ده خطر امنیتی برتر در برنامه‌های موبایل

پروژه امنیت برنامه‌های کاربردی موبایل

نسخه ۱,۰

توضیحات

پروژه امنیت موبایل OWASP باهدف کمک به گروه‌های امنیتی و به‌منظور حفاظت از برنامه‌های موبایلی ،اطلاعاتی را درباره امنیت موبایل گردآوری و تحلیل می‌نماید. در واقع با دسته‌بندی خطرات امنیتی موبایل و ارائه راهکارهای کنترلی سعی می‌شود تا تأثیرات و احتمال سوءاستفاده‌ها کاهش یابد. تمرکز اصلی در این پروژه بر لایه برنامه کاربردی است. بااین‌حال در هنگام مدل کردن تهدیدات و ارائه کنترل‌ها به خطرات شبکه‌های انتقالی و پلتفرم موبایل نیز توجه می‌شود. به‌علاوه فقط به برنامه‌های کاربردی موبایل در طرف کاربر نگاه نمی‌شود بلکه زیرساخت‌های سمت سرور که برنامه‌ها با آن‌ها مرتبط‌اند نیز موردتوجه است و همچنین به‌طور ویژه بر ادغام بین برنامه‌ها و سرویس‌های احراز هویت راه دور و خصوصیات خاص بستر ابری تمرکز می‌شود.

این سند اولین نسخه از ترجمه فارسی پروژه امنیت برنامه‌های کاربردی موبایل OWASP است و بر اساس نسخه انگلیسی ۲۰۱۴ از پروژه امنیت برنامه‌های کاربردی موبایل OWASP است.

این سند ترجمه توسط اعضای آزمایشگاه پیشرفته شبکه و امنیت از دانشگاه امام رضا علیه‌السلام منتشرشده است.

اعضای گروه:

- دکتر حمیدرضا محروقی[۱]
- مهندس محمد حامد دادپور[۲]
- مهندس سبحان علی‌آبادی[۳]

از همه‌کسانی که ما را در این پروژه یاری رساندند به‌ویژه از جناب میلان سین تاکور[۴] کمال تشکر را داریم.

[۱] Dr. Hamid Reza Mahrooghi, Email: mahrooghi@ce.sharif.edu

[۲] Mohammad Hamed Dadpour, Email: Hamed.dadpour@gmail.com

[۳] Sobhan Aliabady, Email: sobhan.aliabady@gmail.com

[۴] Milan Singh Thakur

فهرست

مقدمه

در سال ۲۰۱۳ آمارهایی از آسیب‌پذیری‌های جدید برنامه‌های کاربردی موبایل جمع‌آوری شد؛ آنچه اینجا مشاهده می‌شود نتیجه‌ای از این اطلاعات است.

شکل ۱ – ده خطر برتر امنیتی موبایل

M1- کنترل‌های ضعیف سمت سرور

این مربوط به آسیب‌پذیری‌ها در سمت سرور و برنامه‌های وب است که OWASP به‌طور جداگانه آن را در پروژه های OWASP Web Top Ten یا Cloud Top Ten بررسی نموده‌است.

M2- ذخیره ناامن داده‌ها

عوامل تهدید

دزدیده شدن یا گم‌شدن موبایل ؛ یک بدافزار یا برنامه دوباره بسته‌بندی‌شده[۵] برای دزدیدن اطلاعات

[۵] Repackaged app

نحوه حمله

با دسترسی فیزیکی به موبایل می‌توان آن را به یک کامپیوتر متصل نمود و با کمک ابزارهای آزاد، اطلاعات برنامه‌های نصب‌شده را استخراج نمود و همچنین با نصب یک بدافزار یا نرم‌افزار معتبر دستکاری‌شده می‌توان اطلاعات را دزدید. این اطلاعات معمولاً شامل اطلاعات هویتی و یا دیگر اطلاعات حساس هست.

علت ضعف امنیتی

گروه‌های برنامه‌نویس به اشتباه فرض می‌کنند که کاربران یا بدافزارها به سیستم فایل موبایل دسترسی ندارند و ازاین‌رو نمی‌توانند داده‌های حساس ذخیره‌شده در حافظه را ببینند. سیستم فایل‌ها به‌راحتی قابل‌دسترس هستند؛ به همین دلیل همگان باید منتظر یک کاربر خرابکار یا بدافزاری باشند که قصد دزدیدن اطلاعات را دارد. روش‌های حفاظت رمزنگاری در موبایل‌های روت شده یا قفل‌شکسته به‌راحتی دور زده می‌شود. زمانی که داده‌ای موردحفاظت نباشد می‌توان از ابزارهایی برای مشاهده اطلاعات برنامه‌ها استفاده کرد.

تأثیرات فنی

در بهترین حالت اطلاعات یک نفر و در بدترین حالت اطلاعات افراد بی‌شماری قابل‌دسترسی است. این اطلاعات می‌تواند شامل :نام کاربری، توکن ها ی احراز هویت، رمزها، کوکی‌ها، اطلاعات مکانی، UDID/EMEI، نام موبایل، نام ارتباط شبکه‌ای، اطلاعات شخصی نظیر DOB و آدرس و روابط اجتماعی و اطلاعات کارت اعتباری، اطلاعات برنامه‌ها نظیر لاگ های ذخیره‌شده و اطلاعات عیب‌یابی و پیام‌های ذخیره‌شده برنامه و تاریخچه مبادلات باشد.

تأثیرات تجاری

این نوع آسیب‌پذیری‌ها معمولاً خطرات جدی برای تجارت‌ها دارند ازجمله سرقت شناسه، کلاه‌برداری، صدمه به اعتبار، نقض سیاست خارجی و از دست رفتن اسناد.

تشخیص آسیب‌پذیری

برنامه‌ها برای ذخیره اطلاعات باید از APIهایی استفاده کنند که داده‌ها را به‌صورت امن ذخیره می‌کنند. OWASP مشاهده کرده است که اغلب داده‌ها با روش‌های ناامن ذخیره‌شده‌اند؛ ازجمله: پایگاه داده SQLite، فایل‌های لاگ، فایل Plist، فایل اظهارنامه و ذخیره داده‌های XML، ذخیره داده‌های باینری،ذخیره کوکی‌ها، کارت حافظه SD، همسان‌سازی ابری. همچنین باوجود ذخیره اطلاعات به‌صورت رمز شده حمله‌کننده‌ها می‌توانند با انجام یک حمله باینری بر روی برنامه کلیدهای رمزنگاری را به دست آورند.

جلوگیری از آسیب‌پذیری

قاعده اصلی در برنامه‌های موبایل این هست که فقط اطلاعات لازم را ذخیره کنند ولی به‌عنوان یک برنامه‌نویس باید بدانیم که اطلاعات با یک لمس در موبایل از بین می‌رود و از طرفی باید احتمال سرقت اطلاعات با یک سوءاستفاده ریشه را در نظربگیریم و آن را بررسی نماییم. با این حال اگر قابلیت استفاده در برابر امنیت بسیار مهم‌تر بود پیشنهاد می‌شود که APIهای امن ذخیره داده به‌طور دقیق بررسی و به طرز صحیح استفاده شود.

