρμογής. Το κάθε component αποτελεί είσοδο για το σύστημα ή για τον χρήστη. Υπάρχουν τέσσερα βασικά components τα οποία αναλύονται παρακάτω:

1. Activities
2. Services
3. Broadcast Receivers
4. Content Providers

3.5.1 Activities

Τα activities είναι ο βασικός τρόπος αλληλεπίδρασης του χρήστη με την εφαρμογή. Ως activity ορίζεται η κάθε διαφορετική οθόνη που μπορεί να παρουσιάσει μία εφαρμογή. Για παράδειγμα, όταν έχουμε ανοίξει την εφαρμογή της μουσικής θα υπάρχει διαφορετική οθόνη – activity για την λίστα των τραγουδιών και διαφορετική για την λίστα των albums. Με άλλα λόγια, μία εφαρμογή αποτελείται από διάφορα activities (μπορεί να υπάρξει και μόνο ένα activity σε μία εφαρμογή) τα οποία «συνεργάζονται» ώστε να συνθέσουν το τελικό αποτέλεσμα. Ωστόσο, το κάθε activity λειτουργεί ανεξάρτητα, γεγονός που επιτρέπει σε άλλες εφαρμογές να καλέσουν κάποιο activity – οθόνη εφόσον το επιτρέπουν.

3.5.2 Services

Τα services είναι εξίσου σημαντικά και σε αντίθεση με τα activities δεν είναι ορατά μέσω κάποιας οθόνης. Τρέχουν στο παρασκήνιο και συνήθως είναι διεργασίες που διατηρούνται ενεργές για μεγάλα χρονικά διαστήματα. Ένα service μπορεί να συνδεθεί άμεσα με κάποιο άλλο component ώστε να επιτευχθεί ένα task. Για παράδειγμα, ένας χρήστης μπορεί να λάβει ένα notification ότι έλαβε ένα email ενώ δεν είχε ενεργοποιημένη την εφαρμογή της ηλεκτρονικής αλληλογραφίας. Τα services διακρίνονται σε δύο κατηγορίες:

Started: ορίζονται τα services τα οποία μπορούν να τρέχουν στο προσκήνιο επ’ αόριστον και συνήθως εκτελούν μία διεργασία.

Bound: ορίζονται τα services τα οποία συνδέονται άμεσα με κάποιο component. Συνήθως, ένα bound service λειτουργεί ως Server σε μία επικοινωνία τύπου Client – Server και τερματίζεται όταν ολοκληρωθεί το task σε αντίθεση με τα started services.

3.5.3 Broadcast Receivers

Τα broadcast receivers είναι components τα οποία απαντούν σε broadcast messages που στέλνουν άλλες εφαρμογές ή ακόμα και το σύστημα. Πιο συγκεκριμένα, είναι components τα οποία ενεργοποιούνται όταν συμβαίνει μία ενέργεια από το σύστημα ή από κάποια άλλη εφαρμογή. Για παράδειγμα ένας broadcast receiver ενεργοποιείται όταν η συσκευή έχει χαμηλή μπαταρία ή όταν ένα αρχείο έχει κατέβει με επιτυχία.

3.5.4 Content Providers

Τα δεδομένα της εφαρμογής μπορεί να βρίσκονται στο file system, σε μία SQLite Database ή σε κάποιον Server. Οι content providers λειτουργούν ως διαχειριστές των δεδομένων μιας εφαρμογής και μέσω αυτών άλλες εφαρμογές μπορούν να αναζητήσουν τα δεδομένα αυτά ή ακόμα και να τα τροποποιήσουν εάν υπάρχει το αντίστοιχο permission. Για παράδειγμα, το Android παρέχει έναν Content Provider για την διαχείριση των στοιχείων επικοινωνίας ενός χρήστη. Ως εκ τούτου, οποιαδήποτε εφαρμογή διαθέτει τα κατάλληλα permissions μπορεί να δημιουργήσει ένα query βάσει των στοιχείων αυτών για να διαβάσει ή να τροποποιήσει δεδομένα μίας άλλης εφαρμογής.

3.6 Επικοινωνία μεταξύ Components

Η επικοινωνία των components μεταξύ τους είναι αρκετά σημαντική. Το λειτουργικό Android χρησιμοποιεί IPC (Inter-Process Communication) μηχανισμούς προκειμένου να επιτυγχάνεται η επικοινωνία μεταξύ των διεργασιών. Τα IPC είναι ουσιαστικά διεπαφές που επιτρέπουν στους developers να συντονίσουν activities μαζί με διάφορες διεργασίες που τρέχουν ταυτόχρονα στο λειτουργικό σύστημα. Η τεχνολογία αυτή επιτρέπει στο πρόγραμμα να διαχειρίζεται πολλά αιτήματα την ίδια χρονική στιγμή. Από την στιγμή που μία ενέργεια του απλού χρήστη μπορεί να οδηγήσει σε ένα σύνολο διεργασιών, αναγνωρίζουμε το πόσο σημαντικό είναι οι διεργασίες να έχουν την δυνατότητα να επικοινωνούν. Τα IPC διαθέτουν μεθόδους, οι οποίες ανάλογα την περίπτωση μπορούν να λειτουργήσουν θετικά ή αρνητικά αν υλοποιηθούν με λάθος τρόπο. Υπάρχουν τρία είδη IPC:

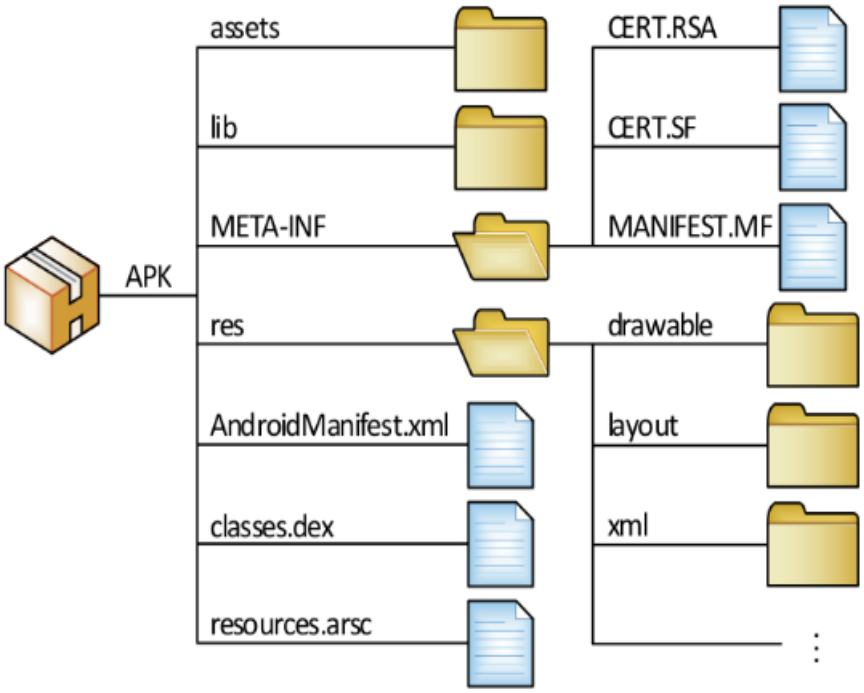
1. Intents: είναι μηνύματα που αποστέλλονται μεταξύ των components. Είναι ένας μηχανισμός ώστε να περνούν μηνύματα μεταξύ των διεργασιών (π.χ. μήνυμα για να ξεκινήσει ένα service ή ένα activity).

2. Bundles: είναι οντότητες οι οποίες περικλείουν δεδομένα.

3. Binders: είναι οντότητες οι οποίες επιτρέπουν στα permissions να αποκτήσουν τρόπο επικοινωνίας με κάποιο άλλο service ή να αποστείλουν κάποιο μήνυμα.

3.7 Android Application Package (APK)

Το Android Package ή αλλιώς «APK» είναι η compiled έκδοση μίας εφαρμογής για το λειτουργικό του Android. Στην μορφή αυτή είναι δυνατή η εγκατάσταση της εφαρμογής στην συσκευή. Εσωτερικά του apk περιλαμβάνονται όλα τα δομικά αρχεία της εφαρμογής, όπως φαίνεται στην Εικόνα 5, τα οποία παρουσιάζονται αναλυτικά στην συνέχεια.



Εικόνα 5. Η εσωτερική δομή ενός APK αρχείου

- Assets: στον φάκελο περιλαμβάνονται διάφορα αρχεία π.χ. ήχου, εικόνας κ.α. που είναι χρήσιμα για την εφαρμογή.
- Lib: στον φάκελο περιλαμβάνονται native libraries, ωστόσο η πλειοψηφία των εφαρμογών είναι υλοποιημένη σε Java.
- META-INF: στον φάκελο περιλαμβάνονται τα metadata της εφαρμογής καθώς και τα απαραίτητα certificates.
- Res: στον φάκελο περιλαμβάνονται στοιχεία τα οποία είναι απαραίτητα για την δομή της εμφάνισης της εφαρμογής.
- AndroidManifest.xml: το αρχείο περιλαμβάνει σημαντικές οδηγίες για την εφαρμογή προς το σύστημα (π.χ. permissions).
- Classes.dex: το αρχείο περιλαμβάνει τις java classes που έχουν υλοποιηθεί σε dex format.
- Resources.arsc: το αρχείο διατηρεί μία λίστα με τα resources του apk.

3.8 Android Manifest Overview

Κάθε εφαρμογή στο Android διαθέτει ένα Android Manifest.xml αρχείο το οποίο βρίσκεται στον root φάκελο του συστήματος. Το συγκεκριμένο αρχείο αποτελεί ένα από τα πιο σημαντικά στοιχεία μίας εφαρμογής διότι καθορίζει τα παρακάτω:

1. Το όνομα του java package της εφαρμογής κάτω από το οποίο υλοποιείται όλη η εφαρμογή.
2. Την περιγραφή των components της εφαρμογής, δηλαδή τα activities, services, broadcast receivers και content providers που δομούν την εφαρμογή.
3. Τις διεργασίες που θα κάνουν συμπεριλάβουν τα components της εφαρμογής.
4. Τα permissions που χρειάζεται η εφαρμογή προκειμένου να έχει πρόσβαση σε μέρη του συστήματος τα οποία είναι προστατευμένα. Ακόμα, διαθέτει permissions που άλλες εφαρμογές θα έπρεπε να έχουν σε περίπτωση που θέλουν να αποκτήσουν πρόσβαση στα δεδομένα της ίδιας εφαρμογής.
5. Το κατώτερο όριο Android API που υποστηρίζει.
6. Απαιτήσεις της εφαρμογής σε software και hardware ώστε να γίνει έλεγχος συμβατότητας με την εκάστοτε συσκευή.

4. Permission System

4.1 Application Sandboxing

Το Android προκειμένου να επιτύχει την απομόνωση της λειτουργίας κάθε εφαρμογής που είναι ενεργή, εκμεταλλεύεται το user management που παρέχουν τα Linux. Ωστόσο, η διαδικασία που ακολουθείται δεν είναι η ίδια με εκείνη που υφίσταται στις κανονικές διανομές Linux, όπου πολλές εφαρμογές μπορεί να τρέχουν κάτω από τον ίδιο χρήστη. Στην περίπτωση του Android, δημιουργείται ένα «UID» (User ID) για κάθε εφαρμογή η οποία τρέχει ως ξεχωριστή διεργασία. Συνεπώς, η κάθε εφαρμογή έχει πρόσβαση μόνο στα δικά της resources. Ο περιορισμός αυτός επιβάλλεται από το Linux Kernel.

Το UID λαμβάνει ακέραιες τιμές και το εύρος του είναι από 10000 έως 99999. Βάσει του κωδικού, η κάθε εφαρμογή λαμβάνει και την αντίστοιχη ονομασία. Για παράδειγμα αν ανατεθεί ο κωδικός 10345, τότε αντίστοιχα η ονομασία που θα λάβει θα είναι η «u0_a345». Αν μία εφαρμογή ζητήσει permissions για μία λειτουργία, τότε το αντίστοιχο «GID» (Group ID) θα ανατεθεί στην εφαρμογή. Για παράδειγμα, αν η εφαρμογή ζητήσει πρόσβαση στο διαδίκτυο, για το permission αυτό θα του ανατεθεί το GID «3003».

Οι εφαρμογές - όπως προαναφέρθηκε – εκτελούνται ξεχωριστά. Η εκτέλεση πραγματοποιείται στο Android Application Sandbox το οποίο ξεχωρίζει τα δεδομένα της εφαρμογής και του κώδικα εκτέλεσης από άλλες εφαρμογές που βρίσκονται στην συσκευή. Αυτός ο διαχωρισμός επιδρά ενισχυτικά στο κομμάτι της ασφάλειας.

4.2 Application Permissions

Ο βασικός σκοπός των permissions είναι να προστατέψουν την ιδιωτικότητα των χρηστών του Android. Για τον λόγο αυτό οι εφαρμογές θα πρέπει να ζητούν άδεια από τον χρήστη είτε αυτό αφορά ευαίσθητα δεδομένα (π.χ. επαφές) είτε κάποιο feature που παρέχει η συσκευή (π.χ. Bluetooth).

Οι Android εφαρμογές by default όταν δημιουργούνται δεν παρέχουν κανένα permission ώστε να μην μπορεί να επηρεαστεί κάποια άλλη εφαρμογή, το σύστημα ή και ο ίδιος ο χρήστης. Για τον λόγο αυτό κατά την εξέλιξη της εφαρμογής οι προγραμματιστές θα πρέπει να δώσουν την μέγιστη προσοχή στα permissions που ζητούν έτσι ώστε να μην εκτεθούν οι χρήστες από κακόβουλες ενέργειες.

4.3 Protection Levels

Τα permissions χωρίζονται σε τέσσερις βασικές κατηγορίες. Στις κατηγορίες αυτές ορίζεται το πότε θα πρέπει να δίνονται αυτόματα οι άδειες και πότε θα πρέπει να γίνεται άμεση αναφορά στον χρήστη. Οι κατηγορίες αυτές είναι:

4.3.1 Normal Permission

Δηλώνοντας στο manifest file ένα permission ως «normal» θεωρούμε ότι δεν είναι επικίνδυνο και δεν πρόκειται να επηρεάσει με κάποιον τρόπο την ιδιωτικότητα του χρήστη ή την λειτουργία άλλων εφαρμογών. Σε τέτοια περίπτωση το σύστημα δίνει αυτόματα την άδεια για την συγκεκριμένη ενέργεια. Ως normal ορίζεται by default η τιμή των νέων permissions.

4.3.2 Dangerous Permission

Δηλώνοντας στο manifest file ένα permission ως «dangerous» θεωρούμε ότι υπάρχει μεγαλύτερος κίνδυνος για την ιδιωτικότητα του χρήστη αλλά και για την εύρυθμη λειτουργία των υπόλοιπων εφαρμογών καθώς ζητείται πρόσβαση σε ευαίσθητα δεδομένα. Για τον λόγο αυτό, στην συγκεκριμένη περίπτωση ο χρήστης θα πρέπει να δώσει μη-αυτοματοποιημένη άδεια στην εφαρμογή. Σε περίπτωση που ο χρήστης δεν επιτρέψει την πρόσβαση, η εφαρμογή δεν μπορεί να έχει πρόσβαση στους πόρους που αιτήθηκε.

4.3.3 Signature Permission

Δηλώνοντας στο manifest file ένα permission ως «signature» ορίζουμε ότι μία εφαρμογή Β μπορεί να έχει πρόσβαση σε μία εφαρμογή Α μόνο στην περίπτωση που το certificate της ταυτίζεται με αυτό της Α. Σε διαφορετική περίπτωση η πρόσβαση απορρίπτεται αυτόματα. Με άλλα λόγια δύο ή περισσότερες εφαρμογές θα πρέπει να έχουν τον ίδιο δημιουργό.

4.3.4 SystemOrSignature

Δηλώνοντας στο manifest file ένα permission ως «signature|system» δηλώνουμε ότι πρόσβαση θα έχουν οι προ-εγκατεστημένες εφαρμογές της συσκευής ή εφαρμογές με το ίδιο certificate.

5. Penetration Testing

5.1 Τι είναι το Penetration Testing;

Penetration Testing, «pen-testing» ή διαφορετικά «ethical hacking» ονομάζεται η διαδικασία ελέγχου ενός υπολογιστικού συστήματος (π.χ. υπολογιστής, κινητό, tablet κ.α.), δικτύου ή web εφαρμογής προκειμένου να εντοπιστούν ευπάθειες που ένας κακόβουλος χρήστης θα μπορούσε να εκμεταλλευτεί. Η διαδικασία του penetration testing μπορεί να εφαρμοστεί είτε χειροκίνητα είτε μέσω αυτόματων εργαλείων. Σε κάθε περίπτωση, τα βήματα συνήθως είναι τα ίδια. Αρχικά γίνεται η συλλογή πληροφοριών για τον στόχο – «θύμα», η αναγνώριση entry points από τα οποία μπορεί να πραγματοποιηθεί η είσοδος στο σύστημα, η πραγματοποίηση της επίθεσης και στη συνέχεια η αναφορά των ευρημάτων.

Ο βασικός στόχος του penetration testing είναι ο έλεγχος των αδυναμιών του συστήματος. Σε πολλές περιπτώσεις γίνεται με σκοπό τον έλεγχο των πολιτικών ασφαλείας, το επίπεδο συμμόρφωσης απέναντι σε κανόνες και νομοθεσίες (π.χ. GDPR) ή την εγρήγορση μίας εταιρείας – οργανισμού σε ζητήματα ασφάλειας.

Γενικότερα, τα ευρήματα των penetration tests συγκεντρώνονται και αποστέλλονται στο υπεύθυνο τμήμα (π.χ. IT) με σκοπό την αναβάθμιση του συστήματος και την δημιουργία ή την τροποποίηση των ήδη υπαρχουσών στρατηγικών με στόχο την μέγιστη ασφάλεια.

5.2 Τύποι Penetration Testing

Υπάρχουν τρεις διαφορετικοί τύποι – τρόποι για να εξετάσει ο «επιτιθέμενος» το σύστημα που τον ενδιαφέρει και να εντοπίσει ευπάθειες:

5.2.1 Black-box Testing

Στην περίπτωση του black-box testing ο tester λειτουργεί όπως θα λειτουργούσε ένας πραγματικός κακόβουλος χρήστης. Αυτό σημαίνει ότι δεν έχει καμία πληροφορία πάνω στο σύστημα στο οποίο πρόκειται να ελέγξει, δηλαδή γνωρίζει για παράδειγμα τα outputs που μπορούν να παραχθούν αλλά όχι τον τρόπο με τον οποίο δημιουργήθηκαν.

5.2.2 White-box Testing

Στην περίπτωση του white-box testing ισχύουν τα ακριβώς αντίστροφα από το black-box testing. Ο tester έχει πλήρη γνώση του συστήματος που πρόκειται να ελέγξει και έχει στην διάθεση του τον πηγαίο κώδικα. Ο συγκεκριμένος τρόπος pentesting είναι χρήσιμος σε περιπτώσεις που γνωρίζουμε ότι υπάρχει ένα bug και θέλουμε να το εντοπίσουμε ώστε να το επιδιορθώσουμε άμεσα.

5.2.3 Gray-box Testing

Στην περίπτωση του gray-box testing ισχύει ένας συνδυασμός των προηγούμενων τρόπων. Ουσιαστικά ο tester έχει περιορισμένη πληροφόρηση σχετικά με το σύστημα που πρόκειται να εξετάσει και τις λειτουργίες του.

5.3 Γενική Μεθοδολογία

Για την διεξαγωγή του penetration test, ο tester θα πρέπει να έχει μία μεθοδολογία την οποία θα πρέπει να ακολουθήσει ώστε να μπορέσει να λάβει το μέγιστο μέσα από την διαδικασία αυτή. Η μεθοδολογία χωρίζεται σε 4 στάδια:

1. Preparation: στο πρώτο στάδιο απαιτείται από τον pentester να συλλέξει κάποια βασικά γεγονότα - δεδομένα σχετικά με την εφαρμογή που θα εξεταστεί ώστε να οδηγηθεί σε ένα επιτυχημένο test.
2. Evaluation: στο δεύτερο στάδιο ο pentester θα πρέπει να περιηγηθεί στην εφαρμογή και να εντοπίσει τυχόν entry points τα οποία θα μπορούσαν να είναι εκμεταλλεύσιμα από κάποιον κακόβουλο χρήστη.
3. Exploitation: στο τρίτο στάδιο ο pentester προσπαθεί να εκμεταλλευτεί τις ευπάθειες του συστήματος με τρόπο που δεν έχει προβλεφθεί από τον δημιουργό του.
4. Reporting: στο τέταρτο και τελευταίο στάδιο πραγματοποιείται η λεπτομερής καταγραφή των ευρημάτων.

5.3.1 Preparation

Το «Intelligence Gathering» αποτελεί ένα από τα πιο σημαντικά στάδια ενός penetration test. Το να μπορεί κανείς να εντοπίσει σημεία τα οποία οδηγούν ή μπορεί να οδηγήσουν σε ευπάθειες ξεχωρίζει και το επιτυχημένο penetration test. Το πρώτο στάδιο περιλαμβάνει τα παρακάτω βήματα:

- Open-Source Intelligence Gathering: ο pentester στο βήμα αυτό φροντίζει να συλλέξει όσες περισσότερες πληροφορίες μπορεί από δημόσιες πηγές που αφορούν την εφαρμογή που πρόκειται να εξετάσει. Ως πηγές μπορεί να είναι διάφορες μηχανές αναζήτησης, τα κοινωνικά δίκτυα κ.α.
- Architecture Understanding: αφού γίνει η συλλογή των παραπάνω πληροφοριών, θα πρέπει να γίνει μελέτη της αρχιτεκτονικής της εφαρμογής ώστε ο pentester να δημιουργήσει ένα πλάνο πιθανών κινδύνων.
- Client and Server-Side scenarios: σε κάθε περίπτωση, ο pentester θα πρέπει να μπορεί να αντιληφθεί τι τύπου εφαρμογή είναι και να είναι σε θέση να εξετάσει διάφορες πτυχές και τρόπους επικοινωνίας της εφαρμογής (π.χ. που αποθηκεύονται τα δεδομένα, με ποιες third-party εφαρμογές μπορεί να επικοινωνεί κ.α.)

5.3.2 Evaluation

Στο στάδιο του evaluation ακολουθούνται οι τεχνικές που αναλύονται παρακάτω:

- File System Analysis: ο pentester ελέγχει τα local files ώστε να εντοπίσει τυχόν παραβιάσεις ή κενά ασφάλειας.
- Reverse engineering: ο pentester θα πρέπει την compiled εφαρμογή να την μετατρέψει σε πηγαίο κώδικα που θα είναι κατανοητός από τον χειριστή. Στην συνέχεια αναλύει τον κώδικα ώστε να καταλάβει την λειτουργικότητά της και να βρει ευπάθειες.
- Static Analysis: ο pentester δεν τρέχει το πρόγραμμα. Αντίθετα, επεξεργάζεται τα αρχεία που έχει στην διάθεσή του ή τον κώδικα που απέκτησε μέσω του decompiling του αρχικού αρχείου.

- Dynamic Analysis: ο pentester στην περίπτωση αυτή πραγματοποιεί ελέγχους κατά την λειτουργία της εφαρμογής. Με αυτόν το τρόπο μπορεί να ελέγξει σε πραγματικό χρόνο τα αποτελέσματα που επιστρέφει η εφαρμογή.
- Inter-Process Communication Endpoint Analysis: ο pentester αναλαμβάνει να αναλύσει την συμπεριφορά των components της εφαρμογής (Content Providers, Activities, Broadcaster Receivers, Services)

5.3.3 Exploitation

Ο penetration tester στο σημείο αυτό αναλαμβάνει να εκτελέσει διάφορες ενέργειες ώστε να εκμεταλλευτεί – επιτεθεί στην εφαρμογή βάσει των πληροφοριών που σύλλεξε κατά την φάση του «information gathering». Μέσω των επιθέσεων, ο pentester στοχεύει στην απόκτηση υψηλών δικαιωμάτων όπως ακριβώς θα έκανε ένας κακόβουλος χρήστης.

5.3.4 Reporting

Με το πέρας του exploitation ο pentester αναλαμβάνει να συλλέξει όλες τις πληροφορίες που απέκτησε κατά την διάρκεια της προαναφερθείσας μεθοδολογίας και να συντάξει δύο reports. Το πρώτο θα είναι γραμμένο σε γλώσσα κατανοητή από την διοίκηση της εκάστοτε εταιρείας – οργανισμού με λεπτομερή περιγραφή των ευρημάτων. Το δεύτερο θα αφορά το τεχνικό κομμάτι στο οποίο θα αναφέρονται αναλυτικά οι ευπάθειες, που εντοπίστηκαν, πως μπορεί κάποιος να τις εκμεταλλευτεί και τι θα πρέπει να πραγματοποιηθεί έτσι ώστε να εξαλειφθούν.

6. Open Web Application Security Project (OWASP)

6.1 Τι είναι το «OWASP»;

Το «Open Web Application Security Project» είναι μία κοινότητα από developers που ξεκίνησε το 2001 να δημιουργεί μεθοδολογίες, εγχειρίδια και τεχνολογίες στο πεδίο της ασφάλειας των web και mobile εφαρμογών. Προκειμένου να αυξηθεί η ασφάλεια και η ετοιμότητα των developers δημιούργησαν τις «top 10 lists» οι οποίες περιγράφουν τους κορυφαίους κινδύνους σε web - mobile εφαρμογές και ανανεώνονται ανά τακτά χρονικά διαστήματα. Όσον αφορά τις λίστες αυτές, για να υπάρχει όσον δυνατόν μεγαλύτερη ακρίβεια, συλλέγονται ανά τακτά χρονικά διαστήματα δεδομένα από εφαρμογές web ή mobile σχετικά με τα vulnerabilities τα οποία αναλύονται και οδηγούν στο «top 10» της κάθε χρονιάς.

Το OWASP πέρα από τις προαναφερθείσες δημοφιλείς λίστες, είναι υπεύθυνο και για άλλα σημαντικά εγχειρήματα, όπως είναι το «OWASP Software Assurance Maturity Model» το οποίο παρέχει στρατηγικές για την αντιμετώπιση business risks σε διάφορες εφαρμογές και το «OWASP Development Guide για J2EE, ASP.NET και PHP. Ακόμα, σημαντικό είναι να αναφερθούν το «OWASP Testing Guide» που είναι οδηγός σωστών πρακτικών για ένα penetration test και το «OWASP Code Review Guide» στο οποίο περιγράφονται οδηγίες για τον έλεγχο του κώδικα.

6.2 OWASP Top 10 Mobile Risks

Παρακάτω θα αναφερθούν και θα αναλυθούν οι κορυφαίοι δέκα κίνδυνοι για τις κινητές συσκευές βάσει της λίστας του OWASP:

- M1: Improper Platform Usage
- M2: Insecure Data Storage
- M3: Insecure Communication
- M4: Insecure Authentication
- M5: Insufficient Cryptography
- M6: Insecure Authorization
- M7: Client Code Quality
- M8: Code Tampering
- M9: Reverse Engineering
- M10: Extraneous Functionality

- M1: Improper Platform Usage

Οι mobile platforms παρέχουν μία μεγάλη ποικιλία από λειτουργίες στις οποίες έχουν πρόσβαση οι developers. Ωστόσο, η χρήση τους με λανθασμένο τρόπο (π.χ. Android Intents, Permissions κ.α.) μπορεί να οδηγήσει σε ευπάθειες της εφαρμογής.

Για το λειτουργικό του Android, η συγκεκριμένη ευπάθεια προκύπτει συνήθως από τα Android Intents. Όπως προαναφέραμε, τα intents είναι messaging objects τα οποία επιτρέπουν την επικοινωνία μεταξύ πολλών λειτουργιών (π.χ. services, broadcast messages κ.α.). Καθώς υπάρχει πληθώρα από intents η πιθανότητα να υπάρξει διαρροή δεδομένων είναι αρκετά αυξημένη.

Ο οδηγός του OWASP αναφέρει την ευπάθεια αυτή ως συνηθισμένη και εύκολα εκμεταλλεύσιμη.

- M2: Insecure Data Storage

Στην περίπτωση του «Insecure Data Storage» ο επιτιθέμενος μπορεί να έχει πρόσβαση σε δεδομένα με δύο διαφορετικούς τρόπους: είτε μέσω φυσικής πρόσβασης (π.χ. κλοπή συσκευής) είτε μέσω malware.

Στην περίπτωση της φυσικής πρόσβασης, ο επιτιθέμενος μπορεί να αποκτήσει εύκολα πρόσβαση στην συσκευή συνδέοντάς την στον υπολογιστή και μέσω διάφορων προγραμμάτων είναι εφικτή η πρόσβαση στα directories των εφαρμογών.

Ο κύριος λόγος για την ύπαρξη της ευπάθειας είναι η ελλιπής γνώση των developers στο πως αποθηκεύονται 1) τα δεδομένα στην cache, 2) τα images, 3) τα Key Presses κ.α. η οποία προήλθε από τα αντίστοιχα ελλιπή documentation που παρέχονται στο επίπεδο του operation system και των development frameworks.

Ο OWASP την ορίζει ως συνηθισμένη και εύκολη ευπάθεια η οποία μπορεί να έχει σοβαρό αντίκτυπο στην ιδιωτικότητα του χρήστη.

- M3: Insecure Communication

Η περίπτωση του «Insecure Communication» προκύπτει συνήθως σε εφαρμογές αρχιτεκτονικής client – server.

Παρά την προσοχή που δίνουν οι developers στην προστασία των δεδομένων, σπανίως δίνουν προσοχή στην κρυπτογράφηση της μεταφοράς τους. Με το να μην κρυπτογραφούνται τα δεδομένα κατά την μεταφορά τους, η εφαρμογή μπορεί να τεθεί εκτεθειμένη σε επιθέσεις τύπου Man-in-the-Middle.

Ο OWASP την ορίζει ως συνηθισμένη ευπάθεια που μπορεί να την εκμεταλλευτεί εύκολα ένας κακόβουλος χρήστης και μπορεί να έχει σοβαρό αντίκτυπο.

- M4: Insecure Authentication

Η περίπτωση του «Insecure Authentication» προκύπτει όταν η εφαρμογή αποτυγχάνει κατά την διαδικασία της αυθεντικοποίησης. Για να επιτύχει ένας κακόβουλος χρήστης την μη εξουσιοδοτημένη πρόσβασή του, χρησιμοποιεί εργαλεία τα οποία κάνουν bypass την αυθεντικοποίηση η οποία συνήθως είναι ελλιπώς υλοποιημένη.

Ο OWASP την ορίζει ως συνηθισμένη και εύκολα εκμεταλλεύσιμη αδυναμία με μεγάλο αντίκτυπο στην ιδιωτικότητα του χρήστη.

- M5: Insufficient Cryptography

Η περίπτωση του «Insufficient Cryptography» προκύπτει από την αδυναμία κρυπτογράφησης ή την χρήση αδύναμων αλγορίθμων κρυπτογράφησης. Κάτι τέτοιο μπορεί εύκολο να θέσει σε κίνδυνο την ασφάλεια των πληροφοριών καθώς εύκολα ένα κακόβουλος χρήστης θα μπορεί να αποκρυπτογραφήσει τα δεδομένα.

Ο OWASP την ορίζει ως αρκετά συνηθισμένη καθώς η διαδικασία της αποκρυπτογράφησης σε αρκετές περιπτώσεις μπορεί να επιτευχθεί χωρίς μεγάλη δυσκολία.

- M6: Insecure Authorization

Η περίπτωση του «Insecure Authorization» προκύπτει παράλληλα με την διαδικασία της αυθεντικοποίησης και δημιουργείται από αδύναμους ελέγχους που πραγματοποιούνται στην μεριά του server.

Ο OWASP την ορίζει ως αρκετά συνηθισμένη ευπάθεια με μεγάλη δυσκολία στον εντοπισμό της και σημαντικό αντίκτυπο.

Οι ευπάθειες που αφορούν το authorization έχουν αποκτήσει έδαφος τα τελευταία χρόνια. Για τον λόγο αυτό ο OWASP πλέον περιλαμβάνει την ευπάθεια αυτή στην λίστα του.

- M7: Client Code Quality

Η περίπτωση του «Client Code Quality» προκύπτει από τις αδυναμίες που δημιουργούνται κατά την ανάπτυξη της εφαρμογής στον κώδικα. Ανεξαιρέτως, κάθε κώδικας κρύβει αδυναμίες είτε σε μικρή είτε σε μεγάλη κλίμακα.

Τα memory leaks και τα buffer overflows είναι τα κλασσικά παραδείγματα στην περίπτωση αυτή. Επιτρέποντας ένα buffer overflow, ο επιτιθέμενος μπορεί να αποκτήσει συνολικό έλεγχο του συστήματος γεγονός που οδηγεί και στην κλοπή δεδομένων.

Ο OWASP την ορίζει ως αρκετά σημαντική καθώς το αντίκτυπο θα είναι μεγάλο. Ωστόσο, το να εντοπίσει κάποιος τέτοιες αδυναμίες είναι αρκετά δύσκολο.

- M8: Code Tampering

Η περίπτωση του «Code Tampering» αναφέρεται στις αλλαγές που μπορεί ο επιτιθέμενος να πραγματοποιήσει στον κώδικα της εφαρμογής. Με τον τρόπο αυτό ένας κακόβουλος χρήστης μπορεί να αποκτήσει πρόσβαση σε premium λειτουργίες, παραβιάζοντας τα copyrights και κάνοντας bypass κάθε περιορισμό της εφαρμογής.

Ίσως η πιο σημαντική τροποποίηση που μπορεί να γίνει είναι η απευθείας προσθήκη malware στα binaries ή στα resources.

Ο OWASP την ορίζει ως αρκετά σημαντική καθώς το αντίκτυπο θα είναι μεγάλο. Ωστόσο, το να εντοπίσει κάποιος τέτοιες αδυναμίες είναι αρκετά δύσκολο.

- M9: Reverse Engineering

Η περίπτωση του «Reverse Engineering» αναφέρεται στην διαδικασία αποδόμησης του αρχείου της εφαρμογής με σκοπό να αποκαλυφθεί η αρχιτεκτονική της, ο κώδικας κ.α. Ωστόσο, κάτι τέτοιο μπορεί να πραγματοποιηθεί και από έναν καλόβουλο χρήστη με ερευνητικούς σκοπούς ή για penetration testing (ethical hacking).

Ο OWASP την ορίζει ως αρκετά συνηθισμένη καθώς σε όλες τις εφαρμογές μπορεί να πραγματοποιηθεί με την δυσκολία να ποικίλει ανάλογα την περίπτωση.

- M10: Extraneous Functionality

Η περίπτωση του «Extraneous Functionality» προκύπτει από τις πρόσθετες λειτουργίες που αρχικά δημιουργούνται για λόγους ευκολίας χρήσης της εφαρμογής και δεν αφαιρούνται από τους developers.

Ένα σχετικό παράδειγμα είναι η διατήρηση «developer account» το οποίο έχει πρόσβαση σε όλες τις λειτουργίες και ουσιαστικά μπορεί να χρησιμοποιηθεί ως backdoor για έναν κακόβουλο χρήστη.

Ο OWASP την ορίζει ως συνηθισμένη και εύκολα εκμεταλλεύσιμη.

6.3 OWASP Top 10 Mobile Risks – Γενικές Οδηγίες Εντοπισμού Ευπαθειών

Σημείωση: Οι παρακάτω οδηγίες είναι ενδεικτικές και ενθαρρύνεται η παράλληλη χρήση του «Mobile App Security Checklist» για καλύτερη καθοδήγηση. Το συγκεκριμένο «checklist» το κατεβάσαμε από τον σύνδεσμο:https://github.com/OWASP/owasp-mstg/blob/master/Checklists/Mobile_App_Security_Checklist-English_1.1.2.xlsx

1. M1: Improper Platform Usage

   Για την περίπτωση M1 Θα πρέπει να γίνει έλεγχος του AndroidManifest.xml αρχείου. Πιο συγκεκριμένα, ευπάθειες μπορούν να παρατηρηθούν εσωτερικά του tag <application> σε μη ασφαλή components (activities, broadcast receivers, content providers ή services). Θα πρέπει να γίνει προσεκτικός έλεγχος στα components που είναι ορισμένα ως «exported: **true**» καθώς τότε θα μπορούν να είναι εύκολα προσβάσιμα από τρίτες εφαρμογές. Σημαντικό είναι να αντιληφθούμε πότε χρειάζεται το flag ως true και πότε όχι.

   Σημείωση: Ακόμα και αν δεν υπάρχει το flag «exported» σε κάποιο component, αν εσωτερικά του ρυθμιστεί κάποιο intent, τότε by default το flag ορίζεται ως «exported: **true**»

   Αντιστοίχως, θα πρέπει να γίνει έλεγχος για τα flags «android:allowBackup» και «android:debuggable».
   allowBackup: αν το flag είναι true τότε ο επιτιθέμενος μπορεί να δεσμεύσει όποια ευαίσθητη πληροφορία υπάρχει και να την αποθηκεύσει ενώ σε χειρότερη περίπτωση μπορεί ο επιτιθέμενος να πραγματοποιήσει code injection.
   Debuggable: το flag μπορεί να είναι true και να μην αποτελεί πρόβλημα διότι το debugging κατά το στάδιο ανάπτυξης της εφαρμογής είναι βασική διαδικασία. Σε αντίθετη περίπτωση, μπορεί να αποτελέσει σημαντικός κίνδυνος καθώς ο επιτιθέμενος μπορεί να αποκτήσει πρόσβαση σε ευαίσθητη πληροφορία, να ελέγχει την ροή της εφαρμογής και να κάνει code execution.

   Ενδεικτικά εργαλεία: MobSF, Drozer, Jadx

2. M2: Insecure Data Storage

   Για την περίπτωση M2 βασικό προαπαιτούμενο είναι η συσκευή να είναι rooted. Αν χρησιμοποιούμε προσομοιωτή τότε η εικονική θα είναι rooted by default. Διαφορετικά, αν χρησιμοποιήσουμε φυσική συσκευή τότε θα πρέπει να ακολουθηθεί μία συγκεκριμένη διαδικασία[1].

[1] Digital Trends -How to root Android phones and tablets (and unroot them) https://www.digitaltrends.com/mobile/how-to-root-android/

Μέσω του adb, τα βασικά σημεία που πρέπει να ψάξουμε είναι οι αποθηκευτικοί χώροι της εφαρμογής:

- /data/data/ «όνομα φακέλου εφαρμογής»
- /sdcard/
- /sdcard1/

Ακόμα, θα πρέπει να παρατηρήσουμε τα logs μέσω της εντολής «adb logcat» για αποθήκευση ευαίσθητων δεδομένων.

Εντολή για αναζήτηση:

Readable αρχείων: find / -perm -o+r
Writable αρχείων: find / -perm -o+w

Τέλος, σημαντική είναι η αναζήτηση στον κώδικα της εφαρμογής (αφού έχει πραγματοποιηθεί το reverse engineer) των τρόπων αποθήκευσης ώστε να εντοπίσουμε κακές τακτικές διαχείρισης δεδομένων.

Ενδεικτικά εργαλεία- εντολές: adb logcat, Jadx, MobSF

3. M3: Insecure Communication

Για την περίπτωση του M3 θα πρέπει να γίνει έλεγχος για:

- Poor Handshake
- Incorrect SSL Versions
- Weak Negotiation
- Clear Text communication

Για τον έλεγχο των παραπάνω μπορεί να γίνει χρήση του Wireshark για τον εντοπισμό και διάβασμα των πακέτων και κάνοντας χρήση των παρακάτω εργαλείων μπορεί να γίνει λήψη χρήσιμων πληροφοριών.

Ενδεικτικά εργαλεία: Wireshark, sslyze[2], sslscan [3]

[2] https://github.com/nabla-c0d3/sslyze
[3] https://github.com/rbsec/sslscan

4. M4: Insecure Authentication

Για την περίπτωση του M4 θα πρέπει να γίνει χρήση:
- εργαλείων που λειτουργούν ως επιτιθέμενα proxies (π.χ. Burp Suite, ZAP),
- εργαλείων καταγραφής πακέτων (π.χ. Wireshark)

Αφού γίνει η καταγραφή των πακέτων, υπάρχουν κάποια βασικά σημεία που θα πρέπει να ελεγχθούν:

1) Session Management και Αυθεντικοποίηση μέσω attack proxy.
2) Μη ασφαλή WebView.
3) Έλεγχος για καταχώρηση των δεδομένων στον client ή στην πλευρά του server.
4) Έλεγχος για αυθεντικοποίηση κατά την κλήση components.

Ενδεικτικά Εργαλεία: Burp Suite, Wireshark, ZAP

5. M5: Insufficient Cryptography

Για την περίπτωση του M5 θα πρέπει να γίνει έλεγχος των αλγορίθμων σε όλα τα σημεία που ενδεχομένως υπάρχει κρυπτογράφηση καθώς υπάρχουν αρκετοί αλγόριθμοι που είναι σημαντικά αδύναμοι και μέσα σε δευτερόλεπτα μπορούν να αποκρυπτογραφηθούν.

Ενδεικτικά κάποια σημεία που μπορούν να ελεγχθούν είναι:
- Η μνήμη καθώς μπορεί να διαθέτει ευαίσθητα δεδομένα σε cleartext.
- Γνωστοί μη ασφαλείς Random Generators.
- Hardcoded μυστικό κλειδί.
- Μη ασφαλείς κρυπταλγόριμοι (MD4, MD5, RC2, RC4, SHA1 κ.α.).

Ενδεικτικά Εργαλεία: Jadx, MobSF, Wireshark

6. M6: Insecure Authorization

Για την περίπτωση του M6 θα πρέπει να γίνει χρήση ενός επιτιθέμενου proxy (π.χ. Burp Suite) για να γίνει έλεγχος των responses από τον server. Τα βήματα που θα πρέπει να ακολουθηθούν είναι:

- Να γίνει έλεγχος των Credentials, δηλαδή αν η εφαρμογή χρησιμοποιεί authorization tokens αντί να κάνει συνεχείς ελέγχους για username – password.
- Να γίνει έλεγχος αν η εφαρμογή επιτρέπει την πρόσβαση σε εξουσιοδοτημένους χρήστες.
- Να γίνει έλεγχος για αποθήκευση username – password τοπικά αντί για Account Manager.

Ενδεικτικά Εργαλεία: Jadx, Burp Suite

7. M7: Client Code Quality

   Για την περίπτωση M7 υπάρχουν δύο επιλογές:

   - Αν έχουμε πρόσβαση στον πηγαίο κώδικα να τον εξετάσουμε.
   - Αν δεν έχουμε να χρησιμοποιήσουμε κάποιο εργαλείο ώστε να τον αποκτήσουμε.

   Συγκεκριμένα, ενθαρρύνεται η χρήση κάποιου checklist για code reviewing ώστε να εντοπιστούν τυχόν ευπάθειες.[4]

   Ενδεικτικά Εργαλεία: Jadx, MobSF

8. M8: Code Tampering

   Για την περίπτωση M8 θα πρέπει να γίνει αρχικά Decompile του apk (π.χ. apktool, dex2jar) και στην συνέχεια θα πρέπει να γίνει έλεγχος[5]:

   - Για το αν είναι δυνατό το bypassing λειτουργιών ή το hooking μεθόδων κάνοντας χρήση των frameworks Xposed ή Frida.
   - Για το αν είναι obfuscated ο κώδικας κάνοντας αναζήτηση κρίσιμων strings μέσα στον κώδικα.

   Ενδεικτικά Εργαλεία: Substrate, Frida , Xposed, Jadx

9. M9: Reverse Engineering

   Για την περίπτωση του M9 θα πρέπει να γίνει χρήση εργαλείων για code reviewing (π.χ. Jadx). Πιο συγκεκριμένα για το M9 αφού πραγματοποιηθεί το reverse engineering θα πρέπει:

   - Να γίνει αναζήτηση για χρήσιμα strings με εργαλεία όπως το Bytecodeviewer ή JEB
   - Να γίνει έλεγχος αν ο κώδικας είναι obfuscated ώστε να γίνει χρήση de-obfuscators και να εντοπιστούν εσωτερικά πληροφορίες σημαντικές με την δομή της εφαρμογής, της επικοινωνίας με τον server, κλειδιά κρυπτογράφησης κ.α.

   Ενδεικτικά Εργαλεία: Jadx, MobSF, Bytecodeviewer, JEB

10. M10: Extraneous Functionality

    Για την περίπτωση του M10 θα πρέπει να γίνει χρήση εργαλείων για code reviewing (π.χ. Jadx) και εσωτερικά στον κώδικα θα πρέπει να γίνει αναζήτηση για λογαριασμούς (π.χ. developer account) καθώς διαθέτουν μεγαλύτερα δικαιώματα ακόμα και από root accounts.

    Ενδεικτικά Εργαλεία: Jadx, MobSF

[4] How to Do Code Review for Android - https://yalantis.com/blog/how-to-do-code-review-for-android/
[5] OWASP – Code Tampering: https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04c-Tampering-and-Reverse-Engineering.md

7. Δημιουργία Penetration Testing Περιβάλλοντος

Για την διεξαγωγή penetration testing είτε αναφερόμαστε σε Android, iOS ή web εφαρμογές, βασικό στοιχείο είναι η δημιουργία περιβάλλοντος που θα διαθέτει όλα τα εργαλεία που θα είναι κατάλληλα για τις ενέργειες του test. Ένα τέτοιο περιβάλλον είναι απαραίτητο για όποιον επιθυμεί να εξετάσει ένα πρόγραμμα ή ένα λογισμικό διότι κατά την διεξαγωγή του penetration testing είμαστε σε θέση να ελέγξουμε την συμπεριφορά του συστήματος.

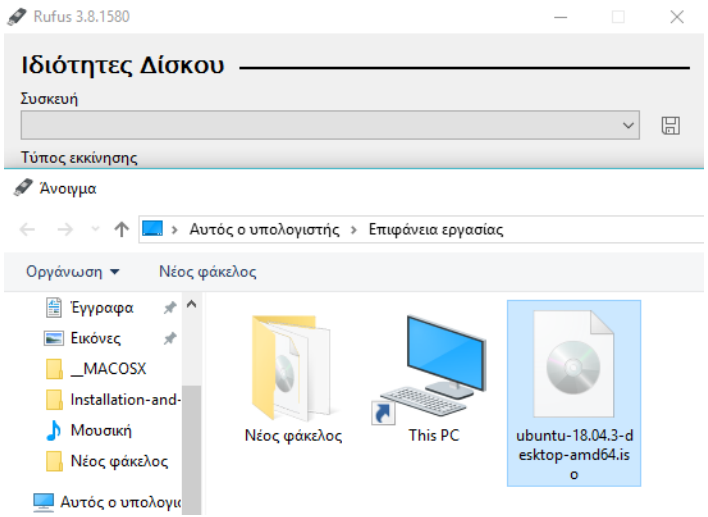
Στην συνέχεια, παρουσιάζεται μία συλλογή εργαλείων που επιλέξαμε για την δημιουργία του penetration testing περιβάλλοντος καθώς και τα βήματα για την εγκατάστασή τους.

7.1 Εγκατάσταση Ubuntu 18.04.3 LTS

Για τις ανάγκες της διπλωματικής εργασίας, ως λειτουργικό σύστημα χρησιμοποιήσαμε το Linux Ubuntu 18.04.3 LTS το οποίο είναι και η τελευταία έκδοση των Ubuntu για την χρονική στιγμή εκπόνησης της εργασίας.
Για την ομαλή εγκατάσταση του Ubuntu τα προτεινόμενα χαρακτηριστικά που πρέπει να έχει ένα μηχάνημα είναι:

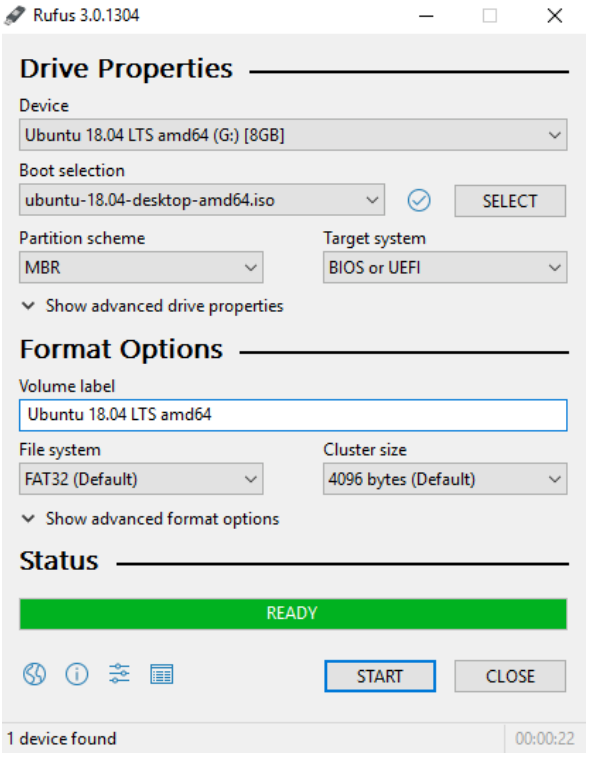
- 2GHz Dual Core processor
- 4 GB System Memory
- 25 GB Hard Drive
- USB Port για την εγκατάσταση
- Πρόσβαση στο διαδίκτυο

Αρχικά, προηγηθήκαμε στην επίσημη σελίδα του Linux Ubuntu (https://ubuntu.com/download/desktop) ώστε να κατεβάσουμε το λογισμικό σε μορφή «.iso». Στην συνέχεια μέσω του λογισμικού «Rufus» (https://rufus.ie) χρησιμοποιήσαμε ένα κενό USB ώστε να το μετατρέψουμε σε bootable.
Αφού ολοκληρώσαμε την λήψη των λογισμικών Ubuntu και Rufus, ανοίξαμε το Rufus και επιλέξαμε το «Select». Αφού εμφανίστηκε το παρακάτω παράθυρο, επιλέξαμε την τοποθεσία του .iso αρχείου που κατεβάσαμε.



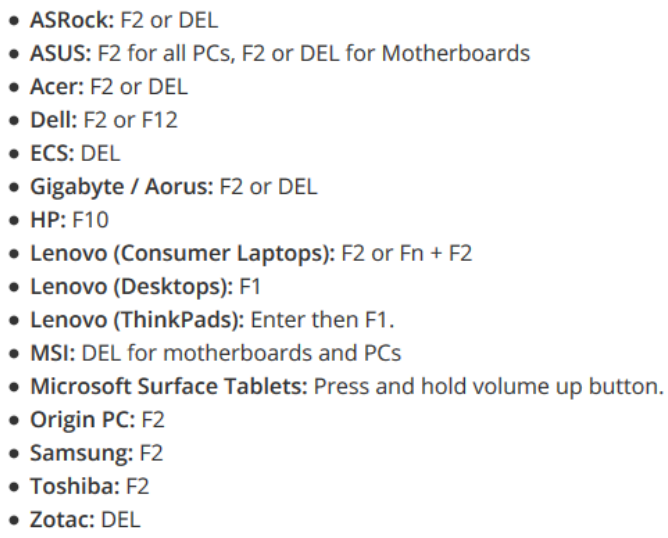
Εικόνα 6. Χρήση του λογισμικού Rufus (1/2)

Στην συνέχεια, θέσαμε τις παρακάτω ρυθμίσεις και πατήσαμε «Start» για την εκκίνηση της διαμόρφωσης του USB. Σημαντικό είναι να αναφερθεί ότι κατά την παρακάτω διαδικασία διαγράφονται όλα τα αρχεία που μπορεί να διατηρεί το USB.



Εικόνα 7. Χρήση του λογισμικού Rufus (2/2)

Μετά την ολοκλήρωση της παραπάνω διαδικασίας, ενεργοποιήσαμε το μηχάνημα στο οποίο επιθυμούσαμε να πραγματοποιήσουμε την εγκατάσταση του Ubuntu. Για να μπούμε στο BIOS, ανάλογα με τον κατασκευαστή, επιλέγουμε την αντίστοιχη επιλογή στο πληκτρολόγιο. Στο παρακάτω στιγμιότυπο, παρουσιάζονται τα κουμπιά για την ενεργοποίηση του BIOS ανάλογα με τον κατασκευαστή.

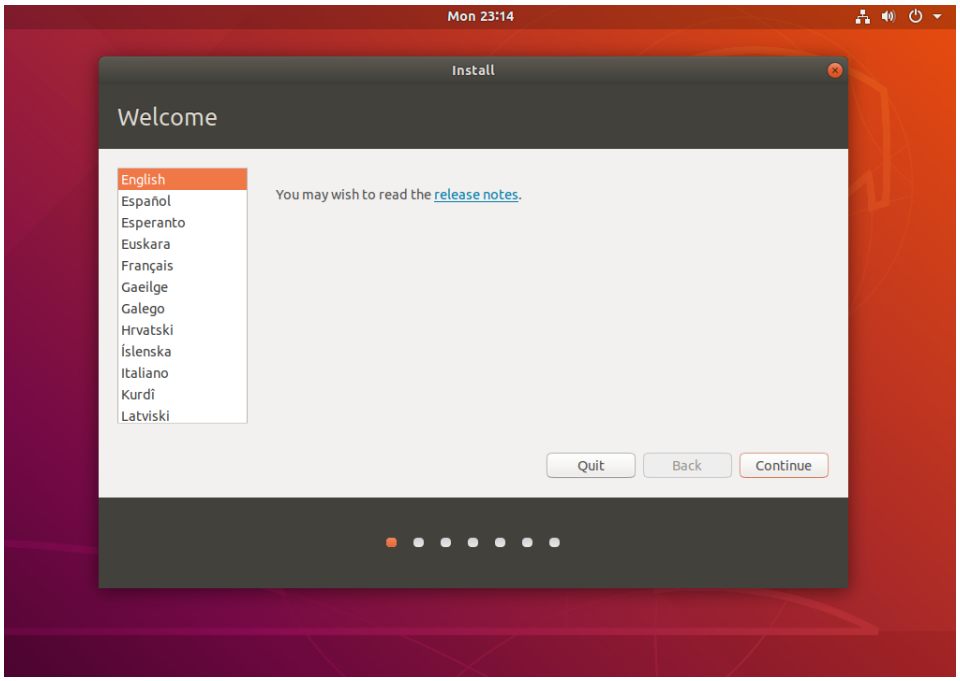


Εικόνα 8. Ενεργοποίηση BIOS για διαφορετικό κατασκευαστή

Κατά την είσοδό μας στο BIOS επιλέξαμε να κάνουμε boot από το USB που διαμορφώσαμε νωρίτερα ώστε να ξεκινήσει η διαδικασία εγκατάστασης του λογισμικού.

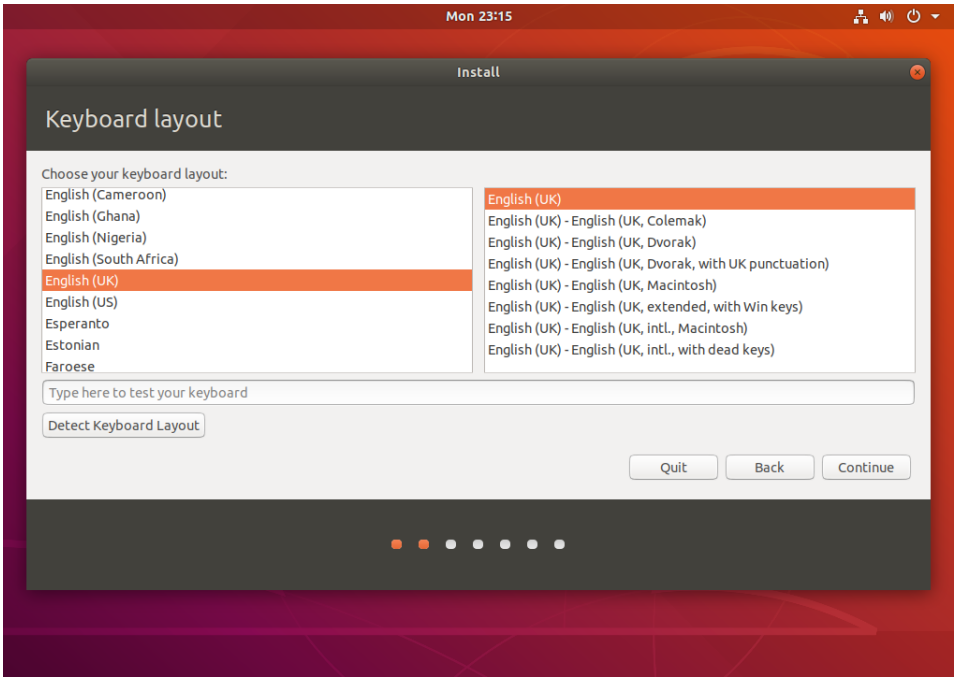
Παρακάτω, παραθέτονται τα στιγμιότυπα από κάθε βήμα εγκατάστασης του λογισμικού:

1) Επιλέγουμε την γλώσσα που επιθυμούμε.



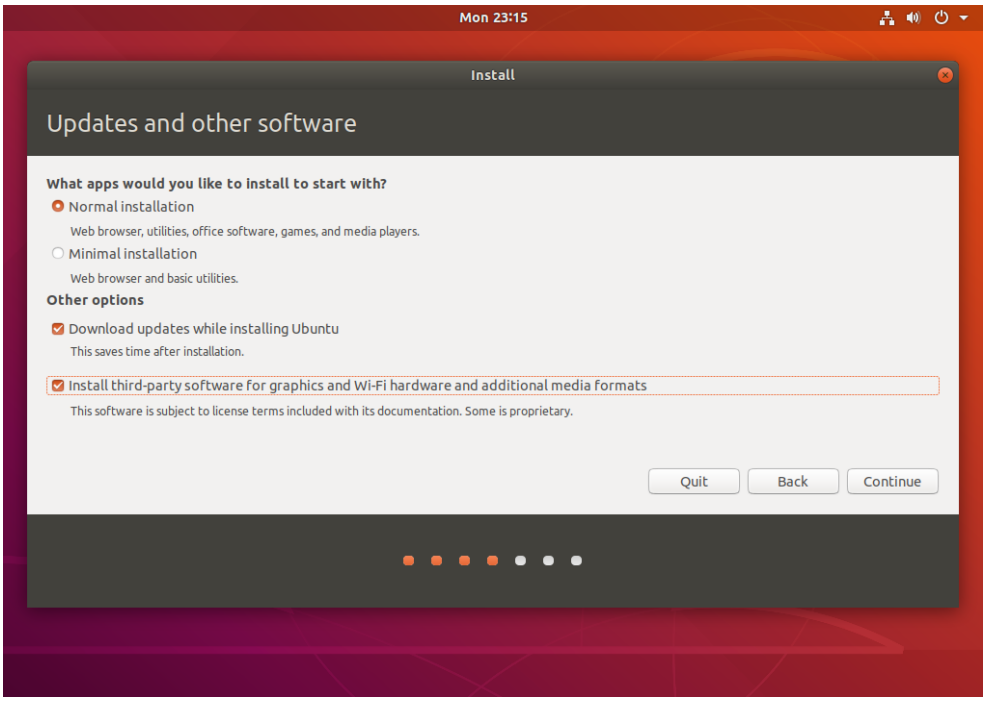
Εικόνα 9. Εγκατάσταση Ubuntu - Βήμα 1ο

2) Επιλέγουμε την default γλώσσα του πληκτρολογίου.



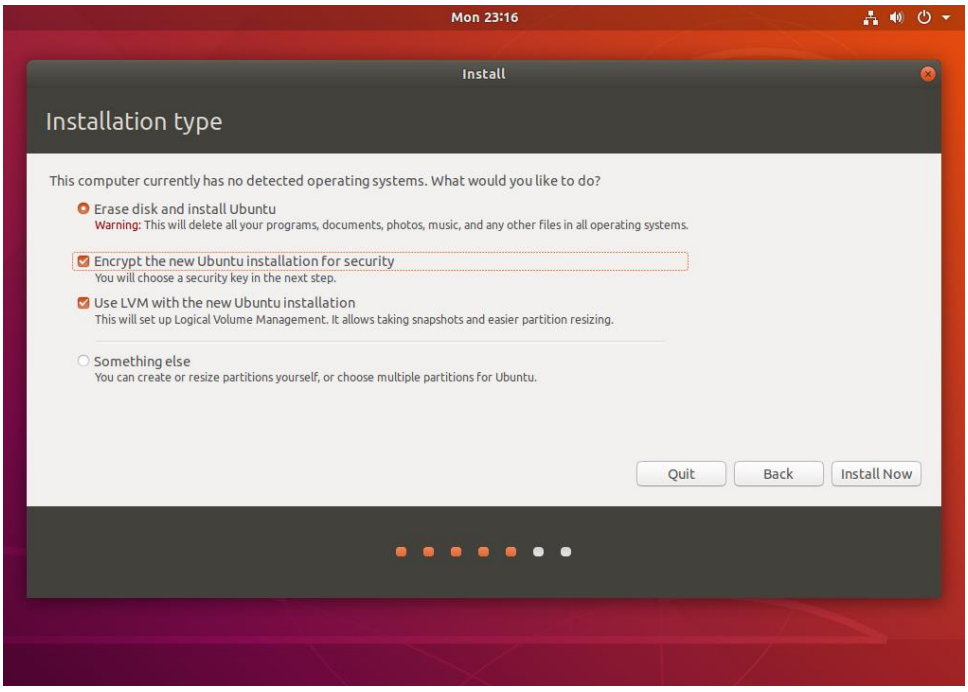
Εικόνα 10. Εγκατάσταση Ubuntu - Βήμα 2ο

3) Επιλέγουμε το είδος της εγκατάστασης. Στο σημείο αυτό αν θέλουμε επιλέγουμε να εγκατασταθούν κάποια πρόσθετα προγράμματα και updates.



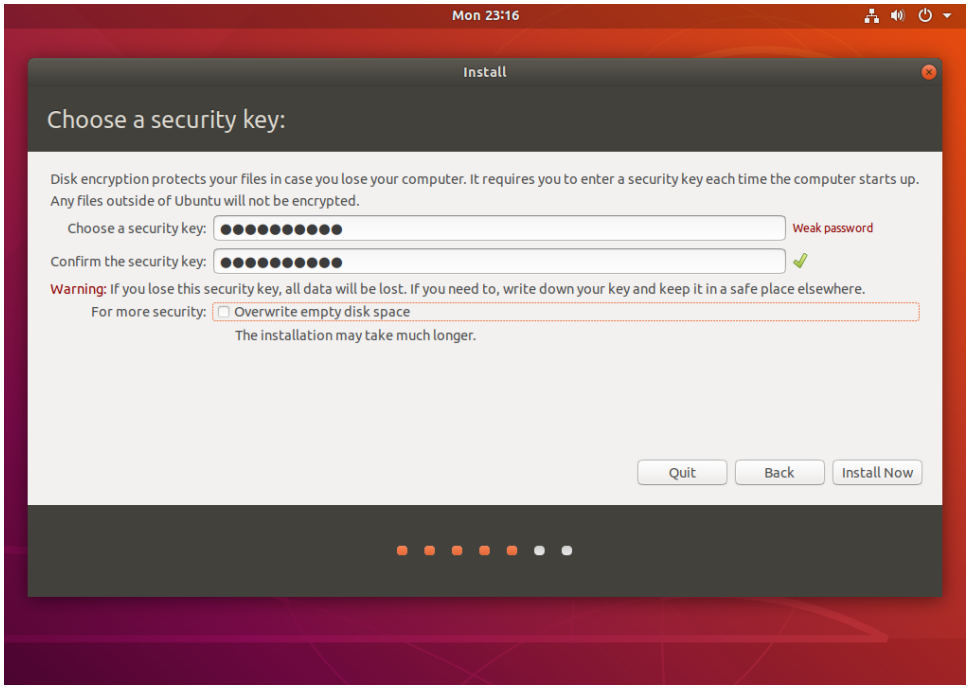
Εικόνα 11. Εγκατάσταση Ubuntu - Βήμα 3ο

4) Στον επόμενο βήμα επιλέγουμε εάν θέλουμε να γίνει πλήρης διαγραφή των δεδομένων του δίσκου ώστε να γίνει η εγκατάσταση. Αν δεν επιθυμούμε, έχουμε την επιλογή για partition στον δίσκο. Ακόμα, επιλέγουμε την κρυπτογράφηση της εγκατάστασης για λόγους ασφαλείας.



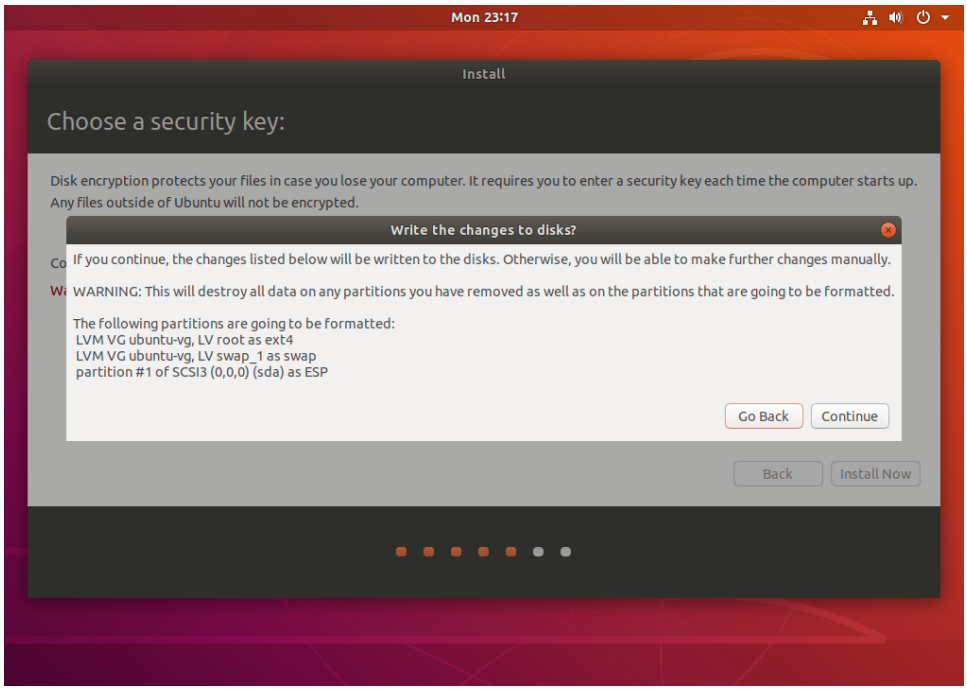
Εικόνα 12. Εγκατάσταση Ubuntu - Βήμα 4ο

5) Επιλέγουμε το security key για την κρυπτογράφηση που επιλέξαμε στο προηγούμενο βήμα. Κάθε φορά που θα θέλουμε να εκκινήσουμε το μηχάνημα, θα μας ζητείται το συγκεκριμένο κλειδί.



