Detekcia malvéru pomocou hlbokého učenia založeného na vízii *

Jakub Martinak
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xmartinakj@stuba.sk

11. oktober 2021

Abstrakt

Keďže v 21. storočí je čím ďalej tím viac potreba chrániť naše elektronické zariadenia pred nebezpečným malwarom tak sa chcem zamerať na túto tému v oblasti informačnej bezpečnosti a implementovanie strojového učenia do nej. V tomto článku na začiatku objasním modelovanie v softwarovom inžinierstve a ako to súvisí s mojou témou a neskôr sa budem zameriavať na presnosť určenia či sa jedná o malware alebo nie pomocou Deep Learning Vision a taktiež budem rozoberať problémy a výhody spojené s touto formou detekcie, výhody a nevýhody klasickej detekcie malwaru a jeho limity oproti strojovému učeniu.

1 Úvod

Túto tému som si vybral z dôvodu nárastu kybernetických útokov vo svete a následnej obrane voči nim pomocou moderných technológií a využitím umelej inteligencie. V prvej časti článku sa budem venovať modelovanie v softwarovom inžinierstve 2. Druhá časť sa bude zaoberať nedávnemu pokroku v kybernetickej bezpečnosti 3, z ktorej následne prejdem na hlavnú tému tohto článku Detekcia malwaru pomocou hlbokého učenia založeného na vízii 4. Záverečné poznámky prináša časť **??**.

2 Modelovanie v Softwarovom inžinierstve

Softvérové modelovanie je oveľa viac ako len algoritmus v programe alebo samotná funkcia, malo by poukazovať na celý softvérový desing, vrátane rozhrania, interakcií s iným softvérom a všetky softvérové funkcie. Pre objektovo-orientované programy, je používaný objektovo modelovací jazyk ako UML, ktorý je použitý na vytvorenie softvérového dizajnu. Takmer vo všetkých prípadoch je použitý určitý druh modelovacieho jazyka na vytvorenie dizajnu. Tento prístup umožňuje dizajnérovy programu vyskúšať rôzne dizajny a vyvodiť, ktorý bude

*Semestrálny projekt v predmete Metódy inžinierskej práce, ak. rok 2021/22, vedenie: Fedor Lehocki

najlepší ako finálne riešenie. Je to podobný proces ako pri navrhovaní štruktúry a dizajnu domu, začne sa náčrtom skice poschodí a rozložením izieb. Kreslenie je v tomto prípade modelovací jazyk a výsledkom toho je návrh, ktorý môže byť použitý ako finálny dizajn. Následne sa bude pokračovať upravovaním náčrtu až náš návrh nebude obsahovať všetky naše potreby, jedine vtedy by sme mali začať s využivaním materiálu, v našom prípade písaním kódu. Výhodou dizajnovania programu použitím modelovacieho jazyka je nájdenie problémov v skorom štádiu a jeho vyriešením bez nutnosti menenia kódu.

3 Nedávne pokroky v Kybernetickej Bezpečnosti

3.1 Pokročilá detekcia Malvéru

Napriek pokrokom v metodológií programovania, tak táto metodológia má stále veľký vplyv na bezpečnosť softvéru. Komunita zaoberajúva sa kybernetickou bezpečnosťou, sa hlavne sústredí na detekciu samotného malvéru. Práve vtedy keď všetkých záujem je v zamedzení zneužitia chýb, tak existuje zhoda, že pri extémne komplexnom a časovo náročnom kóde to je priam nemožné zabrániť takýmto chybám.

Kým klasivké metódy detekcie malvéru spočívajú v zhode aplikačného kódu bez zmien, tak moderné metódy sú založené na kontrole správania, aby sa zistilo či ide o neprijateľnú aktivitu alebo nie. Kontrola správania je možná, vďaka dynamickému zabezpečeniu virtuálnych mašín, kde je následne možné bezpečne spustiť nebezpečný súbor. Bez takýchto virtuálne uzavretých prostredí by takáto analýza bola príliš nebezpečná pre produkčné systémy.

Moderný výskum v detekcií malvéru zahŕňa strojové učenie, ktoré je nápomocné v identifikovaní škodlivého kódu na báze vzoriek. Tak isto ako sú AI-Systémy neustále hústené obrázkami rôznych zvierat aby sa ich naučili rozoznávať, tak to isté sa deje aj v kybernetickej bezpečnosti len namiesto fotiek zvierat je vkladaných veľké množstvo öbrázkov"súborov, ktoré sú infikované malvérom. Techniky hlbokého učenia využívajú paralelizmus na vylepšenie efektivity algoritmov.

3.2 Dospelosť softvérového procesu

Mnohí odborníci v softvérovej bezpečnosti sa zhodli, aby sa namiesto priamej kontroly softvéru na prítomnosť škodlibého softvéru sústredili na samotný proces tvorby tohto softvéru a podľa toho skômali zraniteľnosti. Táto teória je do značnej miery založená na skúsenostiach, z toho dôvodu, že dobrý kód pochádza od dobre vyškolených vývojárov pracujúcimi s najmodernejšími technológiami v dobre organizovaných vývojových prostrediach.

Presne naopak zlý kód pochádza z rúk slabo vyškolených vývojárov, ktorí pracujú v málo organizovaných vývojóvych prostrediach a s nemodernými technológiami, ktoré už nie sú podporované a pravidelne aktualizované. Z tohto vyplývajú určité modely vyspelosti, ktoré nám vedia prepojiť stupeň bezpečnosti s kvalitou procesu.

To má za následok pozitívny vedľajší účinok v podobe zvýšenej bezpečnosti pre celý kód, ktorý sa objaví v procese. Bežné metódy požadované v takýchto procesoch zahŕňajú automatizáciu, pravidelné penetračné testovanie a správne postupy aktualizácie a údržby softvéru.

3.3 Kontrola a skenovanie softvéru

Môže sa zdať, že problém vlastne nejestvuje [1], ale bolo dokázané, že to tak nie je [2, 3]. Napriek tomu, aj dnes na webe narazíme na všelijaké pochybné názory [4]. Dôležité veci možno *zdôrazniť kurzívou*.

4 Detekcia malwaru pomocou hlbokého učenia založeného na vízii

Literatúra

[1] James O. Coplien. *Multi-Paradigm Design for C++*. Addison-Wesley, 1999.

[2] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Staged configuration through specialization and multi-level configuration of feature models. *Software Process: Improvement and Practice*, 10:143–169, April/June 2005.

[3] Krzysztof Czarnecki and Chang Hwan Peter Kim. Cardinality-based feature modeling and constraints: A progress report. In *International Workshop on Software Factories, OOPSLA 2005*, San Diego, USA, October 2005.

[4] Carnegie Mellon University Software Engineering Institute. A framework for software product line practice—version 5.0. `http://www.sei.cmu.edu/productlines/frame_report/`.