در اندروید می‌توان از روش‌های زیر استفاده نمود:

- استفاده از متد setStorageEncryption برای ذخیره رمز شده داده‌های محلی
- استفاده از توابع کتابخانه javax.crypto یا کلیدهای متقارن AES128 برای ذخیره رمز شده داده‌ها بر کارت حافظه SD
- اطمینان از MODE_WORLD_READABLE نبودن اشیاء Shared Preferences مگر برای اشتراک اطلاعات بین برنامه‌ها
- اجتناب از وابستگی فقط به کلیدهای رمزنگاری کد شده در هنگام ذخیره اطلاعات حساس
- رسیدگی برای فراهم آوردن یک‌لایه رمزنگاری اضافی در بالای هر روش رمزنگاری پیش‌فرض در سیستم

M3- حفاظت ناکافی در لایه انتقال

عوامل تهدید

معمولاً برنامه‌های موبایل، داده‌ها را در یک مد مشتری-سرویس‌دهنده تبادل می‌کنند. برای انتقال داده‌ها، از شبکه اینترنت و دیگر بسترهای انتقال داده در موبایل استفاده می‌شود. حمله‌کنندگان می‌توانند با سوءاستفاده از آسیب‌پذیری‌های موجود در این بسترها داده‌های حساس را شنود کنند. از عوامل تهدیدات می‌توان دسترسی متخاصم به شبکه محلی برای آگاهی از ترافیک شبکه ، دستگاه‌های شبکه نظیر روتر و... و بدافزارهای موبایلی نام برد.

نحوه حمله

حمله‌کننده با نظارت بر ترافیک شبکه به‌خصوص ترافیک غیر رمز شده می‌تواند به اطلاعات دلخواه برسد که معمولاً برای حمله‌کنندگان آسان است.

علت ضعف امنیتی

برنامه‌ها نمی‌توانند در هرجایی از SSL استفاده نمایند لذا به‌طور مداوم از ترافیک شبکه خود محافظت نمی‌کنند. در برخی موارد هم پیاده‌سازی امنیت انتقال در برنامه‌ها به‌درستی انجام نمی‌شود. البته نقص‌های اساسی را می‌توان با نظارت بر ترافیک شبکه مشاهده کرد. رفع آن‌ها با بررسی برنامه و پیکربندی آن حاصل می‌شود.

تأثیرات فنی

نمایش آشکار اطلاعات کاربری می‌تواند موجب سرقت حساب کاربری شود و در مواردی که آن حساب کاربری مربوط به کاربر مدیر یک سایت باشد موجب حمله به‌کل سایت می‌شود. پیکربندی ضعیف SSL می‌تواند منجر به حملات مردمیانی و فیشینگ شود.

تأثیرات تجاری

مهم‌ترین مشکل نقض حریم شخصی است که موجب سرقت هویت، کلاه‌برداری و صدمه به اعتبار می‌شود.

تشخیص آسیب‌پذیری

با مشاهده ترافیک برنامه از طریق یک واسط و با پاسخ به این سؤالات می‌توان به وجود آسیب‌پذیری پی برد:

- آیا همه ارتباطات رمز شده‌اند؟
- آیا گواهی‌های SSL در اطلاعات موجود هستند؟
- آیا گواهی‌های SSL با امضای خودشان هستند؟
- آیا SSL از یک طول رمز کافی استفاده می‌کند؟
- آیا برنامه موردنظر گواهی‌نامه‌های معتبر پذیرفته‌شده کاربر را به‌عنوان منابع موثق می‌پذیرد؟

جلوگیری از آسیب‌پذیری

- برنامه‌نویسان باید به‌طورکلی لایه شبکه را ناامن فرض کنند و به خطر شنود توجه کنند؛
- برای انتقال داده‌های حساس حتماً باید از SSL استفاده نمود که کلید رمز استاندارد قوی با طول مناسب را به کار می‌برد و گواهی‌نامه‌های آن توسط یک مرکز معتبر امضاشده باشد.
- هیچ‌گاه نباید از گواهینامه‌های "خود امضاشده" استفاده نمود.
- باید از نشست‌های SSL تودرتو به دلیل امکان فاش شدن توکن نشست اجتناب شود.
- یک اتصال امن باید بعد از احراز هویت سرور مقصد برقرار شود و فوراً از طریق پیامی تشخیص نامعتبر بودن گواهی‌نامه را به کاربر اطلاع دهد.
- هیچ‌گاه اطلاعات حساس از طریق کانال‌های غیر امن مانند پیامک و... نباید ارسال شود.
- به دلیل امکان شنود اطلاعات در دستگاه قبل از رمز شدن توسط SSL و وجود آسیب‌پذیری در این پروتکل بهتر است از یک‌لایه امنیتی جداگانه قبل از SSL برای اطمینان بیشتر استفاده شود.
- در اندروید باید بعد از پایان توسعه نرم‌افزار کدهای مربوط به پذیرش همه گواهینامه‌ها را پاک نمود. مانند:

  org.apache.http.conn.ssl.AllowAllHostnameVerifier
  SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER

  در صورت استفاده از کلاس SSLSocketFactory باید مطمئن شد که بعدازآن متدهای CheckServerTrust برای بررسی گواهی‌نامه سرور پیاده‌سازی شده است.

سناریوهای متخصصان آزمون نفوذ

- عدم بررسی گواهی‌نامه

برنامه با سرور یک ارتباط امن مبتنی بر SSL ایجاد می‌کند ولی برنامه هر گواهی‌نامه ارائه‌شده از سمت سرور را بدون بررسی می‌پذیرد و این تصدیق اعتبار دوطرفه را از بین می‌برد و برنامه نسبت به حمله مردمیانی از طریق یک پراکسی SSL مستعد است.

- مذاکره ضعیف در دست تکانی

بخشی از مرحله دست تکانی مذاکره برای تعیین رشته رمز است اگر این رشته رمز کوتاه تعیین شود رمزنگاری ضعیف انجام می‌شود و درنتیجه به‌آسانی توسط یک متخاصم رمزگشایی می‌شود.

- نشت اطلاعات حریم خصوصی

انتقال اطلاعات شخصی بین برنامه و سرور از طریق کانال ناامن به‌جای SSL، قابلیت اعتماد به این اطلاعات را به خطر می‌اندازد.

M4- نشت ناخواسته اطلاعات

عوامل تهدید

متخاصم با عامل‌هایی همچون یک بدافزار موبایلی یا یک برنامه دستکاری‌شده یا با دسترسی مستقیم می‌تواند از آسیب‌پذیری‌های در این زمینه سوءاستفاده کند.

نحوه حمله

با دسترسی مستقیم به موبایل و با ابزارهای آزاد جرم یابی می‌توان یک حمله انجام داد. همچنین متخاصم می‌تواند با اجرای یک کد مخرب که APIهای مجاز را فراخوانی می‌کند یک حمله را هدایت کند.

علت ضعف امنیتی

معمولاً نشت ناخواسته اطلاعات زمانی اتفاق می‌افتد که برنامه‌نویس اطلاعات حساس برنامه را در مکانی ذخیره می‌کند که توسط سایر برنامه‌ها به‌آسانی قابل‌دسترسی است و یا در هنگام پردازش برنامه، اطلاعات واکشی شده توسط سیستم‌عامل در مکانی قرار می‌گیرد که در دسترس برنامه‌های دیگر است. این‌ها از بی‌اطلاعی برنامه‌نویس درباره نحوه ذخیره و پردازش اطلاعات توسط سیستم‌عامل ناشی می‌شود. البته با بررسی مکان‌های در دسترس همه برنامه‌ها می‌توان به وجود این آسیب‌پذیری پی برد.

تأثیرات فنی

این آسیب‌پذیری با استخراج اطلاعات حساس ضربه شدیدی به کاربر می‌زند.

تأثیرات تجاری

اطلاعات دزدیده‌شده می‌تواند موجب نقض حریم خصوصی و صدمه به اعتبار و کلاه‌برداری شود.

تشخیص آسیب‌پذیری

آسیب‌پذیری‌های موجود در سیستم‌عامل ، محیط کامپایلر، فریمورک ها و سخت‌افزارهای جدید و... به همراه عدم دانش کافی برنامه‌نویسان می‌تواند نشت ناخواسته اطلاعات را به دنبال داشته باشد. این آسیب‌پذیری در فرایندهای داخلی بیشتر دیده‌شده است:

- روش سیستم‌عامل برای ذخیره کردن اطلاعات، ضرب کلیدها، لاگ ها، بافرها در حافظه نهان
- روش فریمورک توسعه برای ذخیره کردن این اطلاعات در حافظه نهان

- روش فریمورک های کسب‌وکار ، اجتماعی، آنالیز و تبلیغاتی برای ذخیره کردن اطلاعات در حافظه نهان

جلوگیری از آسیب‌پذیری

در ابتدا مدل تهدید برای سیستم‌عامل و فریمورک را بررسی می‌کنیم تا نحوه اداره هر یک از موارد زیر را دریابیم:

- URL Caching (Both request and response)
- Keyboard Press Caching
- Copy/Paste buffer Caching
- Application backgrounding
- Logging
- HTML5 data storage
- Browser cookie objects
- Analytics data sent to 3rd parties

آشنایی با چگونگی کار پیش‌فرض سیستم‌عامل و فریمورک برای تعیین و اجرای کنترل‌ها لازم است.

M5- ضعف در احراز هویت و اعطای مجوز

عوامل تهدید

متخاصم با ابزارهای موجود می‌تواند حملات خودکاری برای سوءاستفاده از آسیب‌پذیری‌های فرایندهای احراز هویت و اعطای مجوز اجرا نماید.

نحوه حمله

متخاصم فقط یک‌بار لازم است فرایند احراز هویت را بررسی نماید و چگونگی آسیب‌پذیری را کشف کند. یک بدافزار یا باتنت با ارسال پیام‌های درخواست سرویس به سرویس‌دهنده برنامه فرایند احراز هویت را دور می‌زند و هرگونه ارتباط مستقیم بین برنامه و سرویس‌دهنده را جعل می‌کند.

علت ضعف امنیتی

یک متخاصم با دور زدن فرایند احراز هویت ضعیف می‌تواند به‌طور گمنام قابلیت‌هایی را در سرویس‌دهنده یا برنامه کاربردی اجرا کند. احراز هویت ضعیف بیشتر به دلیل شکل کلمه عبور است که بر مبنای پین‌های ۴ رقمی است.

نیازمندی‌های احراز هویت در برنامه‌های کاربردی با احراز هویت در وب کاملاً متفاوت است. در وب این کار به‌صورت آنلاین و بلادرنگ انجام می‌شود به همین دلیل دسترسی به اینترنت لازم و همیشگی است ولی در برنامه‌های کاربردی کاربرآن‌همیشه آنلاین نیستند و از طرفی اتصال اینترنت در موبایل قابل‌اتکا و همیشگی نیست ازاین‌رو احراز هویت آفلاین انجام می‌شود. در هنگام پیاده‌سازی فرایند احراز هویت در برنامه‌های کاربردی باید موارد بسیاری را در نظر گرفت.

برای تشخیص احراز هویت ضعیف باید با حمله باینری سعی کرد با دور زدن احراز هویت آفلاین، قابلیت‌های برنامه را اجرا کرد. بعلاوه با از بین بردن هرگونه توکن نشست از درخواست‌های POST/GET باید سعی کرد تا قابلیت‌های سرویس‌دهنده پشتیبان را اجرا کرد.

برای تشخیص ضعف در فرایند اعطای مجوز باید بتوان با حمله باینری قابلیت‌هایی از برنامه را اجرا کرد که فقط کاربران ویژه مجوز اجرای آن را دارند. به‌علاوه متخصصان باید سعی کنند هر قابلیت ممتازی را در سمت سرویس‌دهنده پشتیبان اجرا کنند برای این منظور از توکن نشست کم امتیاز در درخواست‌های POST/GET استفاده می‌شود.

تأثیرات فنی

وجود این آسیب‌پذیری سبب می‌شود هویت کاربر را نتوان احراز کرد و ازاین‌رو راه‌حل‌های شناسایی کاربری که درخواست قابلیت یا خدمتی را می‌دهد و همچنین راه‌حل‌های ثبت اطلاعات و بازرسی کاربر دیگر کارایی نداشته باشد. به همین دلیل در هنگام رخداد یک حمله نمی‌توان منبع آن و ماهیت سوءاستفاده و چگونگی مقابله با آن را تشخیص داد.

زمانی کنترل‌های احراز هویت شکسته می‌شود که هویت کاربر غیرقابل تشخیص باشد. هویت کاربر به نقش کاربر در سیستم و مجوزهایی مرتبط است که حمله‌کننده می‌تواند با جعل هویت کدهایی را اجرا کند و سیستم قادر به اعتبارسنجی مجوزهای کاربر نیست؛ بنابراین هم کنترل‌های احراز هویت و هم کنترل‌های اعطای مجوز شکسته می‌شود. شکسته شدن کنترل‌های اعطای مجوز می‌تواند موجب مشکل امتیاز بیش‌ازحد شود.

تأثیرات تجاری

حداقل نتیجه احراز هویت ضعیف صدمه به اعتبار است و درواقع یک کاربر به‌طور گم نام یا مشخص با شکستن کنترل‌های اعطای مجوز قابلیت‌هایی را با امتیاز بالا در سیستم‌عامل اجرا کند که می‌تواند موجب صدمه به اعتبار و کلاه‌برداری و سرقت اطلاعات شود.

