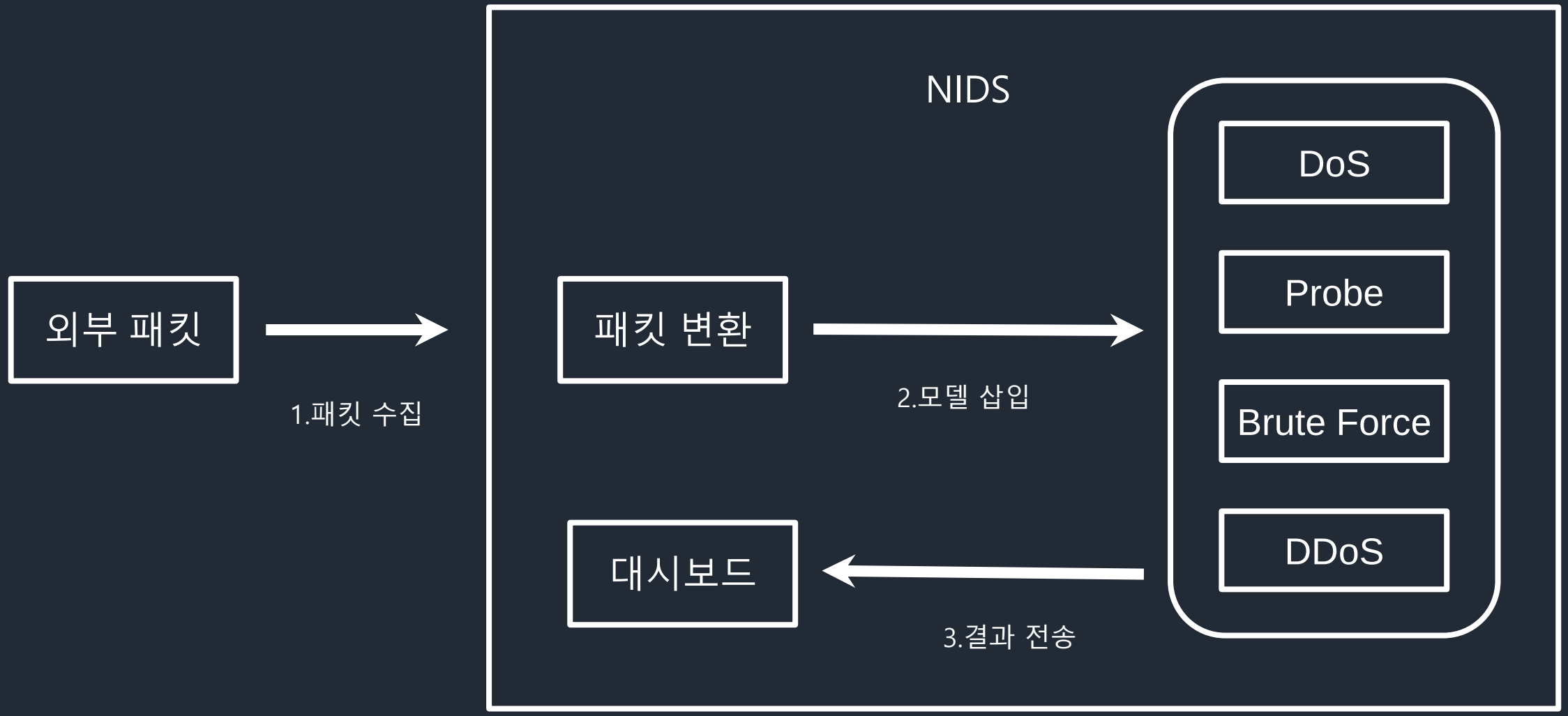
인공지능 기반 침입탐지시스템

김명섭
박재희
송우영
천호범
박채환

목차

구상도


NIDS
외부 패킷
1.패킷 수집
패킷 변환
2.모델 삽입
DoS
Probe
Brute Force
DDoS
대시보드
3.결과 전송

개발 환경 및 개발 내용

OS

개발언어

개발환경

개발 환경 및 개발 내용

NSL-KDD

	Number of Records:					
	Total	Normal	DoS	Probe	U2R	R2L
KDDTrain+20%	25192	13449 (53%)	9234 (37%)	2289 (9.16%)	11 (0.04%)	209 (0.8%)
KDDTrain+	125973	67343 (53%)	45927	11656	52	995
KDDTest+	22544	9711 (43%)				

CEC-CIC-IDS

attack	Tools	Duration	Attacker	Victim
Bruteforce attack	FTP – Patator, SSH – Patator	One day	Kali linux	Ubuntu 16.4 (Web Server)
DoS attack	Hulk, GoldenEye, Slowloris, Slowhttptest	One day	Kali linux	Ubuntu 16.4 (Apache)
DoS attack	Heartleech	One day	Kali linux	Ubuntu 12.04 (Open SSL)
Web attack	Damn Vulnerable Web App (DVWA), In-house selenium framework (XSS and Brute-force)	Two days	Kali linux	Ubuntu 16.4 (Web Server)
Infiltration attack	First level: Dropbox download in a windows machine, Second Level: Nmap and portscan	Two days	Kali linux	Windows Vista and Macintosh
Botnet attack	Ares (developed by Python): remote shell, file upload/download, capturing screenshots and key logging	One day	Kali linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)
DDoS+PortScan	Low Orbit Ion Canon (LOIC) for UDP, TCP, or HTTP requests	Two days	Kali linux	Windows Vist

개발 환경 및 개발 내용

	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	...	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Date	Time	label
0	0	0	112641719	3	0	0	0	0	0	0.000000	...	0.0	0	0	56320859.5	139.300036	56320958	56320761	14022018	83101	0
1	0	0	112641466	3	0	0	0	0	0	0.000000	...	0.0	0	0	56320733.0	114.551299	56320814	56320652	14022018	83350	0
2	0	0	112638623	3	0	0	0	0	0	0.000000	...	0.0	0	0	56319311.5	301.934596	56319525	56319098	14022018	83639	0
3	22	6	6453966	15	10	1239	2273	744	0	82.600000	...	0.0	0	0	0.0	0.000000	0	0	14022018	84013	0
4	22	6	8804066	14	11	1143	2209	744	0	81.642857	...	0.0	0	0	0.0	0.000000	0	0	14022018	84023	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
383028	22	6	18	1	1	0	0	0	0	0.000000	...	0.0	0	0	0.0	0.000000	0	0	14022018	32720	1
383060	22	6	8	1	1	0	0	0	0	0.000000	...	0.0	0	0	0.0	0.000000	0	0	14022018	31959	1
383064	22	6	8	1	1	0	0	0	0	0.000000	...	0.0	0	0	0.0	0.000000	0	0	14022018	32605	1
				1	1	0	0	0	0	0.000000	...	0.0	0	0	0.0	0.000000	0	0	14022018	32917	1
				1	1	0	0	0	0	0.000000	...	0.0	0	0	0.0	0.000000	0	0	14022018	32931	1

