عیب پایان نامه مورد بررسی: کلمات اختصاری نداشت خودم استخراج کردم

کلمات اختصاری

محیط اجرایی قابل اعتماد (**TEE**)

Trusted Execution Environment

sensitive program information (SPI)

اطلاعات برنامه حساس

برنامه های معتمدTrusted Applications **(TA)**

برنامه نویسی شی گرا object-oriented programming **(OOP)**

رویکردهای پردازش زبان طبیعی**Natural language processing (NLP)**

دستگاههای بردار پشتیبانی ساختار یافته (**SSVM**) **structured support vector machines**

variable usage semantics analysis (VUSA) تجزیه و تحلیل معنایی کاربرد متغیر

پایگاه محاسباتی مورد اطمینان**trusted computing base (TCB)**

trusted computing base (TCB).

پایگاه محاسبات مورد اعتماد

یادگیری ماشین**machine learning (ML)**

deep learning (DL), and natural language

processing (NLP)

یادگیری عمیق **(DL)** و زبان طبیعی پردازش**(NLP)**

، نمودار عملکرد قابل تقسیمpartitionable function graph (PFG)

متدولوژی ها ، تکنیک ها و ابزارهای فهم و مدیریت اطلاعات برنامه های حساس

ییین لیو

پایان نامه ارائه شده به دانشکده موسسه پلی تکنیک ویرجینیا و دانشگاه ایالتی

تز دکترای مهندسی کامپیوتر

۱۱مه ۲۰۲۱

بلکسبورگ ، ویرجینیا

کلیدواژه ها:

مهندسی نرم افزار ، تحلیل و تحویل برنامه ، فهم برنامه، محیط اجرای مورد اعتماد ، **Middleware**

چکیده :

تجزیه و تحلیل، الگوریتم ها و داده های تجاری خاص می تواند به امنیت و حریم خصوصی سازمان و کاربران نهایی آسیب برساند. اطلاعات برنامه حساس **(SPI)** ، اجزای سازنده جزئی از نرم افزارهای مدرن سیستم هایی در حوزه های مختلف از برنامه های سازمانی تا تنظیمات سایبری فیزیکی هستند. از این رو ، حفاظت از **SPI** به یکی از چالش های برجسته توسعه نرم افزار مدرن تبدیل شده است .با این حال ، چندین مانع اساسی بر سر راه حفاظت **SPI** قرار دارد

(۱) : درک و مکان یابی **SPI** برای هر پایگاه داده ای با اندازه واقعی دشوار است

(۲) .جداسازی **SPI** برای محافظت از آن سنگین و مستعد خطا است.

(۳) **SPI** در بین اجزای توزیع شده در داخل و بین دستگاه ها ، در برابر امنیت و حملات به حریم خصوصی آسیب پذیر می شود. برای پرداختن به این مشکلات ، این پایان نامه در حوزه تجزیه و تحلیل برنامه های خودکار ، تبدیل کد و برنامه نویسی جدید برای بهبود وضعیت هنر در حفاظت از **SPI** به طور خاص نوآوری هایی را ارائه می دهد.

این پایان نامه شامل سه مورد تحقیق مرتبط است که:

(۱) طراحی و توسعه تجزیه و تحلیل برنامه و پشتیبانی برنامه نویسی برای استنباط معناشناسی استفاده از سازه های برنامه ، با هدف کمک به توسعه دهندگان برای درک و شناسایی **SPI**

(۲) برنامه نویسی قدرتمند و ابزارهایی که به طور خودکار کد را تغییر می دهند تا با هدف کمک به توسعه دهندگان به طور موثر **SPI** را از بقیه پایگاه کد جدا کند.

(۳) ارائه مکانیسم برنامه نویسی برای محیط های اجرایی مدیریت شده توزیع شده که **SPI** را پنهان می کند ، با هدف فعال کردن اجزای برای تبادل ایمن **SPI** .

متدولوژی ها ، تکنیکها و ابزارهای نرم افزاری ، پشتیبانی از برنامه نویسی ، تجزیه و تحلیل خودکار برنامه و تحول کد این پژوهش پایان نامه ، زمینه را برای ایجاد ایمن ، قابل درک و کارآمد برای محافظت از **SPI** فراهم می کند .

این پایان نامه بر اساس ۴ مقاله کنفرانسی ارائه شده در **TrustCom'۲۰** ، **GPCE'۲۰** ، **GPCE'۱۸**و **ManLang'۱۷**و همچنین ۱ مقاله مجله ای که در مجله زبان های کامپیوتری **COLA** منتشر شده است.

خلاصه بیانیه عمومی

برخی از بخشهای یک برنامه کامپیوتری می توانند حساس باشند ، که به آنها اطلاعات برنامه حساس گفته می شود **(SPI)**. با به خطر انداختن **SPI** ، مهاجمان می توانند به امنیت و حریم خصوصی کاربران آسیب برسانند. برای توسعه دهندگان،شناسایی و محافظت از **SPI** ، به ویژه برای برنامه های بزرگ دشوار است . این پایان نامه روش ها ، تکنیک ها و ابزارهای نرم افزاری جدیدی که نرم افزار را تسهیل می کند و وظایف مربوط به مکان یابی و حفاظت از**SPI** را معرفی می کند.

تقدیم به

خانواده ام تشکر که همیشه با من هستید

فصل ۱ :

مقدمه

"من واقعاً فکر می کنم اگر ما رویکرد خود را تغییر دهیم و به آنچه در دسترس داریم فکر کنیم ، این چیزی است که توانایی ما را برای برتری واقعی در امنیت افزایش می دهد. " گرگ یورک

۱-۱ بیان مسئله :

سیستم های نرم افزاری مدرن عموما منطق تجاری حساس ، الگوریتم ها و داده هایی که حذف یا دستکاری آنها به کاربران نهایی یا کل سازمان ها آسیب می رساند؛ را با هم ترکیب می کنند. بلوک های سازنده نرم افزار در مجموع اطلاعات حساس برنامه **(SPI)**نامیده می شوند .نشت داده های حساس می تواند امنیت و حریم خصوصی کاربر را به خطر بیندازد. مثلا، پس از سرقت رمز عبور کاربر در بانکداری آنلاین ، مهاجم می تواند وارد حساب قربانی سیستم شود و وجوه را غیرقانونی انتقال دهید. وجود کسب و کارهای حساس می تواند رقابت یک شرکت را تضعیف کند. به عنوان مثال ، داشتن مهندسی معکوس منطق تجاری یک موتور توصیه رایج ، یک مهاجم می تواند به سرعت، اطلاعات را از بین ببرد و باعث اختلال یا حتی خرابی سیستم شود. به عنوان مثال ، با دستکاری در الگوریتم ناوبری هواپیماهای بدون سرنشین ، یک دشمن می تواند پهپاد را به تغییر مسیر اشتباه دهد. **SPI** حفاظت یک منطقه ثابت در تحقیقات امنیتی است ، زمان مهندسی نرم افزار رسیده است محققان بایستند و از دید توسعه دهندگان در مورد برجسته ترین آنها تجدید نظر کنند.

از دیدگاه توسعه دهنده ، **SPI**شامل متغیرهای برنامه است که داده ها را به عنوان توابع که منطق و الگوریتم های کسب و کار را پیاده سازی می کنند؛ ذخیره می کند. بنابراین ، از **SPI** با وظایف توسعه ای که از صداقت و محرمانه بودن متغیرهای حساس و کارکرد اطمینان دارند؛ محافظت می شود .فرض کنید آلیس یک توسعه دهنده نرم افزار است که وظیفه حفاظت از متغیرهای حساس را بر عهده دارد و در سیستم موجود کار می کند .برای این منظور ، او به راه حل های عملی و موثر نیاز دارد که به سوالات زیر پاسخ قطعی می دهد

(۱)چگونه می توان متغیرهایی را ذخیره کرد که **SPI** را ذخیره می کنند؟

(۲)چگونه می توان از این متغیرها و توابع آنها محافظت کرد؟

(۳)چگونه می توان سیستم ها را برای تبادل ایمن **SPI** فعال کرد؟

سناریوهای انگیزشی

سناریو ۱

فرض کنید که آلیس قصد دارد مشخص کند کدام مورد می تواند بر موقعیت جغرافیایی پهپاد تأثیر بگذارد؛ برای این منظور ، آلیس می تواند با جستجوی متغیرها شبیه به کلمه **"GPS"** شروع کند . متغیرهای موقعیت مکانی **GPS** اغلب به طور قابل توجهی متفاوت هستند. با یک جستجوی ساده در **GitHub** در مورد نامهای زیر مشخص می شود.:

"gps" ۱۶ میلیون مورد ؛ **"geo"** ۴۳ میلیون مورد ؛ "موقعیت جغرافیایی" ۷ میلیون مورد و "عرض جغرافیایی" ۵۲ میلیون مورد.

ممکن است یک توسعه دهنده، نام متغیر موقعیت مکانی با یک حرف **"g"** یا حتی یک نام غیر مرتبط **"abc"** نمایش دهد . از این رو ، تطبیق رشته به تنهایی در شناسایی همه نامهای متغیرهای موقعیت جغرافیایی بالقوه ناکافی است.

برای حل این مشکل ، ما یک تجزیه و تحلیل برنامه جدید ارائه می دهیم که میزان استفاده از یک متغیر را از معناشناسی از اطلاعات متنی و زمینه ای آن (به عنوان مثال ، نام نمادین ، نوع ، محدوده ، اطلاعات جریان) را مشخص می کند. برای پشتیبانی از این تجزیه و تحلیل ؛ **VarSem**، یک زبان مخصوص ، که در آن مقوله معنایی یک متغیر به عنوان مجموعه ای از قوانین اعلانی بیان می شود؛ را معرفی می کنیم. به طور مشخص، با استفاده از **VarSem** ، آلیس می تواند قوانینی را تعریف کند که آن متغیرها را مشخص کند.که در پیاده سازی عملکردهای مرتبط با موقعیت جغرافیایی استفاده می شود. سپس ، قوانین سطح بالا به سطح پایین پردازش زبان طبیعی و جریان داده تبدیل می شوند تجزیه و تحلیل ، که احتمال هر متغیر برنامه مربوط به عملکرد را محاسبه می کند ؛ یعنی احتمالات محاسبه شده نشان می دهد که آیا از یک متغیر توسط برخی از قابلیت های مربوط به موقعیت جغرافیایی ، استفاده شده است یا خیر . بخشی از اطلاعاتی است که می تواند به آلیس در شناسایی متغیرهای حساس کمک کند .

سناریوی ۲

فرض کنید که آلیس به نحوی موفق شده است موقعیت مکانی را به درستی شناسایی کند ، او باید این متغیرها و توابع وابسته به آنها را در یک محیط اجرایی ایمن که متغیرها و عملکردهای حساس در معرض خطر را بدون ابزارهای توسعه کافی محافظت کند؛

قرار دهد. این کار می تواند بسیار سنگین و مستعد خطا باشد . برای حل این مشکل ، **RT-Trust** را ارائه می دهیم که می تواند به آلیس کمک کند: ۱) جداسازی موقعیت جغرافیایی مربوط متغیرها و توابع ؛ ۲) بازطراحی و تغییر مسیر فراخوانی توابع وابسته برقراری ارتباط با سازه های جدا شده ؛ ۳) بررسی کنید که سیستم بازطراحی شده همچنان ادامه دارد برای برآوردن الزامات اصلی (به عنوان مثال ، محدودیت های زمان واقعی). بعلاوه ، اگر آلیس برخی از توابع را در محیط اجرای امن قرار ندهید ، **TEE-DRUP**را به آن ارائه می دهیم که توابع نامناسب را شناسایی و منتقل می کند.

سناریوی ۳

فرض کنید که آلیس موفق شده است همه حساسیت های مربوطه به متغیرها و توابع جغرافیایی در محیط اجرای امن را به درستی قرار دهد.حالا ، او ممکن است نیاز به تبادل ایمن اطلاعات منطقه جغرافیایی به برنامه هایی که بر روی پلت فرم تلفن همراه اجرا می شوند و پهپاد را کنترل می کنند؛ داشته باشد. اگر یک برنامه غیرقابل اعتماد است دریافت یا دسترسی به موقعیت جغرافیایی ،با نشت داده ها رخ می دهد. در واقع ، در سیستم عامل های تلفن همراه مدرن (به عنوان مثال ، **Android**و**IOS**) نشت داده ها ، عموما توسط حملات امنیتی و حریم خصوصی که مستند شده است مورد سوء استفاده قرار می گیرد .

برای حل این مشکل ، ما **ObEx** ، مکانیسم برنامه نویسی برای پنهان کردن حساس ها را ارائه می دهیم داده ها ، که توسعه دهندگان را قادر می سازد تا با پشتیبانی از آمار از پیش تعریف شده ، از **SPI** به طور موثر برای پرس و جو در مورد داده های پنهان به طور خاص ،محافظت کنند آلیس می تواند اطلاعات جغرافیایی حساس را در یک شی **ObEx** ، قرار دهد. پیکربندی آن برای پشتیبانی از مجموعه ای از رابط های پرس و جو مورد نیاز و سیاستهای انقضا هنگامی که از یک میزبان به میزبان دیگر منتقل می شود ، شی **ObEx** آن را حفظ می کند و داده های حساس پنهان هستند ، بنابراین نمی توان آنها را مستقیماً خواند یا نوشت. با این حال ، مشتریان می توانند درخواست های پیکربندی شده توسط آلیس در برابر شی اجرا کنند. همه داده های حساس شی **ObEx** هستند با اتمام عمر مشخص شده ، به طور قابل اعتماد از بین می رود ، بنابراین از نشت بیشتر داده ها جلوگیری می کند**.**

مشارکت ها و برنامه های کاربردی

هدف اصلی این تحقیق پایان نامه ، ایجاد یک روش امن ، قابل درک ، و پایه ای کارآمد برای درک و مدیریت **SPI** است . به طور خاص ، این پایان نامه به سه طریق به این هدف کمک می کند**:**

(۱) طراحی و توسعه برنامه های پشتیبانی و برنامه نویسی برای استنباط استفاده معناشناسی اجزای برنامه ، با هدف کمک به توسعه دهندگان برای درک و شناسایی **SPI** یعنی تحلیل معنایی متغیر و **VarSem** ؛ **(۲)**برنامه نویسی و ارائه ابزارهای قدرتمندی که کد را به طور خودکار تغییر می دهد ، با هدف کمک به توسعه دهندگان به طور موثرتر **SPI** را از بقیه پایگاه کد جدا می کند. **(۳)**ارائه مکانیسم برنامه نویسی برای محیط های اجرایی توزیع شده مدیریت شده که ساختار **SPI** را پنهان می کند ، با این هدف که اجزاء بتوانند **SPI** را به طور ایمن مبادله کنند . به طور کلی ، ما مفاهیم جدیدی را در زمینه های تجزیه و تحلیل برنامه و بازآرایی (به عنوان مثال ، تحلیل معنایی متغیر ، و امکان سنجی و کاربرد آنها را نشان داد با پیاده سازی ابزارهای نرم افزاری کاربردی جدید (به عنوان مثال ، **VarSem**، **RT-Trust**، **TEE-DRUP**، .روش هایی را معرفی می کند که می تواند طراحی تجزیه و تحلیل تخصصی جدید را هدایت کند و تکنیک های بازسازی دومی می تواند به توسعه دهندگان نرم افزار که نقش های متفاوتی را ایفا می کنند ، کمک کند .برای برنامه نویسان تعمیر و نگهداری ، **VarSem**می تواند در شناسایی متغیرها با استفاده خاص در یک کد بزرگ کمک کند ؛ برای تحلیلگران امنیتی ، **RT-Trust**و **TEE-DRUP** می توانند در جداسازی کمک کنند عملکردهای حساس به **TEE** بدون متحمل شدن هزینه های غیر ضروری عملکرد ، در حالی که **ObEx** می تواند با خیال راحت داده های حساس را ذخیره و منتقل کند ، بنابراین از نشت جلوگیری می کند. برای زمان واقعی توسعه دهندگان سیستم ، **RT-Trust** می تواند تلاش دستی مورد نیاز برای سازگاری سیستم ها را کاهش دهد و اجرای قابل اعتماد تحت محدودیت های زمان واقعی را داشته باشد. علاوه بر این ، با استفاده از روشهای ما ، تکنیک ها و ابزارهای پروژه های خود را ، سازمان های توسعه نرم افزار می توانند بهبود بخشند. امنیت و حریم خصوصی محصولات آنها ، هدفی بسیار مهم برای همه افراد وابسته به نرم افزار محیط ها است.

۳-۱ ساختار

بقیه این پایان نامه به شرح زیر تنظیم شده است .فصل ۲ موارد فنی را معرفی می کند پیشینه و کار مربوط به این تحقیق را مورد بحث قرار می دهد .فصل ۳ مدل ها و مفروضات اصلی را توضیح می دهد فصل ۴ رویکرد ما را برای استنباط معناشناسی استفاده توضیح می دهد متغیرهای برنامه فصل ۵ و فصل ۶ رویکردهای جدا شده ما را شرح می دهد داده ها و توابع حساس در **TEE** و به ترتیب موارد غیر حساس را به خارج منتقل می کنند .فصل ۷رویکرد ما برای تبادل امن داده ها را توضیح می دهد .فصل ۸ و فصل ۹ به ترتیب کارها و نتیجه گیری را بیان میکند.

فصل ۲

پیش زمینه و کارهای وابسته

در این فصل ، ابتدا تعاریف و پیشینه فنی مورد نیاز برای فهم را معرفی می کنیم. در باره ی فعالیت های ما سپس در مورد کارهای مربوطه بحث می کنیم.

۲-۱ تعاریف و تکنیک های پس زمینه

در ادامه ما تعاریف مربوطه و فناوریهای اصلی که به رویکردهای ما قدرت می دهند را شرح می دهیم .

۲-۱-۲ تعاریف

اطلاعات برنامه حساس **SPI :(SPI)** می تواند شامل منطق تجاری ، الگوریتم ها ، و داده ها ، در صورتی که شامل اطلاعات حساسی باشند که تغییر آن ها به کاربران نهایی یا کل سازمان ها آسیب می رساند"**.**حساس" همه موارد مرتبط با امنیت را توصیف می کند مانند اشیاء (مانند گذرواژه ها ، کلیدها ، آدرس های حافظه) و عملیات (به عنوان مثال ، کنترل دسترسی ، رمزگذاری ، دسترسی به حافظه) . این اشیاء و عملیات با آنچه **SANS۱** از آن به عنوان "شرایط امنیتی" یاد می کند مطابقت دارد (۱۳۹) به طور خاص ، داده های حساس (یا متغیرها) اطلاعات مربوط به امنیت را ذخیره می کنند یا در رابطه با امنیت به آنها ارجاع داده می شود عملیات کد حساس (یا توابع) بر روی داده های حساس عمل می کند

اطلاعات اطلاعات محتوایی و متنی: خصوصیات به صورت روبرو تقسیم می شوند: داخلی (به عنوان مثال ، نام نمادین ، نوع و محدوده) و بیرونی (داده ها و جریان کنترل) .اطلاعات متنی به نام نمادین یک متغیر ، نام عملکرد ، نوع توسعه دهنده تعریف می شود نام (در صورت وجود) ، و مسیر فایل ؛ اطلاعات زمینه به همه خواص دیگر آن اشاره دارد. (به عنوان مثال ، نوع داده ، جریان داده کنترل) . توجه داشته باشید که نام نوع و نوع داده ، خواص مختلفی دارند. در حالی که اولی اطلاعاتی را که متنی هستند توصیف می کند و دومی یکی زمینه و بستر است. برای مثال ، نوع متغیر را در نظر بگیرید **.struct type_name :**نام نوع **"type_name"**و نوع داده **"struct"** است**.**

اطلاعات قابل شناسایی شخصی **PII :[۱۴۰] (PII)** شامل تمام اطلاعاتی است که در صورت دسترسی غیرمجاز به امنیت یا حریم شخصی افراد آسیب می رساند .نمونه های معمولی **PII** اجتماعی ، شماره های امنیتی ، شماره کارت اعتباری و بدهی و داده های مربوط به مراقبت های بهداشتی هستند.

محدودیت های زمان واقعی: به طور کلی ، محدودیت های زمان واقعی [۸۵] محدودیت های زمان بندی رویدادهایی هستید که باید توسط سیستم بلادرنگ ارضا شود .این محدودیت ها در مهلت های زمانی و محدودیت های دوره ای طبقه بندی شوند. (۱۰۵) به عنوان مثال ، با توجه به محدودیت دوره ای ۵۰ میلی ثانیه و مهلت زمانی ۲۰ میلی ثانیه ، یک کار هواپیمای بدون سرنشین باید مکان **GPS** خود در هر دوره را در ۲۰ میلی لیتر برای هر ۵۰ میلی ثانیه بدست آورد.

در این حالت ، به دلیل محدودیت حافظه ، مصرف حافظه رویداد یک محدودیت دیگر است همانطور که در بخش ۲,۱,۲ ذکر شد ، حافظه باید یک مقدار کوچک ردپایی با اشغال فضای محدود در حافظه را حفظ کند .

محدودیت های زمان واقعی به دو دسته سخت و نرم طبقه بندی می شوند تا محدودیت های قبلی برآورده شود. به عنوان مثال ، کنترل موتور و سطح پرواز هواپیمای بدون سرنشین باید انجام شود پاسخ به موقع (محدودیت سخت) ، در حالی که هدایت آن با توجه به نقاط مورد انتظار است مقاوم در برابر انحرافات ناشی از از بین رفتن موقت سیگنال **GPS** یا حتی وزش باد (محدودیت نرم)

۲-۱-۲ محیط اجرایی قابل اعتماد (**TEE**)

[۶۵] TEE یک راه حل سخت افزاری استاندارد ارائه می دهد که از **SPI** در برابر آسیب محافظت می کند .اولا ، **TEE**یک منطقه امن پردازنده (یعنی دنیای امن برای برنامه های قابل اعتماد) را از ناحیه عادی (یعنی دنیای عادی برای کاربردهای رایج) جدا می کند. یعنی ، دنیای امن دارای یک واحد محاسبه جداگانه و یک سیستم عامل مستقل است که مانع اجرای مستقیم برنامه های خارجی غیر مجاز امی شود. علاوه بر این، **TEE**فضای ذخیره سازی قابل اعتمادی را ارائه می دهد که تنها از طریق برنامه های ارائه شده می توان به طور ایمن به آن دسترسی پیدا کرد .در نهایت ، **TEE** به عنوان تنها یک برنامه به کانال ارتباطی به عنوان راهی برای ارتباط نهادهای خارجی با جهان امن ایمن ارائه می دهد .

OP-TEE [128]: OP-TEE نمونه کامل شده مشخصات پلتفرم جهانی **TEE** ، یک مکانیزم جداسازی سخت افزاری است که در درجه اول به منطقه امن ، با سه ویژگی های اساسی متکی است: ۱) برای محافظت از سیستم عامل مورد اعتماد از سیستم عامل

غنی (به عنوان مثال ، لینوکس) جدا می شود اجرای برنامه های معتمد **(TA)** از طریق پشتیبانی سخت افزاری اساسی انجام می شود ؛ **(**۲به حافظه کافی نیاز دارد. ؛ **(**۳می توان آن را به راحتی به انواع مختلف معماری و سخت افزار متصل کرد .

[۴۷**] SGX :**پیاده سازی دیگربا **TEE** افزونه های نرم افزاری اینتل است که با گسترش معماری اینتل از محرمانگی و امنیت داده ها محافظت می کند. همانند **OP-TEE** ، **SGX** از توسعه دهندگان می خواهد که کد اصلی را به دو قسمت تقسیم کنند قطعات معمولی و قابل اعتماد در داخل منطقه حفاظت شده قسمت اول که منابع اجرا را از محیط خارجی (قسمت دوم) جدا می کند. علاوه بر این ، اجزای معمولی فقط می توانند از طریق برنامه های ویژه به حوزه دسترسی داشته باشند. بنابراین ، در صورت اجرا یا بارگیری در داخل محوطه ، **SPI** در برابر حملاتی خارجی ، آسیب ناپذیر می شود**.**

۱-۲-۳ زبانهای برنامه نویسی و تجزیه و تحلیل

Scala، یک زبان برنامه نویسی مدرن ، ترکیبی از ویژگی های برنامه نویسی شی گرا و کاربردی است.(۸۳) **Scala** به عنوان زبان میزبان و به صورت تعبیه شده عمل می کند. **Scala** را به دلیل نحو انعطاف پذیر (به عنوان مثال ، پرانتزهای اختیاری) و پشتیبانی آن انتخاب می کنیم. تعریف کلمات کلیدی جدید و نحو سفارشی و ویژگی های برنامه نویسی کاربردی (به عنوان مثال ، پشتیبانی از توابع مرتبه بالاتر) از ویژگی های **Scala** است که امکان آن را فراهم می کند که کتابخانه به راحتی با تکنیک های تجزیه و تحلیل جدید گسترش می یابد **.**سرانجام ، به عنوان یک **JVMbased Scala** روی زبان های مختلف اجرا می شود و محیط مستقل از پلت فرم را ایجاد می کند.

کپسوله سازی شی: یکی از مفاهیم اساسی برنامه نویسی شی گرا **(OOP)** کپسوله سازی است که داده ها و رفتارهای حساس را از کاربران شی پنهان می کند **.**علاوه بر این ، جاوا اصلاح کننده های دسترسی را برای اطمینان از حریم خصوصی داده ها فراهم می کند **.**با استفاده از کلمه کلیدی برنامه نویسان انتظار دارند که این حوزه از خارج از حوزه خود قابل دسترسی نباشد. اعلام کلاس حفاظت ارائه شده توسط اصلاح کننده های دسترسی جاوا را می توان با مجوز مناسب بدست آورد.. مهاجم می تواند با دسترسی مستقیم، ویژگی های خصوصی را تغییر دهد و از تابع های خصوصی استفاده کند **.**برای جلوگیری از این حمله ، قابلیت مدیریت امنیت [۷۰] به جاوا اضافه شده است که اثربخشی آن به پیکربندی مناسب و واحد اجزا بستگی دارد.

چرخه حیات شی: وقتی شیء برنامه نویسی حاوی داده های حساس خارج محدوده ، در دسترس قرار می گیرد. اختیارات سیاستی رعایت می شود. اما در برخی موارد ، انتظار برای برقراری امنیت در حفظ اطلاعات حساس ممکن است کافی نباشد **.**در عوض ،

داده های حساس ممکن است پس از رسیدن به یک آستانه مشخص ، توسط سیاست دسترسی به هر شیء به طور قابل اعتماد پاک شوند.

رویکردهای پردازش زبان طبیعی **(NLP)** به طور گسترده ای برای شناسایی اطلاعات متنی حساس برنامه (به عنوان مثال ، نظرات ، توضیحات) استفاده شده است (۸۶ ، ۱۲۴ ، ۱۷۸) .به ابزارهایی متکی است تا تجزیه و تحلیل متنی جدیدی متکی بر **NLP** ارائه دهد که قدرتمندتر از تحلیل متغیرهایی که باید در **TEE** محافظت شوند؛ باشد.

تجزیه و تحلیل جریان داده ، یک تکنیک استاندارد تجزیه و تحلیل برنامه ها، برای نتیجه ها از طریق برنامه تجزیه و تحلیل جریان داده برای تشخیص آسیب پذیری های کد است (۴۱ ، ۹۹ ، ۱۳۸) .با استفاده از تجزیه و تحلیل استاندارد جریان داده ، برنامه متغیرها اطلاعات زمینه (به عنوان مثال مقادیر ورودی کاربر برای متغیر ها) را شناسایی می کند .

۲-۲- کارهای مرتبط

این پایان نامه مربوط به چندین حوزه تحقیقاتی ، از جمله درک معانی برنامه ، تغییر شکل کد و نشت اطلاعات ، در این قسمت در مورد آنها بحث خواهیم کرد.

۲-۲-۱ درک معنایی برنامه

برای درک تاثیر معنایی برنامه ، توسعه دهنده باید پاسخ هایی به سوالات زیر داشته باشد : (۱) استراتژی های پیشرفته درک مطلب چیست؟ (۲)از کدام ابزارهای نرم افزاری می توان برای تسهیل فرایند درک استفاده کرد؟ (۳) از کدام تکنیک می توان درک برای تشخیص اطلاعات مربوط به برنامه استفاده کرد؟ در ادامه در باره این سوالات ما بحث می کنیم.

استراتژی های درک معناشناسی: استراتژی های درک معنایی متکی بر تکنیک ها هستند. تکنیک ها را می توان در گروه های زیر طبقه بندی کرد:

الف) داده کاوی: وایمر و همکاران، کد و برنامه ای برای اشکال زدایی برنامه ها استخراج کردند [۱۶۴]. هاست و همکاران از داده کاوی برای استخراج قوانین از برنامه های کاربردی جاوا برای شناسایی نام توابع غیر معمول و غیرقابل قبول استفاده می کنند [۸۴].

(ب) تحلیل استاتیک: میشن و همکاران، جستجوی معنایی مبتنی بر تحلیل استاتیک را برای درک استفاده از **API** اعمال می کنند.(۱۲۱)

(ج) **ML/NLP**: آلامانیس و همکاران، از طریق مدل زبان و سبک برنامه نویسی برای توصیه نام شناسه و قراردادهای قالب بندی استفاده می کنند. [۲۱ ، ۲۲]. رایچف و همکاران از دستگاههای بردار پشتیبانی ساختار یافته (**SSVM**) برای پیش بینی نام و نوع شناسه ها برای اعلانات پروژه های جاوا اسکریپت استفاده می کنند. [۱۳۳]. آلون و همکاران از یادگیری عمیق برای یادگیری کد جاسازی و پیش بینی نام تابع ها استفاده می کنند. [۲۵]

(د) : الگوریتم رایس و همکاران تشخیص می دهد که آیا فراخوانی یک تابع پارامترهای صحیح را از نام یک شناسه ارسال می کند یا خیر[۱۳۵]. الگوریتم **Sridhara** و همکاران اسناد را برای روش های جاوا با انتخاب دستورات مهم کد و بیان آنها به عنوان زبان طبیعی نشان می دهد. [۱۴۴]. مدل توصیفی بوس و همکاران ، اندازه گیری خوانایی کد را مشخص می کند. [۳۸]

این راه حل ها برای کاربردهای خاص در نظر گرفته شده است و بر نظریه های متفاوت متکی است و الگوریتم ها آنها همچنین بر اطلاعات زمینه کد منبع ، تمرکز می کنند ، درک معنایی استراتژی - تجزیه و تحلیل معنایی کاربرد متغیر (**VUSA**) - هم متنی و هم زمینه ای اطلاعات برای استنباط معناشناسی متغیر را در نظر می گیرد.

ابزارهای کنونی برای حمایت از درک معناشناسی **:**مارتین و همکاران مطرح کردند که **PQL** یک زبان درخواست برنامه برای جستجوی الگوهای کد منبع خاص است. (۱۱۶). چن و همکاران مطرح کردند که **VFQL** یک زبان درخواست برنامه برای بیان و جستجوی نقص کد از طریق نمودارهای جریان مقدار است (۴۰) . اورما و همکاران **Wiggle**را توصیف می کنند که یک سیستم پرس و جو است که برای جستجوی کد منبع با ویژگی های متنی یا زمینه مشخص شده توسط کاربر ، از استفاده می شود (۱۵۵) کوهن و همکاران طراحی زبانی برای جستجوی الگوهای خاص در کد منبع جاوا را مطرح کردند (۴۶). برخی ابزارهای آنلاین و افزونه های می توانند الگوهای کد داده شده را پیدا کنند (۲۷ ، ۱۲۶). با این حال ، هیچ کدام این زبانها بر استنباط معناشناسی متغیر تمرکز نمی کنند.

درک معنایی برای تشخیص**SPI** : از آنجا که اطلاعات متنی داده های حساس ، می تواند افشا شود؛ خطرات احتمالی امنیت و حریم خصوصی را می توان با راه حل مناسب تشخیص داد. معناشناسی داده که به طور مستقل اجرا می شود؛ به طور خودکار ورودی کاربر حساس با استفاده از تکنیک های استخراج شده برای شناسایی کلمات کلیدی مشکوک شناسایی می کند(۸۶و۱۲۴)

۲-۲-۲ تغییر شکل کد برای اجرای مطمئن

برای محافظت از **SPI** ، توسعه دهنده باید کد برنامه را با سه مرحله تغییر شکل دهد مراحل اصلی: (۱) برنامه را به قسمتهای حساس و غیر حساس تقسیم کنید. ۲. تبدیل کد و داده های شناسایی شده را به برنامه ای مبتنی بر **TEE** برای اجرای مطمئن ، که پایگاه محاسباتی مورد اطمینان **(TCB)** کوچک باقی بماند **(۳) .**اطمینان حاصل کند که کد محدودیت های اجرا را حفظ می کند (به عنوان مثال ، محدودیت های زمان واقعی) . در ادامه درباره کارهای موجود مربوط به پارتیشن بندی برنامه ، تغییر کد و پروفایل اجرا و تأیید محدودیت ها بحث می کنیم.

برنامه های تقسیم بندی: کد بایت جاوا را در یک برنامه متمرکز به یک برنامه توزیع شده تقسیم می کند(۱۵۲) . با توجه به اعلانات برنامه نویس ،یک برنامه وب به یک برنامه وب امن تغییر می کند و در قسمت سمت سرور جاوا و قسمت جاوا اسکریپت سمت مشتری از طریق **HTTP** با یکدیگر ارتباط برقرار می کنند(۴۴). **ZØ** کامپایل اعلانی است که شامل کد **C#** یک برنامه متمرکز در نسخه توزیع شده چند سطحی برای افزایش محرمانگی است (۶۰) . با اجرای مکانیزم پویای کنترل جریان اطلاعات ، به طور خودکار و ایمن یک برنامه جاوا اسکریپت را به قسمت سرویس گیرنده و سرور تقسیم می کند(۸۰) . به طور خودکار برنامه های پشتیبانی شده از پایگاه داده را به سرور برنامه تقسیم می کند

تبدیل کد و داده ها به برنامه های مبتنی بر **TEE**

انتقال کد و داده ها به **TEE** ، امکان راه اندازی خودکار و غیر قابل حمل پارامترهای اشاره گر در ارتباطات **RPC** را فراهم می کند(۱۰۶) .سنیر و همکاران یک مجموعه ابزار ارائه دادند که پروتکل های امنیتی را به چندین پارتیشن جداگانه تقسیم می کند تا نیازهای امنیتی را برآورده کند **[**۱۴۱**]**روبینوف و همکاران از تجزیه و تحلیل برای پارتیشن بندی خودکار برنامه های اندروید برای اجرای قابل اعتماد ، استفاده می کرد (۱۲۶) به طور خودکار کد حساس را تشخیص داده و قطعات را برش می دهد [۱۷۳].

چارچوب تبدیل منبع به منبع لیند و همکاران زیر مجموعه هایی از آن را در برنامه های **C** برای استفاده ، استخراج می کند

فصل ۳

مدل های تهدید

بیشتر بخش فنی این پایان نامه مربوط به بهبود امنیت و حریم خصوصی است . در این زمینه تحقیقاتی ، سناریوهای خاصی تعریف می شود که شرح می دهد چگونه دشمنان می توانند علیه یک سیستم عمل کنند .برای افزایش امنیت سیستم و حریم خصوصی ، باید از این اقدامات خصمانه جلوگیری یا حداقل مانع آن شد .این پایان نامه شامل سه بخش عمده است: (۱) تجزیه و تحلیل

برنامه و پشتیبانی از نتیجه گیری معناشناسی استفاده از متغیرهای برنامه، **(۲)**ابزارهای توسعه برای جداسازی داده ها و عملکردهای حساس (۳)مکانیسم برنامه نویسی برای تبادل امن داده ها. در ادامه درباره این سه بخش بحث می کنیم .

رفتارهای مهاجمان

(۱)آنچه انتظار می رود مهاجمان نتوانند انجام دهند**:** رفتارهای مهاجمان **(۱)**آنچه انتظار می رود مهاجمان نتوانند انجام دهند**:** الف) ما فرض می کنیم مهاجمان قادر به تغییر کد منبع یا نمایش کدهای میانی نیستند ، تا تحلیل و تحول برنامه را مشکل همراه سازند. به طور خاص ، رویکرده ما در فرآیندهای تجزیه و تحلیل و تحول برنامه نمی تواند توسط مهاجمان به خطر بیفتد ب) فرض می کنیم مهاجمان نمی توانند اجرا را به خطر بیاندازند ، مهاجمان نمی توانند **JVM** را به خطر بیاندازند**.**

(۲)آنچه از مهاجمان انتظار می رود انجام دهند **:**مهاجمان ممکن است سعی کنند **SPI** را در یک سیستم معین به خطر بیاندازند. در واقع ، امکان نشت اطلاعات به شدت توسط توابع آسیب پذیر وجود دارد [۱ ، ۲ ، ۷] ، به ویژه عملکردهای پردازش داده های حساس افزایش می یابد به عنوان مثال ، با به خطر انداختن انتقال داده ها ، مهاجمان مکان های **GPS** فعلی را به شکل مخرب دریافت می کنند [۵]. علاوه بر این ، خودسرانه افشای عملکردهای حساس برای تعامل با بازیگران خارجی می تواند به طور غیرقانونی مورد سوء استفاده قرار گیرد ، که باعث حذف فایل [۴] یا افشای اعتبار [۳] می شود**.** با قرار دادن کد **SPI** واقعی در **TEE**، آنها از آسیب رساندن مهاجمان به **SPI**در سیستمهای بازسازی شده جلوگیری می کنند این راه حل ها را در فصل های ۴ ۵ و ۶ بحث خواهیم کرد ، به ترتیب **.**علاوه بر این ، مهاجمان ممکن است تلاش کنند تا فرآیندهای تبادل داده اطلاعات حساس را به منظور نشت کردن به خطر بیاندازند. به عنوان مثال ، به دلیل حمله داده های حساس زیادی مانند مکان **GPS** را در معرض دید عموم قرار می دهد. مکان کاربران است در وب سایت افشا می شود زیرا برنامه تلفن همراه از موقعیت جغرافیایی اشتباه استفاده می کند . داده های حساس را بدون کسب مجوز کاربر ارسال می کند بدتر این است که ، داده های حساس چه به صورت درست و چه نامناسب به اشتراک گذاشته شوند ، می توانند به طور مداوم ذخیره شوند **.**سپس مهاجمان می توانند از این داده های حساس برای ایجاد انواع حریم خصوصی بعدی استفاده کنند . علاوه بر این ، مهاجمان ممکن است سعی کنند اشیای جاوا حاوی اطلاعات حساس را در حافظه به خطر بیاندازند در واقع ، دسترسی های مخرب به اشیاء جاوا تعداد برنامه های زیادی را تهدید کرده است مانند کپسوله سازی اشیا. حملات مفصل ، به ویژه در محیط های توزیع شده، دسترسی خواندن و نوشتن به داده های حساس را مسدود می کند ، در حالی که توسعه دهندگان را قادر می سازد از آنها استفاده کنند. درباره ویژگی های آماری این داده ها در برنامه های کاربردی آنها در فصل ۷ در مورد آن بحث خواهیم کرد**.**

مفروضات

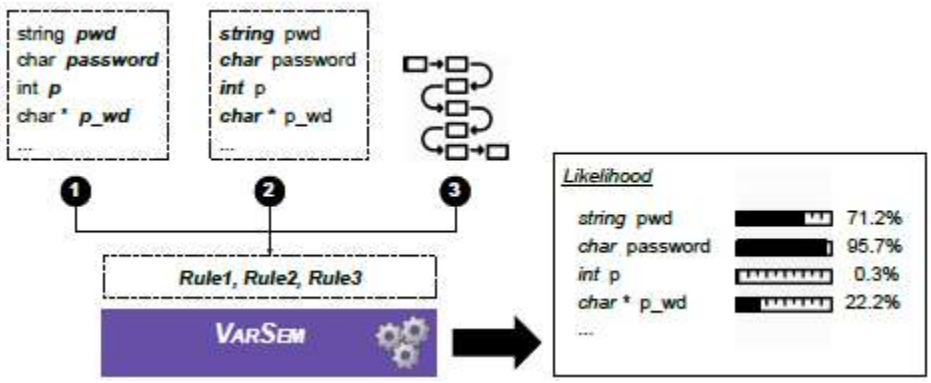
(۱فرض می کنیم که اکثر متغیرها ، توابع و فایل های موجود در برنامه به صورت معنی دار نامگذاری شده اند به عنوان مثال ، بسیار محتمل است که متغیری به نام "رمز عبور" برخی از اطلاعات مربوط به رمز عبور را نشان می دهد. در واقع ، شرکت های بزرگ فناوری اطلاعات ، از جمله گوگل ، **IBM**و مایکروسافت قراردادهای برنامه نویسی ایجاد کرده اند که نیاز به شناسه های برنامه دارد که به طور شهودی نامگذاری شود [۷۲ ، ۸۷ ، ۱۱۸]. مرور و تصحیح منظم کد، اغلب با پیشنهاداتی برای تغییر نام شناسه های معنادارتر همراه است.(۷۳) (۲ .ما فرض می کنیم که کاربران دارای سابقه کافی در مورد معماری سیستم های نرم افزاری مورد تجزیه و تحلیل برای توصیف متغیرهای هدف هستند. انتظار می رود که کاربران در مورد چگونگی عملکرد مورد علاقه اطلاعاتی داشته باشند. به عنوان مثال ، برای تعیین متغیرهایی که گذرواژه ها را ذخیره می کنند ، یک کاربر انتظار می رود که نحوه احراز هویت مبتنی بر رمز عبور در پروژه های تجزیه و تحلیل شده را درک کند .به عنوان مثال ، در یک عملکرد خاص "ورود" ، یک رمز عبور ، که توسط کاربر نهایی وارد می شود ، در مقایسه با رمز عبور شناخته شده ، از حافظه بازیابی شده است. متغیرها/توابع حساس و غیر حساس برای محافظت از داده های حساس را تشخیص دهدو بتواند زمان و نحوه کار مشتریان را تعیین کنند . از ویژگی های اشیاء داده حساس (به عنوان مثال ، مدت زمان دسترسی داده ها ، نحوه دسترسی) قابلیت پرس و جو است که بارها می توان ویژگی های آن را مورد پرسش قرار داد.

فصل ۴

معناشناسی متغیرهای برنامه

در یک برنامه کامپیوتری ، متغیرها برای اهداف پیاده سازی خاص معرفی می شوند. اهدافی مانند معناشناسی استفاده متغیر. درک معانی کاربرد متغیر برای اکثر کارهای نگهداری و تکامل برای حسابرسی کد و درک برنامه مورد نیاز است. (۶۴و۱۰۱) برای این منظور ، هر دو متن (طرح نامگذاری و اطلاعات زمینه و جریان داده) یک متغیر باید بررسی شوند تا کاربرد متغیر آشکار شود(۱۰۱) وظیفه شناسایی متغیرهایی را که رمزهای ورود کاربر را ذخیره می کنند ، در نظر بگیرید .وظیفه آن بررسی نحوه محافظت از رمزهای عبور در یک سیستم می باشد. همانطور که در شکل ۱ نشان داده شده است ، یک مهندس امنیت باید متغیرهایی را که نام آنها شبیه به است جستجو کند کلمه "رمز عبور" (به عنوان مثال ، **"pwd"**، **"passwd"**و **"pass_word"**) ، ۰ متغیرهایی را که نوعشان می تواند اطلاعات رمز عبور (به عنوان مثال ، رشته) را ذخیره کند و متغیرهایی را که زمینه آنهاست جستجو کند اطلاعات با الگوهای استفاده خاصی مطابقت دارد (به عنوان مثال ، اطلاعات از ورودی کاربر به متغیر منتقل می شود . این قوانین جستجو را می توان به صورت مستقل یا به صورت یک زنجیره منطقی اجرا کرد.

برای کار ۱ ، برنامه نویسان معمولاً از امکانات جستجوی کد منبع ، مانند یونیکس استفاده می کنند .به عنوان مثال ، صدور فرمان **grep "pass *word" *.c** در پوسته یونیکس ، متن با پیشوند **"pass"** و پسوند **"word"** موجود در فایل های منبع **C** مطابقت دارد ؛ را برمی گرداند. با این حال ، این روش کاملاً شکننده است.



شکل ۱ذخیره متغیرهای رمز کاربر

اسامی متغیرهای رمز عبور اغلب با کلمه "رمز عبور" تفاوت زیادی دارند .یک جستجوی ساده در **GitHub**نام های زیر را برای متغیرهای رمز عبور نشان می دهد**passwd" :** ۱۱ میلیون مورد ، **"p_wd"**۹۲۵۸ مورد ، **"pass_wd"** ۲۸۰ مورد و **"p_w_d"** ۱۶۴مورد .توسعه دهنده ممکن است یک متغیر رمز عبور را با یک حرف **"p"** یا **an** یا رشته بی ربط **"abc"** نامگذاری کند . بنابراین ، تطبیق رشته در توانایی آن کاملاً ناکافی است .در نتیجه نیاز به تلاش های اضافی دستی برای فیلتر کردن است.

وظایف ۱و۲ ، بازرسی دستی انواع متغیرها و جریان اطلاعات برای پروژه های کوچک می باشد ، اما برای پروژه های بزرگ غیر واقعی است .تجزیه و تحلیل مبتنی بر کامپایلر می تواند برای جستجوی انواع متغیرها و جریان اطلاعات به صورت خودکار استفاده شود .با این حال، استفاده صحیح از این ابزارها مستلزم آن است که برنامه نویسان تخصص کافی در کامپایلرها را کسب کنند و آن سطح انتظار از تخصص مربوط به همه برنامه نویسان برنامه یا مهندسان امنیت غیر واقعی است.

آخرین تحولات در یادگیری ماشین **(ML)** ، یادگیری عمیق **(DL)** و زبان طبیعی پردازش **(NLP)** ابزارهای جدیدی را برای تجزیه و تحلیل معانی کاربرد متغیر ارائه می دهد. با این حال، استفاده صحیح از این رویکردهای پیشرفته به دلیل موارد زیر کاملاً چالش برانگیز است دلایل: (الف) کاربردهای موجود این ابزارها با نام شناسه [۲۱ ، ۲۲ ، ۲۴ ، ۲۵] کاملاً متفاوت هستند .. به عبارت دیگر ، این رویکردها با توجه به مدلهای متفاوت متناسب با سناریوهای خاص خود، کاربرد دارند ، بنابراین هیچ مدل و تکنیکی

مخصوصی برای تجزیه و تحلیل معناشناسی کاربرد متغیر وجود ندارد. ب) دقت و فراخوانی بسیاری از این تکنیک ها برای کاربردهای متغیر عملی و وظایف تجزیه و تحلیل معناشناسی کافی نیست. به عنوان مثال توصیه می کنید نام شناسه های توصیفی استفاده شود. ج) اهداف این تکنیک ها معمولاً روی بخش هایی از کد کاربرد دارد و نه متغیرهای فردی. یعنی متنی و زمینه متغیرهای موجود اطلاعات به عنوان نشانه و ویژگی کل قطعه کد تلقی می شوند. با این حال، متغیرها اجزای برنامه متمایزی هستند. یک متغیر نه تنها اطلاعات را ذخیره می کند ، بلکه اطلاعات را نیز ذخیره می کند همچنین می تواند با سایر اجزای برنامه (به عنوان مثال ، خصوصیات کلاس ، پارامترهای تابع) و بخشی از هر زمینه برنامه (به عنوان مثال ، کنترل و جریان داده) باشد. انتخاب متغیرهایی (به عنوان مثال ، معانی و کاربرد آنها) تجزیه و تحلیل برنامه مشکلی دارد.

نتیجه گیری

این فصل تجزیه و تحلیل معانی کاربرد متغیر را ارائه کرده است ، **NLP.** دارای قواعد اعلانی و قابل تنظیم برای تجزیه و تحلیل روال برای استنباط اطلاعات متغیر و متنی است . که می تواند هم برای توسعه دهندگان و هم تحلیلگران امنیتی مفید باشد.

فصل ۵ جداسازی داده های حساس و توابع

اجرای سیستم های زمان واقعی حیاتی باید با محدودیت های زمان واقعی مطابقت داشته باشد .بسیاری از چنین سیستم هایی همچنین حاوی اطلاعات حساس برنامه **(SPI)** هستند**.** الگوریتم ها و داده ها باید محافظت شوند. عدم رعایت هریک از این الزامات می تواند منجر به عواقب فاجعه بار شود. استفاده از تحویل خودکار را در نظر بگیرید حمل بسته ها ، حاوی غذا ، آب ، دارو یا واکسن ، از راه دور و مکانهای سخت دسترسی پرسنل اورژانس حرفه ای هنگام مواجهه با بحران های مختلف است . ناوبری پهپاد دارای محدودیت های زمان واقعی است ؛ اگر نتواند دستورالعمل تنظیم خلبان خودکار را محاسبه کند جهت پرواز یا سرعت پرواز به موقع ، پهپاد ممکن است نتواند تنظیم شود که مسیر آن را به درستی تغییر دهد و از مسیر تحویل برنامه ریزی شده منحرف شود. از آنجا که بار اغلب باید تحت شرایط سخت زمانی تحویل داده شود و از کوتاهترین مسیر می تواند منحرف شود و باعث شکست کل ماموریت تحویل شود. علاوه بر این ، ماژول کنترل نرم افزار (به عنوان مثال ، ناوبری) اطلاعات حساس برنامه **(SPI)**را تشکیل می دهد. اگر یک عامل مزاحم ، کنترل عملکرد ماژول را مختل کند ، کل هواپیمای بدون سرنشین را می توان به اشتباه مسیریابی کرد و باعث شکست در تحویل شود.

با جداسازی توابع **SPI** در یک اجرای امن می توان آسیب پذیری از محیطی که تعاملات آنها را با جهان خارج نیز کنترل می کند را کاهش داد. به عنوان یک راه برای تحقق این ایده ، تولید کنندگان سخت افزار شروع به ارائه اجرای قابل اعتماد در محیط امن می کنند. پردازنده های خاص که می توانند برای اجرای وابسته به **SPI** برای عملکرد امن استفاده شوند. می تواند با اطمینان کد قابل اعتماد را از حالت عادی جدا کند ؛ محیط امن با سخت افزار قابل اعتماد ، ذخیره سازی و سیستم عامل ارتباطی ویژه تنها راه برای تعامل با کد مبتنی بر **TEE** است . برای حل مشکل سازش **TEE** ها ، جدا کردن **SPI** در امنیت محیط به طور موثر با حملات خصمانه مقابله می کند و از سرقت مالکیت معنوی جلوگیری می کند .با این حال ، برای بهره مندی از اجرای مطمئن ، سیستم ها باید طراحی و پیاده سازی مختلفی از **TEE** انجام شوند . سیستم های زمان واقعی برای استفاده از **TEE** نیاز به تغییر و تطبیق برنامه ها مستعد خطا دارند .

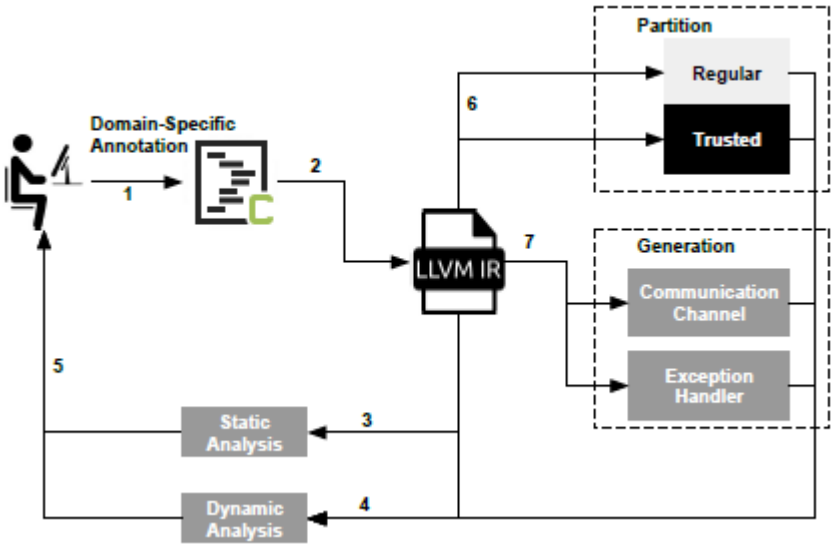
به طور خاص ، توسعه دهنده یک سیستم را تغییر می دهد تا از مزایای آن استفاده کند. ماژول **TEE** به انجام کارهای زیر نیاز دارد: ۱) جدا کردن کد وابسته به **SPI** ۲. هدایت فراخوانی توابع **SPI** به در محیط ارتباط **TEE** ؛ ۳) بررسی این که سیستم تغییر یافته همچنان با محدودیت های اصلی زمان واقعی مطابقت دارد. توجه کنید که همه انجام این کارها دشوار است .برای تکمیل وظیفه ۱) ، توسعه دهنده نه تنها باید به درستی کد وابسته به **SPI** را استخراج کند بلکه باید همه وابستگی ها را نیز به درستی شناسایی کند. به دلیل پتانسیل پیچیدگی این وابستگی ها ، برخی از کد های وابسته به **SPI** را نمی توان از **TEE** جدا کرد .

برای تکمیل وظیفه ۳) ، توسعه دهنده باید مایل به توسعه موارد آزمایشی دیگری باشد که می تواند بررسی کنید که آیا سیستم تبدیل شده محدودیت های اصلی زمان واقعی را برآورده می کند یا خیر. رویکردها از ابزارهای پروفایل ، از جمله ابزار Pin [۱۱۳] و gperftools [۷۷] استفاده می کنند ، که مستلزم آن است که پروفایل بهبود یابد. برای تسهیل فرایند تطبیق سیستم های زمان واقعی برای محافظت از کد وابسته به SPI آنها این مقاله با استفاده از TEE ، یک مجموعه ابزار تجزیه و تحلیل برنامه و ارائه می دهد که از توسعه دهندگان برای تقسیم بندی سیستم های زبان C در زمان واقعی پشتیبانی می کند. محدودیت ها. توسعه دهنده می تواند پیاده سازی TEE به عنوان یک گزینه کامپایلر ، یا برای تعیین خودکار موجود ، اعمال کند. پیاده سازی با تحلیل سیستم از طریق یک مدل برنامه نویسی با تک تک عملکردها برای جدا شدن در محیط امن اعمال شود ، بنابراین می توان آن را با موفقیت تقسیم کرد .در نهایت ، سیستم را به قسمتهای معمولی و قابل اعتماد ، تبدیل می کند. در صورت عدم موفقیت کد تبدیل شده برای برآوردن محدودیت های زمان واقعی ، محیط را اصلاح می کند.

برای ادامه ۴ حوزه را مطرح می کنیم : ۱. :یک مدل برنامه نویسی فراگیر برای تقسیم بندی سیستم های زمان واقعی برای استفاده از TEE ها که با زبان C نوشته شده است. این مدل به صورت خاص الزامات سناریوهای مختلف پارتیشن بندی را در بر می گیرد ۲. .مکانیزم های بررسی استاتیک و پویا که مشخص می کند آیا یک سیستم می تواند برای پیاده سازی TEE مشخص شده باشد یا خیر و احتمال تقسیم بندی آن چقدر است. هدف برآوردن محدودیت های اصلی در زمان واقعی است. مکانیزمی که به توسعه دهندگان اطلاع می دهد چگونه می توانند سیستم های خود را بازسازی کنند ، بنابراین آنها را می توان با موفقیت تقسیم کرد ۳. ،تغییر برنامه مبتنی بر کامپایلر برای برنامه های زبان C کار می کند ، در حالی که کانال های ارتباطی سفارشی را نیز ایجاد می کند و زمان واقعی ماسب را دارد ۴- .معیار مستقل از محیط برای سنجش میزان تابع SPI که انتظار می رود هنگامی که به TEE منتقل می شود ، عملکرد آن را پایین بیاورد و چنین مواردی را مقایسه کند تخریب بین TEE های مختلف ؛ ما کاربرد این معیار را مورد ارزیابی قرار می دهیم.

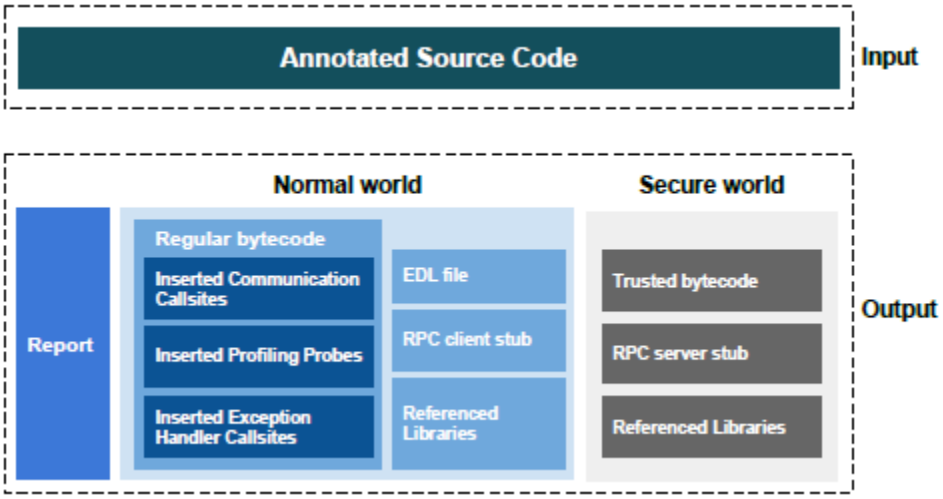
۱-۵ مرور راه حل

در این بخش ، ما زنجیره ابزار تجزیه و تحلیل مبتنی بر کامپایلر و بازسازی کد را معرفی می کنیم. ابزار و سپس ورودی و خروجی محیط امن را شرح می دهیم.



شکل ۲ مراحل پردازش محیط امن

فرایند توسعه نرم افزار شکل ۲ نشان داده شده است که فرایند توسعه نرم افزار برای پارتیشن بندی بلادرنگ را تشریح می کند. استفاده از سیستم های TEE با توجه به سیستم زمان واقعی به این گونه است که ابتدا توسعه دهنده توابع وابسته به SPI را در کد منبع با استفاده از دامنه امن مشخص می کند (مرحله ۱). سپس کد منبع به واسطه کامپایل می شود مرحله ۲) پس از آن ،محیط امن تعیین می کند که آیا TEE یا SGX با بررسی محیط اجرا یا پیکربندی ساخت به صورت درست پیاده سازی شده است یا خیر همچنین آیا سناریوی پارتیشن بندی مشخص می تواند محقق شود یا خیر. محیط امن به طور آماری، نمودار تماس سیستم تجزیه و تحلیل می کند (مرحله ۳). با توجه به نمودار فراخوانی سیستم و پارتیشن بندی، نمودار عملکرد قابل تقسیم (PFG) را که شامل تمام اطلاعات مورد نیاز برای تعیین صحت مشخصات است ؛ می سازد. در حالی که تحلیل استاتیک اعتبار معنایی مشخصات پارتیشن بندی ، یک تحلیل پویا جداگانه را تعیین می کند. در این مرحله مشخص می شود که آیا سیستم پارتیشن بندی شده مطابق محدودیت های زمان واقعی عمل می کند یا خیر. (مرحله ۴)

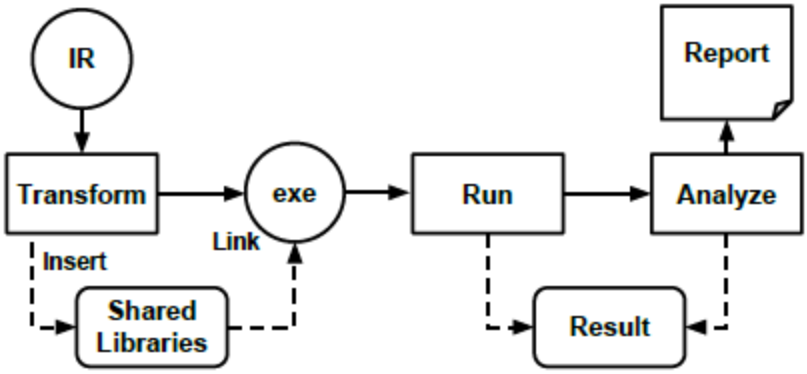


شکل ۳ورودی و خروجی RT-Trust

تولید کد

شکل ۳ ورودی ها و خروجی های کد RT-Trust را نشان می دهد .به عنوان ورودی ، RT-Trust ، کد منبع حاشیه نویسی شده را دریافت می کند و به عنوان خروجی ، مشخصات منبع ورودی را تغییر می دهد و همچنین کد اضافی تولید می کند که کامپایل شده و در حالت عادی و امن ادغام می شود. پارتیشن های محیطعادی ، RT-Trust را با درج پروفایل در کاوشگرها ، کنترل کننده های استثنا و کانال های ارتباطی تغییر می دهد .

تجزیه و تحلیل پویا RT-Trust به شناسایی احتمال تقسیم بندی مشخص کمک می کند تا محدودیت های اصلی زمان واقعی را برآورده کند .از آنجا که تضمین اینکه آیا اجرای پروفایل بدترین حالت را دارد؛ تجزیه و تحلیلهای ما فقط برای سیستم های زمان واقعی، قابل اجرا است.



شکل ۴ مراحل تحلیل RT-Trust

شکل۴ نشان می دهد که چگونه RT-Trust قابلیت تجزیه و تحلیل پویا را فراهم می کند .تحلیل ها با تغییر برنامه اصلی یعنی درج RT-Trust شروع می شود. کد پروفایل در توابع محدودیت های زمان واقعی به جای وارد کردن کل کد نمایه وارد می شود؛ RT-Trust تماس هایی را با توابع پروفایل ویژه ، برقرار می کند. به عنوان بخشی از کتابخانه های مشترک در دسترس قرار می دهد. اکنون RT-Trust توابع را به تنهایی ارائه می دهد. عملکردهای مشابهی را می توان توسط کتابخانه ها ارائه داد. این طراحی انعطاف پذیر توسعه دهندگان را قادر می سازد تا خدمات سفارشی خود را ارائه دهند نمایه کردن کتابخانه ها یا افزودن ویژگی های جدید به کتابخانه هایی که توسط RT-Trust ارائه شده است منطق پروفایل را تقویت می کند. پس از پیوند دادن کتابخانه های مشترک با محیط تبدیل شده برنامه ، توسعه دهندگان با فراخوانی تابع درج شده، عملکرد مناسب را انجام می دهند.
عملکردهای پروفایل در کتابخانه های مشترک توابع، محدودیت های زمان واقعی را اندازه گیری می کنند و داده های نتیجه را برای تجزیه و تحلیل آینده حفظ می کند .در نهایت ، RT-Trust داده ها را تخمین زده و مشخص می کند آیا توابع حاشیه نویسی می توانند نیازهای اصلی زمان واقعی را برآورده کنند و گزارش مجدد نتایج به توسعه دهنده داده می شود.

جدول ۱ : تلاش های برنامه نویس ها (ULOC)

Algorithm	RTTAs	Generate and Transform		Adjust	
		OP-TEE	SGX	OP-TEE	SGX
CRC32	۵	۳۸۸	۸۷	۰	۰
PC1	۴	۳۴۴	۷۳	۶	۶
RC4	۳	۲۹۲	۶۱	۳	۱
MD5	۳	۳۶۴	۸۶	۳	۱
DES	۲	۲۴۴	۴۶	۱۵	۳

نتایج

ما صحت RT-Trust را با اعمال تمام شرایط (یعنی تجزیه و تحلیل کد ، تبدیل ، و تولید) را به شرح زیر ارزیابی می کنیم.

با استفاده از RT-Trust تعداد کل ULOC به طور خودکار توسط RT-Trust تولید و تبدیل می شود. به طور مثال ۳۸۴ ~ ۲۴۴ برای OPTEE ؛ ۴۶ ~ ۸۷ برای SGX . نتایج کامل در جدول ۱ نشان داده شده است.

جدول ۲ - سربار پروفایل RT-Trust برحسب میلی ثانیه

	Execution Time		Invocation Intervals		Memory	
Algorithm	Normal	Secure	Normal	Secure	Parameter	Local
OP-TEE	۰,۴۴۲	۱۴۴	۰,۴۱۸	۱۳۹	۰,۰۵۱	۰,۰۵۳
SGX	۰,۲	۵۲	۰,۲۱۲	۲۵,۵		

عملکرد: جدول ۲ گزارش مربوط به سربار پروفایل RT-Trust و زمان اجرا را نشان می دهد. فواصل فراخوانی و مصرف حافظه را محاسبه می کند. RT-Trust سیستم ها را قبل و بعد از بازسازی آنها مشخص می کند. برآورد حالت قبل که آیا سیستم بازسازی شده همچنان به محدودیت های زمان واقعی پاسخ می دهد ، در حالی که حالت بعد ویژگی های اجرایی برآورد شده را با مواردی که در آن اجرا شده است مقایسه می کند. سخت افزارمحیط ها به شدت تفاوت زیادی در سربار پروفایل دارند:

برای OP-TEE، ۰,۴ میلی ثانیه در دنیای عادی در مقابل ۱۴۰ میلی ثانیه در دنیای امن.

برای SGX ، در دنیای عادی ۰,۲ میلی ثانیه در مقابل ۵۰ میلی ثانیه در دنیای امن.

این تفاوت عمدتا به دلیل تفاوت بین کارایی سیستم استاندارد لینوکس است.

جدول ۳ - محدودیت های TEE

Limitations	OP-TEE	SGX
Language	C	c/c++
Memory	No limit	Hard limit
Threading	No	No
Sys./Lang.APIs	Sepecial version	Sepecial Version

محدودیت های TEE: جدول ۳محدودیت های OP-TEE و SGX را نشان می دهد. برای پشتیبانی زبان ، قسمت مورد اعتماد OP-TEE فقط به زبان C قابل نوشتن است. که برای SGX را می توان در دو زبان C و C ++ نوشت ، در حالی که کانال ارتباطی بین قطعات مورد اعتماد و غیرقابل اعتماد را فقط می توان با C نوشت. برای تخصیص حافظه ، OP-TEE محدودیت اندازه ثابت ندارد و با محدوده بالایی به مقدار حافظه فیزیکی تبدیل می شود. در مقابل ، حداکثر اندازه حافظه محافظت شده SGX توسط سیستم محدود می شود.

جدول ۴ – FPI مربوط به OP-TEE و SGX

Algorithm	OP-TEE	SGX
CRC۳۲	۰,۹۸۲	۰,۹۷۳
PC۱	۰,۵۸۱	۰,۶
RC٤	۰,۱۴۲	۰,۴۳۶
MD٥	۰,۲۱۸	۰,۲۰۵
DES	۰,۷۶۶	۰,۸۰۳

انتخاب بین OP-TEE یا SGX

جدول ۴ شاخص عملکرد عملکرد (FPI) هر یک از معیارهای خرد را برای OPTEE و SGX را نشان می دهد . به طور کلی ، ارزش FPI برای هر دو TEE در همه معیارها قابل مقایسه است. هرچه سرعت اجرا قبل از انتقال به TEE بیشتر باشد ، مقدار FPI بزرگتر است. (یعنی بیشتر از کاهش عملکرد). دلیل آن این است که اگر یک تابع سریع اجرا شود ؛ هزینه های اضافی کانال ارتباطی وجود دارد.

به طور خلاصه ، توسعه دهندگان همیشه باید از TEE با استفاده کنند کخ کوچکترین مقدار FPI را دارد . با این حال ، اگر زمان اجرای یک عملکرد SPI بسیار کمتر از زمان صرف شده توسط کانال ارتباطی باشد. سپس OP-TEE و SGX قابلیت تخریب بالا دارد.

نتیجه

در این فصل ، ما RT-Trust را که یک برنامه نویسی فوق العاده کامل را ارائه می دهد؛ مطرح می کنیم. مدل با تجزیه و تحلیل استاتیک و پویا برای تعیین اینکه آیا استراتژی پارتیشن بندی منطقی است و آیا سیستم پارتیشن بندی مطابق با محدودیت های اصلی در زمان واقعی است و یک بازسازی خودکار که نسخه اصلی را هنگام ایجاد ارتباط RPC سفارشی و کد مدیریت استثنا تغییر می دهد. ما رویکرد به طور خودکار سیستم های زمان واقعی را با عملکردهای وابسته به SPI برای موارد قابل اعتماد بازآرایی می کند. اجرا تحت محدودیت های زمان واقعی نتایج ارزیابی استفاده از RT-Trust به نشانهای کوچک

نشان دهنده برنامه نویسی امن به منظور کاهش تلاش برنامه نویس مورد نیاز برای جداسازی SPI تحت محدودیت های زمان واقعی است .

فصل ۶

شناسایی و مهاجرت کد غیر حساس در TEE

تعداد زیادی از دستگاه های محاسباتی به طور مداوم حجم عظیمی از داده ها را جمع آوری می کنند (به عنوان مثال ، شناسه های بیومتریک ، موقعیت جغرافیایی و تصاویر) ، که بسیاری از آنها حساس هستند [۱۶۰]. داده های حساس و پردازش کد آنها هدف بسیاری از افشای داده ها و حملات تغییر کد است.[۱۱ ، ۱۲ ، ۱۳ ، ۱۴ ، ۴۸ ، ۴۹]. مکانیسم حفاظتی که به طور فزاینده ای محبوب می شود ، کد و داده های دنیای خارج در یک محیط اجرای مطمئن (TEE) حساس را جدا می کند کد و داده های دنیای خارج ۱ در یک محیط اجرای مطمئن (TEE)

با افزایش حجم کد در TEE ،همه کد حساس نیست ، بنابراین پایگاه محاسباتی مورد اعتماد (TCB) بی جهت رشد می کند و باعث می شود تا مسائل مربوط به عملکرد و امنیت وقتی نوبت به عملکرد می رسد ، تحقیقات قبلی آن را مشخص می کند.

ارتباط بین TEE و جهان خارج به عنوان یک تنگنای عملکردی که می تواند بیشتر زمان اجرا را مصرف می کنند [۱۰۸]. به عنوان مثال ، فراخوانی تابع متعدد ، ورود یا خروج از TEE ، باعث ایجاد حجم زیادی از ارتباطات ورودی/خروجی و کند شدن سرعت کل سیستم ارتباط می شود. [۱۶۳]

مبادله داده های حساس

موبایل ، اینترنت اشیا و دستگاه های پوشیدنی به طور مداوم حجم بیشتری از داده های کاربر را جمع آوری می کند؛ مانیتورهای بهداشتی علائم حیاتی صاحبان خود را ردیابی می کنند. تلفن های هوشمند داده های شخصی حسی ، می خوانند و از جمله مکان GPS ، سرعت ، جهت و غیره. دستگاه های اینترنت اشیا به دست می آورند. وقتی صحبت از داده های حساس می شود ، یک اصل اساسی وجود دارد : تضاد بین نیازهای کاربر نهایی و آرزوهای توسعه دهنده برنامه. کاربران نهایی

می خواهند مطمئن شوند که اطلاعات حساس آنها خصوصی و غیرقابل دسترسی برای افراد غیرقابل اعتماد است. توسعه دهندگان برنامه می خواهند از خواص داده های حساس برای ارائه استفاده کنند. برنامه های کاربردی هوشمندی که تجربیات شخصی کاربر را ارائه می دهند و هوشمند و حساس به زمینه خدمات. هستند؛ این دو هدف ظاهراً آشتی ناپذیرند.