تشخیص آسیب‌پذیری

باید از طراحی الگوی های احراز هویت ناامن زیر در برنامه‌های موبایل اجتناب شود:

- در هنگام ایجاد برنامه موبایلی از یک برنامه وب نباید عوامل احراز هویت آن کمتر از عوامل احراز هویت برنامه وب باشد.
- احراز هویت محلی می‌تواند موجب آسیب‌پذیری‌های در سمت مشتری شود. برنامه‌ها به دلیل ذخیره محلی داده‌ها یا برای نیازهای تجاری فوری باید احراز هویت کاربر را آفلاین انجام دهند ولی این نوع احراز هویت می‌تواند با دست‌کاری در زمان اجرای برنامه یا تغییرات باینری در دستگاه‌های روت شده دور زده شود.
- در صورت امکان باید همه درخواست‌های احراز هویت در سمت سرور رسیدگی شود و همچنین باید مطمئن شد که داده‌ها بعد از تأیید هویت کاربر برای برنامه مشتری ارسال شود.
- اگر در برنامه‌ای ذخیره محلی داده‌ها نیاز است به‌منظور اطمینان از در دسترس بودن داده‌ها فقط با ارائه گواهی‌نامه معتبر باید داده‌ها توسط یک کلید رمزنگاری مشتق شده از گواهی‌نامه ورود کاربر رمز شوند. البته این روش در برابر حمله باینری آسیب‌پذیر است و داده‌ها قابل رمزگشایی است.

- قابلیت "یادآوری مشخصات[۶]" برای احراز هویت پایا در برنامه‌ها نباید هرگز کلمه عبور را در دستگاه ذخیره کند.
- برای اطمینان از کاهش خطر دسترسی به برنامه در دستگاه‌های دزدیده‌شده باید برنامه موبایل از یک توکن احراز هویت خاص دستگاه و قابل‌ابطال توسط کاربر استفاده کند.
- نباید از مقادیر قابل جعل همچون شناسه‌های دستگاه یا مکان جغرافیایی برای احراز هویت یک کاربر استفاده نمود.
- در صورت امکان باید قواعدی برای انتخاب کلمه عبور تعیین نمود تا برای نمونه کابران نتوانند کلمات عبور ۴ رقمی انتخاب نمایند.

جلوگیری از آسیب‌پذیری

توسعه‌دهندگان باید فرض کنند هر فرایند احراز هویت و اعطای مجوز در سمت مشتری مانند برنامه‌های موبایلی قابل دور زدن است و در صورت امکان باید همه کنترل‌ها را در سمت سرویس‌دهنده وب نیز به‌منظور بررسی بیشتر، اعمال کنند ولی از طرفی لازمه برخی برنامه‌های موبایل احراز هویت و اعطای مجوز در داخل برنامه و به‌صورت آفلاین است در این صورت باید هرگونه تغییر کدهای غیرمجاز با ابزارهای بررسی صحت جاسازی‌شده در کد محلی را تشخیص داد.

سناریوهای نمونه

توسعه‌دهندگان به‌اشتباه فرض می‌کنند که فقط کاربران محرز شده می‌توانند درخواست‌های یک سرویس را به سمت سرور ارسال کنند. از طرفی سرور نیز در طول پردازش یک درخواست، اعتبار کاربر درخواست دهنده را بررسی نمی‌کند؛ ازاین‌رو متخاصم می‌تواند با ایجاد و ارسال یک درخواست جعلی سرویس، یک قابلیت مجاز برای کاربران معتبر را اجرا کنند.

توسعه‌دهندگان به‌اشتباه فرض می‌کنند فقط کاربران مجاز می‌توانند از وجود یک تابع خاص در برنامه موبایل اطلاع یابند. ازاین‌رو آن‌ها انتظار دارند که فقط کاربران مجاز بتوانند از برنامه درخواستی را برای یک سرویس صادر کنند. ولی بخش پشتیبان درخواست را بدون توجه به بررسی هویت کاربر درخواست دهنده پردازش می‌کند و از این طریق متخاصم می‌تواند یک قابلیت خاص را با کاربر کم امتیاز نیز اجرا کند.

توسعه‌دهندگان به‌منظور سهولت استفاده از برنامه ، به کاربران اجازه می‌دهند تا کلمه عبور با طول ۴ رقمی انتخاب کنند. سرور این کلمات عبور را به‌صورت درهم شده ذخیره می‌کند. به دلیل طول کوتاه این کلمه عبور، متخاصم قادر خواهد بود به‌راحتی با کمک جداول درهم کلمه عبور اصلی را پیدا کند. حال اگر فایل کلمات عبور در سرور به هر دلیلی در دسترس قرار گیرد کلمات عبور کاربران به‌راحتی قابل بازیابی است.

M6- رمزنگاری شکننده

عوامل تهدید

با دسترسی مستقیم به دستگاه یا با یک بدافزار می‌توان به داده‌هایی که به گونه نادرست رمز شده‌اند دست‌یافت.

---

[۶] Remember me

نحوه حمله

متخاصم با دسترسی مستقیم به دستگاه یا با کمک یک بدافزار یا ثبت ترافیک شبکه می‌تواند داده‌های رمز شده را مشاهده و رمزگشایی نماید.

علت ضعف امنیتی

متخاصم به دلیل استفاده برنامه از الگوریتم‌های رمزنگاری ضعیف یا نقص‌های فرایند رمزنگاری می‌تواند داده‌های رمز شده را رمزگشایی نماید.

تأثیرات فنی

این آسیب‌پذیری موجب بازیابی غیرمجاز داده‌های حساس رمز شده در موبایل می‌شود.

تأثیرات تجاری

ازجمله نتایج رمزنگاری شکننده می‌توان به نقص حریم خصوصی ، سرقت کد و اطلاعات و خسارت اعتباری اشاره کرد.

تشخیص آسیب‌پذیری

معمولاً برنامه‌ها به دو شیوه از رمزنگاری شکننده استفاده می‌کنند: اول، برنامه‌ها از یک فرایند زمینه برای رمزنگاری و رمزگشایی استفاده می‌کنند که عیب اساسی دارد و قابل سوءاستفاده است.دوم،برنامه یک الگوریتم رمزنگاری و رمزگشایی پیاده‌سازی می‌کند که ذاتاً ضعیف است و توسط متخاصم قابل رمزگشایی است. این دو شیوه در سناریوهای زیر تشریح شده است:

- تکیه به فرایندهای رمزنگاری تعبیه‌شده در برنامه

مدل امنیتی iOS برنامه‌ها را وادار می‌کند تا برای اجرا و همچنین مقابله با مهندسی معکوس، کد خود را رمز و امضا کنند. iOS برنامه را در حافظه رمزگشایی می‌کند و بعد از بررسی صحت امضا کد را اجرا می‌کند. ولی با ابزارهای در دسترس همچون GBD و ClutchMod می‌توان برنامه رمز شده را دانلود نمود و روی یک دستگاه قفل‌شکسته شده اجرا کرد و بعد از رمزگشایی برنامه در حافظه و قبل از اجرا یک کپی از حافظه گرفته می‌شود و با کمک IDA Pro یا Hopper به‌راحتی می‌توان تحلیل ایستا و پویا روی برنامه انجام داد و یک حمله باینری را اجرا کرد. همیشه باید فرض کرد که یک متخاصم می‌تواند هرگونه رمزنگاری کد، فراهم‌شده توسط سیستم‌عامل دستگاه را دور بزند.

- فرایندهای مدیریت کلید ضعیف

اگر کلیدها به‌درستی مدیریت نشوند بهترین الگوریتم‌ها هم نمی‌توانند امنیت را حفظ کنند. برخی خطاها در استفاده صحیح از الگوریتم رمزنگاری وجود دارد برای نمونه کلیدها در مکانی قابل‌دسترس ذخیره می شوند و یا کد نمودن کلیدها در باینری فراموش می شود که موجب آسیب‌پذیری در برابر حمله باینری می‌شود.

