عنوان: **آشنایی با CLR: قسمت هفتم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۳/۲۸ ۱:۲۵
آدرس: www.dotnettips.info
گروه‌ها: CLR, .NET Framework

کدهای IL و تایید آن ها

ساختار استکی

IL از ساختار استک استفاده می‌کند. به این معنی که تمامی دستور العمل‌ها داخل آن push شده و نتیجه‌ی اجرای آن‌ها pop می‌شوند. از آنجا که IL به طور مستقیم ارتباطی با ثبات‌ها ندارد، ایجاد زبانهای برنامه نویسی جدید بر اساس CLR بسیار راحت‌تر هست و عمل کامپایل، تبدیل کردن به کدهای IL می‌باشد.

بدون نوع بودن(Typeless)

از دیگر مزیت‌های آن این است که کدهای IL بدون نوع هستند. به این معنی که موقع افزودن دستورالعملی به داخل استک، دو عملگر وارد می‌شوند و هیچ جداسازی در رابطه با سیستم‌های 32 یا 64 بیت صورت نمی‌گیرد و موقع اجرای برنامه است که تصمیم می‌گیرد از چه عملگرهایی باید استفاده شود.

Virtual Address Space

بزرگترین مزیت این سیستم‌ها امنیت و مقاومت آن هاست. موقعی که تبدیل کد IL به سمت کد بومی صورت می‌گیرد، CLR فرآیندی را با نام verification یا تاییدیه، اجرا می‌کند. این فرآیند تمامی کدهای IL را بررسی می‌کند تا از امنیت کدها اطمینان کسب کند. برای مثال بررسی می‌کند که هر متدی صدا زده می‌شود با تعدادی پارامترهای صحیح صدا زده شود و به هر پارامتر آن نوع صحیحش پاس شود و مقدار بازگشتی هر متد به درستی استفاده شود. متادیتا شامل اطلاعات تمامی پیاده سازی‌ها و متدها و نوع هاست که در انجام تاییدیه مورد استفاده قرار می‌گیرد.
در ویندوز هر پروسه، یک آدرس مجازی در حافظه دارد و این جدا سازی حافظه و ایجاد یک حافظه مجازی کاری لازم اجراست. شما نمی‌توانید به کد یک برنامه اعتماد داشته باشید که از حد خود تخطی نخواهد کرد و فرآیند برنامه‌ی دیگر را مختل نخواهد کرد. با خواندن و نوشتن در یک آدرس نامعتبر حافظه، ما این اطمینان را کسب می‌کنیم که هیچ گاه تخطی در حافظه صورت نمی‌گیرد.
قبلا به طور مفصل در این مورد ذخیره سازی در حافظه صحبت کرده ایم.

Hosting

از آنجا که پروسه‌های ویندوزی به مقدار زیادی از منابع سیستم عامل نیاز دارند که باعث کاهش منابع و محدودیت در آن می‌شوند و نهایت کارآیی سیستم را پایین می‌آورد، ولی با کاهش تعدادی برنامه‌های در حال اجرا به یک پروسه‌ی واحد می‌توان کارآیی سیستم را بهبود بخشید و منابع کمتری مورد استفاده قرار می‌گیرند که این یکی دیگر از مزایای کدهای managed نسبت به unmanaged است. CLR در حقیقت این قابلیت را به شما می‌دهد تا چند برنامه‌ی مدیریت شده را در قالب یک پروسه به اجرا درآورید. هر برنامه‌ی مدیریت شده به طور پیش فرض بر روی یک appDomain اجرا می‌گردد و هر فایل EXE روی حافظه‌ی مجازی مختص خودش اجرا می‌شود. هر چند پروسه‌هایی از قبیل IIS و SQL Server که پروسه‌های CLR را پشتیبانی یا هاست می‌کنند می‌توانند تصمیم بگیرند که آیا appDomainها را در یک پروسه‌ی واحد اجرا کنند یا خیر که در مقاله‌های آتی آن را بررسی می‌کنیم.

کد ناامن یا غیر ایمن UnSafe Code

به طور پیش فرض سی شارپ کدهای ایمنی را تولید می‌کند، ولی این اجازه را می‌دهد که اگر برنامه نویس بخواهد کدهای ناامن بزند، قادر به انجام آن باشد. این کدهای ناامن دسترسی مستقیم به خانه‌های حافظه و دستکاری بایت هاست. این مورد قابلیت قدرتمندی است که به توسعه دهنده اجازه می‌دهد که با کدهای مدیریت نشده ارتباط برقرار کند یا یک الگوریتم با اهمیت زمانی بالا را جهت بهبود کارآیی، اجرا کند.
هر چند یک کد ناامن سبب ریسک بزرگی می‌شود و می‌تواند وضعیت بسیاری از ساختارهای ذخیره شده در حافظه را به هم بزند و امنیت برنامه را تا حد زیادی کاهش دهد. به همین دلیل سی شارپ نیاز دارد تا تمامی متدهایی که شامل کد unsafe هستند را با کلمه کلیدی unsafe علامت گذاری کند. همچنین کمپایلر سی شارپ نیاز دارد تا شما این کدها را با سوئیچ unsafe/ کامپایل کنید.

موقعیکه جیت تلاش دارد تا یک کد ناامن را کامپایل کند، اسمبلی را بررسی می‌کند که آیا این متد اجازه و تاییدیه آن را دارد یا خیر. آیا System.Security.Permissions.SecurityPermission با فلگ SkipVerification مقدار دهی شده است یا خیر. اگر پاسخ مثبت بود JIT آن‌ها را کامپایل کرده و اجازه‌ی اجرای آن‌ها را می‌دهد. CLR به این کد اعتماد می‌کند و امیدوار است که آدرس دهی مستقیم و دستکاری بایت‌های حافظه موجب آسیبی نگردد. ولی اگر پاسخ منفی بود، یک استثناء از نوع System.InvalidProgramException یا System.Security.VerificationException را ایجاد می‌کند تا از اجرای این متد جلوگیری به عمل آید. در واقع کل برنامه خاتمه میابد ولی آسیبی به حافظه نمی‌زند.

پی نوشت: سیستم به اسمبلی هایی که از روی ماشین یا از طریق شبکه به اشتراک گذاشته می‌شوند اعتماد کامل میکند که این اعتماد شامل کدهای ناامن هم می‌شود ولی به طور پیش فرض به اسمبلی هایی از طریق اینترنت اجرا می‌شوند اجازه اجرای کدهای ناامن را نمی‌دهد و اگر شامل کدهای ناامن شود یکی از خطاهایی که در بالا به آن اشاره کردیم را صادر می‌کنند. در صورتی که مدیر یا کاربر سیستم اجازه اجرای آن را بدهد تمامی مسئولیت‌های این اجرا بر گردن اوست.

در این زمینه مایکروسافت ابزار سودمندی را با نام PEVerify.exe را معرفی کرده است که به بررسی تمامی متدهای یک اسمبلی پرداخته و در صورت وجود کد ناامن به شما اطلاع میدهد. بهتر است از این موضوع اطلاع داشته باشید که این ابزار نیاز دارد تا به متادیتاهای یک اسمبلی نیاز داشته باشید. باید این ابزار بتواند به تمامی ارجاعات آن دسترسی داشته باشد که در مورد عملیات بایندینگ در آینده بیشتر صحبت می‌کنیم.

IL و حقوق حق تالیف آن

بسیاری از توسعه دهندگان از اینکه IL هیچ شرایطی برای حفظ حق تالیف آن‌ها ایجاد نکرده است، ناراحت هستند. چرا که ابزارهای زیادی هستند که با انجام عملیات مهندسی معکوس می‌توانند به الگوریتم آنان دست پیدا کنند و میدانید که IL خیلی سطح پایین نیست و برگرداندن آن به شکل یک کد، کار راحت‌تری هست و بعضی ابزارها کدهای خوبی هم ارائه می‌کنند. از دست این ابزارها می‌توان به ILDisassembler و JustDecompile اشاره کرد.
اگر علاقمند هستید این عیب را برطرف کنید، می‌توانید از ابزارهای ثالث که به ابزارهای obfuscator (یک نمونه سورس باز) معروف هستند استفاده کنید تا با کمی پیچیدگی در متادیتاها، این مشکل را تا حدی برطرف کنند. ولی این ابزارها خیلی کامل نیستند، چرا که نباید به کامپایل کردن کار لطمه بزنند. پس اگر باز خیلی نگران این مورد هستید می‌توانید الگوریتم‌های حساس و اساسی خود را در قالب unmanaged code ارائه کنید که در بالا اشاراتی به آن کرده‌ایم.
برنامه‌های تحت وب به دلیل عدم دسترسی دیگران از امنیت کاملتری برخوردار هستند.