Using Monte Carlo Simulation for Cyber Security Risk Assessment

```python
import numpy as np
import matplotlib.pyplot as plt

# List of risks
risks = [
    "Unauthorised access", "Elevation of privilege attack", "Insider threat", "Phishing attack",
    "DDoS", "Password attack", "Man-in-the-middle attack", "Web application and in-app attacks",
    "Advanced persistent threat attack", "Privacy compromise", "DC Disaster",
    "Cloud vendor disaster", "Legal issues", "Third-party failure attack", "Accidental data breach",
    "Sensitive data disclosure", "Sales losses"]

# Parameters for frequency and severity distributions
binomial_params = (40, 0.002)  # Number of trials and probability of success
poisson_params = (12,)  # Mean (average) occurrence rate
interuniform_params = (10, 30, 20)  # Lower bound, upper bound, and mode

# Number of iterations for Monte Carlo simulation
num_iterations = 1000

# Lists to store total losses for each risk
total_losses = []

# Simulate risk events
for risk in risks:
    if risk in ["Unauthorised access", "Password attack", "Advanced persistent threat attack",
                "DC Disaster", "Sales losses"]:
        # Binomial distribution
        frequency = np.random.binomial(binomial_params[0], binomial_params[1], num_iterations)
    elif risk in ["Elevation of privilege attack", "Phishing attack", "Web application",
                  "Insider threat", "DDoS",
                  "Accidental data breach", "Man-in-the-middle attack", "Privacy compromise",
                  "Cloud vendor disaster"]:
        # Poisson distribution
        frequency = np.random.poisson(poisson_params[0], num_iterations)
    else:
        # Interuniform distribution
        frequency = np.random.triangular(interuniform_params[0], interuniform_params[2],
                                         interuniform_params[1], num_iterations)
```

```python
    # Severity distribution
    severity = np.random.normal(1000, 5000, num_iterations)

    # Calculate total loss for each iteration
    total_loss = frequency * severity
    total_losses.append(total_loss)

# Calculate mean total loss
mean_total_losses = np.mean(total_losses, axis=1)

# Plot distribution graph
plt.hist(mean_total_losses, bins=50, color='c')
plt.xlabel('Total Loss Expectancy')
plt.ylabel('Frequency')
plt.title('Monte Carlo Simulation: Distribution of Total Loss Expectancy')
plt.show()
```