- ایجاد و استفاده از پروتکل‌های رمزنگاری سفارشی

استفاده از پروتکل‌ها و الگوریتم‌های رمزنگاری سفارشی و خودساخته ساده‌ترین و بدترین روش رمزنگاری است. همیشه باید از پروتکل‌ها و الگوریتم‌های مورد تأیید متخصصان استفاده نمود و در صورت امکان باید APIهای جدیدترین فناوری‌های رمزنگاری را به کار برد. یک متخاصم با حمله باینری می‌تواند کتابخانه‌های معمول رمزنگاری به همراه کلیدهای کد شده را بیابد. نیازمندی‌های امنیتی فراوان پیرامون رمزنگاری، استفاده از رمزنگاری جعبه سفید[۷] را پراهمیت می‌کند. این فناوری خاص به‌گونه‌ای رمزنگاری را انجام می‌دهد که هیچ قسمتی از اطلاعات حساس مانند کلیدهای رمزنگاری فاش نشود.

- استفاده از الگوریتم‌های رمزنگاری ناامن

بسیاری از الگوریتم‌ها نقص‌های قابل‌توجهی دارند و یا همه نیازمندی‌های امنیتی جدید را برطرف نمی‌کنند.ازجمله : RC2، MD4، MD5، SHA1.

M7- تزریق در سمت مشتری

عوامل تهدید

تزریق داده‌های غیرقابل اطمینان به برنامه‌ها از طریق کاربران خارجی ، کاربران داخلی ، خود برنامه و دیگر برنامه‌های مخرب امکان‌پذیر است.

نحوه حمله

متخاصم حملات متنی ساده‌ای را اجرا می‌کند که از نحو مفسر در برنامه هدف سوءاستفاده می‌کند و بردار تزریق تقریبا می‌تواند هر منبع داده‌ای شامل فایل یا خود برنامه باشد.

علت ضعف امنیتی

تزریق در سمت مشتری موجب اجرای کد مخرب توسط یک برنامه در دستگاه موبایل می‌شود. این کد مخرب به‌عنوان داده‌ی ورودی به یک برنامه تزریق می‌شود و این داده مانند سایر داده‌ها توسط فریمورک پشتیبان کننده برنامه پردازش می‌شود ولی در طول پردازش این داده خاص، وضعیت[۸] برنامه تغییر داده می‌شود و فریمورک آن را به‌عنوان یک کد اجرایی تفسیر می‌نماید. این کد در بهترین حالت در همان محدوده و مجوزهای برنامه عمل می‌کند ولی در بدترین حالت می‌تواند با مجوزهای بالاتر و در محدوده بیشتری اجرا شود که آسیب آن بیشتر از حالت قبلی است.

روش دیگر ، تزریق باینری در یک حمله باینری است که این حمله می‌تواند حتی خطرناک‌تر از تزریق داده به برنامه عمل کند.

---

[۷] White Box Cryptography

[۸] Context

تأثیرات فنی

برای تشخیص صحیح تأثیرات فنی یک برنامه باید مدل تهدید آن را ایجاد کنیم. در هنگامی که برنامه با بیش از یک کاربر در یک دستگاه یا با یک دستگاه اشتراکی یا با پرداخت برای محتوا سروکار داشته باشد حمله تزریق می‌تواند سخت باشد. نکته دیگر، تزریق برای سرریز مؤلفه‌های برنامه است که به دلیل کد مدیریت حفاظت از زبان برنامه، احتمالاً اثرات فنی کمتری دارد.

تأثیرات تجاری

نتایج و اثرات تجاری این آسیب‌پذیری به ماهیت کد مخرب بستگی دارد. معمولاً این کدها اطلاعات حساس مانند رمزهای عبور، کوکی‌های نشست و اطلاعات شناسایی را می‌دزدند و ازاین‌رو اثرات تجاری کلاه‌برداری و نقض محرمانگی را در پی‌دارند.

تشخیص آسیب‌پذیری

بهترین روش برای تشخیص، شناسایی راه‌های ورود اطلاعات به برنامه و بررسی درستی داده‌های ارائه‌شده کاربر و برنامه‌ها است. سریع‌ترین و دقیق‌ترین روش برای اطمینان از کنترل صحیح برنامه بر داده‌ها، بررسی کد برنامه است. تحلیل گران امنیتی با کمک ابزارهای تحلیلی می‌توانند کاربرد مفسرها را پیدا کنند و همچنین جریان داده‌ها در برنامه را ردیابی نمایند. متخصصان آزمون نفوذ با سوءاستفاده ماهرانه از این آسیب‌پذیری‌ها، وجود این مشکلات را تأیید می‌کنند.

به این دلیل که داده‌های ورودی یک برنامه از منابع زیادی می‌آیند؛ لیست کردن این منابع در مشخص کردن هدف کارشان اهمیت دارد. به‌طورکلی حملات تزریق شامل انواع زیر است:

- تزریق SQL: پایگاه داده SQLite می‌تواند مانند برنامه‌های وب در معرض این حمله قرار گیرد. آسیب‌پذیری نسبت به این حمله و درنهایت مشاهده اطلاعات با این روش می‌تواند بسیار خطرناک باشد.
- فایل‌های محلی: اداره فایل‌ها بر روی دستگاه می‌تواند همان خطرات بالا را داشته باشد مگر این‌که این خواندن فایل‌ها فقط مربوط به برنامه‌ای باشد و فایل‌ها در مسیر آن برنامه ذخیره‌شده باشد.
- تزریق JavaScript(XSS) : مرورگرهای موبایل در معرض این حمله هستند و این مرورگرها به کوکی‌های برنامه‌های موبایل نیز دسترسی دارند که می‌تواند منجر به سرقت نشست شود.
- رابط‌های کاربری برنامه‌ها و توابع می‌توانند داده‌هایی را بپذیرند که در یک آزمون فازینگ منجر به شکست برنامه می‌شود. البته به دلیل مکانیسم‌های حفاظتی سیستم‌عامل موبایل این نواقص نمی‌تواند منجر به سرریز شود بااین‌حال چند نمونه به‌عنوان آسیب‌پذیری "Userland" در زنجیره آسیب‌پذیری‌ها برای دستگاه‌های قفل‌شکسته یا روت شده وجود دارد.
- بدافزارها می‌توانند یک حمله باینری علیه لایه "نمایش" (html,css,javascript) یا علیه باینری اجرایی برنامه انجام دهند. این تزریق کد باینری یا با چارچوب برنامه موبایل یا در زمان اجرای برنامه انجام می‌شود.

جلوگیری از آسیب‌پذیری

به‌طورکلی برای جلوگیری از تزریق کد به برنامه نیاز هست تا همه راه‌های ورودی برنامه را پیداکرده و برای آن‌ها نوعی اعتبارسنجی ورودی قرار داده شود.

