دیوار آتش لایه هفت (سطح برنامه)

آرش طاهرکلاته 882980

این نوع از دیواره آتش، بسته‌ها در لایه برنامه را واکاوی میکند برای بررسی تطابق با تعدای امضا، مثلاً چک کردن URL میتواند در اینجا صورت بگیرد.
به واسطه اینکه این برنامه در لایه هفت کار می‌کند، از محتوای رد و بدل شده مطلع است و نسبت به دیوارهای آتش لایه سه و چهار میتواند کنترل بهتر و دقیقتری روی ترافیک داشته باشد.

برای مثل اگر ادمین تصمیم به رد کردن درخواست‌های HTPP بگیرد، بستن پورت ۸۰ (که پورت معمول برای برقراری این ارتباط است) کافی نخواهد بود، که نیاز داریم بسته‌ها روی پورت‌های دیگر هم مشاهده شوند.

از آنجا که تا زمانی که ارتباط TCP برقرار نشود، داده‌های سطح برنامه بین دو سمت منتقل نمیشوند، یکی از مکان‌های متداول برای راه انداختن این نوع از دیوار آتش بر روی پروکسی سرور است. به این صورت که ایستگاه ابتدا به پروکسی سرور متصل میشود، و پس از آن، زمانی که درخواست خود را میفرستد، این درخواست در پروکسی سرور برای تطابق با الگوهای از پیش تعیین شده بررسی شود. اگر مشکلی نداشته باشد که Forward میشود، در غیر اینصورت Reject میشود.
تمامی پروتکل‌ها از قبیل FTP ، SMTP، POP۳ و غیره به اینصورت میتوانند فیلتر شوند.

در مقایسه با دیوارهای آتشی که در لایه Transport کار میکنند و تمام وظیفه آنها محدود به بررسی قسمت خاصی از بسته است، پیاده سازی و طرز کار کاوش محتوا در لایه برنامه از پیچیدگی‌های بالاتری برخوردار است. در اینجا ما باید همانند برنامه مقصد،

بسته را گرفته و به طور کامل پارس کنیم. محتوای آن را خوانده و در برابر با مجموعه امضاهایی که داریم قرار دهیم. اندازه بسته‌ها، اندازه امضاها و مکان این امضاها هیچ کدام ثابت و قابل پیشبینی نیست که این موضوع بر محاسباتی سنگینی را به برنامه تحمیل می‌کند.

علاوه بر اینها، دیوار آتش کاربردی که به صورت پروکسی سرور پیاده سازی میشود، مشکل این را دارد که کاربر باید به صورت دستی پروکسی را تنظیم کند و دیوار آتش دیگر به صورت شفاف کار نخواهد کرد .

یک نوع دیگر برای پیاده سازی این برنامه، MAC (Mandatory Access Control) است. در این روش که بر روی سیستم کاربر کار خواهد کرد، ورودی‌ها و خروجی‌های برنامه، به صورت درخواست‌های سیستمی آن، تک تک مورد بررسی قرار خواهند گرفت. در اینجا ما دیگر کاری با Network Stack نداریم و خود برنامه‌ها مستقیما هدف ما هستند.

استراتژی کلی به این صورت است که آیا یک برنامه خاص اجازه دسترسی به این منبع خاص را دارد یا خیر. برای برقراری ارتباطت شبکه، دیوار آتش اصطلاحا از طریق Hooking جلوی راه درخواست‌های ساخت سوکت میایستد و در صورتی که این پردازش اجازه برای سخت سوکت نداشته باشد، جلوی آن را می‌گیرد. در کنار این برنامه‌ها معمولا یک برنامه در ارتباط با فیلتر کردن بسته‌ها هم قرار دارد تا هم Per-Process-Filteringداشته باشیم و هم Per-Packet-filtering .

Port Knocking

این روش، یکی از روش‌های هوشمند برای محدود کردن دسترسی‌ها با پورت‌ها بر روی سرور است. به نوعی می‌توان آن را نوعی رمز گذاری در نظر گرفت.

ابتدای کار تمام پورت‌های روی سیستم از خارج به صورت بسته دیده میشوند. حال ما تعدادی در خواست برای تعدادی از پورت‌ها می‌فرستیم. مثلا به ترتیب پورت‌های ۱، ۲ و ۳. به ازای هیچ کدام از این درخواست‌ها پاسخی از سمت دیوار آتش دریافت نمی‌کنیم، اما سرانجام، چنانچه به ترتیب درست و از پیش تعیین شده این کار را انجام داده باشیم، پس از آن یکی از پورت‌ها روی سرور باز شده و اجازه دسترسی به برنامه‌های اجرائی بر روی سرور از طریق آن به ما داده میشود.

این وظیفه معمولا در قالب یک Monitoring Daemon که وظیفه آن قرار گرفتن در سر راه درخواست‌ها به منظور جمع آوری اطلاعات است صورت می‌گیرد. توجه شود این برنامه به درخواست‌ها پاسخ نمیدهد و فقط دنبال آنها را نگاه میدارد تا در صورت تطابق با الگویه از پیش تعیین شده، اجازه باز شدن پورت را بدهد.