امنیت ابری سیار:

یک مدل دشمن برای امنیت مرورگر های بسیار سبک

چکیده

مرورگر های بسیار سبک بر روری دستگاه های تلفن همراه به طور فزاینده برای دسترسی به سرویس های ابری و آپلود مورد استفاده واقع شده است/ داده های ذخیره شده بر روی ابر، با توجه به قابلیت بارگذاری منابعِ سریعترِ آنها نمایش داده می شوند. این مرورگر ها از اقدامات بهره وری سمت سرویس گیرنده استفاده می کنند، مانند ذخیره سازیِ حافظه نهان بزرگتر و پلاگین های کمتر. با این حال، تاثیر بر امنیت داده ها از چنین اقداماتی در محدوده ی مورد مطالعه است. در این مقاله، ما یک مدل دشمن را برای بررسی امنیت مرورگر های سبک پیشنهاد می کنیم. با استفاده از این مدل دشمن ، ما آسیب پذیری منتشر نشده در چهار مرورگرِ سَبکِ محبوب نشان می دهند، یعنی: مرورگر UC، دلفین، مرورگر CM، و مرورگر های سهام سامسونگ، اجازه می دهند تا یک مهاجم به صورت غیر مجاز به داده های خصوصی کاربران دست یابد. در حالت دوم، شامل تاریخچه ی مرورگر، محتوای ایمیل، و جزئیات حساب بانکی می شود. به عنوان مثال، ما نیز نشان می دهیم که این امر ممکن است به جای تصاویرِ حافظه نهان در یکی از مرورگرها، که می تواند به منظور تسهیل فیشینگ و دیگر فعالیت های جعلی، مورد استفاده قرار گیرد. با شناساییِ نقص طراحی در این مرورگرها (به عنوان مثال ذخیره سازی فایل نادرست)، ما امیدواریم طراحان مرورگر های آینده می توانند از خطا های مشابه جلوگیری کنند.

کلید واژه ها: امنیت ابری سیار، امنیت مرورگر های بسیار سبک، مرورگر های UC، دلفین، مرورگر های CM، مرورگر های سهام سامسونگ

1. مقدمه

در سال های اخیر، ما شاهد یک تغییر سریع در رفتارهای مروری اینترنت از استفاده از کامپیوترهای شخصی (PC) به دستگاه های تلفن همراه هستیم، به ویژه در دسترسی به سرویس های ابری و ذخیره سازیِ داده ها در ابر [20،45،23]. به عبارت دیگر، مرور اینترنت به طور فزاینده ای در دستگاه های تلفن همراه انجام می شود[51]. این امر همچنین منجر به افزایش استفاده از مرورگرهای بسیار سبک در دستگاه های تلفن همراه می شود.

مرورگر های سبک برای قابلیتِ بارگذاری منابع سریع آنها محبوب هستند، به خصوص برای مشاهده ی فایل هایِ رسانه ای بزرگ و یا برای بازی. با این وجود، تجارت کردن به معنیِ کاهش دادنِ ویژگی های کاربر و مکانیزم های امنیتی ضعیف است[57،58]. به عنوان مثال، نیازمندی های امنیتی مرورگر اصلی توسط W3C مشخص می شود[46] اجرا در مرورگرهای معمولی، مانند گوگل کروم و موزیلا فایرفاکس، ممکن است روی مرورگر های سبک نصب نشوند[3].

مرورگرها برنامه های کاربردی حساس امنیتی هستند. به عنوان مثال، آنها قادر به دسترسی به اطلاعات شناسایی شخصی(PII) و داده های حساس مانند اطلاعات حساب بانکی هستند. ارتباطات مرورگر را در مراحل مختلف ارتباط می توان هدف قرار داد، به عنوان مثال بر روی دستگاه های سرویس گیرنده، در طول انتقال شبکه، و در سرویس دهنده ها. مسائل امنیتی و راهبرد های کاهشِ مربوط به شبکه و سرور منافع قابل توجهی به دست آورده اند(نگاه کنید به[14،15،52]).امنیت مرورگر ها در دستگاه های تلفن همراه، با این حال، منطقه مورد مطالعه به نظر می رسد. به عنوان مثال، این پرسش مطرح می شود که آیا حافظه نهان و فایل های دیگر به صورت امن توسط مرورگرها ذخیره می شوند یا نه! به طوری که آنها نتوانند توسط فرد ناخواسته یا برنامه ای در دسترس قرار گیرند، این مسئله به خوبی

مطالعه نشده است(به عنوان مثال آیا حافظه نهان و فایل های دیگر با اجازه ی فایل مناسب رمزنگاری و ذخیره شده اند؟).

در این مقاله، ما در تلاشیم تا امنیتِ اطلاعاتِ کاربرِ ذخیره شده توسط مرورگر بسیار سبک بر روی دستگاه های تلفن همراه را ارزیابی کنیم. با استفاده از مدل دشمن با اقتباس از متون امنیتی، ما چهار مرورگر سبک و محبوب را برای دستگاه اندروید بررسی کردیم و آسیب پذیری منتشر نشده را نشان دادیم. ما به این سهم دوگانه توجه می کنیم:

1) یک مدل دشمن برای مطالعه ی امنیت مرورگر تلفن همراه سبک طراحی شده است. و

2) شناسایی آسیب پذیری های منتشر نشده در چهار مرورگر سبک.

بقیه ی این مقاله به شرحی که در ادامه می خوانیم سازماندهی شده است. مواد پس زمینه و متون مرتبط به ترتیب در بخش های 2 و 3 توصیف شده است. در بخش 4، ما مدل دشمن پیشنهادی و برنامه نمونه را ارائه شده است. راه اندازی آزمایشی و یافته ها به ترتیب در بخش 5 و 6 مشخص شده است. در بخش آخر استراتژی های کاهش بالقوه و نتیجه گیری مقاله مطرح شده است.

2. سابقه و هدف: مرورگرهای موبایل

مرور یک صفحه ی وب نیاز به بارگذاریِ مجموعه های متعددی از منابع دارد، مانند HTML، CSS، JavaScript و فایل های رسانه ای. به عنوان مثال، مطابق با وانگ و همکارانش. [54]، بارگذاری این منابع با توجه به تفاوت های معماری و محدودیتهای محاسباتی می تواند بر روی دستگاه های تلفن همراه آهسته تر از رایانه های شخصی باشد. سرعت در طی مرور وب سایت یکی از نگرانی های اصلیِ کاربران است. به عنوان مثال، بنا به گزارش، یک ثانیه تاخیر در بارگذاری صفحه ی وب می تواند نتیجه را در دیدگاه های مربوط به صفحه ی وب 11٪ و رضایت مشتریان را 16٪ کاهش دهد[39]. مشاهدات مشابهی در مطالعات انجام شده توسط آمازون [33] و گوگل [10] نیز تکرار شد.

مرورگرهای سبک به منظور بهبود سرعتِ مرور، و به تبعِ آن بهبودِ کیفیت تجربه ی کاربران، در سمت سرویس گیرنده راه کار های مفیدی را به کار می گیرند. این مسئله موجبِ ایجاد یک حافظه ی نهان بزرگ تر برای ذخیره سازی و

اجتناب از هر گونه پلاگین است که می تواند باعثِ تاخیرِ بارگذاریِ منابع وب شود. حافظه ی نهان یک حافظه ی موقت برای ذخیره سازیِ منابع وب دانلود شده است. اگر یک کاربر تلاش کند تا به URL یا همان صفحه ی وبی که قبلا دیده، دسترسی پیدا کند، مرورگر چک می کند که آیا این صفحه در محتوای حافظه ی نهان وجود دارد یا نه! اگر محتوا یافت شود، مرورگر آن صفحه را از منابعِ حافظه ی نهان بارگذاری می کند، بنابراین، باعثِ صرفه جویی در زمان و منابع شبکه می شود.

مرورگر های محبوب، مانند گوگل کروم، موزیلا فایرفاکس و اپرا، از استاندارد های ذخیره سازیِ وب برای ذخیره ی داده ها در حافظه ی نهان بهره می برند. ذخیره سازی وب را می توان به عنوان بخشی از HTML5 معرفی کرد که توسط اتحادیه ی (کنسرسیوم) جهانی وب(W3C) استاندارد سازی شده است. ذخیره سازی وب شامل دو بخش عمده می شود: ذخیره سازی محلی و جلسه ی ذخیره سازی، که به ترتیب رفتار مشابهی مانند کوکی های ماندگار و جلسه ی کوکی ها از خودشان نشان می دهند. جلسه ی ذخیره سازی تا باز شدنِ صفحه ی وب، منابعِ وب را ذخیره می کند. در موردِ ذخیره سازی محلی، تا زمانی که مرورگر بسته است، حافظه ی نهان تولید شده بر روی دستگاه ها باقی می ماند[55].

در هر دو محیطِ کامپیوتر های شخصی و دستگاه های تلفن همراه، ذخیره سازی وب امنیت بیشتری نسبت به حافظه ی نهان مرورگر بومی دارد. با توجه به W3C، ذخیره سازی وب اگر به درستی اجرا شود، می تواند برای ذخیره ی اطلاعات حساس کاربر استفاده شود[55].در دستگاه های اندروید، ذخیره سازی وب معمولا از حافظه ی داخلی دستگاه استفاده می کند(مانند داده/داده/ نام بسته/ فهرست راهنما(دایرکتوری)). بنابراین، این موارد ذخیره شده در حافظه ی نهان نمی توانند به غیر از برنامه مالک توسط هیچ کاربر و برنامه ی دیگری مورد دسترسی قرار گیرند.

با این حال، ذخیره سازی وب توسط حجم حافظه ی نهان محدود می شود. برای استفاده ی عمومی، W3C توصیه می کند که از حجم ذخیره سازی 5مگابیت در هر وب سایت استفاده شود، اما این حجم می تواند در هنگام اجرا بر روی دستگاه های تلفن همراه کاهش یابد. مرورگر های سبک معمولا برای بهبود سرعت بارگذاریِ مرورگر بر روی

حافظه های نهان بزرگتر تکیه می کنند. بنابراین، این مرورگر ها اغلب مقدار زیادی از داده های حافظه ی نهانِ خارج از ذخیره سازی وب را در ذخیره سازیِ خارجی انبار می کنند(مانند کارتِ SD).

در دستگاه های اندروید، حافظه ی داخلی به طور کلی برای داده های برنامه، امنیت ذخیره سازیِ محلی بیشتری در نظر گرفته است، چرا که، به طور پیش فرض، داده های ذخیره شده تنها می توانند توسط برنامه ی خالق مورد دسترسی یا تغییر قرار گیرند. در مقایسه، هر گونه منابعِ ذخیره شده در ذخیره سازیِ خارجی توسط هر برنامه ای که اجازه ی READ_EXTERNAL_STORAGE را دارد، می توانند مورد دسترسی، تغییر یا حذف قرار گیرد[16].

جدول1- مرورگر های سبک.

S. no.	نام مرورگر ها	نسخه no.	ذخیره شده در گوگل پلی(Google play)(در میلیون; ملاحظات از سپتامبر 2015)	ملاحظات
.1	مرورگر UC	10.6.2	100-500	
.2	دلفین	11.4.19	50-100	
.3	مرورگر CM	5.20.06	10-50	
.4	مرورگر های سهام سامسونگ	N/A	N/A	پیش نصب شده با تلفن همراه سامسونگ، بنابراین تعداد کل کاربران و نسخه ی نرم افزار نمی تواند شناسایی شود.

جدول2 حافظه ی نهان مورد هدف و مکان ذخیره سازیِ فایل هایِ مرورگر ها در این تحقیق

مرورگر ها	مکان حافظه ی نهان مورد هدف	محتویات
دلفین	/sdcard/TunnyBrowser/cache/speeddial_covers /sdcard/TunnyBrowser/cache/tablist_cache /sdcard/TunnyBrowser/cache/webViewCache	-URL های ذخیره شده به منظور دسترسی سریع -تصاویر برداریِ فایل هایِ تصویری -تمام فایل های حافظه ی نهان(HTML, CSS, JavaScript, media)

مرورگر UC	//sdcard/UCDownloads/cache
	//sdcard/UCDownloads/config
	/sdcard/UCDownloads/offline/
		-تمام فایل های حافظه ی نهان
		-TrafficStatus.db ; شامل زمان بندی و پاسخ ارتباط بین سرویس گیرنده و سرویس دهنده است.
		-ApplicationCache.db ; شامل داده برای مدیریت بارگذاریِ حافظه ی نهان است.

مرورگر CM	/sdcard/CheetahBrowser/.data/	تاریخچه ی URL مرورگرها

مرورگر های سهام سامسونگ	/data/data/com.sec.android.app.sbrowser/files/	تصاویر برداریِ فایل هایِ تصویری

3. کار مربوطه

امنیت وب(نرم افزار) دارای تحقیقات متمرکزی در چندین سال بوده است[40]. مرورگرهای رایانه های شخصی، لپ تاپ ها و دستگاه های تلفن همراه، قوانین اساسی برای بارگذاری صفحات وب و ارتباط با سرورها را به اشتراک می گذارند. بنابراین، متون موجود در مورد امنیت مرورگر تمایل به تمرکز بر روی مرورگر های "سنتی" دارد (برای رایانه های شخصی و لپ تاپ ها)، و همچنین تمرکز بر روی امنیت شبکه های دیگر و یا بر روی تشخیص وب سایت های مخرب است(نگاه کنید به [18،19،21،24،50]).

در سال 2014، Wadka، Misha و Dixit یک "سیستم پاسخ" را ارائه دادند که روشِ کار آن نظارت بر روی جلوگیری از نشت اطلاعات از مرورگر است[53]. "سیستم پاسخ" یک رابط بین برنامه ی مرورگر و هسته ی سیستم عامل(لینوکس) است، که در طول اجرایِ روندِ مرورگر استناد می کند. محققان یک لایه ی میانی بین هسته و لایه ی برنامه پیشنهاد کردند که "سیستم پاسخ" را کنترل می کند و اطلاعات شخصی که در طول مرور به بیرون درز پیدا می کند را فیلتر می کند.

Virvilis و همکارانش [52] تاثیراتِ روش فیلترینگِ لیست سیاه در مرورگر ها را ارزیابی کردند که برای جلوگیریِ کاربران از دیدن صفحه های وب مخرب طراحی شده است. در تحقیق مرتبطی دیگر، Amrutkar و همکارانش [2] یک مدل تهدید را ارائه کردند، که این مدل امکان کشف ضعف های معماری بر روی دستگاه های تلفن همراه و مرورگرها را امکان پذیر می کند. محققین مسیر های حمله، مانند نمایش بالونی، جعل درخواست میان وبگاهی(CSRF) و سرقت از طریق کلیک کردن(ClickJacking) را نشان دادند، که می توانند برای فیشینگ یا به طور مستقیم برای سرقت اطلاعات از دستگاه های کاربران استفاده شوند(جدول2).

اخیرا در سال 2015، Amrutkar، Traynor و van Oorschot [3] شاخص های امنیتیِ(بر اساس دستورالعمل امنیتیِ W3C-[46]) مورد استفاده در مرورگر های محبوب تلفن همراه را ارزیابی می کنند. برای مثال، آنها این مسئله

را بررسی می کنند که آیا مرورگر، هویت صاحب سایت و صادرکننده گواهی نامه را تعیین می کنند یا نه! و همچنین اینکه آیا مرورگر از فیلترِ URL ضد فیشینگ استفاده می کند یا نه! خلاصه ای از این تحقیق برای گوگل کروم و موزیلا فایرفاکس در جدول 4 ارائه شده است.

مطالعات قبلی، حافظه ی نهان را به عنوان یک مسیر بالقوه به سازش حریم خصوصی کاربر بر روی ارتباطات مرورگر شناخته است[47،6،8]. به عنوان مثال، Bernstein [7]نشان داد که یک مهاجم می تواند رفتارهای مرورِ وبِ کاربر را توسط محاسبه ی زمان بارگذاریِ محتوا شناسایی کند. از آن زمان، چندین راه حل برای کاهش چنین حمله هایی پیشنهاد شده است[28،30،37].

مطالعات موجود به طور کلی بر روی حفاظت از اطلاعات کاربران در حالی که آنها داده های خود را در طول شبکه منتقل می کنند، متمرکز است. با توجه به محبوبیت دستگاه های تلفن همراه، مرورگر ها به طور فزاینده ای مورد هدفِ مجرمان سایبری هستند که به دنبال بهره برداری از مسائل امنیتی ذاتی، مانند بهره برداری از مجوز کاربر و مجوز فایل هستند. به عنوان مثال، Hay [22] نشان داد که چگونه نقص های امنیتی مرورگرِ Opera میتواند توسط مهاجم برای دزدیدن اطلاعات شخصی کاربر با بهره برداری از مجوز فایل از حافظه ی نهانِ مرورگرِ Opera مورد سوء استفاده قرار بگیرد.

Jia و همکارانش [26]قدرت امنیتیِ حافظه ی نهان مرورگرِ 5 کامپیوتر و 15 تلفن همراه را مورد ارزیابی قرار دادند تا تعیین کنند که آنها در معرضِ خطرِ مسمومیتِ حافظه ی نهان مرورگر(BCP) در یک مرد در وسط حمله(MITM) هستند یا نه! محققین نتیجه گرفتند که هر 5 مرورگر کامپیوتر و اغلب مرورگر های تلفن همراهِ مورد بررسی در این تحقیق در معرضِ چنین خطری هستند. در کار مربوطه، Jia و همکارانش[27] یک روش برای شناساییِ موقعیت جغرافیایی کاربر تلفن همراه(مانند کشور، شهرستان و محله) توسط خرناس در حافظه ی نهان مرورگر و اندازه گیری زمان مورد نیازِ حافظه ی نهان مرورگر ارائه کردند. یک رویکرد مشابه توسط Liang و همکارانش[32] ارائه شده است، که نشان دهنده ی تاریخچه ی مرورِ کاربر است که می تواند با استفاده از زمانی که حمله بیش از حافظه ی مرورگر است ربوده شود.

جدول3 - خلاصه از یافته ها.

مرورگر ها	اهداف دشمن			
	تاریخچه مرورگر	شرایط جستجو	عصاره ی(دانش) محتوای وب	تغییر محتوای حافظه ی نهان
1.مرورگر دلفین	بله	بله	بله	بله
2.مرورگر CM	بله			
3.مرورگر UC	بله	بله	بله	
مرورگر های سهام سامسونگ	بله		بله	

جدول4 - خلاصه ی مقایسه ای از پیاده سازیِ شاخص های امنیتی در مرورگر های تلفن همراه.

سلسله ی مرورگر[3]			
مرورگر ها	فیلتر ضد فیشینگ	نمایش هویت ارائه دهندگان	نمایش گواهی نامه
گوگل کروم	بله	بله	بله
موزیلا فایرفاکس	بله	بله	بله
مرورگر های سبک(نتیجه ی ارزیابی ما)			
دلفین	نه	بله	بله
مرورگر UC	نه	نه	نه
مرورگر های سهام سامسونگ	نه	بله	بله
مرورگر CM	بله	بله	بله

امنیت ذخیره سازی در اندروید نیز اخیرا مورد توجه محققین بوده. در سال 2015، برای مثال، Liu و همکارانش [34] بر روی رفتار های ذخیره سازی داده های برنامه ی اندروید مطالعه کرده اند. آنها داده های ذخیره شده توسط برنامه های ارتباطیِ محبوب را مورد بررسی قرار دادند(مانند ویبو، فیسبوک، اینستاگرام، لاین، اسکایپ و وایبر) برای اینکه تعیین کنند چه مقدار از داده های خصوصی کاربران از این برنامه ها بدون آگاهیِ کاربران توسط مهاجم قابل دزدین است. این مطالعه همچنین نشان می دهد که امنیت اطلاعات خصوصی کاربران وابسته به این است که آیا طراحِ برنامه این اطلاعات را حساس یا غیر حساس تعیین کرده است. داده هایی توسط طراحِ برنامه غیر حساس در نظر گرفته خواهد شد که به منظور اشتراک در حافظه ی ذخیره سازیِ خارجی ذخیره شده باشد، که در دسترس دیگر کاربران هم قرار می گیرد. با این حال، هیچ تعریف یکسانی برای داده های حساس و غیر حساس وجود ندارد. این مسئله نشان

دهنده ی این است که در بعضی موارد، شماره تلفن، لیست تماس و دیگر اطلاعات خصوصیِ کاربر به دست آمده از فایل های عمومی به اشتراک گذاشته شده توسط کاربر است.

مطالعات انجام شده توسط Zhang و همکارانش [59] نیز بقایای به جا مانده از داده توسط برنامه های اندروید پس از حذف آنها را گزارش می دهد که می تواند اطلاعات حساسی را به مهاجم نشان دهد. آنها در آزمایششان، بقایای بهبود داده از سرویس های سیستم و سرویس برنامه ی سیستم را مورد بررسی قرار دادند. آنها قادر به بازیابیِ اطلاعات حساسی مانند اعتبار ورودی کاربران، کلید های عمومی/خصوصی، URL و محتوای قصد انتظار شدند. یافته های مشابهی از تجزیه و تحلیل برنامه های تلفن همراه نیز گزارش شده است[48،41،36،35،44،43،42،31،13،5].

Liu و همکارانش [34] نشان دادند که چگونه یک مهاجم با قابلیت نصب یک برنامه با مجوزِ WRITE_EXTERNAL_STORAGE, READ_EXTERNAL_STORAGE and INTERNET می تواند دسترسی غیر مجاز به فایل های ذخیره شده ی عمومی در دستگاه های اندروید پیدا کند. مشابه به روش Liu و همکارانش [34]، Zhua و همکارانش [60] از یک مدل دشمن با قابلیت های یک مهاجم برای دزدیدن و ارسال داده از طریق مرورگر با استفاده از قصدِ URL ACTION-VIEW استفاده کردند. اما، این روش(مدل دشمنِ مجوز صفر) تنها زمانی می تواند داده را انتقال دهد که صفحه نمایش دستگاه در حال اجرا باشد. استفاده از مدل دشمن در امنیت برنامه ی تلفن همراه نیز در این مطالعه یافت می شود که توسط Martini و Choo [16] انجام شده است.

4. مدل دشمنِ پیشنهادی ما و یک نمونه ی اولیه ی نرم افزار

4.1. مدل دشمن

در یک کار اخیرِ انجام شده، Martini و Choo [17] اولین مدل دشمن برای خروج داده های پنهان اندروید را ارائه کردند. در این مدل، دشمن دارای قابلیت رهگیری، تزریق، ویرایش، حذف، رمزگذاری، رمزگشایی، انتقال، و گوش دادن به ارتباطات کاربر است. در یک کار مرتبط D’Orazio و Choo [12] یک مدل دشمن ارائه کردند که این مدل قطاری از قابلیت های دنیای واقعیِ را دارد، از جمله: مدیریت حقوق دیجیتال (DRM) مهاجم برای دستگاه های تلفن

همراه. در این مقاله، ما یک مدل ضعیفتر (و احتمالا، واقعی تر) از آنچه Martini و Choo [17] و D'Orazio و Choo [12] انجام داده اند، ارائه کردیم، و نشان دادیم که دشمن به عنوان چنین مدلی نیز می تواند برای کشف و بهره برداری از آسیب پذیری در مرورگر های سبک برای دستگاه های آندروید استفاده شود.

ما یک مرورگر موبایل را در صورتی نا امن در نظر می گیریم که هر کدام از این اهداف برآورده شود.

هدف 1: دشمن تاریخچه ی URL مرورگر را می آموزد.

هدف 2: دشمن عبارات جستجوی کاربر را می آموزد.

هدف 3: دشمن محتوای صفحه وب(مانند محتوای ایمیل کاربر،اطلاعات حساب بانکی) را می آموزد.

هدف 4: دشمن می تواند محتوای حافظه ی نهان(مانند فایل تصویری) را تغییر دهد.

4.2. نمونه ی اولیه ی نرم افزار

برای نشان دادن ابزار های مدل دشمنمان، ما یک نمونه ی اولیه از برنامه ی اندروید را آماده کرده ایم که قادر به خواندن و نوشتن فایل ها در حافظه ی به اشتراک گذاری شده ی خارجیِ دستگاه است(از طریق مجوز های READ_ EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE). ما با ملاحظه ی اینکه اگر به این برنامه دسترسی به مجوزِ WRITE_EXTERNAL_STORAGE اعطا شود، مجوزِ READ_ EXTERNAL_STORAGE به طور خودکار اعطا شده است. به منظور انتقال داده از دستگاه کاربر، برنامه نیازمند مجوزِ اینترنت خواهد بود. با این حال، برنامه های اندروید نیازی به اطلاع و یا درخواست برای تصویب کاربر برای این مجوز ندارند، زیرا این مجوز اگر در فایل بیانیه اظهار شود، به طور پیش فرض به تمام برنامه ها اعطا می شود. مجوز دیگری که برنامه ی ما نیاز دارد ACCESS_NETWORK_STATE است، که به دشمن اجازه می دهد تا نوع شبکه ای که دستگاه به آن وصل است را بشناسد(مانند WiFi و یا 3G/4G). به عبارت دیگر برنامه ی ما نیازمند تاییدِ کاربر برای مجوز WRITE_EXTERNAL_STORAGE و ACCESS_NETWORK_STATE است.

یک چالش کلیدی برای هر برنامه ی مخرب (از جمله برنامه ما) جلوگیری از تشخیص توسط کاربران و فیلترهای نرم افزارهای مخرب است. الگوریتم های فیلتر مخرب به طور کلی بر اساس ارزیابی مجوز کاربران [1،4،49] و وضعِ فعالیت است. این الگوریتم ها یک برنامه را در صورتی مشکوک در نظر می گیرند که بیش از حد مجوز هایی یا مجوز های مرتبط با فعالیت تحمیل هزینه به دست آورد، مانند ارسال SMS یا MMS یا تماس گرفتن(به عنوان مثال، به شماره حق بیمه) [25،9،61]. بنابراین، برنامه ی ما برای استفاده ی تعداد کم، به طور مداوم و بدون هزینه ی تحمیل مجوز مربوطه برای جلوگیری از تشخیص طراحی شده است.

زه کشی بیش از حد از باتری و یا مصرفِ زیاد پهنای باند شبکه نیز منجر به تشخیص و یا بالا بردن سوء ظن کاربر می شود. بنابراین، به منظورِ جلوگیری از استفاده ی بیش از حد باتری، برنامه ی ما برای استفاده ی اندروید ساخته شده در تنها توابع شنونده طراحی شده است. به طور مشابه برای جلوگیری از مصرفِ زیاد پهنای باند، برنامه ی ما اطلاعات کاربر را تنها زمانی که دستگاه به شبکه ی WiFi متصل است، به یک سرور شخص ثالث که تحت کنترل دشمن است آپلود می کند. اگر این کاربر در حال مرور صفحات وب با استفاده از 3G/4G باشد، برنامه ی ما داده های حافظه ی نهان مرورگر را بر روی پوشه ی خصوصی برنامه کپی می کند و منتظر می ماند تا دستگاه به WiFi متصل شود. به عبارت دیگر این برنامه در WiFi و شبکه ی 4 G اجرا می شود.

4.2.1. فعالیت 1: تعیینِ زمانی که مرورگر شروع به اجرا می شود.

اندروید روش های متعددی را به منظورِ بازگشتِ برنامه های در حال اجرا فراهم می کند. و یک روش موثر برای ایجاد یک گیرنده ی پخش برای برنامه ی مرورگرِ مورد هدف استفاده از ActivityManager.getRunningAppProcess() است. با این حال، این ویژگی در اندرویدِ API سطحِ 21 قدیمی شده است. بنابراین، ما از ویژگیِ fileObserver بومی در اندروید استفاده خواهیم کرد. زمانی که در فایل مقصد یا دایرکتوری تغییراتی رخ دهد، به اطلاع برنامه می رساند. FileObserver از زیر سیستمِ inotify از هسته

ی لینوکس استفاده می کند، که fileSystems را برای حریق اطلاعات زمانی که یک دایرکتوریِ داده شده توسط این مرورگر در دسترس قرار گیرد، گسترش می دهد.

4.2.2. فعالیت2: کپی کردن از فایل های حافظه ی نهان

از آنجا که در اندروید فرمانِ "Cp" وجود ندارد، ما از InputStream و OutputStream بومی برای کپی کردن فایل ها از دایرکتوریِ حافظه ی نهان مرورگر استفاده می کنیم(در الگوریتم2) سپس ما فایل هایی را که با استفاده از الگوریتم 1 کپی شده اند، شناسایی می کنیم. فایل های کپی شده در دایرکتوریِ حافظه ی نهانِ برنامه ی اولیه(در حافظه ی داخلی) به منظورِ جلوگیری از بالا بردن سوء ظن کاربر ذخیره خواهند شد. برای به حداقل رساندن استفاده از منابع، ما دایرکتوریِ حافظه ی نهانِ برنامه ی اولیه را به 5MB محدود خواهیم کرد و سیستم به طور خودکار وقتی که این حد پر شود، فایل های قدیمی تر را پاک خواهد کرد.

الگوریتم1. لیست کردنِ فایل های حافظه ی نهان.

```
File applicationCache=this.getCacherDir();
File browserCache=new File(Environment.getExternalStorage()get.absolutebath() +
"/cacheDirectoryName");
String[] filename = browserCacheList();
For (int i=0; i<browserCacheListFiles().length; i++){
copyFiles(new file(browserCache, filename[i]),
new File(applicationCache, filename[i]));
}
```

الگوریتم2. کپی کردنِ فایل های حافظه ی نهان.

```
copyFiles(File source, File dest){
InputStream toCopyFile = new FileInputStream(source);
OutputStream copiedFile = new FileOutputStream
(dest);
Byte[] buff = new byte[1024];
Int leng;
While ((leng = toCopyFile.read(buff)) > 0){
}
toCopyFile.close();
copiedFile.close();
}
```

به منظورِ بهبودِ بهره وری از برنامه ی نمونه ی اولیه، ما می توانیم توسط اجرای این دستور "این فایل از قبل موجود است" از کپی شدنِ همان فایل بیش از یک بار جلوگیری کنیم.

4.2.3. فعالیت3: بررسی کردن اتصال شبکه و انتقال فایل

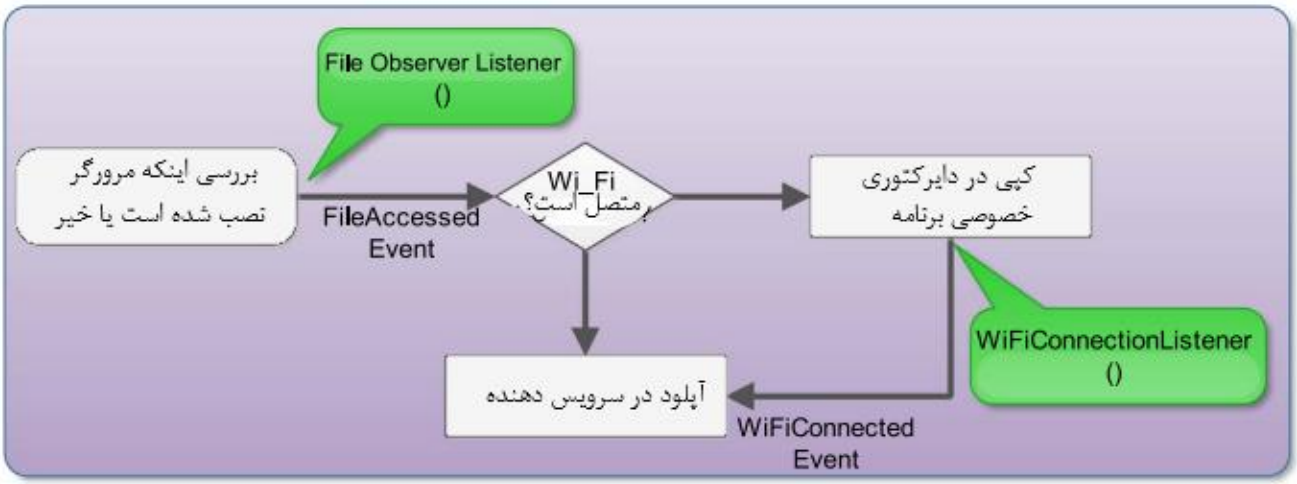
آپلود مکرر فایل های حافظه ی نهان از دستگاه های سرویس گیرنده می تواند سوء ظن را با توجه به مصرف پهنای باند بالا ببرد(مانند زمانی که کاربر یک برنامه ی داده محدود دارد). بنابراین، برنامه ی نمونه ی اولیه ی ما برای بررسی اتصالِ WiFi طراحی شده است(در الگوریتم 3).

الگوریتم3. بررسی اتصالِ WiFi

```
ConnectivityManager manager = (ConnectivityManager) get-
SystemService (CONNECTIVITY_SERVICE);
NetworkInfo wifiConn = connManager.getNetworkInf
(ConnectivityManager.TYPE_WIFI);
return wifiConn.isAvailable();
```

اگر یک اتصالِ WiFi شناسایی شود، داده های ذخیره شده در حافظه ی خصوصیِ برنامه ی ما بر روی سرور تعیین شده، آپلود خواهد شد.

آپلودِ فایل ها بر روی سرور می تواند بر اساس اتصالِ به WiFi، یک فرآیند مصرف منابع و طولانی باشد. برنامه ی ما به عنوان یک سرویس در پس زمینه اجرا می شود، ما می توانیم برای آپلود فایل ها از "asyncTask" استفاده کنیم. حلقه ی این برنامه تا زمانی که آپلود کامل شود، چندین مرتبه اجرا خواهد شد- در شکل1. سپس فایل ها از حافظه ی خصوصی برنامه پاک خواهند شد.



شکل1. آپلود فایل ها بر روی سرورِ تعیین شده

4.2.4. فعالیت 4: درج تصویر در حافظه ی نهان (مسمومیت حافظه ی نهان)

با استفاده از مجوزِ WRITE_EXTERNAL، برنامه ی ما قادر به درج تصویر در دایرکتوریِ حافظه ی نهانِ مرورگر است- در الگوریتم 4 و 5. بارگذاریِ تصویر درج شده توسطِ مرورگر به تعدادی از شرایط بستگی دارد، مانند زمان انقضا، نوع فایل حافظه ی نهان و فرکانس دسترسی کاربر به URL مربوطه.

الگوریتم 4. تعیین اینکه آیا فایل حافظه نهان حاویِ یک تصویر است یا نه!

```
BitmapFactory.Options imageOptions = new BitmapFactory.
Option ();
imageOptions.inJustDecodeBounds = true;
BitmapFactory.decodeFile (file. getPath, imageOptions);
If (options.outWidth != -1 && options.outHeight!= -1)
return true;
```

الگوریتم 5. جایگزینی فایل تصویر

```
Byte [] buff = new byte [1024];
Int leng;
While (( leng = new FileInputStream(toBeCopied).read
(buff)) > 0) {
toBeReplaced.write(buff, 0, leng);
}
toBeCopied.close();
toBeReplaced.close();
```

5. راه اندازی آزمایش

بنا به آمارِ گوگل پلی استور، مرورگر هایِ گوگل کروم، موزیلا فایرفاکس و مرورگر های UC محبوب ترین مرورگر ها در برنامه های اندروید هستند. کروم و فایرفاکس دارای سازگاریِ مرور کراس-پلتفرم و اجرای اقدامات امنیتی مختلف هستند. با این حال، مرورگر های سبک بین کاربرانی که سرعتِ بارگذاریِ منابعِ آنها به طور قابل ملاحظه ای کمتر است، محبوب هستند [11]، مخصوصا بارگذاریِ فایل های رسانه ای بزرگ و در محیط بازی.

بنابراین، در این تحقیق، ما سه مرورگر سبک اندروید را انتخاب کرده ایم: دلفین، مرورگر های CM و مرورگر های UC(از سپتامبر سال 2015، بر اساس دانلود های گوگل پلی استور). از آنجایی که سامسونگ یک مدل محبوب اندروید است، ما مرورگر سهام سامسونگ را نیز قرار دادیم (نام بسته: com.sec.android. app.sbrowser) که به طور پیش فرض بر روری سامسونگِ Galaxy S4 با نسخه ی اندرویدِ 4.4.2 نصب شده است.

جدول 1 اطلاعاتی در موردِ سه مرورگرِ انتخاب شده به ما نشان می دهد. ما در حال حاضر به طور خلاصه توضیحی در موردِ وضعیتِ ذخیره سازیِ حافظه ی نهانِ مرورگر های انتخاب شده ارائه می دهیم.

مرورگر UC یکی از محبوب ترین مرورگر های اندروید است که بسش از 100 میلیون بار دانلود شده است. پس از بازرسی اولیه، ما کشف کردیم که مرورگرِ UC، بخشِ بزرگی از فایل های حافظه ی نهان را در حافظه ی به اشتراک گذاری شده ی خارجی ذخیره می کند، که شاملِ فایل های HTML از صفحه های وبِ دیده شده، JavaScript، Css، فایل های تصویری و طیف وسیعی از پایگاه داده های پر شده توسطِ فعالیت مرور کاربران است. این پایگاه داده ها تنها اطلاعات مربوط به صفحه های وبِ مرور شده توسط کاربران را نشان نمی دهند، بلکه تعداد دفعاتی که دیده شده اند و همچنین زمان دسترسی و اینکه هر بار چه پاسخی از سرویس دهنده دریافت کرده اند را نیز نشان می دهند. همچنین این مسئله که چه مقداد داده(پهنای باند) را هنگام دسترسی به هر صفحه به طور جداگانه استفاده کرده اند را نیز نشان می دهند.

مرورگر دلفین یکی از قدیمی ترین مرورگر های تعریف شده برای اندروید است. بازرسی اولیه ی حافظه ی نهان آن نشان می دهد که مرورگر دلفین اکثر منابع حافظه ی نهان را در حافظه ی به اشتراک گذاری شده ی خارجی در /sdcard/TunnyBrowser/cache/ directory ذخیره می کند. این مرورگر سه زیر-دایرکتوری دارد: speeddial_covers، tablist_cache و webviewCache. همانطور که از نام آن مشخص است، Speed dial(دسترسیِ سریع) به کاربر اجازه می دهد تا URL های مورد علاقه و URL هایی که بیش تر استفاده می کند را بر روی عکس های ریزِ(thumbnails) جلوی مرورگر ذخیره کند. دایرکتوریِ Speeddial_covers، URL و فایل های مربوط به آن وب سایت را ذخیره می کند.

ما مشخص کردیم که دلفین، تصاویرِ محتوای وبِ نمایش داده شده را دخیره کرده است. به عنوان مثال، هنگامی که یک کاربر به طور هم زمان بیش از یک صفحه را مرور می کند، محتوای آن صفحه به عنوان یک تصویر قابل مشاهده خواهد بود، که حجم آن در حدود 50KB خواهد بود و در دایرکتوریِ tablist_cache ذخیره خواهد شد. تمامِ دیگر

فایل های حافظه ی نهان، که شاملِ HTML، Css و فایل های رسانه ای می شود در دایرکتوریِ webViewCache ذخیره می شود.

بازرسی اولیه نشان داده که مرورگر CM، اکثر فایل های حافظه ی نهان را در حافظه ی داخلی ذخیره می کند، و ما مشخص کردیم که ذخیره سازیِ محلی در دایرکتوریِ app_webview اجرا می شود، مشابهِ ذخیره سازیِ حافظه ی نهانِ استانداردِ ارائه شده در HTML5. با این وجود، URL های دیده شده در فایلی در حافظه ی به اشتراک گذاری شده، ذخیره و لیست می شوند.

در موردِ مرورگر سهام سامسونگ، فایل های حافظه ی نهان، شاملِ HTML، javaScript و فایل های رسانه ای می شود که در حافظه ی داخلی ذخیره شده اند. مشابهِ مرورگرِ دافین، همچنین هر زمان که محتوای جدید در مرورگر بارگذاری شود، تصاویر را به عنوانِ فایلِ bitmap ذخیره می کند. این bitmap زمانی که کاربر به کلید سوئیچ برنامه ی سریع در دستگاه دسترسی دارد یا زمانی که کاربر اقدام به تغییرِ صفحه در مرورگر می کند، ممکن است برای نمایشِ پیش نمایش هایِ برنامه های کاربردیِ در حال اجرا استفاده شود.

به طور پیش فرض، فایل های ذخیره شده در حافظه ی داخلی، خصوصی و تنها قابل دسترسی توسط صاحب برنامه در نظر گرفته می شوند. با این وجود، این مجوز می تواند در هنگام ایجاد یک فایل با استفاده ازپرچم هایِ MODE_APPEND، MODE_WORLD_READABLE و یا MODE_WORLD_WRITABLE توسط خالقِ برنامه تغییر کند. این مسئله به عنوانِ "نقص امنیتی جدی" برای اندروید در نظر گرفته می شود و از اندرویدِ API در سطحِ 17 حذف شده است. با این حال، یک نقص در مرورگر سهام سامسونگ یافت شده است که توسط این تحقیق ما نشان داده می شود. به طور خاص، ما مشخص کردیم که تصاویر ذخیره شده توسط مرورگر سهام سامسونگ در حافظه ی نهانِ آن با سطحِ دسترسیِ 644 (rw-r–r–) مجوز کاربر اختصاص داده شده است.، بدین معنی است که می تواند توسط هر برنامه ی دیگری مورد دسترسی قرار بگیرد.

در این تحقیق، ما از سامسونگِ Galaxy S4(بدون ریشه) در حال اجرا برای اندروید با نسخه ی 4.4.2 به عنوان دستگاهِ اصلیِ آزمایش استفاده کردیم. ما سپس برنامه ی نمونه(در بخش بعدی) را بر روی دستگاه نصب کردیم، که

در فرایند پس زمینه اجرا می شود. ما همچنین هر مرورگر را برای 5 دقیقه در URL های باز شده در چند صفحه اجرا کردیم. به منظور بررسی اثرِ مدل دشمنِ ما، ما همین آزمایش را بر روی سامسونگِ Galaxy S5(بدون ریشه root -) با اندرویدِ نسخه ی .5.0 نیز اجرا کردیم.

6. یافته ها

در این بخش ما یافته های تحقیقمان را توضیح می دهیم.

6.1. دلفین

ما موفق به کپی کردنِ تمام فایل ها در دایرکتوریِ ریشه ی حافظه ی نهان (/sdcard/TunnyBrowser/Cache) و در زیر شاخه های آن شده ایم. همچنین فایل ها با موفقیت در سرویس گیرنده ی ما آپلود شده اند.

در بازرسیِ فایل های آپلود شده، با مشاهده ی محتوای(شاملِ فایل های رسانه ای می شود.) صفحه های وب دیده شده و غیره، ما قادر به بازیابیِ تاریخچه ی URL مرورگر شدیم. ما همچنین با دسترسی به لینک در فایلِ HTML قادر به بازیابیِ محتوای خصوصیِ صفحه وب شدیم، مانند تصاویر و پست های دیوارِ صفحه ی فیسبوکِ کاربر. از تصاویر آپلود شده، ما توانستیم ایمیل ها، جزئیات حساب کاربری و دیگر اطلاعات حساس کاربر را به دست آوریم. ما همچنین مشخص کردیم که دلفین، تصاویرِ حافظه ی نهان را بدون تغییرِ هدر ذخیره می کند. به طور کلی، زمانی که کاربر تلاش می کند تا به همان URL دسترسی پیدا کند، مرورگر توسط حافظه ی نهانِ سرویس دهنده بررسی می شود که مشخص شود آیا حافظه ی نهانِ ذخیره شده تغییر یافته است یا نه! با این وجود، دلفین قادر به شناساییِ حافظه ی نهانِ تغییر یافته در تحقیق ما نیست. به طور خاص، ما صفحه ی اصلیِ آمازون(https://www. amazon.com/) را مرور کردیم و مشخص کردیم که تمام تصاویر محصولات در حافظه ی نهان ذخیره می شوند. پس از آن ما تعدادی از این تصاویر را با تصاویر دیگر با همان نامِ فایل جایگزین کردیم. ما بلافاصله همان صفحه را دوباره مرور کردیم و آن را با تصاویر جایگزین شده(به جای تصاویر اصلی) ارائه کردیم. ما همچنین به این مسئله که زمان بین مرور اول و دوم

کمتر از 2 دقیقه بود توجه کردیم. یافته های ما برای 2 دستگاهِ دیگر نیز صادق بود- سامسونگِ Galaxy S4 و سامسونگِ Galaxy S5.

6.2. مرورگرِ UC

مشابهِ یافته های توضیح داده شده در بخشِ 1.6، ما موفق به کپی کردن و آپلود کردنِ داده های ذخیره شده در دایرکتوریِ ریشه ی حافظه ی نهان شده ایم(/sdcard/UCDownloads). ما قادر به شناسایی تصاویر نیستیم، اگرچه بر اساس اطلاعات جمع آوری شده از دیگر فایل های حافظه ی نهان و پایگاه داده، ما در دوره ی نقاهت تاریخچه ی مرورگرِ وب و شناسایی وضعیت مرور کاربر موفق بوده ایم. برای مثال، ما قادر به تعیین این هستیم که صفحه ی وب در چه زمانی و توسط چند نفر دیده شده است. همچنین قادر به بازیابیِ محتوای صفحه ی وبِ دیده شده هستیم. ما نتایجِ مشابهی برای هر دو دستگاه به دست آوردیم.

6.3. مرورگر سهام سامسونگ

همانطور که قبلا در بخش 5 توضیح داده شد، مرورگر سهام سامسونگ تمام منابعِ حافظه ی نهان را در حافظه ی داخلی دستگاه ذخیره می کند. با این وجود، در سامسونگِ Galaxy S4، ما قادر به کپی کردن و آپلود کردن تصاویر از دستگاه شده ایم، یکبار هم محتوای جدیدی در مرورگر بارگذاری شد. از طریق این تصاویر، ما قادر به شناساییِ فعالیت های مرور کاربر به خوبیِ محتوای سایت های دیده شده، هستیم. ما همچنین قادر به بازیابی اطلاعات دیگری مانندِ پیام های ایمیل و جزئیات حساب کاربری هستیم. محتوای ذخیره شده با استفاده از مجوزِ دسترسیِ 644، ما را از هر گونه اصلاح باز میدارد.

ما دریافتیم که این مرورگر بر روی سامسونگِ galaxy S5 با اندرویدِ نسخه ی 5.0 به صورت پیش فرض نصب شده است. با این وجود، بازرسی ما از دیگر موبایل ها و تبلت های سامسونگ نشان دهنده ی این است که این مرورگر بر روی دستگاه های اندروید با نسخه ی 4.4.2 نصب شده است.

6.4. مرورگرِ CM

با کمال تعجب، مرورگر CM تمام فایل های حافظه ی نهان، از جمله تصاویر برای جهت یابیِ چند صفحه ای، پایگاه داده، فایل های HTML و فایل های رسانه ای را بر روی حافظه ی داخلی دستگاه در حالت خصوصی ذخیره می کند. تنها اطلاعاتی که ما می توانیم بازیابی کنیم، لیستِ URL های دیده شده توسط کاربر است که به عنوان یک فایل در /sdcard/CheeahBrowser/.data/ موجود است. مشابهِ یافته های گزارش شده در موردِ دو دستگاه دیگر.

6.5. خلاصه

جدول 3 یافته های این تحقیق در مورد 4 مرورگر را به طور مختصر بیان کرده است.

جدول 4 پیاده سازیِ امنیتیِ(مورد استفاده در تحقیقِ [3]) مرورگر های سبکِ مورد مطالعه در این مقاله، و همچنین ملتزمینِ دو مرورگر محبوب دیگر را به طور مختصر بیان کرده است. این مسئله روشن که است که دلفین، مرورگرِ UC و مرورگر های سهام سامسونگ فاقد مکانیزم های امنیتی اساسی هستند.

7. بحث و بررسی

تعدادی از برنامه های تجاری به منظورِ پیگیری و نظارت بر فعالیت کاربران بر روی دستگاه های اندروید طراحی شده اند، مانند FlexySpy و MobileSpy. این برنامه ها به کاربر اجازه می دهند تا تاریخچه ی مرورگر را بخواند و بر روی دستگاه های بدون ریشه علامت گذاری کند، اما آنها نیاز به مالک دستگاه تلفن همراه و اجازه دسترسی به برنامه را دارند، مانند مجوز مرورگرِ .com.android" . "READ_HISTORY_BOOKMARKS" برای دسترسی به تاریخچه ی مرورگر و "android.permission.ACCESS_FINE_LOCATION " یا " android.permission.ACCESS_COARSE_LOCATION" برای ردیابی محل کاربر استفاده می شوند. این برنامه قادر به گرفتن تصاویری از فعالیت های مرورگر و یا دسترسی به محتوا نیستند(ایمیل و جزئیات حساب شخصی) مگر اینکه دستگاه روت شده باشد و ه آن مجوزهای خاص اضافی اعطا شده باشد.

ما نشان دادیم که برنامه ی نمونه ی ما می تواند تاریخچه و محتوای مرورگر های مقصد را بخواند(از طریق حافظه ی نهان) و همچنین از مرورگر سهام سامسونگ تصویر بردارب نماید. در مرورگر دلفین، برنامه ی ما همچنین می تواند فایل های تصویری در حافظه ی نهان را جایگزین کند و تصاویر جایگزین شده را نمایش دهد(به جای فایل های وب قانونی).

از آسیب پذیری های شناسایی شده در این مقاله، سه آسیب پذیری توسط پروژه ی امنیتی برنامه ی وبِ باز(OWASP) به عنوان 10 آسیب پذیریِ اصلی برای برنامه های تلفن همراه شناسایی شده است [38]- در جدول 5.

ذخیره سازیِ نا امن داده ها شاملِ عمل ذخیره سازی اطلاعات حساس بدون رمزنگاری و یا حفاظت از اطلاعات در مکانی نا امن است. برای مثال، ما در این تحقیق مشخص کردیم که مرورگر سهام سامسونگ تصاویر را در حافظه ی داخلی با یک مجوز فایلِ ضعیف ذخیره می کند. برنامه ی نمونه ی ما قادر به تزریقِ محتوای مخرب در حافظه ی نهانِ مرورگر دلفین است. بنابراین، این مرورگر به یک حمله ی تزریقی از سمت سرویس گیرنده آسیب پذیر است.

نشت اطلاعات به طورِ ناخواسته می تواند نتیجه ی ذخیره ی اطلاعات در مکانی نا امن باشد (به عنوان مثال داده ی ذخیره شده می تواند توسط شخص یا برنامه ی غیر مجاز مورد دسترسی قرار گیرد). برای مثال، این 4 مرورگرِ مورد بررسی در این مقاله، داده را در حافظه ی به اشتراک گذاری شده ی خارجی ذخیره می کنند، که می تواند توسط برنامه های دیگر مورد دسترسی قرار گیرد.

8. نتیجه گیری و کارِ پیشِ رو

در این مقاله ما یک مدل دشمن ارائه کردیم که می تواند در مطالعه ی امنیت مرورگر های سبک استفاده شود، که یک روش محبوب از دسترسی به سرویس های ابری است. برای نشان دادنِ عملیِ این مدلِ دشمن، ما یک برنامه ی نمونه ساختیم. با استفاده از 4 مرورگرِ سبک محبوبِ مورد مطالعه، ما مشخص کردیم که دلفین، مرورگرِ CM و مرورگرِ UC اطلاعات کاربری حساس را در حافظه ی خارجیِ اشتراک گذاری شده ذخیره می کنند. اگرچه مرورگرِ سهام

سامسونگ اطلاعات کاربری حساس را در حافظه ی داخلی دستگاه ذخیره می کند. ما نشان دادیم که یک مهاجم می تواند از مجوز های فایل ضعیف برای دسترسی به اطلاعات ذخیره شده بهره برداری کند.

این آسیب پذیری به علتِ یک نقص طراحی است؛ ذخیره سازی فایلِ نادرست توسط مرورگرها. این مسئله همچنین نیاز به اطمینان برای ذخیره سازی فایل های اندروید و سیستم مجوز فایل به منظور بهبود را برای جلوگیری از دسترسیِ افراد غیر مجاز به فایل های ایجاد شده توسط برنامه های دیگر تقویت می کند. از دیدگاهِ اجرایی، بارگذاری محتوا از حافظه داخلی بدون به خطر انداختن امنیت، بسیار کارآمد تر است[29]. با این وجود، به نظر می رسد که مرورگر ها هنوز هم برای استفاده از حافظه ی خارجی طراحی شده اند تا استفاده از حافظه ی داخلی، شاید با توجه به حجم ذخیره سازیشان اینطور باشند. بنابراین، ما باور نداریم که توصیه ای ساده به مرورگر ها برای ذخیره سازی فایل ها در حافظه ی داخلی استراتژیِ قابل دوامی باشد، مگر اینکه خطر مرتبط با مجوز فایل همچنان مورد خطاب قرار بگیرد(در [34] [56]). بنابراین، یک راه حل کوتاه مدت برای مرورگر ها می تواند ذخیره ی فایل های غیر حساس یا بزرگ (مانند کلیپ های ویدئویی) در حافظه ی خارجی باشد. با این حال، بهتر است به کاربران هشدار داده شود که این فایل ها ممکن است توسط برنامه های دیگر مورد دسترسی قرار بگیرند.

جدول5 - شناسایی آسیب پذیریِ امنیت 10 برنامه ی تلفن همراه توسط OWASP

آسیب پذیریِ امنیت 10 برنامه ی تلفن همراه OWASP	شناسایی آسیب پذیری در مرورگر ها			
	دلفین	UC	CM	سهام سامسونگ
کنترل سمت سرویس دهنده ی ضعیف ذخیره سازیِ داده ی نا امن	بله	بله	بله	
حفاظتِ ناکافیِ لایه ی انتقال نشت داده به طور ناخواسته	بله	بله		بله
احراز هویت و مجوز ضعیف				
رمزنگاری شکسته تزریق سمت سرویس گیرنده	بله			
تصمیم گیری های امنیتی از طریقِ ورودیِ غیر قابل اطمینان				
اداره ی جلسه ی نا مناسب				
حمایتِ فاقد دودویی				

تشکر و قدردانی

دیدگاه ها و نظرات بیان شده در این مقاله توسط خودِ نویسندگان است، نه سازمان هایی که آنها را همراهی می کنند. همچنین نویسندگان مایل اند از سردبیر و منتقدان ناشناس برای ارائه ی انتقادات و پیشنهادات سازنده و سخاوتمندانه تشکر کنند. با وجود کمک های ارزشمندِ آنها، هرگونه خطای باقی مانده در این مقاله تنها به نویسندگانِ این مقاله نسبت داده می شود.

References

[1] O.S. Adebayo, N.A. Aziz, Static code analysis of permission-based features for Android Malware Classification using Apriori Algorithm with Particle Swarm Optimization, J. Inf. Assur. Secur. 10 (4) (2015).
[2] C. Amrutkar, K. Singh, A. Verma, P. Traynor, VulnerableMe: Measuring Systemic Weaknesses in Mobile Browser Security, Information Systems Security, Springer 2012, pp. 16–34.
[3] C. Amrutkar, P. Traynor, P.C. van Oorschot, An empirical evaluation of security indicators in mobile Web browsers, Mobile Comput. IEEE Trans. 14 (5) (2015) 889–903.
[4] Z. Aung, W. Zaw, Permission-based Android malware detection, Int. J. Sci. Technol. Res. vol. 2 (3) (2013) 228–234.
[5] A. Azfar, K.-K.R. Choo, L. Liu, An Android communication App forensic taxonomy, J, Forensic Sciences (2016), http://dx.doi.org/10.1111/ 1556-4029.13164].
[6] C. Bansal, S. Preibusch, N. Milic-Frayling, Cache Timing Attacks Revisited: Efficient and Repeatable Browser History, OS and Network Sniffing', ICT Systems Security and Privacy Protection, Springer 2015, pp. 97–111.
[7] D.J., Bernstein 2005, Cache-timing attacks on AES.
[8] B.B. Brumley, 'Cache Storage Attacks', Topics in Cryptology–-CT-RSA, Springer 2015, pp. 22–34.
[9] I. Burguera, U. Zurutuza, S. Nadjm-Tehrani, Crowdroid: behavior-based malware detection system for android, in: Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices, ACM, 2011, pp. 15–26.
[10] L. Bustos, Speed Kills Conversion Rates, 2012 o⟨http://www.getelastic.com/ site-speed-infographic/⟩4.
[11] P. Chris, Best Android browsers, 2015 edition: speed, features, and design, updated 8 April 2015, Phonearena.com, viewed 12 September 2015, o⟨http:// www.phonearena.com/news/Best-Android-browsers-2015-edition-design-fea tures-and-performance_id67848⟩4.
[12] C. D'Orazio, K.-K.R. Choo, An adversary model to evaluate DRM protection of video contents on iOS devices, Comput. Secur. 56 (2015) 94–110.
[13] D. Daryabar, A. Dehghantanha, B. Eterovic-Soric, K.-K.R. Choo, Forensic Investigation of OneDrive, Box, GoogleDrive and Dropbox Applications on Android and iOS Devices, Aust. J. Forensic Sci. (2016), http://dx.doi.org/10.1080/ 00450618.2015.1110620].
[14] M.L. Das, N. Samdaria, On the security of SSL/TLS-enabled applications, Appl. Comput. Inf. 10 (1) (2014) 68–81.
[15] L. Desmet, M. Johns, Real-time communications security on the web, Internet Computing, IEEE 18 (6) (2014) 8–10.
[16] Q. Do, B. Martini and K.-K.R. Choo, Enforcing File System Permissions on Android External Storage. In Proceedings of 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2014), IEEE Computer Society Press, 2014, pp. 949–954.

[17] Q. Do, B. Martini, K.-K.R. Choo, Exfiltrating data from Android devices, Comput. Secur. 48 (2015) 74–91.
[18] W. Du, L. Yang, J. Kizza & X. Yuan, New hands-on labs on browser security, in: Proceedings of the 45th (ACM) Technical Symposium on Computer Sience Education, (ACM), 2014, pp. 717–717.
[19] A.P. Felt, R.W. Reeder, H. Almuhimedi & S. Consolvo, 2014, Experimenting at scale with google chrome's SSL warning, in: Proceedings of the 32nd Annual ACM Conference on Human factors in Computing Systems, ACM, pp. 2667– 2670.
[20] X. Fu, X. Sun, Q. Liu, L. Zhou, J. Sh, Achieving efficient Cloud search services: multi-keyword Ranked Search over Encrypted Cloud data supporting parallel computing, IEICE Trans. Commun. E98-B (1) (2015) 190–200, 2015.
[21] M. Hanif, M.S. Vighio, Z. Hussain, N.A. Memon, Comparative Study of TopRanked Web Browsers, Bahria Univ. J. Inf. Commun. Technol. 8 (1) (2015) 93.
[22] R., Hay, Opera Mobile Cache Poisoning XAS, September 2011.
[23] D. He, D. Zeadally, L. Wu, 'Certificateless public auditing scheme for cloudassisted wireless body area networks, IEEE Syst. J. (2016), http://dx.doi.org/ 10.1109/JSYST.2015.2428620.
[24] C. Hothersall-Thomas, S. Maffeis & C. Novakovic, BrowserAudit: automated testing of browser security features, in: Proceedings of the 2015 International Symposium on Software Testing and Analysis, ACM, 2015, pp. 37–47.
[25] T. Isohara, K. Takemori & A. Kubota, Kernel-based behavior analysis for android malware detection, Computational Intelligence and Security (CIS), 2011 Seventh International Conference on, IEEE, 2011, pp. 1011–1015.
[26] Y. Jia, Y. Chen, X. Dong, P. Saxena, J. Mao, Z. Liang, 'Man-in-the-browser-cache: Persisting HTTPS attacks via browser cache poisoning', Comput. Secur. 55 (2015) 62–80.
[27] Y. Jia, X. Dong, Z. Liang, P. Saxena, 'I know where you've been: Geo-inference attacks via the browser cache', Internet Computing, IEEE 19 (1) (2015) 44–53.
[28] E. Käsper, P. Schwabe, 'Faster and timing-attack resistant AES-GCM', Cryptographic Hardware and Embedded Systems-CHES 2009, Springer 2009, pp. 1–17.
[29] Kim, H., Agrawal, N. & Ungureanu, C. 2011, 'Examining storage performance on mobile devices', in: Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds, ACM, p. 6.
[30] R. Könighofer, 'A fast and cache-timing resistant implementation of the AES', Topics in Cryptology–CT-RSA 2008, Springer 2008, pp. 187–202.
[31] M.D. Leom, K.-K. R. Choo and R. Hunt, Remote wiping and secure deletion on mobile devices: A review. Journal of Forensic Sciences, 2016 (In press).
[32] Liang, B., You, W., Liu, L., Shi, W. & Heiderich, M. 2014, 'Scriptless timing attacks on web browser privacy', Dependable Systems and Networks (DSN), 2014 44th Annual (IEEE)/IFIP International Conference on, (IEEE), pp. 112–123.
[33] G., Linden 2006, Make data useful.
[34] X. Liu, Z. Zhou, W. Diao, Z. Li, K. Zhang, 'An Empirical Study on Android for Saving Non-shared Data on Public Storage', ICT Systems Security and Privacy Protection, Springer 2015, pp. 542–556.
[35] B. Martini, K.-K.R. Choo, Cloud Storage Forensics: ownCloud as a Case Study, Digit. Investig 10 (4) (2013) 287–299.
[36] F. Norouzi, A. Dehghantanha, B. Eterovic-Soric, K.-K.R. Choo, Investigating Social Networking Applications on Smartphones: Detecting Facebook, Twitter, LinkedIn, and Googleþ Artifacts on Android and iOS Platforms, Aust. J.4 Forensic Sci. (2016), http://dx.doi.org/10.1080/00450618.2015.1066854].
[37] D.A. Osvik, A. Shamir, E. Tromer, 'Cache attacks and countermeasures: the case of AES', Topics in Cryptology–CT-RSA 2006, Springer 2006, pp. 1–20.
[38] OWASP 2014, OWASP Mobile Security Project: Top 10 Mobile Risk, o⟨https:// www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab¼Top_10_ Mobile_Risks⟩4.

[39] The performance of Web Application 2008, Aberdeen Group. o⟨http://v1. aberdeen.com/launch/report/research_report/5136-RR-performance-web-ap plication.asp⟩4.
[40] V. Prokhorenko, K.K.R. Choo, H. Ashman, Web application protection techniques: a taxonomy', J. Netw. Comput. Appl. 60 (2016) 95–112.
[41] D. Quick, K.-K.R. Choo, Google drive: forensic analysis of Cloud Storage Data Remnants, J. Netw. Comput. Appl. 40 (2014) 179–193.
[42] D. Quick, K.-K.R. Choo, Digital Droplets: Microsoft SkyDrive forensic data remnants, Future Gener. Comput. Syst. 29 (6) (2013) 1378–1394.
[43] D. Quick, K.-K.R. Choo, Dropbox Analysis: Data Remnants on User Machines, Digit. Investig. 10 (1) (2013) 3–18.
[44] D. Quick, K.-K.R. Choo, Forensic collection of cloud storage data: does the act of collection result in changes to the data or its metadata? Digit. Investig. 10 (3) (2013) 266–277.
[45] Y. Ren, J. Shen, J. Wang, J. Han & S. Lee, Mutual Verifiable Provable Data Auditing in Public Cloud Storage,' Journal of Internet Technology, vol. 16, 2, pp. 317-323.
[46] T., Roessler & A., Saldhana, Web Security Context: User Interface Guidelines, 2010 o⟨http://www.w3.org/TR/wsc-ui/⟩4.
[47] V. Saraswat, D. Feldman, D.F. Kune, S. Das. Remote cache-timing attacks against AES' Proceedings of the First Workshop on Cryptography and Security in Computing Systems, ACM, 2014, pp. 45–48.
[48] M. Shariati, A. Dehghantanha, K.-K.R. Choo, SugarSync forensic analysis, Aust. J. Forens. Sci. 48 (1) (2016) 95–117.
[49] K.A. Talha, D.I. Alper, C. Aydin, APK Auditor: permission-based Androidmalware detection system, Digit. Investig. 13 (2015) 1–14.
[50] N. Tsalis, N. Virvilis, A. Mylonas, T. Apostolopoulos, D. Gritzalis, Browser Blacklists: A Utopia of Phishing Protection, Springer, 2015.
[51] The U.S. Mobile App Report 2014, ComScore.
[52] N. Virvilis, A. Mylonas, N. Tsalis, D. Gritzalis, Security busters: web browser security vs. rogue sites, Comput. Secur. 52 (2015) 90–105.
[53] H. Wadkar, A. Mishra, & A. Dixit, 2014, 'Prevention of information leakages in a web browser by monitoring system calls', Advance Computing Conference (IACC), 2014 IEEE International, IEEE, pp. 199–204.
[54] Z. Wang F.X. Lin L. Zhong M. Chishtie. 'Why are web browsers slow on smartphones? in: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, ACM 2011 91 96.
[55] Web Storage, W3C (2015). [56] D., Wu & R.K., Chang 2011, Indirect File Leaks in Mobile Applications.
[57] Y. Yang, H. Cai, Z. Wei, H. Lu and K.-K.R. Choo, 2016. Towards lightweight anonymous entity authentication for IoT applications, in: Proceedings of 21st Australasian Conference on Information Security and Privacy - ACISP 2016, Melbourne, Australia, Volume 9722/2016 of Lecture Notes in Computer Science (pp. 265–280), Springer-Verlag, 4–6 July.
[58] Y. Yang, J. Lu, K.K.R. Choo and J. Liu, 2015. On lightweight security Enforcement in Cyber-physical Systems, in: Proceedings of International Workshop on Lightweight Cryptography for Security & Privacy (LightSec 2015), Bochum, Germany, Volume 9542/2016 of Lecture Notes in Computer Science (pp. 97– 112), Springer-Verlag.
[59] X., Zhang, K., Ying, Y., Aafer, Z., Qiu & W., Du, Life after App Uninstallation: Are the Data Still Alive? Data Residue Attacks on Android', NDSS, 2016.
[60] X. Zhou S. Demetriou D. He M. Naveed X. Pan X. Wang C.A. Gunter K. Nahrstedt. 'Identity, location, disease and more: Inferring your secrets from android public resources', in: Proceedings of the 2013 ACM SIGSAC conference on Computer communications security ACM, 2013, pp. 1017–01028.
[61] Y. Zhou, X. Jiang, X. 2012, 'Dissecting android malware: Characterization and evolution', Security and Privacy (SP), 2012 IEEE Symposium on, IEEE, pp. 95– 109.