چهار شنبه 19 آبان 1389 - 13:38

حفظ امنيت كدهاي نوشته شده به زبان جاوا

با وجود كارايي زياد و انعطاف‌پذيري بالاي زبان برنامه‌نويسي جاوا، اين ابزار قدرتمند و رو به‌توسعه هنوز نتوانسته جايگاه مناسبي براي پياده‌سازي پروژه‌هاي تجاري در مقايسه با ساير Frameworkهاي موجود بيابد. از جمله دلايل اصلي عدم استفاده از جاوا در بيشتر پروژه‌هاي بزرگ تجاري، مي‌توان به مشكل بازگشت‌پذيري يا Decompile شدن كدهاي نوشته شده، با اين زبان برنامه‌نويسي اشاره كرد كه اين خود به‌تنهايي يک معضل بزرگ امنيتي در مسير توسعه نرم‌افزارهاي تجاري با استفاده از اين ابزار به‌شمار مي‌آيد.

با وجود كارايي زياد و انعطاف‌پذيري بالاي زبان برنامه‌نويسي جاوا، اين ابزار قدرتمند و رو به‌توسعه هنوز نتوانسته جايگاه مناسبي براي پياده‌سازي پروژه‌هاي تجاري در مقايسه با ساير Frameworkهاي موجود بيابد. از جمله دلايل اصلي عدم استفاده از جاوا در بيشتر پروژه‌هاي بزرگ تجاري، مي‌توان به مشكل بازگشت‌پذيري يا Decompile شدن كدهاي نوشته شده، با اين زبان برنامه‌نويسي اشاره كرد كه اين خود به‌تنهايي يک معضل بزرگ امنيتي در مسير توسعه نرم‌افزارهاي تجاري با استفاده از اين ابزار به‌شمار مي‌آيد.

برنامه‌هاي نوشته شده در جاوا به‌دليل استفاده از واسط زمان اجراي Java Runtime Environment يا به‌طور اختصار JRE براي مهيا كردن بستري مناسب براي جلوگيري از كامپايل مجدد كد منبع و همچنين حفظ قابليت جابه‌جايي و استفاده در پلتفرم‌هاي مختلف، پس از كامپايل به‌جاي تبديل شدن به زبان ماشين يا همان «صفر» و «يک» به قالب خاصي از داده‌ها به‌نام Byte Code تبديل مي‌شود كه اين خروجي همان قالب قابل فهم و اجرا براي JRE است.

همين موضوع باعث مي‌شود كد برنامه‌هاي نوشته شده به زبان جاوا را با استفاده از متدهاي مهندسي معكوس تا حدود زيادي بازيابي كرد. اين مشكل به‌حدي جدي است كه شركت توسعه‌دهنده زبان جاوا يعني سان‌ميكروسيستمز، اقدام به ارائه ابزاري به‌نام Obfuscator براي مبهم‌سازي كد منبع نوشته شده از طريق تزريق كدهاي اضافي به‌زبان جاوا كرده تا پس از انجام عمليات ديكامپايل، كاربران نتوانند به‌راحتي قطعات مختلف كد حاصل را درك و تفسير كنند.

اجراي اين برنامه به‌گونه‌اي است كه در بدنه توابع و كلاس‌ها در نتيجه نهايي و خروجي اصلي برنامه تغييري را در پي نخواهد داشت. البته استفاده از اين متد كه به‌صورت توكار (يا درون‌ساخت Built-in) هم در كتابخانه‌هاي جاوا و هم توسط IDE پيش‌فرض شركت سان، يعني Netbeans نيز پشتيباني مي‌شود، نمي‌تواند تمام آنچه كه برنامه‌نويسان براي ايجاد امنيت بيشتر به آن نياز دارند را تامين كند. گذشته از اين، استفاده از روش مذكور سبب افزايش حجم فايل اجرايي پس از كامپايل، كندي در بارگذاري و اجراي برنامه، سخت‌تر شدن انجام عمليات اشكال‌زدايي (ديباگ)، بالا بردن ميزان استفاده از حافظه اصلي و تحميل بار مضاعف بر دوش پردازنده به‌دليل استفاده مكرر از حلقه‌هاي تكرار و تصميم‌گيري مي‌شود.

در اين ميان روشي كاملا ابتكاري توسط برخي برنامه‌نويسان براي هر چه بالاتر بردن ضريب امنيت در كدهاي جاوا يا حتي ديگر زبان‌هاي برنامه‌نويسي كه از يک قالب همسان با فريمورك جاوا بهره مي‌برند، ارائه شده است. اين روش كه به متد رمزگذاري (Encryption) معروف است، هم‌اكنون به‌عنوان يكي از بهترين و موثرترين راه‌حل‌هاي مطرح‌شده براي حفاظت از كدهاي جاوا شناخته مي‌شود. تفاوت عمده اين روش با ديگر متدها در اين است كه روش رمزگذاري از آغاز تا پايان بايد توسط برنامه‌نويس و گروه توسعه‌دهنده پروژه پياده‌سازي شود؛ برخلاف روش Obfuscator كه براي مبهم كردن كد منبع از الگوهاي محدود و تكراري استفاده مي‌كند.

شايد اين موضوع در نگاه اول كمي دشوار به‌نظر برسد اما كارايي واقعي روش رمزگذاري به اين موضوع وابسته است كه منطق رمزنگاري در برنامه شما مي‌تواند كاملا متفاوت با تصور ديگران يا به‌عبارت ديگر كاملا ابتكاري و شخصي باشد.

رفتار متد رمزگذاري در اين روش فرض بر اين است كه تمام يا تعدادي از كلاس‌هاي نوشته شده، به‌جز كلاس اصلي پروژه و كلاسي كه قرار است عمليات كدبرداري ديگر كلاس‌ها را انجام دهد، با استفاده از يک برنامه واسط و يک كليد اصلي به مجموعه‌اي از كاراكترهاي غيرقابل فهم و نامفهوم تبديل شده است. سپس كلاس‌هاي كدشده در مسير مناسب به پروژه افزوده مي‌شوند.

كلاس اصلي پروژه بايد بتواند با استفاده از توابعي كه براي رمزگشايي كلاس‌هاي كدشده توسط برنامه‌نويس طراحي شده‌اند، كلاس‌هاي مذكور را در حافظه ديكد و بازسازي كند تا در محل و زمان مناسب فراخواني و استفاده شوند. به اين ترتيب حتي بعد از ديكامپايل شدن برنامه نيز كاربران با انبوهي از كلاس‌هاي رمزگذاري شده و كاراكترهاي غيرقابل فهم روبه‌رو خواهند شد. **نحوه پياده‌سازي** پيش از هر چيز بايد گفت كه بهترين حالت ممكن براي پياده‌سازي روش رمزگذاري، زماني است كه تمام برنامه‌نويسي‌هاي مربوط به كلاس‌ها و متدهايشان به‌صورت كامل انجام شده و پروژه آماده كامپايل شدن نهايي باشد. همان‌طور كه پيشتر نيز گفته شد، براي پياده‌سازي اين متد ما نياز به يک برنامه واسط براي رمزنگاري كلاس‌هاي مورد نظرمان با استفاده از يک كليد اصلي و بر اساس الگويي كه برنامه‌نويس مشخص كرده است داريم. براي نمونه، برنامه ساده‌اي بنويسيد كه تمام كاراكترهاي يک فايل از نوع كلاس جاوا را به‌عنوان ورودي دريافت كرده و با يک مقدار به‌عنوان كليد اصلي XOR كند. سپس بايد كلاس يا تابع مستقلي براي كدبرداري كلاس‌هاي رمزگذاري‌شده با استفاده از همان كليد اصلي نوشته شود تا در هنگام اجراي برنامه بتوانيم با فراخواني آن، كلاس‌هاي كد شده را در حافظه رمزگشايي و دوباره‌سازي كنيم و سرانجام برنامه بدون هيچ مشكلي بتواند اجرا شود. توجه داشته باشيد كه كليد اصلي به‌عنوان يک عنصر حياتي براي حفظ امنيت و حتي ضامن اجراي صحيح برنامه، مي‌تواند براي هر كلاس به‌صورت متفاوت توليد شود يا از مجموعه چند عمليات پيچيده رياضي روي مقادير گوناگون به‌دست آمده باشد، يا حتي روي يک قفل سخت‌افزاري ذخيره شود تا برنامه بدون وجود آن اصلا اجرا هم نشود.

براي پنهان ماندن الگوريتم كدبرداري كلاس‌ها نيز مي‌توانيد از فايل‌هاي كتابخانه‌اي پويا و استاندارد (براي ويندوز پسوند dll و براي گنولينوكس پسوند so) نوشته‌شده با زبان C استفاده كنيد. البته در اين حالت پياده‌سازي الگوريتم رمزگشايي بايد در همين كتابخانه‌ها انجام شود و همين‌طور براي حفظ قابليت اجرا و انتقال روي ساير پلتفرم‌ها نيز بايد كتابخانه‌هاي مورد نياز و قابل استفاده روي ساير پلتفرم‌ها را هم ايجاد كرده و همراه با ديگر منابع به پروژه بيفزاييد تا بتوانيد با توجه به نوع سيستم‌عامل، فايل كتابخانه مورد نظر را بارگذاري كرده و توابع مورد نياز براي رمزگشايي را از داخل اين فايل‌ها فراخواني كنيد. **نتيجه‌گيري** روش يا متد رمزگذاري راه‌حل مناسبي براي حفظ امنيت كد برنامه‌هاي نوشته شده به زبان جاواست كه بر خلاف ديگر روش‌هاي معمول در روند عادي اجراي برنامه خللي وارد نمي‌كند و در صورتي كه به‌درستي پياده‌سازي شود حتي مي‌توان با اتكا به آن از فريمورك جاوا، بدون ترس از ديكامپايل شدن برنامه در پروژه‌هاي تجاري بزرگ و مستقل از سكو نيز استفاده كرد و با خيال راحت از مواهب بسيار زياد جاوا برخوردار شد. **منابع**

http://java.sun.comhttp://www.javaworld.comhttp://www.devarticles.com

محمد جواد احمدي