ДОДАТОК А

Графічний матеріал атестаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра ЕОМ

МЕТОД ПОБУДОВИ СИГНАТУР ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ ГЛИБОКОГО НАВЧАННЯ

АТЕСТАЦІЙНА РОБОТА
ДРУГИЙ (МАГІСТЕРСЬКИЙ) РІВЕНЬ

Автор:
Букін І.О.

Керівник:
к.т.н. Мартовицький В.О.

Мета і задачі роботи

- **Метою роботи є підвищення якості виявлення шкідливого програмного забезпечення шляхом розробки алгоритму вилучення сигнатур з використанням алгоритмів глибокого навчання.**

- **Об'єктом роботи** є програмне забезпечення, що виконується з елементами шкідливого коду.

- **Предмет дослідження** є розробка алгоритму виділення ознак шкідливого програмного забезпечення і подальшого створення компактного представлення цієї поведінки.

Типовий ряд етапів вирішення завдань методами Data Mining

№ п/п	Етап
1	Формування гіпотези;
2	Збір даних;
3	Підготовка даних (фільтрація);
4	Вибір моделі;
5	Підбір параметрів моделі і алгоритму навчання;
6	Навчання моделі (автоматичний пошук інших параметрів моделі);
7	Аналіз якості навчання, якщо незадовільний перехід на п. 5 або п. 4;
8	Аналіз виявлених закономірностей, якщо незадовільний перехід на п. 1, 4 або 5.

Методи пошуку шкідливого коду

Статичні методи пошуку шкідливого коду

- хеші,
- байтова характеристика,
- евристична характеристика

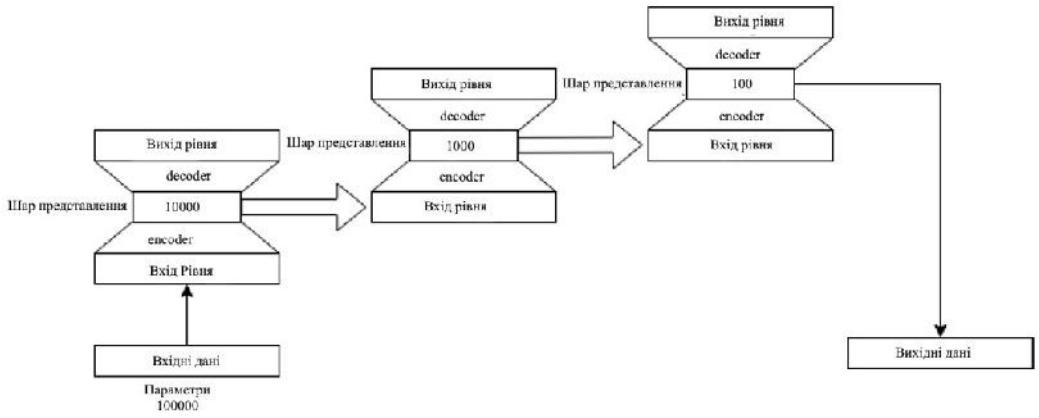
Динамічні методи пошуку шкідливого коду

Оцінки ефективності різних методів відбору ознак

Метод	FP	FP	Точність (%)
Signature + HexDump	0.34	0.00	49.32
SVN + DLLs	0.58	0.09	83.61
SVN + DLL, функції, що використовуються	0.71	0.08	89.36
SVN + DLL, підрахунок функцій	0.53	0.05	89.07
Naive Bayes + strings	0.97	0.04	97.11
Voting Naive Bayes + HexDump	0.98	0.06	96.88

5

Структурна схема моделі



6

Характеристики моделі

- кількість рівнів репрезентації $L_0, L_1, ..., L_c$;
- необхідний розмір останнього шару репрезентації dim $(L_c^{n//2+1})$, де $dim(L)$ - функція визначення розмірності шару
- кількість внутрішніх шарів кожного рівня моделі, що складаються з шарів кодування L_i^e, репрезентації L_i^r і декодування L_i^d, які організовані в наступному порядку: $L_i^e{}_1, ..., L_i^e{}_m, L_i^r, L_i^d{}_1, ..., L_i^d{}_m$ і загальна кількість яких дорівнюватиме $L_i n\_layers = 2m + 1$
- функція зменшення розмірності на кожному шарі моделі $fR(L_i)$, яка повертає необхідний від рівня розмір шару репрезентації
- внутрішні параметри кожного рівня моделі:
  - кількість нейронів в кожному шарі кожного рівня;
  - параметри нормалізації даних;
  - параметри додавання шуму до навчальних даних шару;
  - параметри функцій активації шарів;
  - параметри функції помилки;
  - кількість епох навчання рівня.

7

Типи даних, які підлягають захисту

$\dim(L_0) = len(X)$ (3.1) де $len(X)$ - розмірність вхідних даних,

$\dim(L_i) = fR(L_{i-1})$. (3.2)

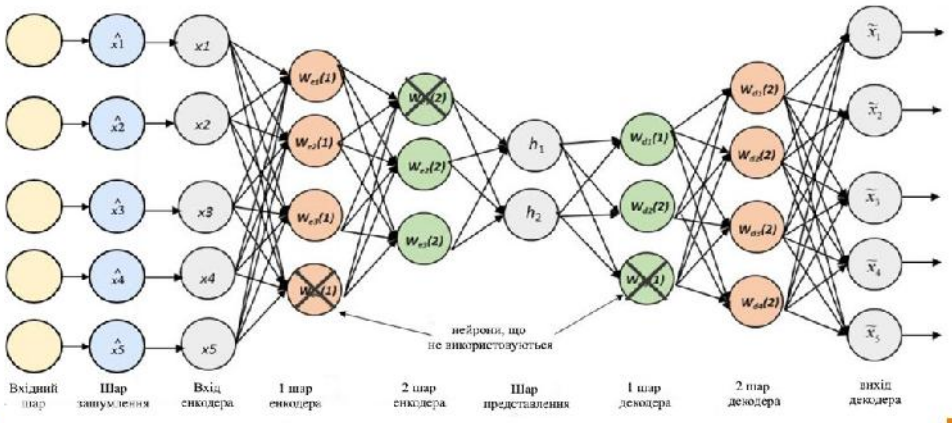
В рамках роботи були реалізовані наступні функції зменшення розмірності:

$fR(I_i) - \frac{\dim(I_i^0)}{c}$, (3.3) де c - деяка константа, функція dim(L) - функція визначення розмірності шару, а - вхідний шар рівня;

$$\begin{cases} c*1000, & \dim(I_i^0) > c*1000 \\ \dim(L^0) - c*100, & \dim(L_i^0) > c*100 \\ \dim(L_i^0) - c*10, & \dim(L^0) > c*10 \end{cases}$$ (3.4) де c - деяка константа, функція dim(L) - функція визначення розмірності шару, а - вхідний шар рівня;

$fR(I_i) = \dim(I_i^0) - c$, (3.5) де c - деяка константа, функція dim(L) - функція визначення розмірності шару, а - вхідний шар рівня.

Структурна схема шарів глибокого автоенкодера



9

Властивості бозової одениці моделі

1) базова одиниця моделі здатна зменшувати розмірність вхідних даних до подання характеристик поведінки у вигляді структури малої розмірності;

2) Декодувальна частина базової одиниці здатна відновлювати інформацію про аспект поведінки примірника ШПЗ, маючи в наявності тільки коротке представлення про нього;

3) базова одиниця моделі навчається комплексним ознаками, створюючи їх ієрархії на кожному рівні моделі. Цей факт важливий у зв'язку з тим, що вибірка даних для навчання детектора характеризується значним розкидом даних, і при цьому не мають між собою загальних ознак на рівні вхідних даних. Здатність детектора инкапсулировать ознаки таких відмінних даних досягається за допомогою використання нелінійних функцій активації штучних нейронів в шарах моделей;

4) ознаки, які виявляються базовою одиницею моделі, мають властивості стійкості до неінформативним змін. Це відбувається завдяки використанню пошкоджених вхідних даних і призводить до підвищення узагальнюючої здатності за рахунок видалення «викидів» і поліпшенню інтерпретується ознак.

Алгоритм добування сигнатури

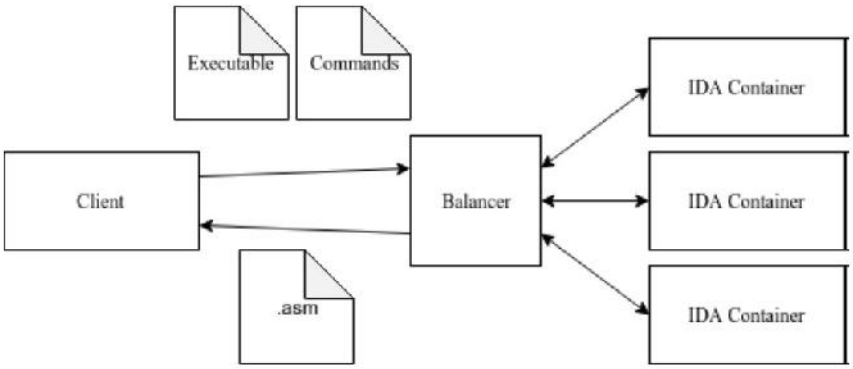
```
1. for i=1 to n do
2. ProfileExtractori = Profilersi,1
3. VectorReducer i = Profilersi,2
4. BehaviourVectori= ProfileExtractori(F)
5. Profilei=VectorReduceri(BehaviourVectori)
6. end
7. Sign={{1, Profile1}{2, Profile2}…{n, Profilen}}
8. return Sign
```

11

Алгоритмічна база розробленої моделі

- алгоритм перетворення виконуваного файлу в дамп дизассемблера;
- алгоритми перетворення дампі дизасемблера в характеристичний вектор поведінки. При цьому, розроблені алгоритми засновані на статичному аналізі, і враховують як статистичні метрики (кількість входжень машинних слів), так і метрики поведінки, виражені через граф управління;
- алгоритм навчання базової одиниці моделі, що представляє собою модифікацію алгоритму зворотного поширення помилки в умовах наявності множини залежних одна від одної частин моделі;
- алгоритм оцінки ефективності роботи, заснований на проведенні порівняльних тестів точності класифікації на отриманих від моделі даних.

Логічна схема системи вилучення ассемблерних дампів

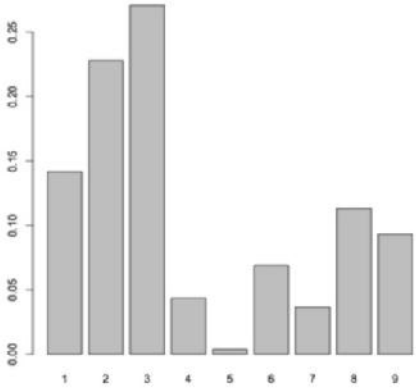


Фрагменти векторів, що генеруються

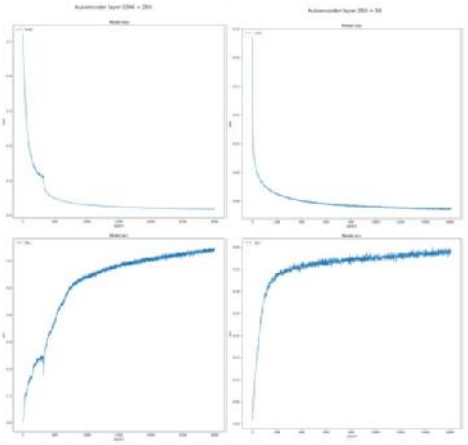
name	01JKJHo[Mh5gxyDYTl4C2	01SuzwMJEI7A8dQbl	01azqd4InC7m9JpocCv5
edx	750	1121	1493
esi	496	24	1900
bl	25	4	47
imul	30	4	82
jb	112	4	510
lstrcpynA	3	8	7
EndPaint	3	1	14
SetErrorMode	3	3	6

Вміст даних від Microsoft Malware Classification Challenge

	Category	Count
1	Ramnit	1541
2	Lollipop	2478
3	Kelihos_ver3	2942
4	Vundo	475
5	Simda	42
6	Tracur	751
7	Kelihos_ver1	398
8	Obfuscator.ACY	1228
9	Gatak	1013
	Total	10868



–Криві навчання рівнів моделі



Оцінка результатів класифікації по набору статистичних даних після зменшення розмірності різними методами

Метод	Transforming time, сек	Learning time, сек	Accuracy	Precision	Recall	f1-score
Розроблений	00.290277	02.343170	0.939±0.02	0.867±0.05	0.839±0.04	0.844±0.04
PCA	00.338247	02.375980	0.689±0.28	0.667±0.25	0.600±0.27	0.592±0.29
ICA	00.588323	02.236536	0.781±0.15	0.737±0.15	0.623±0.23	0.641±0.22

Висновки

В рамках роботи була представлена система попередньої обробки виконуваних файлів з метою конвертації виконуваних файлів в асемблерний дамп;

Для вилучення ознак, з якими покликана працювати модель, було розроблено два алгоритму виділення ознак описів аспектів поведінки ШПЗ на основі асемблерних дампів шкідливих програм, що представляють як статистичні дані про файлі методом підрахунку машинних команд, так і логічну систему роботи шкідливого ПЗ, через побудову графа управління.

Розроблено алгоритм навчання структурного елементу моделі. Використання алгоритмом проріджування вибірки робочих нейронів дозволяє детекторам ознак навчатися за незалежними ознаками, і задіяти при навчанні тільки обмежену частину нейронів моделі, формуючи нові зв'язки для кращого вилучення інформації з даних, а використання зашумлення вибірки дозволяє позбавити мережу від перенавчання та навчити формувати більш стійкі функції апроксимації для даних, завдяки необхідності краще відновлювати дані.

У підсумку, розроблені алгоритми реалізовані у вигляді програмного комплексу, який надає API для використання в антивірусних і аналітичних системах. Реалізований програмний комплекс включають в себе необхідний набір інструментів для створення, навчання, зберігання моделей і включає інструменти, що дозволяють ефективно аналізувати результати роботи моделей.

ДОДАТОК Б

ПРИКЛАДИ ОПИСУ МОДЕЛІ В РАМКАХ РЕАЛІЗОВАНОГО ПРОГРАМНОГО КОМПЛЕКСУ

```
class AsmDeepTransformer(DeepTransformerBase):
reduction_function = divider_reduction_function(3)
scale = (0, 1)
input_layer_properties = LayerProperties(
(
{'name': 'encoder_input_layer', },{}
)
)
encoder_layer_properties = LayerProperties(
(
{
'activation': 'relu',
'bias_constraint': constraints.MinMaxNorm(0.0, 1.0),
'name': 'encoder_layer-1'
}, {
'dropout': 0.1,
'noise': 0.1,
'normalization': {'momentum': 0.99, 'epsilon': 0.001},
}
),(
{
'activation': 'sigmoid',
'name': 'representation_layer',
},{
'normalization': {'momentum': 0.99, 'epsilon': 0.001},
}
)
)
decoder_layer_properties = LayerProperties(
(
{
'activation': 'sigmoid',
'bias_constraint': constraints.MinMaxNorm(0.0, 1.0),
'name': 'decoder_layer-1'
},{
'dropout': 0.1,
'noise': 0.1,
'normalization': {'momentum': 0.99, 'epsilon': 0.001},
}
)
)
output_layer_properties = LayerProperties(
(
{
'activation': 'sigmoid',
'name': 'decoder_ouput'
}, {
'normalization': {'momentum': 0.99, 'epsilon': 0.001},
}
)
)
fit_options = dict(epochs=1500, batch_size=128, shuffle=True)
compiler_options = dict(
optimizer=optimizers.Adadelta(),
```

```
      loss=lambda true, pred: K.sqrt(K.mean(K.square(pred - true))), # RMSE
      metrics=['accuracy'])
      reduction_function: '<function>_divider_reduction_function(x):'
      scale: (0,1)
      compiler_options:
      loss: '<function>loss=lambda true, pred: K.sqrt(K.mean(K.square(pred -
true))), # RMSE'
      metrics: [accuracy]
      optimizer:
      kwargs: {decay: 0.0, epsilon: 1.0e-07, lr: 1.0, rho: 0.95}
      name: keras.optimizers.Adadelta
      fit_options: {batch_size: 128, epochs: 1500, shuffle: true}
      layers:
      input_layer_properties:
      encoder_input_layer:
      layer_modifiers: {}
      layer_properties: {}
      encoder_layer_properties:
      encoder_layer-1:
      layer_modifiers:
      dropout: 0.1
      noise: 0.1
      normalization:
      epsilon: 0.001
      momentum: 0.99
      layer_properties:
      activation: relu
      bias_constraint:
      kwargs: {axis: 0, max_value: 1.0, min_value: 0.0, rate: 1.0}
      name: keras.constraints.MinMaxNorm
      representation_layer:
      layer_modifiers:
      normalization:
      epsilon: 0.001
      momentum: 0.99
      layer_properties:
      activation: sigmoid
      decoder_layer_properties:
      decoder_layer-1:
      layer_modifiers:
      dropout: 0.1
      noise: 0.1
      normalization:
      epsilon: 0.001
      momentum: 0.99
      layer_properties:
      activation: sigmoid
      bias_constraint:
      kwargs: {axis: 0, max_value: 1.0, min_value: 0.0, rate: 1.0}
      name: keras.constraints.MinMaxNorm
      output_layer_properties:
      decoder_ouput:
      layer_modifiers:
      normalization:
      epsilon: 0.001
      momentum: 0.99
      layer_properties: {activation: sigmoid}
```