وال‌پکر، ابزاری برای شناسایی خودکار آسیب‌پذیری‌ها مبتنی بر شباهت کدمنبع

محمد حدّادیان – mhadadian@ce.sharif.edu

مقدمه

وجود آسیب‌پذیری‌ها در برنامه‌ها و سامانه‌های مختلف، منبع اصلی حملات کامپیوتری است. طبق آمار، تعداد برنامه‌هایی که نسبت به یک آسیب‌پذیری کشف‌شده همچنان آسیب‌پذیر هستند، بسیار زیاد است. علت این موضوع مشخص است، ما برنامه‌های زیادی داریم که از کدهای مشترک استفاده می‌کنند و حال آن‌که این کدها آسیب‌پذیر باشند، منجر به آسیب‌پذیری خیل عظیمی از برنامه‌ها می‌شود. چه از مرورگرهای با موتور یکسان همچون سافاری و کروم، و چه سیستم‌عامل‌هایی که از یک کد آسیب‌پذیر عمومی در خود استفاده کرده‌اند.
کارکرد این ابزار در دو اصل پایدار است، نخست ویژگی‌هایی که یک رفع‌آسیب‌پذیری باید داشته باشد و سپس تشخیص شباهت‌ها در کدمنبع برنامه‌های مختلف که باید با الگوریتم‌های متفاوتی تشخیص داده شوند زیرا یک الگوریتم به‌تنهایی کارا نخواهد بود.

چالش‌ها

در طراحی این ابزار ما با دو چالش روبرو هستیم. نخست آن‌که الگوریتم‌های تشخیص شباهت در کدها خیلی دقیق نیستند و بعضاً برای تشخیص آسیب‌پذیری مناسب نخواهند بود چه بسا تغییر نام متغیرها یا اضافه کردن خطوط خالی میان آن‌ها، الگوریتم را به اشتباه می‌اندازد. دوم پایگاه‌داده‌ی مناسبی برای کدهای آسیب‌پذیر وجود ندارد تا ما بتوانیم از روی آن بخش‌های آسیب‌پذیر یک برنامه یا سیستم‌عامل را شناسایی کنیم. برای حل چالش دوم، ما از مجموعه‌ای از برنامه‌های منبع‌باز استفاده کردیم که با توجه به یک CVE، آسیب‌پذیر بوده‌اند. سپس نسخه‌ی قبل و بعد از رفع این آسیب‌پذیری را باهم مقایسه کردیم تا به یک تکه کد آسیب‌پذیر و نحوه‌ی رفع‌آسیب‌پذیری آن برسیم. همچنین برای چالش بخش اول، از مجموعه‌ای از الگوریتم‌های تشخیص شباهت استفاده کرده‌ایم که هرکدام برای یک نوع خاص از آسیب‌پذیری کارا خواهد بود. این‌که برنامه‌ی ما از کدام الگوریتم برای چه آسیب‌پذیری استفاده کند، توسط یک فاز یادگیری، به وال‌پکر آموزش داده خواهد شد.

نحوه کارکرد دو فاز

همان‌طور که گفته شد، برنامه‌ی ما از دو فاز تشکیل شده است. یک فاز یادگیری برای انتخاب بهترین الگوریتم کشف شباهت، و یک فاز تشخیص آسیب‌پذیری. برای فاز یادگیری، الگوریتم ما بدین صورت کار می‌کند: هر تغییر در کدمنبع یک برنامه پیش و پس از رفع آسیب‌پذیری را به پنج دسته تقسیم می‌کنیم. هر یک از این دسته‌ها مشخص‌کننده‌ی نوع خاصی از تغییر هستند. به‌عنوان مثال تغییرات Non-substantive بیان‌گر تغییر در کامنت‌ها، فاصله‌ها و چیزهایی از این دست است یا تغییرات Function-level بیان‌گر تغییر در توابع استفاده شده است. پس از آن‌که برای هر تغییر یک برنامه پیش و پس از رفع آسیب‌پذیری این پنج دسته را مشخص کردیم، به‌ازای هر کدام یک بردار تغییرات تشکیل می‌دهیم که از این بردار در بخش دوم برای تعیین الگوریتم مناسب استفاده می‌کنیم.

حال برای تشخیص یک الگوریتم مناسب، آن الگوریتم را بر روی رفع-آسیب‌پذیری‌هایی که جمع‌آوری کردیم اجرا می‌کنیم و بردارهای تغییرات حاصل از الگوریتم را با بردار تغییرات واقعی آن پچ مقایسه می‌کنیم. سپس الگوریتم کارا را الگوریتمی تعریف می‌کنیم که اولاً میان این برنامه پیش و پس از تغییر، تمایز قائل شود و ثانیاً بردار تغییرات آن بیشترین شباهت را به بردار تغییرات حقیقی داشته باشد.

نتایج

ما برای ارزیابی این برنامه، ۲۴۶ آسیب‌پذیری را انتخاب کردیم، و هدفمان آن است که با دادن این آسیب‌پذیری‌ها به عنوان ورودی به برنامه، وجود یا عدم‌وجود آن‌ها را در یک برنامه‌ی هدف مشخص کنیم. خروجی برنامه ۴۰ آسیب‌پذیری بود که از وجودداشتن ۱۸تای آن‌ها بی‌خبر بودیم و ۲۲ آسیب‌پذیری دیگر به‌طور متوسط ۷.۳ ماه پس از انتشار عمومی آسیب‌پذیری توسط شرکت سازنده رفع شده بودند.