Here is a comprehensive description of feed-forward neural networks, along with a practical cybersecurity example and Python code with sample data:

Feed-Forward Neural Networks (FNNs)

A feed-forward neural network (FNN) is a type of artificial neural network where connections between nodes do not form a cycle. Information moves in only one direction – forward – from the input nodes through the hidden layers to the output nodes.

FNNs consist of the following components:

Input Layer: Receives the initial data/inputs.
Hidden Layers: Perform computations and transfer information from the input to the output layer. There can be one or more hidden layers.
Output Layer: Provides the final output/prediction of the network.

Each node in a layer is connected to every node in the next layer, and connections have numeric weights that encode the strength/importance. During training, these weights are optimized to minimize the error between predicted and true outputs.

Common activation functions like ReLU, sigmoid, etc. introduce non-linearity, allowing FNNs to model complex relationships. Backpropagation is the core algorithm for training by calculating gradients and updating weights.

Cybersecurity Application: Malware Detection

One application of FNNs in cybersecurity is malware detection. Given features extracted from files (e.g. byte sequences, header info, etc.), an FNN can learn patterns to classify whether a file is malicious or benign.

Sample Data:

Assuming we have a dataset with the following features for each file:

File size
Entropy
Number of strings
Number of imported DLL functions

We can generate a small random sample dataset:

```
import numpy as np

# Sample data - 10 examples with 4 features each
data = np.random.randint(1, 101, size=(10, 4))
labels = np.random.randint(0, 2, size=(10, 1))  # 0 = benign, 1 = malicious
```

```
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Generate toy data
data = np.random.randint(1, 101, size=(1000, 4))
labels = np.random.randint(0, 2, size=(1000, 1))
```

```
# Split into train/test
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2)

# Build FNN model
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(4,)))  # Input layer
model.add(Dense(16, activation='relu'))  # Hidden layer
model.add(Dense(1, activation='sigmoid'))  # Output layer

# Compile and train
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))
```

This builds a basic FNN with 1 input layer, 1 hidden layer, and 1 output layer to predict if a file is malicious based on the 4 input features. The model is trained on the generated toy data.