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 malvérom 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 softvérovom 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 malvéru 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 využívaním 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

Medzi najtradičnejšie prostriedky na zlepšenie softvérovej bezpečnosti patrí priama kontrola kódu, niekedy aj pomocou nástrojov na skenovanie kódu. Pokračujúce používanie manuálnej kontroly kódu je v softvérovej komunite diskutované, mnoho ľudí, hlavne tradicionalisti trvajú na tom, že ľudská kontrola je nevyhnutná na vyprodukovanie najkvalitnejšieho a najbezpečnejšieho produktu. Pri rýchlych cykloch ako sú v prostredí DevOps nie je dostatok času na ľudskú kontrolu zdrojového kódu. Takže jediná možnosť bolo zautomatizovať skenovanie kódu, čo sa už stalo normou v týchto prostrediach, má to svoje pro a proti.

Vzhľadom na to, že to kontroluje počítač tak tam vzniká riziko nezachytenia chyby, ktorá by bola ľahko odhaliteľná človekom, ale v prípade komponentov, ktoré už raz boli úspešne skontrolované a zoskenované, tak to urýchľuje skenovanie, pretože tieto komponenty nie je potreba skenovať znovu.

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

4 Detekcia malwaru pomocou hlbokého učenia založeného na vízii

4.1 Úvod

Detekcia a klasifikácia malvéru je jedným z najväčších problémov v kybernetickej bezpečnosti. Metódy založené na báze podpisov, sú efektívne iba proti už známim malvérom, ale sú veľmi neefektívne proti neznámim malvérom. Autori malvérov používajú techniky ako šifrovanie, packovanie, atď. na už známom malvéry aby predišli detekcií, čo má za následok vačšie množstvo nového malvéru. Súbory s rovnakým malvérovým správaním zapadajú do rovnakej rodiny malvéru. Tieto súbory su neustále upravované používaním rozličných taktík, čo ich robí veľmi rozličnými.

Tieto množstvá súborov sú neskôr zaradené do ich reprezentatívnej rodiny pomocou efektívneho klasifikovania a procesu detekcie malvéru. Veľa exitujúcich klasifikátorov malvéru extraktuje zásadné vlastnosti zo súborov malvéru a trénuje strojový model s týmito vlastnosťami. Trénovaný model vie potom rozoznať rozdiel medzi malvérom a cleanvérom. Klasifikátory čelia mnoho výzvam, pretože taktiky ako šifrovanie, packing a iné jemné úpravy malvéru, môžu spôsobiť únik pred klasifikátorom, čiže takýto malvér nemusí byť zachytený. Vlastnosti, ktoré sú extrahované z týchto upravených malvérov, nevykazujú žiadne dôkazy

pre klasifikátor a teda toto spôsobí nezachytenie malvéru.

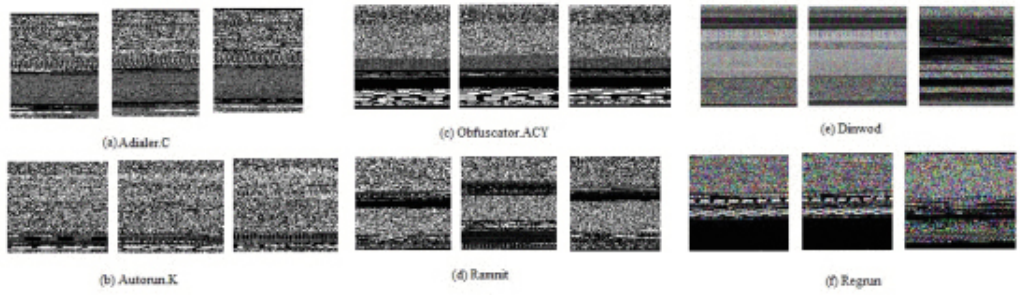
Potreba priblížiť detekciu malvéru do bodu kde budu extrahované robustné časti akéhokoľvek upraveného skriptu malvéru. Ďalšie varianty existujúceho malvéru by mali byť už zachytené týmto systémom.

Sú tu analyzované spustiteľné binárne súbory s malvérom a cleanvérom použitím techniky založenej na analýze videnia. Autori malvéru znovu používajú rovnaké segmenty kódu na vygenerovanie nových variant malvéru. Vizualizácia malvéru bola nedávno použitá ako alternatíva a efektívny prístup k malvérovej analýze. Podobnosti variánt nebezpečného kódu sú vizualizované a identifikované. Každá skupina malvéru vystavila určitý vzorec v malvérových obrázkoch.

Tieto vzorce v obrázkoch sú veľmi podstatné vo vizuálnych podobách malvéru, ktorý zapadá do jednotlivých skupín. Ďalšia výhoda Analýzy založenej na vízií je, že nie je potreba dynamickej exekúcie binárnych súborov. Vzorce vybraté z obrázkov jednotlivých malvérov sú potom následne použité pri trénovaní klasifikátora. Tieto vlastnosti môžu fungovať ako prezrádzajúci faktor na detekovanie výskytu vzorcov v obrázku a teda pomôže nájsť malvér a jeho nové varianty.

Binárne súbory s malvérom sú konvertované do 8 bitových vektorov pozostávajúcich z reťazca núl a jednotiek a sú organizované do dvojdimenzálnej matice formujúcej obrázky v šedých farbách. Autorovia týchto malvérov zmenia malú časť binárneho súboru a obrázky tieto malé zmeny zachytia v globálnej štruktúre. Obrázok 1 zobrazuje niekoľko vzorcov v obrázkoch malvéru v troch datasetoch. Podobnosť vo vzorcoch je vidno viac v jedlotivých skupinách malvéru rovnakej rodiny.

Obr. 1: Podobnosť obrázkov jednotlivých rodín malvéru



Ďalší krok v procese je extrahovať vlastnosti týchto vzorcov z obrázkov a trénovať klasifikátor s týmito vlastnosťami. Techniky Data miningu a strojového učenia sú použité na vytvorenie inteligetnej detekcie malvéru a systémy klasifikátorov. Hlboké neurónové siete dosiahli veľký úspech v rôznych odvetiach, najmä v odvetí počitačového videnia. Aj keď modely hlbokého učenia sú výkonné, tak majú určité limity v realnych úlohach detekcie, hlavne v bezpečnostných doménach. Efektivita detekcie nekatogarizovaného malvéru a prúd zero-day, je malá.

Modely hlbokého učenia sú trénované na spätnom šírení, kde je model trénovaný

použitím stromového učenia. Keď zoberieme v potas všetky faktory, tak najlepší návrh je použiť algoritmus deep forest, ktorý má mnoho výhod oproti exitujúcim modelom strojového a hlbokého učenia. Má zlepšenú presnosť detekcie a nízku výpočtovú jednotku.

Literatúra

[1] Edward Amoroso. Recent progress in software security. 35(2):11–13, March 2018.

[2] Joanne M. Atlee, Robert France, Geri Georg, Ana Moreira, Bernhard Rumpe, and Steffen Zschaler. Modeling in software engineering. IEEE, May 2007.