In [1]:
```python
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score
```

In [2]:
```python
data = pd.read_csv(r"malware_MultiClass.csv")
data
```

Out[2]:

	hash	millisecond	classification	os	state	usage_counter	
0	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	0	malware	CentOS	0	0	30
1	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	1	malware	Windows	0	0	30
2	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	2	malware	Mac	0	0	30
3	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	3	malware	Ubuntu	0	0	30
4	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	4	malware	Mac	0	0	30
...	...	...	...	...	...	...	
99995	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	995	unknown	CentOS	4096	0	30
99996	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	996	unknown	Windows	4096	0	30
99997	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	997	unknown	CentOS	4096	0	30
99998	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	998	unknown	Ubuntu	4096	0	30
99999	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	999	unknown	Mac	4096	0	30

100000 rows × 36 columns

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 36 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   hash               100000 non-null  object
 1   millisecond        100000 non-null  int64 
 2   classification     100000 non-null  object
 3   os                 100000 non-null  object
 4   state              100000 non-null  int64 
 5   usage_counter      100000 non-null  int64 
 6   prio               100000 non-null  int64 
 7   static_prio        100000 non-null  int64 
 8   normal_prio        100000 non-null  int64 
 9   policy             100000 non-null  int64 
 10  vm_pgoff           100000 non-null  int64 
 11  vm_truncate_count  100000 non-null  int64 
 12  task_size          100000 non-null  int64 
 13  cached_hole_size   100000 non-null  int64 
 14  free_area_cache    100000 non-null  int64 
 15  mm_users           100000 non-null  int64 
 16  map_count          100000 non-null  int64 
 17  hiwater_rss        100000 non-null  int64 
 18  total_vm           100000 non-null  int64 
 19  shared_vm          100000 non-null  int64 
 20  exec_vm            100000 non-null  int64 
 21  reserved_vm        100000 non-null  int64 
 22  nr_ptes            100000 non-null  int64 
 23  end_data           100000 non-null  int64 
 24  last_interval      100000 non-null  int64 
 25  nvcsw              100000 non-null  int64 
 26  nivcsw             100000 non-null  int64 
 27  min_flt            100000 non-null  int64 
 28  maj_flt            100000 non-null  int64 
 29  fs_excl_counter    100000 non-null  int64 
 30  lock               100000 non-null  int64 
 31  utime              100000 non-null  int64 
 32  stime              100000 non-null  int64 
 33  gtime              100000 non-null  int64 
 34  cgtime             100000 non-null  int64 
 35  signal_nvcsw       100000 non-null  int64 
dtypes: int64(33), object(3)
memory usage: 27.5+ MB
```

In [4]:
```python
# Drop the 'hash' column as it's not useful for modeling
data = data.drop(columns=['hash'])

# Encode categorical columns
categorical_cols = ['classification', 'os']
le = LabelEncoder()
for col in categorical_cols:
    data[col] = le.fit_transform(data[col])

# Split features and target
X = data.drop(columns=['classification'])
y = data['classification']

# Initialize the decision tree classifier (you can replace it with any other classifier)
clf = DecisionTreeClassifier()

# Perform 10-fold cross-validation
scores = cross_val_score(clf, X, y, cv=10, scoring='accuracy')

# Print the cross-validation results
print("Accuracy scores for each fold:", scores)
print("Mean accuracy:", scores.mean())

# Optionally, you can fit the model on the entire dataset if needed
clf.fit(X, y)
```

```
Accuracy scores for each fold: [0.6816 0.8703 0.9899 0.8676 0.8869 0.8391 0.833  0.876  0.9941
 0.8504]
Mean accuracy: 0.86889
```

Out[4]:
```
DecisionTreeClassifier()
```

In [5]:
```python
# Initialize a list to store accuracy scores for different models
accuracy_scores = []

# Initialize a variable to keep track of the best AUC score
best_accuracy = 0
best_model = None
```

In [6]:
```python
# Try different parameters for the Decision Tree model
for max_depth in [None, 10, 20, 30]:
    for min_samples_split in [2, 5, 10]:
        for min_samples_leaf in [1, 2, 4]:
            # Create the Decision Tree model with the current parameters
            clf = DecisionTreeClassifier(
                max_depth=max_depth,
                min_samples_split=min_samples_split,
                min_samples_leaf=min_samples_leaf,
                random_state=42  # Set a random state for reproducibility
            )

            # Perform 10-fold cross-validation and calculate the accuracy score
            accuracy_scores_cv = cross_val_score(clf, X, y, cv=10, scoring='accuracy')
            mean_accuracy = np.mean(accuracy_scores_cv)

            # Store the accuracy score and model if it's the best so far
            accuracy_scores.append(mean_accuracy)
            if mean_accuracy > best_accuracy:
                best_accuracy = mean_accuracy
                best_model = clf
```

In [7]:
```python
# Print the accuracy scores for different models
for i, accuracy_score in enumerate(accuracy_scores):
    print(f"Model {i+1}: Accuracy = {accuracy_score:.4f}")
```

```
Model 1: Accuracy = 0.8636
Model 2: Accuracy = 0.8690
Model 3: Accuracy = 0.8450
Model 4: Accuracy = 0.8599
Model 5: Accuracy = 0.8598
Model 6: Accuracy = 0.8450
Model 7: Accuracy = 0.8599
Model 8: Accuracy = 0.8598
Model 9: Accuracy = 0.8488
Model 10: Accuracy = 0.8669
Model 11: Accuracy = 0.8665
Model 12: Accuracy = 0.8602
Model 13: Accuracy = 0.8573
Model 14: Accuracy = 0.8573
Model 15: Accuracy = 0.8602
Model 16: Accuracy = 0.8573
Model 17: Accuracy = 0.8573
Model 18: Accuracy = 0.8602
Model 19: Accuracy = 0.8636
Model 20: Accuracy = 0.8690
Model 21: Accuracy = 0.8450
Model 22: Accuracy = 0.8599
Model 23: Accuracy = 0.8598
Model 24: Accuracy = 0.8450
Model 25: Accuracy = 0.8599
Model 26: Accuracy = 0.8598
Model 27: Accuracy = 0.8488
Model 28: Accuracy = 0.8636
Model 29: Accuracy = 0.8690
Model 30: Accuracy = 0.8450
Model 31: Accuracy = 0.8599
Model 32: Accuracy = 0.8598
Model 33: Accuracy = 0.8450
Model 34: Accuracy = 0.8599
Model 35: Accuracy = 0.8598
Model 36: Accuracy = 0.8488
```

In [8]:
```python
# Print the best model's parameters and accuracy score
print("\nBest Model Parameters:")
print(best_model.get_params())
print(f"Best Model Accuracy: {best_accuracy:.4f}")
```

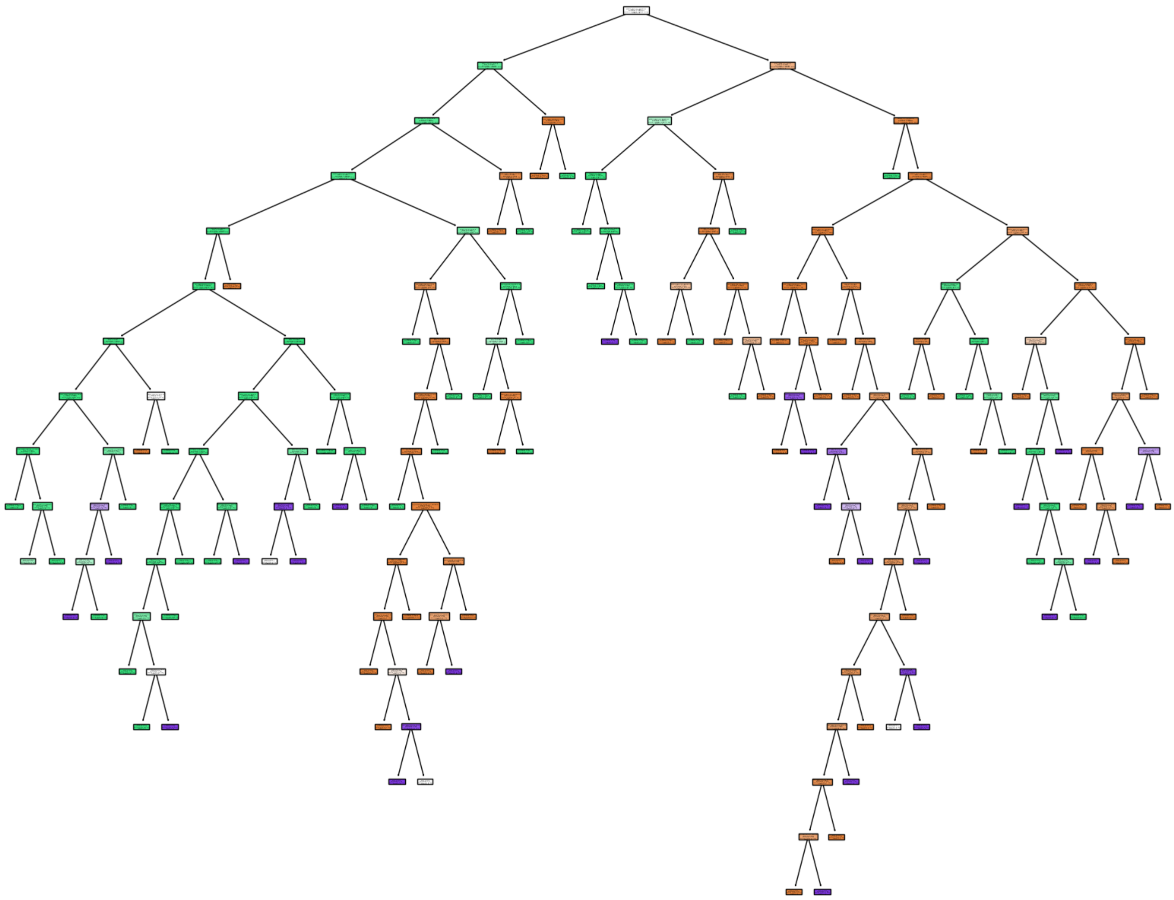
```
Best Model Parameters:
{'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features':
None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 2, 'min_samples_
split': 2, 'min_weight_fraction_leaf': 0.0, 'random_state': 42, 'splitter': 'best'}
Best Model Accuracy: 0.8690
```

In [9]:
```python
# Convert class labels to strings
class_names = data['classification'].unique().astype(str)

# Fit the best model on the entire dataset
best_model.fit(X, y)

# Visualize the best decision tree model
plt.figure(figsize=(20, 16))
plot_tree(best_model, filled=True, feature_names=X.columns, class_names=class_names)
plt.title("Best Decision Tree Model")
plt.show()
```

Best Decision Tree Model



In [ ]: