طراحی و پیاده سازی CPU امنیتی نهفته براساس استراتژی چندگانه

چکیده

پایشگری روند کنترل ، ردگیری روند اطلاعات و پایشگری حافظه، سه راه حل اصلی برای بالا بردن امنیت سیستم های نهفته در سطح معماری سخت افزار می باشند. با این حال اکثر مطالعات کنونی درباره ی امنیت سیستم های نهفته، راه حل های بالا را در ابعاد جداگانه در نظر می گیرند. ما کار خود را از یک مدل عملیاتی در سطح دستور العمل آغاز کرده و یک استراتژی چندگانه ی امنیتی که به وسیله ی مطالعه ی مکانیزم عملیاتی ایمنی سیستم نهفته، ردگیری روند اطلاعات را با پایشگری حافظه ترکیب می کند را پیشنهاد می کنیم. به عنوان یک رویکرد سخت افزاری، این استراتژی معماری پردازشگر نهفته را با کنترل دفاعی ایمنی فرعی گسترش می دهد. نتایج آزمایشی نشان می دهند که این استراتژی چندگانه موثر تر است و قادر به شناسایی حملات مخرب بیشتری نسبت به یک راه حل واحد خواهد بود. اثر بخشی استراتژی چندگانه ی ایمنی پیشنهادی ما در یک پلتفرم نمونه ی اولیه آرایه گیت از لحاظ زمینه قابل برنامه ریزی (FPGA) و بر اساس یک ریز پردازشگر Leon3 سفارشی بررسی شده است.

کلمات کلیدی : ایمنی روند اطلاعات ، ردگیری اثر نا خواسته ، ماژول پایشگر حافظه ، معماری پردازشگر نهفته.

مقدمه :

الف) با توجه به استفاده ی گسترده از سیستم های نهفته، مسائل امنیتی پردازشگر های نهفته توجهات در حال افزایش را جلب می کنند. تا به اینجای کار تحقیق بروی ایمنی پردازشگر نهفته به طور عمده بر کنترل پایشگری روند کنترل، ردگیری روند اطلاعات و پایشگری حافظه تمرکز داشته است. با این حال بیشتر کار موجود، این موارد را به عنوان ابعاد جداگانه در نظر گرفته و درصد کمی از آن رویکرد یکپارچه را پیشنهاد کرده و ابعاد گوناگونی را به طور همزمان نشانه گرفته است که باعث ایجاد انگیزه در کار ما می شود.

ب) ردگیری روند اطلاعات که به نام های کنترل روند اطلاعات یا ردگیری اثر نا خواسته نیز شناخته شده است یک سیاست امنیتی مهم به شمار می رود. ردگیری روند اطلاعات پویا (DIFT)، اطلاعات غیر قابل اطمینان را به عنوان یک مورد اثر نا خواسته علامت گذاری می کند و انتشار آن را در یک سیستم ایمنی ردگیری می کند. این DIFT هر کلمه در حافظه ی سیستم را با یک علامت پیوست می دهد و اطلاعات جدیدی که از سوی مورد غیر قابل اطمینان در راه است را به عنوان یک اطلاعات اثر نا خواسته علامت گذاری می کند. در مورد اطلاعات اثر نا خواسته که در یک روش ناامن ممکن استفاده شده اند مانند اجرای یک دستور العمل زبان جستار ساخت یافته ی اثر نا خواسته (SQL) یا انتشار یک اشاره گر اثر نا خواسته، سیستم ایمنی یک استثناء امنیتی ایجاد می کند. در حقیقت مطالعات زیادی در زمینه ی ردگیری روند اطلاعات به انجام رسیده است.

ج) پایشگری روند اطلاعاتی بر ردگیری روند داده ی خارجی در درون پردازشگر (مانند داده هایی از ورودی خروجی چند منظوره (GPIO)، درگاه های متوالی ، و شبکه ها) تمرکز داشته است که می تواند برای اجتناب از عملیات های غیر قانونی که ناشی از این داده ها یا برنامه های خارجی هستند (مانند دزدیدن اطلاعات شخصی کاربران که در سیستم ذخیره سازی شده)، مفید باشد. با این حال مکانیزم پایشگری روند اطلاعات تحلیل دقیقی در مورد ایمنی برنامه ها یا داده های خارجی ارائه نمی دهد. این مکانیزم تنها تصمیم می گیرد که کدام اثر نا خواسته داده ای نیاز به انتشار دارد و کدام داده زمانی که در حال بررسی اثر نا خواسته است نیاز به بررسی دارد. اگر چه مکانیزم پایشگری روند اطلاعات قادر به تشخیص برخی حملات متداول است ممکن است منجر به میزان مثبت کاذب بالایی در برنامه ی ایمنی دیگر در سیستم شود. علاوه بر این به منظور تشخیص نوع بخصوصی از حملات این مکانیزم نیاز به پیکربندی یک ثبات انتشار اثر نا خواسته (TRP) و یک ثبات تشخیص اثر نا خواسته (TDP) دارد و در صورتیکه نوع حملات تغییر کند نتیجتاً هر دوی آن ها نیاز به تغییر پیدا می کنند که بدون شک انعطاف پذیری آن را نیز محدود می کند.

د) مکانیزم پایشگری حافظه، هدف تشخیص حملات مخرب را به وسیله ی محافظت از فضای داده در هنگامیکه برنامه اجرا می شود و حفاظت از کد مخرب در برابر تغییر غیر مجاز فضای داده ی یک برنامه، بدست می آورد. فضای داده ی برنامه شامل بخش پشته، بخش توده ای، بخش داده ی جهانی و بخش متن می باشد. پیاده سازی

استراتژی پایشگری حافظه در پردازشگر نهفته می تواند از بسیاری از حملات فرا روندی میان گیر متداول مانند حملات فرا روندی پشته و حملات فرا روندی توده ای جلوگیری کند. چندین روش مبتنی بر سخت افزار در زمینه ی پایشگری حافظه وجود دارد.

ر) پایشگری حافظه نیازمند ایجاد تحلیلی دقیق درباره ی ایمنی خود برنامه که شامل نوع دستور العمل اجرا شده و اطلاعات کرانه ی فضای داده ی برنامه که مشخص می کند که آیا دستور العمل های اجرا شده دارای تهدیداتی در فضای داده ی برنامه هستند یا خیر می باشد. با این حال فرآیند گردآوری کد منبع که توسط زبان برنامه نویسی پیشرفته در دستور العمل های ماشینی نوشته شده دارای ارتباط مناسبی با نوع گرد آوردنده می باشد. به ازای یک گذرگاه معین کد منبع که توسط زبان برنامه نویسی پیشرفته نوشته شده است، دستورالعمل ماشینی که به وسیله ی گرد آوردنده ی متفاوتی گرد آوری شده است ممکن است متفاوت باشد. بنابراین نتیجه ای که توسط ماژول پایشگری حافظه تحلیل شده همچنین ممکن است متفاوت باشد که می تواند منجر به یک نرخ مثبت کاذب بالا و یک نرخ منفی کاذب بالا شود. تحلیل بالا نشان می دهد که در سیستم های نهفته، یک استراتژی پایشگری حافظه ی واحد برای جلوگیری از تمامی حملات مخرب کافی نمی باشد.

ز) به طور خلاصه سه روش بالا با استفاده از مشخصات و مزایا یا معایب خود قادر به بهبود بخشیدن به ایمنی پردازشگر های نهفته می باشند. بر اساس تحلیل بالا، ما توجه کاملی به قدرت های ردگیری روند اطلاعات و پایشگری حافظه ی خود و ترکیب آن ها با یکدیگر خواهیم داشت. ما پایشگری روند اطلاعات را به وسیله ی تغییر کد سطح انتقال ثَبات (RTL) واحد عدد صحیح هسته و اضافه کردن TCR در واحد عدد صحیح هسته طراحی می کنیم. پایشگری روند اطلاعات، عملکرد طبقه بندی انواع حملات، قابلیت برنامه نویسی سیاست های امنیتی به طور انعطاف پذیر، و قابلیت چند حمله ای همزمان که با هزینه ی بسیار کمی دفاع می شود را فراهم می کند. ما به وسیله ی اضافه کردن یک ماژول سخت افزاری که همراه با پردازنده ی نهفته در مدار عمل می کند و همچنین قادر به شناسایی حملات فرا روندی میانگیر متداول به طور موثر می باشد، پایشگری حافظه را پیاده سازی می کنیم. در آخر طراحی خود را به یک مدار توسعه ی FPGA نگاشت کردیم و یک سیستم نمونه ی اولیه را توسعه دادیم. به منظور استفاده ی بهتر از این دو روش ما سطح ایمنی ردگیری روند اطلاعات را تنظیم می کنیم. نتایج

آزمایشی نشان می دهند که در مقایسه با یک استراتژی واحد ردگیری روند اطلاعات و استراتژی واحد پایشگری حافظه، استراتژی چندگانه ی ما می تواند به طور موثر انواع بیشتری از حملات را در هنگام اجرا شناسایی کند که هم از ردگیری روند اطلاعات و هم از پایشگری حافظه بهره می برد و در نهایت امنیت کلی سیستم های نهفته را افزایش می دهد.

معماری :

الف) از آنجاییکه طراحی ما ترکیبی از ردگیری روند اطلاعات و پایشگری حافظه است، معماری طراحی ما به دو بخش تقسیم می شود : معماری ردگیری روند اطلاعات و معماری پایشگری حافظه.

1. معماری ردگیری روند اطلاعات :

ب) با دنبال کردن ایده ی DIFT، هسته ی پردازشگر نهفته به منظور فراهم کردن ردگیری اثر نا خواسته در یک محیط انتشار با علائم انتشار دهنده ی اثر نا خواسته گسترده شده است. هر کلمه با چهار بیت که در جدول 1 مشخص است پیوست شده است. به دلیل وجود بیت های اضافه شده تغییراتی در معماری سخت افزار مانند گسترش چهار بیت در تمامی ثبات، نهان گاه ها (کَش ها)، حافظه ها و گذرگاه داده برای انتشار اثر نا خواسته اعمال شده اند. هنگامیکه واحد پردازشی مرکزی (CPU) در حال اجرای پایپ لاین دستور العمل است تمامی داده ها برای ردگیری انتشار منبع داده به یک چهار بیتی پیوست شده اند. بیت هایی که در گذرگاه و حافظه گسترش یافته اند در قابل توجه ترین بیت (MSB) داده اضافه می شوند.

Tag bits	Definition
Tag [0]	Taint mark bit: '1': The data is untrusted; '0': The data is trusted.
Tag [2:1]	Threat level classification bits: '00': The data has no threat; '01': The data is in a low threat level; '10': The data is in a middle threat level; '11': The data is in a high threat level.
Tag [3]	Sensitive information mark bit: '1': The data is sensitive; '0': The data is insensitive.

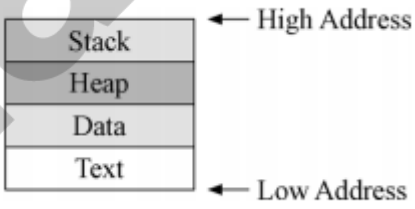
جدول 1

ج) فرآیند ردگیری اثر نا خواسته شامل سه تکلیف می شود که عبارتند از علامت گذاری اثر نا خواسته ، انتشار اثر نا خواسته ، و بررسی اثر نا خواسته. بسته به فعالیت های پردازشگر نسبت به داده ی خارجی، پیشنهاد ما استفاده از طبقه بندی سطح تهدید است. همچنین در نظر داریم که دقت ردگیری اثر نا خواسته می تواند زمانی که سیستم در حال عمل است افزایش یابد و تمامی رفتار های غیر طبیعی را می توان به طور داخلی پایشگری کرد. ما همچنین منطق انتشار را تحت عنوان تمام انتشار قرار می دهیم، به این معنی که تمامی آثار نا خواسته از منطق تمام انتشار عبور خواهند کرد.

2. معماری پایشگری حافظه :

د)مقدار اطلاعات میان گیر که از طریق پایشگر پویا بدست آمده، سیاست دفاع ماژول امنیتی سخت افزار را تعیین می کند. میان گیر هایی که در برنامه تعریف شده اند در فضای داده ی برنامه قرار دارند.

ر)یک فضای داده ی برنامه ی معمول چنانکه در شکل یک نشان داده شده شامل بخش پشته، بخش توده، بخش داده ی جهانی و بخش متن می باشد. اشاره گر های محلی که متغیر ها و آرایه های محلی هستند که در برنامه تعریف شده اند در بخش پشته ذخیره شده اند. یک میان گیر که توسط یک تابعی که به حافظه اختصاص داده شده مانند مالوک تعریف شده است به بخش توده تعلق دارد. اشاره گر های جهانی، متغیر ها و آرایه های جهانی که در برنامه تعریف شده اند در بخش داده ی جهانی ذخیره شده اند. حمله ی فرا روندی میان گیر معمولاً در این سه بخش اتفاق می افتد که بیشترین حملات به بخش پشته صورت می گیرد. همچنین کد قابل اجرای مضاعف برنامه در بخش متن ذخیره شده است. بخش متن تنها خواندنی است بنابراین باز نویسی آن امر دشواری است و همچنین دارای مقاومت خوبی در برابر حملات دستکاری می باشد.



شکل 1

1) محافظت بخش پشته

ز) در مورد محافظت بخش پشته، ما محدوده ی محافظت شده را کاهش داده و تنها نشانی برگشت و اشاره گر های پشته را محافظت می کنیم که دارای موقعیت نسبتاً ثابتی در پشته داشته و مرتباً در این بخش مورد حمله قرار می گیرند.

2) محافظت بخش توده

ژ) از آنجاییکه دسترسی به اطلاعات کران در پشته محدود است که باعث غیر ممکن شدن محافظت کامل داده در فضای داده ی پشته می شود، وضعیت در بخش توده به طور کامل متفاوت است. حافظه ی توده به صورت پویا به وسیله ی یک تابع ویژه مانند مالوک اختصاص داده شده است. از دید سخت افزاری می توانیم تابع مالوک را از دستور العمل های نهادی آن تشخیص داده و سپس نشانی آغازین میان گیر و اطلاعات طول را از پارامتر های تابع به دست آوریم که می توان با استفاده از آن به طور کامل مانع فرا روند میان گیر در بخش توده شد.

3) محافظت بخش داده

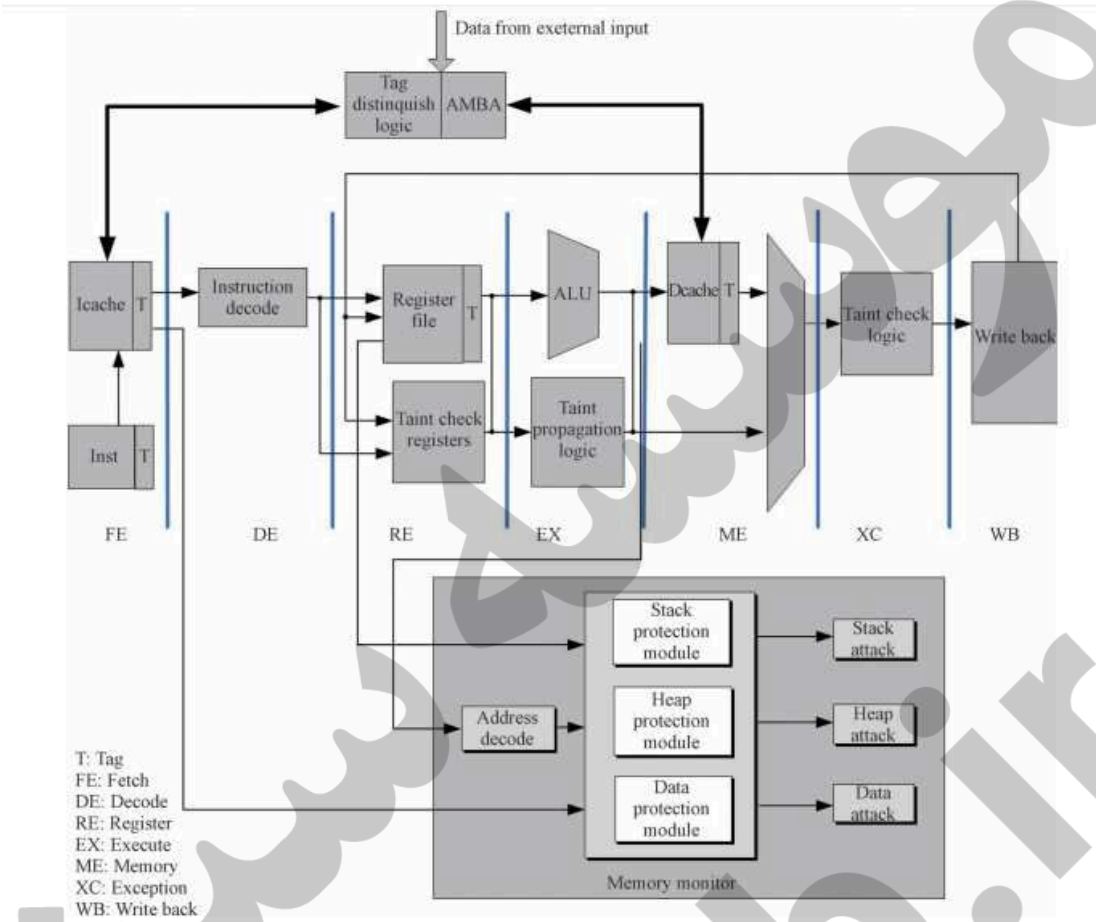
و) متغیر جهانی، آرایه ی جهانی، متغیر ایستا و آرایه ی ایستا همگی در فضای داده ی جهانی ذخیره شده اند. داده ای که تصمیم گیری درباره ی جهت در حال اجرای برنامه را بر عهده دارد داده ی کنترل نامیده می شود مانند نشانی برگشت ، اشاره گر تابع و غیره. در صورتیکه یک حمله کننده قصد بدست گرفتن کنترل برنامه را داشته باشد ابتدا می بایست داده ی کنترل را تغییر دهد. داده ی کنترل، نشانی دستور العمل معین در برنامه است پس بنابراین در یک واحد حافظه ی کلمه ذخیره شده است. بر این اساس عملیات پیرو مربوط به واحد حافظه بایستی عملیات کلمه ای باشد. عملیات مربوط به یک واحد حافظه که با یک نویسه یا یک آرایه ی رشته مرتبط است، یک عملیات بایت می باشد. به طور معمول یک حمله کننده به وسیله ی تغییر محتوای واحد های حافظه ی مجاور، یک آرایه ی رشته را فرا روند می سازد. در صورتیکه واحد های حافظه به طور اتفاقی یک اشاره گر تابع را ذخیره کرده و عملیات بروی آن یک عملیات بایتی باشد که با حالت عملیات قبلی مربوط به واحد حافظه ناسازگار است، آنگاه حمله قابل تشخیص است. بدین ترتیب ما ماژول محافظت سخت افزاری خود را بر اساس سیاست امنیتی بالا خواهیم ساخت.

پیاده سازی سخت افزار

الف) به منظور معتبر سازی اثر بخشی و سنجیدن کارایی ترکیب ردگیری روند اطلاعات و پایشگری حافظه، ما همانند پلتفرم نمونه سازی خود از پردازشگر LEON3 که یک پردازشگر 32 بیتی با معماری SPARC-V8 است، استفاده می کنیم.

ب) ما به وسیله ی تغییر کد RTL هسته ی IU و با اضافه کردن TCR در هسته ی IU و همچنین با اضافه کردن یک ماژول سخت افزاری که همراه با پردازشگر نهان در مدار عمل می کند به ترتیب، پایشگری روند اطلاعات و پایشگری حافظه را پیاده سازی می کنیم. شکل 2 نمودار ساده شده ای از سخت افزار طراحی ما را نشان می دهد.

ج) به منظور یکی سازی تابع پایشگری روند اطلاعات، ما کد های RTL منبع باز پردازشگر LEON3 را به منظور اضافه کردن پشتیبانی سخت افزاری برای افزودن علائم 4 بیتی به ثبات یا نهان گاه ها، ثبات بررسی اثر نا خواسته و منطق انتشار اثر نا خواسته ، تغییر دادیم. علاوه بر این ما گذرگاه معماری گذرگاهی ریز کنترل کننده ی پیشرفته (AMBA) را به منظور سازگار بودن با واحد های ذخیره ی علامت دار خود گسترش دادیم و فرآیند بررسی اثر نا خواسته را همراه با یک مکانیزم تولید استثنای جدید مهیا کردیم. علامت گذاری، انتشار و بررسی توابع به ترتیب به گذرگاه AMBA، واحد منطق محاسباتی (ALU) و مرحله ی استثنای پایپ لاین اضافه شده اند. زمانی که داده ی خارجی بروی گذرگاه AMBA انتقال یابد به وسیله ی یک علامت، علامت گذاری می شود و علامت [0] به 1 تغییر می یابد .



شکل 2

د) شکل 2 همچنین معماری سخت افزاری پایشگر حافظه را نشان می دهد. ماژول پایشگر ، دستور العمل ها را در مسیر داده بین نهان گاه (کَش) و واحد IU قطع کرده و اطلاعات امنیتی مورد نیاز برای پایشگری از پایپ لاین را بدست می آورد. ماژول پایشگر حافظه شامل سه بخش است : ماژول حافظت پشته، ماژول حافظت توده و ماژول حافظت داده ی جهانی. رمز گشای نشانی، نشانی های مورد هدف دستور العمل های عملیات حافظه را مطابق فضای داده ای که نشانی به آن تعلق دارد، به ماژول حفاظت سخت افزار متناظر منتقل می کند. مشهود است که تمام ماژول پایشگر حافظه قادر به عمل به موازات پردازشگر به صورت کامل می باشد که این موضوع باعث می شود که جریمه ی عملکرد در رابطه با سیستم کلی بسیار کوچک باشد.

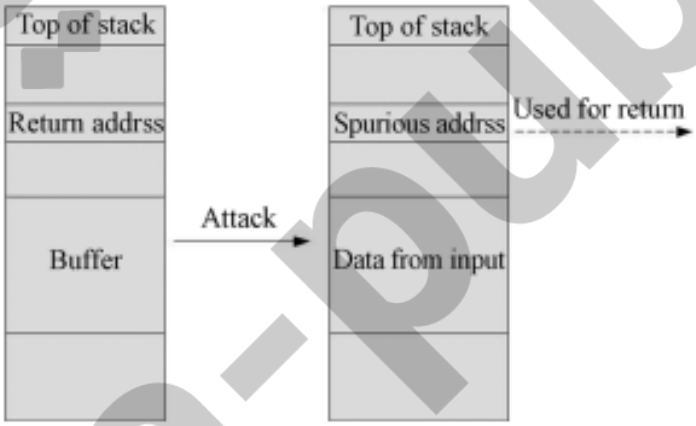
ارزیابی امنیتی

الف) به منظور تایید اثر بخشی و مقدار سنجی عملکرد فراتر از طرح پیشنهادی ما که در برابر حملات نرم افزاری مخرب دفاع می کند، ما به ترتیب از دو نوع حمله ی مخرب در فرآیند تایید استفاده کردیم که از حمله ی فرا روندی پشته و حمله ی فرا روندی توده تشکیل شده اند.

ب) برای سازگار ساختن این دو روش با یکدیگر و بدست آوردن مزایای مکمل هر دوی این روش ها، سطح امنیتی ردگیری روند اطلاعات را کاهش داده ایم. تعدادی از حملات از دست رفته برای شناسایی شدن قادرند تا از ماژول پایشگری حافظه عبور کنند که این موضوع نه تنها نرخ مثبت کاذب ردگیری روند اطلاعات را کاهش می دهد بلکه همچنین نرخ منفی کاذب کل سیستم را افزایش نخواهد داد. نتایج تایید نشان می دهند که ترکیب پایشگری روند اطلاعات و پایشگری حافظه می تواند به طور موثر در برابر انواع حملات در هنگام اجرا دفاع کند.

1. حمله ی فراروند پشته

ج) مدل حمله ی فراروند میان گیر که در شکل 3 نشان داده شده در یکی از برپایی های آزمایشی ما در نظر گرفته شده است.



شکل 3

د) داده ی خارجی می تواند از طریق کانال های مختلفی مانند UART، GPIO، کیبورد و اترنت به داخل سیستم منتقل شود. در مورد برپایی آزمایشی ما، استراتژی ردگیری اثر نا خواسته عمدتاً داده ی خارجی را از UART در نظر می گیرد. بخش اصلی کد حمله در شکل 4 نشان داده شده است . اکنون تحلیلی از چگونگی شناسایی حملات توسط ردگیری روند اطلاعات و پایشگری حافظه را ارائه می کنیم. در مورد ردگیری روند اطلاعات، این حمله از

طریق فرا روند میان گیر، نشانی بازگشت را تغییر می دهد. منطق بررسی اثر نا خواسته روند های داده را زمانیکه پردازشگر در حال عمل است پایشگری می کند. بعد از این که نشانی های بازگشت به وسیله ی داده ی اثر نا خواسته شده در طی حملات پوشش داده شده اند یک استثنا ایجاد می شود.

```
int sum_2 (int a, int b)
{
    char buffer [15];
    char shell [80];
    int i;
    ...
    for(i=0; i<80; i++)
      buffer [i] = shell [i];
    return c;
}
void spurious_attack()
{
    while(1)
    {
      console_print_string ("AttackSuccessful!!!");
    }
}
```

شکل 4

ذ) پایشگری حافظه همچنین می تواند به راحتی حملات فرا روند پشته ی بالا را شناسایی کند. در ماژول پایشگری حافظه یک واحد اجرای دستور العمل مجازی وجود دارد. ماژول پایشگری حافظه به وسیله ی بازرسی کران فضای پشته ی یک برنامه قادر است تا به راحتی شناسایی کند که نشانی بازگشت تابع () به طور غیر مجاز پوشش داده شده است و سپس حمله ی فرا روند پشته را تشخیص می دهد.

2. حمله فرا روند توده

ر) حمله ی فرا روند توده یکی از متداول ترین حملات است اما در عین حال به حملات فرا روند میان گیر تعلق دارد. در آزمایشات ما همچنین از یک کد حمله ی فراروند توده به منظور انجام تست حمله بروی پلتفرم آزمایشی که ساخته بودیم استفاده کردیم. تفاوت عمده این است که این کد مخرب، یک ورودی از سوی رابط خارجی نیست، اما در حافظه ی دسترسی تصادفی پویای همگام سیستم (SDRAM)، پیش از این که سیستم در حال اجرا باشد نوشته شده است.

ز) کد حمله ی فرا روند توده که در شکل 5 نشان داده شده به منظور ارائه ی بهتر ساده شده است. ما به شرح زیر کد کلیدی این نوع از حمله را حفظ کردیم. تابع main () به صورت پویا دو بلوک توده را تخصیص می دهد که نشانی های آغازین buf1 و buf2 هستند. فاصله ی بین buf1 و buf2، BUFFER_DISTANCE می باشد. ابتدا buf2 را به وسیله ی فراخوانی تابع memset مقدار دهی اولیه می کند و "A" را به buf2 اختصاص می دهد. سپس buf1 را به وسیله ی فراخوانی تابع memset مقدار دهی اولیه می کند اما طول این واگذاری BUFFER_DISTANCE + OVERSIZE است که از محتوای buf1 طولانی تر است. از آنجاییکه تابع memset مرز buf1 و buf2 را بررسی نمی کند باعث یک فرا روند توده شده و داده ی buf2 را پوشش می دهد.

```
# include<studio.h>
#define BUFSIZE 16
#define OVERSIZE 8
main()
{
   unsigned long BUFFER_DISTANCE;
   char *buf1 = (char*)malloc(BUFSIZE);
   char *buf2 = (char*)malloc(BUFSIZE);
   BUFFER_DISTANCE = (unsigned long) buf2 - (unsigned long buf1;
   printf("buf1=%p, buf2=%p, BUFFER_DISTA|NCE=0x%x(%d)bytes\n",
   buf1, buf2, BUFFER_DISTANCE);
   memset(buf2, 'A', BUFSIZE-1);
   buf2[BUFSIZE-1] = '\0';
   printf("Before overflow: buf2 = %s\n", buf2);
   memset(buf1, 'B', (unsigned int) BUFSIZE + OVERSIZE);
   printf("After overflow: buf2 = %s\n", buf2);
}
```

شکل 5

ژ) نتایج آزمایشی نشان می دهند که استراتژی چندگانه ی ما به راحتی قادر به تشخیص حمله ی فرا روندی توده می باشد. تفاوت در این است که بر خلاف پایشگری روند اطلاعات، تنها ماژول پایشگری حافظه قادر به شناسایی حملات مخرب می باشد. ردگیری روند اطلاعات، امنیت برنامه ها یا داده ی در راه از سوی رابط خارجی را تجزیه و تحلیل نمی کند. این در حالیست که بر طبق این فرض که ردگیری روند اطلاعات نیاز به علامت گذاری داده ی خارجی به عنوان نمونه های غیر قابل اطمینان در هنگامیکه از هر درگاه متصل به CPU منتقل شود دارد، قادر به دفاع در برابر حملات مخرب می باشد. از آنجاییکه کد مخرب حمله ی فرا روند پشته پیش از این که سیستم راه

اندازی شود در SDRAM نوشته شده است، سیاست ردگیری روند اطلاعات هیچ گونه علامت گذاری انجام نمی دهد پس بنابراین قادر به تشخیص حمله نیست.

و) در ماژول پایشگری حافظه، پارامتر های ورودی malloc (طول میان گیر) و مقدار بازگشت (نشانی آغازین میان گیر) در واحد ذخیره سازی اطلاعات کران میان گیر را ذخیره می کند. سپس نشانی مورد هدف هر عملیات حافظه ای را پایشگری می کند. در صورتیکه یک نشانی مورد هدف در میان فضای داده ی توده باشد، نشانی پایه ی نشانی مورد نظر را با نشانی آغازینی که در واحد اطلاعات کران میان گیر ذخیره شده است تا زمانیکه طول میان گیر را پیدا کند تطابق می دهد. سپس طول را با مقدار انحراف نشانی مورد نظر مقایسه می کند. زمانی که طول کمتر از مقدار انحراف است یک هشدار تولید می شود. در چنین موقعیتی طول buf1 کمتر از BUFFER_DISTANCE + OVERSIZE انحراف است و سپس یک سیگنال هشدار دهنده به وسیله ی ماژول پایشگری حافظه آزاد می شود.

ه) طبق نتایج آزمایشی بالا، می توان دید که استراتژی چندگانه ی ما که ترکیبی از پایشگری روند اطلاعات و پایشگری حافظه است قادر به دفاع در برابر انواع حملات نرم افزاری مخرب می باشد. کاربران می توانند به صورت مناسب نیازمندی های پیکربندی ثبات TCR را در ردگیری روند اطلاعات و به منظور کاهش میزان مثبت کاذب یک استراتژی کنترل روند اطلاعات واحد پایین آورند. پس از آن تهدیداتی که توسط ردگیری روند اطلاعات از دست رفته اند می توانند به وسیله ی ماژول پایشگری حافظه شناسایی شوند. بدین ترتیب طرح پیشنهادی ما یک استراتژی امنیتی دو محافظتی سیستم های نهفته را شکل داده است.

مازاد سخت افزار

برای ارزیابی عملکرد طراحیمان، ما تمامی سیستم را به یک پلتفرم نمونه سازی FPGA همراه با یک Xilinx Virtex - 5 FPGA (XC5VFX70T – FFG1136) نگاشت کردیم. ما همچنین تایید تابعی نمونه ی اولیه ی در حال راه اندازی را اجرا کرده ایم. شکل 2 مقایسه ی نسبت های بهره برداری انواع مختلف منابع FPGA را قبل و بعد از یک پارچگی استراتژی چندگانه ی ما، نمایش می دهد. همچنین می توان متوجه شد که مازاد ناحیه تقریباً ناچیز است. در مورد مازاد زمان، گزارش پسا مکان و زمان بندی مسیر نشان می دهد که یکپارچگی استراتژی

چندگانه ی ما همچنین دارای تاثیر جزئی در تاخیر مسیر بحرانی دارد. نتایج فوق نشان می دهند که استراتژی چندگانه ی پیشنهادی ما یک راه حل ایده آل برای پردازشگر های نهفته با هزینه ی سخت افزاری بسیار کم و تقریباً بدون هیچگونه جریمه ی عملکردی می باشد.

	Leon3 with security strategy	Leon3
Number of slice registers	10%	10%
Number of slice LUTs	21%	21%
Number used as logic	22%	21%
Number used as memory	1%	1%
Number of route-thrus	1%	1%
Number of occupied slices	40%	39%
Number with an unused flip-flop	57%	57%
Number with an unused LUT	15%	15%
Number of fully used LUT-FF pairs	26%	26%
Number of bonded	37%	37%
Number of block RAM/FIFO	12%	12%
Number of BUFG/BUFGCTRLs	32%	31%
Number of BSCANs	50%	50%
Number of DCM-ADVs	41%	41%
Number of DSP48Es	3%	3%

جدول 2

نتیجه گیری

در این مقاله ما یک استراتژی پایشگری ایمنی جدیدی از پردازشگر های نهفته را پیشنهاد کردیم. این استراتژی درگیری روند اطلاعات و پایشگری حافظه را در یک راه حل ترکیبی ادغام می کند که ثابت کرده است که موثر تر بوده و قادر به شناسایی حملات مخرب بیشتری نسبت به راه حل های واحد می باشد. به علاوه هزینه ی سخت افزار و جریمه ی عملکردی طراحی ما هر دو بسیار ناچیز هستند. در کار های آینده سعی می کنیم که مکانیزم های ایمنی بیشتری را بر اساس این کار ادغام کنیم و برای اولین بار، بررسی تمامیت داده ی حافظه ی اصلی سیستم نهفته را وارسی خواهیم کرد. بررسی تمامیت داده ی حافظه، روش بسیار موثری در زمینه ی دفاع در برابر حملات تقلیدی، حملات جابجایی و حملات بازپخشی می باشد. امیدواریم که روش بررسی تمامیت داده ی حافظه بتواند همچنین در استراتژی امنیتی ما ادغام شود. در تلاش بعدی سعی خواهیم کرد تا به ترکیبی از کنترل روند اطلاعات ، پایشگری حافظه و بررسی تمامیت داده دست یابیم و یک سیستم دفاعی اینی چند بعدی را در رابطه با سیستم های نهفته ایجاد کنیم .