در IOS:

- Sqlite injection: در هنگام طراحی یک پرس‌وجو از پایگاه داده باید مطمئن شد که داده‌های ارائه‌شده کاربر در قالب یک پرس‌وجوی پارامتری یا Prepared Statment ارسال می‌شود؛ این نوع پرس‌وجو، یک الگوی پرس‌وجو کامپایلر شده است که با استفاده از پارامترهای متغیر قابل سفارشی‌سازی است و برای مقابله با این‌گونه حملات به کار می‌رود. البته استفاده از کاراکترهای %,@ در ورودی پرس‌وجو به‌جای ؟ می‌تواند خطرناک باشد.
- تزریق JavaScript: باید از اعتبارسنجی ورودی در فرمان‌های UIWebView مطمئن شد و فیلترهایی برای کاراکترهای خطرناک جاوا اسکریپت قبل از پردازش آن‌ها و باسیاست لیست سفید روی لیست سیاه در نظر گرفته شود. در صورت امکان به‌جای فراخوانی UIWebkit در برنامه‌ها از مرورگر safari برای باز کردن صفحات وب اجرا شود.
- از اعتبار سنجی ورودی برای فرمان NSFFilemanage استفاده شود.
- از LibXML2 روی NSXMLparser برای جلوگیری از حمله تزریق XML استفاده شود.
- تزریق Format string: این حمله زمانی اتفاق می‌افتد که داده پذیرفته‌شده از یک ورودی رشته‌ای به‌عنوان یک دستور ارزیابی شود. توابع مختلفی از زبان C نسبت به این حمله آسیب‌پذیرند. ازجمله :

NSLog, [NSString stringWithFormat:], [NSString initWithFormat:], [NSMutableStringappend-Format:], [NSAlert informativeText-WithFormat:], [NSPredicate predicateWithFormat:], [NSException format:], NSRunAlert-Panel .

نباید به منابع خارج از کنترل ما اجازه داده شود تا با ارسال پیام‌ها و داده‌های کاربر از دیگر برنامه‌ها، بخش‌هایی از Format String را کنترل کنند.

- حملات به توابع C : معمولاً توابع قدیمی C آسیب‌پذیرند که باید از به کار بردن آن‌ها اجتناب شود. مانند: strcat, strcpy, strncat, strncpy, sprint, vsprintf, gets,...

در اندروید:

- هنگام سروکار داشتن با پرس‌وجوهای پویا و مؤلفه‌های "فراهم‌کننده محتوا" باید از پرس‌وجوهای پارامتری یا Prepared statement استفاده کرد.
- باید جاوا اسکریپت و پلاگین پشتیبان کننده آن به‌طور پیش‌فرض برای هر webview غیرفعال باشد.
- برای هر WebView دسترسی به سیستم فایل باید غیرفعال باشد.

```
webview.getSettings().setAllowFileAccess(false);
```

- برای همه مؤلفه‌های "فعالیت" باید داده‌ها و اقدامات با یک فیلتر اینتنت تأیید اعتبار شود.

جلوگیری در تزریق باینری: در ادامه به‌طور کامل تشریح می‌شود.

سناریوهای نمونه

اگر داده بازیابی شده از سرور برنامه شامل داده‌های مخرب باشد این داده‌ها به پایگاه داده محلی موبایل تزریق می‌شود که می‌تواند منجر به حمله تزریق SQL شود.

اینتنت های مخرب از دیگر برنامه‌ها می‌تواند منجر به سرریز بافر شود و درنتیجه کدهای مخرب اجرا شود.[۹]

تغییرات Html از طریق بدافزارها یا دیگر برنامه‌ها می‌تواند منجر به اجرای کد جاوا اسکریپت مخرب در لایه نمایش شود.[۱۰]

M8- تصمیم‌گیری‌های امنیتی بر اساس ورودی‌های نامعتبر

عوامل تهدید

کاربران و بدافزارها و برنامه‌های آسیب‌پذیر می‌توانند داده‌های نامعتبر را به متدهای حساس ارسال کنند.

نحوه حمله

سوءاستفاده از این آسیب‌پذیری آسان است؛ حمله‌کننده می‌تواند با دسترسی به یک برنامه فرمان‌ها را شنود کند و پارامترهای آن را تغییر دهد.

علت ضعف امنیتی

معمولاً برنامه‌نویسان برای تمایز بین کاربران سطح بالا و پایین از پارامترها و مقادیر پنهان و عملکردهای پنهان استفاده می‌کنند. پیاده‌سازی ضعیف این عملکردها امکان شنود و تغییر آن‌ها توسط حمله‌کننده‌ها را در پی دارد که موجب رفتار نامناسب برنامه و حتی اعطای مجوزهای بیشتر به حمله‌کننده می‌شود.

تأثیرات فنی

این آسیب‌پذیری باعث ارتقای سطح دسترسی برای حمله‌کننده می‌شود و حتی می‌تواند مکانیسم‌های امنیتی پیاده‌سازی شده در برنامه را دور بزند که باعث از بین رفتن قابلیت اعتماد و صحت می‌شود.

تأثیرات تجاری

این آسیب‌پذیری موجب از دست رفتن اعتبار می‌شود و همچنین صحت و قابلیت اعتماد را از بین می‌برد.

تشخیص آسیب‌پذیری

به‌طورکلی برنامه‌ها می‌توانند داده‌ها را از منابع مختلف دریافت کنند و در بیشتر موارد از مکانیسم ارتباط بین فرایندی (IPC) برای دریافت داده استفاده می‌کنند.

به‌طورکلی به الگوهای طراحی IPC زیر باید پایبند بود:

- اگر نیازمندی‌های تجاری برای ارتباطات IPC وجود داشته باشد برنامه باید با ایجاد یک لیست سفید این ارتباطات را محدود کنند.
- کنش کاربر برای انجام هرگونه اقدام حساسی که از طریق رابط‌های IPC فعال می‌شوند ضروری است.

---

[۹] CrossApplication Scripting Attacks

[۱۰] CrossSite Script Attacks

- برای جلوگیری از حملات مبتنی بر ورودی باید ورودی‌های دریافت شده از رابط‌های IPC اعتبارسنجی شوند.
- به دلیل خطر شنود اطلاعات حساس توسط برنامه‌های دیگر، به‌هیچ‌وجه نباید این داده‌ها از طریق مکانیسم‌های IPC ارسال شوند.

جلوگیری از آسیب‌پذیری

در IOS:

نباید از متد handleOpenURL برای کار با URLها استفاده شود زیرا آرگومان BundleID برنامه مبدأ را شامل نمی‌شود. در عوض از متد openURL:sourceApplication:annotation استفاده شود و آرگومان sourceApplication با یک لیست سفید برنامه معتبر بررسی شود.

از iOS Pasteboard استفاده نشود زیرا توسط سایر برنامه‌های غیر معتبر قابل‌خواندن و مقداردهی است.

M9- اداره نادرست نشست

عوامل تهدید

هر کاربر یا برنامه‌ای می‌تواند به ترافیک شبکه و کوکی‌ها و ... دسترسی داشته باشد.

نحوه حمله

یک متخاصم با دسترسی فیزیکی یا یک بدافزار می‌تواند ترافیک شبکه را ثبت کند.

