COMPUTER & SECURITY 42 (2014) 66-76

Adli Bilişim Açısından Android Mobil Uygulamalarının Gizliliğini Değerlendirmek

(*Orj: Evaluating The Privacy Of Android Mobile Applications Under Forensic Analysis*)

Christoforos Ntantogian,
Dimitris Apostolopoulos, Giannis Marinakis, Christos Xenakis

Department of Digital Systems, University of Piraeus, Piraeus, Greece

Çeviren: *Özgür KOCA, (ozgur.koca@linux.org.tr / www.tankado.com),*

Özet

Bu çalışmada Android işletim sisteminin RAM adı verilen geçici/uçucu belleğinde yer alan çeşitli Android uygulamalarına ait kimlik doğrulama verilerinin elde edilmesine yönelik teknikler hakkında incelemeler yapılmıştır. Çalışma sırasında tamamen açık kaynak kodlu inceleme araçlarından yararlanılmıştır. Birinci bölümde hassas kullanıcı verisi içerebilecek çeşitli kategorilerde 13 farklı Android uygulamasının kimlik doğrulama bilgileri, 30 farklı senaryo altında, mobil aygıtın fiziksel belleğinden ayıklanarak hassas verileri elde edilmiştir. İkinci bölümde elde edilen bellek dökümleri içerisinde, ilgili uygulamaların hassas verilerinin yer aldığı bellek konumları tespit edilmiştir. Üçüncü bölümde ise toplanan veriler ışığında Android mobil uygulamaların ve cihazların mahremiyeti ile ilgili bir dizi kritik gözlem sonucu ortaya koyulmuştur.

1. Sunuş

Diğer tüm tüketici elektroniği pazarlarına göre mobil aygıt pazarı çok daha hızlı bir şekilde büyümektedir [1].

Bu mobil cihazlar, hareket halindeyken son kullanıcıların geniş bir veri ve servis setine erişmesini ve yönetmesini sağlayan yeni bir işleme ve iletişim paradigmasını beraberinde getiriyor. Bunu gerçekleştirmek için, eğlence ve oyunlardan kritik mobil bankacılığa ve kurumsal kaynaklara erişim için özel kurumsal uygulamalara kadar geniş bir yelpazede mobil uygulamalar geliştirilmiştir. Bu yönüyle mobil cihazlar, büyük fırsatların yanı sıra ilgili taraflar için yeni saldırı vektörleri ortaya çıkartmıştır (Mylonas et al., 2013). Küçük boyutlarından dolayı mobil cihazların kolayca çalınabileceği veya bir yerde unutulabileceği bir gerçektir. E-postalar, sosyal etkinlikler, resimler veya başka herhangi bir depolanmış veri açığa çıkabileceğinden, bir mobil cihazın kaybedilmesi büyük bir gizlilik ihlaline neden olabilir.

2011'de, kayıp telefon sorunu (Ponemon Institute LLC, 2011) olarak adlandırılan bir araştırmada, 12 aylık dönemde, 3.297.569 çalışanın akıllı telefonunun 142.708'inin kaybolduğunu veya çalındığını, yani yıllık yüzde 4.3'lük bir performans gösterdiğini tespit etti. Ayrıca, 2012'de Symantec firmasından araştırmacılar, Smartphone Honey Stick Project'in sonuçlarını sundular (Wright, 2012). Bu projede 50 adet akıllı telefon, kasıtlı olarak, ABD ve Kanada'daki şehirlerde kayboldu.

Symantec'in kaybolan mobil cihazları ve bulma ve veri güvenliği konusunda geliştirdiği uygulama ile cihazlar uzaktan takip edilmiş ve yaşanan vakarda kullanıcıların %96'sının cihaz ve verilerine erişebildiğini göstermiştir. Symantec'in uygulaması ile cihaz internete bağlandığı

anda sahibinin online hesabı üzerinden cihazın konumu öğrenme ve kritik verileri silme gibi özellikleri çalıştırabildiği bir imkan sunar. Bu yollar sahibine ulaşan telefonların neredeyse yarısında (%43) cihazı eline geçiren kişinin telefondaki mobil bankacılık uygulamalarını kullanmaya çalıştığı tespit edilmiştir.

Mobil cihazların yaygınlaşması ile Mobil Adli Bilişim alt bilim dalı da oluşmuştur. Bu dal verilerin değişmesine izin vermeden tam bir kopyanın elde edilmesini ve delillerin araştırılmasını amaçlar. Bu alandaki araştırmaların çoğu alt başlıklarda yoğunlaşmıştır.

1) Dahili flaş NAND hafızası ve SD Kartlarının elde edilmesi ve analizi
2) Kayıt ortamının kullandığı dosya sistemlerinin anlaşılması
3) Kötü amaçlı yazılımları tanımlamak için uygulama dosyalarının incelenmesi

Bununla birlikte, taşınabilir aygıtların rasgele erişimli bellek (RAM) olarak da adlandırılan uçucu belleğinin içeriğinin elde edilmesi ve analizi üzerine yapılan araştırmalara çok az dikkat edilmiştir. Bu çalışmanın ana motivasyonu açıkça mobil cihazların uçucu belleğine odaklanmaktır. Dahası, bu tür bir bellek (volatile memory), güvenlik kritik uygulamaları (ör. Mobil bankacılık, şifre yöneticileri, vb.) etkinleştirmek için kullanıcılar tarafından gönderilen kimlik doğrulama bilgilerini (diğer bir deyişle kullanıcı adı ve parolaları) geçici olarak tutar.

Adli araştırmacılar, masa üstü bilgisayarların uçucu belleğinin dökümlerinde (*memory dump*) yaptıkları analizlerde kullanıcıların kimlik bilgilerine ulaşabildiklerini kanıtlamıştır. Kullanıcıların %61'inin (Tüketici anketi 2012) aynı kimlik doğrulama bilgilerini birden fazla internet sitesinde veya online serviste kullanmayı tercih ettiği bir ortamda, online hizmetler ile büyük oranda entegre çalışan mobil cihazların incelenmesi motive edicidir. Bazen tek bir kullanıcı adı ve parolasının elde edilmesi tüm kullanıcı uygulamalarının mahremiyetinden ödün vermek için yeterli olabilir (Mylonas et Al., 2013). Özellikle hassas veri veya işlevselliklerini (örneğin, bankacılık, şifre yöneticileri, e-alışveriş vb.) kullanan uygulamalar olması durumunda kimlik doğrulama bilgilerinin görülmesi, büyük bir gizlilik ihlaline neden olabilir.

Bu yazıda, otuz (30) farklı senaryoyu takip ederek mobil uygulamaların kimlik doğrulama bilgilerini cihazın uçucu belleğinde bulup bulamayacağımızı deneysel bir analizle araştırıp değerlendiriyoruz. En yaygın olanı olduğu için Android işletim sistemi (OS) ile çalışan mobil cihazlara odaklanıyoruz (IDC Worldwide Quarterly, 2013). Deneyleri gerçekleştirmek için, root'lu mobil cihazların uçucu belleğini, adli olarak sağlıklı koşullar altında edinmek için bir prosedür izliyoruz. Uygulanan deneyler ve analiz boyunca, sadece açık kaynaklı, ücretsiz adli araçlar kullandık.

Hassas kullanıcı verilerini ayrıntılı olarak anlatan dört yaygın uygulama kategorisini (yani mobil bankacılık, e alışveriş / finansal uygulamalar, şifre yöneticileri ve şifreleme / veri gizleme uygulamaları) temsil eden toplamda on üç (13) popüler Android uygulamasının gizliliğini değerlendirdik . İncelenen her uygulama ve incelenen her senaryo için, her biri farklı amaçlarla iki deney seti oluşturduk. Birincisi, kendi gönderilen kimlik bilgilerimizi bir taşınabilir aygıtın bellek dökümünden kurtarabilir miyiz kontrol etmek oldu. İkinci deneyde amaç, kimlik bilgilerinin bir bellek imajında nerede olduğunu belirten kalıpları bulmaktı. Genel olarak, bu çalışmanın katkısı şöyledir:

1) İncelenen her uygulama için mobil aygıtların fiziksel belleğinde kimlik doğrulama kimlik bilgilerini keşfedebilir miyiz?
2) Belli uygulamaların bellek dökümünde kimlik doğrulama kimlik bilgilerinin konumunu belirten kalıpları ve ifadeleri keşfedebilir miyiz?
3) Çeşitli mobil kullanım senaryoları altında mobil uygulamaların gizliliği konusunda bilgi sağlayan bir dizi kritik gözlem türetme.

Makalenin geri kalan kısmı aşağıdaki şekilde organize edilmiştir. Bölüm 2, Android OS ve ilgili iş için arka plan bilgisi verir. Bölüm 3, Android mobil cihazların uçucu belleğini edinme prosedürünü sunmaktadır. Bölüm 4 gerçekleştirilen deneyleri analiz eder. Bölüm 5, Android cihazlarda kimlik doğrulama bilgilerinin mahremiyeti ile ilgili genel gözlemler ve açıklamalar sağlayan sonuçları

ayrıntılı bir şekilde açıklar. Son olarak, bölüm 6, makalenin sonuçlarını aktarır.

2. Çalışmanın Arka Planı

2.1 Android İşletim Sistemi

Android, öncelikle akıllı telefonlar ve tablet bilgisayarlar gibi dokunmatik ekranlı mobil cihazlar için tasarlanmış Linux tabanlı bir işletim sistemidir. İlk ortaya çıkışından bu yana, Android geçen yıla göre üç basamaklı büyümeye ulaşan yükseliş eğilimi ile geniş bir kabul gördü (IDC Worldwide Quarterly, 2013). Bugün, dünya pazarının yaklaşık yüzde 75'ini elinde bulunduruyor ve şu ana kadar en hızlı büyüyen mobil işletim sistemi olarak nitelendiren 48 milyardan fazla Android uygulaması bulunuyor.

Android, işletim sistemi görevlerini gerçekleştirmek için yerel açık kaynaklı C kütüphanelerini ve uygulamaları geliştirirken de Java'yı bir dil olarak kullanmaktadır. Bunları yürütmek için Dalvik sanal makinesini kullanır (Bornstein, 2008). Dalvik yürütülebilir dosyaları .dex (yani, .class ve .jar dosyalarından gelen bayt kodları) oluşturan, belleği ve işlemci hızı gibi sistem kaynakları sınırlı sistemlere uygun olacak şekilde tasarlanmıştır. Her bir Android uygulaması, kendi Dalvik örneği içinde ayrı bir süreçte çalışır ve bellek ile süreç yönetimi görevleri birbirinden yalıtılmış şekilde yürütülür [2].

Android cihazları, her biri farklı amaçlara hizmet eden üç farklı türde hafıza kullanmaktadır:

1) Güç kesildiğinde verilerini kaybeden uçucu hafıza (RAM)
2) Verileri korumak için güç gerektirmeyen, NAND flash teknolojisine dayanan dahili, uçucu olmayan bellek
3) SD kart biçiminde harici, genişletilebilir, kalıcı olmayan bellek. Hem flaş hem de SD bellek kartı, uygulamaların ve multimedya dosyalarının yanı sıra YAFFS2 adlı Android dosya sistemini de depolar.

Bazı güvenlik hassas mobil uygulamalarda (örneğin, mobil bankacılık, finansal uygulamalar, şifre yöneticileri, vb.), kullanıcıların kimlik bilgileri (örn. Kullanıcı adı ve şifre), kalıcı olarak depolanmaz (örn., Flash bellek veya SD kart), bu şekilde kullanıcı mahremiyetinin tehlikeye atılma olasılığı ortadan kaldırılmaya çalışılır. Böyle bir uygulama her etkinleştirildiğinde, kullanıcı sağlanan hizmetlere erişmek için yeniden kimlik bilgilerini yazıp yeniden göndermek zorundadır. Yalnızca mobil aygıtların uçucu belleğinde bulunan kimlik bilgileri, hareket halindeki veri olarak tanımlanır (Hoog, 2011). Bu çalışmada, hareket halindeki veriler olarak var olan kimlik doğrulama kimlik bilgileri, bu mobil uygulamaların desteklediği güvenlik ve gizlilik potansiyelini inceliyoruz.

2.2 Yaptığımız Çalışma

Mobil cihazlar lokasyon, gelen ve giden çağrı ve mesajlar, e-postalar, tarama bilgileri, uygulama kullanımı, multimedya dosyaları vb. ilginç bilgilerin büyük bir kısmını depoladıkları için her adli araştırmacı için önemli bir delil kaynağı oluşturmaktadır. Bu gerçeğe göre hareket ederek , Android cihazlardaki mevcut adli araştırmaların büyük bir çoğunluğu dahili NAND flash belleği ve SD kartlarının elde edilmesi ve analizi üzerine yoğunlaşmış ve YAFFS2 dosya sistemi ve bunun içinde depolanan verilerle ilgili önemli sonuçlar elde edilebileceğini göstermiştir. Adli teknikleri kullanarak, araştırmacılar, Android'de uygulama dağıtmak ve yüklemek için kullanılan kötü amaçlı yazılımlar için Android uygulama paketi (.apk) dosyalarının incelenmesini sağlamışlardır. Ancak, bildiğimiz kadarıyla, Android cihazların uçucu belleğinin edinilmesi ve analizine çok az dikkat çekilmiştir.

Önceki çalışmamızda (Apostolopoulos ve diğerleri, 2013), çalışmakta olan bir işlemin (process) bellek içeriğini boşaltmak için Android yazılım geliştirme kitiyle birlikte gelen Dalvik Hata Ayıklama Monitörü Sunucusu (DDMS) aracını [3] kullandık. DDMS kullanarak yeterli sayıda mobil uygulama örneğinin anlık bellek görüntüsünü inceledik ve çoğunluğunda

kimlik doğrulama bilgilerini keşfettik. Bu çalışmanın bir kısıtlaması, bellek dökümlerinin gerçek bir Android cihaz yerine bir Android emülatöründen edinildiği gerçeğiyle ilgilidir. Dahası, DDMS, adli analiz için sınırlı yeteneklere sahiptir, çünkü cihazın tüm belleğini boşaltamaz.

Müller and Spreitzenbarth (2013) adlı çalışmada, yazarlar, ARM işlemcileri ile donatılmış Android mobil cihazlara yönelik soğuk önyükleme saldırılarının mümkün olduğunu kanıtladı. Bunu başarmak için, önyükleyicinin kilidi kaldırılmış ve kullanıcı verilerinin disk bölümü şifrelenmiş bir Galaxy Nexus mobil cihazı kullanıyorlardı. Diskin şifrelenmiş bölümünde depoalan anahtarı aygıtın uçucu belleğinden alabildiler ve daha sonra dosya sisteminin şifresini çözdüler. Yazarlar ayrıca, önyükleyicinin kilitli olduğu aygıtlarda, disk şifrelemesini kırmaya yönelik herhangi bir girişimin içindeki verilerin silinmesine neden olacağını belirtti. Ancak, kişi listeleri, e-postalar, fotoğraflar, vb. gibi kişisel bilgileri aygıtın uçucu belleğinden almayı başardılar. Son olarak, yazarlar, kullanılan şifreleme anahtarlarını alma sürecini otomatikleştirmek için FROST adlı bir kurtarma görüntüsü geliştirdi.

Thing ve ark. (2010), bir sürecin başka bir işlemin yürütülmesini denetleyip kontrol etmesini sağlayan **ptrace** sistem çağrısını kullanarak, çalışan bir sürecin belirli bellek bölgelerini boşaltmayı başarmışlardır. Gerçekleştirilen deneyler sohbet uygulamalarının kanıtlarının keşfedilmesine (yani, gelen ve giden iletiler) odaklanıyordu. Bu yaklaşımın bir sınırlaması, üzerine yazmalara neden olabilecek ve kanıtların kaybolmasına neden olabilecek birden fazla mobil cihazla etkileşime girmesine neden olan özel bir bellek çıkarımını (memory dump) gerektirmesidir. Buna değinmeye çalışan, (Case, 2011) yazarlar, Android'in Dalvik sanal makinasının ilk kamuoyu analizini sundular. Bu eser çağrıların geçmişi, sesli postalar, tarama geçmişi ve kablosuz yerel ağ anahtarı gibi adli olarak ilginç bilgilerin toplanmasına yol açan bir erişim yöntemi öneriyor.

Volatilitux (Girault, 2010), Linux tabanlı bir sistemin uçucu belleğini analiz etmek için bir çerçeve (python programlama dili ile yazılmış) sağlar. Dijital kanıtların bir bellek dökümünden çıkarılmasını sağlar, ancak çalışan işlemlerin numaralandırılması, çalışan işlemlerin bellek haritaları gibi sınırlı sayıda analiz kabiliyetini destekler. Dahası, şu ana kadar Android cihazların uçucu hafızasından delil elde etmek için bir araç yoktur. [4] 'deki çalışma, bir uygulamanın çalışan bir işlemini sonlandıran **kill** komutu kullanılarak bir Android uygulamasının belleğini boşaltma tekniğini açıklamaktadır. Yakalanan bellek anlık görüntüsünü analiz eden yazarlar, incelenen uygulama tarafından kullanılan bir şifreleme anahtarı bulmayı başarmışlardır.

İlgili çalışmanın temel sınırlamaları şu iki olgu ile ilgilidir:

1) Yalnızca Android'in uçucu belleğinin bir kısmı;
2) Elde edilen anlık bellek görüntüleri spesifik uygulamalarla ilgilidir.

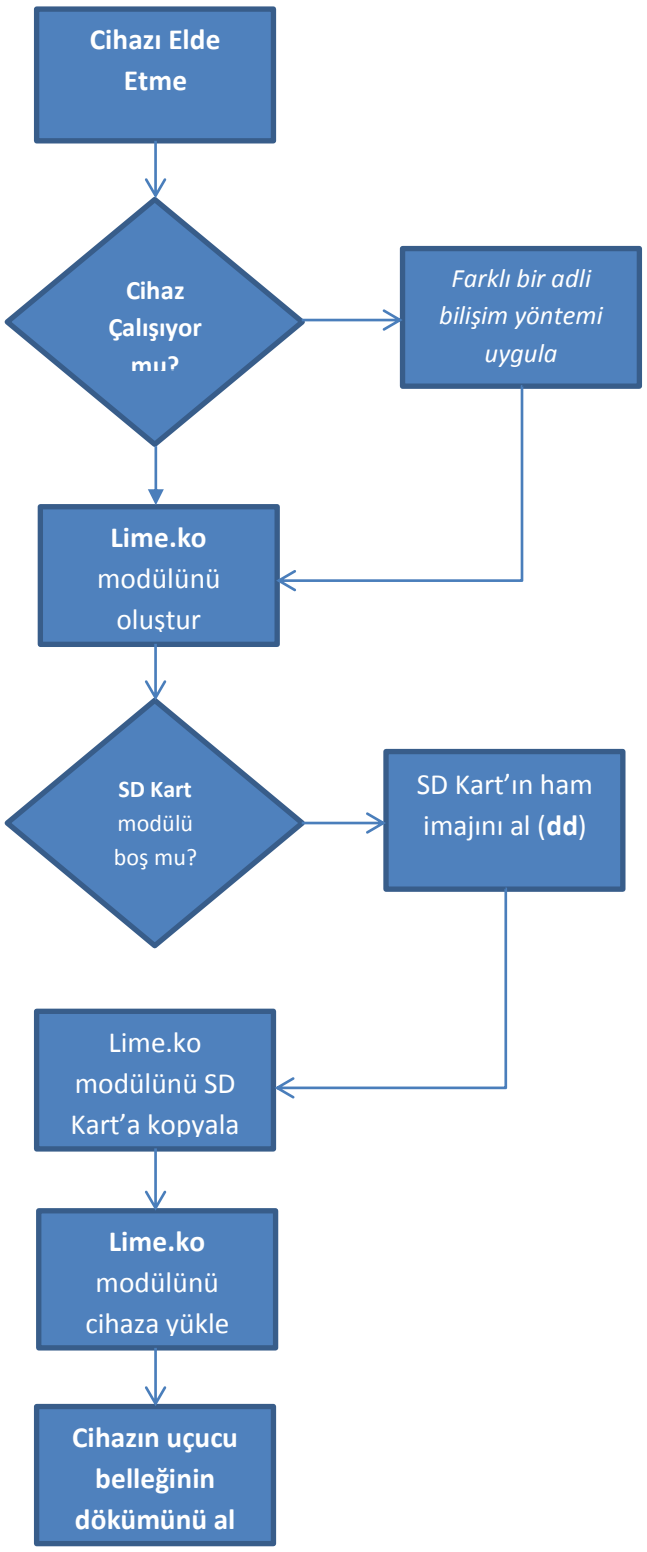
Bu nedenle yapılan analizler ve ilgili bulgular yukarıdaki ile sınırlıdır. Bu yazıda, bu sınırlamaların üstesinden gelmeyi, bir Android cihazının uçucu belleğini, adli olarak sağlam ve tamamen analiz etmenin üstesinden gelmeyi başardık. Çalışmamızda ortaya koyduğumuz adli olarak güvenilir bir delil elde etme süreci, taşınabilir aygıtın fiziksel belleğiyle aynı olan uçucu belleğin kopyalarını edinmemizi sağlar; prosedürün kendisi orijinal cihazı değiştirmez.

3. Uçucu Belleği (RAM) Elde Etme (memory dump)

Root'lu bir Android mobil cihazının uçucu belleğini boşaltmak için Linux bellek çıkarıcı (LiME) yazılımı olan açık kaynaklı bir adli araç kullandık [5]. LiME, yüklü bir çekirdek modülüdür ve uçucu belleğin Linux ve Linux tabanlı aygıtlardan (örneğin, Android tarafından desteklenenler) edinilmesini sağlar. LiME, ek olarak olarak sağlıklı bir şekilde (bellek sayfalarının yaklaşık yüzde 99'u) hafıza sayfalarını elde edebilir çünkü hedef cihazın kendisine ve bu cihazdan veri aktarımı yaparken fiziksel hafıza üzerindeki etkisini en aza indirir. Bunu başarmak için, LiME aşağıdaki özelliklerle tasarlanmıştır:

1) Sadece bir binary dosya (yani, LiME modülü), cihaza aktarılmalı ve hafıza edinimini gerçekleştirmek için çalıştırılmalıdır.
2) LiME modülü, çok küçük olduğu için (~ 70 KB) minimum bir bellek izine sahiptir.
3) LiME, bellek dökümünü gerçekleştirmek için çok az çekirdek işlevi (fonksiyonu) gerektirir.
4) Veri okuma ve yazma işlemi çekirdek içinde ele alındığından, LiME kullanıcı alalnı ile az etkileşim gerektirir. Bu yolla, LiME yüzlerce sistem çağrısı ile uçucu belleği değiştirecek diğer işlev çağrılarından kaçınır.

Şekil 1'de yer alan prosedür, bir Android mobil cihazından adli olarak sağlıklı bir bellek görüntüsü elde etmek için izlenecek usul ve gerekli adımları göstermektedir. Bu prosedürde, Android cihazının root'lu olduğunu varsayıyoruz. Root'lu değilse, aygıtta root ayrıcalıkları kazanmaya çalışmalıyız, çünkü LiME belleğin dölümünü alma işlemini gerçekleştirmek için root ayrıcalıkları gerektirir (Sylve et al., 2012). Android işletim sistemi için, normal kullanıcıdan root'a ayrıcalık yükselmesine izin veren çeşitli güvenilir yöntemler vardır (Abbott, 2012; Kramer, 2010) ve bunları daha fazla analiz etmiyoruz. Root'lama prosedürünün en önemli kısmı, uçucu bellek verilerini güç olmadan koruyamayacağından mobil aygıtın yeniden başlatılmadığından emin olmaktır. Ardından, Android gibi çalışan bir işletim sisteminin çekirdek işlevselliğini genişleten bir kod içeren bir nesne dosyası olan lime.ko adlı yüklenebilir bir çekirdek modülü oluşturduk. Lime.ko modülünü oluşturma işlemi ile ilgili teknik ayrıntılar şu adreste bulunabilir: (http: // code.google.com/p/lime-forensics). Lime.ko dosyasını oluşturduktan sonra, dd adlı bir açık kaynak görüntüleme aracı [5] kullanarak cihazın SD hafıza kartının adli olarak sağlıklı bir görüntüsünü elde ediyoruz. Bu adımı gerçekleştirmemizin nedeni, lime.ko modülünün SD karta kopyalanması gerektiğidir. Adli soruşturmalarda önemli bir ilke, delillerin değiştirilmesini önlemek olduğundan, lime.ko modülünü cihazın SD hafıza kartına kopyalamadan önce SD kartın ham görüntüsünü elde etmeliyiz. Bu şekilde, sonuçlarımızın adli olarak doğru olmasını sağlıyoruz.



Şekil 1 - Root'lu bir Android mobil cihazının uçucu belleğini edinme prosedürünün akış şeması

4. Deneyler

Bu bölümde gerçekleştirilen deneylerin analizi yer almaktadır. Üç aylık süre zarfında hassas kullanıcı verilerini ayrıntılı olarak inceleyen toplam 13 Android uygulaması kullandık.

Senaryo	Adımların açıklaması
Senaryo 1	
S1.a	Giriş yap, çıkış yap, hem döküm al
S1.b	Giriş yap, çıkış yap, 10 dakika bekle, döküm al
S1.c	Giriş yap, çıkış yap, 20 dakika bekle, döküm al
S1.d	Giriş yap, çıkış yap, 60 dakika bekle, döküm al
Senaryo 2	
S2.a	Giriş yap, kullan, çıkış yap,10 dk. telefon olarak kullan, döküm al
S2.b	Giriş yap, kullan, çıkış yap,20 dk. telefon olarak kullan, döküm al
S2.c	Giriş yap, kullan, çıkış yap,60 dk. telefon olarak kullan, döküm al
Senaryo 3	
S3.a	Giriş yap, kullan, çıkış yap,10 dk. akıllı telefon olarak kullan, döküm al
S3.b	Giriş yap, kullan, çıkış yap,20 dk. akıllı telefon olarak kullan, döküm al
S3.c	Giriş yap, kullan, çıkış yap,60 dk. akıllı telefon olarak kullan, döküm al
Senaryo 4	
S4.a	Giriş yap, uygulamayı geri planı gönder, hemen döküm al
S4.b	Giriş yap, uygulamayı geri planı gönder, 10dk. bekle, döküm al
S4.c	Giriş yap, uygulamayı geri planı gönder, 20dk. bekle, döküm al
S4.d	Giriş yap, uygulamayı geri planı gönder, 60dk. bekle, döküm al
Senaryo 5	
S5.a	Giriş yap, uygulamayı geri planı gönder, 10dk. telefon olarak kullan , döküm al
S5.b	Giriş yap, uygulamayı geri planı gönder, 20dk. telefon olarak kullan, döküm al
S5.c	Giriş yap, uygulamayı geri planı gönder, 60dk. telefon olarak kullan, döküm al
Senaryo 6	
S6.a	Giriş yap, uygulamayı geri planı gönder, 10dk. akıllı telefon olarak kullan , döküm al
S6.b	Giriş yap, uygulamayı geri planı gönder, 20dk. akıllı telefon olarak kullan, döküm al
S6.c	Giriş yap, uygulamayı geri planı gönder, 60dk. akıllı telefon olarak kullan, döküm al
Senaryo 7	
S7	Giriş yap, uygulamayı kullan, görev sonlandırıcıyı ile uygulamayı kapat, hemen döküm al
Senaryo 8	
S8.a	Giriş yap, kullan, çıkış yap, telefon uçak moduna al, hemen döküm al
S8.b	Giriş yap, kullan, çıkış yap, telefon uçak moduna al, 10dk. bekle, döküm al
S8.c	Giriş yap, kullan, çıkış yap, telefon uçak moduna al, 20dk. bekle, döküm al
S8.d	Giriş yap, kullan, çıkış yap, telefon uçak moduna al, 60dk. bekle, döküm al
Senaryo 9	
S9.a	Giriş yap, kullan, çıkış yap, telefonu uçak moduna al, oyun uygulamalarını 10dk. oyna, döküm al
S9.b	Giriş yap, kullan, çıkış yap, telefonu uçak moduna al, oyun uygulamalarını 20dk. oyna, döküm al
S9.c	Giriş yap, kullan, çıkış yap, telefonu uçak moduna al, oyun uygulamalarını 60dk. oyna, döküm al
Senaryo 10	
S10.	Giriş yap, kullan, çıkış yap, cihazı yeniden başlat, hemen döküm al
Senaryo 11	
S11.	Giriş yap, kullan, çıkış yap, cihazı kapat, cihazın pilini çıkart, 5sn bekle, pili geri tak, cihazı başlat, hemen döküm al

Tablo 1 - Uygulanan Senaryoların Özeti

İncelenen uygulamaların çoğunluğunun güncelleştirmeleri sık sık yayınlanmaktadır. Şekil-1'de Root'lu bir Android mobil cihazının uçucu belleğini edinme prosedürünün akış şeması yer alıyor. Çalışmada incelenen her bir uygulama, hareket halindeki veriler olarak bilinen bir kullanıcı adı ve / veya şifre kullanmaktadır. Sağlanan işlevselliğe dayanarak, temel alınan mobil uygulamalar dört kategoriye ayrılmıştır:

1) Mobil bankacılık (m-bankacılık) uygulamaları
2) E-alışveriş / finansal uygulamalar (mobil kullanıcıların çevrimiçi ödemeleri gerçekleştirmesini veya mal satın almasını kolaylaştıran uygulamalar)
3) Şifre yöneticileri (örn. Kullanıcı şifreleri)
4) Şifreleme ve veri gizleme uygulamaları (yani kullanıcının mobil cihazında gizliliğini artırmayı amaçlayan uygulamalar)

Mobil-bankacılık ve e-alışveriş / finansal uygulamalar, mobil kullanıcıların kimliğini doğrulamak için kullanıcı adlarını ve şifreleri kullanır ve verilen hizmetlere uzaktan erişime olanak tanır. Öte yandan, şifre yöneticileri, şifreleme ve veri gizleme uygulamaları, diğer uygulamaların şifrelerini şifrelemek / şifrelerini çözmek veya diğer kullanıcıların hassas verileri için anahtarlar sağlar. Mobil-bankacılık kategorisi, Yunanistan'da altı büyük bankanın mobil bankacılık uygulamasını (yani, bank1, bank2, bank3, bank4, bank5

ve bank6) kapsamaktadır. E-alışveriş / finansal kategori, finansal işlemleri mümkün kılan tanınmış, uluslararası, e-ticaret işletmelerinin üç uygulamasından (finansal1, finansal2, finansal3) oluşur. Parola yöneticisi kategorisi, parolaları güvenli bir şekilde depolayan ve alan iki uygulamadan (yani, parola1, parola2) oluşur. Diğer tüm mobil kullanıcıların şifrelerini korumak için bir ana şifre kullanırlar. Son olarak, dördüncü kategori, iki uygulama (yani sifreleme1, sifreleme2) içermektedir. Şifreleme1, mobil kullanıcı tarafından gönderilen ve alınan mesajları (yani SMS'ler) ve e-postaları şifrelemek için gizli bir şifreyi anahtar olarak kullanmaktadır. Öte yandan, sifreleme2 uygulaması, belirli bir kişiden gelen aramaları ve mesajları mobil cihazın kişi listesinden gizler ve onları tasvir etmenin tek yolu, cihaz sahibinin ilgili uygulamanın ara yüzüne gizli bir kod girip sonra da Arama tuşuna basmasıdır. Böylece kullanıcı uygulamanın gizlediği kayıtlara ulaşabilir.

Test ortamımız köklü bir Samsung Galaxy S Plus (i9001) ile donatılmıştır. Bu akıllı telefon, Google'ın istatistiklerine göre en popüler Android sürümünü (diğer bir deyişle, bu Android sürümünü çalıştıran cihazların göreceli sayısına dayanıyor) olan bir Android v2.3'ünü kullanıyor. Bu çalışmada farklı zaman aralıkları temel alan on bir (11) genel senaryoyu, otuz farklı alt senaryoya bağlayarak inceledik (bkz. Tablo 1), yani bellek dökümlerini uçucu belleği incelenen cihazdan farklı zaman aralıklarında elde ettik. Her bir senaryonun her varyasyonunda, kendi kimlik bilgilerimizi kullanarak tüm uygulamaları inceledik. Tüm senaryolar için, bellek dökümü boyutu yaklaşık 512 MB iken (Samsung Galaxy S plus'ın fiziksel belleğinin boyutuna eşittir), döküm işlemi her biri için 9 dakika sürmüştür. Araştırılan senaryolar ve varyasyonları kısaca aşağıda açıklanmıştır:

- **Senaryo 1:** İncelenen uygulamalardan giriş yapıldı, kullanıldı ve oturumdan çıkılarak aşağıdaki zaman aralıklarında bellek dökümüne başlandı:

  a) çıkıştan hemen sonra
  b) 10 dakika bekledikten sonra
  c) 20 dakika bekledikten sonra
  d) 60 dakika bekledikten sonra.

  Bekleme süresi boyunca, mobil cihazı boşta tutuyoruz (başka bir deyişle, herhangi bir kullanım olmadan açılıyor).
- **Senaryo 2:** İncelenen uygulamalara giriş yapıldı, kullanıldı ve oturumdan çıkıldı. Aşağıdaki bekleme sürelerinin ardından bellek dökümü alınmaya başlandı:
  a) 10 dak
  b) 20 dak
  c) 60 dak.

  Bekleme süresi boyunca, mobil cihazı yalnızca telefon olarak kullanıldı; bu, telefon görüşmesi ve kısa mesaj gönderme ve alma anlamına gelir.

- **Senaryo 3:** İncelenen uygulamalara giriş yapıldı, kullanıldı ve oturumdan çıkıldı. Aşağıdaki bekleme sürelerinin ardından bellek dökümü alınmaya başlandı:
  a) 10 dak
  b) 20 dak
  c) 60 dak.

  Bekleme süresi boyunca, Gmail, Play Store, YouTube, MP3 çalar, Haber okuyucu uygulaması vb. Gibi çeşitli yaygın Android uygulamalarını etkinleştirerek mobil cihaz yalnızca akıllı telefon olarak kullanıldı.

- **Senaryo 4:** İncelenen uygulamalara giriş yapıldı, kullanıldı ve ancak çıkış yapılmadı (oturum kapatılmadı). Bunun yerine, mobil cihazın ana ekran düğmesine basarak arka planda çalışacak şekilde ayarlandı. Aşağıdaki şartlarda bellek dökümüne başlandı:

  a) Uygulamayı arka plana gönderdikten hemen sonra
  b) 10 dakika bekledikten sonra
  c) 20 dakika bekledikten sonra
  d) 60 dakika bekledikten sonra.

  Bekleme süresi boyunca, mobil cihazı boşta

tutuyoruz (başka bir deyişle, herhangi bir kullanım olmadan çalışıyor). Bu senaryonun seçilmesinin nedeni pek çok kullanıcı çıkış yapmadan ve uygulamaları doğru bir şekilde kapatmadan, yalnızca giriş düğmesine basıp ana ekrana dönüyorlar.

- **Senaryo 5:** Giriş yapılan uygulamalar kullanıldı, ancak onlardan çıkış yapılmadı. Bunun yerine, mobil cihazın ana ekran düğmesine basarak arka planda çalışacak şekilde ayarlandı. Hafızanın dökümü almaya başlıyoruz:
  a) 10 dakika bekledikten sonra
  b) 20 dakika bekledikten sonra
  c) 60 dakika bekledikten sonra
  Bekleme süresi boyunca, mobil cihazı yalnızca telefon olarak kullanıldı; bu telefon görüşmesi ve kısa mesaj gönderilip ve alındığı anlamına gelir.

- **Senaryo 6**: Giriş yapan uygulamaları kullandık, ancak onlardan çıkış yapmadık. Bunun yerine, mobil cihazın ana ekran düğmesine basarak arka planda çalışacak şekilde ayarladık. Bellek dökümünü almaya başlıyoruz:
  a) 10 dakika bekledikten sonra
  b) 20 dakika bekledikten sonra
  c) 60 dakika bekledikten sonra.
  Bekleme süresi boyunca, Gmail, Play Store, YouTube, MP3 çalar, haber okuyucu uygulaması vb. çeşitli yaygın Android uygulamalarını etkinleştirerek mobil cihazı yalnızca akıllı telefon olarak kullandık.

- **Senaryo 7**: Giriş yapan uygulamaları kullandık ve incelenen uygulamalardan çıkış yaptık. Çıkış yaptıktan sonra, çalışan tüm işlemleri sonlandırmak için bir görev sonlandırıcı çalıştırdık ve hemen sonra cihazın uçucu belleğinin dökümünü aldık.

- **Senaryo 8**: İncelenen uygulamalara giriş yaptık, kullandık ve oturumdan çıktık. Çıkış yaptıktan sonra cihazı uçak moduna geçiriyoruz (diğer bir deyişle, tüm iletişim arayüzlerini, kablosuz, 3G vb. devre dışı bırakıyoruz). Belleğin dökümünü almaya başlıyoruz:
  a) geçişten hemen sonra
  b) 10 dakika bekledikten sonra
  c) 20 dakika bekledikten sonra
  d) 60 dakika bekledikten sonra.
  Bekleme süresi boyunca, mobil cihazı boşta tutuyoruz (başka bir deyişle, herhangi bir kullanım olmadan açılıyor).

- **Senaryo 9:** İncelenen uygulamalardan giriş yaptık, kullandık ve oturumdan çıktık. Çıkış yaptıktan sonra cihazı uçak moduna geçiriyoruz (diğer bir deyişle, tüm iletişim ara yüzlerini, kablosuz, 3G vb. devre dışı bırakıyoruz). Bellek dökümü alma işlemine başlıyoruz:
  a) 10 dakika bekledikten sonra
  b) 20 dakika bekledikten sonra
  c) 60 dakika bekledikten sonra.
  Bekleme süresi boyunca sadece oyun uygulamalarını kullanıyoruz.

- **Senaryo 10**: İncelenen uygulamalardan giriş yaptık, kullandık ve oturumdan çıkış yaptık. Çıkış yaptıktan sonra, mobil cihazı kapattık ve ardından cihazı tekrar açtık (yani, yeniden başlatma). Yeniden başlattıktan hemen sonra cihazın uçucu belleğinin dökümünü aldık.

- **Senaryo 11**: İncelenen uygulamalardan giriş yaptık, kullandık ve oturumdan çıkış yaptık. Çıkış yaptıktan sonra mobil cihazı kapattık ve pili 5 saniye boyunca çıkardık. Ardından, çıkarılan pili geri taktık ve mobil cihazı açtık, önyüklemenin tamamlanmasından hemen sonra cihazın uçucu belleğinin dökümünü aldık.

Araştırılan her bir başvuru ve incelenen senaryo veya senaryo varyasyonu için, her biri farklı amaçlarla iki deney yapmış bulunmaktayız. Birincisi, kendi gönderilen kimlik bilgilerimizi mobil cihazın bellek dökümünden kurtarabilir miyiz kontrol etmek oldu. Öte yandan, ikinci denemede, kimlik bilgilerinin bir bellek dökümünde nerelerde bulunduğunu gösteren kalıpları ve ifadeleri bulmak amaçlandı. Bu, araştırmacıların

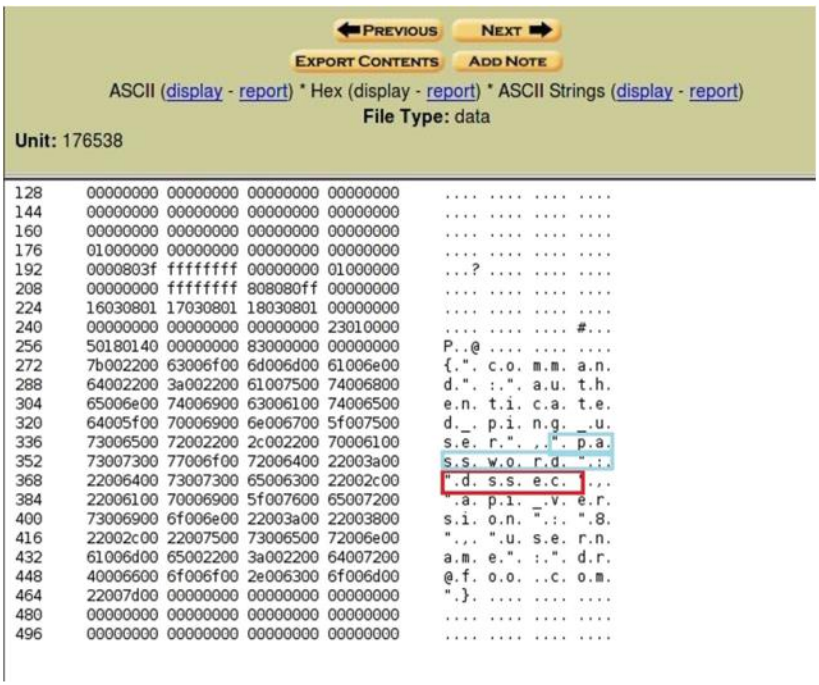
Android cihazlarının bellek imgelerine sahip olduğu adli araştırmalarda yararlı olacak ve bilinmeyen kimlik bilgilerini bulmak için bu kalıpları kullanabileceklerdir. Aksine olumsuz bir yan etki olarak, kötü amaçlı bir yazılım bir Android cihazını ele geçirirse, cihazın sahibinin kimlik bilgilerini bulmak için bu kalıpları deneyecektir.

Deneyleri gerçekleştirmek için, incelenen her uygulama için aşağıdaki adımları tekrarladık ve senaryoyu inceledik. Önce, kendi oluşturduğumuz kimlik bilgilerinizi (ör. Kullanıcı adı ve / veya şifre) göndererek rasgele seçtiğimiz araştırma kapsamındaki uygulama

(test ortamı cihazımızda zaten yüklü) içine giriyoruz. Uygulamayı 2 ila 10 dakika arasında değişen keyfi bir süre için kullandıktan sonra, her kabul edilen senaryoda açıklanan belirli işlemleri gerçekleştirdik. Daha sonra, Bölüm 3'te açıklanan prosedürü kullanarak cihazın uçucu hafızasını boşaltıyoruz. Son olarak, The Sleuth Kit (TSK) [7] adlı başka bir açık kaynak kodlu adli bilişim aracını kullanarak, gönderilen kimlik bilgilerini. TSK, adli araştırmalar yapmak ve Windows, Linux ve Unix bilgisayarlarının görüntülerinden veri çıkarmak için kullanılır. Meta veri girdilerini bulmak, bir dosya sisteminde veri bloklarını görüntülemek ve bir dosya sisteminde ayrılmış ve ayrılmamış dosya adlarını aramak için çeşitli yardımcı programları içerir.

5. Elde edilen sonuçlar

İlk deneme grubunda, neredeyse hiç değişiklik yapmadan, düz metin oldukları için, başvuruların çoğunda kendi tanımladığımız kimlik bilgilerini başarılı bir şekilde bulduk. Bazı durumlarda, bellek görüntüleri içindeki alınan kimlik bilgilerinin karakterleri nokta simgesi ile ayrılmıştı. Örneğin, bir uygulamanın gönderilen şifresinin "password" ifadesi olması durumunda, hafıza görüntüsünde "p.a.s.s.w.o.r.d." ifadesini bulduk. Bu basit değişikliğin nedeni, kullanılan Unicode kodlamaya (yani, UTF-16) bağlı olmuştur. Aynı uygulamada, şifre dizesinin karakterlerinin HTML kodlanmış olduğunu gözlemledik. Örneğin, uygulamanın şifresinin "p @ ssword!" Dizesi olması durumunda, bellek görüntüsünde "p% 40ssword% 21" dizesini bulduk. Aşağıdaki resimde (Şekil 2), keşfedilen şifre dizisi ".d.s.s.e.c" i, karakterleri nokta sembolü ile ayrılmış olan düz metin halinde içeren, TSK açık kaynaklı adli araç kullanan bir bellek anlık görüntüsü sunulmaktadır.



Şekil 2- Bulunan parolalar olan "dssec" ve "password:" desenlerini gösteren bir bellek dökümünün anlık görüntüsü

Aşağıdaki tablo (Şekil 3) araştırılan her bir başvuru için (aynı zamanda kategoriler halinde gruplandırılmış) tüm bulguları sunar ve senaryoyu inceleyen ilk deney setinin sonuçlarını özetler (diğer bir deyişle, gönderilen kimlik bilgilerinin geri kazanımı). Aşağıdaki tabloda U ve P harflerini içeren hücreler, belirli uygulama ve kabul edilen senaryo için kullanıcı adının ve şifrenin başarıyla geri yüklendiğini gösterir. Ayrıca, X harfini içeren gri renkli hücreler bir kullanıcı adı veya parolanın başarısız biçimde kurtarılmasını ifade eder. Son olarak, çizgi sembolünü içeren hücreler, ilgili uygulamaların bir kullanıcı adı kullanmadığını (yani yalnızca bir şifre kullandıklarını) ifade eder.

		Applications																								Total	Total per scenario
		m-banking												financial/e-shopping						password managers				encryption/hiding					
		bank1		bank2		bank3		bank4		bank5		bank6		financial1		financial2		financial3		password1		password2		encryption1		encryption2			
Scenario 1	s1.a	U	P	U	P	U	P	U	P	U	P	X	X	U	P	U	P	U	P	-	P	-	P	-	P	-	P	20/22	71/88 80%
	s1.b	U	P	U	P	U	P	U	P	U	P	X	X	U	P	U	P	U	X	-	P	-	P	-	P	-	P	19/22	
	s1.c	U	P	U	P	U	P	U	P	U	P	X	X	U	X	U	P	U	X	-	P	-	P	-	P	-	P	18/22	
	s1.d	U	P	U	P	U	P	U	P	U	P	X	X	U	X	X	X	X	X	-	P	-	P	-	X	-	P	14/22	
Scenario 2	s2.a	U	P	U	P	U	P	U	P	U	P	X	X	U	P	U	P	U	X	-	P	-	P	-	P	-	P	19/22	51/66 77%
	s2.b	U	P	U	P	U	P	U	P	U	P	X	X	U	X	U	P	U	X	-	P	-	P	-	P	-	P	18/22	
	s2.c	U	P	U	P	U	P	U	P	U	P	X	X	U	X	X	X	X	X	-	P	-	P	-	X	-	P	14/22	
Scenario 3	s3.a	X	X	U	P	U	P	U	P	U	P	X	X	U	X	U	X	U	X	-	X	-	X	-	P	-	P	13/22	32/66 48%
	s3.b	X	X	U	P	U	X	U	P	U	P	X	X	U	X	U	X	U	X	-	X	-	X	-	P	-	P	12/22	
	s3.c	X	X	X	X	U	X	X	X	U	P	X	X	U	X	U	X	U	X	-	X	-	X	-	X	-	P	7/22	
Scenario 4	s4.a	U	P	U	P	U	P	U	P	U	P	U	P	U	P	U	P	U	P	-	P	-	P	-	P	-	P	22/22	71/88 80%
	s4.b	U	P	U	P	U	P	U	P	U	P	X	X	U	P	U	P	U	P	-	P	-	X	-	P	-	P	19/22	
	s4.c	U	P	U	P	U	P	U	P	U	P	X	X	U	P	U	P	U	P	-	P	-	X	-	P	-	P	19/22	
	s4.d	U	P	U	P	U	X	X	X	U	P	X	X	U	X	U	X	X	X	-	P	-	X	-	X	-	P	11/22	
Scenario 5	s5.a	U	P	U	P	U	P	U	P	U	P	X	X	U	P	U	P	U	P	-	P	-	X	-	P	-	P	19/22	49/66 74%
	s5.b	U	P	U	P	U	P	U	P	U	P	X	X	U	P	U	P	U	P	-	P	-	X	-	P	-	P	19/22	
	s5.c	U	P	U	P	U	X	X	X	U	P	X	X	U	X	U	X	X	X	-	P	-	X	-	X	-	P	11/22	
Scenario 6	s6.a	U	P	U	P	U	P	U	P	U	P	X	X	U	P	U	P	U	P	-	P	-	X	-	P	-	P	19/22	48/66 72%
	s6.b	U	P	U	P	U	P	U	P	U	P	X	X	U	P	U	P	U	P	-	P	-	X	-	P	-	P	19/22	
	s6.c	U	P	U	P	U	X	X	X	U	P	X	X	U	X	X	X	X	X	-	P	-	X	-	X	-	P	10/22	
Scenario 7	s7	U	P	U	P	U	P	U	P	U	P	X	X	X	X	U	P	X	X	-	P	-	P	-	P	-	P	16/22	16/22 72%
Scenario 8	s8.a	U	P	U	P	U	P	U	P	U	P	X	X	U	P	X	X	U	X	-	X	-	X	-	X	-	P	14/22	51/88 58%
	s8.b	U	P	U	P	U	P	U	P	U	P	X	X	U	X	X	X	U	X	-	X	-	X	-	X	-	P	13/22	
	s8.c	U	P	U	P	U	P	U	P	U	P	X	X	U	X	X	X	U	X	-	X	-	X	-	X	-	P	13/22	
	s8.d	U	P	U	P	U	P	U	P	U	P	X	X	X	X	X	X	X	X	-	X	-	X	-	X	-	P	11/22	
Scenario 9	s9.a	X	X	X	X	X	X	X	X	U	P	X	X	U	X	X	X	U	X	-	X	-	X	-	X	-	P	5/22	11/66 16%
	s9.b	X	X	X	X	X	X	X	X	U	P	X	X	X	X	X	X	X	X	-	X	-	X	-	X	-	P	3/22	
	s9.c	X	X	X	X	X	X	X	X	U	P	X	X	X	X	X	X	X	X	-	X	-	X	-	X	-	P	3/22	
Scenario 10	s10	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-	X	-	X	-	X	-	X	0/22	0/22 0%
Scenario 11	s11	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-	X	-	X	-	X	-	X	0/22	0/22 0%
Total		22/30	22/30	24/30	24/30	25/30	20/30	21/30	21/30	28/30	28/30	1/30	1/30	24/30	11/30	18/30	13/30	19/30	8/30	-	18/30	-	9/30	-	15/30	-	28/30		
Total per category		237/360 - 65%												93/180 - 51%						27/60 - 45%				43/60 - 71%					

Şekil 3 - Denenen her uygulama senaryo için (diğer bir deyişle, kimlik bilgilerinin geri kazanımı) sonuçları, kategoriler halinde gruplandırılmıştır.

Şekil 3, incelenen uygulamanın desteklediği gizlilik seviyesinin yanı sıra, farklı kullanım koşulları altında Android mobil cihazların uçucu hafızanın davranışıyla ilgili bazı ilginç sonuçları ortaya koymaktadır. Daha spesifik olarak, deneme senaryosu başına sonuçlarını inceleyerek, 1. ve 4. senaryolarda, gönderilen kimlik doğrulama bilgilerinin çoğunun keşfedildiğini (diğer bir deyişle her ikisinde de yüzde 80) görebiliriz. Bu, her iki senaryoda da gözlemlenmeye neden olan bellek dökümünü gerçekleştirmeden önce mobil aygıtın boşta kalması gerçeğine atfedilebilir.

Gözlem 1: Kullanıcı mobil cihazı kullanmadığı sürece (yani açıkken ve boşta kaldığı sürece), cihazın uçucu belleğinde kimlik doğrulama bilgilerinin (yani hareket halindeki veriler) bozulmadan kalması muhtemeldir.

Öte yandan, senaryolar 10 ve 11'de, kimlik doğrulama bilgileri daima uçucu bellekten silindi ve bunlardan hiçbirini bulamadık (yani her ikisinde de yüzde 0).

Gözlem2: Bir taşınabilir aygıtın geçici belleğinin herhangi bir kimlik doğrulama sertifikası (veya diğer hassas veriler) içermediğinden emin olmanın en iyi yolu, cihazı yeniden başlatmak veya pilini çıkarmaktır. Bu, masaüstü/dizüstü bilgisayarlar için de kanıtlanmıştır (Karayianni ve ark., 2012). Bununla birlikte, bu gözlemi çok kritik hale getiren mobil aygıtların ve masaüstü/ dizüstü bilgisayarların kullanımında temel bir farklılık var. Diğer bir deyişle, masaüstü/ dizüstü bilgisayar kullanıcılarının bellek kayıtları bilgisayarın yeniden başlatılması ile silinir ve yeniden oluşturulur. Öte yandan, mobil cihaz kullanıcıları nadiren cihazlarını ve uygulamaları kapatır veya yeniden başlatır. Aslında mobil kullanıcılar, hiçbir telefon görüşmesini kaçırmamak için cihazlarını mümkün olduğunca kapamama veya yeniden başlatmama eğilimindedirler. Bu nedenle, taşınabilir aygıtların uçucu belleğinde masaüstü / dizüstü bilgisayarlardan çok kimlik doğrulama bilgisi keşfetmek olasılık dahilindedir.

Değişken süreli (örneğin senaryo 1, 2, 3, 4, 5, 6, 8, 9 gibi) tüm senaryolardan kaynaklanan ilginç bir gözlem de (yani gözlem3), bekletilen sürenin uzunluğuna bağlı olarak bellek görüntüsü, daha az kimlik doğrulama bilgisi sağlamıştır.

Gözlem 3: Zaman bellekteki hassas verilerin güvenliği ile doğrudan ilgilidir. Bir kullanıcının kimlik doğrulama bilgilerini gönderdiği andan itibaren geçen süre ne kadar fazla olursa kimlik doğrulama kimlik bilgisi de silinir. İncelenen uygulamaların arka plana yerleştirildiği senaryolar 4, 5 ve 6'da, sırasıyla %80, %74 ve %72 oranında kimlik doğrulama bilgilerini keşfedebildik. Ayrıca, senaryo 7'de, araştırılan başvuruların görev sonlandırıcı uygulamayla sonlandırılmış olmasına rağmen, kimlik doğrulama kimlik bilgilerini % 72'lik bir oranda keşfedebildik. Bu gerçeklerden yola çıkarak aşağıdaki gözlemleri yaparız:

Gözlem4: Çalışan bir uygulamanın arka plana ayarlanması, kimlik doğrulama bilgilerini mobil aygıtın uçucu belleğinden silmez. Bu, kullanıcılar arasında, düzgün bir şekilde çıkış yapmak yerine, çalışan uygulamaları arka plana göndermek yaygın bir davranıştır ve endişe verici bir sonuçtur.

Gözlem5: Çalışan bir uygulamayı sonlandırmak için görev sonlandırıcı uygulamasını kullanmak, ilgili kimlik doğrulama bilgilerini cihaz belleğinden silmez. 2. ve 3. senaryolardaki sonuçları karşılaştırarak, birincisinde, kimlik doğrulama kimlik bilgilerini yüzde 77'lik bir yüzdede bulabildiğimizi fark ettik; İkincisinde de sadece% 48 oranında. Buna dayanarak gözlem 6'yı yorumlayabiliriz:

Gözlem 6: Bir mobil cihazı akıllı telefon olarak kullanmak (yani, çeşitli Android uygulamalarını etkinleştirmek), cihazın geçici belleğinde bulunan kimlik doğrulama bilgilerini etkilemez (Örn., Aramaları cevaplamak / yapmak ve SMS gönderip almak). Bunun nedeni, çalışan bir

Android uygulaması, daha önce depolanan verileri cihazın geçici belleğine yazar. Öte yandan, telefon görüşmeleri veya SMS gönderme / alma gibi işlemler, mobil cihazın uçucu belleğini etkilemez ve belleğin içeriği korunur. Senaryo 8 ve 9'da (diğer bir deyişle, cihazı uçak moduna geçirirken) sırasıyla %58 ve %16 oranında kimlik doğrulama bilgilerini keşfedebildik. Bu iki senaryonun arasındaki önemli fark, senaryo 9'da cihazın uçak moduna geçmesinden sonra bir oyun uygulaması başlattığımızı ve oynadığımız gerçeğinden kaynaklanıyor olabilir; Senaryo 8'de cihaz boşta kaldı. Sonuç olarak, aşağıdaki gözlemin sonucunu çıkartabiliriz.

Gözlem 7: Mobil cihazın uçak moduna geçirilmesi, cihazların uçucu hafızanın içeriğinin silinmesi anlamına gelmez ve böylece kimlik doğrulama bilgileri geri kazanılabilir. Bununla birlikte, geçiş yaptıktan sonra mobil kullanıcı oyun gibi bir uygulamayı etkinleştirir ve çalıştırırsa, cihazın geçici belleğinde bulunan kimlik doğrulama kimlik bilgilerinin çoğunun silindiğini fark ettik.

Uygulamaların bakış açısından, test edilen uygulamaların tümü için en az bir kere kimlik doğrulama bilgilerini bulabildiğimizi düşünüyoruz. Özellikle, mobil bankacılık ve finansal / e-alışveriş uygulamaları kategorisinde sırasıyla %65 ve %51 oranında gönderilen kimlik doğrulama bilgilerini keşfedebildik. Benzer şekilde, şifre yöneticileri grubunda olduğu kadar şifreleme / gizleme uygulamaları da keşif oranımız sırasıyla %45 ve %71'e ulaştı. Bu bulgulara dayanarak, aşağıdaki gözlemleri çıkarabiliriz:

Gözlem 8: İncelenen Android uygulamalarının çoğunluğu, geçici bellekteki kimlik doğrulama bilgilerinin kurtarılmasına karşı savunmasızdır.

Gözlem 9: Mobil bankacılık uygulamaları gibi güvenliği öncelikli uygulamalarının dahi, kimlik doğrulama bilgilerinin bulunmasına karşı savunmasız olduğu kanıtlanmıştır. Kimlik doğrulama bilgilerini neredeyse tüm senaryolarda bulduğumuzdan (ör., Senaryolar 10 ve 11 hariç) bankacılık uygulamalarından bank5 en savunmasızıydı. Öte yandan, en güvenli uygulama yalnızca 4. senaryo (yani, çalışmakta olan uygulamayı arka plana ayarlamak ve hemen cihazın uçucu belleğinin dökümünü almak) için kimlik doğrulama kimlik bilgilerini keşfettiğimiz bank6'ydı. Finansal / online alışveriş uygulamalarında, gönderilen kullanıcı adlarının iyileşme yüzdesi parolaların yüzdesini aştı. Parola yöneticileri, kimlik doğrulama bilgilerinin keşfedilmesine karşı savunmasız davrandılar, ancak bu, password2 uygulamasında password1 'e göre daha sık oldu. Son olarak, şifreleme uygulaması1, 10. ve 11. senaryolar dışında neredeyse tüm incelenen senaryolara karşı savunmasız olan uygulama şifreleme uygulaması 2'ye kıyasla daha sağlam olduğunu kanıtladı. Bu bulgular aşağıdaki gözlemlere yol açmaktadır.

Gözlem10: Kimlik doğrulama bilgilerinin keşfedilebilmesi tehdidi altında güvenli olan bazı Android uygulamaları olduğunu öğrendik (ör. Bank6 uygulaması); Tehditlere tamamen maruz kalan diğer bazıları (örneğin, şifreleme2 ve bank5 uygulamaları) da vardır. Bu çelişkili sonuçlar, bazı uygulamaların güvenlik önlemleri göz önüne alınarak geliştirildiğini, diğer bir deyişle güvenlik önleminin dikkate alınmadığını göstermektedir.

Gözlem11: Değerlendirilen uygulamaların kritikliğine bakılmaksızın, tüm geliştiriciler, mobil platformlar tarafından sağlanan güvenlik seviyesini arttırmak için doğru ve güvenli programlama tekniklerini kullanmalıdır (diğer bir deyişle, kimlik doğrulama bilgilerini kullanılmadıklarında uygulama belleğinden silmelidirler).

Gözlem12: Parolalarını koruyarak kullanıcıların gizliliğini arttırmayı amaçlayan şifre yöneticileri, savunmasız oldukları bulundu. Bu, bir kullanıcı cihazını kaybederse, kötü niyetli bir kişi,

yalnızca, çalışan şifre yöneticisi uygulamasının ana şifresini keşfederek kullanıcının tüm şifrelerini keşfedebilir.

Kullanıcı adı	Parola
j_username=	j_password=
username=	password=
userid>	password:
login i:type=	pass i:type:

Tablo 2 - Keşfedilen kullanıcı adı ve parolaların desenleri

Denemelerin ikinci kümesinde, incelenen uygulamaların tümü için yakalanan bellek görüntüleri içinde kimlik bilgilerinin yerini (ör. Kullanıcı adı ve şifre) belirten belirli kalıp ve ifadeler oluşturuldu. Örneğin, Şekil-2'de de görülebileceği üzere, gönderilen şifrenin (ör."`dssec"`) hemen önüne, "`password:`" dizesini, ardından gelen dizenin kullanıcının şifresi olduğunun belirtildiği bir yöntem kullandık. Şifreler için bulduğumuz diğer bazı kalıplar (incelemesi yapılan uygulamalarda) "`password=`", "`j_password=`" ve "`pass i: type`" (bkz. Tablo 2). Bu nedenle, "`password`" veya "`pass`" ifadesi, gönderilen şifrelerin depolandığı fiziksel konumun, incelenen uygulamaların tümü için açık metin olarak belirtilmesini sağladı. Benzer şekilde, "`j_username=`", "`username=`", "`userid`" ve "`login i: type=`" (bkz. Tablo 2) verilen kullanıcı adlarının konumunu keşfetmeyi sağlayan ifade desenleridir. Dolayısıyla, kullanıcı adı, kullanıcı kimliği veya oturum açma desenleri, alınan hafıza görüntülerinde kullanıcı adlarının nerede saklandığının bulunabilmesini sağlar. Bir adli araştırmacı (veya kötü niyetli bir kişi), Android cihazının uçucu belleğinin görüntüsünü alıp cihaz sahiplerinin kullanıcı adlarını ve şifrelerini keşfetmek için bu kalıplar ile arama gerçekleştirebilir. Bu ifadelerin ve kalıpların tanımlanması, son gözlemi açıklar:

Gözlem 13: Her uygulamanın kimlik doğrulama bilgilerinin tam olarak bir bellek dökümünde bulunduğu noktaları belirten kalıp ve ifadelerin varlığını kanıtladık. Bu nedenle, kötü amaçlı biri kimlik doğrulama bilgilerini, çalınan bir cihazdan, bu kalıplar veya ifadeler ile bellek dökümünde arayarak kolayca kurtarabilir. Bunun tersine, ilgili geliştiriciler, sağlanan mobil uygulamalarda bu kalıpları veya ifadeleri kullanmaktan kaçınmalıdır.

6. Sonuç

Bu çalışmada, Android mobil uygulamalarının gizliliğini inceledik ve değerlendirdik. Özellikle, açık kaynaklı adli bilişim araçlarını kullanarak Android mobil cihazların uçucu belleğindeki kimlik doğrulama bilgilerinin keşfedilip keşfedilemeyeceğini inceledik. Sonuçların analizi, incelenen Android uygulamalarının çoğunun uçucu bellekteki kimlik doğrulama bilgilerini kurtarma konusunda savunmasız olduğunu ortaya koydu. Mobil bankacılık uygulamaları gibi güvenlik öncelikli uygulamaların bile savunmasız olduğu kanıtlandı. Dahası, uçucu belleğin yalnızca kimliği doğrulama bilgilerini içermediğini ancak cihazı yeniden başlattığımızda veya pilini çıkardığımızda gözlemledik. Ayrıca, uygulamanın kimlik doğrulama kimlik bilgilerinin nerede, tam olarak bir bellek dökümünde bulunduğunu gösteren kalıp ve ifadelerin varlığını kanıtladık. .Son olarak, kullanıcıların çeşitli web sitelerinde ve uygulamalarda aynı şifreyi tekrar kullanma eğiliminde olduklarını göz önüne alarak; tüm geliştiricilerin, uygulamanın kritikliğine bakılmaksızın, doğru ve güvenli programlama teknikleri ve yönergelerini kullanmaları gerektiği sonucuna vardık. Kimlik doğrulama bilgisi keşfini engellemek ve mobil platformlar tarafından sağlanan gizlilik düzeyini arttırmak için, incelenen senaryolardan yararlanılmalıdır.

Referanslar

[1] - http://blog.flurry.com/bid/88867/iOS-and-Android-Adoption-Explodes-Internationally

[2] - http://mobworld.wordpress.com/ 2010/07/05/memory-management-in-Android/

[3] - http://developer.Android.com/tools/debugging/ddms.html

[4] - http:// thomascannon.net/projects/android-reversing

[5] - http://www.forensicswiki.org/wiki/dd

[6] - http://developer.android.com/about/dashboards/index.html

[7] - http:// www.sleuthkit.org/sleuthkit/index.php

Orijinal Makalenin Kaynakları

- Abbott D. Linux for embedded and real-time applications. 3rd ed. December 2012 [chapter 7].
- Apostolopoulos D, Marinakis G, Ntantogian C, Xenakis C.
- Discovering authentication credentials in volatile memory of Android mobile devices. In: the 12th IFIP conference on ebusiness, e-services, e-society (I3E 2013), Athens, Greece; April 2013.
- Bornstein D. Dalvik VM internals. In: Google I/O developer conference; June 2008.
- Case A. Memory analysis of the Dalvik (Android) virtual machine. Seattle: SOURCE; Dec. 2011.
- Consumer Survey. Password habits; September 2012.
- Girault E. Volatilitux: physical memory analysis of Linux systems; Dec. 2010.
  Hoog A. Android forensics: investigation, analysis, and mobile security for Google Android. Syngress, Elsevier; June 2011.
- http://blog.flurry.com/bid/88867/iOS-and-Android-AdoptionExplodes-Internationally [accessed on May 2013].
- http://code.google.com/p/lime-forensics [retrieved on Nov. 2012].
- http://developer.Android.com/tools/debugging/ddms.html [accessed on Nov. 2012].
- http://developer.android.com/tools/help/adb.html [accessed on Nov. 2012].
- http://developer.android.com/about/dashboards/index.html [accessed on 01.05.13].
- http://mobworld.wordpress.com/2010/07/05/memorymanagement-in-Android/ [accessed on Nov. 2012].
- http://thomascannon.net/projects/android-reversing [accessed on Oct. 2013].
- http://www.forensicswiki.org/wiki/dd [accessed on Oct. 2013].
- http://www.sleuthkit.org/sleuthkit/index.php [accessed on Oct. 2013].
- IDC worldwide quarterly mobile phone tracker; May 2013.
- Karayianni S, Katos V, Georgiadis CK. A framework for password harvesting from volatile memory. Int J Electron Secur Digit Forensics 2012;4(2e3):154e63.
- Kramer S. http://c-skills.blogspot.com/2010/08/droid2.html [accessed on Oct. 2013].
- Müller T, Spreitzenbarth M. Frost forensic recovery of scrambled telephones. In: 11th International conference on applied cryptography and network security (ACNS 2013), Alberta, Canada; June 2013.
- Mylonas A, Kastania A, Gritzalis D. Delegate the smartphone user? Security awareness in the smartphone platforms. Comput Secur May 2013;34(1):47e66. Elsevier Science.
- Mylonas A, Theoharidou M, Gritzalis D. Assessing privacy risks in Android: a user-centric approach. In: Proceedings of the 1st
  international workshop on risk assessment and risk-driven testing (RISK-2013). Turkey: Springer; November 2013.
- Ponemon Institute LLC. The lost smartphone problem: benchmark study of U.S. organizations. Ponemon Institute research report, sponsored by McAfee; Oct. 2011.
- Sylve J, Case A, Marziale L, Richard GG. Acquisition and analysis of volatile memory from android device. Digit Investig Feb 2012;8(3e4):175e84. Elsevier
- .Thing V, Ng K-Y, Chang E-C. Live memory forensics of mobile phones. In: 10th Annual conference of digital forensic research workshop (DFRW); 2010. Wright S. The Symantec smartphone honey stick project. Symantec Corporation; Mar. 2012.