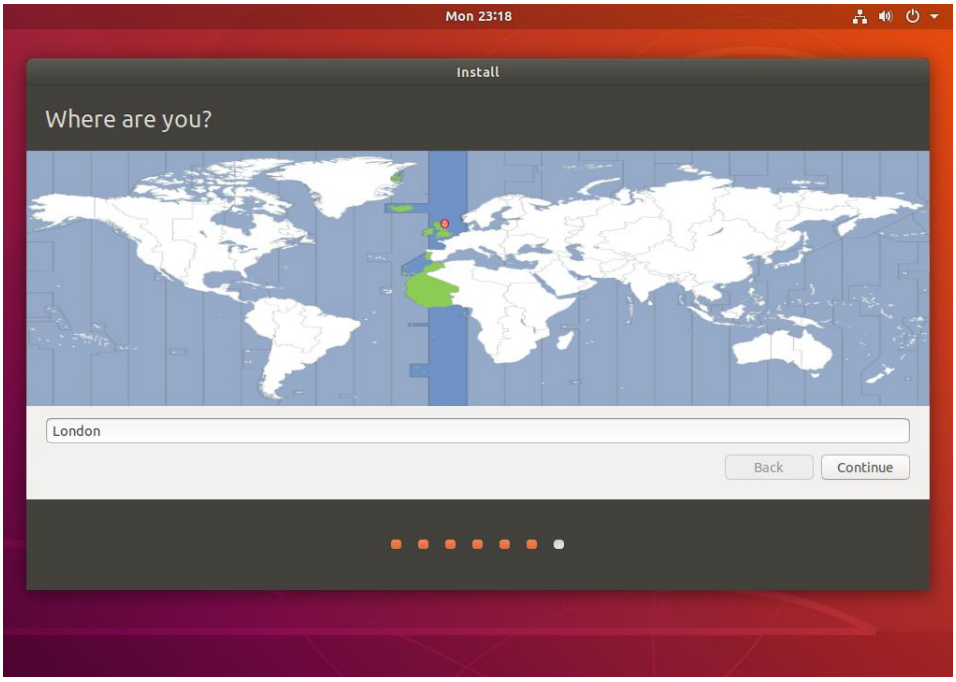
Εικόνα 13. Εγκατάσταση Ubuntu - Βήμα 5ο

6) Στην συνέχεια, ενημερωνόμαστε σχετικά με το πως θα διαμορφωθεί ο δίσκος μετά την εγκατάσταση. Αν θέλουμε να συνεχίσουμε, επιλέγουμε το «Continue».



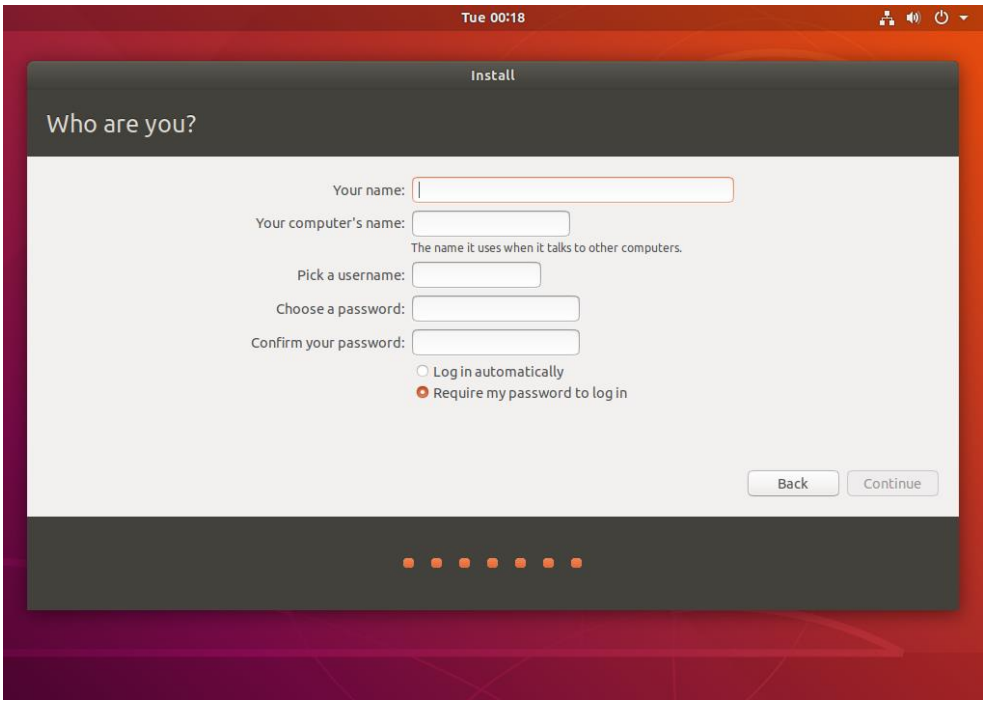
Εικόνα 14. Εγκατάσταση Ubuntu- Βήμα 6ο

7) Στο σημείο αυτό επιλέγουμε την τοποθεσία μας.



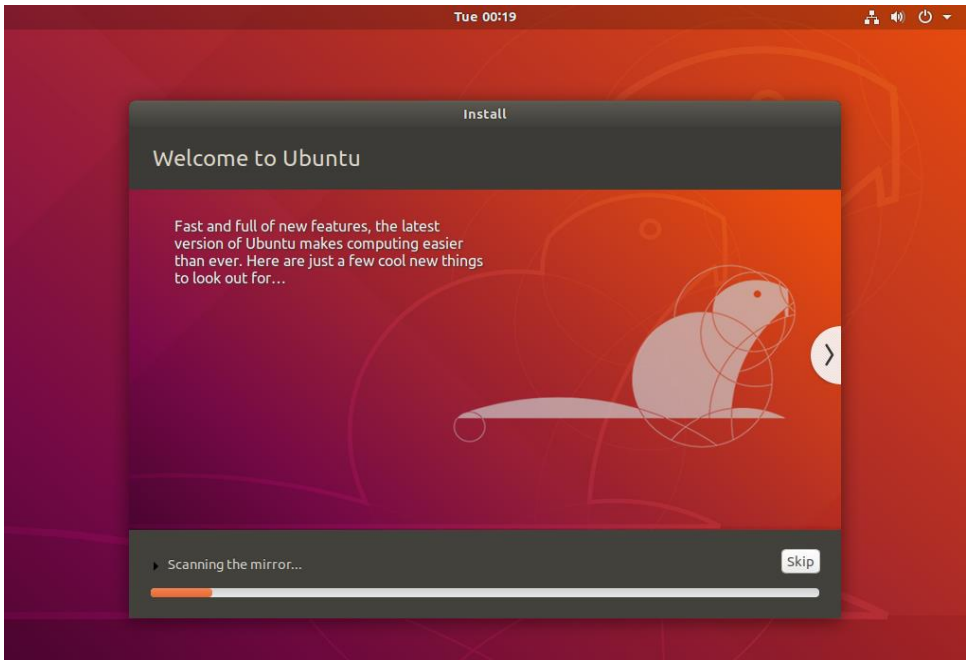
Εικόνα 15. Εγκατάσταση Ubuntu - Βήμα 7ο

8) Στην συνέχεια, συμπληρώνουμε τα στοιχεία που επιθυμούμε να εμφανίζει και να δέχεται το Ubuntu (όνομα χρήστη,username,password, ονομασία μηχανήματος).



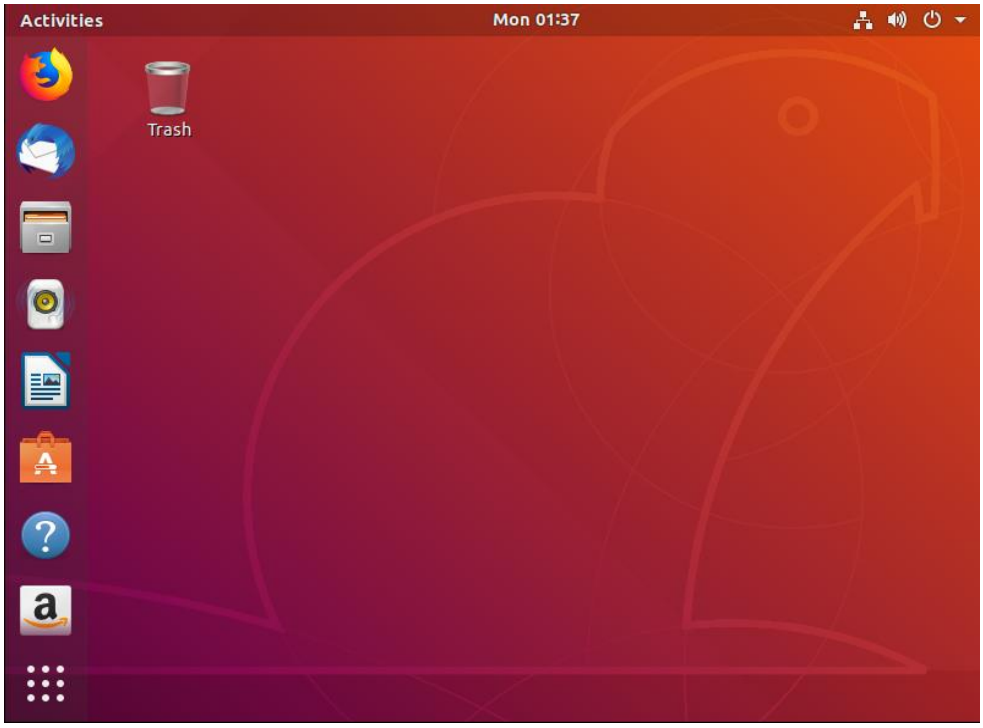
Εικόνα 16. Εγκατάσταση Ubuntu - Βήμα 8ο

9) Τέλος, θα εμφανιστεί η παρακάτω οθόνη και μετά από λίγα λεπτά η εγκατάσταση θα έχει ολοκληρωθεί.



Εικόνα 17. Εγκατάσταση Ubuntu - Βήμα 9ο

10) Στιγμιότυπο από την επιφάνεια εργασίας του Ubuntu ύστερα από την ολοκλήρωση της εγκατάστασης.



Εικόνα 18. Εγκατάσταση Ubuntu - Βήμα 10

7.2 Εγκατάσταση Προσομοιωτή Κινητού - Genymotion

Προκειμένου να ελέγξουμε κάποια εφαρμογή, θα πρέπει να επιλέξουμε ανάμεσα σε φυσική συσκευή είτε σε emulator. Για τις ανάγκες της εργασίας ακολουθήσαμε την επιλογή του emulator. Πιο συγκεκριμένα, χρησιμοποιήσαμε το εργαλείο Genymotion (https://www.genymotion.com/download/) το οποίο διατίθεται δωρεάν. Σημαντικό είναι να αναφερθεί ότι απαιτείται το VirtualBox για να μπορεί να λειτουργήσει ο emulator.

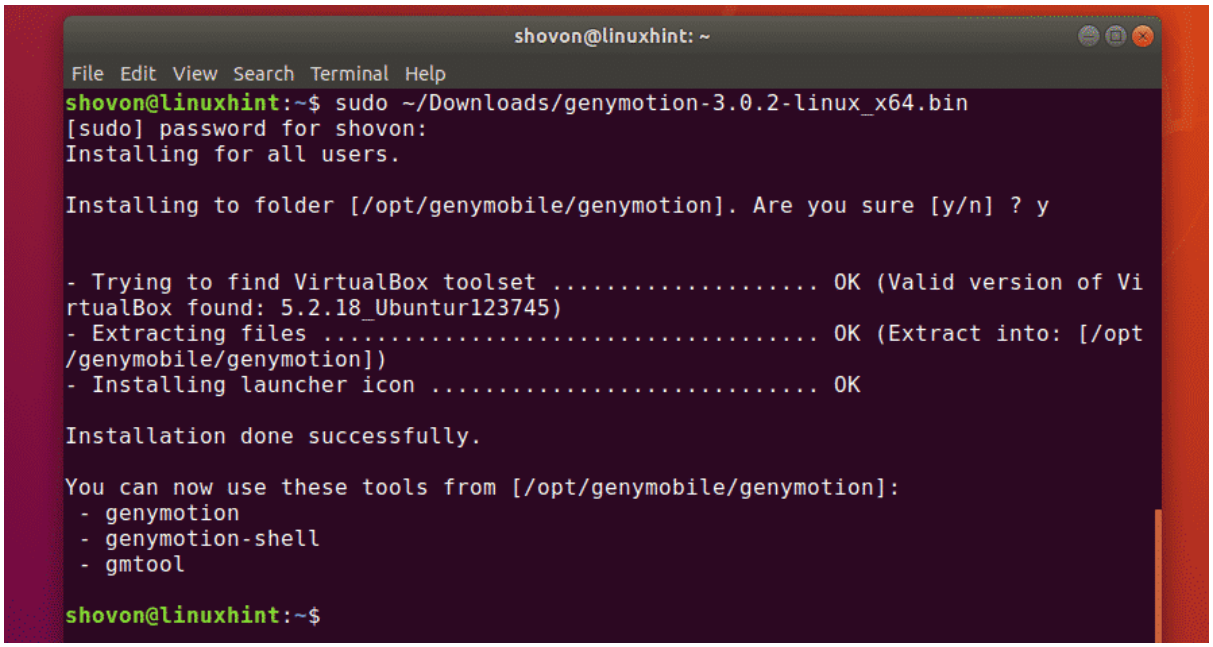
Για την ομαλή εγκατάσταση του Ubuntu τα προτεινόμενα χαρακτηριστικά που πρέπει να έχει ένα μηχάνημα είναι:

- Microsoft Windows 7, 8/8.1, 10 (32/64 bit)
- 64-bit CPU, with VT-x or AMD-V capability, enabled in BIOS settings
- Hardware accelerated GPU
- 400 MB disk space
- 4GB RAM
- VirtualBox 6.0.4

Αφού κατεβάσουμε το αρχείο από τον παραπάνω σύνδεσμο ακολουθούμε τα παρακάτω βήματα:

1) Εντοπίζουμε στον φάκελο Downloads το αρχείο που κατεβάσαμε.
2) Δίνουμε την παρακάτω εντολή ώστε το αρχείο να μετατραπεί σε εκτελέσιμο:
   chmod +x ~/Downloads/genymotion-3.0.2-linux_x64.bin
3) Στην συνέχεια εκτελούμε την παρακάτω εντολή:
   sudo ~/Downloads/genymotion$ ./genymotion

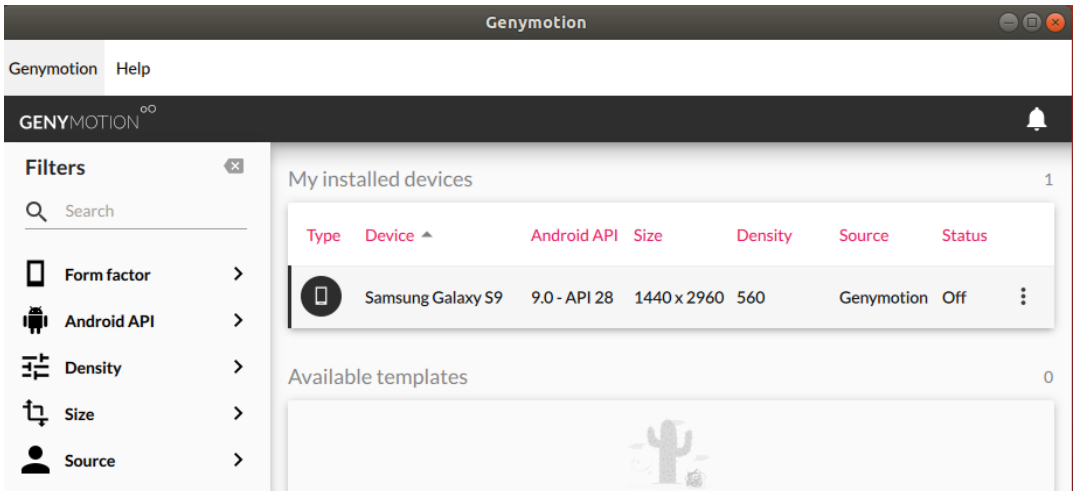
Μετά την επιτυχή εγκατάσταση του Genymotion θα πρέπει να εμφανιστούν τα παρακάτω στο terminal:



Εικόνα 19. Εγκατάσταση Genymotion (1/3)

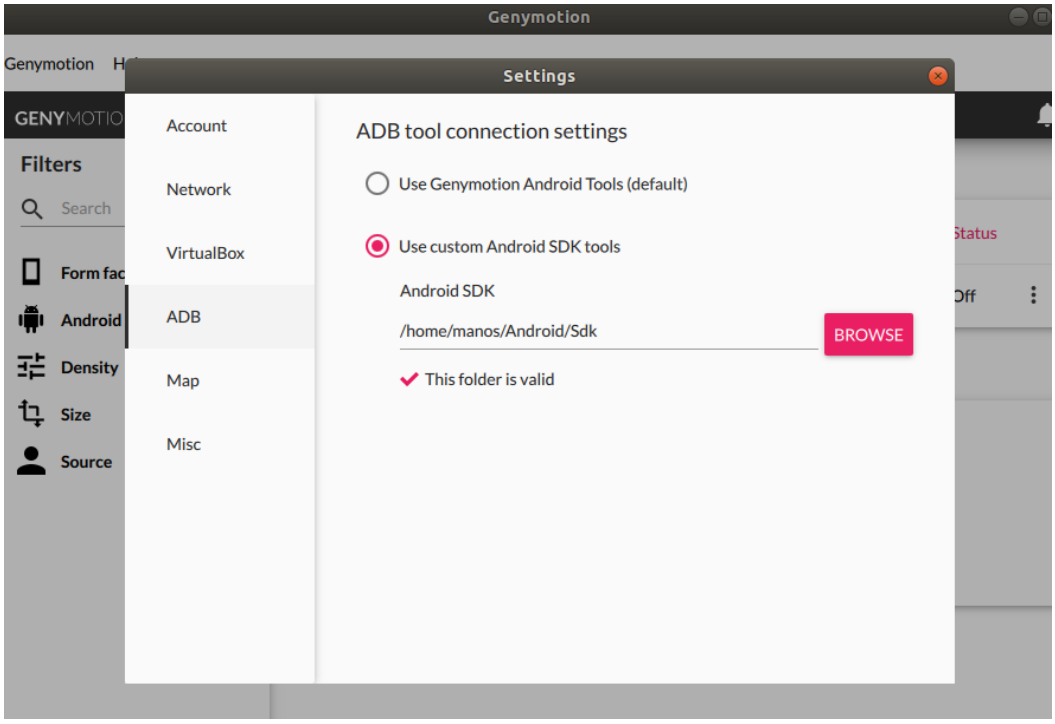
Στην συνέχεια, ανοίγουμε την εφαρμογή του Genymotion, δημιουργούμε λογαριασμό -σε περίπτωση που δεν έχουμε – και κάνουμε log in. Αφού ολοκληρώσουμε την διαδικασία, εμφανίζεται το παρακάτω παράθυρο που είναι και το βασικό πλαίσιο του Genymotion. Ανάλογα με τις προτιμήσεις μας (π.χ. Android Version, API κ.α.) επιλέγουμε για εγκατάσταση κάποια συσκευή και στην συνέχεια με διπλό κλικ ενεργοποιείται ο αντίστοιχος emulator.

Για τις ανάγκες της εργασίας, επιλέξαμε την συσκευή Samsung Galaxy S9 με Android Version 9.0 και API 28 καθώς ήταν από τις λίγες συσκευές με τα πιο πρόσφατα χαρακτηριστικά.



Εικόνα 20. Εγκατάσταση Genymotion (2/3)

Ακόμα, από το μενού του Genymotion πατήσαμε τα Settings και στην επιλογή του ADB ορίσαμε το path στο οποίο είναι αποθηκευμένο το SDK. Αν δεν το έχουμε ήδη, θα πρέπει να το κατεβάσουμε.



Εικόνα 21. Εγκατάσταση Genymotion (3/3)

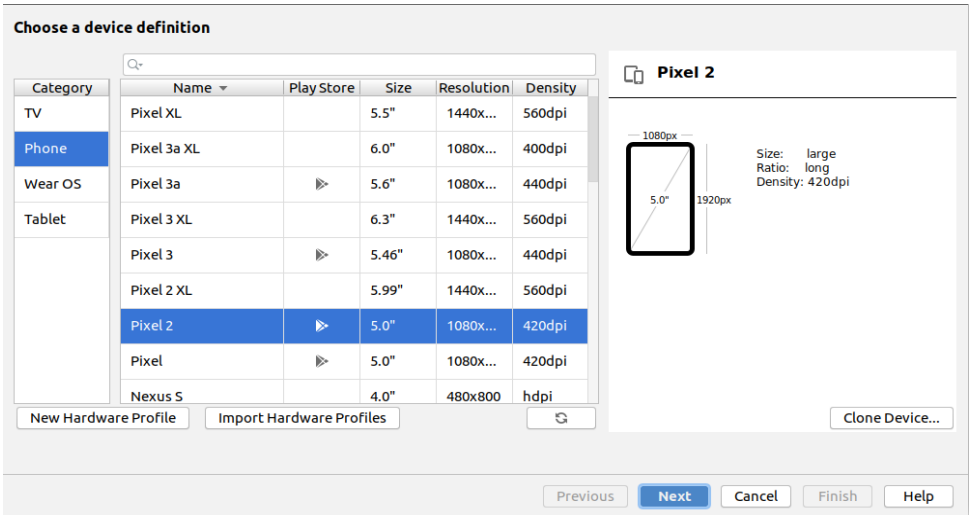
7.3 Εγκατάσταση Android Studio

Ένα βασικό εργαλείο που εγκαταστήσαμε στο penetration testing environment είναι το Android Studio το οποίο αποτελεί το επίσημο IDE για την δημιουργία Android apps. Πέρα από την δημιουργία εφαρμογών, διαθέτει emulators όπως το Genymotion για τον έλεγχο των εφαρμογών. Ακόμα, μέσω του Android Studio καθίσταται εύκολη η εγκατάσταση των SDK.

Για την εγκατάσταση τα βήματα που ακολουθήσαμε ήταν τα εξής:

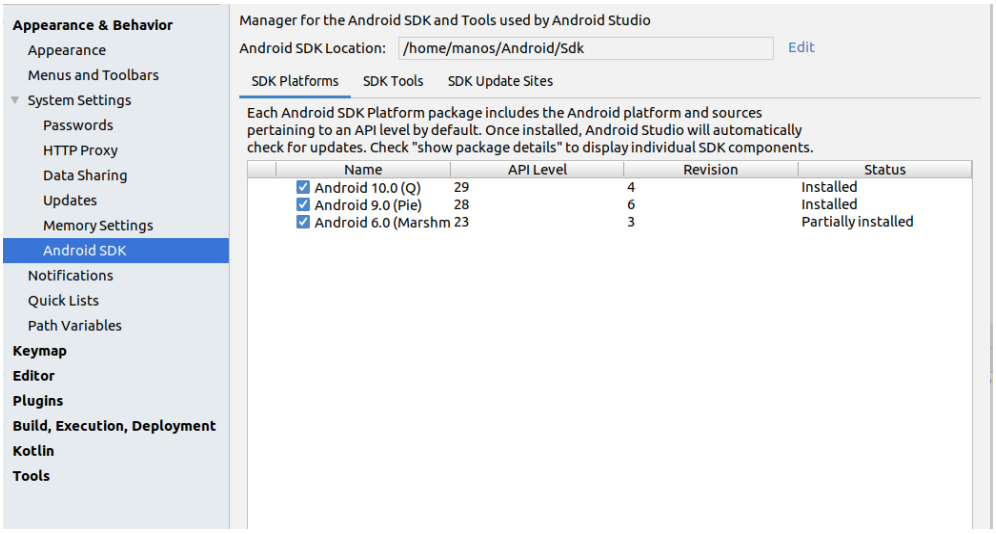
1) Επισκεφθήκαμε το default πρόγραμμα «Ubuntu Software».
2) Αναζητήσαμε το Android Studio και επιλέξαμε το «Install».
3) Αφού ολοκληρώθηκε πατήσαμε το «Launch»

Σε περίπτωση που θέλουμε να εγκαταστήσουμε κάποιον emulator ακολουθούμε το μονοπάτι: Tools -> AVD Manager -> Create Virtual Device και επιλέγουμε την συσκευή που επιθυμούμε να εγκαταστήσουμε.



Εικόνα 22. Android Studio - Επιλογή Εικονικής Συσκευής

Σε περίπτωση που θέλουμε να εγκαταστήσουμε κάποιο SDK ακολουθούμε το μονοπάτι: Tools -> SDK Manager και επιλέγουμε την version που επιθυμούμε.



Εικόνα 23. Android Studio - Εγκατάσταση SDK

7.4 Εγκατάσταση MobSF

Το MoBSF είναι ένα αυτοματοποιημένο «all-in-one» framework για pentesting σε εφαρμογές κινητών Android, Windows και iOS ικανό να πραγματοποιήσει static, dynamic και malware analysis.

Για την εγκατάστασή του αρχικά πλοηγηθήκαμε στην ιστοσελίδα του στο Github (https://github.com/MobSF/Mobile-Security-Framework-MobSF) και στην συνέχεια ακολουθήσαμε τα εξής βήματα:

1) Αντιγράψαμε το URL και στην συνέχεια μέσω του terminal εκτελέσαμε την παρακάτω εντολή ώστε να κατέβει στον φάκελό μας το MobSF:
   git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
2) Στην περίπτωση που δεν είναι εγκατεστημένο το git δίνουμε την παρακάτω εντολή:
   sudo apt get install git
3) Κατεβάσαμε την απαιτούμενη έκδοση Python (3.6-3.7) με την παρακάτω εντολή:
   sudo apt-get install python3
4) Κατεβάσαμε την απαιτούμενη έκδοση JDK (8) με την παρακάτω εντολή:
   sudo apt-get install openjdk-8-jdk
5) Κατεβάσαμε – όπως ορίζεται στο documentation – μέσω της παρακάτω εντολής τις παρακάτω προαπαιτούμενες βιβλιοθήκες:
   sudo apt install python3-venv python3-pip python3-dev build-essential libffi-dev libssl-dev libxml2-dev libxslt1-dev libjpeg8-dev zlib1g-dev wkhtmltopdf

Μετά την επιτυχή εγκατάσταση των παραπάνω εντολών, τρέξαμε την παρακάτω εντολή στον φάκελο του MobSF:
./run.sh

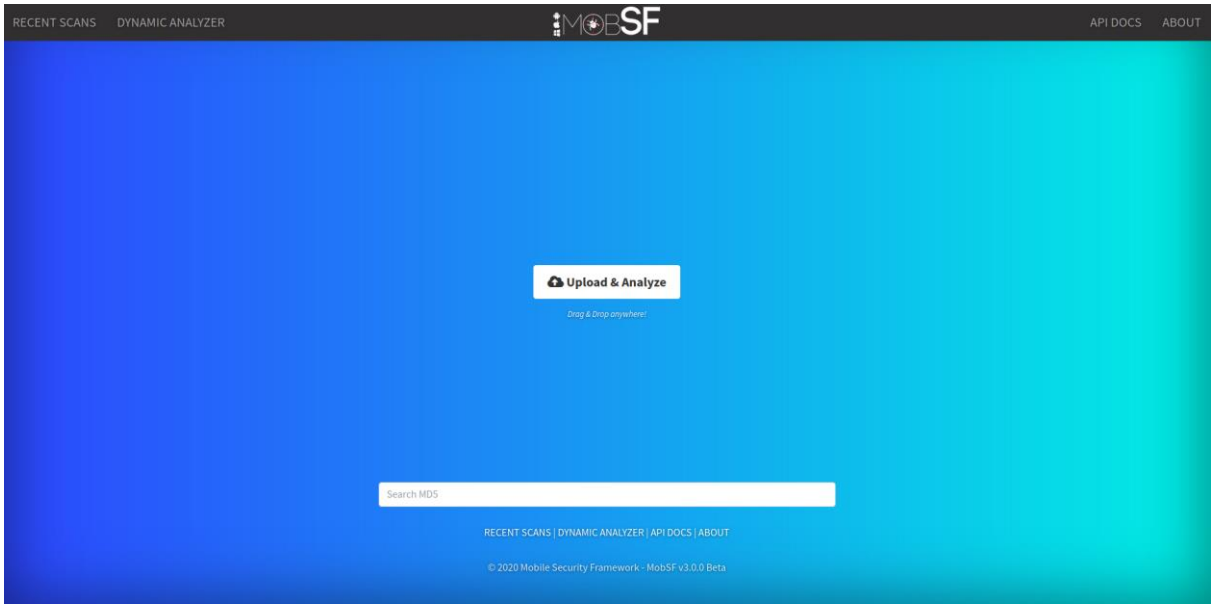
Η επιτυχής εκτέλεση του MobSF φαίνεται στο παρακάτω στιγμιότυπο

```
manos@manos:~$ cd Mobile-Security-Framework-MobSF/
manos@manos:~/Mobile-Security-Framework-MobSF$ ls
db.sqlite3          LICENSE          README.md         setup.bat       tox.ini
docker-compose.yml  LICENSES         requirements.txt  setup.sh        uploads
Dockerfile          logs             run.bat           signatures      venv
downloads           MalwareAnalyzer  run.sh            static
DynamicAnalyzer     manage.py        scripts           StaticAnalyzer
install             MobSF            secret            templates
manos@manos:~/Mobile-Security-Framework-MobSF$ ./run.sh
[2020-02-03 18:39:41 +0200] [14676] [INFO] Starting gunicorn 20.0.4
[2020-02-03 18:39:41 +0200] [14676] [INFO] Listening at: http://0.0.0.0:8000 (14
676)
[2020-02-03 18:39:41 +0200] [14676] [INFO] Using worker: threads
[2020-02-03 18:39:41 +0200] [14679] [INFO] Booting worker with pid: 14679
[2020-02-03 18:39:44 +0200] [14676] [INFO] Handling signal: winch
[2020-02-03 18:39:44 +0200] [14676] [INFO] Handling signal: winch
[2020-02-03 18:39:44 +0200] [14676] [INFO] Handling signal: winch
```

Εικόνα 24. Εγκατάσταση MobSF - Εκτέλεση μέσω Τερματικού

Τέλος, για να χρησιμοποιήσουμε το MobSF θα πρέπει να ανοίξουμε τον browser και να πληκτρολογήσουμε στο URL http://localhost:8000/. Στην συνέχεια, για να γίνει ο έλεγχος μίας εφαρμογής θα πρέπει να πατήσουμε την επιλογή «Upload & Analyze» και να αναζητήσουμε το apk που μας ενδιαφέρει.

Η σελίδα που θα εμφανιστεί θα είναι η παρακάτω:



Εικόνα 25. MobSF - Αρχική Οθόνη

7.5 Εγκατάσταση Drozer

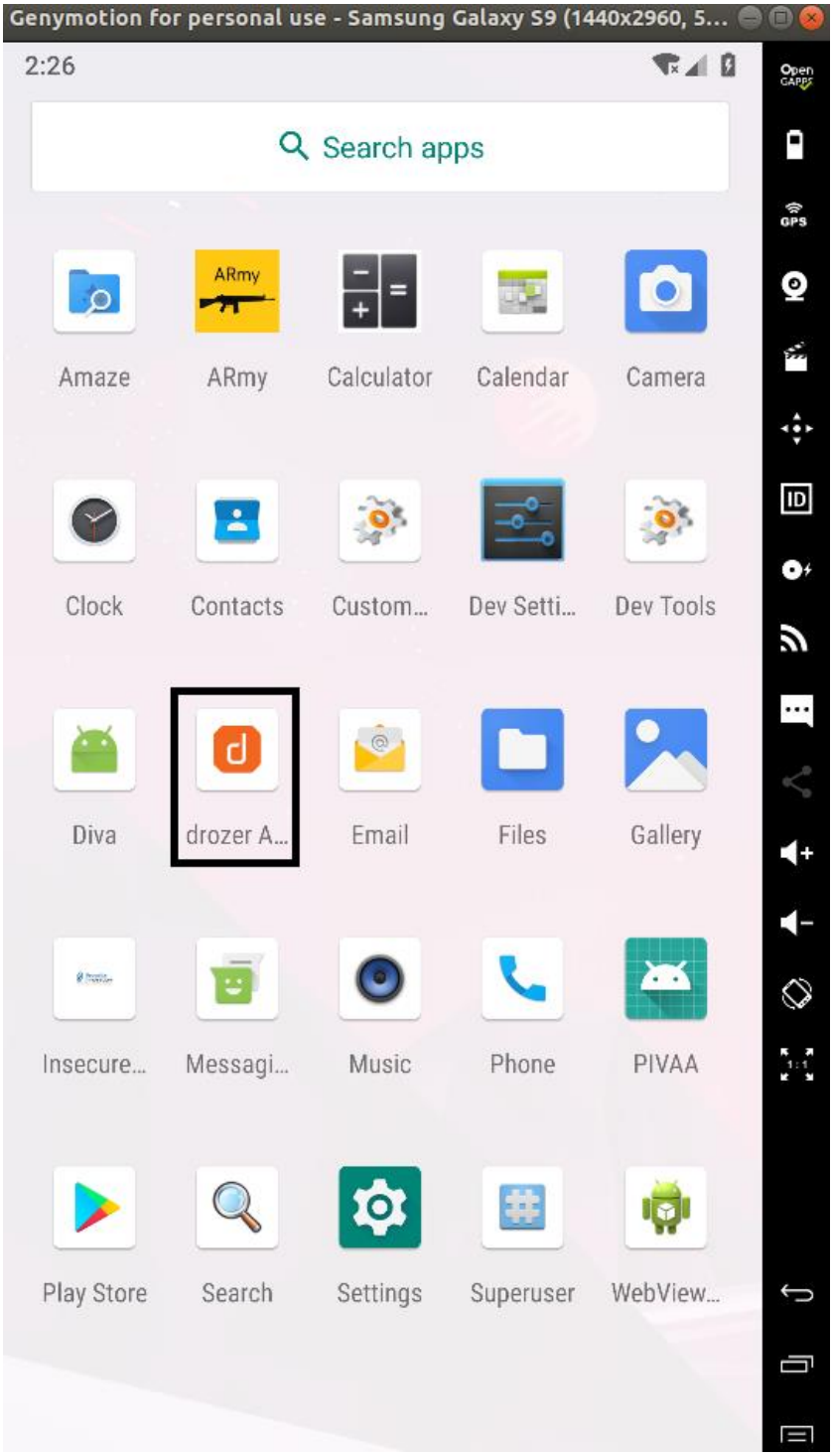
Το Drozer αποτελεί μία από τις πιο σημαντικές εφαρμογές για την αναζήτηση ευπαθειών σε εφαρμογές. Διαθέτει μεγάλη γκάμα από εντολές που επιτρέπουν στην ανακάλυψη ευπαθειών αλλά και στην εκμετάλλευσή τους.

Για την εγκατάστασή του Server αρχικά πλοηγηθήκαμε στην ιστοσελίδα του στο Github (https://github.com/FSecureLABS/drozer) και στην συνέχεια εκτελέσαμε τις παρακάτω εντολές για την εγκατάσταση των απαραίτητων προαπαιτούμενων εργαλείων:

1) **sudo apt update, sudo apt upgrade, sudo apt install python2.7 python-pip, sudo apt install python3-pip** (Για την εγκατάσταση της python 2.7)
2) **pip install protobuf** (Για την εγκατάσταση του protobuf 2.6 ή πιο πρόσφατης έκδοσης)
3) **pip install pyOpenSSL** (Για την εγκατάσταση του pyOpenSSL 16.2 ή πιο πρόσφατης έκδοσης)
4) **pip install Twisted** (Για την εγκατάσταση του Twisted 10.2 ή πιο πρόσφατης έκδοσης)
5) **sudo add-apt-repository ppa: openjdk-r/ppa, sudo apt update, sudo apt install openjdk-7-jdk** (Για την εγκατάσταση του JDK 1.7)

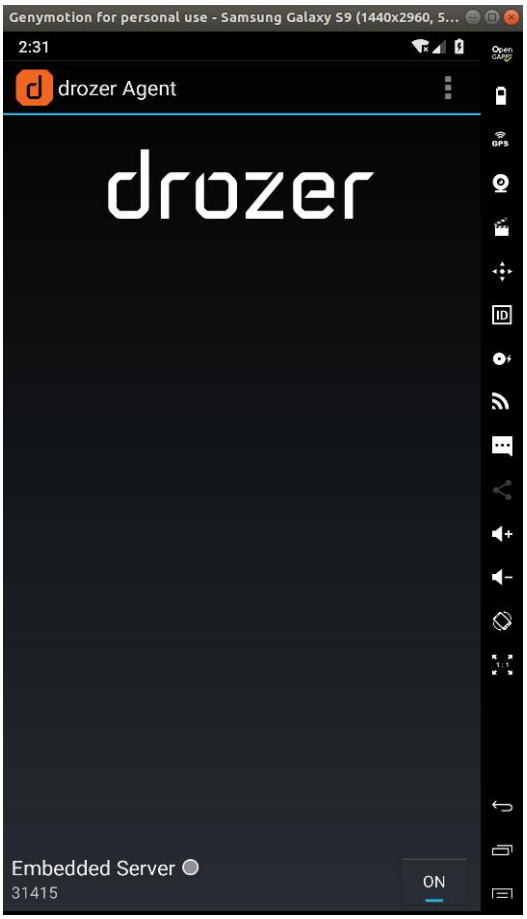
Σημείωση: Τα παραπάνω θα πρέπει να εγκατασταθούν επιτυχώς ώστε να λειτουργήσει το Drozer ενώ μεγάλη προσοχή θα πρέπει να δοθεί στην έκδοση του JDK.

Για την εγκατάσταση του Client ακολουθήσαμε τον σύνδεσμο που ακολουθεί (https://github.com/mwrlabs/drozer/releases/download/2.3.4/drozer-agent-2.3.4.apk). Αφού το αποθηκεύσαμε, ενεργοποιήσαμε τον Emulator και κάναμε «drag and drop» το apk αρχείο στην εικονική συσκευή. Με τον τρόπο αυτό πραγματοποιήσαμε την εγκατάσταση του client στον emulator όπως φαίνεται και στο παρακάτω στιγμιότυπο.



Εικόνα 26. Η εφαρμογή Drozer στο κεντρικό μενού

Στην συνέχεια, για να πραγματοποιηθεί η σύνδεση client – server εκκινήσαμε την εφαρμογή και ενεργοποιήσαμε τον server πατώντας το κουμπί «ON» όπως φαίνεται στο παρακάτω στιγμιότυπο.



Εικόνα 27. Αρχική οθόνη Drozer

Ακόμα, μέσω του terminal, για να δημιουργηθεί μεταξύ client – server ένα TCP Socket εκτελέσαμε την εντολή: **adb forward tcp:31415 tcp:31415.** Τέλος, δώσαμε την εντολή **drozer console connect** για να πραγματοποιηθεί η σύνδεση στον server. Στο terminal θα πρέπει να εμφανιστούν τα μηνύματα όπως φαίνεται και στο παρακάτω στιγμιότυπο:

```
manos@manos:~$ adb forward tcp:31415 tcp:31415
31415
manos@manos:~$ drozer console connect
Selecting e2e3502b371a3240 (unknown Samsung Galaxy S9 9)

           ..                    ..:.
          ..o..                  .r..
           ..a..  . ....... .  ..nd
             ro..idsnemesisand..pr
             .otectorandroidsneme.
          .,sisandprotectorandroids+.
        ..nemesisandprotectorandroidsn:.
       .emesisandprotectorandroidsnemes..
     ..isandp,..,rotectorandro,..,idsnem.
     .isisandp..rotectorandroid..snemisis.
     ,andprotectorandroidsnemisisandprotec.
    .torandroidsnemesisandprotectorandroid.
    .snemisisandprotectorandroidsnemesisan:
    .dprotectorandroidsnemesisandprotector.

drozer Console (v2.4.4)
dz>
```

Εικόνα 28. Σύνδεση στο Drozer μέσω τερματικού

7.6 Εγκατάσταση Jadx

Το Jadx αποτελεί μία εναλλακτική λύση για την υλοποίηση του reverse engineering. Χάρη στην λειτουργία του μπορεί - όπως και πολλά αντίστοιχα εργαλεία – να εμφανίσει τον κώδικα από το apk, να αποκωδικοποιήσει το manifest file και άλλα resources που περιλαμβάνει το αρχείο. Σημαντικό είναι να αναφερθεί ότι περιλαμβάνει de-obfuscator.

Για την εγκατάστασή του αρχικά πλοηγηθήκαμε στην ιστοσελίδα του στο Github (https://github.com/skylot/jadx) και στην συνέχεια εκτελέσαμε τις παρακάτω εντολές:

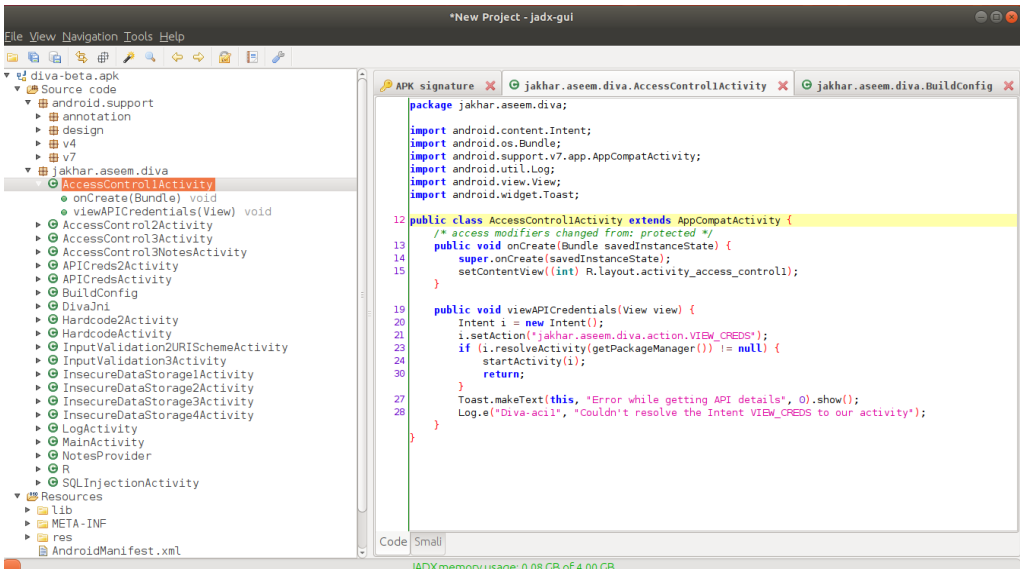
1) **git clone https://github.com/skylot/jadx.git** (Για να κατεβάσουμε το αρχείο).
2) **cd jadx** (Για να βρεθούμε στον φάκελο που εγκαταστήσαμε).
3) **./gradlew dist** (Για την εγκατάσταση)
4) **./build/jadx/bin/jadx-gui** (Για να εμφανιστεί το γραφικό περιβάλλον)

Αφού εκτελέσαμε τις παραπάνω εντολές, μας εμφανίστηκαν τα παρακάτω μηνύματα στο τερματικό:

```
manos@manos:~/jadx$ ./gradlew dist
To honour the JVM settings for this build a new JVM will be forked. Please consider using the daemon: htt
ps://docs.gradle.org/6.0.1/userguide/gradle_daemon.html.
Daemon will be stopped at the end of the build stopping after processing

> Configure project :
jadx version: dev

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.0.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 37s
18 actionable tasks: 18 up-to-date
manos@manos:~/jadx$ ./build/jadx/bin/jadx-gui
Gtk-Message: 22:19:24.539: Failed to load module "canberra-gtk-module"
INFO   - output directory: diva-beta
INFO   - loading ...
```

Εικόνα 29. Εκτέλεσε Jadx μέσω terminal

Στην συνέχεια εμφανίστηκε το παρακάτω γραφικό περιβάλλον στο οποίο επιλέξαμε το apk που θέλουμε να επεξεργαστούμε.



Εικόνα 30. Γραφικό περιβάλλον Jadx

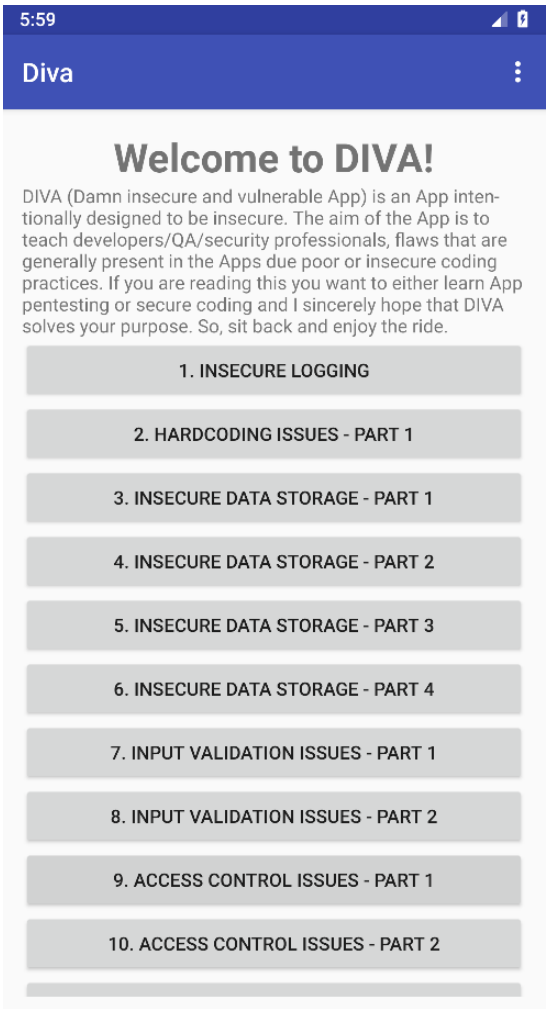
8. Penetration Testing Case Study: Damn Insecure and Vulnerable Application (DIVA)

Για τις ανάγκες της εργασίας, ως Case Study για το Penetration Testing επιλέξαμε την εφαρμογή «Damn Insecure and Vulnerable Application». Όπως υποδηλώνει και το όνομά της είναι μία εφαρμογή εξοπλισμένη με διάφορα vulnerabilities με σκοπό να εκπαιδεύσει νέους pentesters.

Η εφαρμογή δεν θυμίζει τις κλασσικές εφαρμογές που συναντάμε στο Google Play Store καθώς είναι χωρισμένη σε challenges με σκοπό τον εντοπισμό των αδυναμιών που διαθέτει.

Αρχικά, κατεβάσαμε την εφαρμογή από τον σύνδεσμο που ακολουθεί (https://payatu.com/wp-content/uploads/2016/01/diva-beta.tar.gz) και στην συνέχεια, αφού ενεργοποιήσαμε τον emulator, κάναμε «drag and drop» το apk στην εφαρμογή. Αφού εγκαταστάθηκε με επιτυχία, ανοίξαμε την εφαρμογή.

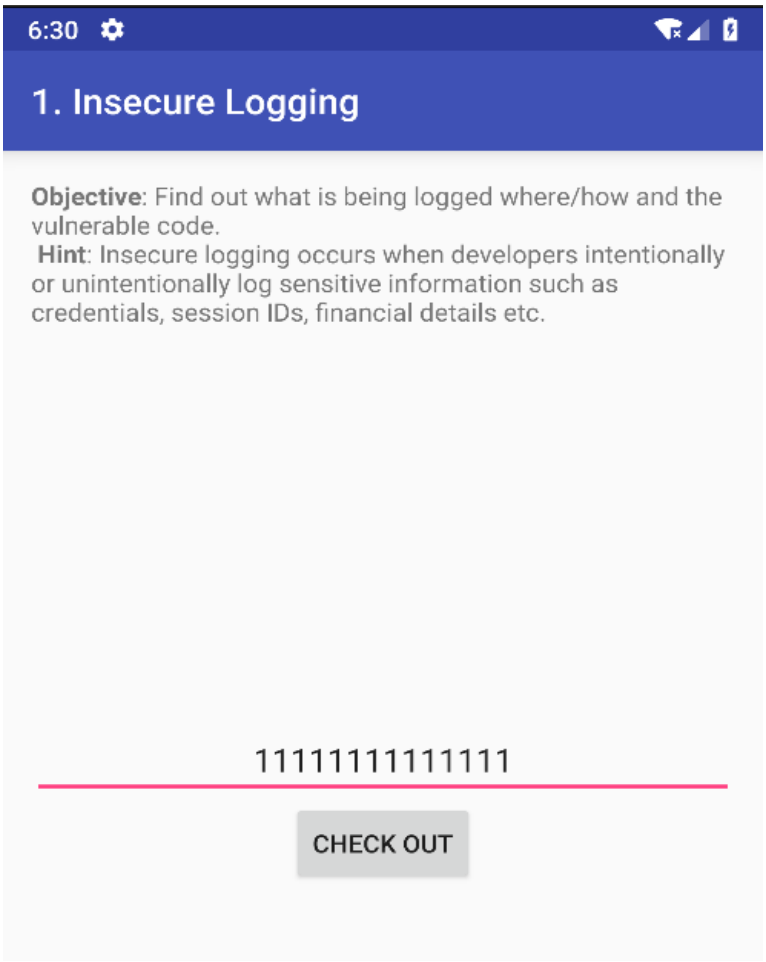
Στο παρακάτω στιγμιότυπο φαίνεται το main activity της εφαρμογής το οποίο διαθέτει τις επιλογές για την κάθε αδυναμία.



Εικόνα 31. Αρχική οθόνη εφαρμογής DIVA

8.1 Insecure Logging

Αφού επιλέξαμε την πρώτη ευπάθεια μας εμφανίστηκε η παρακάτω οθόνη. Όπως διαβάσαμε στην περιγραφή, έπρεπε να αναζητήσουμε το που αποθηκεύονται τα όσα πληκτρολογούμε, όπως είναι τα credentials, session IDs κ.α.
Για να ελέγξουμε τα logs πληκτρολογήσαμε μία σειρά από άσσους ώστε να είναι ευδιάκριτοι όταν θα αναζητούσαμε τα Logs.



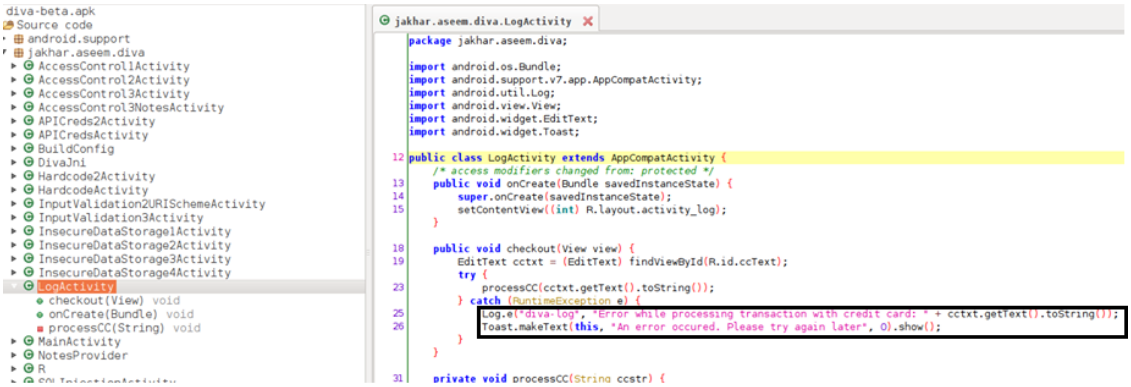
Εικόνα 32. Insecure Logging - Αρχική Οθόνη

Για να ελέγξουμε ότι πράγματι έγινε εσφαλμένα καταχώρηση στα logs, χρησιμοποιήσαμε το adb μέσω του terminal δίνοντας την εντολή **adb shell** για αποκτήσουμε shell και στην συνέχεια πληκτρολογήσαμε την εντολή **logcat** η οποία μας επέστρεψε - αφού πατήσαμε το «Check Out» - την παρακάτω καταχώρηση.

```
02-03 18:37:34.709  1793  1793 E diva-log: Error while processing transaction with credit card: 1111111111111
02-03 18:37:35.054  1793  1832 E EGL_emulation: tid 1832: eglSurfaceAttrib(1354): error 0x3009 (EGL_BAD_MATCH)
02-03 18:37:35.054  1793  1832 W OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR on surface 0xe33e3e00, error=EGL_
```

Εικόνα 33. Καταχώρηση της εισόδου ως plaintext στα Logs

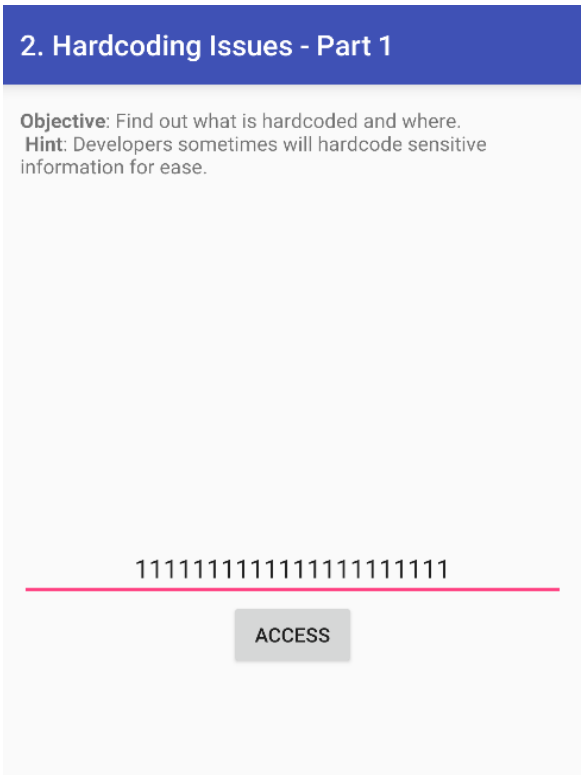
Στην συνέχεια, για να εντοπίσουμε τον κώδικα που προκαλεί την παραπάνω ευπάθεια, χρησιμοποιήσαμε το εργαλείο Jadx. Όπως φαίνεται παρακάτω στην κλάση «LogActivity» βρίσκεται ο κώδικας που καταγράφει ως text στα Logs όσα καταχωρεί ο χρήστης.



Εικόνα 34. Jadx - Insecure Logging - Κώδικας Ευπάθειας

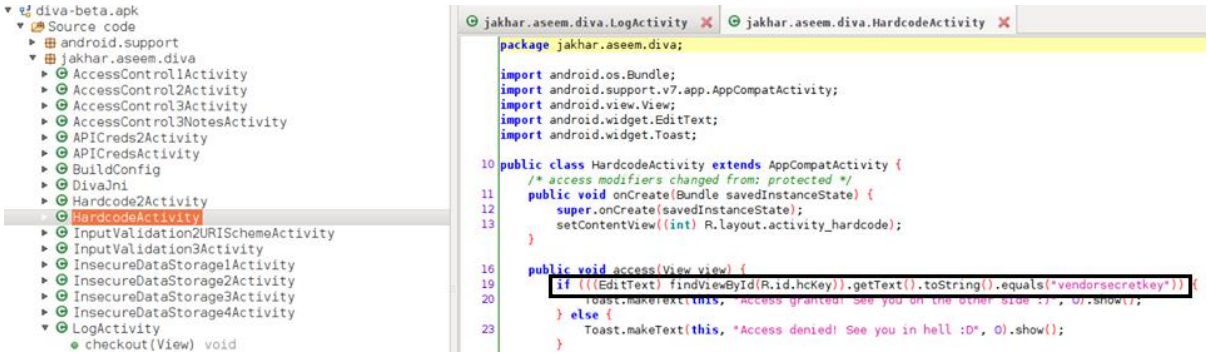
8.2 Hardcoding Issues

Για την επόμενη ευπάθεια συνεχίσαμε από το κεντρικό μενού στην δεύτερη επιλογή. Όπως αναφέρεται και στο παρακάτω στιγμιότυπο, θα πρέπει να βρούμε το που γίνεται το hardcode πληροφοριών ώστε να αποκτήσουμε πρόσβαση στην παρακάτω οθόνη. Πληκτρολογώντας μία σειρά από άσσους δεν καταφέραμε να αποκτήσουμε πρόσβαση όπως ήταν λογικό.



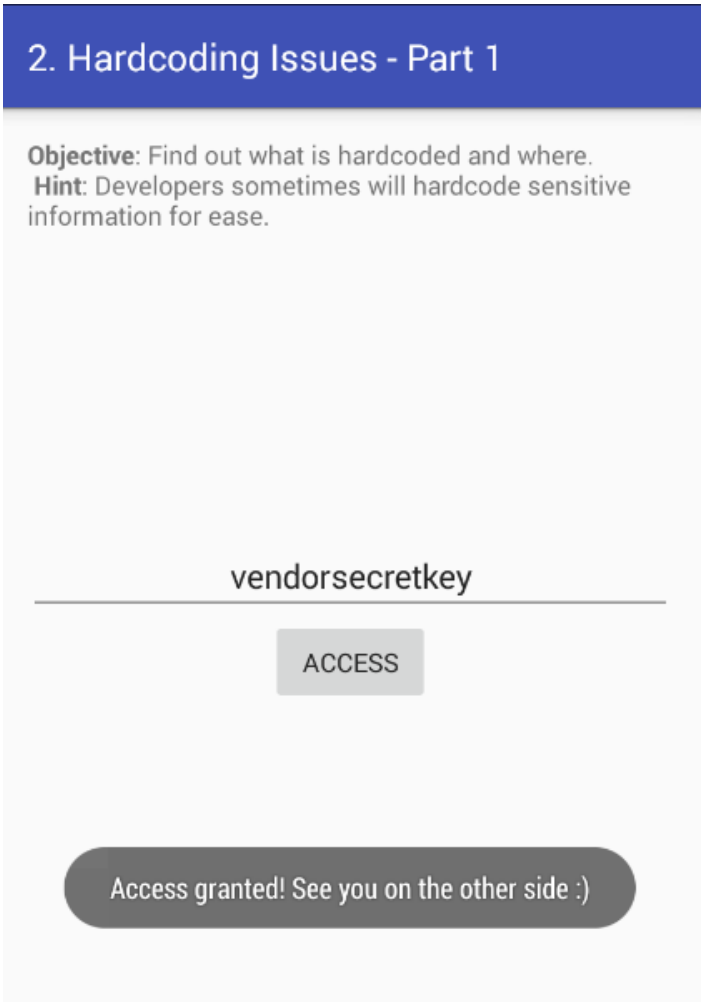
Εικόνα 35. Hardcoding Issues Part 1 - Αρχική Οθόνη

Όπως και πριν, έτσι και σε αυτό το βήμα θα πρέπει να ελέγξουμε τον πηγαίο κώδικα. Μέσω του Jadx εργαλείου θα αναζητήσουμε την κλάση που σχετίζεται με την παραπάνω ευπάθεια. Πράγματι, στην κλάση HardcodeActivity εντοπίσαμε αποθηκευμένο σε plaintext τον κωδικό, ο οποίος είναι το «vendorsecretkey»



Εικόνα 36. Jadx - Hardcoding Issues Part1 - Κώδικας Ευπάθειας

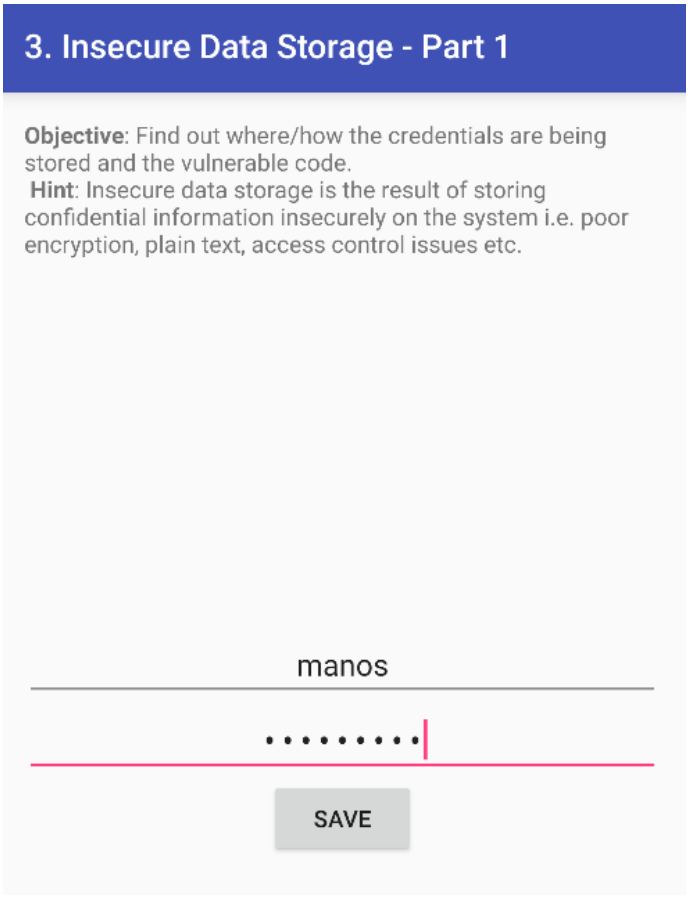
Επιστρέφοντας στην οθόνη για να εισάγουμε τον κωδικό που εντοπίσουμε, βλέπουμε όπως φαίνεται και παρακάτω, πως αποκτήσαμε πρόσβαση.



Εικόνα 37. Hardcoding Issues Part 1 - Επιτυχής Είσοδος

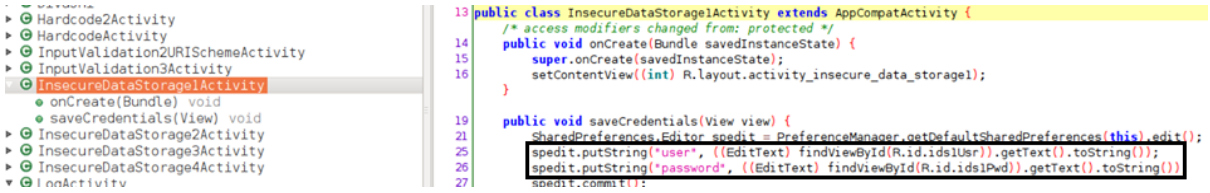
8.3 Insecure Data Storage – Part 1

Για την συγκεκριμένη ευπάθεια όπως εξηγεί η περιγραφή, θα πρέπει να εντοπίσουμε το πως και που αποθηκεύονται τα credentials. Για να το ελέγξουμε, συμπληρώνουμε τα παρακάτω πεδία με στοιχεία που επιλέξαμε (manos -123456789) ώστε να δούμε αν θα μπορέσουμε να τα εντοπίσουμε.



Εικόνα 38. Insecure Data Storage Part 1 - Αρχική Οθόνη

Για να μπορέσουμε να καταλάβουμε καλύτερα το που αποθηκεύονται τα δεδομένα, αναζητούμε την αντίστοιχη κλάση για την παραπάνω οθόνη. Μέσω του Jadx εντοπίσαμε την κλάση InsecureDataStorageActivity1 στην οποία όπως φαίνεται παρακάτω, αποθηκεύονται σε plaintext ο user και το password στα SharedPreferences.



Εικόνα 39. Jadx - Insecure Data Storage Part 1 - Κώδικας Ευπάθειας

Για να έχουμε πρόσβαση στα SharedPreferences συνδεόμαστε μέσω του adb shell. Στην συνέχεια δίνουμε τις παρακάτω εντολές ώστε να εντοπίσουμε το πακέτο της εφαρμογής μας:

1) cd data/data (Για να βρούμε τα πακέτα όλων των εφαρμογών)
2) ls (Για να δούμε τα περιεχόμενα και να εντοπίσουμε την εφαρμογή μας)
3) cd jakhar.aseem.diva (Για να βρεθούμε στο πακέτο της εφαρμογής)
4) ls (Για να βρούμε τα περιεχόμενα του πακέτου)
5) cd shared_prefs (Για να δούμε τι αποθηκεύτηκε)

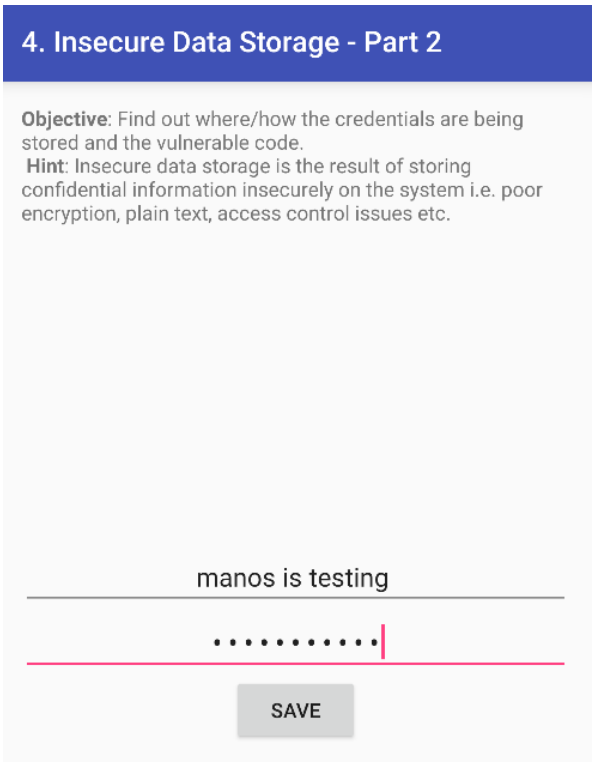
Στο παρακάτω στιγμιότυπο μπορούμε να δούμε πως πράγματι, τα credentials αποθηκεύτηκαν ως plaintext.

```
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs # ls
jakhar.aseem.diva_preferences.xml
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs # cat jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">123456789</string>
    <string name="user">manos</string>
</map>
```

Εικόνα 40. Insecure Data Storage Part1 - Εντοπισμός Ευπάθειας μέσω τερματικού

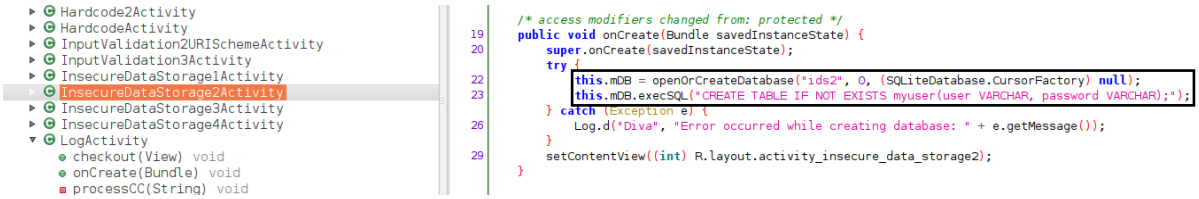
8.4 Insecure Data Storage – Part 2

Όπως και στο προηγούμενο βήμα, θα πρέπει να βρεθεί το που και πως αποθηκεύονται τα credentials της εφαρμογής.



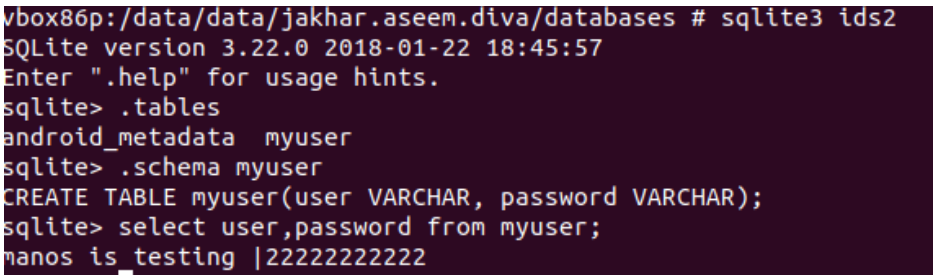
Εικόνα 41. Insecure Data Storage Part 2 - Αρχική Οθόνη

Ομοίως, αναζητούμε την αντίστοιχη κλάση, στην οποία θα μπορούμε να εντοπίσουμε το που γίνεται η καταχώρηση των δεδομένων. Στο παρακάτω στιγμιότυπο φαίνεται η κλάση InsecureDataStorage2Activity η οποία υποδηλώνει ότι τα δεδομένα αποθηκεύονται στην SQLite Database.



Εικόνα 42. Jadx – Insecure Data Storage Part 2 – Κώδικας Ευπάθειας

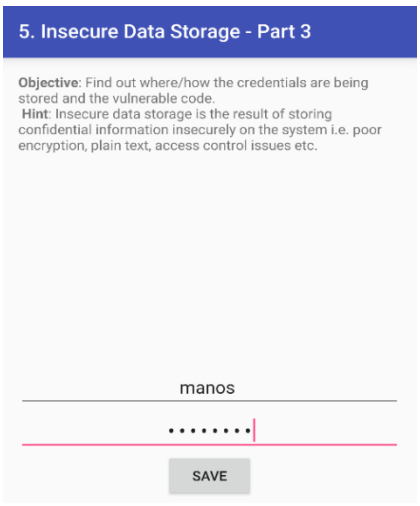
Ακολουθώντας την ίδια λογική, ανοίγουμε σύνδεση με την συσκευή μέσω του adb και κατευθυνόμαστε στο μονοπάτι: data/data/jakhar.aseem.diva/databases. Προκειμένου να ανοίξουμε το αρχείο της βάσης πληκτρολογούμε την εντολή **sqlite3 ids2**. Στην συνέχεια μέσω των εντολών **.tables** -> **.schema myuser** -> **select user, password from myuser**; καταφέραμε να δούμε τα στοιχεία της βάσης τα οποία ήταν τα credentials που είχαμε αποθηκεύσει μέσω του activity όπως φαίνεται και στο παρακάτω στιγμιότυπο.



Εικόνα 43. Insecure Data Storage Part 2 - Αποτέλεσμα μέσω τερματικού

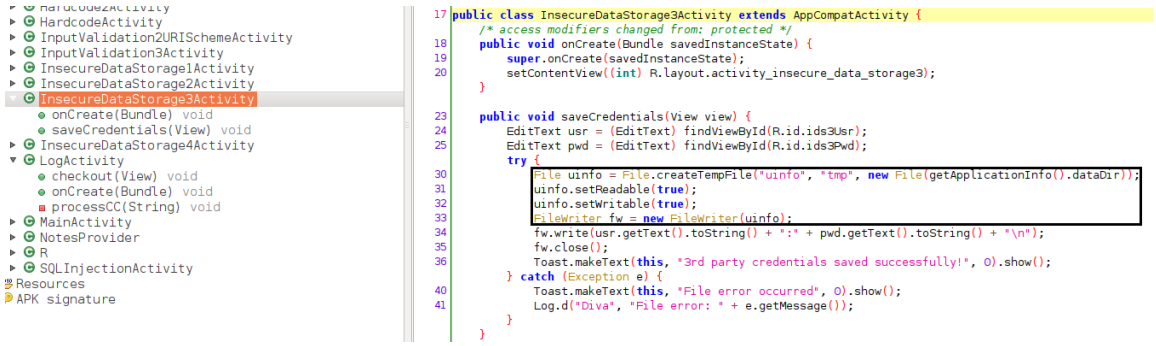
8.5 Insecure Data Storage – Part 3

Ομοίως με τα προηγούμενα βήματα, ακολουθούμε την ίδια διαδικασία. Αρχικά αποθηκεύουμε τα στοιχεία που επιθυμούμε για να εντοπίσουμε στην συνέχεια με ποιον μη ασφαλή τρόπο έχουν αποθηκευτεί στην συσκευή.



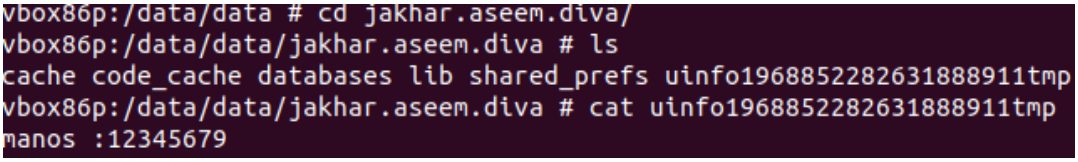
Εικόνα 44. Insecure Data Storage Part 3 - Αρχική Οθόνη

Μέσω του Jadx αναζητούμε την κλάση για την συγκεκριμένη ευπάθεια. Αφού την εντοπίσαμε με ονομασία InsecureDataStorage3Activity, παρατηρήσαμε ότι δίνεται εντολή για δημιουργία αρχείου όπως φαίνεται παρακάτω.



Εικόνα 45. Insecure Data Storage Part 3 - Κώδικας Ευπάθειας

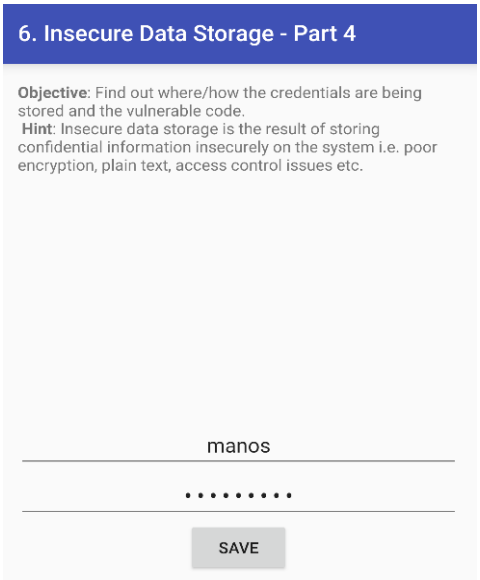
Ομοίως με πριν, ακολουθούμε το path της εφαρμογής μας και εκτελούμε την εντολή **ls** για να ελέγξουμε τα αρχεία του πακέτου. Όπως φαίνεται, έχει δημιουργηθεί ένα νέο αρχείο το οποίο εκτελώντας την εντολή **cat** μας επιστρέφει τα credentials που καταχωρήσαμε.



Εικόνα 46. Insecure Data Storage Part 3 - Εμφάνιση Credentials μέσω τερματικού

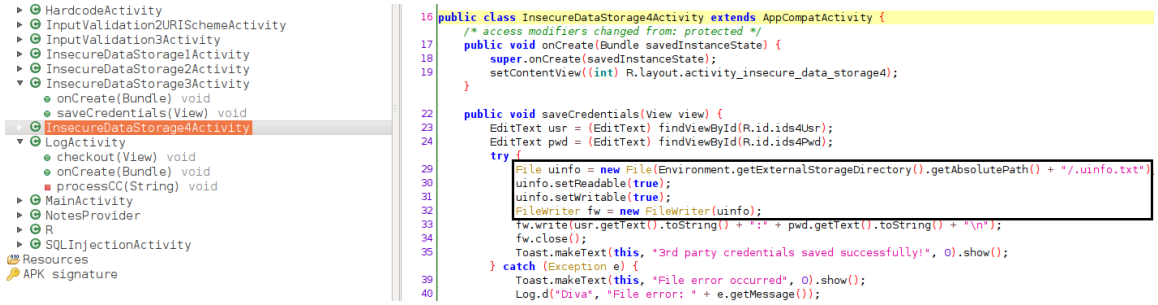
8.6 Insecure Data Storage – Part 4

Στο τέταρτο και τελευταίο μέρος της κατηγορίας «Insecure Data Storage», ομοίως με τα προηγούμενα βήματα, ακολουθούμε την ίδια διαδικασία. Αρχικά αποθηκεύουμε τα στοιχεία που επιθυμούμε για να εντοπίσουμε στην συνέχεια με ποιον μη ασφαλή τρόπο έχουν αποθηκευτεί στην συσκευή.



Εικόνα 47. Insecure Data Storage Part 4 - Αρχική Οθόνη

Μέσω του Jadx αναζητούμε την κλάση για την συγκεκριμένη ευπάθεια. Αφού την εντοπίσαμε με ονομασία InsecureDataStorage4Activity, παρατηρήσαμε ότι δίνεται εντολή για αποθήκευση στην sdcard σε αρχείο με όνομα «.uinfo.txt»

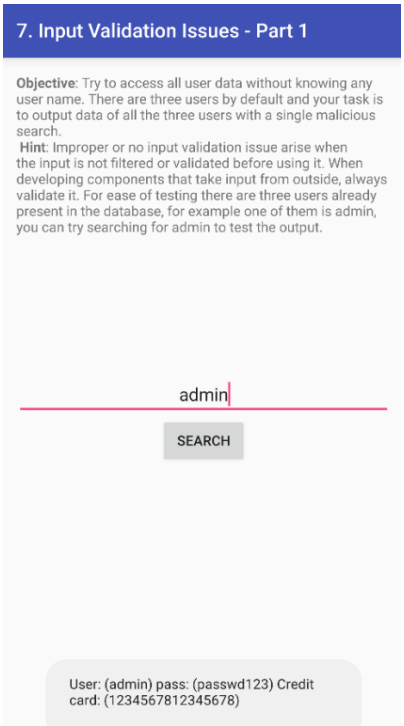


Εικόνα 48. Insecure Data Storage Part 4 - Κώδικας Ευπάθειας

Για να επιβεβαιώσουμε τα παραπάνω, ανοίγουμε σύνδεση μέσω της εντολής **adb shell** και κατευθυνόμαστε στο μονοπάτι της sdcard δίνοντας την εντολή: **cd /mnt/sdcard**. Στην συνέχεια μέσω της εντολής **ls –l** εντοπίσαμε το αρχείο «.uinfo.txt». Μέσω της εντολής cat εμφανίσαμε τα περιεχόμενά του ενώ φαίνεται πως τα credentials αποθηκεύτηκαν στο συγκεκριμένο αρχείο.

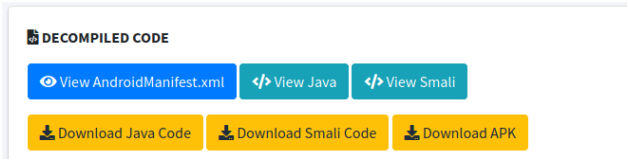
8.7 Input Validation Issues – Part 1

Στην επόμενη ευπάθεια, όπως διαβάζουμε και από την περιγραφή, θα πρέπει να δημιουργήσουμε ένα query το οποίο θα μας επιστρέφει όλους τους χρήστες. Υποθέτουμε λοιπόν ότι για να γίνει αυτό θα πρέπει να κάνουμε SQL Injection. Στο παρακάτω στιγμιότυπο φαίνεται η οθόνη για την συγκεκριμένη ευπάθεια και το αποτέλεσμα που μας φέρνει η αναζήτηση του admin.



Εικόνα 49. Input Validation Issues Part 1 - Αρχική Οθόνη

Μέσω του εργαλείου MobSF θα επεξεργαστούμε το apk με σκοπό να διαβάσουμε τον κώδικα της κλάσης για την συγκεκριμένη ευπάθεια. Για να γίνει αυτό, στην αρχική οθόνη, επιλέγουμε το «View Java» όπως φαίνεται παρακάτω και στην συνέχεια την κλάση «SQLInjectionActivity».



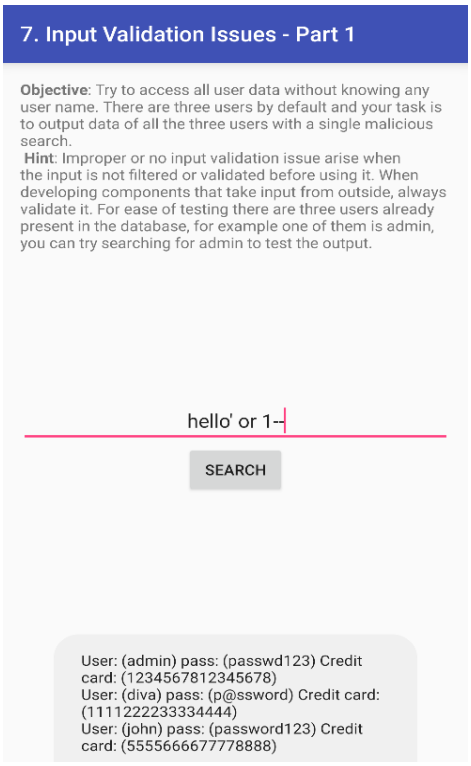
Εικόνα 50. MobSF - Επιλογές Decompiled Κώδικα

Όπως φαίνεται από τον κώδικα στο παρακάτω στιγμιότυπο, τα στοιχεία των χρηστών είναι εμφανή σε plaintext μορφή ενώ παράλληλα το query του «search» δεν προστατεύεται οπότε είναι εύκολη η χρήση SQL Injection.

```
        this.mDB.execSQL("DROP TABLE IF EXISTS sqliuser;");
        this.mDB.execSQL("CREATE TABLE IF NOT EXISTS sqliuser(user VARCHAR, password VARCHAR, credit_card VARCHAR);");
        this.mDB.execSQL("INSERT INTO sqliuser VALUES ('admin', 'passwd123', '1234567812345678');");
        this.mDB.execSQL("INSERT INTO sqliuser VALUES ('diva', 'p@ssword', '1111222233334444');");
        this.mDB.execSQL("INSERT INTO sqliuser VALUES ('john', 'password123', '5555666677778888');");
    } catch (Exception e) {
        Log.d("Diva-sqli", "Error occurred while creating database for SQLI: " + e.getMessage());
    }
    setContentView((int) R.layout.activity_sqlinjection);
}

public void search(View view) {
    EditText srchtxt = (EditText) findViewById(R.id.ivi1search);
    try {
        Cursor cr = this.mDB.rawQuery("SELECT * FROM sqliuser WHERE user = '" + srchtxt.getText().toString() + "'", (String[]) null);
        StringBuilder strb = new StringBuilder("");
```

Εικόνα 51. MobSF - Κώδικας Ευπάθειας

Πράγματι, γράφοντας «hello’ or 1- - » καταφέρνουμε να μας επιστρέψει στο παρακάτω μήνυμα όλους τους χρήστες.

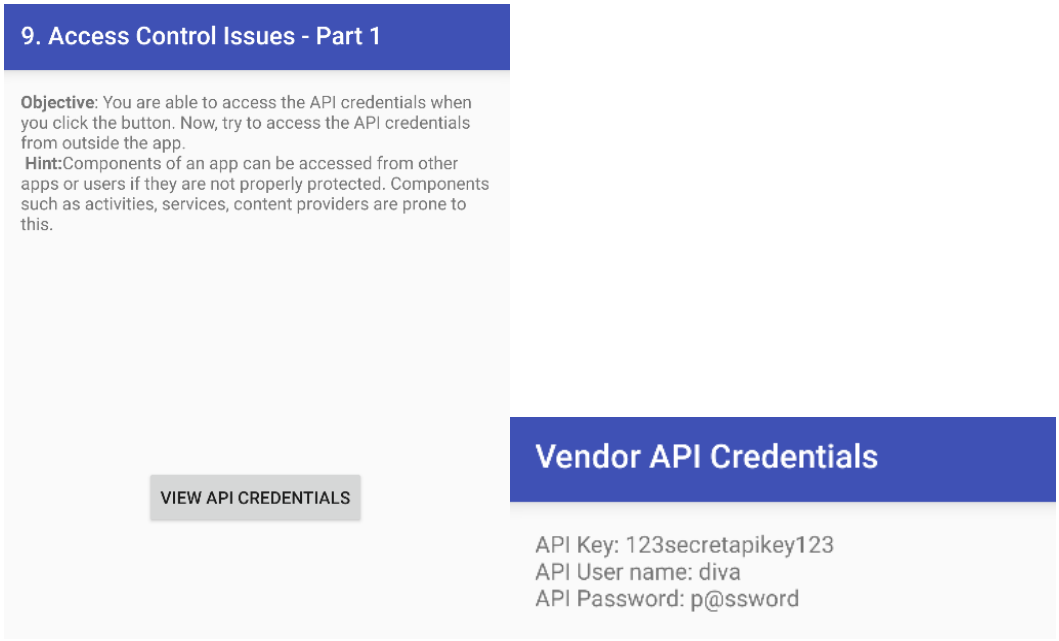


Εικόνα 52. Input Validation Issues - Επιτυχία SQL Injection

8.8 Access Control Issues – Part 1

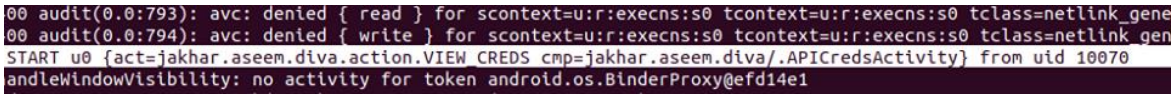
Στην συγκεκριμένη ευπάθεια, όπως ορίζεται και στην περιγραφή της, αν πατήσουμε το κουμπί «VIEW API CREDENTIALS» τότε θα μας εμφανιστούν τα API Credentials σε κάποιο άλλο activity όπως φαίνεται και στα παρακάτω στιγμιότυπα.

Σκοπός του challenge είναι να έχουμε πρόσβαση στα δεδομένα εκτός της εφαρμογής.



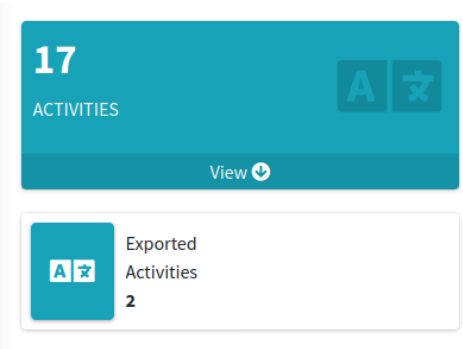
Εικόνα 53. Access Control Issues Part 1 - Αρχική Οθόνη και «VIEW API CREDENTIALS» οθόνη

Προκειμένου να μπορέσουμε να πάρουμε την ονομασία του activity συνδεόμαστε μέσω adb shell στην συσκευή. Στην συνέχεια, επιχειρούμε μέσω του προηγούμενο βήματος να εμφανίσουμε το activity πατώντας κλικ στο «VIEW API CREDENTIALS» ώστε να εμφανιστεί η ονομασία του στα Logs. Πράγματι όπως φαίνεται και παρακάτω καταφέραμε να δούμε την ονομασία του activity στα Logs.



Εικόνα 54. Εμφάνιση της ονομασίας του Activity μέσω τερματικού

Στην συνέχεια εντοπίσαμε με την βοήθεια του MobSF πόσα exported activities υπάρχουν, δηλαδή πόσα activities θα μπορούσαν να κληθούν από εξωτερικούς παράγοντες. Όπως φαίνεται και παρακάτω τα exported activities είναι δύο.



Εικόνα 55. MobSF - Εμφάνιση των exported activities

Στην συνέχεια με την βοήθεια του drozer αναζητήσαμε τα exported activities μέσω της εντολής **run app.activity.info -a jakhar.aseem.diva** και μας εμφανίστηκαν τα παρακάτω activities.

```
dz> run app.activity.info -a jakhar.aseem.diva
Package: jakhar.aseem.diva
  jakhar.aseem.diva.MainActivity
    Permission: null
  jakhar.aseem.diva.APICredsActivity
    Permission: null
  jakhar.aseem.diva.APICreds2Activity
    Permission: null
```

Εικόνα 56. Drozer - Εμφάνιση των exported activities

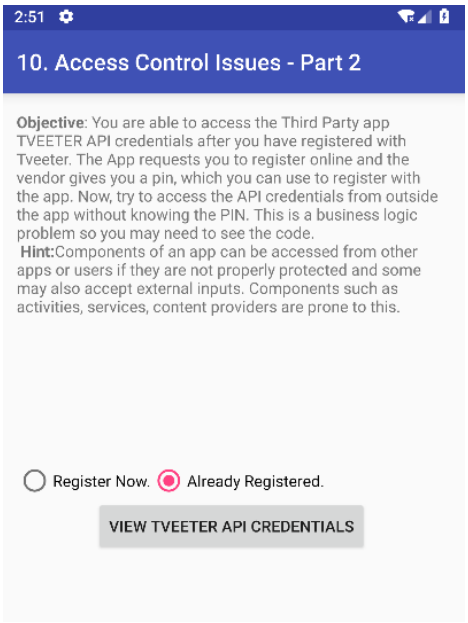
Πράγματι, μέσω του drozer και της παρακάτω εντολής καταφέραμε να ενεργοποιήσουμε απευθείας το συγκεκριμένο activity.

```
dz> run app.activity.start --component jakhar.aseem.diva  jakhar.aseem.diva.APICredsActivity
```

Εικόνα 57. Drozer - Ενεργοποίηση του activity μέσω της παραπάνω εντολής

8.9 Access Control Issues – Part 2

Όπως και στο προηγούμενο vulnerability, έτσι και στην περίπτωση αυτή έχουμε να διαχειριστούμε ένα Access Control Issue όπως φαίνεται παρακάτω.



Εικόνα 58. Access Control Issues Part 2 - Αρχική Οθόνη

Ομοίως με πριν, για να εντοπίσουμε την ονομασία του activity, θα πρέπει να συνδεθούμε με την συσκευή μέσω του adb shell και στην συνέχεια να ελέγξουμε τα Logs. Όπως φαίνεται παρακάτω, καταφέραμε να πάρουμε την ονομασία του Activity.

```
va.action.VIEW_CREDS2 cmp=jakhar.aseem.diva/.APICreds2Activity (has extras)} from uid 10070
```

Εικόνα 59. Drozer - Εμφάνιση της ονομασίας του activity

Για να έχουμε μία πιο ξεκάθαρη εικόνα αλλά και για να συλλέξουμε στοιχεία, ελέγχουμε μέσω του MobSF το Manifest File. Όπως φαίνεται παρακάτω, το activity του συγκεκριμένου vulnerability περιλαμβάνει ένα Intent-filter γεγονός που το αφήνει ως exported και οποιοσδήποτε τρίτος θα μπορούσε να το καλέσει.

```
<activity android:label="@string/apic2_label" android:name="jakhar.aseem.diva.APICreds2Activity">
    <intent-filter>
        <action android:name="jakhar.aseem.diva.action.VIEW_CREDS2" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

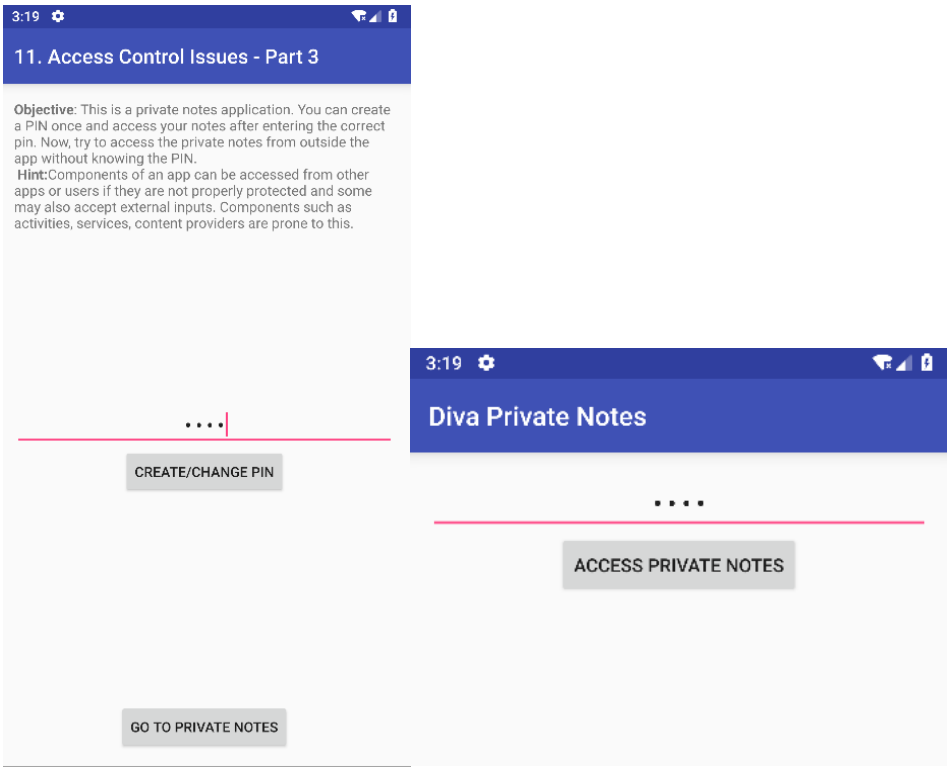
Εικόνα 60. MobSF - Manifest File - Intent Filter

Έτσι μέσω της παρακάτω εντολής είναι εφικτό να καλέσουμε το activity:

adb shell am start jakhar.aseem.diva/.APICredsActivity

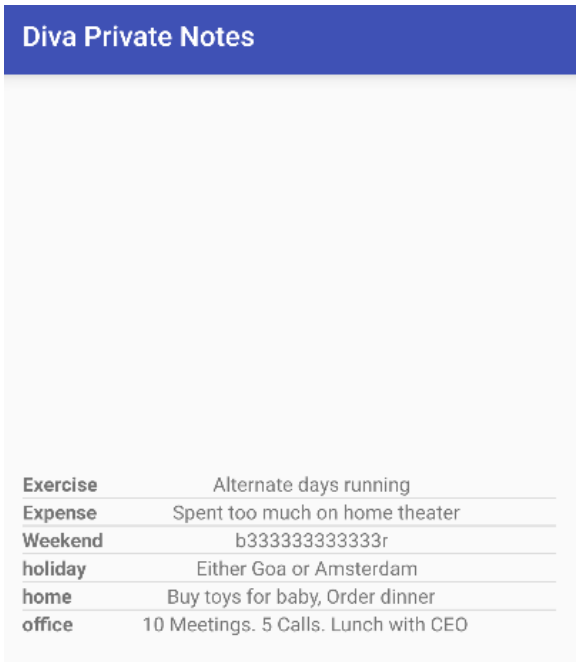
8.10 Access Control Issues – Part 3

Στην συγκεκριμένη ευπάθεια, όπως ορίζεται και στην περιγραφή της ζητείται να έχουμε πρόσβαση σε Private Notes εκτός της εφαρμογής. Αρχικά, δώσαμε τέσσερις αριθμούς και στην συνέχεια εμφανίστηκε το κουμπί «GO TO PRIVATE NOTES». Αφού το πατήσαμε εμφανίστηκε το επόμενο activity στο οποίο μας ζητείται ο κωδικός που ορίσαμε νωρίτερα και πατήσαμε το «ACCESS PRIVATE NOTES».



Εικόνα 61. Access Control Issues Part 3 - Αρχική Οθόνη και «GO TO PRIVATE NOTES» Οθόνη

Αφού έγιναν όλα τα παραπάνω, εμφανίστηκε το παρακάτω παράθυρο με τα δεδομένα που θέλουμε να έχουμε πρόσβαση εκτός εφαρμογής.



Εικόνα 62. Access Control Issues Part 3 - Diva Private Notes Οθόνη

Για να έχουμε πρόσβαση στα δεδομένα αυτά, χρησιμοποιούμε το MobSF το οποίο μας δείχνει ότι ένας content provider είναι exported, δηλαδή μπορεί να τον εκμεταλλευτεί οποιοσδήποτε τρίτος. Για να δούμε ποιος content provider είναι, χρησιμοποιούμε το drozer με την παρακάτω εντολή:

```
dz> run app.provider.info -a jakhar.aseem.diva
Package: jakhar.aseem.diva
  Authority: jakhar.aseem.diva.provider.notesprovider
    Read Permission: null
    Write Permission: null
    Content Provider: jakhar.aseem.diva.NotesProvider
    Multiprocess Allowed: False
    Grant Uri Permissions: False
```

Εικόνα 63. Drozer - Έλεγχος Exported Activities

Όπως βλέπουμε και στο παραπάνω στιγμιότυπο ο content provider ανήκει στον NotesProvider οπότε με την παρακάτω εντολή θα πάρουμε το URI του πακέτου.

```
dz> run app.provider.finduri  jakhar.aseem.diva
Scanning jakhar.aseem.diva...
content://jakhar.aseem.diva.provider.notesprovider/notes/
content://jakhar.aseem.diva.provider.notesprovider
content://jakhar.aseem.diva.provider.notesprovider/
content://jakhar.aseem.diva.provider.notesprovider/notes
```

Εικόνα 64. Drozer - Scanning για εύρεση URIs

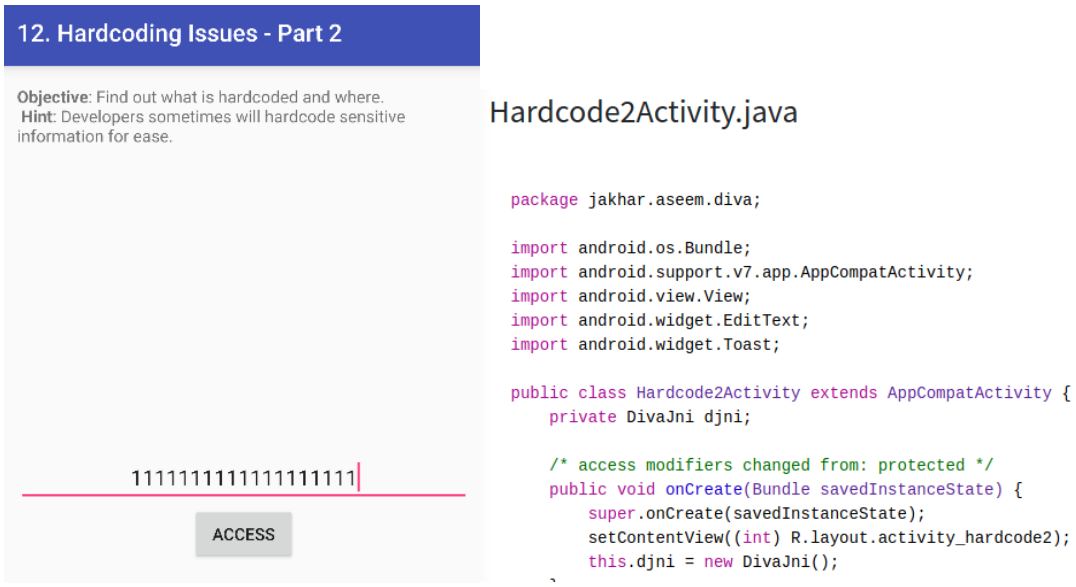
Τέλος, τρέχοντας την παρακάτω εντολή για το URI που ανακαλύψαμε, βλέπουμε πως πράγματι εμφανίζονται τα στοιχεία του Notes Provider.

```
dz> run app.provider.query  content://jakhar.aseem.diva.provider.notesprovider/notes/
| _id | title    | note                                |
| 5   | Exercise | Alternate days running              |
| 4   | Expense  | Spent too much on home theater      |
| 6   | Weekend  | b333333333333r                      |
| 3   | holiday  | Either Goa or Amsterdam             |
| 2   | home     | Buy toys for baby, Order dinner     |
| 1   | office   | 10 Meetings. 5 Calls. Lunch with CEO |
```

Εικόνα 65. Drozer - Query για τα περιεχόμενη του URI

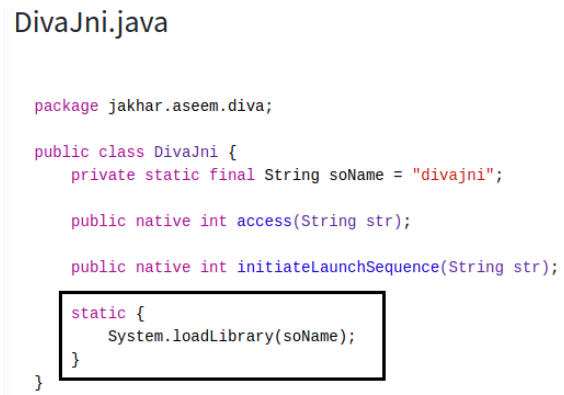
8.11 Hardcoding Issues – Part 2

Στην συγκεκριμένη ευπάθεια θα πρέπει να βρούμε hardcoded στοιχεία που θα μας επιτρέψουν την πρόσβαση στην παρακάτω οθόνη, όπως ορίζεται και από την περιγραφή. Αρχικά, αναζητήσαμε στον κώδικα την κλάση που σχετίζεται με το συγκεκριμένο ζήτημα και μας οδήγησε σε μία επόμενη κλάση με όνομα DivaJni που θα μπορούσε να μας βοηθήσει περισσότερο.



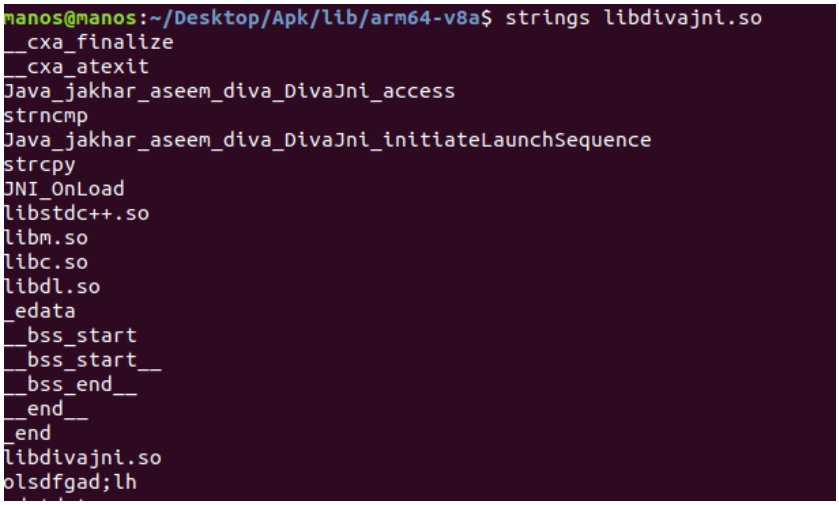
Εικόνα 66. Hardcoding Issues Part 2 - Αρχική Οθόνη (αριστερά) και ο κώδικας του activity (δεξιά)

Από τον παρακάτω κώδικα συνειδητοποιούμε πως θα πρέπει να ψάξουμε στα libraries της εφαρμογής.



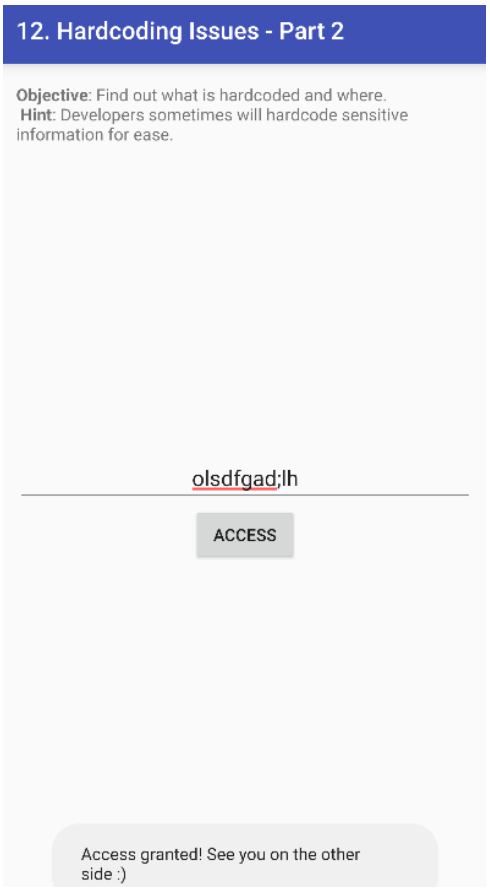
Εικόνα 67. MobSF - Κώδικας DivaJni.java

Για να αναζητήσουμε στα libraries, αρχικά κάναμε unzip το apk και στην συνέχεια αναζητήσαμε τον φάκελο lib στον οποίο αποθηκεύονται οι βιβλιοθήκες. Ύστερα από αρκετή αναζήτηση βρέθηκε το παρακάτω αρχείο «libdivajni.so», το οποίο διαθέτει έναν κωδικό «olsdfgad;lh»



Εικόνα 68. Εμφάνιση Strings που διαθέτει το αρχείο libdivajni.so

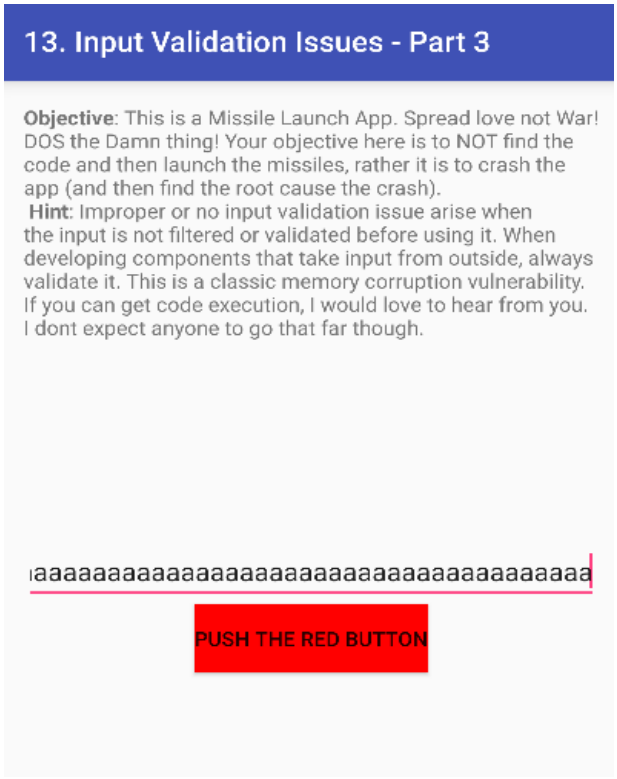
Εν τέλει, όταν δοκιμάσαμε τον παραπάνω κωδικό, το activity μας επέτρεψε την είσοδο.



Εικόνα 69. Hardcoding Issues Part 2 - Επιτυχής πρόσβαση

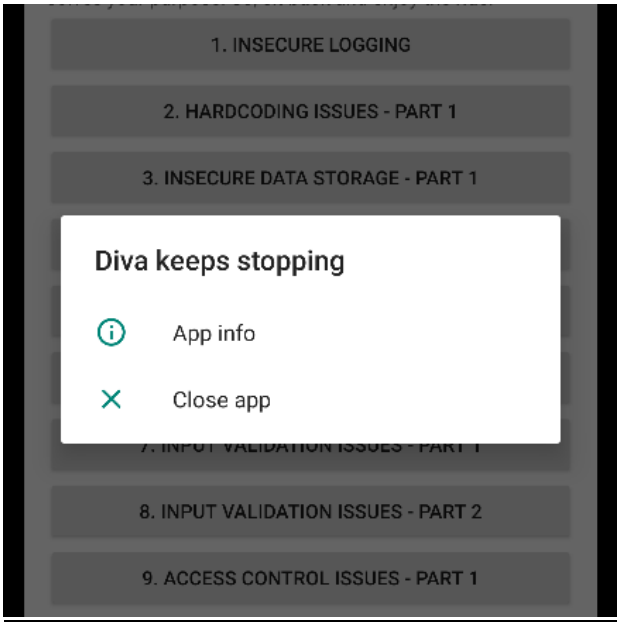
8.12 Input Validation Issues – Part 3

Για την τελευταία ευπάθεια θα πρέπει να ελέγξουμε αν γίνεται έλεγχος του input και αν μπορούμε με κάποιον τρόπο να την παρακάμψουμε.



Εικόνα 70. Input Validation Issues Part 3 - Αρχική Οθόνη

Όπως παρατηρήσαμε δίνοντας μεγάλη είσοδο το πρόγραμμα έκανε shut down. Αυτό προκύπτει διότι δεν είχε προβλέψει ο προγραμματιστής τα μεγάλα input που θα μπορούσε να δεχθεί το πρόγραμμα.



Εικόνα 71. Αποτυχία λειτουργίας της εφαρμογής λόγω buffer overflow

9. Συμπεράσματα

Όπως προαναφέρθηκε, ο αριθμός των ενεργών χρηστών και ο ρυθμός παραγωγής εφαρμογών στο λειτουργικό Android αυξάνεται. Είναι λογικό, λοιπόν, να αυξάνεται ο αριθμός των επιθέσεων καθώς σε αρκετές περιπτώσεις οι δημιουργοί αγνοούν ή δεν γνωρίζουν τις σωστές πρακτικές για την δημιουργία εφαρμογών με υψηλό επίπεδο ασφάλειας.

Όσο περισσότερες είναι οι εφαρμογές με ευπάθειες, τόσο πιο εύκολα θα προσβάλλεται ο χρήστης, η ιδιωτικότητά του αλλά και το ίδιο το λειτουργικό. Επομένως, εύκολα συμπεραίνουμε ότι η διαδικασία του penetration testing, η καλή γνώση σε θέματα ασφάλειας των εφαρμογών και εν γένει της ιδιωτικότητας είναι κλειδί στην προστασία των χρηστών και των εφαρμογών.

10. Βιβλιογραφία

[1] Tam H. Doan (2019, August 4) Virtual Machine in Android: Everything you need to know - https://android.jlelse.eu/virtual-machine-in-android-everything-you-need-to-know-9ec695f7313b
[2] Govindraj Basatwar (2020, January 23) OWASP Mobile Top 10: A comprehensive guide for mobile developers to counter risks - https://www.appsealing.com/owasp-mobile-top-10-a-comprehensive-guide-for-mobile-developers-to-counter-risks/
[3] Jonas Torstensson (2017) Android SecurityAn evaluation of applicationsin Google Play - https://pdfs.semanticscholar.org/2912/e7a8ccd364b1f49ef20706e583f6d05b5a91.pdf
[4] Fredrik Andersson & GustafAndersson (2012, June 26) Android Environment Security - https://www.diva-portal.org/smash/get/diva2:537572/FULLTEXT01.pdf
[5] Vicente Javier Mozos Pérez (2013, September) A study of vulnerabilities on Android systems - https://riunet.upv.es/bitstream/handle/10251/39560/Tesina_Vicente_Javier_Mozos_Perez.pdf
[6] Michael Heinl (2015, June 23) Android Security: Creation Of A Virtual Learning Environment - https://d-nb.info/1154866548/34
[7] Google (2019) Platform Architecture - https://developer.android.com/guide/platform
[8] Margaret Rouse (2018, October) Penetration Testing - https://searchsecurity.techtarget.com/definition/penetration-testing
[9] Svitla Team (2019, June 29) Black-box and White-box Testing - https://svitla.com/blog/black-box-and-white-box-testing
[10] TutorialsPoint (2020) Types of Penetration Testing - https://www.tutorialspoint.com/penetration_testing/types_of_penetration_testing.htm
[11] Hacken (2019, August 15) Mobile Application Penetration Testing Methodology - https://hacken.io/research/education/mobile-application-penetration-testing-methodology/
[12] Giuseppe Porcu (2018, October 19) How to test your mobile applications against security vulnerabilities - https://www.owasp.org/images/9/99/GiuseppePorcu-OWASPItalyDay19-10-2018.pdf
[13] Pooja Gaonkar (2019, June 17) Android Application Penetration Testing – Part 1 - https://gbhackers.com/android-application-penetration-testing-1/
[14] Sawan Bhan (2019, Febrouary) An Open Source Android Applications Penetration Testing Lab - https://www.ijntr.org/download_data/IJNTR05020027.pdf
[15] Lester Obbayi (2019, January 19) Introduction to the Mobile Application Penetration Testing Methodology - https://resources.infosecinstitute.com/introduction-mobile-application-penetration-testing-methodology/#gref
[16] Github (2020) A collection of android security related resources - https://github.com/ashishb/android-security-awesome
[17] HackTricks (2020) Drozer Tutorial- https://book.hacktricks.xyz/mobile-apps-pentesting/android-app-pentesting/drozer-tutorial
[18] Github (2020) Android-Vulnerabilities-Solution - https://github.com/tjunxiang92/Android-Vulnerabilities/blob/master/Android-Vulnerabilities-Solution.pdf
[19] Srivinas (2018) Cracking Damn Insecure and Vulnerable App (DIVA) - https://resources.infosecinstitute.com/cracking-damn-insecure-and-vulnerable-app-diva-part-3/#gref
[20] Aldeid (2018) Jadx - https://www.aldeid.com/wiki/Jadx
[21] Aditya Agrawal (2019 October 13) Android Application Security - https://manifestsecurity.com/page/2/

[22] Emil Hozan (2019, October 4) Android APK Reverse Engineering: Using JADX - https://www.secplicity.org/2019/10/04/android-apk-reverse-engineering-using-jadx/
[23] Pentester Land (2018, October 12) List of intentionally vulnerable Android apps - https://pentester.land/cheatsheets/2018/10/12/list-of-Intentionally-vulnerable-android-apps.html
[24] Hamza M'hirsi (2018, December 23) How To Pentest Android Application - https://www.peerlyst.com/posts/how-to-pentest-android-application-hamza-m-hirsi
[25] kshitija shirke (2019, January 25) Mobile Security Framework (MobSF) Static Analysis - https://medium.com/@kshitishirke/mobile-security-framework-mobsf-static-analysis-df22fcdae46e
[26] Bernhard Mueller, Sven Schleier, Jeroen Willemsen, The OWASP mobile team (2018) Mobile Security Testing Guide - https://biblio.securityhacklabs.net/Hacking/Moviles/mobile-security-testing-guide.pdf
[27] Thao N. Vo (2019, January 17) Pentesting Android applications by reversing and finding attack surfaces - https://blog.usejournal.com/an-intro-to-pentesting-an-android-phone-464ec4860f39
[28] Ahlam Mohammad Almusallam (2018) Penetration Testing for Android Applications with Santoku Linux - http://broncoscholar.library.cpp.edu/bitstream/handle/10211.3/204208/AlmusallamAhlam_Project2018.pdf?sequence=3
[29] Unknown (2014, September 21) Drozer Commands - A Security & Attack Framework for Android - http://th3-incognito-guy.blogspot.com/2014/09/drozer-security-attack-framework-for.html
[30] Tanprathan (2019) MobileApp-Pentest-Cheatsheet - https://github.com/tanprathan/MobileApp-Pentest-Cheatsheet