سه نمونه زیر را از برنامه های کاربردی با حجم بالا که به طور بالقوه کار می کنند در نظر بگیرید

(۱) برنامه های انتقال تلفن همراه اطلاعات ترافیک را در زمان واقعی به کاربران خود ارائه می دهند.همچنین هنگام استفاده از برنامه های کاربردی ، این اطلاعات را به اشتراک بگذارید. یک برنامه انتقال به طور مداوم GPS فعلی دستگاه خود را روی ابر بارگذاری می کند. مکان و سرعت ، که سپس برای تخمین اطلاعات ترافیک در زمان واقعی استفاده می شود. با اینکه این اطلاعات به صورت ناشناس بارگذاری می شوند ، با توجه به داده های GPS و سرعت کافی ، این کار قابل اعتماد نیست. ممکن است طرف بتواند با روال روز مالک دستگاه آشنا شود و از این اطلاعات برای اهداف نادرست، سوء استفاده کند. بنابراین ، اگرچه مشارکت کنندگان ممکن است مایل به تخمین اطلاعات ترافیکی در زمان واقعی ، باشند و نمی خواهند مکان GPS آنها فاش و ضبط شود.

(۲) گروهی از دوستان به دنبال رستورانی هستند که با هم شام بخورند. تلفن های هوشمند آنها، تاریخ ناهارخوری صاحبان خود را حفظ کنند. بر اساس تاریخ غذاخوری هر فرد ، سرور در یک میدان خرید می تواند پیشنهاد کند که کدام رستوران ها برای این مکان مناسب تر هستند. ممکن است افراد مایل نباشند که تاریخچه ناهار خوری خود را با یک گروه غیر قابل اعتماد به اشتراک بگذارند.

(۳) یک ساختمان هوشمند ممکن است درجه حرارت و سطح روشنایی خود را مطابق با حرارت تنظیم کند. ترجیحات ساکنان فعلی آن کاربرانی که دارای ردیاب سلامت شخصی هستند و مجهز به تلفن های هوشمند هستند ؛ می توانند حیات و مکان ساختمان صاحبان خود را گزارش کنند. این اطلاعات یکی از عناصر کلیدی است که ساختمان را "هوشمند" می کند. افراد ممکن است مایل نباشند مواد حیاتی و مکان فعلی آنها را به یک شخص غیرقابل اعتماد فاش کنند.

لطفاً توجه داشته باشید که در هر سه مثال بالا ، هنوز خدمات هوشمند قابل ارائه است. در حالی که داده های حساس کاربران نامرئی است. برای محاسبه میانگین سرعت گزارش شده ، سیستم نظارت بر ترافیک نباید از سرعت واقعی هر وسیله نقلیه عبوری آگاه باشد. این اطلاعات حساس می تواند مخفی بماند و فقط میانگین آماری آنهامحاسبه و در پیش بینی شرایط فعلی ترافیک استفاده می شود. برای توصیه یک رستوران ، سرور نباید نیاز داشته باشد که افراد درگیر چه رستوران های خاصی دارند و می تواند یک رستوران قابل قبول پیشنهاد کند. در مورد محبوب ترین غذاهای هر فرد ، اطلاعات آماری را می توان به دست آورد.

درسهای آموخته شده

علیرغم مزایای حفظ حریم خصوصی ObEx که در بالا نشان داده شد ، باید توسط JVM بومی پشتیبانی شود. این پشتیبانی بومی می تواند (۱) پیچیدگی افزایش امتیازات و حملات جایگزینی کتابخانه را پیچیده کند ، (۲) ادغام کردن مدیریت چرخه زندگی ObEx (۳) از وابستگی جلوگیری کند کتابخانه های امنیتی شخص ثالث در ادامه این مزایای مورد انتظار را توضیح می دهیم.

پیچیدگی افزایش امتیاز و حملات جایگزینی کتابخانه: از آنجا که زمان اجرا محلی ObEx یک کتابخانه مشترک است که توسط JVM بارگیری شده است ، افزایش امتیاز (به عنوان مثال ، root حساب در لینوکس) می تواند مستقیماً به داده های حساس موجود در حافظه دسترسی پیدا کند. علاوه بر این ، یک حمله می تواند زمان اجرای ObEx را با یک نسخه مخرب در زمان بارگذاری جایگزین کند. ادغام با JVM می تواند یکپارچگی مکانیسم حفاظت از حریم خصوصی ObEx را افزایش دهد.

نتیجه گیری

ظهور دستگاه های تلفن همراه ، پوشیدنی و IoT حجم زیادی از داده ها را ایجاد کرده است که دارای برخی محدودیت های حریم خصوصی است. کاربران نهایی می خواهند حساسیت خود را حفظ کنند. داده ها خصوصی هستند ، در حالی که شرکت مایل است از این داده ها برای ارائه هوشمند و حساس به زمینه خدمات و برنامه های کاربردی استفاده کند.

در این کار ، ما استدلال می کنیم که نوآوری در فناوری برنامه نویسی می تواند به حل این موارد کمک کند. دو دستور کار متضاد: داده های حساس را می توان خصوصی نگه داشت ، در حالی که شرکت ها هنوز می توانند ادغام شوند اطلاعات تجاری ارزشمند از داده ها ، نحوه طراحی ، پیاده سازی را ارزیابی کرد.

در حالی که چرخه حیات خود را کنترل می کند و با سیاست پرس و جو ، رابط های برنامه نویسی را برای انجام محاسبات آماری ، ارائه می دهد ؛ به طوری که توسعه دهندگان قادر به ساخت تلفن همراه هوشمند و برنامه های کاربردی اینترنت اشیا هستند.

فصل ۷

کار آینده

همانطور که در فصل های قبل مورد بحث قرار گرفت ، تحقیقات این پایان نامه در نرم افزار و فضای مهندسی برای درک و مدیریت SPI با استفاده از رویکردها ، نوآوری می کند.

تکنیک های و ابزارها ، یک توسعه دهنده می تواند توابع و متغیرهای SPI را شناسایی کرده و همچنین آنها را در برابر آنها نشت داده ها محافظت کند.

با این وجود ، انتظار این که این پایان نامه داشته باشد که تمام مشکلات مربوط به حفاظت از SPI را برطرف کند؛ غیر واقعی است. به عنوان مثال ، هر زمان که برنامه های تلفن همراه مبادله داده ها را انجام دهند ، این فرآیند ممکن است همچنان خطرات نشت داده احتمالی مانند حملات علیه کانال ارتباطی ، از جمله رهگیری ، استراق سمع را داشته باشند. برای حل این مشکلات پیشنهاد می کنیم که در زمینه طراحی وسایل وسط نیز نوآوری داشته باشیم تا بتوانیم موارد جدیدی را ارائه دهیم

طرح هایی که می توانند خطرات نشت داده را کاهش دهند در حالی که هنوز به افراد غیرقابل اعتماد، اعتماد می کنند

طرفهای مبادله داده:

(۱) به سوی ارتباطات مبتنی بر پیام ایمن: در سیستم عامل های تلفن همراه مدرن ، تبادل اطلاعات بین مولفه ها تحت تأثیر حملات نشت داده قرار می گیرد که از طریق آنها غیرقابل اعتماد است. برنامه ها به داده های منتقل شده دسترسی پیدا می کنند. دفاعیات موجود نیز بیش از حد محدود کننده هستند.

همه مبادلات مشکوک داده ها را مسدود کنید ، بنابراین از دریافت هرگونه برنامه به برنامه جلوگیری می کنید. برای بهتر شدن تبادل اطلاعات بین مولفه های ایمن ، هدف ما طراحی و پیاده سازی مدلی است که امنیت را تقویت می کند ، در حالی که به برنامه های غیرقابل اعتماد اما غیر مخرب اجازه می دهد تا برنامه های خود را در زمینه منطق کسب و کار اجرا کنند. و تحویل داده های حساس در یک پاکت رمزگذاری شده پنهان منتقل می شوند. تحویل آنها چند شکلی است: با توجه به قابلیت اطمینان مقصد ، می تواند باشد. بدون داده ، داده خام یا داده رمزگذاری شده باشد.

(۲) خصوصی سازی اشتراک گذاری داده های دستگاه: برای به حداکثر رساندن رضایت مشتری ، خدمات مدرن تلفن همراه (به عنوان مثال ، تبلیغات ، انتقال ، مراقبت های بهداشتی) باید شخصی باشد. کاربران برای شخصی سازی خدمات خود و ارائه دهندگان برنامه به طور مداوم، موارد لازم را انجام داده و داده ها را به صورت محلی یا از طریق ابر با دیگران به اشتراک می گذارند. با این حال ، مکانیسم های موجود برای به اشتراک گذاری داده های حسگر می تواند با حملات نشت داده مورد سوء استفاده قرار گیرد ، بنابراین حریم خصوصی می تواند به خطر بیفتد. برای حل این مشکل ، در نظر داریم که چارچوبی برای اشتراک گذاری خصوصی داده های حسگر روی دستگاه که از طریق آن برنامه خصوصی سازی می شود

را طراحی و پیاده سازی کنیم.

خلاصه و نتیجه گیری

نشت داده ها و حملات تغییر اطلاعات، انگیزه تلاش های تحقیقاتی با هدف ایجاد مکانیسم هایی برای حفاظت از منطق حساس تجارت ، الگوریتم ها و داده ها است .اگرچه حفاظت SPI از منظر امنیت به طور گسترده مورد مطالعه قرار گرفته است ، این پایان نامه تحقیقات این مشکل را از زاویه مهندسی نرم افزار ، با هدف ارائه فناوری های جدید نرم افزاری که می تواند توسعه دهندگان را بهتر پشتیبانی کند. این پایان نامه تحقیقات در زمینه مهندسی نرم افزار برای درک و مدیریت SPI نوآوری را به طور مشخص، بیان می کند.

(۱) تجزیه و تحلیل برنامه و پشتیبانی برنامه نویسی برای نتیجه گیری را طراحی و توسعه می دهیم. معناشناسی کاربرد سازه های برنامه ، با هدف کمک به توسعه دهندگان برای درک SPI را شناسایی کنید ؛ (۲) ابزارهای برنامه نویسی قدرتمندی را ارائه می دهند که کد را به طور خودکار ، تغییر شکل می دهند. با هدف کمک به توسعه دهندگان به طور موثر SPI را از بقیه کد جدا می کنیم. (۳) ارائه مکانیسم برنامه نویسی برای توزیع محیط های اجرایی مدیریت شده که SPI را با هدف فعال کردن اجزا برای تبادل ایمن و مطمئن SPI پنهان می کند .

هدف اصلی این تحقیق این است که به توسعه نرم افزار و تجزیه و تحلیل ، در نتیجه ایجاد امنیت ، قابل درک و کارآمد پایه ای برای محافظت از SPI کمک کند.