علت ضعف امنیتی

به‌منظور تسهیل در تراکنش پایدار بین برنامه و سرور پشتیبان ، برنامه‌ها از کوکی نشست استفاده می‌کنند که وضعیت را بر روی پروتکل‌های ناپایداری همچون HTTPS و SOAP حفظ می‌کند. برای حفظ وضعیت سرور پشتیبان پس از احراز هویت کاربر برنامه، یک کوکی نشست برای برنامه ارسال می‌کند تا در ارتباطات بعدی و تراکنش‌های سرویس از این کوکی استفاده کند و این به سرور اجازه می‌دهد هر درخواست سرویس از سمت برنامه را به‌راحتی احراز هویت کند و مجوزهای لازم را تجویز کند. اداره نادرست نشست زمانی اتفاق می‌افتد که توکن نشست با یک متخاصم به اشتراک گذارده شود.

تأثیرات فنی

یک متخاصم با دسترسی به توکن های نشست‌ها می‌تواند هویت کاربر را جعل کند و با ارسال آن به سمت سرور پشتیبان، یک سرویس حساس را درخواست کند. بنابراین خطرات این آسیب‌پذیری به نوع کاربر جعل‌شده و سرویس‌های درخواستی آن بستگی دارد. متخاصم در بدترین حالت با جعل هویت کاربر مدیر می‌تواند قابلیت‌هایی حساسی را درخواست کند و در حالت معمولی کاربران کنترل حساب خود را از دست می‌دهند.

تأثیرات تجاری

صدمات تجاری ناشی از سوءاستفاده از این آسیب‌پذیری می‌تواند شامل کلاه‌برداری، سرقت اطلاعات و توقف کسب‌وکار باشد.

تشخیص آسیب‌پذیری

معمولاً نتایج این آسیب‌پذیری مشابه آسیب‌پذیری احراز هویت ضعیف است. کاربر برنامه فقط یک‌بار در طول یک نشست و آن‌هم در ابتدای آن احراز هویت می‌شود بنابراین کد برنامه باید به‌دقت از نشست‌های کاربر محافظت کند.

در زیر به نمونه‌هایی از نحوه اداره نامناسب یک نشست اشاره‌شده است:

- غفلت از باطل کردن نشست‌ها در سمت سرور: برنامه‌نویسان معمولاً نشست‌ها را در سمت موبایل ابطال می‌کنند و درحالی‌که در سمت سرور آن نشست برقرار است و این موجب سوءاستفاده متخاصمین از این موقعیت با کمک ابزارهای دست‌کاری HTTP می‌شود.

- عدم محافظت با مهلت زمانی مناسب: برنامه‌ها می‌توانند با تعیین مهلت زمانی برای نشست‌ها، از آن‌ها در برابر سوءاستفاده متخاصمین محافظت کنند بنابراین متخاصمین دیگر نمی‌توانند با دسترسی به یک نشست قدیمی، هویت کاربر آن نشست را جعل کنند. این مهلت زمانی با توجه به حساسیت برنامه و مشخصات خطر متعلق به آن برنامه و ماهیت ارتباطی کاربر با برنامه تعیین می‌شود معمولاً کاربران یک‌دفعه کارهای زیادی از برنامه‌های موبایلی خود می‌خواهند و به‌علاوه وقفه‌ها بین ارتباطات کاربر با برنامه زیاد است ازاین‌رو این دلایل پیش‌بینی مهلت زمانی را نسبت به برنامه‌های وب سخت‌تر می‌کند و تعیین مهلت زمانی طولانی‌تر خطر دزدیدن نشست را بیشتر می‌کند؛ بااین‌حال معمولاً این مهلت زمانی، زمان‌های ۱۵ دقیقه یا ۳۰ دقیقه یا یک ساعت است.

- غفلت از تعویض کوکی‌ها: در طول تغییرات وضعیت احراز هویت باید کوکی‌ها به‌طور مناسب باز تنظیم شوند. وقایعی موجب تغییر وضعیت احراز هویت می‌شوند که ازجمله تعویض از یک کاربر گمنام به یک کاربر واردشده یا تعویض از هر کاربر واردشده به کاربر واردشده دیگر یا تعویض از یک کاربر معمولی به یک کاربر ویژه یا مهلت‌های زمانی هستند. برای هرکدام از وقایع گفته‌شده نشست‌ها در سمت سرور باطل می‌شوند و دیگر کوکی‌های آن نشست‌ها نباید پذیرفته شود. در حالت ایده آل باید برنامه این‌گونه از کوکی‌ها را تشخیص دهد.

- ساخت توکن غیر ایمن: تولید توکن های مناسب سخت است. برنامه‌نویسان باید از الگوریتم‌های رمزنگاری و روش‌های استاندارد آزموده شده برای ایجاد توکن استفاده کنند به‌طوری‌که به دلیل پیچیده و طولانی و شبه تصادفی بودن آن قابل حدس زدن نباشد.

جلوگیری از آسیب‌پذیری

باید مطمئن شد که برنامه در طول چرخه حیات یک نشست متعلق به آن برنامه توکن های نشست را به‌طور مناسب ایجاد ، حفظ و ابطال می‌کند.

M10- نبود حفاظت‌های باینری

عوامل تهدید

معمولاً یک متخاصم کد برنامه را تجزیه‌وتحلیل و مهندسی معکوس می‌کند و سپس با انجام تغییراتی در آن موجب اجرای برخی قابلیت‌های پنهان می‌شود.

نحوه حمله

تجزیه‌وتحلیل و مهندسی معکوس معمولاً توسط ابزارهای خودکار انجام می‌گردد.

علت ضعف امنیتی

عدم محافظت باینری می‌تواند برنامه و دارنده آن را در معرض خطرات فنی و تجاری بسیاری قرار دهد. البته یک برنامه با محافظت باینری نیز می‌تواند مهندسی معکوس شود و در معرض خطر باشد ولی این محافظت باینری روند عملیات را کند می‌کند. معمولاً برنامه‌ها بدون محافظت باینری ایجادشده‌اند. تشخیص مهندسی معکوس کد یک برنامه توسط متخاصم دشوار است و معمولاً زمانی مالک برنامه می‌فهمد که کد برنامه‌اش در برنامه دیگر به‌کاررفته باشد و این نحوه تشخیص بسیار تصادفی است. همچنین برنامه باید در صورت بروز تغییرات و تزریق در زمان اجرا و پاسخ تشخیص دهد و با روش‌های مختلف واکنش نشان دهد که این واکنش‌های از پیش تعریف‌شده می‌تواند یا تلاش برای خنثی کردن حمله یا شکست حمله با روش ماهرانه باشد.

تأثیرات فنی

عمده برنامه‌های موبایل از حفاظت باینری در برابر مهندسی معکوس و تغییرات در کد باینری بی‌بهره‌اند و توسعه‌دهندگان باید برای مقابله با این موارد حفاظت باینری را در برنامه‌ها بگنجانند.

این نوع محافظت می‌تواند روند مهندسی معکوس را دچار تأخیر کند ولی نمی‌تواند از انجام این حمله جلوگیری کند. در بیشتر موارد متخاصم کد برنامه را می‌دزدد و درحالی‌که مالک برنامه بی‌خبر است در یک برنامه دیگر به کار می‌برد و آن برنامه را در اَپ استورها به فروش می‌گذارد.

همچنین این حفاظت می‌تواند روند تغییر کد باینری برنامه برای اجرا یا غیرفعال کردن قابلیت‌های برنامه را کند کند. معمولاً این تغییرات در برنامه‌ای محتمل است که اطلاعات حساس مانند رمزها و کارت‌های اعتباری را ذخیره و پردازش می‌کنند. تغییر کد معمولاً نوعی از بازبسته بندی یا انضمام بدافزار به برنامه است.

تأثیرات تجاری

نبود حفاظت باینری موجب برخی نتایج درزمینهٔ تجارت می‌شود که ازجمله : سرقت اطلاعات محرمانه ، کلاه‌برداری و دسترسی غیرمجاز ، صدمه به اعتبار ، دزدی و از دست رفتن درآمد، سرقت مالکیت معنوی

تشخیص آسیب‌پذیری

درصورتی‌که کد برنامه در یک محیط غیرقابل‌اعتماد میزبانی شود آن برنامه در معرض خطر است. محیط غیرقابل‌اعتماد محیطی است که سازمان توسعه‌دهنده، دسترسی فیزیکی به آن نداشته باشد مانند: موبایل‌ها، ابرها، مراکز داده و ... .

در موارد زیر می‌توان به آسیب‌پذیر بودن برنامه را پی برد:

- درصورتی‌که بتوان یک برنامه را با ابزارهای آزاد رمزگشایی کرد برای نمونه با ابزارهای ClutchMod یا به‌طور دستی با GDB می‌توان برنامه‌های IPhone را رمزگشایی کرد.
- اگر با کمک ابزارهایی نظیر IDA Pro و Hopper بتوان کنترل جریان برنامه را ترسیم نمود و شبه برنامه را استخراج کرد.
- اگر بتوان لایه ارائه برنامه (html,css,..) در داخل تلفن تغییر داد و جاوا اسکریپت دلخواه را اجرا نمود.
- اگر با کمک یک ابزار ویرایش hex بتوان کد باینری برنامه را تغییر داد و کنترل‌های امنیتی را دور زد.

جلوگیری از آسیب‌پذیری

برای جلوگیری باید از مؤلفه‌های امنیتی برای بررسی موارد زیر در برنامه استفاده شود:

- کنترل تشخیص شکسته شدن قفل سیستم‌عامل موبایل
- کنترل checksum
- کنترل‌های گواهی‌نامه
- کنترل تشخیص عیب‌یابی

برنامه باید بتواند با مؤلفه‌های بالا دو خطر عمده را کاهش دهد:

- جلوگیری از تجزیه‌وتحلیل و مهندسی معکوس برنامه با کمک روش‌های تجزیه‌وتحلیل ایستا و پویا توسط متخاصم
- تشخیص تغییرات کد و یا کدهای اضافه‌شده در زمان اجرا و واکنش نشان دادن نسبت به این نقض صحت کد.

سناریوهای نمونه

در اینجا به برخی آسیب‌پذیری‌ها مربوط به عدم محافظت باینری در برنامه‌ها و ابزارهای تشخیص آن‌ها اشاره‌شده است:

در IOS: غیرفعال کردن رمزنگاری کد(ClutchMod) ، گریز از تشخیص قفل‌شکسته بودن سیستم‌عامل(xcon)، نسخه‌برداری از کلاس(classdumpz) ، تغییر قابلیت یک متد با تعویض پیاده‌سازی آن با متد دیگر در زمان اجرا(Mobile Substrate)، تزریق کد در زمان اجرا(cycript)،نظارت در زمان اجرا(SnoopIt)، تجزیه‌وتحلیل در زمان اجرا(GDB) ، مهندسی معکوس(IDA Pro; Hopper)

در اندروید: تبدیل بایت کد(apktool; dex2jar)، تجزیه‌وتحلیل در زمان اجرا(ADB) ، مهندسی معکوس(IDA Pro; Hopper) ، تبدیل به اسمبلی (baksmali))، تزریق کد(Mobile Substrate)

منابع

1. Adventures with Android WebViews
2. An In Depth Introduction to the Android Permissions Modeland How to Secure MultiComponent Applications
3. Apple's Introduction to Secure Coding
4. Apple’s Secure Coding Guide
5. Fortify On Demand Blog - Exploring The OWASP Mobile Top 10: Insecure Data Storage
6. Fortify On Demand Blog - Exploring The OWASP Mobile Top 10: Insufficient Transport Layer Protection
7. Fortify VulnCat - A Taxonimy of Software Security Errors
8. Fortify Software: Format Strings - Is Objective C Objectively Safer?
9. Google Androids Developer Security Topics 1
10. Google Androids Developer Security Topics 2
11. Youtube: Ilja Van Sprundel – Auditing iPhone and iPad Applications
12. OWASP: *IOS Developer Cheat Sheet*. https://www.owasp.org/index.php/IOS_Developer_Cheat_Sheet
13. SSL Pinning for Cocoa Touch - Blog.securemacprogramming.com
14. Fahl, Sascha, et al. "Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security." *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012.
15. Arxan Research: State of Security in the App Economy, Volume 2, November 2013:
16. HP Research: HP Research Reveals Nine out of 10 Mobile Applications Vulnerable to Attack, 18 November 2013:
17. North Carolina State University: Dissecting Android Malware: Characterization and Evolution, 7 September 2011:
18. Tech Hive: Apple Pulls Ripoff Apps from its Walled GardenFeb 4th, 2012:
19. Tech Crunch: Developer Spams Google Play With RipOffs of Well-Known Apps… Again, January 2 2014:
20. Extreme Tech: Chinese App Store Offers Pirated iOS Apps Without the Need To Jailbreak, April 19 2013:
21. OWASP: Architectural Principles That Prevent Code Modification or Reverse Engineering, January 11th 2014.
22. Gartner report: Avoiding Mobile App Development Security Pitfalls, 24 May 2013:
23. Gartner report: Emerging Technology Analysis: Mobile Application Shielding, March 26th, 2013:
24. Gartner report: Proliferating Mobile Transaction Attack Vectors and What to Do About Them, March 1st, 2013:
25. Gartner report: Select a Secure Mobile Wallet for Proximity, March 1st, 2013:
26. Forrester paper: Choose The Right Mobile Development Solutions For Your Organization, May 6th 2013:
27. John Wiley and Sons, Inc: iOS Hacker's Handbook, Published May 2012, ISBN 1118204123.
28. McGraw Hill Education: Mobile Hacking Exposed, Published July 2013, ISBN 0071817018.
29. Publisher Unannounced: Android Hacker's Handbook, To Be Published April 2014.
30. Software Development Times: More than 5,000 apps in the Google Play Store are copied APKs, or 'thief-ware', November 20 2013:
31. InfoSecurity Magazine: Two Thirds of Personal Banking Apps Found Full of Vulnerabilities, January 3 2014:
32. InfoSecurity Magazine: Mobile Malware Infects Millions; LTE Spurs Growth, January 29 2014: