In [5]:
```
import numpy as np
import pandas as pd
import matplotlib
import seaborn as sns
import sklearn
import imblearn
import matplotlib.pyplot as plt
import time
import sklearn.metrics as m
import xgboost as xgb
```

In [4]:
```
pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting xgboost
  Downloading xgboost-1.7.4-py3-none-win_amd64.whl (89.1 MB)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from xgboost) (1.21.5)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from xgboost) (1.7.3)
Installing collected packages: xgboost
Successfully installed xgboost-1.7.4
Note: you may need to restart the kernel to use updated packages.
```

In [37]:
```
cols = [' Bwd Packet Length Std',' PSH Flag Count',' min_seg_size_forward',' Min Packet Length',' ACK Flag Cou
df1=pd.read_csv("Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv", usecols = cols)#,nrows = 50000
df2=pd.read_csv("Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv", usecols = cols)
df3=pd.read_csv("Friday-WorkingHours-Morning.pcap_ISCX.csv", usecols = cols)
df5=pd.read_csv("Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv", usecols = cols)
df6=pd.read_csv("Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv", usecols = cols)
```

In [44]:
```
df1=pd.read_csv("Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv", usecols = cols)
df1
```

Out[44]:

	Bwd Packet Length Min	Bwd Packet Length Std	Flow IAT Max	Fwd IAT Std	Bwd IAT Total	Bwd Packets/s	Min Packet Length	PSH Flag Count	ACK Flag Count	URG Flag Count	Init_Win_bytes_forward	min_seg_size_forward
0	0	0.0	3	0.0	0	0.000000	6	0	1	0	33	20
1	6	0.0	109	0.0	0	9174.311927	6	0	1	1	29	20
2	6	0.0	52	0.0	0	19230.769230	6	0	1	1	29	20
3	6	0.0	34	0.0	0	29411.764710	6	0	1	1	31	20
4	0	0.0	3	0.0	0	0.000000	6	0	1	0	32	20
...	...	...	...	...	...	...	...	...	...	...	...	...
225740	6	0.0	61	0.0	0	16393.442620	6	0	1	1	288	20
225741	6	0.0	72	0.0	0	13888.888890	6	0	1	1	288	20
225742	6	0.0	75	0.0	0	13333.333330	6	0	1	1	288	20
225743	0	0.0	48	0.0	0	0.000000	6	0	1	0	4719	20
225744	6	0.0	68	0.0	0	14705.882350	6	0	1	1	13140	20

225745 rows × 13 columns

In [39]:
```
df2=pd.read_csv("Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv", usecols = cols)
```

In [40]:
```
df2
```

Out[40]:

	Bwd Packet Length Min	Bwd Packet Length Std	Flow IAT Max	Fwd IAT Std	Bwd IAT Total	Bwd Packets/s	Min Packet Length	PSH Flag Count	ACK Flag Count	URG Flag Count	Init_Win_bytes_forward	min_s
0	0	312.675250	948537	159355.259500	317671	34.745748	0	1	0	0	29200	
1	0	312.675250	955790	159247.900800	363429	33.349680	0	1	0	0	29200	
2	0	0.000000	160	0.000000	0	6250.000000	0	0	1	1	290	
3	0	319.121427	956551	160397.049900	346851	32.221240	0	1	0	0	29200	
4	0	0.000000	49	0.000000	49	25974.025970	0	0	1	1	243	
...	...	...	...	...	...	...	...	...	...	...	...	
286462	0	972.796621	53438	10366.095180	172901	290.616157	0	1	0	0	29200	
286463	0	848.453540	173388	34491.462460	355402	155.909773	0	1	0	0	29200	
286464	0	920.298603	23268	6109.558409	138764	636.588381	0	1	0	0	29200	
286465	0	799.911092	23111	7300.372125	119753	433.979169	0	1	0	0	29200	
286466	0	849.040557	44740	9798.936450	163676	304.930241	0	1	0	0	29200	

286467 rows × 13 columns

In [38]:
```
df
```

Out[38]:

	Bwd Packet Length Min	Bwd Packet Length Std	Flow IAT Max	Fwd IAT Std	Bwd IAT Total	Bwd Packets/s	Min Packet Length	PSH Flag Count	ACK Flag Count	URG Flag Count	Init_Win_bytes_forward
0	0	585.413147	58800000	2.280000e+07	118000000	0.110354	0	1	0	0	8192
1	115	0.000000	34787	2.008284e+04	3	41.208225	34	0	0	0	-1
2	0	0.000000	2320	0.000000e+00	0	0.000000	6	0	1	0	35040
3	6	0.000000	154	0.000000e+00	0	6493.506348	6	0	1	1	115
4	0	581.466064	9999417	2.856299e+06	11100000	1.068847	0	1	0	0	29200
...	...	...	...	...	...	...	...	...	...	...	...
291155	0	1133.389160	4965597	1.288043e+04	5029127	1.391857	0	1	0	0	29200
291156	0	1538.284790	5004324	1.602328e+03	5013310	1.196800	0	1	0	0	29200
291157	0	0.000000	18	0.000000e+00	0	55555.554688	0	0	1	1	330
291158	0	0.000000	13	0.000000e+00	0	76923.078125	0	0	1	1	339
291159	0	783.643921	5000020	2.000626e+06	1552397	0.762997	0	1	0	0	29200

291160 rows × 13 columns

In [41]:
```
df3=pd.read_csv("Friday-WorkingHours-Morning.pcap_ISCX.csv", usecols = cols)
df3
```

Out[41]:

	Bwd Packet Length Min	Bwd Packet Length Std	Flow IAT Max	Fwd IAT Std	Bwd IAT Total	Bwd Packets/s	Min Packet Length	PSH Flag Count	ACK Flag Count	URG Flag Count	Init_Win_bytes_forward	min_
0	72	0.0	16400000	6.848761e+06	113000000	0.141919	0	0	1	0	377	
1	316	0.0	16400000	6.848777e+06	113000000	0.141919	0	0	1	0	955	
2	0	0.0	20800000	1.395543e+06	0	0.000000	0	0	0	0	-1	
3	0	0.0	100055	2.183302e+04	0	0.000000	28	0	0	0	-1	
4	0	0.0	53431	3.046984e+04	0	0.000000	0	0	0	0	-1	
...	...	...	...	...	...	...	...	...	...	...	...	
191028	177	0.0	34304	1.566640e+04	4	32.545727	45	0	0	0	-1	
191029	136	0.0	163	0.000000e+00	4	11695.906430	40	0	0	0	-1	
191030	177	0.0	171	0.000000e+00	4	9009.009009	45	0	0	0	-1	
191031	48	0.0	16842	0.000000e+00	0	59.375371	48	0	0	0	-1	
191032	50	0.0	148	0.000000e+00	3	13071.895420	34	0	0	0	-1	

191033 rows × 13 columns

In [42]:
```
df5=pd.read_csv("Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv", usecols = cols)
df5
```

Out[42]:

	Bwd Packet Length Min	Bwd Packet Length Std	Flow IAT Max	Fwd IAT Std	Bwd IAT Total	Bwd Packets/s	Min Packet Length	PSH Flag Count	ACK Flag Count	URG Flag Count	Init_Win_bytes_forward	min_seg_
0	0	0.0	166	0.000000e+00	0	6024.096386	0	0	1	1	290	
1	0	0.0	49	0.000000e+00	49	24096.385540	0	0	1	1	243	
2	48	0.0	99947	0.000000e+00	0	10.005303	48	0	0	0	-1	
3	48	0.0	37017	0.000000e+00	0	27.014615	48	0	0	0	-1	
4	0	0.0	13600000	2.539814e+06	0	0.000000	0	0	0	0	-1	
...	...	...	...	...	...	...	...	...	...	...	...	
288597	0	0.0	590930	0.000000e+00	0	0.000000	0	0	1	1	238	
288598	0	0.0	1187988	0.000000e+00	0	0.000000	0	0	1	1	238	
288599	6	0.0	3	0.000000e+00	7	900000.000000	6	0	1	0	0	
288600	0	0.0	7	1.900292e+00	0	0.000000	237	0	0	0	-1	
288601	0	0.0	4751966	0.000000e+00	0	0.000000	0	0	1	1	238	

288602 rows × 13 columns

In [43]:
```
df6=pd.read_csv("Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv", usecols = cols)
df6
```

Out[43]:

	Bwd Packet Length Min	Bwd Packet Length Std	Flow IAT Max	Fwd IAT Std	Bwd IAT Total	Bwd Packets/s	Min Packet Length	PSH Flag Count	ACK Flag Count	URG Flag Count	Init_Win_bytes_forward	r
0	316	231.080951	16500000	5.491986e+06	113000000	0.212210	0	0	1	0	571	
1	126	208.261294	16500000	4.719143e+06	113000000	0.352505	0	0	1	0	390	
2	0	0.000000	60100000	5.277837e+06	0	0.000000	0	0	0	0	-1	
3	0	653.594166	60000000	2.120000e+07	211947	0.116160	0	1	0	0	8192	
4	161	0.000000	261	0.000000e+00	4	7434.944238	51	0	0	0	-1	
...	...	...	...	...	...	...	...	...	...	...	...	
170361	6	0.000000	45	0.000000e+00	4	61224.489800	6	0	0	0	0	
170362	6	0.000000	139	0.000000e+00	0	4608.294931	0	0	1	0	137	
170363	0	307.913979	955521	1.608760e+05	431936	33.152030	0	1	0	0	29200	
170364	0	0.000000	207	0.000000e+00	0	4830.917874	0	0	1	1	290	
170365	0	0.000000	49	0.000000e+00	1	40000.000000	0	0	1	1	243	

170366 rows × 13 columns

In [7]:
```
df = pd.concat([df1,df2])
del df1,df2
df = pd.concat([df,df3])
del df3
df = pd.concat([df,df5])
del df5
df = pd.concat([df,df6])
del df6

data = df.copy()

for column in data.columns:
    if data[column].dtype == np.int64:
        maxVal = data[column].max()
        if maxVal < 120:
            data[column] = data[column].astype(np.int8)
        elif maxVal < 32767:
            data[column] = data[column].astype(np.int16)
        else:
            data[column] = data[column].astype(np.int32)

    if data[column].dtype == np.float64:
        maxVal = data[column].max()
        minVal = data[data[column]>0][column]
        if maxVal < 120 and minVal>0.01 :
            data[column] = data[column].astype(np.float16)
        else:
            data[column] = data[column].astype(np.float32)



attackType = data[' Label'].unique()
data[' Label'] = data[' Label'].astype('category')
data[' Label'] = data[' Label'].astype("category").cat.codes
```

In [8]:
```
y = data[' Label'].copy()
X = data.drop([' Label'],axis=1)
```

In [10]:
```
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(sampling_strategy='majority')
X_rus, y_rus = rus.fit_resample(X, y)
```

In [11]:
```
y_rus.value_counts()
```

Out[11]:
```
4    158930
2    128027
1      1966
5      1507
7       652
3        36
0        21
6        21
Name:  Label, dtype: int64
```

In [12]:
```
df = X_rus
df[' Label'] = y_rus
minor = pd.DataFrame(df[(df[' Label']!=4) & (df[' Label']!=2)])
major = pd.DataFrame(df[(df[' Label']==4) | (df[' Label']==2)])
minor[' Label'].value_counts()
```

Out[12]:
```
1    1966
5    1507
7     652
3      36
0      21
6      21
Name:  Label, dtype: int64
```

In [13]:
```
from imblearn.over_sampling import SMOTE
y_rus_ =  minor[' Label']
X_rus_ =  minor.drop([' Label'],axis=1)
strategy = {1:2000, 5:1600, 7:800, 3:300, 6:200, 0:200}
sm = SMOTE(sampling_strategy=strategy)
X_sm, y_sm = sm.fit_resample(X_rus_, y_rus_)
X_min,y_min = X_sm, y_sm
```

```
C:\Users\AKHIL\AppData\Roaming\Python\Python39\site-packages\imblearn\utils\_validation.py:313: UserWarning:
After over-sampling, the number of samples (2000) in class 1 will be larger than the number of samples in th
e majority class (class #1 -> 1966)
  warnings.warn(
```

In [14]:
```
major[' Label'].value_counts()
```

Out[14]:
```
4    158930
2    128027
Name:  Label, dtype: int64
```

In [15]:
```
from imblearn.under_sampling import RandomUnderSampler
y_rus_ =  major[' Label']
X_rus_ =  major.drop([' Label'],axis=1)
strategy = {4:10000, 2:6000}
tom = RandomUnderSampler(sampling_strategy=strategy)
X_tom, y_tom = tom.fit_resample(X_rus_, y_rus_)
y_tom.value_counts()
```

Out[15]:
```
4    10000
2     6000
Name:  Label, dtype: int64
```

In [16]:
```
X_maj,y_maj = X_tom, y_tom
X,y = pd.concat([X_maj,X_min]), pd.concat([y_maj,y_min])
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21100 entries, 0 to 5099
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0    Bwd Packet Length Min  21100 non-null  int16
 1    Bwd Packet Length Std  21100 non-null  float32
 2    Flow IAT Max           21100 non-null  int32
 3    Fwd IAT Std            21100 non-null  float32
 4   Bwd IAT Total           21100 non-null  int32
 5    Bwd Packets/s          21100 non-null  float32
 6    Min Packet Length      21100 non-null  int16
 7    PSH Flag Count         21100 non-null  int8
 8    ACK Flag Count         21100 non-null  int8
 9    URG Flag Count         21100 non-null  int8
 10  Init_Win_bytes_forward  21100 non-null  int32
 11   min_seg_size_forward   21100 non-null  int8
dtypes: float32(3), int16(2), int32(3), int8(4)
memory usage: 824.2 KB
```

In [17]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# extract numerical attributes and scale it to have zero mean and unit variance
cols = X.select_dtypes(include=['float32','float16','int32','int16','int8']).columns
train_X = scaler.fit_transform(X.select_dtypes(include=['float32','float16','int32','int16','int8']))



from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(train_X,y,train_size=0.70, random_state=2)


from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression


RFC_Classifier = RandomForestClassifier(max_depth=40)
RFC_Classifier.fit(X_train, Y_train)
print ('RF Classifier run')

SVM_Classifier = SVC()
SVM_Classifier.fit(X_train, Y_train)
print ('SV Classifier run')


DTC_Classifier = tree.DecisionTreeClassifier(criterion='gini', max_depth=33, random_state=20, max_features=12,
DTC_Classifier.fit(X_train, Y_train)
print ('DTC Classifier run')

KNN_Classifier = KNeighborsClassifier(n_jobs=-1)
KNN_Classifier.fit(X_train, Y_train)
print ('KNN Classifier run')


LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0, max_iter=5000)
LGR_Classifier.fit(X_train, Y_train)
```

```
print ('LGR Classifier run')


BNB_Classifier = BernoulliNB()
BNB_Classifier.fit(X_train, Y_train)
print ('BNB Classifier run')
```

```
RF Classifier run
SV Classifier run
DTC Classifier run
KNN Classifier run
LGR Classifier run
BNB Classifier run
```

In [34]:

```
from sklearn import metrics
import seaborn as sns
import matplotlib.pyplot as plt
import time

models = []
models.append(('Random Forest Classifier', RFC_Classifier))
models.append(('Decision Tree Classifier', DTC_Classifier))
models.append(('Support Vector Classifier',SVM_Classifier))
models.append(('KNeighborsClassifier',KNN_Classifier))
models.append(('LogisticRegression',LGR_Classifier))
models.append(('Gaussian Naive Baye',BNB_Classifier))


for i, v in models:
    Xpred =  v.predict(X_train)
    scores = cross_val_score(v, X_train, Y_train, cv=10)
    accuracy = metrics.accuracy_score(Y_train, Xpred)
    confusion_matrix = metrics.confusion_matrix(Y_train, Xpred)
    cmd = metrics.ConfusionMatrixDisplay(confusion_matrix, display_labels=attackType)
    classification = metrics.classification_report(Y_train, Xpred)
    print()
    print('============================== {} Model Evaluation =============================='.format(i))
    print()
    print ("Cross Validation Mean Score:" "\n", scores.mean())
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
    cmd.plot(cmap='PuBuGn_r', include_values=True)
    time.sleep(1)
```

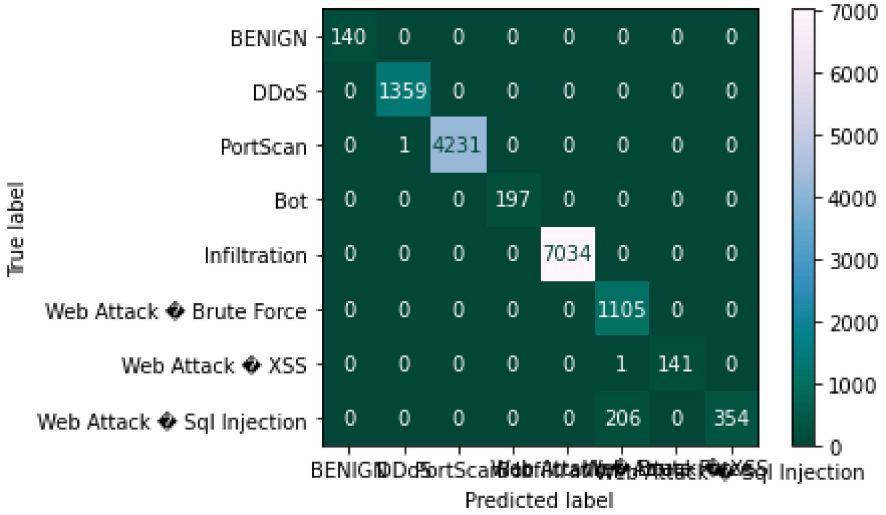
```
============================== Random Forest Classifier Model Evaluation ==============================

Cross Validation Mean Score:
 0.9612704651081717

Model Accuracy:
 0.985916446611145

Confusion matrix:
 [[ 140    0    0    0    0    0    0    0]
 [   0 1359    0    0    0    0    0    0]
 [   0    1 4231    0    0    0    0    0]
 [   0    0    0  197    0    0    0    0]
 [   0    0    0    0 7034    0    0    0]
 [   0    0    0    0    0 1105    0    0]
 [   0    0    0    0    0    1  141    0]
 [   0    0    0    0    0  206    0  354]]

Classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       140
           1       1.00      1.00      1.00      1359
           2       1.00      1.00      1.00      4232
           3       1.00      1.00      1.00       197
           4       1.00      1.00      1.00      7034
           5       0.84      1.00      0.91      1105
           6       1.00      0.99      1.00       142
           7       1.00      0.63      0.77       560

    accuracy                           0.99     14769
   macro avg       0.98      0.95      0.96     14769
weighted avg       0.99      0.99      0.98     14769



============================== Decision Tree Classifier Model Evaluation ==============================

Cross Validation Mean Score:
 0.9591714784784949

Model Accuracy:
```

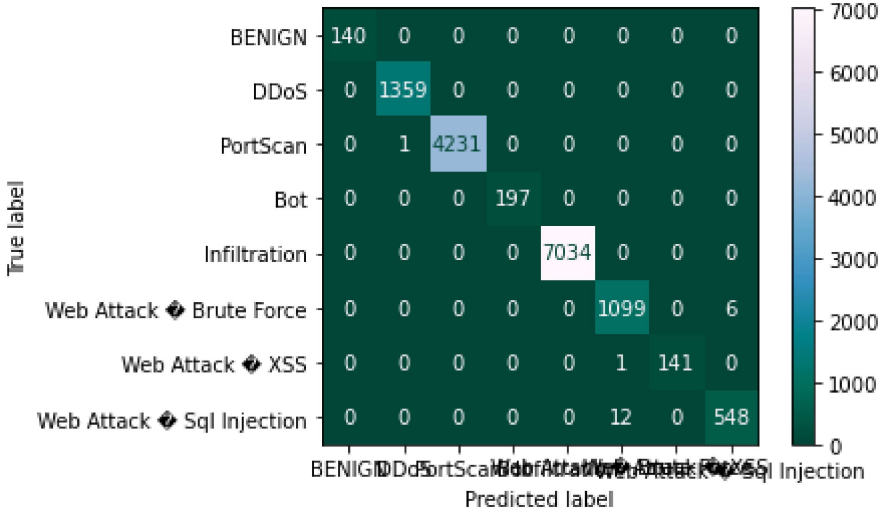
```
 0.9986458121741486

Confusion matrix:
 [[ 140    0    0    0    0    0    0    0]
 [   0 1359    0    0    0    0    0    0]
 [   0    1 4231    0    0    0    0    0]
 [   0    0    0  197    0    0    0    0]
 [   0    0    0    0 7034    0    0    0]
 [   0    0    0    0    0 1099    0    6]
 [   0    0    0    0    0    1  141    0]
 [   0    0    0    0    0   12    0  548]]

Classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       140
           1       1.00      1.00      1.00      1359
           2       1.00      1.00      1.00      4232
           3       1.00      1.00      1.00       197
           4       1.00      1.00      1.00      7034
           5       0.99      0.99      0.99      1105
           6       1.00      0.99      1.00       142
           7       0.99      0.98      0.98       560

    accuracy                           1.00     14769
   macro avg       1.00      1.00      1.00     14769
weighted avg       1.00      1.00      1.00     14769



============================== Support Vector Classifier Model Evaluation ==============================

Cross Validation Mean Score:
 0.9534837701119055

Model Accuracy:
 0.9549732547904395

Confusion matrix:
 [[ 120    3    5    2    4    4    2    0]
 [   0 1350    6    0    0    1    0    2]
 [   0    1 4231    0    0    0    0    0]
 [   0    1   16  180    0    0    0    0]
```

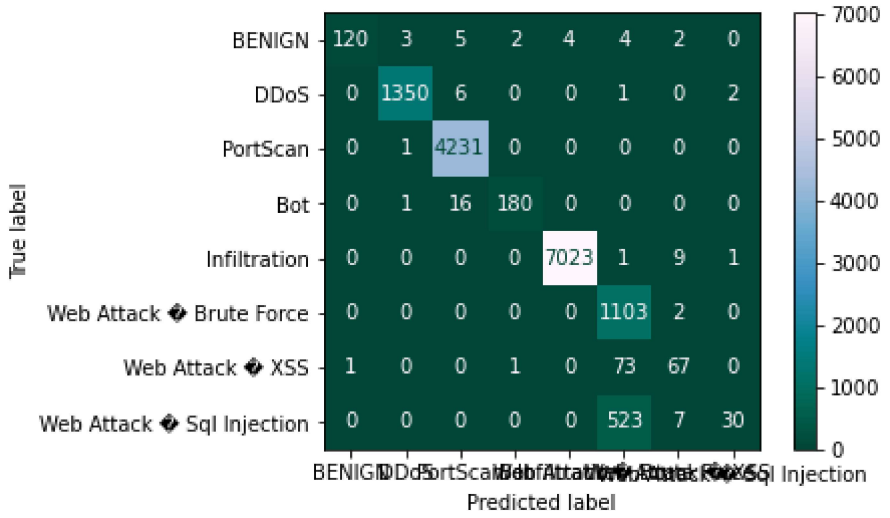
```
 [   0    0    0    0 7023    1    9    1]
 [   0    0    0    0    0 1103    2    0]
 [   1    0    0    1    0   73   67    0]
 [   0    0    0    0    0  523    7   30]]

Classification report:
               precision    recall  f1-score   support

           0       0.99      0.86      0.92       140
           1       1.00      0.99      0.99      1359
           2       0.99      1.00      1.00      4232
           3       0.98      0.91      0.95       197
           4       1.00      1.00      1.00      7034
           5       0.65      1.00      0.79      1105
           6       0.77      0.47      0.59       142
           7       0.91      0.05      0.10       560

    accuracy                           0.95     14769
   macro avg       0.91      0.79      0.79     14769
weighted avg       0.97      0.95      0.94     14769



============================== KNeighborsClassifier Model Evaluation ==============================

Cross Validation Mean Score:
 0.9532129967542058

Model Accuracy:
 0.9713589274832419

Confusion matrix:
 [[ 129    3    0    3    2    2    1    0]
 [   0 1354    0    0    2    0    1    2]
 [   2    3 4226    1    0    0    0    0]
 [   0    1    5  191    0    0    0    0]
 [   0    0    0    1 7031    1    0    1]
 [   0    0    0    0    2  947   16  140]
 [   1    0    0    1    0   16  124    0]
 [   0    0    0    0    1  208    7  344]]

Classification report:
               precision    recall  f1-score   support
```

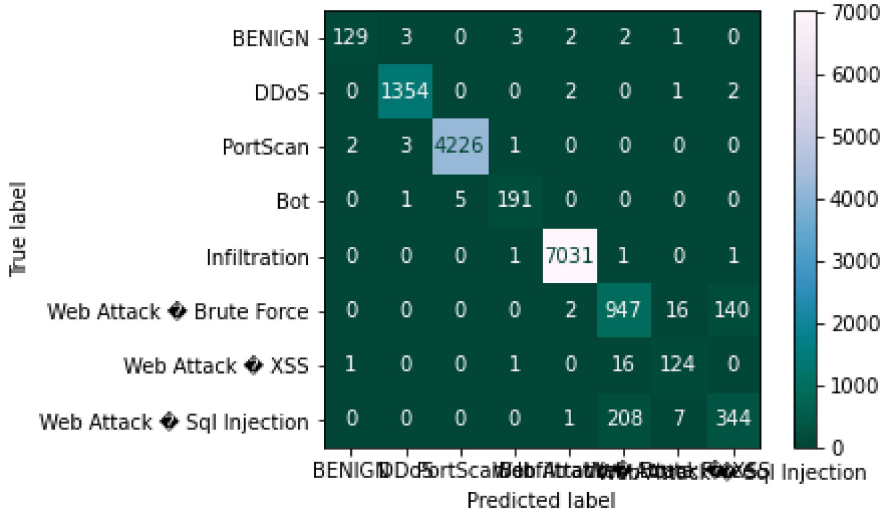
```
              0       0.98      0.92      0.95       140
              1       0.99      1.00      1.00      1359
              2       1.00      1.00      1.00      4232
              3       0.97      0.97      0.97       197
              4       1.00      1.00      1.00      7034
              5       0.81      0.86      0.83      1105
              6       0.83      0.87      0.85       142
              7       0.71      0.61      0.66       560

    accuracy                           0.97     14769
   macro avg       0.91      0.90      0.91     14769
weighted avg       0.97      0.97      0.97     14769



============================== LogisticRegression Model Evaluation ==============================

Cross Validation Mean Score:
 0.9509785546399809

Model Accuracy:
 0.9523325885300291

Confusion matrix:
 [[ 113    7    5    1    3    9    2    0]
 [   0 1347    7    0    2    2    1    0]
 [   0    3 4229    0    0    0    0    0]
 [   1    1   17  177    0    1    0    0]
 [   0    0    1    2 7016    7    7    1]
 [   0    0    0    0    1 1102    2    0]
 [   0    0    0    0    9   70   63    0]
 [   0    0    0    0    1  536    5   18]]

Classification report:
               precision    recall  f1-score   support

           0       0.99      0.81      0.89       140
           1       0.99      0.99      0.99      1359
           2       0.99      1.00      1.00      4232
           3       0.98      0.90      0.94       197
           4       1.00      1.00      1.00      7034
           5       0.64      1.00      0.78      1105
```

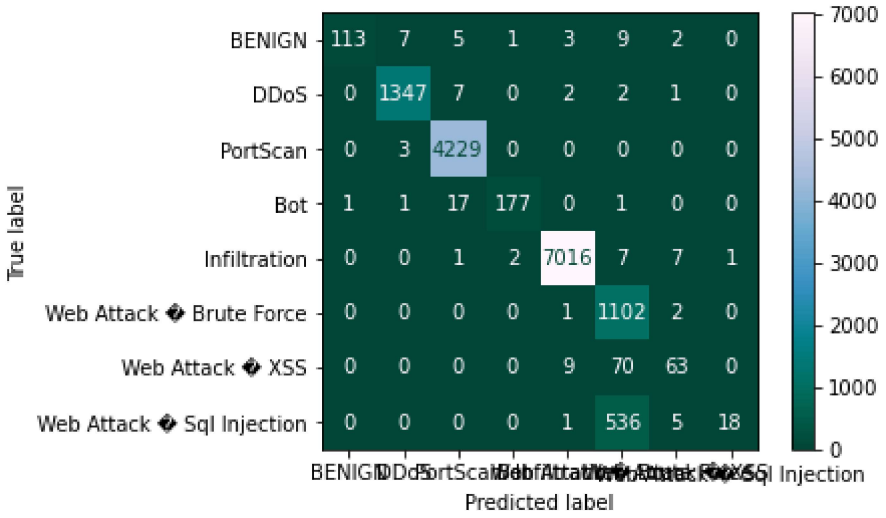
```
           6       0.79      0.44      0.57       142
           7       0.95      0.03      0.06       560

    accuracy                           0.95     14769
   macro avg       0.92      0.77      0.78     14769
weighted avg       0.96      0.95      0.94     14769



============================== Gaussian Naive Baye Model Evaluation ==============================

Cross Validation Mean Score:
 0.8741962118334792

Model Accuracy:
 0.8745344979348636

Confusion matrix:
 [[  55   54   17    0    0   12    2    0]
 [   0  486  518    0  353    2    0    0]
 [   9    5 4218    0    0    0    0    0]
 [   3    3   48  143    0    0    0    0]
 [  13    1   23    2 6964   26    5    0]
 [   0  109    0    0    0  994    2    0]
 [   0   70    0    0    0    9   56    7]
 [   0   22    0    0    0  532    6    0]]

Classification report:
               precision    recall  f1-score   support

           0       0.69      0.39      0.50       140
           1       0.65      0.36      0.46      1359
           2       0.87      1.00      0.93      4232
           3       0.99      0.73      0.84       197
           4       0.95      0.99      0.97      7034
           5       0.63      0.90      0.74      1105
           6       0.79      0.39      0.53       142
           7       0.00      0.00      0.00       560

    accuracy                           0.87     14769
   macro avg       0.70      0.59      0.62     14769
weighted avg       0.84      0.87      0.85     14769
```

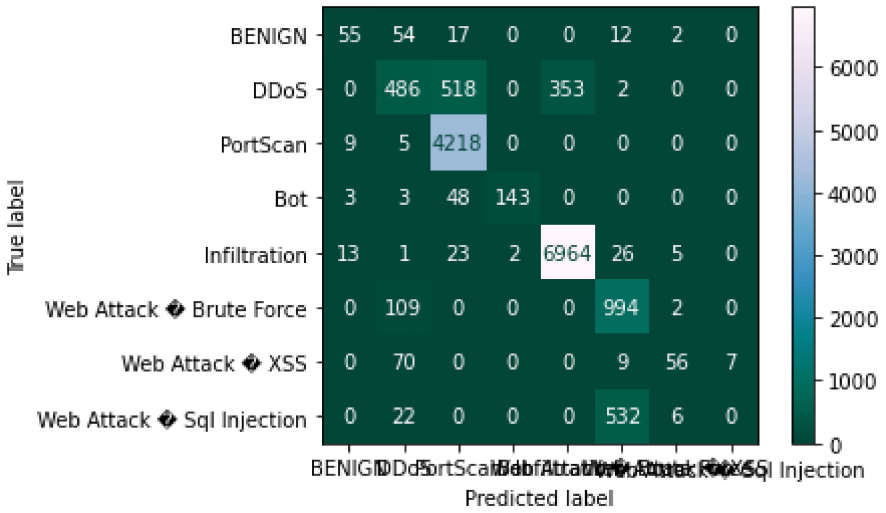

BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack • Brute Force
Web Attack • XSS
Web Attack • Sql Injection
True label
Predicted label
140 0 0 0 0 0 0 0
0 1359 0 0 0 0 0 0
0 1 4231 0 0 0 0 0
0 0 0 197 0 0 0 0
0 0 0 0 7034 0 0 0
0 0 0 0 0 1105 0 0
0 0 0 0 0 1 141 0
0 0 0 0 0 206 0 354
7000
6000
5000
4000
3000
2000
1000
0


BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack • Brute Force
Web Attack • XSS
Web Attack • Sql Injection
True label
Predicted label
140 0 0 0 0 0 0 0
0 1359 0 0 0 0 0 0
0 1 4231 0 0 0 0 0
0 0 0 197 0 0 0 0
0 0 0 0 7034 0 0 0
0 0 0 0 0 1099 0 6
0 0 0 0 0 1 141 0
0 0 0 0 0 12 0 548
7000
6000
5000
4000
3000
2000
1000
0


BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack
Web Attack
Web Attack
True label
Predicted label
7000
6000
5000
4000
3000
2000
1000
0


BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack
Web Attack
Web Attack
True label
Predicted label
7000
6000
5000
4000
3000
2000
1000
0


BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack
Brute Force
Web Attack
XSS
Web Attack
Sql Injection
True label
Predicted label
7000
6000
5000
4000
3000
2000
1000
0


BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack
Brute Force
Web Attack
XSS
Web Attack
Sql Injection
True label
Predicted label
6000
5000
4000
3000
2000
1000
0

In [30]:
```
for i, v in models:
    pred = v.predict(X_test)
    accuracy = metrics.accuracy_score(Y_test,pred)
    confusion_matrix = metrics.confusion_matrix(Y_test, pred)
    cmd = metrics.ConfusionMatrixDisplay(confusion_matrix, display_labels=attackType)
    classification = metrics.classification_report(Y_test, pred)
    print()
    print('============================== {} Model Test Results ==============================='.format(i))
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
    cmd.plot(cmap='cubehelix', include_values=True)
    time.sleep(1)
```

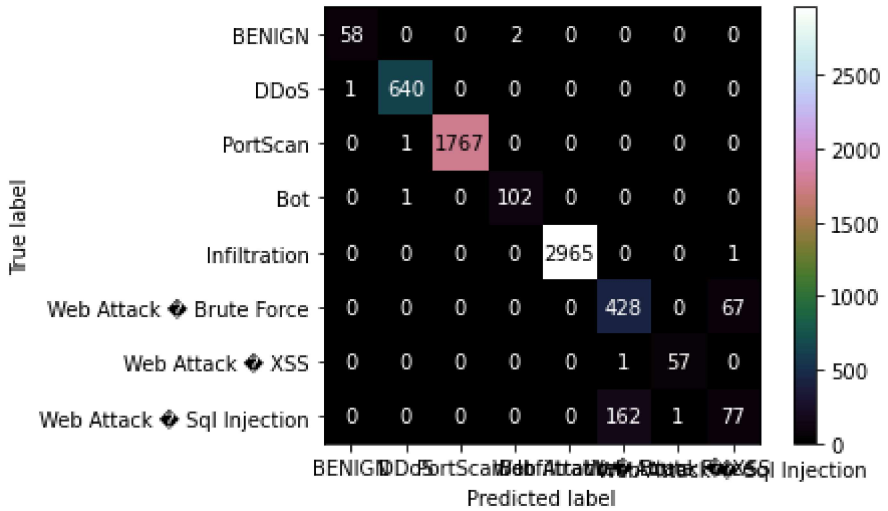
```
============================== Random Forest Classifier Model Test Results ==============================

Model Accuracy:
 0.9625651555836361

Confusion matrix:
 [[  58    0    0    2    0    0    0    0]
 [   1  640    0    0    0    0    0    0]
 [   0    1 1767    0    0    0    0    0]
 [   0    1    0  102    0    0    0    0]
 [   0    0    0    0 2965    0    0    1]
 [   0    0    0    0    0  428    0   67]
 [   0    0    0    0    0    1   57    0]
 [   0    0    0    0    0  162    1   77]]

Classification report:
               precision    recall  f1-score   support

           0       0.98      0.97      0.97        60
           1       1.00      1.00      1.00       641
           2       1.00      1.00      1.00      1768
           3       0.98      0.99      0.99       103
           4       1.00      1.00      1.00      2966
           5       0.72      0.86      0.79       495
           6       0.98      0.98      0.98        58
           7       0.53      0.32      0.40       240

    accuracy                           0.96      6331
   macro avg       0.90      0.89      0.89      6331
weighted avg       0.96      0.96      0.96      6331



============================== Decision Tree Classifier Model Test Results ==============================

Model Accuracy:
 0.9565629442426157

Confusion matrix:
 [[  56    1    0    1    2    0    0    0]
 [   2  638    1    0    0    0    0    0]
 [   1    1 1766    0    0    0    0    0]
```

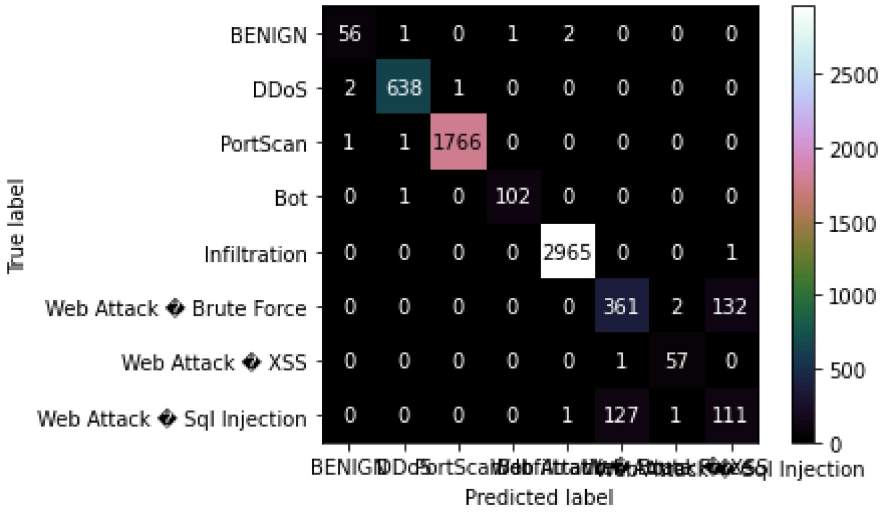
```
 [   0    1    0  102    0    0    0    0]
 [   0    0    0    0 2965    0    0    1]
 [   0    0    0    0    0  361    2  132]
 [   0    0    0    0    0    1   57    0]
 [   0    0    0    0    1  127    1  111]]

Classification report:
               precision    recall  f1-score   support

           0       0.95      0.93      0.94        60
           1       1.00      1.00      1.00       641
           2       1.00      1.00      1.00      1768
           3       0.99      0.99      0.99       103
           4       1.00      1.00      1.00      2966
           5       0.74      0.73      0.73       495
           6       0.95      0.98      0.97        58
           7       0.45      0.46      0.46       240

    accuracy                           0.96      6331
   macro avg       0.88      0.89      0.89      6331
weighted avg       0.96      0.96      0.96      6331



============================== Support Vector Classifier Model Test Results ==============================

Model Accuracy:
 0.9559311325225083

Confusion matrix:
 [[  50    0    1    5    1    2    1    0]
 [   0  639    1    0    0    1    0    0]
 [   0    1 1767    0    0    0    0    0]
 [   0    1    4   98    0    0    0    0]
 [   0    0    0    0 2963    1    1    1]
 [   0    0    0    0    0  493    2    0]
 [   0    0    0    1    0   29   28    0]
 [   0    0    0    0    0  223    3   14]]

Classification report:
               precision    recall  f1-score   support

           0       1.00      0.83      0.91        60
```

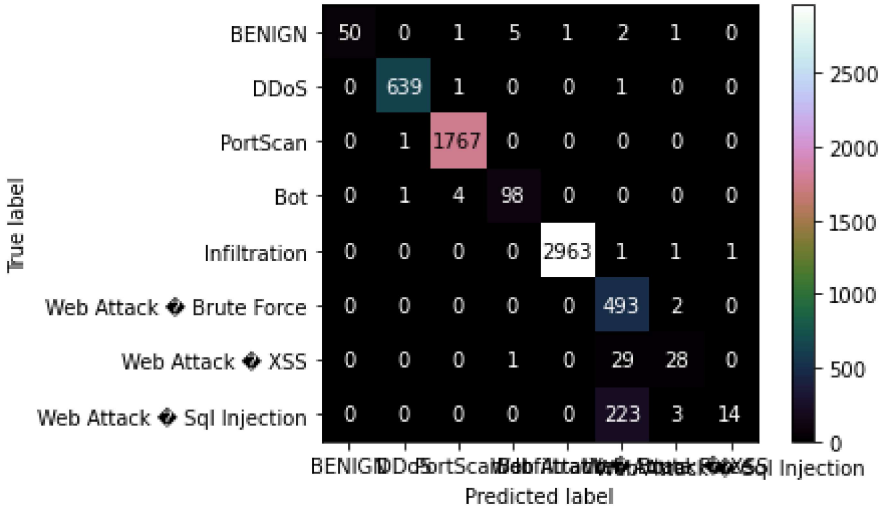
```
              1       1.00      1.00      1.00       641
              2       1.00      1.00      1.00      1768
              3       0.94      0.95      0.95       103
              4       1.00      1.00      1.00      2966
              5       0.66      1.00      0.79       495
              6       0.80      0.48      0.60        58
              7       0.93      0.06      0.11       240

    accuracy                           0.96      6331
   macro avg       0.92      0.79      0.79      6331
weighted avg       0.97      0.96      0.94      6331



============================== KNeighborsClassifier Model Test Results ==============================

Model Accuracy:
 0.9560890854525351

Confusion matrix:
 [[  53    1    0    5    0    1    0    0]
 [   0  638    0    0    2    0    1    0]
 [   0    2 1766    0    0    0    0    0]
 [   0    1    1  101    0    0    0    0]
 [   0    0    0    0 2963    1    1    1]
 [   0    0    0    0    0  380   13  102]
 [   0    0    0    1    0    4   53    0]
 [   0    0    0    0    1  137    3   99]]

Classification report:
               precision    recall  f1-score   support

           0       1.00      0.88      0.94        60
           1       0.99      1.00      0.99       641
           2       1.00      1.00      1.00      1768
           3       0.94      0.98      0.96       103
           4       1.00      1.00      1.00      2966
           5       0.73      0.77      0.75       495
           6       0.75      0.91      0.82        58
           7       0.49      0.41      0.45       240

    accuracy                           0.96      6331
   macro avg       0.86      0.87      0.86      6331
```

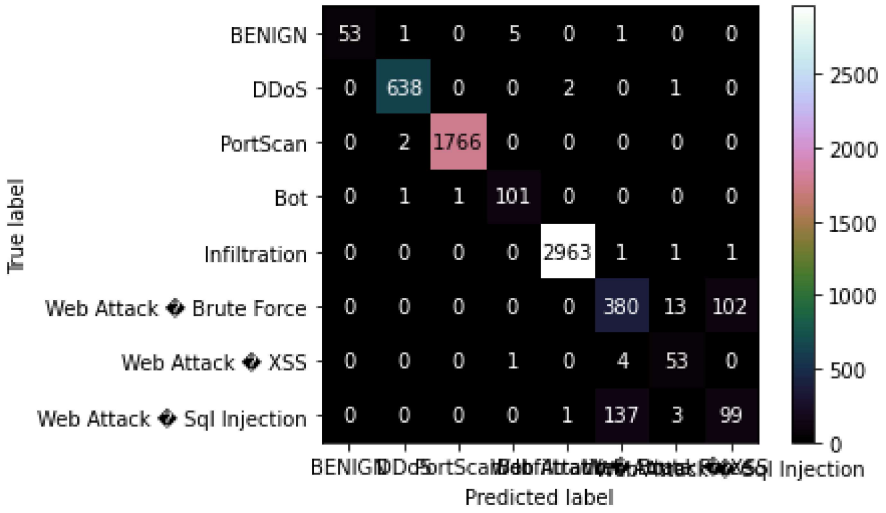
```
weighted avg       0.95      0.96      0.96      6331


============================== LogisticRegression Model Test Results ==============================

Model Accuracy:
 0.9535618385721055

Confusion matrix:
 [[  48    4    1    2    0    3    1    1]
 [   0  636    2    0    2    0    1    0]
 [   0    1 1767    0    0    0    0    0]
 [   0    1    4   98    0    0    0    0]
 [   0    0    0    2 2960    3    0    1]
 [   0    0    0    0    1  493    1    0]
 [   0    0    0    0    5   24   29    0]
 [   0    2    0    0    0  229    3    6]]

Classification report:
               precision    recall  f1-score   support

           0       1.00      0.80      0.89        60
           1       0.99      0.99      0.99       641
           2       1.00      1.00      1.00      1768
           3       0.96      0.95      0.96       103
           4       1.00      1.00      1.00      2966
           5       0.66      1.00      0.79       495
           6       0.83      0.50      0.62        58
           7       0.75      0.03      0.05       240

    accuracy                           0.95      6331
   macro avg       0.90      0.78      0.79      6331
weighted avg       0.96      0.95      0.94      6331


============================== Gaussian Naive Baye Model Test Results ==============================

Model Accuracy:
 0.8767967145790554

Confusion matrix:
```

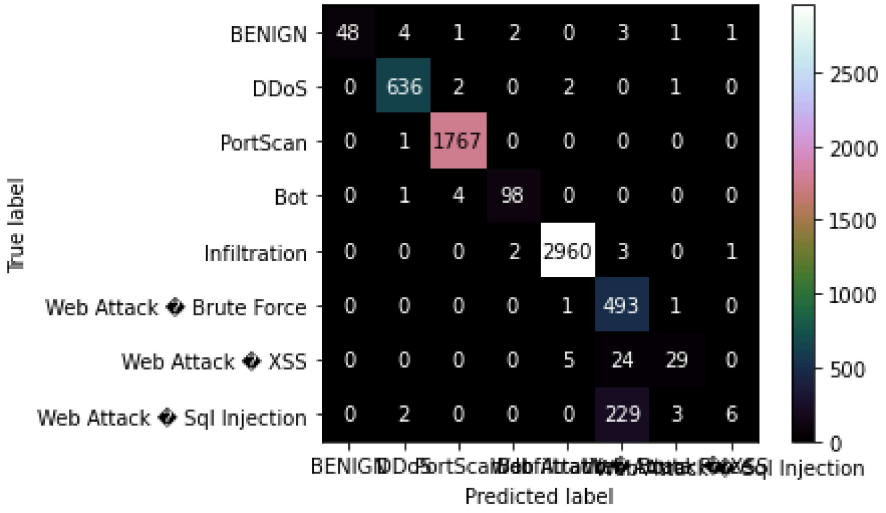
```
[[  22   23    9    1    0    4    1    0]
[   0  266  230    0  145    0    0    0]
[   2    1 1765    0    0    0    0    0]
[   2    3   15   82    1    0    0    0]
[   4    1    4    2 2943   12    0    0]
[   0   43    0    0    0  448    4    0]
[   0   24    0    0    0    6   25    3]
[   0    9    0    0    0  228    3    0]]

Classification report:
              precision    recall  f1-score   support

           0       0.73      0.37      0.49        60
           1       0.72      0.41      0.53       641
           2       0.87      1.00      0.93      1768
           3       0.96      0.80      0.87       103
           4       0.95      0.99      0.97      2966
           5       0.64      0.91      0.75       495
           6       0.76      0.43      0.55        58
           7       0.00      0.00      0.00       240

    accuracy                           0.88      6331
   macro avg       0.71      0.61      0.64      6331
weighted avg       0.84      0.88      0.85      6331
```

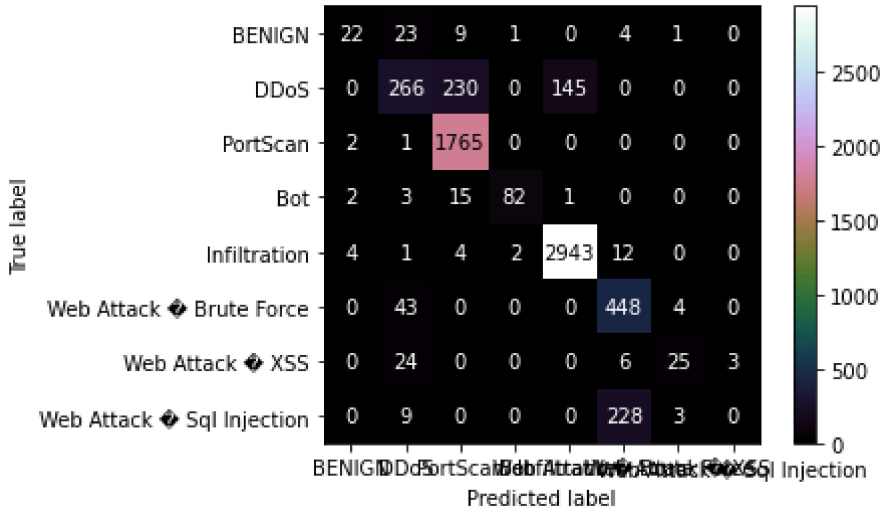



BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack • Brute Force
Web Attack • XSS
Web Attack • Sql Injection
True label
Predicted label
56 1 0 1 2 0 0 0
2 638 1 0 0 0 0 0
1 1 1766 0 0 0 0 0
0 1 0 102 0 0 0 0
0 0 0 0 2965 0 0 1
0 0 0 0 0 361 2 132
0 0 0 0 0 1 57 0
0 0 0 0 1 127 1 111
2500
2000
1500
1000
500
0


BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack • Brute Force
Web Attack • XSS
Web Attack • Sql Injection
True label
Predicted label
50 0 1 5 1 2 1 0
0 639 1 0 0 1 0 0
0 1 1767 0 0 0 0 0
0 1 4 98 0 0 0 0
0 0 0 0 2963 1 1 1
0 0 0 0 0 493 2 0
0 0 0 1 0 29 28 0
0 0 0 0 0 223 3 14
2500
2000
1500
1000
500
0


BENIGN 53 1 0 5 0 1 0 0
DDoS 0 638 0 0 2 0 1 0
PortScan 0 2 1766 0 0 0 0 0
Bot 0 1 1 101 0 0 0 0
Infiltration 0 0 0 0 2963 1 1 1
Web Attack Brute Force 0 0 0 0 0 380 13 102
Web Attack XSS 0 0 0 1 0 4 53 0
Web Attack Sql Injection 0 0 0 0 1 137 3 99
True label
Predicted label
2500
2000
1500
1000
500
0


BENIGN 48 4 1 2 0 3 1 1
DDoS 0 636 2 0 2 0 1 0
PortScan 0 1 1767 0 0 0 0 0
Bot 0 1 4 98 0 0 0 0
Infiltration 0 0 0 2 2960 3 0 1
Web Attack Brute Force 0 0 0 0 1 493 1 0
Web Attack XSS 0 0 0 0 5 24 29 0
Web Attack Sql Injection 0 2 0 0 0 229 3 6
True label
Predicted label
2500
2000
1500
1000
500
0


BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack
Web Attack
Web Attack
True label
22 23 9 1 0 4 1 0
0 266 230 0 145 0 0 0
2 1 1765 0 0 0 0 0
2 3 15 82 1 0 0 0
4 1 4 2 2943 12 0 0
0 43 0 0 0 448 4 0
0 24 0 0 0 6 25 3
0 9 0 0 0 228 3 0
2500
2000
1500
1000
500
0
BENIGN
Injection
Predicted label

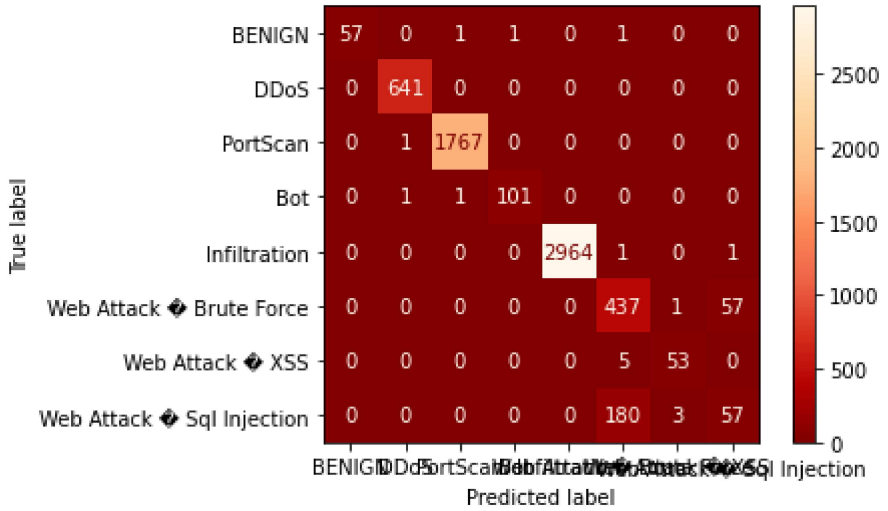
In [20]:
```
from sklearn.ensemble import VotingClassifier

clf1 = tree.DecisionTreeClassifier(criterion='gini', max_depth=33, random_state=20, max_features=12, splitter=
clf2 = RandomForestClassifier(criterion='gini', max_depth=40, random_state=20)
clf3 = SVC()
clf4 = KNeighborsClassifier(n_jobs=-1)
clf5 = LogisticRegression(n_jobs=-1, random_state=0, max_iter=5000)
clf6 = BernoulliNB()

votingC = VotingClassifier(estimators=[('dc',clf1), ('rf', clf2),('svc',clf3),('knn',clf4),('lgr',clf5),('bnb
votingC.fit(X_train,Y_train)
```

Out[20]:
```
VotingClassifier(estimators=[('dc',
                              DecisionTreeClassifier(max_depth=33,
                                                     max_features=12,
                                                     random_state=20,
                                                     splitter='random')),
                             ('rf',
                              RandomForestClassifier(max_depth=40,
                                                     random_state=20)),
                             ('svc', SVC()),
                             ('knn', KNeighborsClassifier(n_jobs=-1)),
                             ('lgr',
                              LogisticRegression(max_iter=5000, n_jobs=-1,
                                                 random_state=0)),
                             ('bnb', BernoulliNB())],
                 weights=[2, 2, 1, 2, 1, 1])
```

In [28]:

```
pred = votingC.predict(X_test)
accuracy = metrics.accuracy_score(Y_test, pred)
conf_mat = metrics.confusion_matrix(Y_test, pred)
confusion_matrix_display = metrics.ConfusionMatrixDisplay(conf_mat, display_labels=attackType)
classification = metrics.classification_report(Y_test, pred)

print('============================== {} Model Test Results ==============================')
print()
print('Model Accuracy:\n', accuracy)
print()
print('Confusion matrix:\n', conf_mat)
print()
print('Classification report:\n', classification)
print()
confusion_matrix_display.plot(cmap='OrRd_r', include_values=True)
time.sleep(1)
```

```
============================== {} Model Test Results ==============================

Model Accuracy:
 0.9598799557731796

Confusion matrix:
 [[  57    0    1    1    0    1    0    0]
 [   0  641    0    0    0    0    0    0]
 [   0    1 1767    0    0    0    0    0]
 [   0    1    1  101    0    0    0    0]
 [   0    0    0    0 2964    1    0    1]
 [   0    0    0    0    0  437    1   57]
 [   0    0    0    0    0    5   53    0]
 [   0    0    0    0    0  180    3   57]]

Classification report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        60
           1       1.00      1.00      1.00       641
           2       1.00      1.00      1.00      1768
           3       0.99      0.98      0.99       103
           4       1.00      1.00      1.00      2966
           5       0.70      0.88      0.78       495
           6       0.93      0.91      0.92        58
           7       0.50      0.24      0.32       240

    accuracy                           0.96      6331
   macro avg       0.89      0.87      0.87      6331
weighted avg       0.96      0.96      0.96      6331
```


BENIGN
DDoS
PortScan
Bot
Infiltration
Web Attack
Web Attack
True label
Predicted label
2500
2000
1500
1000
500
0

In [ ]: