Detekcija pakiranih datoteka

Roko Kokan, Jurica Miletić, Ante Sosa
Prirodoslovno Matematički fakultet
Zagreb, Hrvatska

***Sažetak*—Cilj ovog rada je opisati kako detektirati pakirane datoteke uz pomoć alata baziranih na strojnom učenju. Zadatak je osmislila firma ReversingLabs, koja se sama bavi detekcijom malicioznih datoteka. Također, navedena firma nam je dala podatke za treniranje i testiranje našeg modela.**

I. Uvod

Pakiranje je metoda izmjene izvršnih datoteka bez mijenjanja njihove izvorne funkcionalnosti, ali na način da se datoteka zaštiti od reverznog inženjeringa, da se smanji veličina originalne izvršne datoteke, ili da se obfuscira maliciozni izvršni kod. Pakiranje podrazumijeva izmjenu sadržaja datoteke te dodavanje instrukcija koje će prilikom izvršavanja taj sadržaj obnoviti.

Packeri modificiraju originalnu izvršnu datoteku na razne načine:

- Kompresijom podataka
- Enkripcijom podataka
- Obfuskacijom
- Dodavanjem detekcije izvršavanja unutar debuggera ili virtualnog računala
- Modificiranjem raznih dijelova formata izvršne datoteke

U području računalne sigurnosti posebno su učestali packeri za Windows Portable Executable tj. PE datoteke. PE datoteke su povijesno najčešći nositelji malicioznog koda u obliku virusa, ransomwarea, trojanskih konja, itd., te se packeri koriste da bi se taj maliciozni kod prikrio. Klasična statička analiza (bez pokretanja datoteka) koju provode antivirusi bazira se na potpisima. Oni nastaju tako da se prikupe primjeri nekog malwarea te se pronađe niz byteova specifičan za taj malware, koji se zatim traži prilikom skeniranja datoteka antivirusom.

Primjenom packera mijenja se sadržaj datoteke, zbog čega potpisi mogu prestati biti prisutni. Na taj se način iz jedne maliciozne datoteke može napraviti više različitih inačica. Packeri koji imaju nemalicioznu primjenu vrlo su rijetki u odnosu na packere koji se koriste za malware. Štoviše, antivirusi često rade potpise za same packere koji se koriste isključivo za malware. Za packere generalne namjene razvijaju se unpackeri kojima se obnavlja originalni sadržaj datoteke za analizu, a detekcija pakiranja nepoznatim formatom snažna je naznaka malwarea.

Ovaj pristup pouzdan je za detekciju pojedinih packera, ali vremenski je zahtjevan čak i za specijalizirane reverzne inžinjere. Osim toga, takav pristup zahtijeva i već izdvojene primjere pakiranih datoteka.

Identify applicable funding agency here. If none, delete this.

Metoda automatske detekcije pakiranih datoteka omogućuje:

- Izdvajanje zanimljivih malware datoteka za detaljniju analizu
- Ranu detekciju nikad prije viđenog malwarea
- Prepoznavanje malware kampanja

II. Podaci

Skup podataka bazirat će se na skupu raznovrsnih packera i dodatnim nepakiranim datotekama. Svaki primjer se sastoji od dva dijela: originalne datoteke i TitaniumCore izvještaja za tu datoteku. Cilj zadatka je odvojiti samo pakirane datoteke od nepakiranih i raspakiranih, ali će sudionici za razvoj modela dobiti detaljnije informacije, poput generalnih vrsta packera. U podacima mogu biti varijante višestruko pakiranih datoteka, poput dvostruko pakirane datoteke koja se tijekom TitaniumCore procesiranja zatim jednom raspakira (i dalje se označava kao pakirana).

III. Problem

Problem se sastojao od odabira značajki i izgradnje modela stojnog učenja koji će najbolje predvđati jeli dana datoteka pakirana.

IV. Pristup problemu

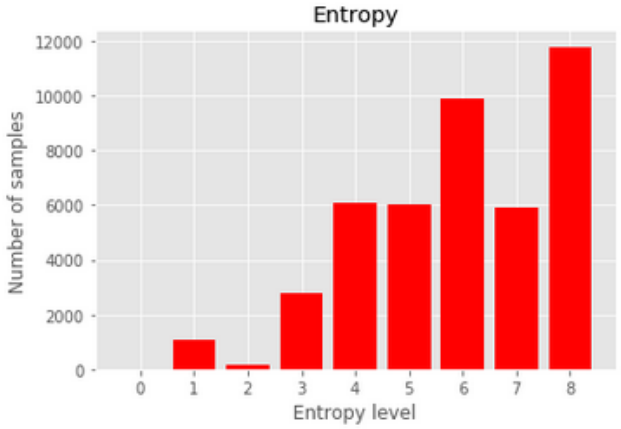
Naš pristup problemu bazirao se na pronalaženju glavnih značajki iz TitaniumCore izvještaja. Značajke koje smo izvukli iz TitaniumCore izvještaja:

- 10 najčešćih imena odjeljaka (section names) i broj pojavljivanja ostalih odjeljaka
- 10 najčešćih imena unešenih datoteka (import names)
- 25 najčešćih programskih sučelja za aplikacije koje koriste izvršne datoteke
- 25 najčešćih upozorenja koje se javljaju u izvršnim datotekama
- 16 najčešćih resursa u izvršnim datotekama
- veličina odjeljaka
- entropije odjeljaka (ukupna, maksimalna i aritmetička sredina entropija svih odjeljaka) [1]
- datum nastanka datoteke
- imena odjeljaka koji se pojavljuju više od 95% puta u pakiranim datotekama
- imena odjeljaka koji se pojavljuju više od 80% puta u pakiranim datotekama
- veličina zaglavlja (optional headers)

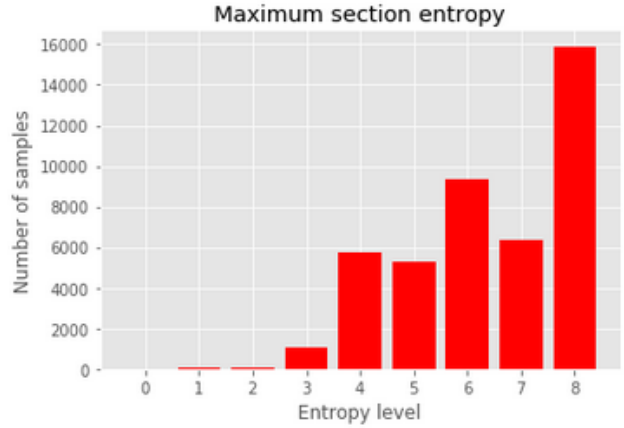
Prvotno smo promatrali samo imena odjeljaka iz čega smo mogli izvući podosta novih značajki. Sama imena odjeljaka

mogu biti promjenjena i samim time, ime odjeljka može biti poizvoljno, pa nije baš prepuzdana značajka. Nadalje, promatrali smo koje sve api-je datoteka poziva, iz čega smo pronašli dosta "sumljivih" api-ja, te imena datoteka koje su nužne za pokretanje datoteke. Promatrali smo upozorenja koja se javljaju prilikom izvođenja datoteke, što se nije pokazalo kao najbolja značajka. Također smo veličinu odjeljaka i veličinu cijele datoteke, te smo zaključili, ako je suma veličina svih odjeljaka veća od veličine cijele datoteke, tada je binarni kod datoteke sigurno mijenjaan. Značajke vezane za razinu entropije su pokazale kao najpuzdanije. Entropija cijele datoteke, sama kao takva, je previše gruba, ali se u kombinaciji sa entorpijama odjeljaka pokazala kao odlična značajka. Također smo uzeli u obzir minimalnu, maksimalnu i prosječnu vrijednost entropije svih odjeljaka, time smo pokrili slučajeve kada se samo jedan odjeljak promjeni pa je maksimalna entropija odjeljaka povećana.

Graf prikazuje broj datoteka u pojedinim skupinama entropije na cijelom skupu koji je primjenjen za treniranje.



Idući graf prikazuje broj datoteka u pojedinim skupinama maksimalne entropije odjeljaka na cijelom skupu koji je primjenjen za treniranje.



Nakon toga smo odlučili analizirati dane značajke izgradnjom modela Random Forest Classifiera i XGBoost treniranjem te bi usporedbom važnosti značajki u jednom od modela zadržali one koje su najviše utjecali na rezultat treniranja danog modela.

V. Realizacija rješenja

Prvo smo odlučili istrenirati model XGBoost Classifiera iz biblioteke xgboost u pythonu. Da bismo odredili hiperparametre modela koristili smo Randomized Search CV iz biblioteke sklearn (modul model_selection) kako bismo pretražili prostor vrijednosti za parametre

- max_depth ($[1, 2, ..., 100]$)
- min_child_weight ($[0.1, 0.2, ..., 3.0]$)
- learning_rate ($[0.1, 0.2, ..., 3.0]$)
- base_score. ($[0.5, 0.6, 0.7, 0.8, 0.9]$)

Nakon toga odabiremo parametre koji daju najbolji rezultat (po srednjoj vrijednosti deseterostruke cross validacije koju provede Randomized Search CV) na nasem skupu podataka za pretraživanje. Također, uz zadane parametre za model xgboosta smo odlučili koristiti 100 stabala (po defaultnim postavkama u XGBClassifieru). Nakon toga smo odlučili istrenirati model Random Forest Classifier, također određujući hiperparametre modela koristeći Randomized Search CV. Željeli smo pretražiti prostor vrijednosti za parametre

- max_depth ($[1, 2, ..., 100]$)
- criterion ($[True, False]$)
- oob_score ($[True, False]$)
- max_features ($[0.001, 0.002, ..., 1.000]$)

Nakon toga odabiremo parametre na isti način kao i kod XGBClassifiera. Također koristit ćemo 100 stabala (po defaultnim postavkama). S obzirom na prikazano, XGBoost model nakon treniranja na podacima daje bolje rezultate, te ćemo njega koristiti za daljnju analizu značajki.

Nakon toga odabiremo parametre koji daju najbolji rezultat (po srednjoj vrijednosti deseterostruke cross validacije koju provede Randomized Search CV) na nasem skupu podataka za pretraživanje. Također, uz zadane parametre za model xgboosta smo odlučili koristiti 100 stabala (po defaultnim postavkama u XGBClassifieru). Nakon toga smo odlučili istrenirati model Random Forest Classifier, također određujući hiperparametre modela koristeći Randomized Search CV. Željeli smo pretražiti prostor vrijednosti za parametre

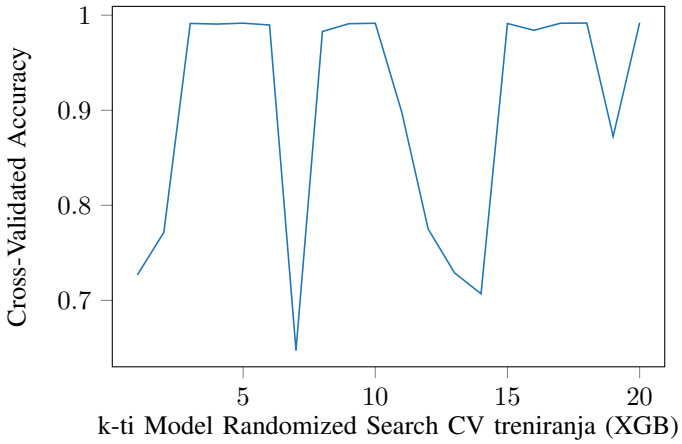
- max_depth ($[1, 2, ..., 100]$)
- criterion ($[True, False]$)
- oob_score ($[True, False]$)
- max_features ($[0.001, 0.002, ..., 1.000]$)

Nakon toga odabiremo parametre na isti način kao i kod XGBClassifiera. Također koristit ćemo 100 stabala (po defaultnim postavkama). S obzirom na prikazano, XGBoost model nakon treniranja na podacima daje bolje rezultate, te ćemo njega koristiti za daljnju analizu značajki.

A. Analiza značajki

Nakon što smo istrenirali model na trening podacima, ispisali smo sve značajke, i koju su ulogu one imale pri treniranju XGB modela. 10 najvažnijih značajki u modelu su

- entropija



Slika 1. Rezultati pretrazivanja parametara sa Random Search CV-om za XGB model

!! Missing graphics !!

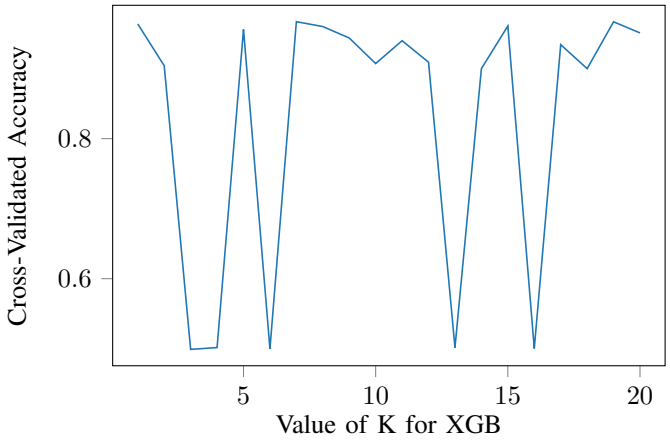
Slika 2. Rezultati treniranja i validacije modela XGB

  - entropy (15.42289%)
  - mean entropy (13.67357%)
  - max entropy (11.02552%)
  - min entropy (8.93918%)
- import api values (8.21698%)
- ime odjeljaka
  - other sections (3.12951%)
  - .reloc (1.36416%)
  - .rdata (1.63698%)
- import name values (2.39127%)
- ime importova
  - other imports (1.974%)

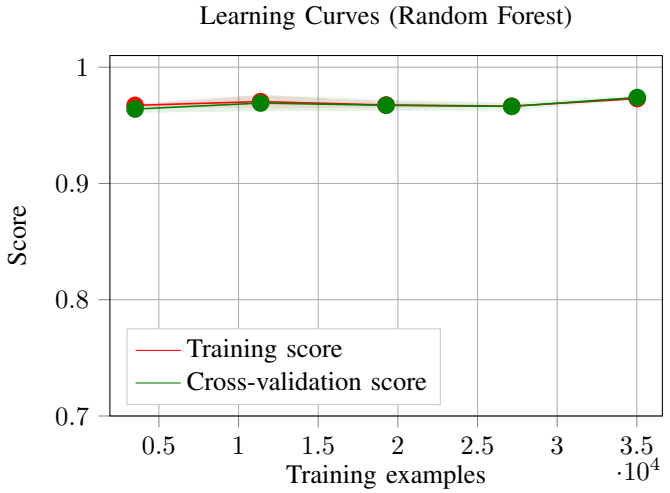
Od sveukupno 113 značajki uzeli smo 30 najznačajnijih, koji imaju utjecaj veći od 0.5% na treniranje modela, te ćemo njih iskoristiti za treniranje krajnjeg XGBClassifier modela. Opet ćemo provrtjeti Randomized Search CV kako bismo našli najbolje parametre za ovaj podskup značajki.

VI. Zaključak

Nas krajnji model ima preciznost od 99.08%. Zakljucili smo da entropija izvrsne datoteke od raznih znacajki koje smo uzeli na promatranje ima najvecu vaznost, te da je pokazala najvecu povezanost sa tim je li izvrsna datoteka pakirana.



Slika 3. Rezultati pretrazivanja parametara sa Random Search CV-om za Random Forest model

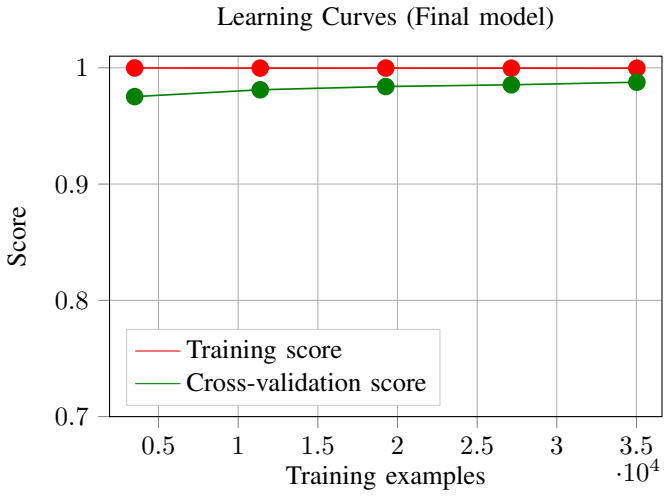


Slika 4. Rezultati treniranja i validacije modela Random Forest

VII. Majka

Literatura

[1] Xabier Ugarte-Pedrero, Igor Santos, Borja Sanz, Carlos Laorden and Pablo Garcia Bringas, “Countering Entropy Measure Attacks on Packed Software Detection,” University of Deusto, Bilbao, Spain.



Slika 5. Rezultati cross validacije krajnjeg modela