CEC-CIC-IDS

DoS
Probe

DDoS
BruteForce

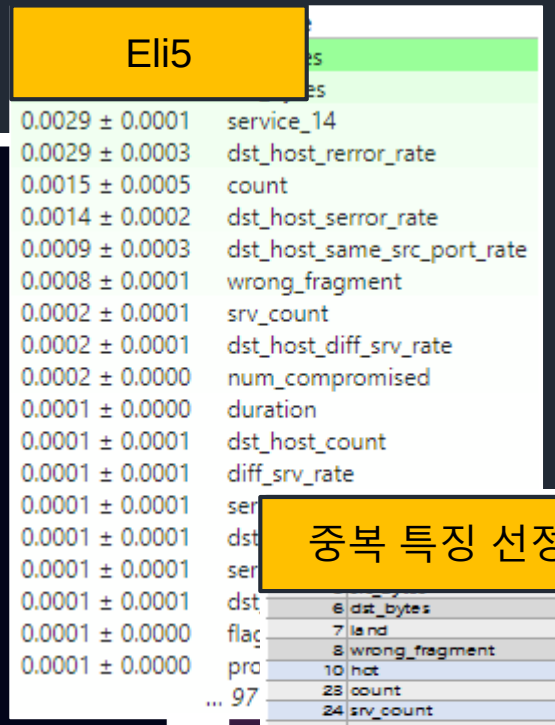
NSL-KDD

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_h
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	0.03	0.17	
1	0	udp	other	SF	146	0	0	0	0	0	...	0.60	0.88	0.00
2	0	tcp	http	SF	232	8153	0	0	0	0	...	0.00	0.03	0.04
3	0	tcp	http	SF	199	420	0	0	0	0	...	0.00	0.00	0.00
4	0	tcp	http	SF	287	2251	0	0	0	0	...	0.00	0.12	0.03
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12126	0	tcp	pop_3	RSTO	0	36	0	0	0	0	...	0.03	0.00	0.00
12127	0	tcp	pop_3	RSTO	0	36	0	0	0	0	...	0.11	0.02	0.03
12128	5	tcp	imap4	RSTO	0	44	0	0	0	0	...	0.04	0.01	0.06
12129	2	tcp	telnet	SF	24	109	0	0	0	0	...	0.03	0.01	0.02
12130	0	tcp	sunrpc	REJ	0	0	0	0	0	0	...	0.03	0.00	0.00

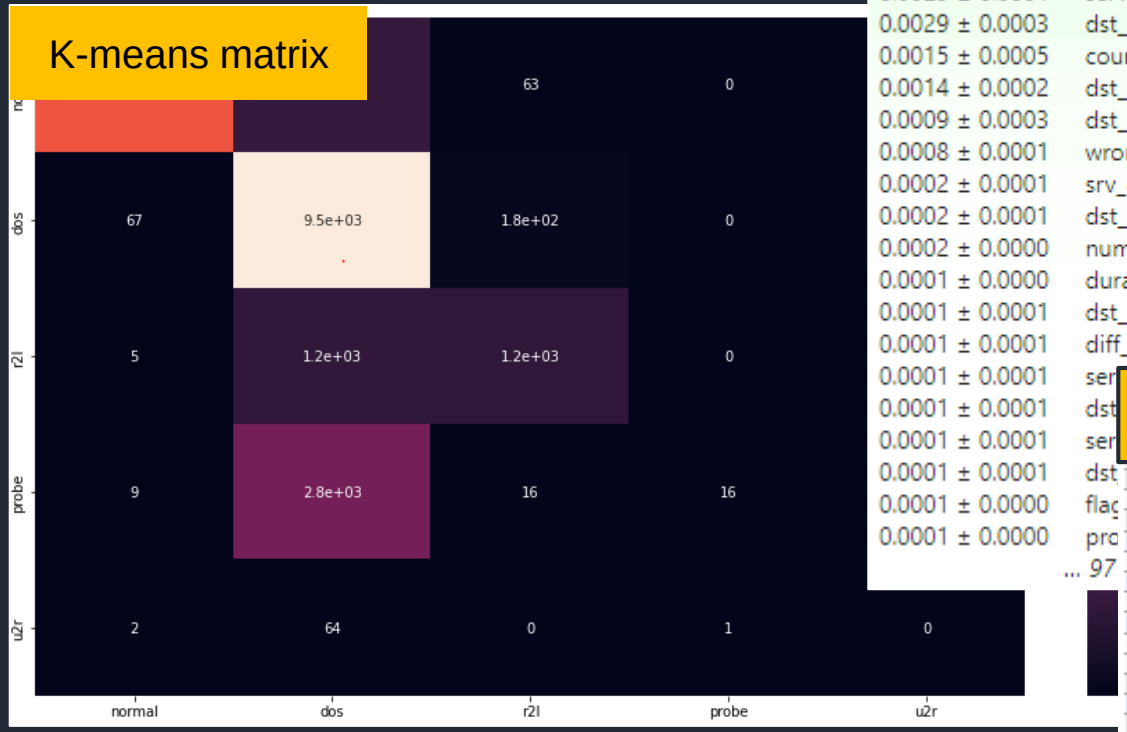
개발 환경 및 개발 내용

DoS

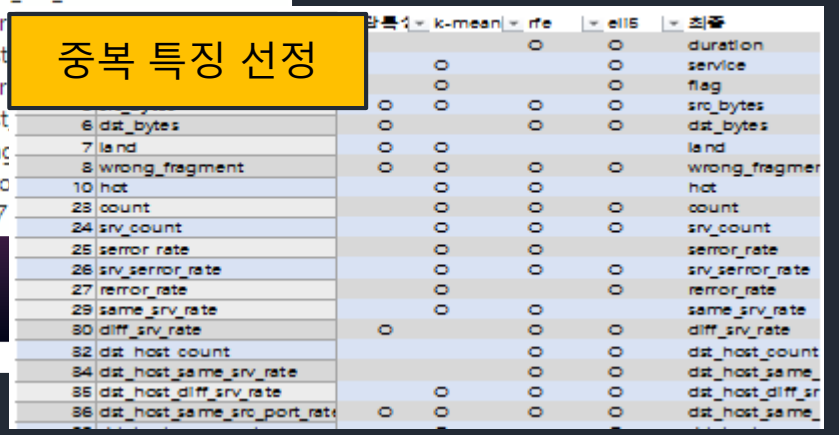
Eli5



K-means matrix



중복 특징 선정



DoS 연관 특징, K-means, RFE, Eli5를 사용하여
2가지 이상 중복되는 22가지의 특징 추출

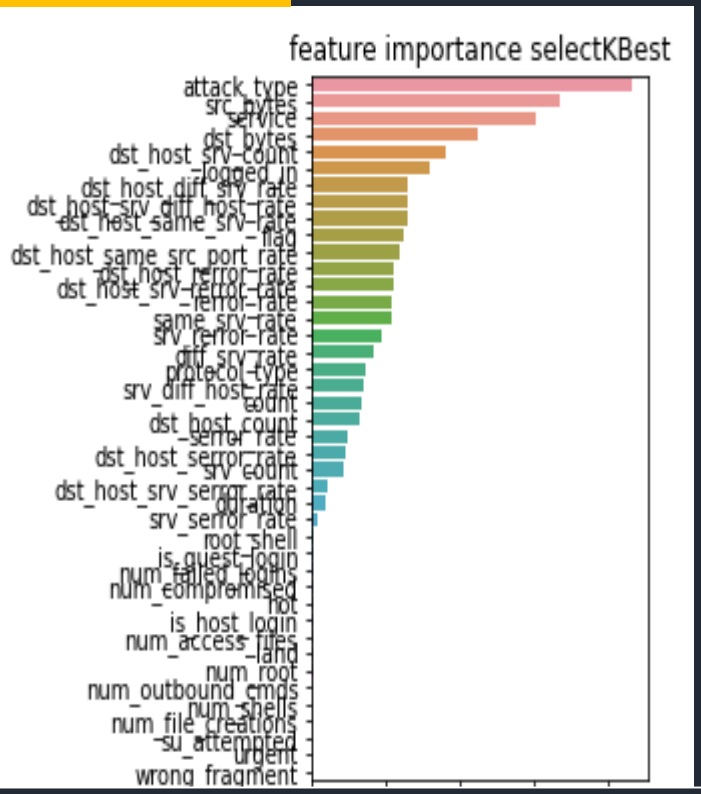
개발 환경 및 개발 내용

Probe

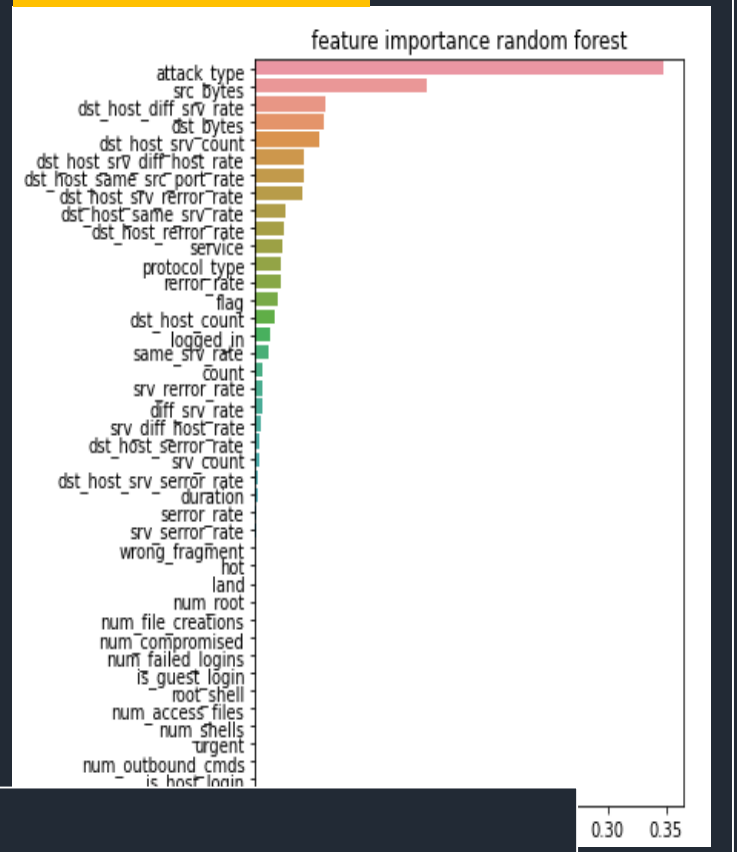
상관관계

srv_count	0.111070
srv_serror_rate	0.137003
dst_host_serror_rate	0.149518
serror_rate	0.149840
dst_host_srv_serror_rate	0.172094
duration	0.184341
srv_diff_host_rate	0.214969
service	0.232685
count	0.248795
protocol_type	0.254831
dst_host_srv_diff_host_rate	0.386236
diff_srv_rate	0.435325
dst_host_same_srv_rate	0.439889
same_srv_rate	0.467348
flag	0.473075
dst_host_rerror_rate	0.485056
dst_host_same_src_port_rate	0.498905
srv_rerror_rate	0.506317
rerror_rate	0.510143
dst_host_srv_rerror_rate	0.513276
logged_in	0.518808
dst_host_srv_count	0.533153
dst_ho	
attack	

selectKBest



Random Forest



상관관계분석, selectKBest, Random Forest
세가지를 이용해서 19가지의 특징 추출

개발 환경 및 개발 내용

DDoS

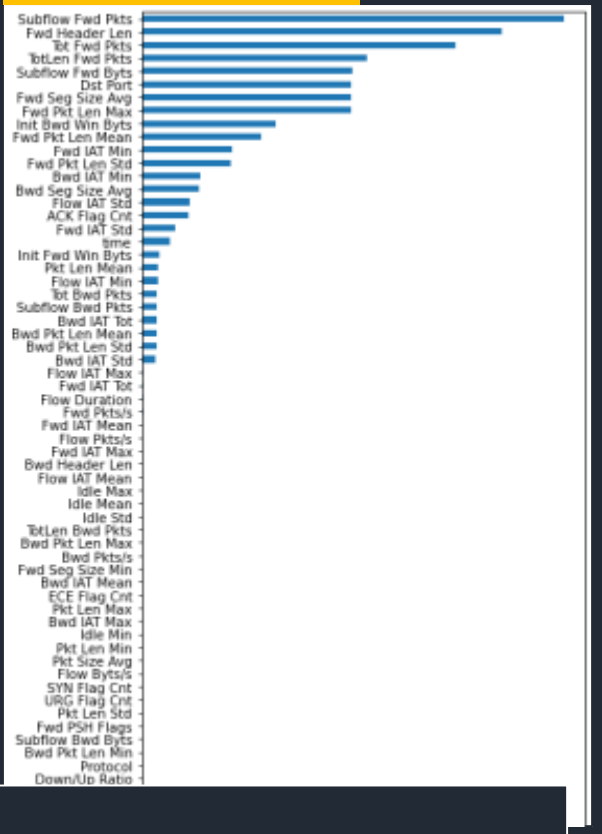
상관관계

DDOS	1.000000
ACKFlagCnt	0.723469
Down_UpRatio	0.311150
FwdSegSizeMin	0.052687
BwdIATStd	0.004575
FINFlagCnt	-0.001348
BwdPktLenMin	-0.003408
SYNFlagCnt	-0.006463
FwdPSHFlags	-0.006463
URGFlagCnt	-0.007623
BwdIATTot	-0.010795
BwdIATMax	-0.013204
FlowIATMin	-0.017639
FwdIATMin	-0.017648
PktLenMin	-0.031391
FwdPktLenMin	-0.031398
ActiveStd	-0.034544
IdleMin	-0.038241
FlowIATMean	-0.043785
FwdIATMean	-0.043818
ActiveMin	-0.047908
ActiveMax	-0.049021
IdleStd	-0.050956
ActiveMean	-0.051045
IdleMean	-0.051471
FwdIATStd	-0.054518
FlowIATStd	-0.054607
FlowIATMax	-0.054678
FwdIATMax	-0.054797
IdleMax	-0.055571
Protocol	-0.057821
FlowDuration	-0.059035
FwdIATTot	-0.059045
BwdIATMean	-0.072481
BwdPktLenMax	-0.109388
SubflowBwdByts	-0.109498
TotLenBwdPkts	-0.109498
time	-0.130305
BwdIATMin	-0.143371
FwdPkts_s	-0.214014
BwdHe	
BwdPkt	
FlowP	

Eli5

Weight	Feature
0.0023 ± 0.0001	Dst Port
0.0016 ± 0.0001	Subflow Fwd Byts
0.0013 ± 0.0001	Fwd Header Len
0.0013 ± 0.0001	Tot Fwd Pkts
0.0012 ± 0.0001	Pkt Len Mean
0.0012 ± 0.0001	Subflow Bwd Pkts
0.0012 ± 0.0001	Tot Bwd Pkts
0.0011 ± 0.0002	Fwd Pkt Len Max
0.0009 ± 0.0001	Subflow Fwd Pkts
0.0009 ± 0.0001	Fwd Pkt Len Mean
0.0008 ± 0.0001	Fwd Pkt Len Std
0.0008 ± 0.0001	Fwd Seg Size Avg
0.0008 ± 0.0001	Init Bwd Win Byts
0.0008 ± 0.0001	TotLen Fwd Pkts
0.0008 ± 0.0001	Bwd IAT Min
0.0008 ± 0.0001	Pkt Len Var
0.0008 ± 0.0001	Pkt Size Avg
0.0007 ± 0.0001	Pkt Len Max
0.0007 ± 0.0001	PSH Flag Cnt
0.0007 ± 0.0001	Bwd Header Len
0.0007 ± 0.0001	Bwd Pkts/s
0.0005 ± 0.0001	Flow IAT Mean
0.0005 ± 0.0001	Fwd IAT Std
0.0001 ± 0.0000	Flow IAT Std
0.0000 ± 0.0000	Init Fwd Win Byts
0.0000 ± 0.0000	Idle Mean
0.0000 ± 0.0000	Idle Max
0.0000 ± 0.0000	Fwd Seg Size Min
0.0000 ± 0.0000	Idle Min
0.0000 ± 0.0000	Fwd Pkt Len Min
0.0000 ± 0.0000	Fwd Act Data Pkts
0.0000 ± 0.0000	Bwd Pkt Len Max
0.0000 ± 0.0000	Bwd Seg Size Avg
0 ± 0.0000	Bwd Pkts/b Avg
0 ± 0.0000	Bwd Blk Rate Avg
0 ± 0.0000	Flow Byts/s
0 ± 0.0000	Bwd Pkt Len Std
0 ± 0.0000	Bwd Pkt Len Mean

Random Forest



상관관계분석, eli5, Random Forest
세가지를 이용해서 21가지의 특징을 추출

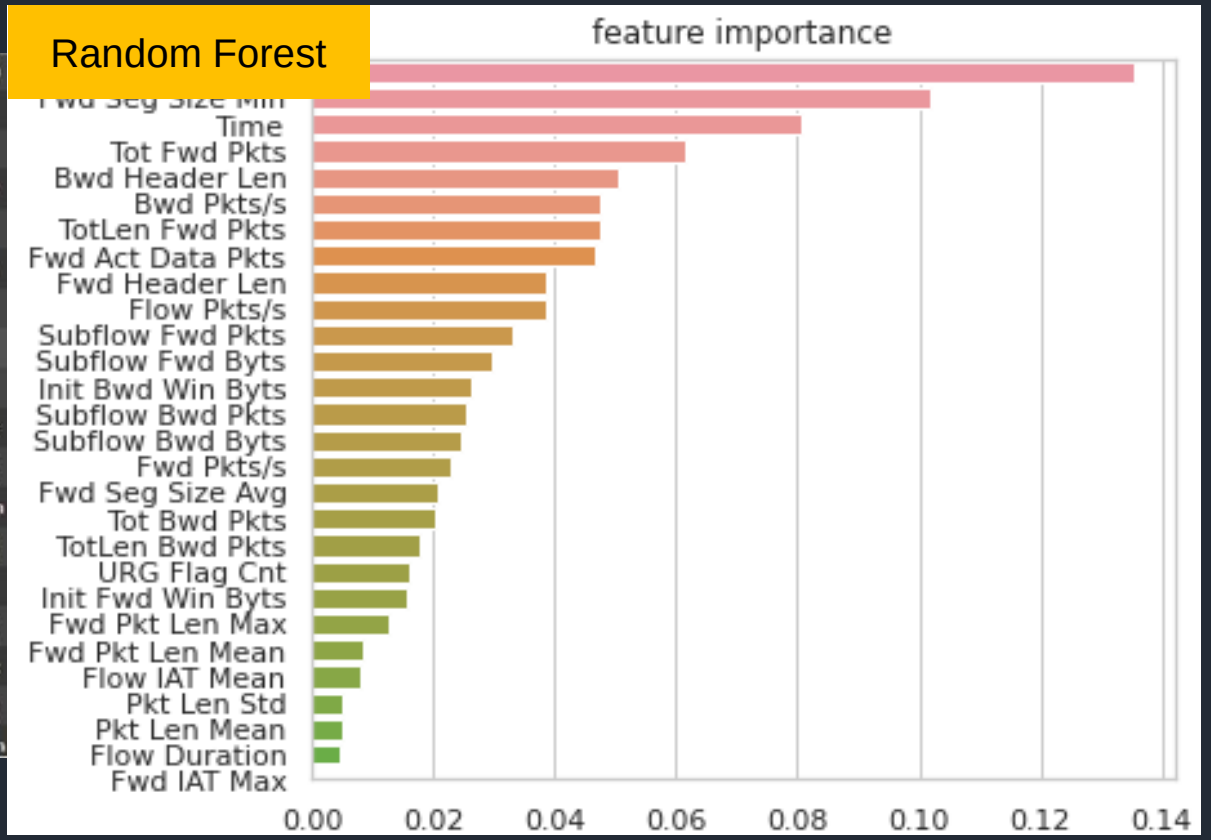
개발 환경 및 개발 내용

BruteForce

상관관계

Time	3.755921e-02	Fwd Seg Size Min	0.246614	Fwd Seg Size Min
Init Bwd Win Byts	2.544292e-02	Dst Port	0.246244	Dst Port
Bwd IAT Max	6.094368e-04	Init Fwd Win Byts	0.091865	Flow Duration
Fwd Seg Size Min	4.919330e-04	TotLen Bwd Pkts	0.000000	Fwd Pkt Len Max
ACK Flag Cnt	3.044140e-05	Flow IAT Std	0.000000	Bwd Pkts/s
Flow Pkts/s	1.278539e-05	Fwd Pkt Len Min	0.000000	Flow IAT Mean
Pkt Len Var	1.035008e-05	Protocol	0.000000	Protocol
Fwd Pkt Len Std	6.088280e-06	Flow Duration	0.000000	Tot Fwd Pkts
Bwd Pkt Len Std	6.088280e-06	Tot Fwd Pkts	0.000000	Tot Bwd Pkts
Bwd Pkts/s	4.870624e-06	Tot Bwd Pkts	0.000000	TotLen Fwd Pkts
Fwd IAT Max	2.435312e-06	TotLen Fwd Pkts	0.000000	TotLen Bwd Pkts
Flow IAT Mean	2.435312e-06	Fwd Pkt Len Mean	0.000000	Fwd Pkt Len Mean
Fwd Pkts/s	1.217656e-06	Fwd Pkt Len Max	0.000000	Fwd Pkt Len Min
Pkt Len Mean	6.088280e-07	Flow IAT Min	0.000000	Flow IAT Min
Fwd Header Len	6.088280e-07	Bwd Pkts/s	0.000000	Fwd Pkt Len Std
Bwd IAT Std	0.000000e+00	Fwd Pkt Len Std	0.000000	Bwd Pkt Len Max
		Bwd Pkt Len Max	0.000000	Bwd Pkt Len Min
		Bwd Pkt Len Min	0.000000	Bwd Pkt Len Mean

Random Forest



Random Forest, Gradient Boosting, XGBoosting를
사용하여 21가지의 특징을 추출

개발 환경 및 개발 내용

DoS

```
def create_model():
    K.clear_session()
    model = Sequential()
    model.add(LSTM(4, input_shape=(28, 1)))
    model.add(Dense(2, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model = create_model()
model.sum
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 4)                 96

 dense (Dense)               (None, 2)                 10

=================================================================
Total params: 106
Trainable params: 106
Non-trainable params: 0
```

```
Epoch 745: saving model to ./LSTM_745.h5
197/825 [======>.....................] - ETA: 10s - loss: 0.0156 - accuracy: 0.9951
Epoch 745: saving model to ./LSTM_745.h5
296/825 [=========>....................] - ETA: 9s - loss: 0.0149 - accuracy: 0.9953
Epoch 745: saving model to ./LSTM_745.h5
398/825 [============>.................] - ETA: 7s - loss: 0.0153 - accuracy: 0.9953
Epoch 745: saving model to ./LSTM_745.h5
498/825 [================>.............] - ETA: 5s - loss: 0.0151 - accuracy: 0.9954
Epoch 745: saving model to ./LSTM_745.h5
597/825 [====================>.........] - ETA: 4s - loss: 0.0152 - accuracy: 0.9953
Epoch 745: saving model to ./LSTM_745.h5
699/825 [========================>.....] - ETA: 2s - loss: 0.0158 - accuracy: 0.9951
Epoch 745: saving model to ./LSTM_745.h5
798/825 [============================>.] - ETA: 0s - loss: 0.0159 - accuracy: 0.9951
```

30000번 학습을 실행했을 때,
loss값은 0.0.149,
accuracy: 0.9953로 가장 성능이 좋았다.

개발 환경 및 개발 내용

Probe

```
# DNN

dnn = Sequential()
dnn.add(Dense(64, activation='relu', input_shape=(19,)))
dnn.add(Dropout(0.1))
dnn.add(Dense(32, activation='relu'))
dnn.add(Dropout(0.1))
dnn.add(Dense(16, activation='relu'))
dnn.add(Dropout(0.
dnn.add(Dense(8, a
dnn.add(Dropout(0.
dnn.add(Dense(4, a
dnn.add(Dense(1, a

dnn.compile(loss='                                    rics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                1280
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                528
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 136
_________________________________________________________________
dropout_3 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 36
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 5
=================================================================
Total params: 4,065
Trainable params: 4,065
Non-trainable params: 0
```

```
528/528 [==============================] - 2s 5ms/step - loss: 0.0062 - accuracy: 0.9983 - val_loss: 0.2256 - val_accuracy: 0.9943
Epoch 5950/10000
528/528 [==============================] - 2s 5ms/step - loss: 0.0042 - accuracy: 0.9985 - val_loss: 0.2988 - val_accuracy: 0.9941
Epoch 5951/10000
528/528 [==============================] - 2s 4ms/step - loss: 0.0059 - accuracy: 0.9980 - val_loss: 0.2543 - val_accuracy: 0.9946
Epoch 5952/10000
528/528 [==============================] - 2s 4ms/step - loss: 0.0070 - accuracy: 0.9980 - val_loss: 0.2219 - val_accuracy: 0.9943
Epoch 5953/10000
528/528 [==============================] - 2s 5ms/step - loss: 0.0055 - accuracy: 0.9981 - val_loss: 0.2090 - val_accuracy: 0.9938
Epoch 5954/10000
528/528 [==============================] - 2s 4ms/step - loss: 0.0059 - accuracy: 0.9985 - val_loss: 0.2450 - val_accuracy: 0.9938
Epoch 5955/10000
528/528 [==============================] - 2s 4ms/step - loss: 0.0040 - accuracy: 0.9988 - val_loss: 0.2824 - val_accuracy: 0.9936
Epoch 5956/10000
528/528 [==============================] - 2s 4ms/step - loss: 0.0058 - accuracy: 0.9982 - val_loss: 0.2040 - val_accuracy: 0.9938
Epoch 5957/10000
528/528 [==============================] - 2s 4ms/step - loss: 0.0069 - accuracy: 0.9981 - val_loss: 0.1779 - val_accuracy: 0.9938
Epoch 5958/10000
528/528 [==============================] - 2s 5ms/step - loss: 0.0086 - accuracy: 0.9984 - val_loss: 0.1817 - val_accuracy: 0.9936
Epoch 5959/10000
528/528 [==============================] - 2s 4ms/step - loss: 0.0057 - accuracy: 0.9984 - val_loss: 0.1913 - val_accuracy: 0.9934
```

160000번 학습을 실행했을 때,
loss값은 0.0832,
accuracy: 0.9741로 가장 성능이 좋았다.

개발 환경 및 개발 내용

DDOS

```
model = Sequential()

model.add(LSTM(128, input_shape=(1,21), return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(128))
model.add(Dropout(0.1))

model.add(Dense(32, activation='tanh'))
model.add(Dropout(0.2))

model.add(Dense(1, act

opt = tf.keras.optimiz

model.compile(
    loss='binary_cross
    optimizer=opt,
    metrics=['accuracy
)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 cu_dnnlstm (CuDNNLSTM)      (None, 1, 128)            77312

 dropout (Dropout)           (None, 1, 128)            0

 cu_dnnlstm_1 (CuDNNLSTM)    (None, 128)               132096

 dropout_1 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 32)                4128

 dropout_2 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 213,569
Trainable params: 213,569
Non-trainable params: 0
```

```
Epoch 1/100
100/100 [==============================] - 1s 6ms/step - loss: 0.6574 - accuracy: 0.6983 - val_loss: 0.6196 - val_accuracy: 0.7000
Epoch 2/100
100/100 [==============================] - 0s 4ms/step - loss: 0.5800 - accuracy: 0.7105 - val_loss: 0.5294 - val_accuracy: 0.7269
Epoch 3/100
100/100 [==============================] - 0s 4ms/step - loss: 0.5118 - accuracy: 0.7342 - val_loss: 0.4603 - val_accuracy: 0.7781
Epoch 4/100
100/100 [==============================] - 0s 4ms/step - loss: 0.4585 - accuracy: 0.7941 - val_loss: 0.4023 - val_accuracy: 0.8050
Epoch 5/100
100/100 [==============================] - 0s 4ms/step - loss: 0.4115 - accuracy: 0.8005 - val_loss: 0.3569 - val_accuracy: 0.8181
Epoch 6/100
100/100 [==============================] - 0s 4ms/step - loss: 0.3810 - accuracy: 0.8128 - val_loss: 0.3225 - val_accuracy: 0.8531
Epoch 7/100
100/100 [==============================] - 0s 4ms/step - loss: 0.3541 - accuracy: 0.8341 - val_loss: 0.2953 - val_accuracy: 0.8838
Epoch 8/100
100/100 [==============================] - 0s 4ms/step - loss: 0.3342 - accuracy: 0.8542 - val_loss: 0.2731 - val_accuracy: 0.8969
Epoch 9/100
100/100 [==============================] - 0s 4ms/step - loss: 0.3178 - accuracy: 0.8600 - val_loss: 0.2543 - val_accuracy: 0.9062
```

88000번 학습을 실행했을 때,
loss값은 0.5674,
accuracy: 0.9399로 가장 성능이 좋았다.

개발 환경 및 개발 내용

Brute Force

```
1 def a_lstm():
2     model = Sequential()
3     model.add(CuDNNGRU(64,input_shape=(1, 21), return_sequences=True))
4     model.add(Dropout(0.2))
5     model.add(CuDNNGRU(128, return_sequences=False))
6     model.add(Dropout(0.2))
7     model.add(Dense(1))
8     model.add(Activation('sigmoid'))
9
10    m                                                          ccuracy'])
11    r
```

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 cu_dnngru (CuDNNGRU)         (None, 1, 64)             16704

 dropout (Dropout)            (None, 1, 64)             0

 cu_dnngru_1 (CuDNNGRU)       (None, 128)               74496

 dropout_1 (Dropout)          (None, 128)               0

 dense (Dense)                (None, 1)                 129

 activation (Activation)      (None, 1)                 0

=================================================================
Total params: 91,329
Trainable params: 91,329
Non-trainable params: 0

_________________________________________________________________
```

→

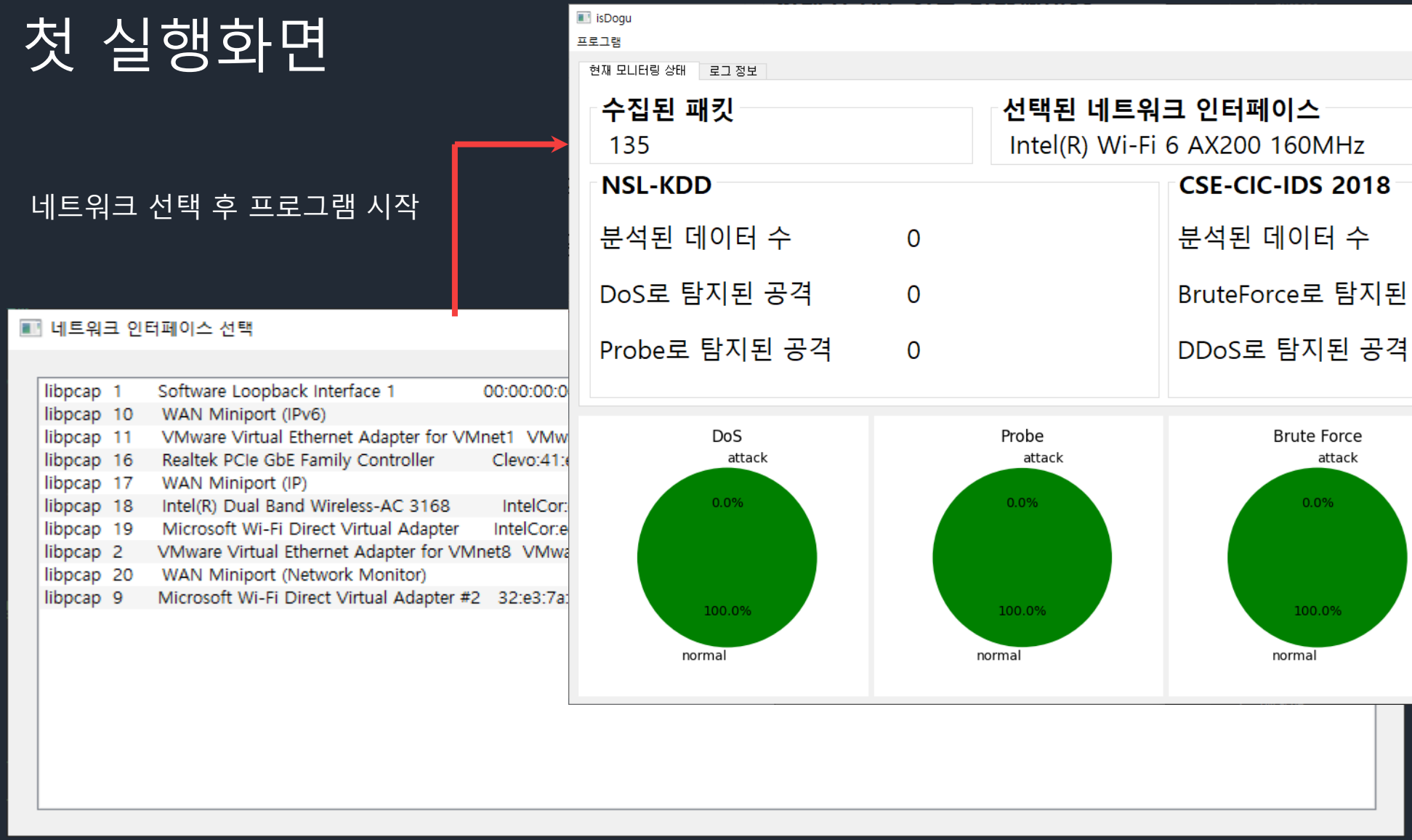
```
********************250000********************
9/9 [==============================] - 0s 3ms/step - loss: 3.9465 - accuracy: 0.9380

********************260000********************
9/9 [==============================] - 0s 3ms/step - loss: 3.7856 - accuracy: 0.9380

********************270000********************
9/9 [==============================] - 0s 3ms/step - loss: 3.9036 - accuracy: 0.8953

********************280000********************
9/9 [==============================] - 0s 3ms/step - loss: 5.2772 - accuracy: 0.0698

********************290000********************
9/9 [==============================] - 0s 3ms/step - loss: 4.3436 - accuracy: 0.8876

********************300000********************
9/9 [==============================] - 0s 3ms/step - loss: 4.8627 - accuracy: 0.9380

********************310000********************
9/9 [==============================] - 0s 3ms/step - loss: 4.3514 - accuracy: 0.9070

********************320000********************
9/9 [==============================] - 0s 4ms/step - loss: 3.1072 - accuracy: 0.9380

********************330000********************
9/9 [==============================] - 0s 3ms/step - loss: 8.6867 - accuracy: 0.1473

********************340000********************
9/9 [==============================] - 0s 3ms/step - loss: 14.6643 - accuracy: 0.0504

********************350000********************
9/9 [==============================] - 0s 4ms/step - loss: 8.8029 - accuracy: 0.0000e+00

********************360000********************
9/9 [==============================] - 0s 3ms/step - loss: 7.8244 - accuracy: 0.0039
```

320000번 학습을 실행했을 때,
loss값은 3.1072,
accuracy: 0.9380로 가장 성능이 좋았다.

개발 시스템 운영

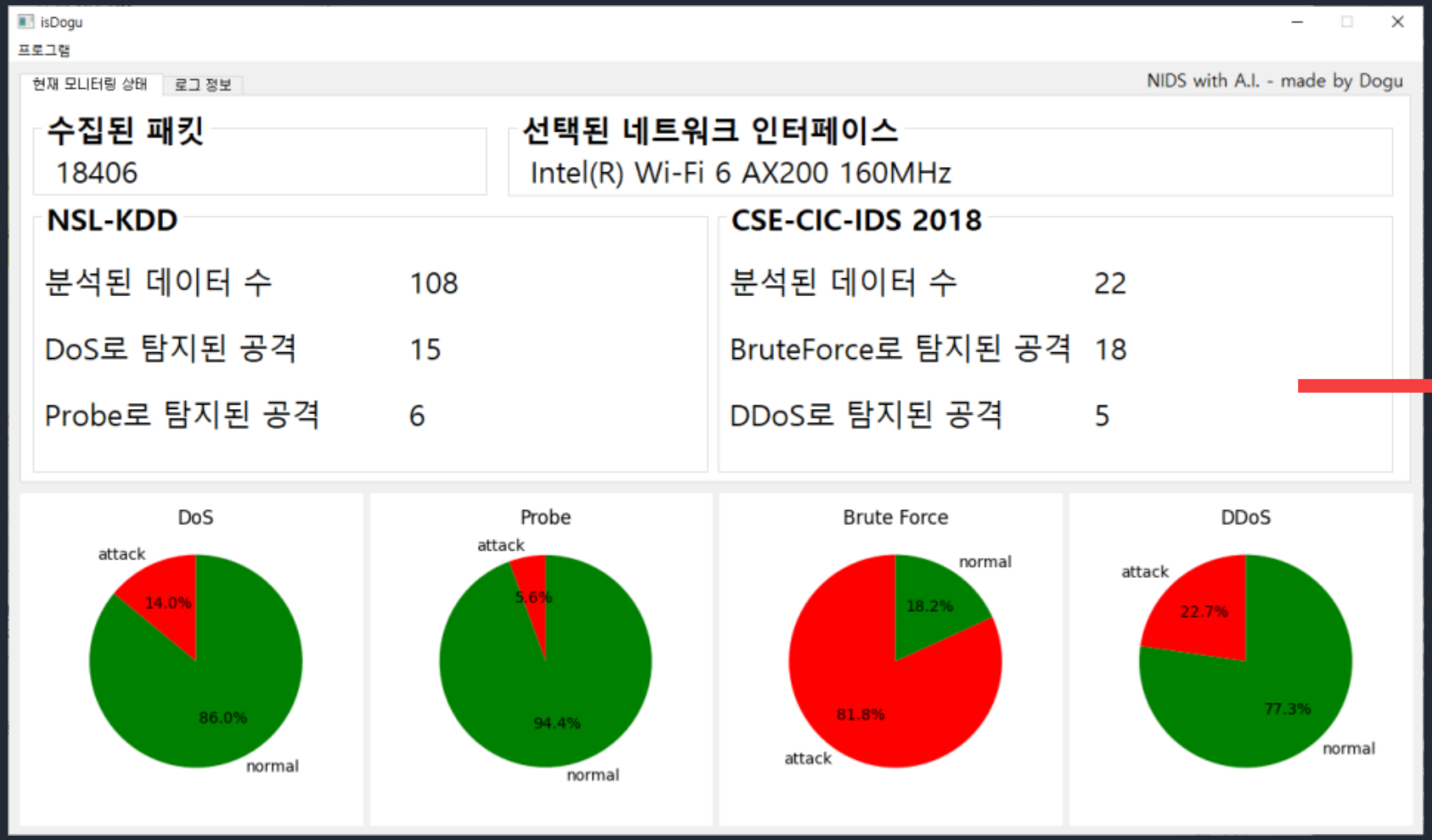
첫 실행화면

네트워크 선택 후 프로그램 시작



개발 시스템 운영

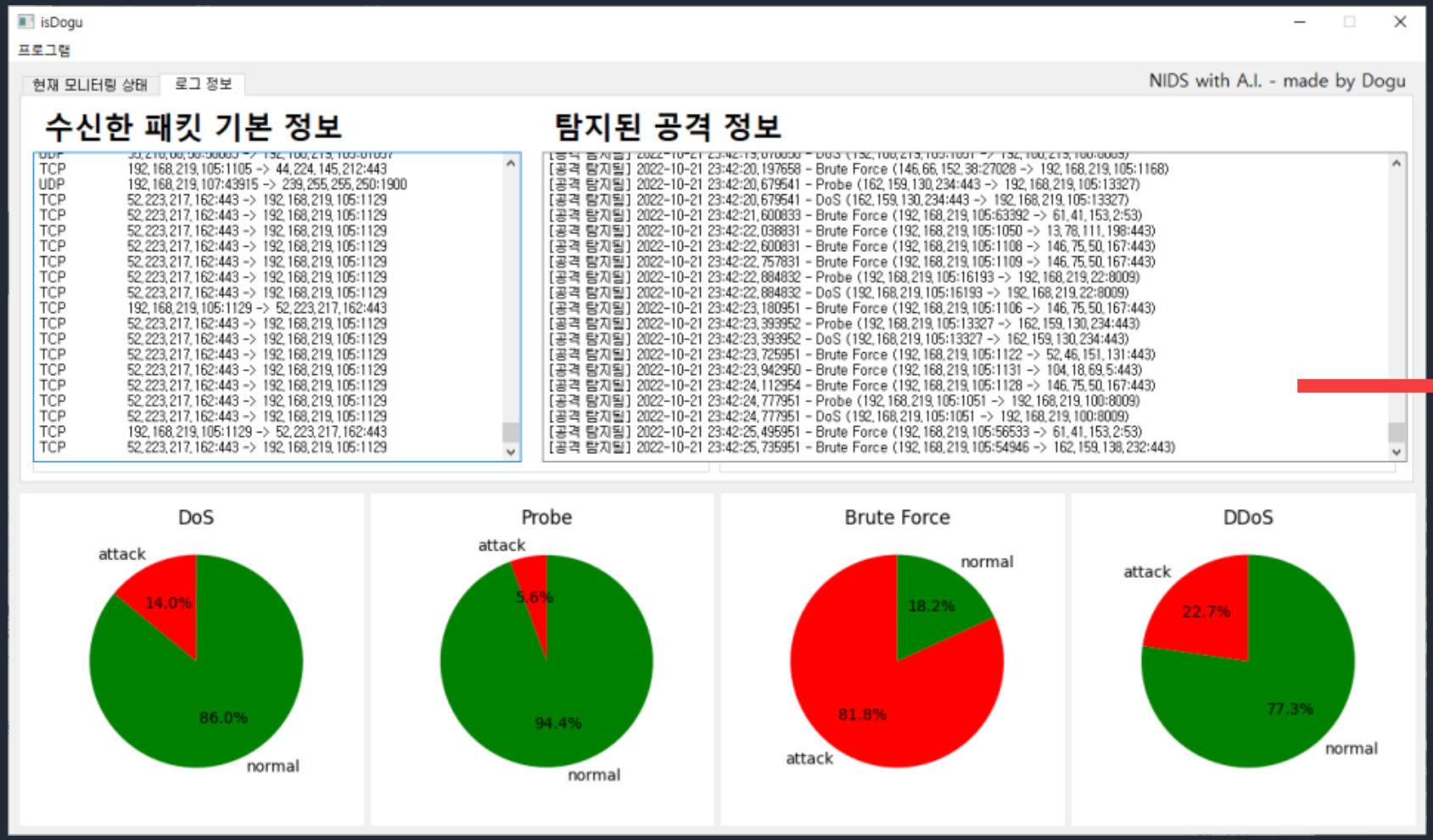
패킷 받고 공격이 탐지됐을때 화면



BruteForce로 공격 실행 시,
나오는 화면

개발 시스템 운영

로그



탐지공격에 대한 상세 정보 출력

결론 및 기대효과

- 4가지의 공격방법을 선정하여 네트워크 침입에 대해 인공지능을 사용하여 공격에 대해 빠르게 탐지할 수 있는지를 살펴보았다
- 모델별, 학습횟수에 따라 성능이 차이나는 것을 확인하였다
- 실제 패킷과 모델을 연결 하여 공격을 시도했을 때 각 공격에 대해 높은 탐지율을 보여주었다
- 지속 적인 연구를 통해 아직 알려지지 않은 네트워크 공격들과 새롭게 등장하는 공격들을 잡아 낼 수 있을 것이다.

감사합니다.