JOONGBU UNIVERSITY
JBU
충부대학교

인공지능을 활용한 악성코드 탐지

TEAM NKLCB

CONTENTS INDEX

CHAPTER.1

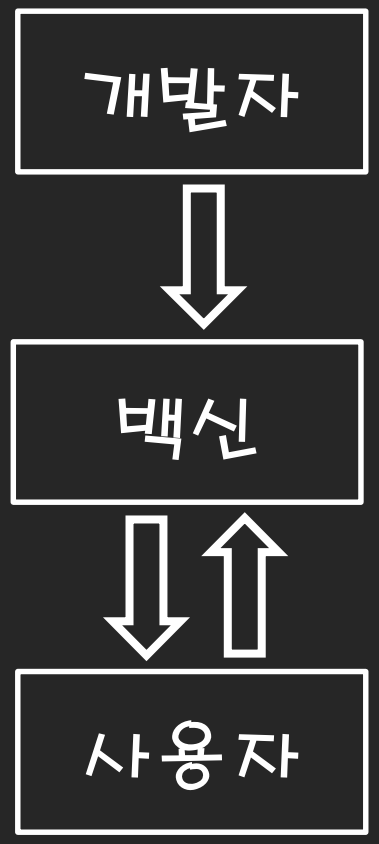
시스템 구성도

개 발 환 경

- 개발 도구

- OS

- 시스템 구성도



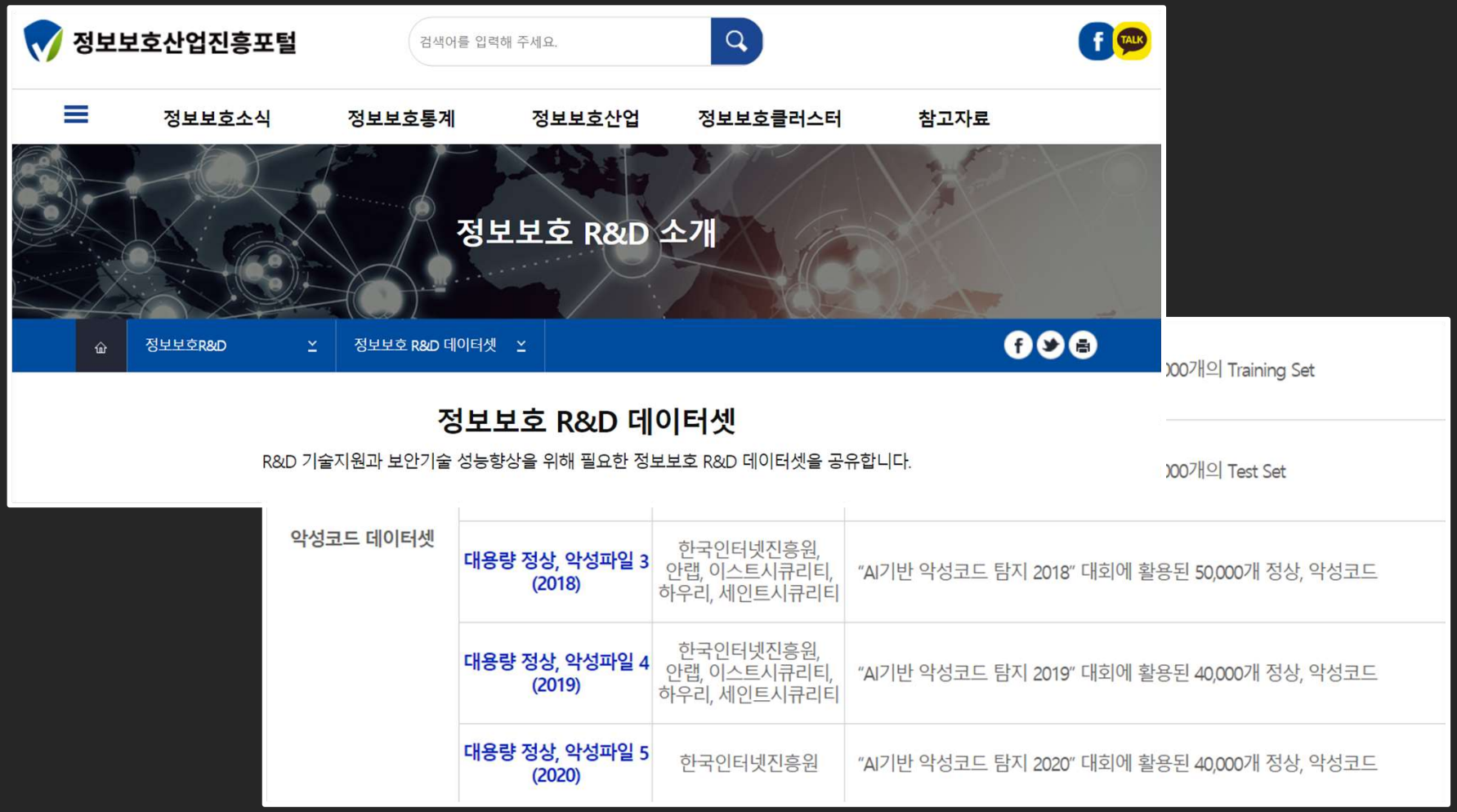
- 인공지능 모델 개발 후 탑재
- 백신 검사
- 결과 반환

CHAPTER.2

진 행 내 용

진 행 내 용 (1 /)

- 데이터셋


정보보호산업진흥포털
검색어를 입력해 주세요.
정보보호소식
정보보호통계
정보보호산업
정보보호클러스터
참고자료
정보보호 R&D 소개
정보보호R&D
정보보호 R&D 데이터셋
정보보호 R&D 데이터셋
R&D 기술지원과 보안기술 성능향상을 위해 필요한 정보보호 R&D 데이터셋을 공유합니다.
)00개의 Training Set
)00개의 Test Set
악성코드 데이터셋
대용량 정상, 악성파일 3 (2018)
한국인터넷진흥원, 안랩, 이스트시큐리티, 하우리, 세인트시큐리티
"AI기반 악성코드 탐지 2018" 대회에 활용된 50,000개 정상, 악성코드
대용량 정상, 악성파일 4 (2019)
한국인터넷진흥원, 안랩, 이스트시큐리티, 하우리, 세인트시큐리티
"AI기반 악성코드 탐지 2019" 대회에 활용된 40,000개 정상, 악성코드
대용량 정상, 악성파일 5 (2020)
한국인터넷진흥원
"AI기반 악성코드 탐지 2020" 대회에 활용된 40,000개 정상, 악성코드

진 행 내 용 (3 /)

- 특징추출 / 분석

- PE 헤더
- PE 이미지(바이너리 이미지)
- N-gram(4-gram)

진 행 내 용 (2 /)

▪ 특징추출 / 분석

```
def extract_dos_header(self,pe):
    IMAGE_DOS_HEADER_data = [ 0 for i in range(6)]
    try:
        IMAGE_DOS_HEADER_data = [
                                pe.DOS_HEADER.e_cblp,₩
                                pe.DOS_HEADER.e_cp, ₩
                                pe.DOS_HEADER.e_cparhdr,₩
                                pe.DOS_HEADER.e_maxalloc,₩
                                pe.DOS_HEADER.e_sp,₩
                                pe.DOS_HEADER.e_lfanew]
```

```
def extract_optional_header(self,pe):
    OPTIONAL_HEADER_data = [ 0 for i in range(21)]
    DLL_char =[]
    OPTIONAL_HEADER_data2 = [ 0 for i in range(6)]

    try:
        OPTIONAL_HEADER_data = [
            pe.OPTIONAL_HEADER.MajorLinkerVersion,₩
            pe.OPTIONAL_HEADER.MinorLinkerVersion,₩
            pe.OPTIONAL_HEADER.SizeOfCode,₩
            pe.OPTIONAL_HEADER.SizeOfInitializedData,₩
            pe.OPTIONAL_HEADER.SizeOfUninitializedData,₩
            pe.OPTIONAL_HEADER.AddressOfEntryPoint,₩
            pe.OPTIONAL_HEADER.BaseOfCode,₩
```

filename	MD5	e_cblp	e_cp	e_cparhdr	e_maxalloc	e_sp	e_lfanew	NumberOf	CreationYe	FH_char0	FH_char1	FH_char2	FH_char3	FH_char4	FH_char5	FH_char6
d7f8d89e1	d7f8d89e1	144	3	4	65535	184	128	3	1	0	1	0	0	0	0	0
d952e206l	d952e206l	0	1	2	17744	332	12	3	1	0	1	1	1	0	0	0
bce82de8	bce82de8	144	3	4	65535	184	200	4	1	1	1	0	0	0	0	0
13C9E872	13c9e872f	144	3	4	65535	184	264	5	1	0	1	0	0	0	0	0
BE0E0BBA	be0e0bba	144	3	4	65535	184	216	5	0	0	1	0	0	0	0	0
9ebaf9f71	9ebaf9f71	80	2	4	65535	184	256	8	1	1	1	1	1	0	0	1
ebd2813e	ebd2813e	80	2	4	65535	184	256	8	1	0	1	1	1	0	0	1
1d6d1570	1d6d1570	80	2	4	65535	184	256	3	1	0	1	1	1	0	0	1
D7CD00A	d7cd00a1f	80	2	4	65535	184	256	9	0	0	1	1	1	0	0	1
d39fdfe34	d39fdfe34	144	3	4	65535	184	128	5	1	1	1	1	0	0	0	0
5502856b	5502856b	144	3	4	65535	184	128	3	0	0	1	0	0	0	0	0
fe5ab4c19	fe5ab4c19	144	3	4	65535	184	128	4	1	0	1	0	0	0	0	0
89f3a3fb9	89f3a3fb9	144	3	4	65535	184	232	5	1	1	1	1	1	0	0	0
e37d3914	e37d3914	80	2	4	65535	184	256	3	1	1	1	1	1	0	0	1
8163C7A4	8163c7a4	80	2	4	65535	184	256	8	1	0	1	1	1	0	0	1
6eb468e4	6eb468e4	80	2	4	65535	184	512	8	0	0	1	1	1	0	0	0
f2cd4cd14	f2cd4cd14	144	3	4	65535	184	216	4	0	0	1	0	0	0	0	0
41f24c96d	41f24c96d	144	3	4	65535	184	264	5	0	0	1	0	0	0	0	0

▪ 특징추출 / 분석

-> Check_packer : 프로그램에 적용된 패커 식별

```
self.rules= yara.compile(filepath='./peid.yara')
```

```
def check_packer(self,filepath):

    result=[]
    matches = self.rules.match(filepath)

    try:
        if matches == [] or matches == {}:
            result.append([0,"NoPacker"])
        else:
            result.append([1,matches['main'][0]['rule']])
    except:
        result.append([1,matches[0]])

    return result
```

▪ 특징추출 / 분석

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder

def hot_encoding(df):

    enc = OneHotEncoder(handle_unknown='ignore', sparse=False)
    lab = LabelEncoder()

    dat = df['packer_type']
    lab.fit(dat)
    lab_dat = lab.transform(dat)

    df = df.drop('packer_type', 1)
    lab_dat = lab_dat.reshape(len(lab_dat), 1)
    enc_dat = enc.fit_transform(lab_dat)
    enc_dat = pd.DataFrame(enc_dat, columns=lab.classes_)

    df = df.reset_index(drop=True)
    enc_dat = enc_dat.reset_index(drop=True)

    df = pd.concat([df, enc_dat], axis=1)

    return df, lab.classes_
```

-> One-hot encoding

▪ 모델링

-> PE헤더에서 Random Forest 모델을 이용,
중요도 값을 조회, 상위 20가지 특징 추출

```
imp = md_pe.do_randomforest(1)

imp = dict(zip(col, imp))
sorted_imp = sorted(imp.items(), key=operator.itemgetter(1), reverse=True)
imp_20 = sorted_imp[0:20]
pe_all = pe_all_tmp
pe_all = pe_all.drop(['filename', 'MD5', 'packer_type'], 1)

Y = pe_all['class']
pe_top20 = pe_all.drop('class', 1)
X = pe_top20[imp_20.keys()]

md_pe_top20 = model.Classifiers(X, Y)
df.loc['pe_top20'] = md_pe_top20.do_all()
```

▪ 모델링

```
class Classifiers():

  def __init__(self, X, Y):

    self.x_train, self.x_test, self.y_train, self.y_test = ₩
    train_test_split(X, Y, test_size=0.2, random_state=0)


  def do_svm(self):
    clf = SVC()
    clf.fit(self.x_train, self.y_train)
    y_pred = clf.predict(self.x_test)

    return accuracy_score(self.y_test, y_pred)

  def do_randomforest(self, mode):

    clf = RandomForestClassifier()
    clf.fit(self.x_train, self.y_train)

    if mode == 1:
      return clf.feature_importances_
    y_pred = clf.predict(self.x_test)

      return accuracy_score(self.y_test, y_pred)

  def do_naivebayes(self):
    clf = clf = GaussianNB()
    clf.fit(self.x_train, self.y_train)
    y_pred = clf.predict(self.x_test)

    return accuracy_score(self.y_test, y_pred)
```

```
def do_all(self):
  rns = []

  rns.append(self.do_svm())
  rns.append(self.do_randomforest(0))
  rns.append(self.do_naivebayes())
  rns.append(self.do_dnn())

  return rns
```

-> DNN은 너무 길어 생략

- 모델링

	svm	randomforest	naivebayes	dnn	avg
pe	0.942222	0.987302	0.151111	0.061587	0.535556
pe_packer	0.942222	0.988571	0.151111	0.061587	0.535873
pe_top20	0.942222	0.986667	0.196825	0.061587	0.546825

-> PE_Packer & Random Forest

▪ 교차검증

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import model

pe_nor = pd.read_csv('nor_mod.csv')
pe_mal = pd.read_csv('mal_mod.csv')
pe_all = pd.concat([pe_nor, pe_mal])

pe_all.shape

pe_all = pe_all.drop(['filename', 'MD5', 'packer_type'], 1)

Y = pe_all['class']
X = pe_all.drop('class', 1)

md = model.Classifiers(X, Y)
acc = md.do_randomforest(0)
acc
```

-> Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

parameters = {'n_estimators':[100, 200, 500, 1000], 'max_features':['auto', None],
              'max_depth':[6, 10, 12]}
rf = RandomForestClassifier()
abc = GridSearchCV(rf, parameters, cv=10)
abc.fit(X, Y)
```

-> Grid Search

▪ 교차검증

```
from sklearn.model_selection import cross_val_score

abc = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=12, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)

abc.fit(X,Y)

score = cross_val_score(abc, X, Y, cv=10)
print score
print np.mean(score)
```

-> 10-fold 교차검증 : 최적의 매개변수 조합 찾기

```
[0.98604061 0.98984772 0.99111675 0.99111675 0.99491741 0.99491741
 0.99364676 0.98854962 0.98727735 0.99618321]
0.9913613581131774
```

- 백신탐재

```
def __scan_ml(self, filehandle, filename, fileformat, path):
        clf = self.ml_model
        ft = PE_features(filename, path)
        data, magic = ft.extract_all()

        if magic != 267 or len(data) != 69:
                return False, '', -1, kernel.NOT_FOUND

        pattern_path = path + "/patterns.csv"
        f = open(pattern_path, 'r')
        rd = csv.reader(f)
        for row in rd:
                patterns = row

        packer_type = [0] * len(patterns)

        try:
                idx = patterns.index(data[63])
        except ValueError:
                idx = 10

        packer_type[idx] = 1
        del data[63]
        data = data + packer_type
        data = np.asarray(data).reshape((1, -1))
        rns = clf.predict_proba(data)[0][1]
        print rns
        pat = "ML Confidence - " + str(rns)

        print pat

        if rns > 0.8:
                return True, pat, 0, kernel.INFECTED
        else:
                return False, '', -1, kernel.NOT_FOUND
```

-> PE 여부 / 임계치 설정

- GUI 개발

-> Tkinter

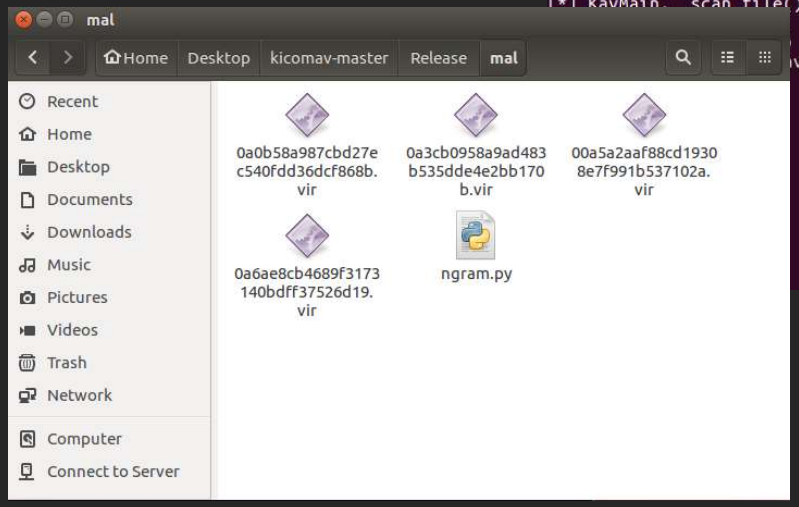
CHAPTER.3

실 행 화 면

실 행 화 면 (1 / 6)

- 실행화면

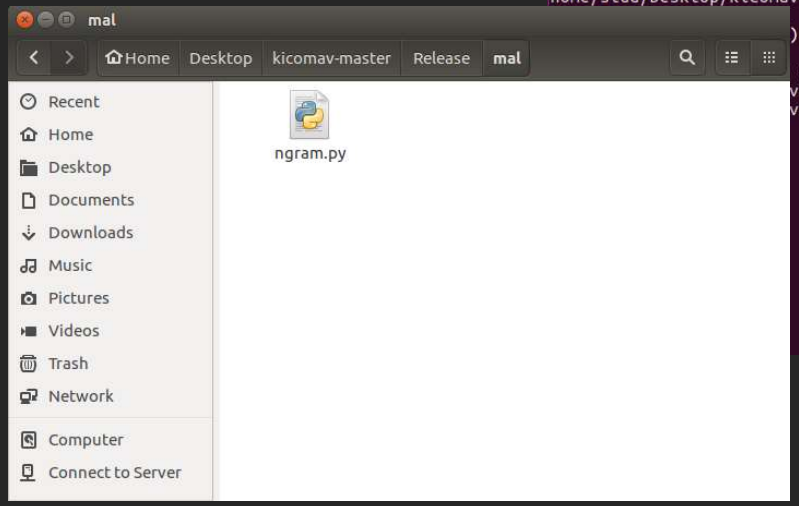
```
Last updated Fri Sep 17 05:54:12 2021 UTC
Signature number: 3,023

1   out of   4
[*] KavMain.__scan_file() :
ML Confidence - 0.778
    [-] ml.__scan_file() : ML Confidence - 0.778
home/stud/Desktop/ki ... 3140bdff37526d19.vir  infected : ML Confidence - 0.778
2   out of   4
[*] KavMain.__scan_file() :
ML Confidence - 0.802
    [-] ml.__scan_file() : ML Confidence - 0.802
home/stud/Desktop/kicomav-master/Rel ... a0b58a987cbd27ec540fdd36dcf868b.vir  infected : ML Confidence - 0.802
3   out of   4
[*] KavMain.__scan_file() :
ML Confidence - 0.754
    [-] ml.__scan_file() : ML Confidence - 0.754
home/stud/Desktop/kicomav-master/Release/mal/00a5a2aaf88cd19308e7f991b537102a.vir  infected : ML Confidence - 0.754
4   out of   4
[*] KavMain.__scan_file() :

    : ML Confidence - 0.964
v-master/Release/mal/0a3cb0958a9ad483b535dde4e2bb170b.vir  infected : ML Confidence - 0.964
```



실 행 화 면 (2 / 6)

- 실행화면

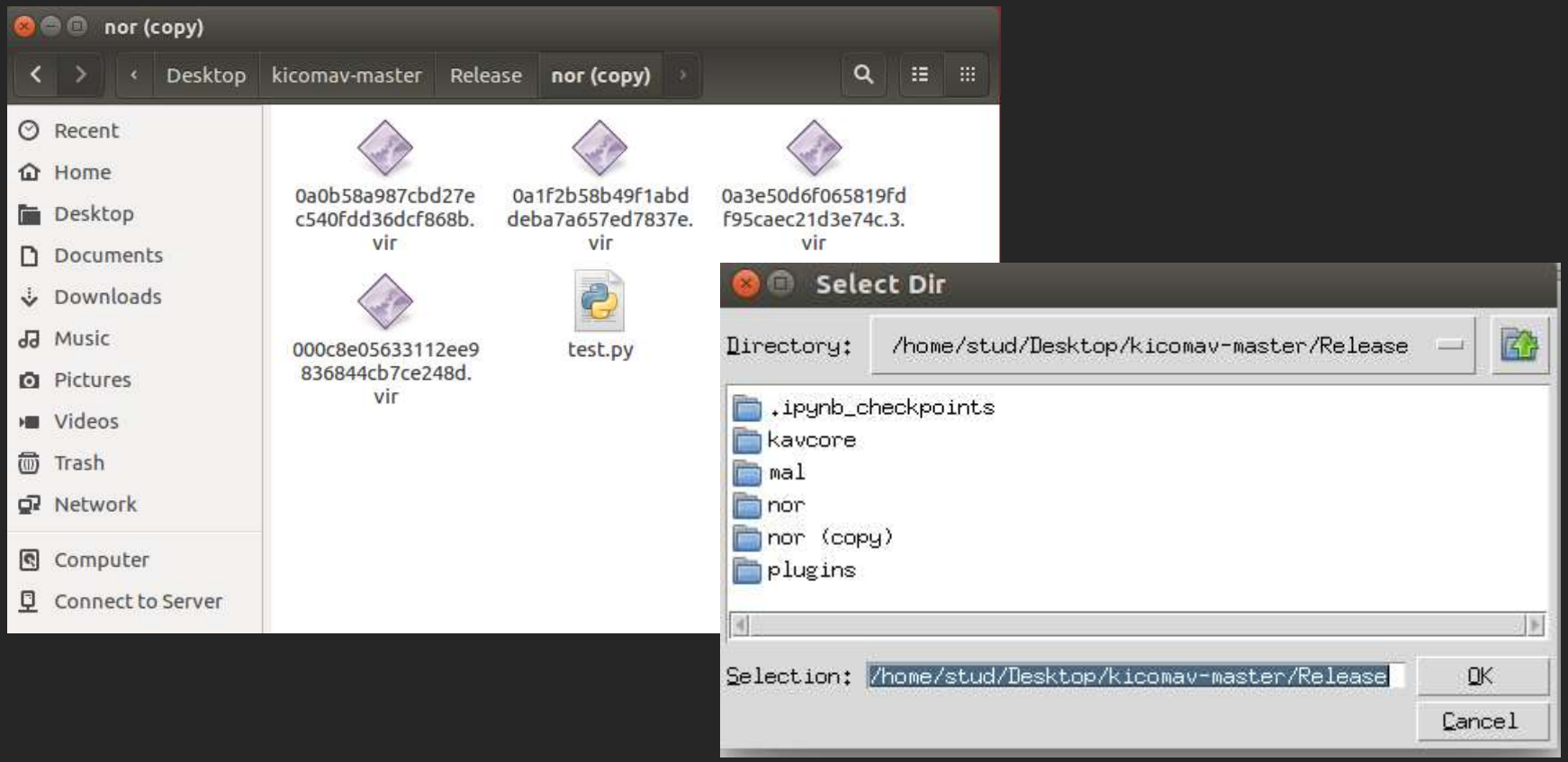
```
1   out of   5
[*] KavMain.__scan_file() :
ML Confidence - 0.778
    [-] ml.__scan_file() : ML Confidence - 0.778
home/stud/Desktop/kicomav-master/Release/mal/0a6ae8cb4689f3173140bdff37526d19.vir  infected : ML Confidence - 0.778
home/stud/Desktop/kicomav-master/Release/mal/0a6ae8cb4689f3173140bdff37526d19.vir  deleted
2   out of   5
[*] KavMain.__scan_file() :
ML Confidence - 0.802
    [-] ml.__scan_file() : ML Confidence - 0.802
home/stud/Desktop/kicomav-master/Release/mal/0a0b58a987cbd27ec540fdd36dcf868b.vir  infected : ML Confidence - 0.802
home/stud/Desktop/kicomav-master/Release/mal/0a0b58a987cbd27ec540fdd36dcf868b.vir  deleted
3   out of   5
[*] KavMain.__scan_file() :
4   out of   5
[*] KavMain.__scan_file() :
ML Confidence - 0.754
    [-] ml.__scan_file() : ML Confidence - 0.754
home/stud/Desktop/kicomav-master/Release/mal/00a5a2aaf88cd19308e7f991b537102a.vir  infected : ML Confidence - 0.754
home/stud/Desktop/kicomav-master/Release/mal/00a5a2aaf88cd19308e7f991b537102a.vir  deleted

) :

: ML Confidence - 0.964
v-master/Release/mal/0a3cb0958a9ad483b535dde4e2bb170b.vir  infected : ML Confidence - 0.964
v-master/Release/mal/0a3cb0958a9ad483b535dde4e2bb170b.vir  deleted
```



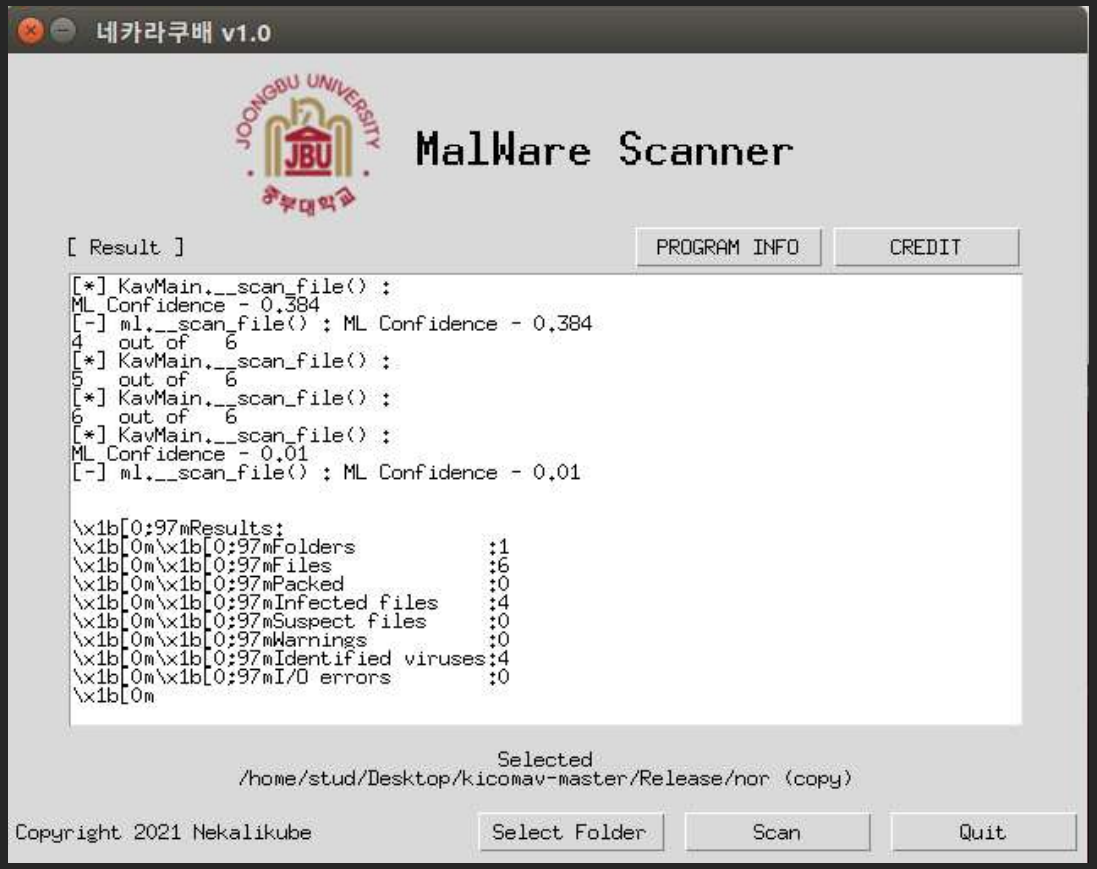
- 실행화면

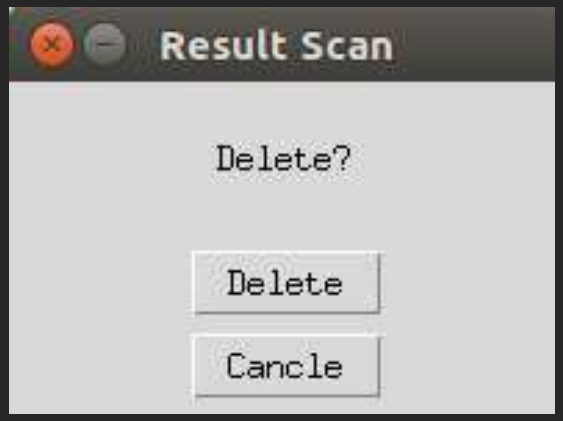

네카라쿠배 v1.0
JOONGBU UNIVERSITY
JBU
중부대학교
MalWare Scanner
[Result]
PROGRAM INFO
CREDIT
Not Search <Dir>
Copyright 2021 Nekalikube
Select Folder
Scan
Quit

- 실행화면

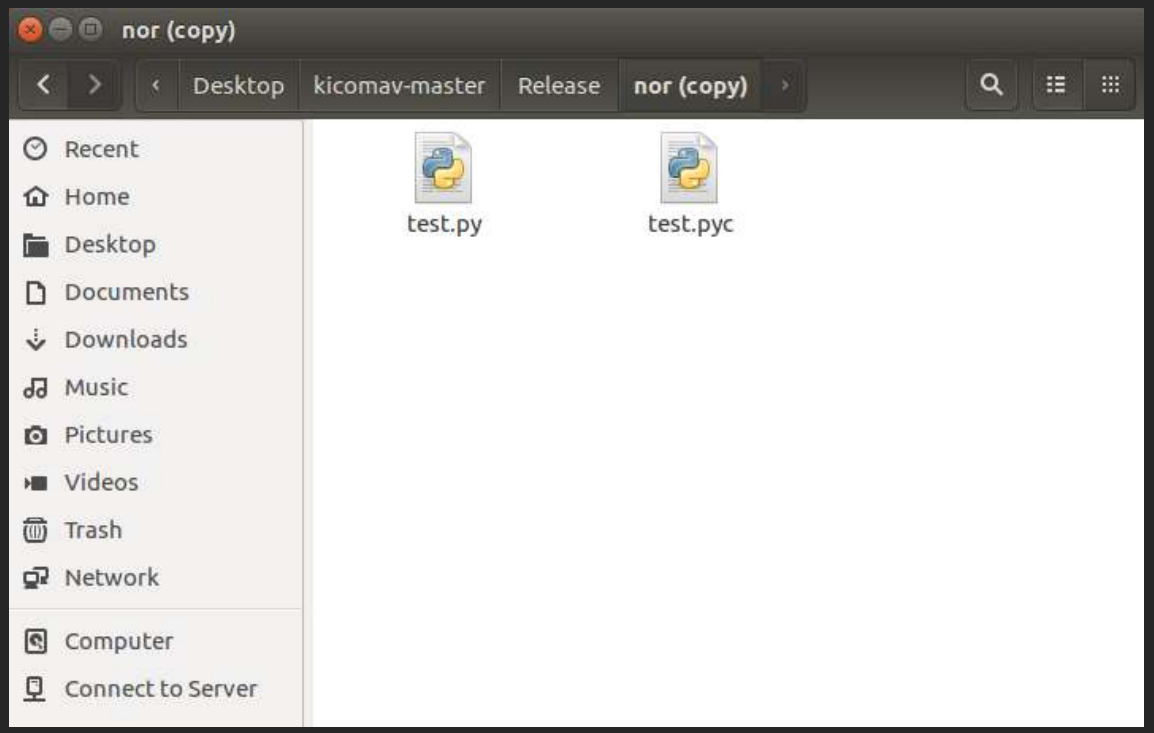

nor (copy)
Desktop
kicomav-master
Release
nor (copy)
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
Connect to Server
0a0b58a987cbd27ec540fdd36dcf868b.vir
0a1f2b58b49f1abddeba7a657ed7837e.vir
0a3e50d6f065819fdf95caec21d3e74c.3.vir
000c8e05633112ee9836844cb7ce248d.vir
test.py
Select Dir
Directory:
/home/stud/Desktop/kicomav-master/Release
.ipynb_checkpoints
kavcore
mal
nor
nor (copy)
plugins
Selection:
/home/stud/Desktop/kicomav-master/Release
OK
Cancel

▪ 실행화면


네카라쿠배 v1.0
JOONGBU UNIVERSITY
JBU
충부대학교
MalWare Scanner
[Result]
PROGRAM INFO
CREDIT
[*] KavMain.__scan_file() :
ML Confidence - 0.384
[-] ml.__scan_file() : ML Confidence - 0.384
4 out of 6
[*] KavMain.__scan_file() :
5 out of 6
[*] KavMain.__scan_file() :
6 out of 6
[*] KavMain.__scan_file() :
ML Confidence - 0.01
[-] ml.__scan_file() : ML Confidence - 0.01
\x1b[0;97mResults:
\x1b[0m\x1b[0;97mFolders :1
\x1b[0m\x1b[0;97mFiles :6
\x1b[0m\x1b[0;97mPacked :0
\x1b[0m\x1b[0;97mInfected files :4
\x1b[0m\x1b[0;97mSuspect files :0
\x1b[0m\x1b[0;97mWarnings :0
\x1b[0m\x1b[0;97mIdentified viruses:4
\x1b[0m\x1b[0;97mI/O errors :0
\x1b[0m
Selected
/home/stud/Desktop/kicomav-master/Release/nor (copy)
Copyright 2021 Nekalikube
Select Folder
Scan
Quit


Result Scan
Delete?
Delete
Cancle

- 실행화면(결과)


nor (copy)
Desktop
kicomav-master
Release
nor (copy)
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
Connect to Server
test.py
test.pyc

CHAPTER.4

결 론

결 론

- 기존 시그니처 탐지 방법과 달리 대상이 악성 코드일 확률로 악성 여부를 결정.
- 아직 알려지지 않은 악성을 잘 탐지하지만, 잡지 말아야 할 파일까지 탐지해내는 위험성 존재.
- 지속적 연구로, 오진을 줄일 수 있도록.
- 보안과 인공지능의 융합 기술 기대.

감사합니다.

2021.11.03