Programok sebezhetőségeinek detektálása statikus forráskódelemzéssel

Ságodi Zoltán
II. évf. programtervező informatikus

Témavezető: Dr. Siket István

SZTE TTIK Szoftverfejlesztés Tanszék

Az informatika hajnalán a programoknál nem kellett nagy figyelmet szentelni a programok biztonságára, mivel nem széles körben és mindenki által elérhető rendszereket fejlesztettek. Azonban programok mindennapos használata, de még inkább az Internet „megjelenése" után már kulcsfontosságú lett a programok biztonsága. Már a kezdetekben is voltak igen komoly biztonsági rések, mint például a Windows 95-ben lévő mappabejárást lehetővé tevő hiba, ami a támadónak lehetőséget adott arra, hogy megosztott tartalmon kívüli mappákhoz és fájlokhoz is hozzáférhessen. Sajnos annak ellenére, hogy egyre többet költünk a sebezhetőségek kiküszöbölésére, mindig akadnak olyan hibák, melyek komoly kockázatot jelentenek, és amelyeket a támadók ki is használnak. 2014-ben a világhírű Sony cég adatait lopták el, és több ezer felhasználónév-jelszó páros került illetéktelen kezekbe. De említhetnénk a hazai példát is, amikor a BKK-nak a T-Systems által fejlesztett e-jegyrendszeréről derült ki, hogy komoly biztonsági réseket tartalmaz. Ilyen hibák alól azonban még nagyobb, köztudatban biztonságosnak hitt rendszerek sem mentesek, csak nem mindig jut el a felhasználókig az információ (vagy csak a javítás után). Minden évben jelennek meg olyan hibák, melyeket a támadók ki tudnak használni, de megoldás még nem született rá, és csak lassan halad a javításuk.

Ezen példák is alátámasztják, hogy a szoftverek sebezhetőségének feltárásához több különböző módszert is érdemes használni a minél jobb eredmények elérése érdekében. Az egyik lehetőség a manuális tesztelés, amely különböző módokon alkalmazható, és sokféle hibát lehet vele megtalálni. Azonban nagyon sok emberi erőforrásra van szükség a végrehajtásához, így igen költséges, és sokszor szaktudást is igényel. Viszont vannak olyan tipikus hibák is, amelyek felismerésére léteznek automatikus megoldások. Ez azt jelenti, hogy más programok segítségével – akár statikusan, azaz csak a forráskódot átvizsgálva, akár dinamikusan, azaz a programot futtatva – megtalálhatunk különböző hibákat. Ezen módszerek előnye, hogy a kezdeti befektetés után minimális emberi erőforrást és időt igényelnek, azaz olcsók és gyakran, akár naponta újra futtathatóak. A statikus elemzés alapja, hogy csupán a forráskód átvizsgálásával olyan kódrészeket azonosíthatunk be, amelyek nagy valószínűséggel hibához vezetnek. A dolgozatban a statikus elemzéssel megtalálható hibák közül a biztonsági hibákra koncentrálunk. Az ilyen típusú hibák már össze vannak gyűjtve a különböző biztonsági hiba-adatbázisokban, melyek közül az egyik legelterjedtebb az OWASP (Open Web Application Security Project) által összegyűjtött és karban tartott adatbázis. Az OWASP nemcsak a sebezhetőségek listáját adja meg, hanem leírást biztosít ezek kivédésére, valamint pozitív és negatív példákat is a tartalmaz, ami alkalmas többek között arra, hogy az automatikus tesztelő programokat kiértékeljük és összehasonlítsuk. A hibák listáját az OWASP adatbázisából gyűjtöttem össze, míg az elemzéshez az SZTE Szoftverfejlesztés Tanszéke által fejlesztett OpenStaticAnalyzer (`https://github.com/sed-inf-u-szeged/OpenStaticAnalyzer`) nyílt forráskódú statikus forráskódelemző keretrendszert használtam.

Az OpenStaticAnalyzer keretrendszerhez már létezik egy VulnerabilityChecker modul, amely képes bizonyos OWASP által javasolt sérülékenységi hibák megtalálására. Ezt egészített ki úgy, hogy további sérülékenységi szabályellenőrzőket valósítottam meg (Weak Randomness, Weak Hash és Crypto, Secure Cookie, XPath Injection és Trust Boundary), illetve bővítettem a már meglévő részeket a Java programozási nyelv fejlődését követve. Az OWASP teszteken az eredményeket az elvárt eredményekkel összevetve jó, bizonyos esetegben 100%-os sikert értem el. Teszteltem valós projekteken is, és valódi hibákat is találtam.