캡스톤 디자인I 5조 어시스트

Contents

목차

01 프로젝트 소개

02 수행 내용

03 기대 효과

asi의 필요성

최소
시간 단위

최대
주 단위

[기사 1]드라마 유령 ' 속 악성코드 , 실제로는? [3]

asi의 필요성

자동 분석 도구

연구/투자 활발

전문가 분석 보조 도구

연구/투자 부족

asi

악성 코드 분석 보조 도구

asi - 핵심 아이디어

```
10001180:    55                      push   %ebp
10001181:    8b ec                   mov    %esp,%ebp
10001183:    6a ff                   push   $0xffffffff
10001185:    68 21 80 00 10          push   $0x10008021
1000118a:    64 a1 00 00 00 00       mov    %fs:0x0,%eax
10001190:    50                      push   %eax
10001191:    83 ec 08                sub    $0x8,%esp
10001194:    a1 00 40 02 10          mov    0x10024000,%eax
10001199:    33 c5                   xor    %ebp,%eax
1000119b:    89 45 f0                mov    %eax,-0x10(%ebp)
1000119e:    53                      push   %ebx
1000119f:    56                      push   %esi
100011a0:    57                      push   %edi
100011a1:    50                      push   %eax
100011a2:    8d 45 f4                lea    -0xc(%ebp),%eax
100011a5:    64 a3 00 00 00 00       mov    %eax,%fs:0x0
100011ab:    6a 14                   push   $0x14
100011ad:    6a 0c                   push   $0xc
100011af:    ff 15 44 b2 01 10       call   *0x1001b244
100011b5:    83 c4 08                add    $0x8,%esp
100011b8:    89 45 ec                mov    %eax,-0x14(%ebp)
100011bb:    8b c8                   mov    %eax,%ecx
100011bd:    c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%ebp)
100011c4:    e8 c7 04 00 00          call   0x10001690
```

disassemble

RNN

CNN

```
10001180:    55                      push   %ebp
10001181:    8b ec                   mov    %esp,%ebp
10001183:    6a ff                   push   $0xffffffff
10001185:    68 21 80 00 10          push   $0x10008021
1000118a:    64 a1 00 00 00 00       mov    %fs:0x0,%eax
10001190:    50                      push   %eax
10001191:    83 ec 08                sub    $0x8,%esp
10001194:    a1 00 40 02 10          mov    0x10024000,%eax
10001199:    33 c5                   xor    %ebp,%eax
1000119b:    89 45 f0                mov    %eax,-0x10(%ebp)
1000119e:    53                      push   %ebx
1000119f:    56                      push   %esi
100011a0:    57                      push   %edi
100011a1:    50                      push   %eax
100011a2:    8d 45 f4                lea    -0xc(%ebp),%eax
100011a5:    64 a3 00 00 00 00       mov    %eax,%fs:0x0
100011ab:    6a 14                   push   $0x14
100011ad:    6a 0c                   push   $0xc
100011af:    ff 15 44 b2 01 10       call   *0x1001b244
100011b5:    83 c4 08                add    $0x8,%esp
100011b8:    89 45 ec                mov    %eax,-0x14(%ebp)
100011bb:    8b c8                   mov    %eax,%ecx
100011bd:    c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%ebp)
100011c4:    e8 c7 04 00 00          call   0x10001690
```

이상 탐지

elastic

md5	cosine	edit
02a7993fcd5fea4442271e91e12d2df7	0.85	0.73
07FADB006486953439CE0092651FD7A6	0.21	0.32
344fbbbedc59a0a5108da10d4afd2152	0.90	0.87

유사도 검사

악성코드 의심 파일 분석

01

프로젝트 소개

02

수행 내용

03

기대 효과

데이터 수집

kaggle
microsoft
malware prediction

정상 파일 80,000개

KISA
Korea Internet & Security Agency
정보보호 R&D
데이터챌린지
금융보안원
FINANCIAL SECURITY INSTITUTE

악성 파일 300,000개

자체 개발
웹 크롤러

시스템 DLL 파일
STEAM사 게임 인스톨러
정상 파일 50,000개

니모닉 추출

disassemble

IDA

```
10001180:    55                    push   %ebp
10001181:    8b ec                 mov    %esp,%ebp
10001183:    6a ff                 push   $0xffffffff
10001185:    68 21 80 00 10        push   $0x10008021
1000118a:    64 a1 00 00 00 00     mov    %fs:0x0,%eax
10001190:    50                    push   %eax
10001191:    83 ec 08              sub    $0x8,%esp
10001194:    a1 00 40 02 10        mov    0x10024000,%eax
10001199:    33 c5                 xor    %ebp,%eax
1000119b:    89 45 f0              mov    %eax,-0x10(%ebp)
1000119e:    53                    push   %ebx
1000119f:    56                    push   %esi
100011a0:    57                    push   %edi
100011a1:    50                    push   %eax
100011a2:    8d 45 f4              lea    -0xc(%ebp),%eax
100011a5:    64 a3 00 00 00 00     mov    %eax,%fs:0x0
100011ab:    6a 14                 push   $0x14
100011ad:    6a 0c                 push   $0xc
100011af:    ff 15 44 b2 01 10     call   *0x1001b244
100011b5:    83 c4 08              add    $0x8,%esp
100011b8:    89 45 ec              mov    %eax,-0x14(%ebp)
100011bb:    8b c8                 mov    %eax,%ecx
100011bd:    c7 45 fc 00 00 00 00  movl   $0x0,-0x4(%ebp)
100011c4:    e8 c7 04 00 00        call   0x10001690
```

parser

python, IDA

```
push
mov
push
push
mov
push
sub
mov
xor
mov
push
push
push
push
lea
mov
push
push
call
add
mov
mov
movl
call
```

File

Assembly code

Mnemonic

단어 임베딩

| 실험 결과 - cbow

	vec 8	vec16	vec64	vec 128
jmp	xend	xor	mov	mov
	inc	mov	jnz	jnz
	fcmovnu	inc	jz	cmp
	vfmadd213pd	or	cmp	jz
	xor	jnz	lea	test
	dec	lea	test	push
	lgdt	test	push	lea
	ht jge	cmp	xor	retn
	setno	movzx	retn	xor
	cvttps2pi	jz	inc	inc

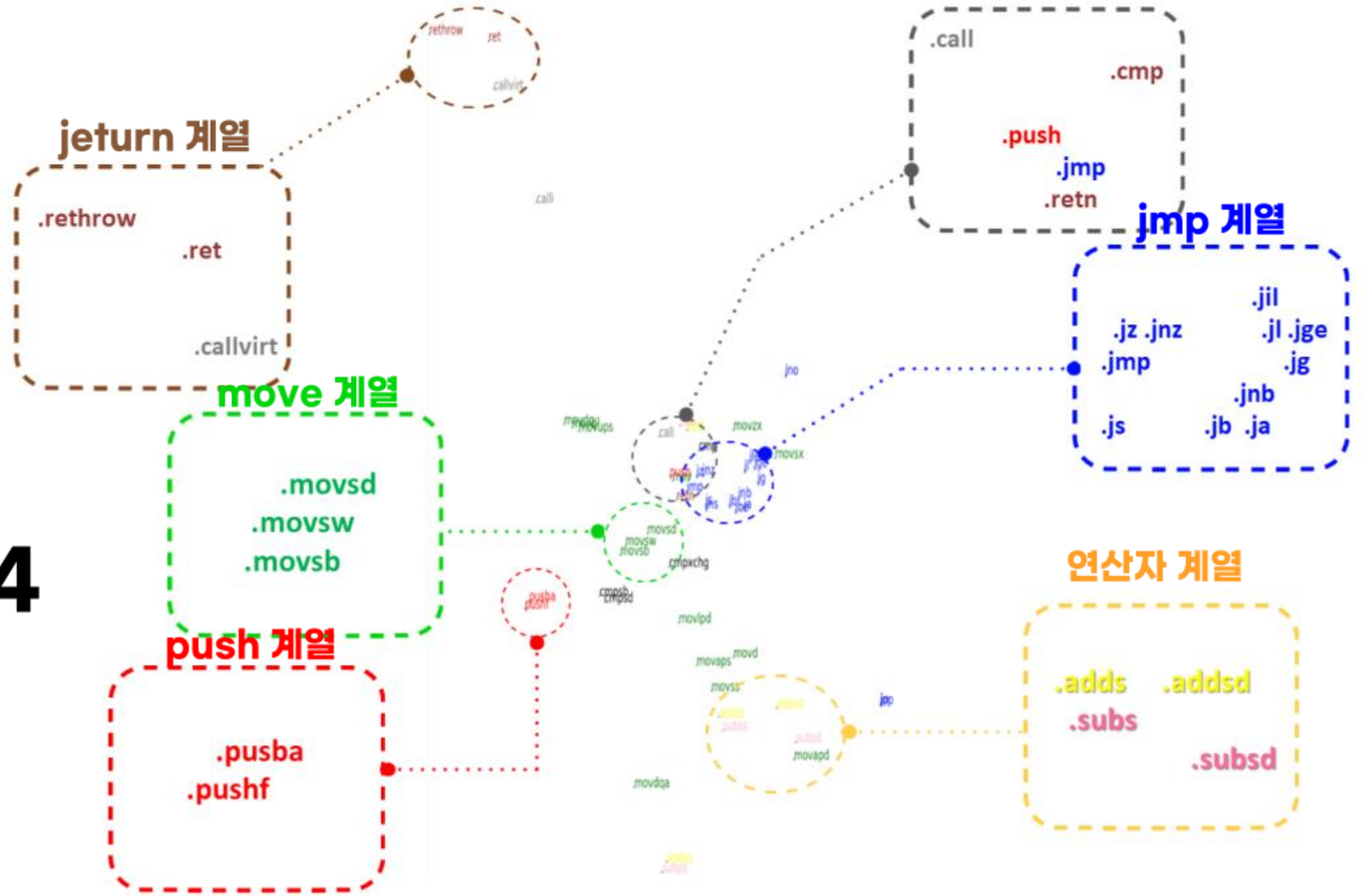
단어 임베딩

| 실험 결과

01 word2vec

02 윈도우 크기 : 2

03 특징 벡터 차원 : 64



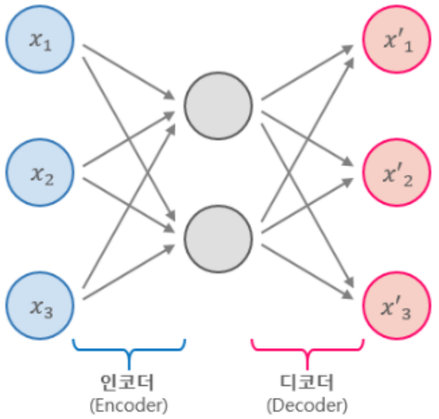
유사한 명령어가 가까운 위치로 임베딩

오토인코더 기반 이상 탐지

disassemble

IDA, Parser

push
mov
push
push
mov
push
sub
mov
xor
mov
push
push
push
push
lea
mov
push
push
call
add
mov
mov
movl
call



funtion 1

push
sub
add
str
ldr
ldr
ldr
⋮
bx

0.2

funtion 2

push
sub
add
mov
strb
ldrb
lsls
⋮
bx

0.7

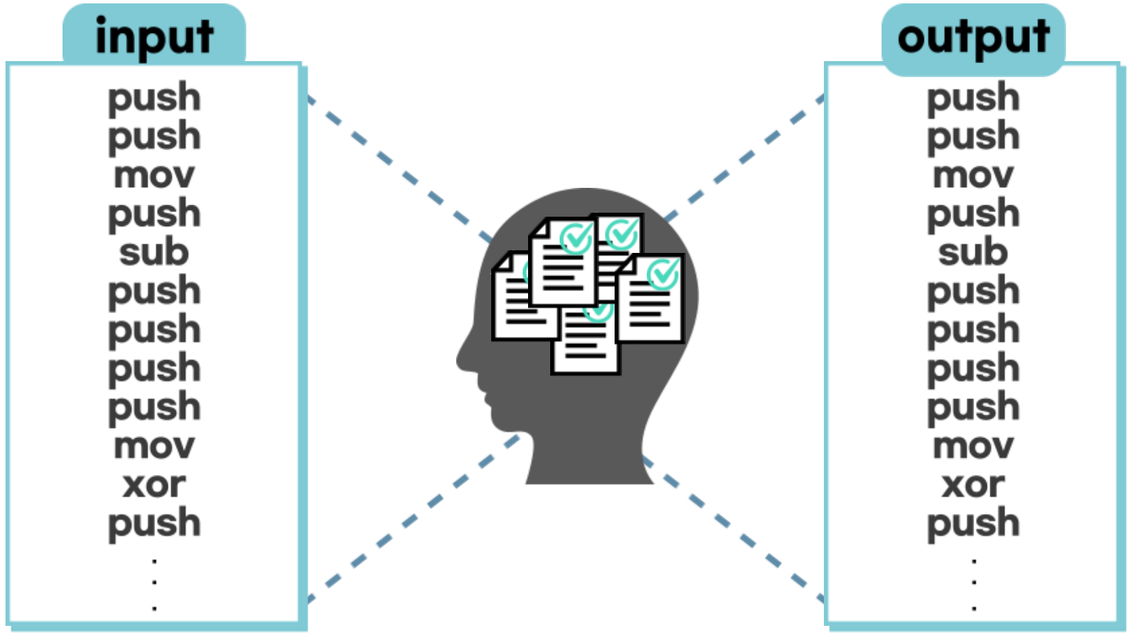
File

Mnemonic

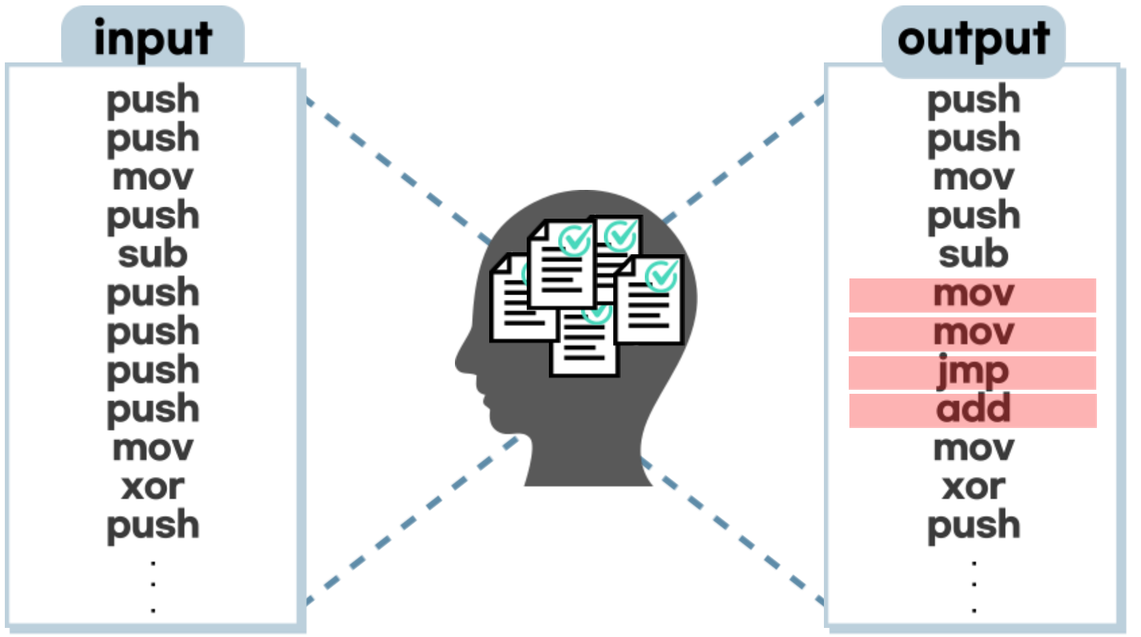
RNN 기반
오토인코더

이상탐지

이상 탐지



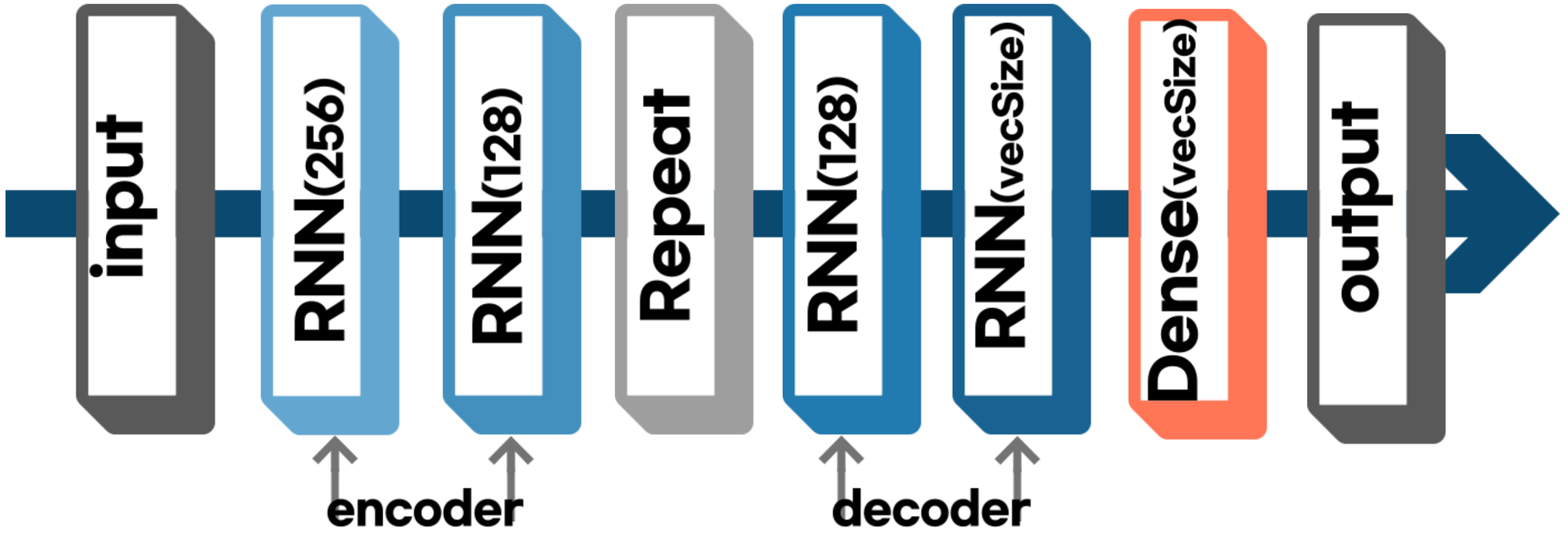
이상 탐지 X



이상 탐지 O

이상 탐지

신경망 구조


input
RNN(256)
RNN(128)
Repeat
RNN(128)
RNN(vecSize)
Dense(vecSize)
output
encoder
decoder

이상 탐지

실험 조건

01 정상 130,000개

02 벡터 크기 : 64

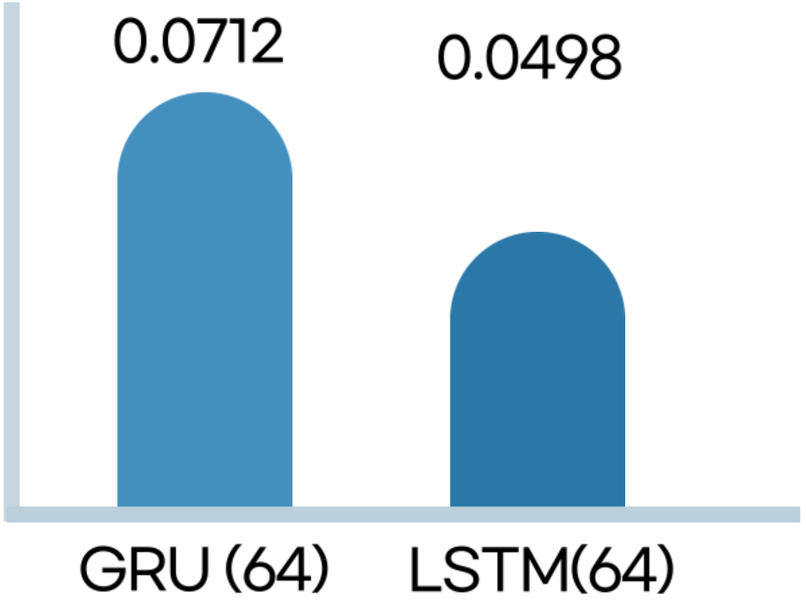
03 신경망 : GRU / LSTM

04 결과 스코어 값이 임계값(0.2)보다 크면 이상탐지

↳ 스코어 산출 방식 : MAE(mean absolute error)

이상 탐지

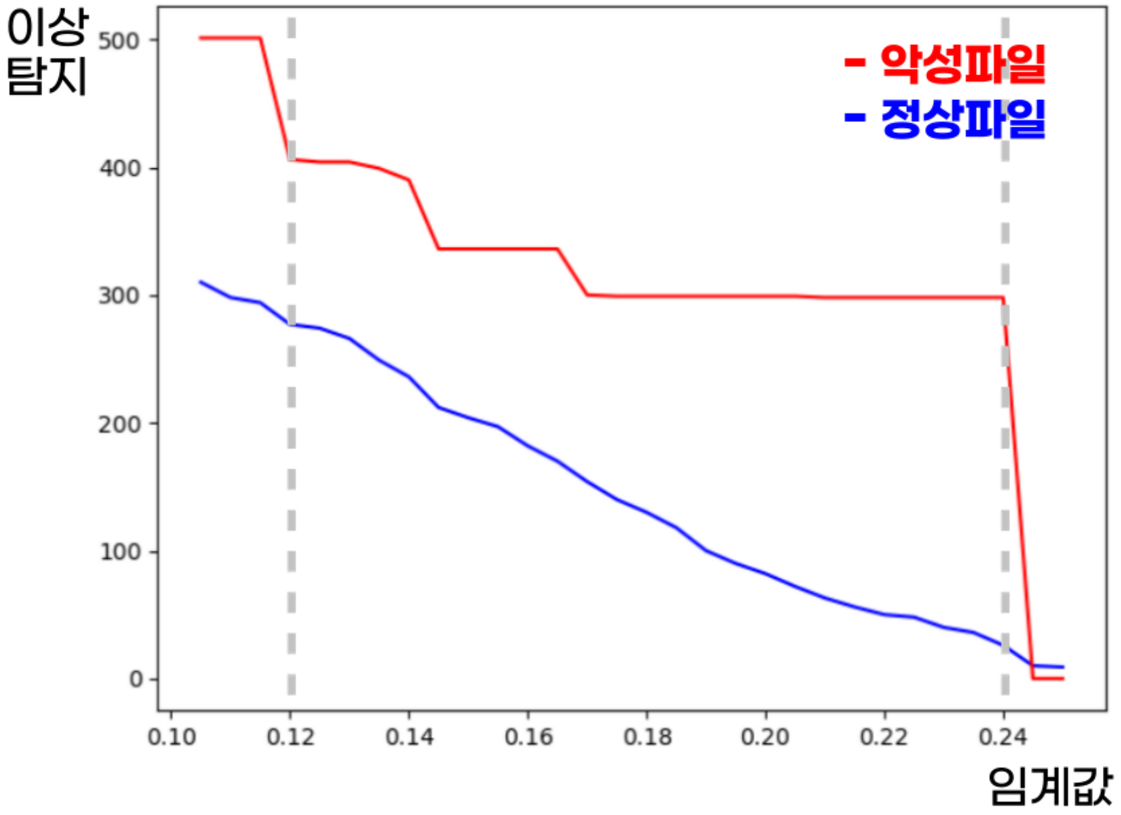
| 실험 결과 - 손실 값



GRU 보다 LSTM이 효과적

이상 탐지

| 실험 결과 - 이상 탐지



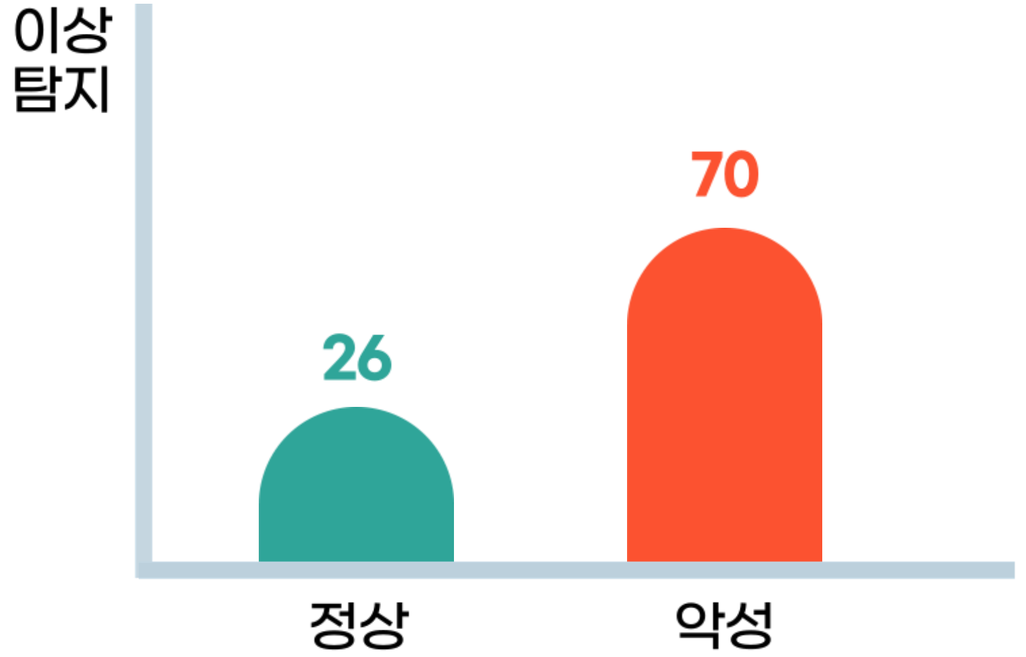
임계값 < 0.12 → 오탐 多

임계값 > 0.24 → 미탐 多

임계 값 : 0.2

이상 탐지

| 실험 결과 - 이상 탐지



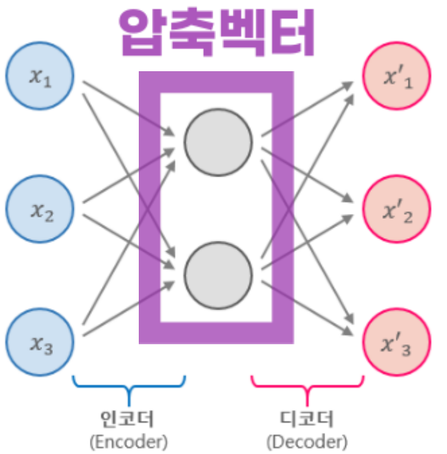
이상 탐지 비율

정상 < 악성

압축벡터 기반 유사도 검사

push
mov
push
push
mov
push
sub
mov
xor
mov
push
push
push
push
lea
mov
push
push
call
add
mov
mov
movl
call

Mnemonic



CNN 기반
오토인코더

압축벡터 기반 유사도 검사

```
"vector" : [
        0.5706483,
        0.7005931,
        0.40261483,
        0.6934074,
        0.54094464,
        0.4278791,
        0.13182177,
        0.40346572,
        0.37148055,
        0.45193487,
        0.44311434,
]
```

cosine 유사도

elastic

정상 767만개 악성 238만개

정상 : 20%

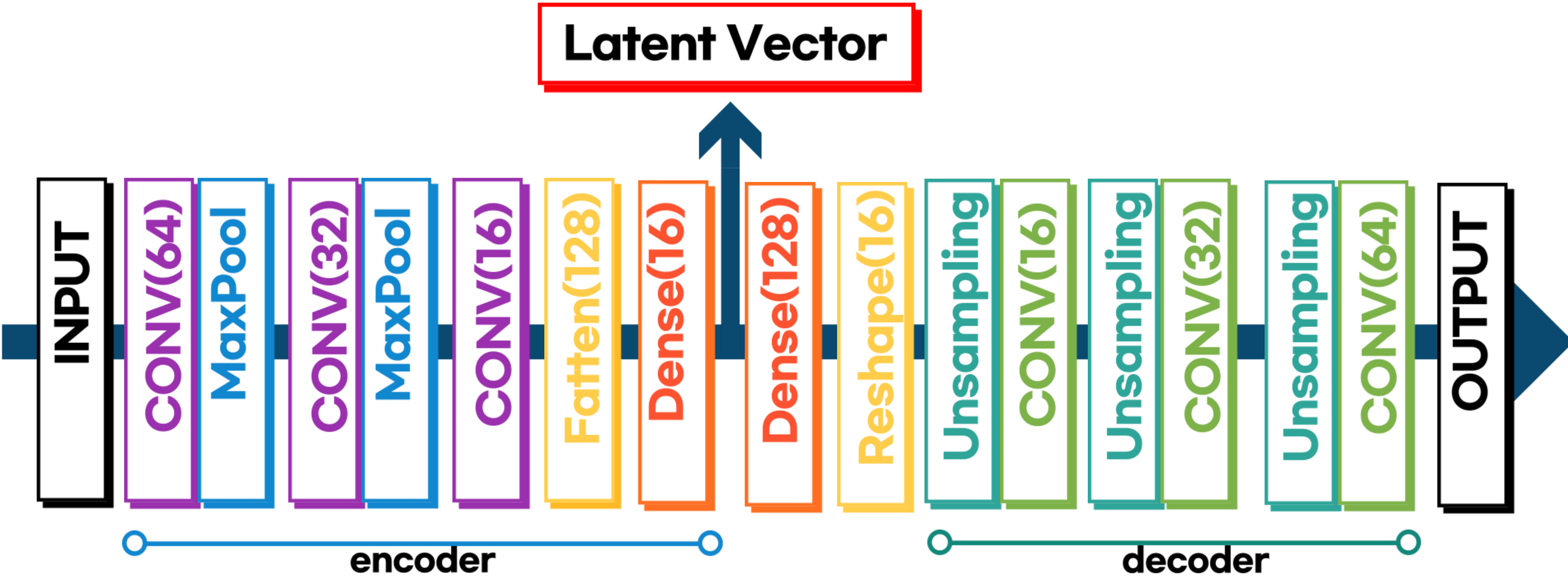
md5 : 02a7993fcd5fea4442271e91e12d2df7
md5 : 07FADB006486953439CE0092651FD7A6
md5 : 344fbbbedc59a0a5108da10d4afd2152
⋮

악성 : 80%

md5 : 02a7993fcd5fea4442271e91e12d2df7
md5 : 07FADB006486953439CE0092651FD7A6
md5 : 344fbbbedc59a0a5108da10d4afd2152
md5 : 02a7993fcd5fea4442271e91e12d2df7
md5 : 07FADB006486953439CE0092651FD7A6
md5 : 344fbbbedc59a0a5108da10d4afd2152
⋮

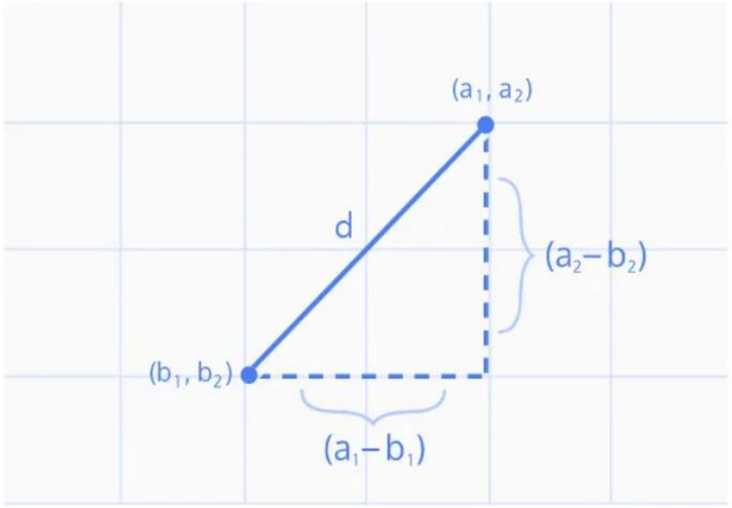
압축벡터 **유사도 검색** **악성/정상 비율**

압축벡터 기반 유사도 검사



압축벡터 기반 유사도 검사

| 압축벡터 검증

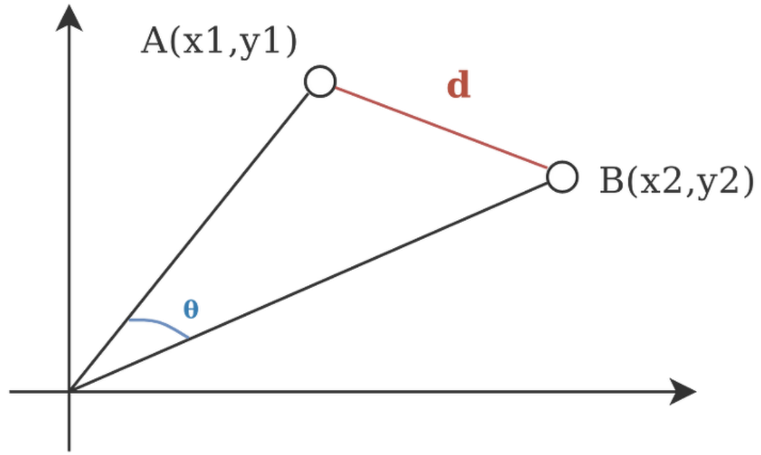


유클리드 거리

		M	O	N	K	E	Y
	0	1	1	1	1	1	1
M	0	2	1	1	1	1	1
O	0	0	2	1	1	1	1
N	0	0	0	2	1	1	1
E	0	0	0	0	2	2	1
Y	0	0	0	0	0	0	2

2 0 1

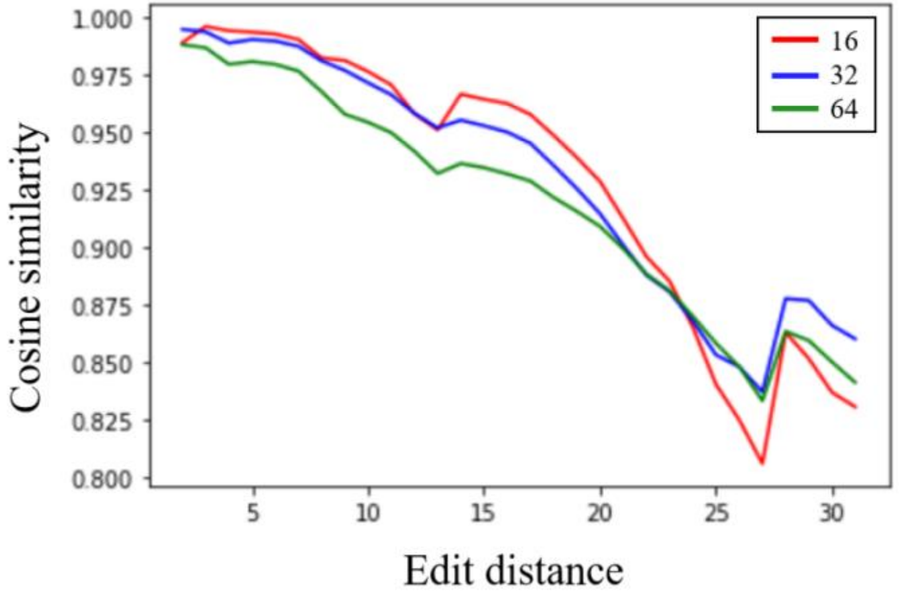
편집 거리



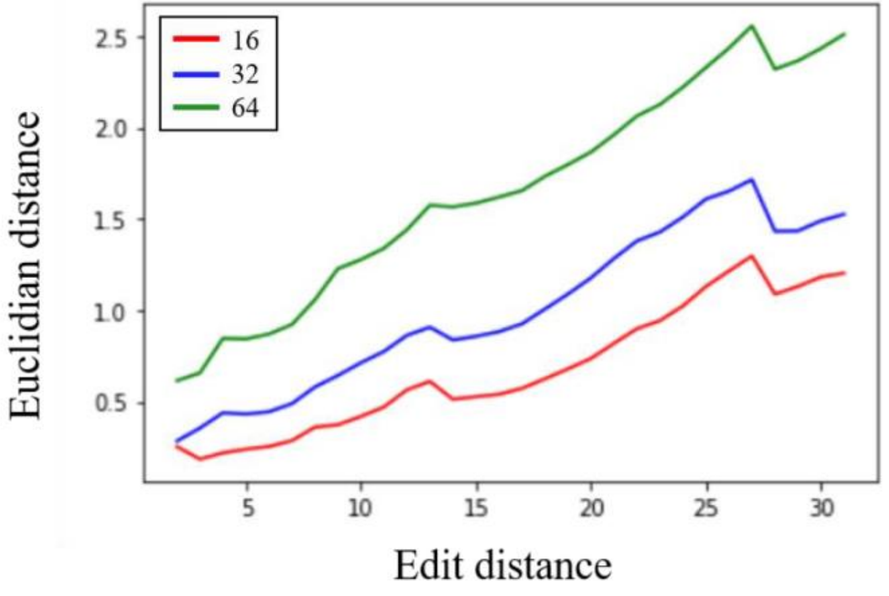
코사인 유사도

압축벡터 기반 유사도 검사

실험 결과



코사인 거리



유클리드 거리

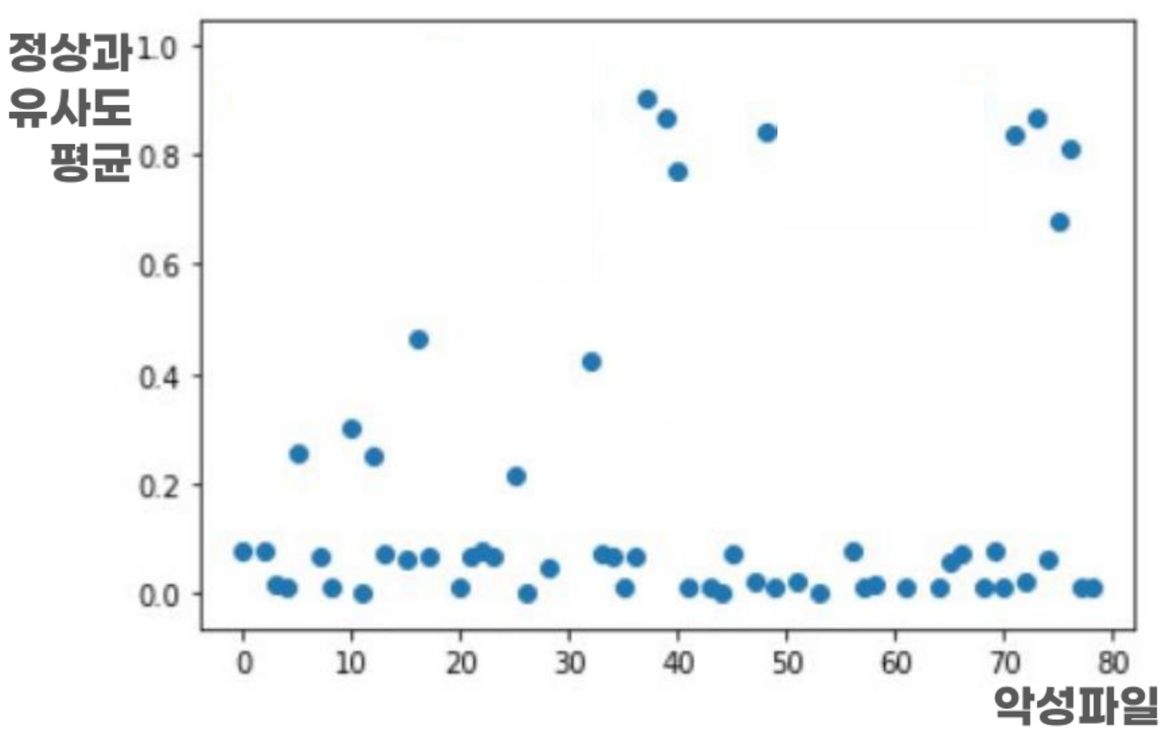
압축벡터 기반 유사도 검사

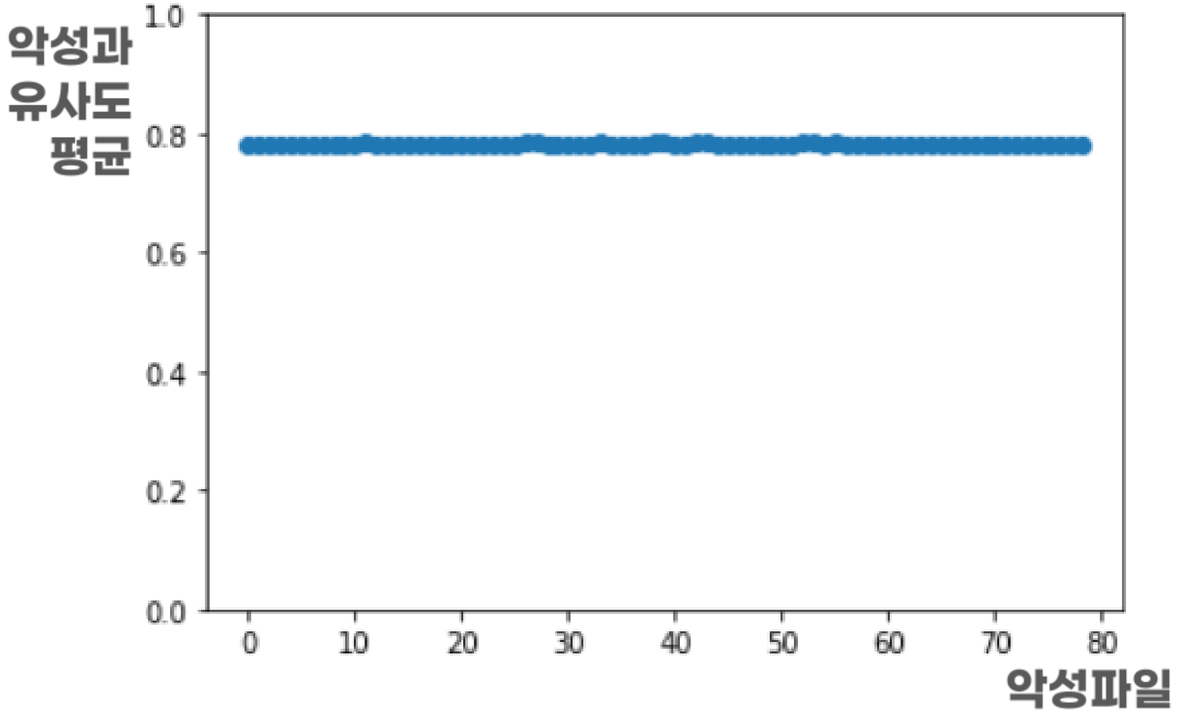
MD5 : 84c82835a5d21bbcf75a61706d8ab549

MD5 : adaf4e58e185f91e3af4c3a47b29ce63

압축벡터 기반 유사도 검사

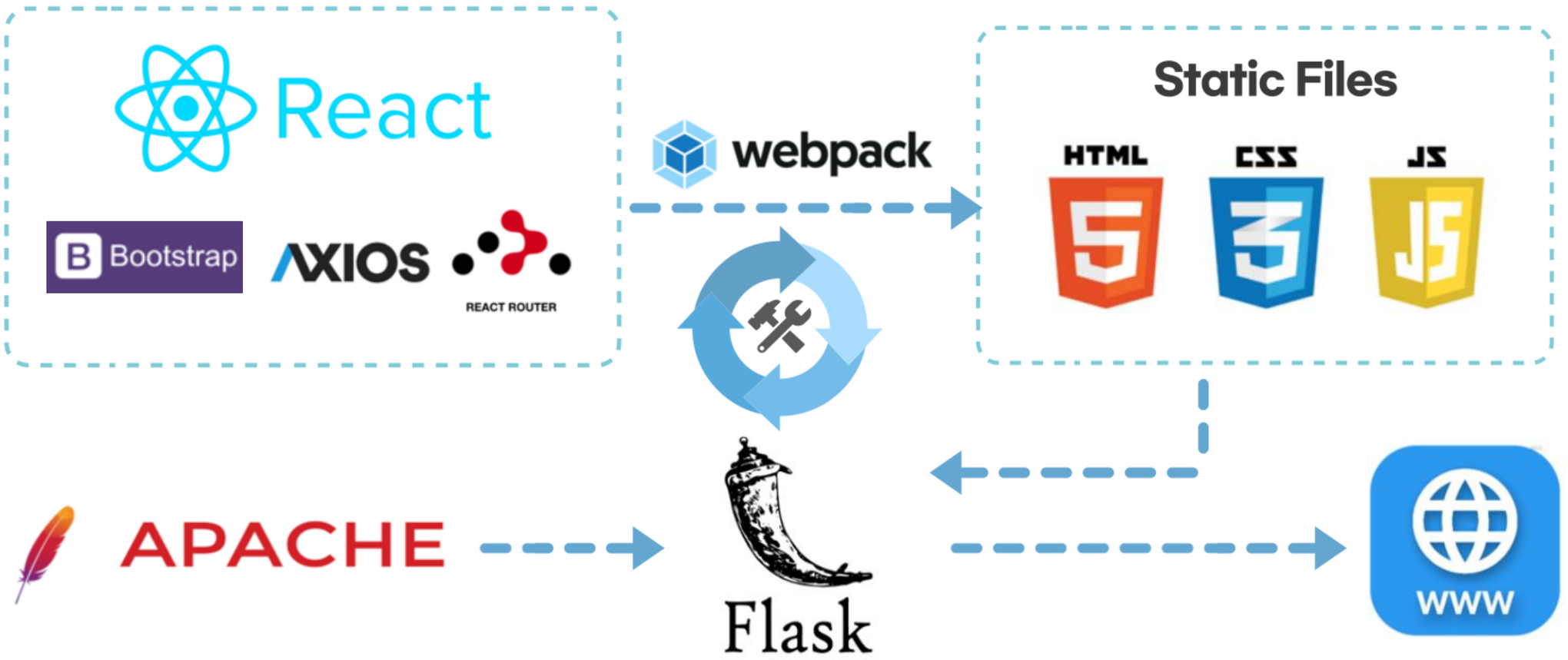
실험 결과



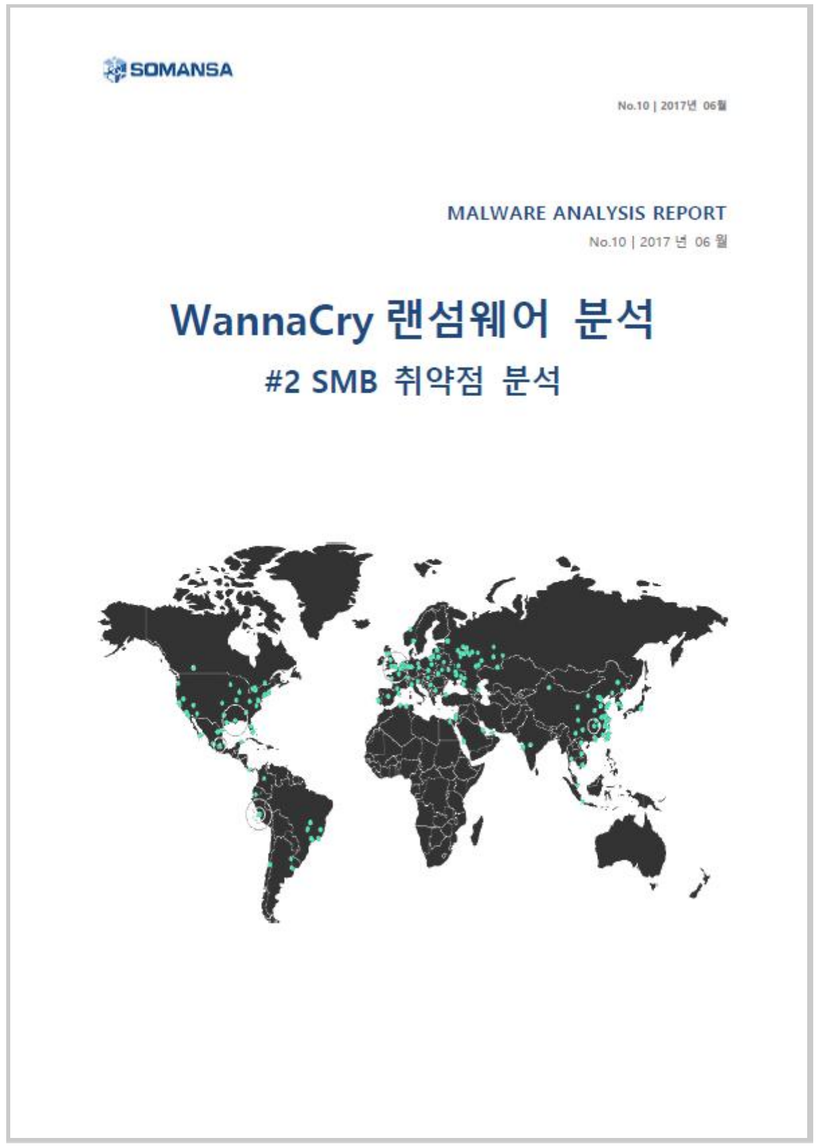


악성 파일간의 유사도가 높음

웹 구현


React
Bootstrap
AXIOS
REACT ROUTER
webpack
Static Files
HTML
CSS
JS
APACHE
Flask
WWW

검증



전문가 분석 보고서

WannaCry랜섬웨어

출처 : SOMANSA

MD5 : 84c82835a5d21bbcf75a61706d8ab549
SHA256 : ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA

검증

158c641f829540b70f69c2f6e5669eb7
b6c51ba69de58e0297a3d95781ce77a8

업로드한 파일의 함수

push	eax; lpStartupInfo
push	esi; lpCurrentDirectory
push	esi; lpEnvironment
push	8000000h; dwCreationFlags
push	esi; bInheritHandles
push	esi; lpThreadAttributes
push	esi; lpProcessAttributes
mov	[ebp+StartupInfo.dwFlags], edi
push	[ebp+lpCommandLine]; lpCommandLine

```
lea     eax, [ebp+ProcessInformation]
mov     [ebp+StartupInfo.cb], 44h
push    eax             ; lpProcessInformation
lea     eax, [ebp+StartupInfo]
push    eax             ; lpStartupInfo
push    edx             ; lpCurrentDirectory
push    edx             ; lpEnvironment
push    8000000h        ; dwCreationFlags
push    edx             ; bInheritHandles
push    edx             ; lpThreadAttributes
push    edx             ; lpProcessAttributes
push    offset Dest     ; lpCommandLine
push    edx             ; lpApplicationName
mov     [ebp+StartupInfo.
mov     [ebp+StartupInfo.
call    ds:CreateProcessA
test    eax, eax
```

randsomeware의 main함수 이상탐지

검증

:8b9faeac0b7a1fd809e7d55a10e9bd1

:c7779eef73bb0d0ec70f5fd4ee9b094

업로드한 파일의 함수

```
lea     eax, [ebp+Buffer]
push    0; lpFilePart
push    eax; lpBuffer
push    208h; nBufferLength
push    offset FileName; "tasksche.exe"
call    ds:GetFullPathNameA
lea     eax, [ebp+Buffer]
push    eax
call    sub_401CE8
```

tasksche.exe randsomeware의 network 통신시 사용

```
lea     edi, [esp+46Ch+var_40F]
push    offset aTaskhsvc_exe ; "taskhsvc.exe"
rep stosd
stosw
push    offset aTor      ; "Tor"
push    offset PathName ; "TaskData"
lea     ecx, [esp+478h+Dest]
push    offset aSSS      ; "%s\%s\%s"
push    ecx              ; Dest
stosb
call    sprintf
mov     esi, ds:GetFileAttributesA
add     esp, 14h
lea     edx, [esp+46Ch+Dest]
push    edx              ; lpFileName
call    esi ; GetFileAttributesA
cmp     eax, 0FFFFFFFFh
```

[그림 14] Tor 파일 확인

아래의 경로에서 Taskhsvc.exe 파일이 존재하는지 확인한다. 해당 파일은 tor 파일을 이름만 변경한 것으로 네트워크 통신 시 이 파일을 사용한다.

01 프로젝트 소개

02 수행 내용

03 기대 효과

기대 효과

비용절약

적은 인원으로
많은 파일 분석

시간절약

새로운 악성코드에
신속히 대처

교육효과

입문자들에게
길잡이의 역할

시연 영상으로 이어집니다.