In [1]:
```
import numpy as np
import pandas as pd
train = pd.read_csv('UNSW_NB15_freq_enc_training_set.csv')
test = pd.read_csv('UNSW_NB15_freq_enc_testing_set.csv')
```

In [2]:
```
x1 = train.iloc[:,1:43]
y1 = train['label']
print(x1)
print(y1)
```

```
             dur  spkts  dpkts  sbytes  dbytes           rate  sttl  dttl  \
0       0.121478      6      4     258     172      74.087490   252   254
1       0.649902     14     38     734   42014      78.473372    62   252
2       1.623129      8     16     364   13186      14.170161    62   252
3       1.681642     12     12     628     770      13.677108    62   252
4       0.449454     10      6     534     268      33.373826   254   252
...          ...    ...    ...     ...     ...            ...   ...   ...
175336  0.000009      2      0     114       0  111111.107200   254     0
175337  0.505762     10      8     620     354      33.612649   254   252
175338  0.000009      2      0     114       0  111111.107200   254     0
175339  0.000009      2      0     114       0  111111.107200   254     0
175340  0.000009      2      0     114       0  111111.107200   254     0

              sload          dload  ...  ct_dst_src_ltm  is_ftp_login  \
0       1.415894e+04    8495.365234  ...               1             0
1       8.395112e+03  503571.312500  ...               2             0
2       1.572272e+03   60929.230470  ...               3             0
3       2.740179e+03    3358.622070  ...               3             1
4       8.561499e+03    3987.059814  ...              40             0
...              ...            ...  ...             ...           ...
175336  5.066666e+07       0.000000  ...              24             0
175337  8.826286e+03    4903.492188  ...               2             0
175338  5.066666e+07       0.000000  ...              13             0
175339  5.066666e+07       0.000000  ...              30             0
175340  5.066666e+07       0.000000  ...              30             0

        ct_ftp_cmd  ct_flw_http_mthd  ct_src_ltm  ct_srv_dst  is_sm_ips_ports  \
0                0                 0           1           1                0
1                0                 0           1           6                0
2                0                 0           2           6                0
3                1                 0           2           1                0
4                0                 0           2          39                0
...            ...               ...         ...         ...              ...
175336           0                 0          24          24                0
175337           0                 0           1           1                0
175338           0                 0           3          12                0
175339           0                 0          30          30                0
175340           0                 0          30          30                0

        proto_freq_encode  service_freq_encode  state_freq_encode
0                0.477508             0.548451           0.454700
1                0.477508             0.548451           0.454700
2                0.477508             0.548451           0.454700
3                0.477508             0.019327           0.454700
4                0.477508             0.548451           0.454700
...                   ...                  ...                ...
175336           0.359762             0.266466           0.451883
175337           0.477508             0.548451           0.454700
175338           0.359762             0.266466           0.451883
175339           0.359762             0.266466           0.451883
175340           0.359762             0.266466           0.451883

[175341 rows x 42 columns]
0         0
1         0
2         0
3         0
4         0
         ..
175336    1
175337    1
175338    1
175339    1
175340    1
Name: label, Length: 175341, dtype: int64
```

In [3]:
```
x2 = test.iloc[:,1:43]
y2 = test['label']
print(x2)
print(y2)
```

```
            dur  spkts  dpkts  sbytes  dbytes           rate  sttl  dttl  \
0      0.000011      2      0     496       0   90909.090200   254     0
1      0.000008      2      0    1762       0  125000.000300   254     0
2      0.000005      2      0    1068       0  200000.005100   254     0
3      0.000006      2      0     900       0  166666.660800   254     0
4      0.000010      2      0    2126       0  100000.002500   254     0
...         ...    ...    ...     ...     ...            ...   ...   ...
82327  0.000005      2      0     104       0  200000.005100   254     0
82328  1.106101     20      8   18062     354      24.410067   254   252
82329  0.000000      1      0      46       0       0.000000     0     0
82330  0.000000      1      0      46       0       0.000000     0     0
82331  0.000009      2      0     104       0  111111.107200   254     0

              sload        dload  ...  ct_dst_src_ltm  is_ftp_login  \
0      1.803636e+08     0.000000  ...               2             0
1      8.810000e+08     0.000000  ...               2             0
2      8.544000e+08     0.000000  ...               3             0
3      6.000000e+08     0.000000  ...               3             0
4      8.504000e+08     0.000000  ...               3             0
...             ...          ...  ...             ...           ...
82327  8.320000e+07     0.000000  ...               2             0
82328  1.241044e+05  2242.109863  ...               1             0
82329  0.000000e+00     0.000000  ...               1             0
82330  0.000000e+00     0.000000  ...               1             0
82331  4.622222e+07     0.000000  ...               1             0

       ct_ftp_cmd  ct_flw_http_mthd  ct_src_ltm  ct_srv_dst  is_sm_ips_ports  \
0               0                 0           1           2                0
1               0                 0           1           2                0
2               0                 0           1           3                0
3               0                 0           2           3                0
4               0                 0           2           3                0
...           ...               ...         ...         ...              ...
82327           0                 0           2           1                0
82328           0                 0           3           2                0
82329           0                 0           1           1                1
82330           0                 0           1           1                1
82331           0                 0           1           1                0

       proto_freq_encode  service_freq_encode  state_freq_encode
0               0.359762             0.548451           0.451883
1               0.359762             0.548451           0.451883
2               0.359762             0.548451           0.451883
3               0.359762             0.548451           0.451883
4               0.359762             0.548451           0.451883
...                  ...                  ...                ...
82327           0.359762             0.548451           0.451883
82328           0.477508             0.548451           0.454700
82329           0.014926             0.548451           0.451883
82330           0.014926             0.548451           0.451883
82331           0.359762             0.548451           0.451883

[82332 rows x 42 columns]
0        0
1        0
2        0
3        0
4        0
        ..
82327    0
82328    0
82329    0
82330    0
82331    0
Name: label, Length: 82332, dtype: int64
```

In [4]:
```
from sklearn.preprocessing import MinMaxScaler
model = MinMaxScaler()
model.fit(x1)
x1 = model.transform(x1)
x2 = model.transform(x2)
```

In [5]:
```
from xgboost import XGBClassifier
import xgboost as xgb
params = {
            'objective':'binary:logistic',
            'max_depth': 4, 'min_child_weight': 12, 'gamma': 0.3, 'subsample': 0.6,
            'colsample_bytree': 0.6, 'scale_pos_weight': 1,
            'alpha': 0.05,
            'learning_rate': 0.03,
            'n_estimators':1484,'seed': 27
         }
xgb_clf = XGBClassifier(**params)
xgb_clf.fit(x1, y1)
```

```
C:\Users\admin\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in X
GBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following:
1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) a
s integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[14:22:16] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'er
ror' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[5]:
```
XGBClassifier(alpha=0.05, base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.6,
              enable_categorical=False, gamma=0.3, gpu_id=-1,
              importance_type=None, interaction_constraints='',
              learning_rate=0.03, max_delta_step=0, max_depth=4,
              min_child_weight=12, missing=nan, monotone_constraints='()',
              n_estimators=1484, n_jobs=4, num_parallel_tree=1,
              predictor='auto', random_state=27, reg_alpha=0.0500000007,
              reg_lambda=1, scale_pos_weight=1, seed=27, subsample=0.6,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [7]:
```
y_pred=xgb_clf.predict(x2)
print(y_pred)
```

```
[1 1 1 ... 0 0 1]
```

In [8]:
```
from sklearn import metrics
from sklearn.metrics import f1_score

print('Accuracy = ', metrics.accuracy_score(y2, y_pred)*100)
print("Confusion Matrix =", metrics.confusion_matrix(y2, y_pred, labels=None,
                                                     sample_weight=None))
print("Recall =", metrics.recall_score(y2, y_pred, labels=None,
                                       pos_label=1, average='weighted',
                                       sample_weight=None))
print("Precision =", metrics.precision_score(y2, y_pred, labels=None,
                                       pos_label=1, average='weighted',
                                       sample_weight=None))
print("Classification Report =\n", metrics.classification_report(y2, y_pred,
                                                                 labels=None,
                                                                 target_names=None,
                                                                 sample_weight=None,
                                                                 digits=2,
                                                                 output_dict=False))
print("F1 Score = ",f1_score(y2, y_pred, average='macro'))
```

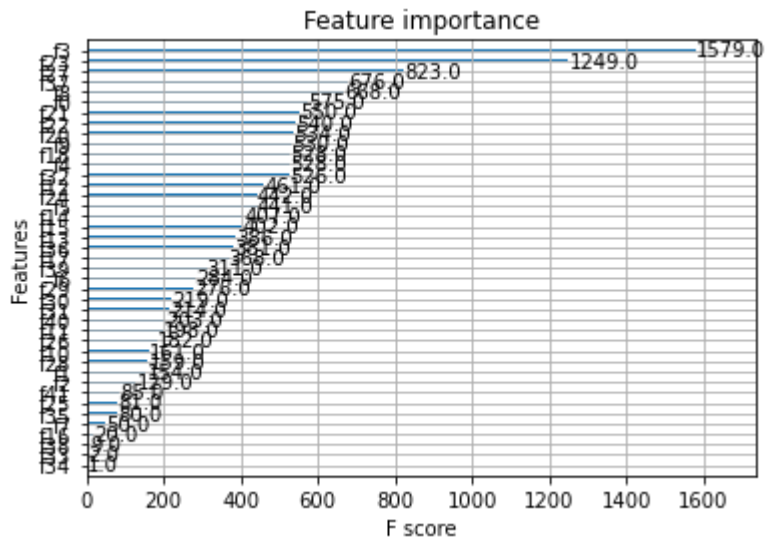
```
Accuracy =  87.71923431958413
Confusion Matrix = [[27610  9390]
 [  721 44611]]
Recall = 0.8771923431958413
Precision = 0.8928216992317198
Classification Report =
              precision    recall  f1-score   support

           0       0.97      0.75      0.85     37000
           1       0.83      0.98      0.90     45332

    accuracy                           0.88     82332
   macro avg       0.90      0.87      0.87     82332
weighted avg       0.89      0.88      0.87     82332

F1 Score =  0.8717226681965957
```

In [9]:
```
import matplotlib.pyplot as plt
xgb.plot_importance(xgb_clf)
plt.rcParams['figure.figsize'] = [15,15]
plt.show()
```




In [10]:
```
pd.Series(xgb_clf.get_booster().get_fscore()).sort_values(ascending=False)
```

Out[10]:
```
f3     1579.0
f23    1249.0
f27     823.0
f37     676.0
f8      668.0
f0      575.0
f21     550.0
f22     540.0
f20     534.0
f9      530.0
f18     528.0
f4      528.0
f32     526.0
f12     461.0
f24     442.0
f5      441.0
f14     407.0
f15     402.0
f13     386.0
f36     381.0
f17     368.0
f39     311.0
f6      284.0
f29     276.0
f30     219.0
f31     214.0
f40     203.0
f11     198.0
f26     182.0
f10     161.0
f28     159.0
f1      154.0
f2      129.0
f41      85.0
f25      81.0
f35      80.0
f7       50.0
f16      20.0
f38       9.0
f33       2.0
f34       1.0
dtype: float64
```

In [11]:
```
from numpy import sort
from sklearn.feature_selection import SelectFromModel
thresholds = sort(xgb_clf.feature_importances_)
print(thresholds)
```

```
[0.         0.00054432 0.00075167 0.00088144 0.00100133 0.00104625
 0.00116126 0.00125949 0.00148376 0.00150489 0.00155229 0.00164402
 0.00196953 0.00204442 0.00211289 0.00354206 0.00424938 0.00485486
 0.00526185 0.00535822 0.00559029 0.00635514 0.00639051 0.00651507
 0.00656803 0.00674823 0.00699365 0.00813449 0.00829311 0.00854332
 0.00965884 0.00986754 0.01239873 0.01510002 0.01607109 0.01839451
 0.03082572 0.03776417 0.05745872 0.13671382 0.13973343 0.40365767]
```

In [12]:
```
n_min = 42
acc_max = 0
thresholds = sort(xgb_clf.feature_importances_)
obj_thresh = thresholds[0]
for thresh in thresholds:
    selection = SelectFromModel(xgb_clf, threshold=thresh, prefit=True)
    select_X_train = selection.transform(x1)
    selection_model = XGBClassifier(**params)
    selection_model.fit(select_X_train, y1)
    select_X_test = selection.transform(x2)
    predictions = selection_model.predict(select_X_test)
    accuracy = metrics.accuracy_score(y2, predictions)
    print("Thresh=%.3f, n=%d, Accuracy: %.2f%%" % (obj_thresh, select_X_train.shape[1], accuracy*100.0))
    if(select_X_train.shape[1] < n_min) and (accuracy > acc_max):
        n_min = select_X_train.shape[1]
        acc_max = accuracy
        obj_thresh = thresh

selection = SelectFromModel(xgb_clf, threshold=obj_thresh, prefit=True)
select_X_train = selection.transform(x1)
selection_model = XGBClassifier(**params)
selection_model.fit(select_X_train, y1)
select_X_test = selection.transform(x2)
predictions = selection_model.predict(select_X_test)
accuracy = metrics.accuracy_score(y2, predictions)
print("Thresh=%.3f, n=%d, Accuracy: %.2f%%" % (obj_thresh, select_X_train.shape[1], accuracy*100.0))
```

```
C:\Users\admin\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the followin
g: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels
(y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


Thresh=0.005, n=1, Accuracy: 76.63%

C:\Users\admin\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the followin
g: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels
(y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[17:41:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starti
ng in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed fro
m 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Thresh=0.005, n=23, Accuracy: 88.05%
```

In [13]:
```
print("Performance report with XGBoost feature selection and classification is: -")
print('Accuracy = ', metrics.accuracy_score(y2, predictions)*100)
print("Confusion Matrix =", metrics.confusion_matrix(y2, predictions, labels=None,
                                                     sample_weight=None))
print("Recall =", metrics.recall_score(y2, predictions, labels=None,
                                       pos_label=1, average='weighted',
                                       sample_weight=None))
print("Precision =", metrics.precision_score(y2, predictions, labels=None,
                                       pos_label=1, average='weighted',
                                       sample_weight=None))
print("Classification Report =\n", metrics.classification_report(y2, y_pred,
                                                                 labels=None,
                                                                 target_names=None,
                                                                 sample_weight=None,
                                                                 digits=2,
                                                                 output_dict=False))
print("F1 Score = ",f1_score(y2, y_pred, average='macro'))
```

```
Performance report with XGBoost feature selection and classification is: -
Accuracy =  88.05203323130739
Confusion Matrix = [[27866  9134]
 [  703 44629]]
Recall = 0.8805203323130739
Precision = 0.8953980491251079
Classification Report =
               precision    recall  f1-score   support

           0       0.97      0.75      0.85     37000
           1       0.83      0.98      0.90     45332

    accuracy                           0.88     82332
   macro avg       0.90      0.87      0.87     82332
weighted avg       0.89      0.88      0.87     82332

F1 Score =  0.8717226681965957
```

In [ ]: