In [1]:
```
# Write sample log data to a file
sample_log_content = """192.168.1.1 - - [03/Dec/2024:10:12:34 +0000] "GET /h
203.0.113.5 - - [03/Dec/2024:10:12:35 +0000] "POST /login HTTP/1.1" 401 128
10.0.0.2 - - [03/Dec/2024:10:12:36 +0000] "GET /about HTTP/1.1" 200 256
192.168.1.1 - - [03/Dec/2024:10:12:37 +0000] "GET /contact HTTP/1.1" 200 312
198.51.100.23 - - [03/Dec/2024:10:12:38 +0000] "POST /register HTTP/1.1" 200
203.0.113.5 - - [03/Dec/2024:10:12:39 +0000] "POST /login HTTP/1.1" 401 128
192.168.1.100 - - [03/Dec/2024:10:12:40 +0000] "POST /login HTTP/1.1" 401 12
10.0.0.2 - - [03/Dec/2024:10:12:41 +0000] "GET /dashboard HTTP/1.1" 200 1024
198.51.100.23 - - [03/Dec/2024:10:12:42 +0000] "GET /about HTTP/1.1" 200 256
192.168.1.1 - - [03/Dec/2024:10:12:43 +0000] "GET /dashboard HTTP/1.1" 200 1
203.0.113.5 - - [03/Dec/2024:10:12:44 +0000] "POST /login HTTP/1.1" 401 128
203.0.113.5 - - [03/Dec/2024:10:12:45 +0000] "POST /login HTTP/1.1" 401 128
192.168.1.100 - - [03/Dec/2024:10:12:46 +0000] "POST /login HTTP/1.1" 401 12
10.0.0.2 - - [03/Dec/2024:10:12:47 +0000] "GET /profile HTTP/1.1" 200 768
192.168.1.1 - - [03/Dec/2024:10:12:48 +0000] "GET /home HTTP/1.1" 200 512
198.51.100.23 - - [03/Dec/2024:10:12:49 +0000] "POST /feedback HTTP/1.1" 200
203.0.113.5 - - [03/Dec/2024:10:12:50 +0000] "POST /login HTTP/1.1" 401 128
192.168.1.1 - - [03/Dec/2024:10:12:51 +0000] "GET /home HTTP/1.1" 200 512
198.51.100.23 - - [03/Dec/2024:10:12:52 +0000] "GET /about HTTP/1.1" 200 256
203.0.113.5 - - [03/Dec/2024:10:12:53 +0000] "POST /login HTTP/1.1" 401 128
192.168.1.100 - - [03/Dec/2024:10:12:54 +0000] "POST /login HTTP/1.1" 401 12
10.0.0.2 - - [03/Dec/2024:10:12:55 +0000] "GET /contact HTTP/1.1" 200 512
198.51.100.23 - - [03/Dec/2024:10:12:56 +0000] "GET /home HTTP/1.1" 200 512
192.168.1.100 - - [03/Dec/2024:10:12:57 +0000] "POST /login HTTP/1.1" 401 12
203.0.113.5 - - [03/Dec/2024:10:12:58 +0000] "POST /login HTTP/1.1" 401 128
10.0.0.2 - - [03/Dec/2024:10:12:59 +0000] "GET /dashboard HTTP/1.1" 200 1024
192.168.1.1 - - [03/Dec/2024:10:13:00 +0000] "GET /about HTTP/1.1" 200 256
198.51.100.23 - - [03/Dec/2024:10:13:01 +0000] "POST /register HTTP/1.1" 200
203.0.113.5 - - [03/Dec/2024:10:13:02 +0000] "POST /login HTTP/1.1" 401 128
192.168.1.100 - - [03/Dec/2024:10:13:03 +0000] "POST /login HTTP/1.1" 401 12
10.0.0.2 - - [03/Dec/2024:10:13:04 +0000] "GET /profile HTTP/1.1" 200 768
198.51.100.23 - - [03/Dec/2024:10:13:05 +0000] "GET /about HTTP/1.1" 200 256
192.168.1.1 - - [03/Dec/2024:10:13:06 +0000] "GET /home HTTP/1.1" 200 512
198.51.100.23 - - [03/Dec/2024:10:13:07 +0000] "POST /feedback HTTP/1.1" 200

# Save to file
with open('sample.log', 'w') as file:
    file.write(sample_log_content)

print("sample.log file created successfully!")
```

```
sample.log file created successfully!
```

In [2]:
```
# Import required libraries
from collections import Counter
import csv
```

In [3]:
```python
# Function to read the log file
def read_log_file(file_path):
    with open(file_path, 'r') as file:
        lines = file.readlines()
    return lines

# Load the sample log file
log_lines = read_log_file('sample.log')
print("Log file loaded successfully!")
```

```
Log file loaded successfully!
```

In [4]:
```python
# Function to count requests per IP address
def count_requests_per_ip(lines):
    ip_counts = Counter()
    for line in lines:
        ip = line.split()[0]
        ip_counts[ip] += 1
    return ip_counts

# Count requests
ip_request_counts = count_requests_per_ip(log_lines)
print("Requests per IP Address:")
print(ip_request_counts)
```

```
Requests per IP Address:
Counter({'203.0.113.5': 8, '198.51.100.23': 8, '192.168.1.1': 7, '10.0.0.
2': 6, '192.168.1.100': 5})
```

In [5]:
```python
# Function to find the most frequently accessed endpoint
def find_most_accessed_endpoint(lines):
    endpoint_counts = Counter()
    for line in lines:
        endpoint = line.split('"')[1].split()[1]
        endpoint_counts[endpoint] += 1
    most_accessed = endpoint_counts.most_common(1)[0]
    return most_accessed

# Identify most accessed endpoint
most_frequent_endpoint = find_most_accessed_endpoint(log_lines)
print("Most Frequently Accessed Endpoint:", most_frequent_endpoint)
```

```
Most Frequently Accessed Endpoint: ('/login', 13)
```

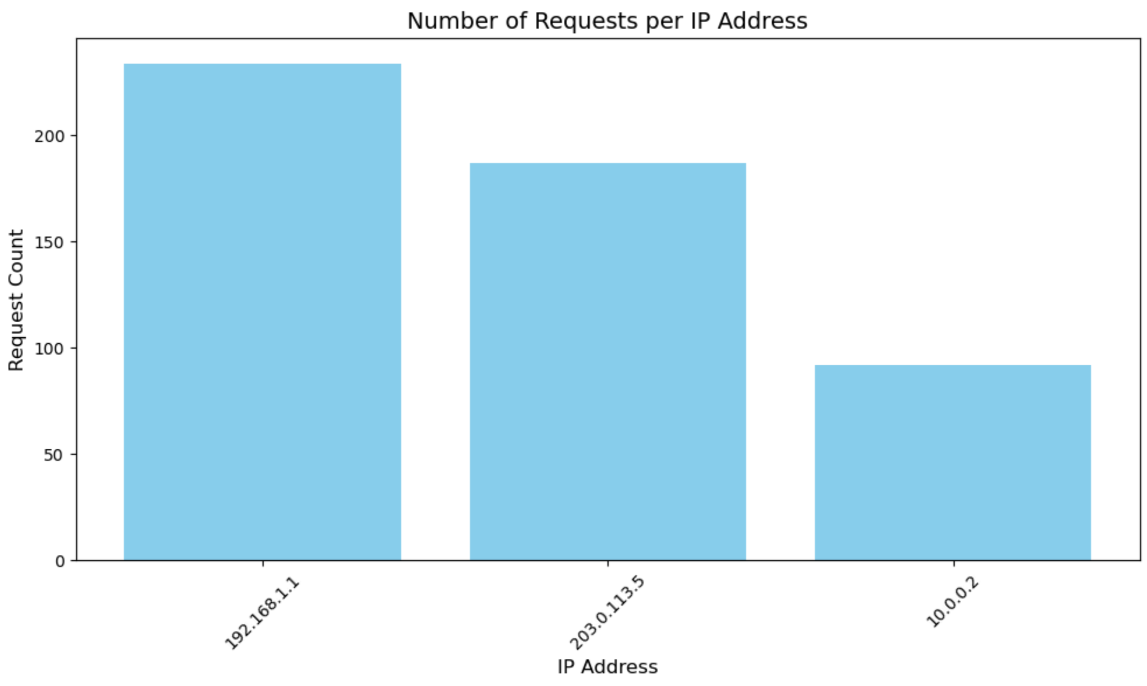
In [6]:
```python
# Function to detect suspicious activity
def detect_suspicious_activity(lines, threshold=10):
    failed_attempts = Counter()
    for line in lines:
        if '401' in line:  # Check for failed login status
            ip = line.split()[0]
            failed_attempts[ip] += 1
    suspicious_ips = {ip: count for ip, count in failed_attempts.items() if
    return suspicious_ips

# Detect suspicious activity
suspicious_activity = detect_suspicious_activity(log_lines)
print("Suspicious Activity Detected:")
print(suspicious_activity)
```

```
Suspicious Activity Detected:
{}
```

In [7]:
```python
# Function to save results to a CSV file
def save_to_csv(ip_counts, most_accessed, suspicious_ips, output_file='log_a
    with open(output_file, mode='w', newline='') as file:
        writer = csv.writer(file)
        # Write headers and data for Requests per IP
        writer.writerow(["IP Address", "Request Count"])
        for ip, count in ip_counts.items():
            writer.writerow([ip, count])
        writer.writerow([])  # Add a blank row
        # Write data for Most Accessed Endpoint
        writer.writerow(["Most Frequently Accessed Endpoint", "Access Count"
        writer.writerow([most_accessed[0], most_accessed[1]])
        writer.writerow([])  # Add a blank row
        # Write data for Suspicious Activity
        writer.writerow(["Suspicious Activity Detected"])
        writer.writerow(["IP Address", "Failed Login Count"])
        for ip, count in suspicious_ips.items():
            writer.writerow([ip, count])

# Save results to CSV
save_to_csv(ip_request_counts, most_frequent_endpoint, suspicious_activity)
print("Results saved to log_analysis_results.csv")
```

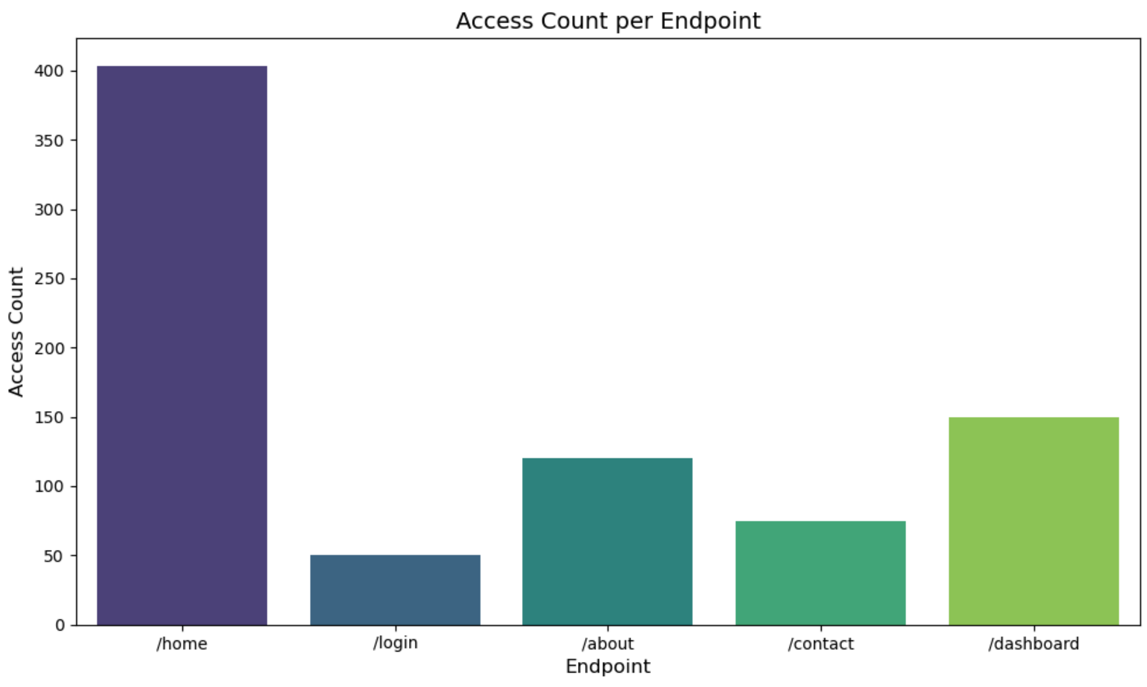
```
Results saved to log_analysis_results.csv
```

In [8]:
```python
#Visualizing IP Requests using matplotlib
import matplotlib.pyplot as plt

# Sample data (replace this with actual data from log analysis)
ip_addresses = ['192.168.1.1', '203.0.113.5', '10.0.0.2']
request_counts = [234, 187, 92]

# Create a bar chart for IP requests
plt.figure(figsize=(10, 6))
plt.bar(ip_addresses, request_counts, color='skyblue')
plt.title('Number of Requests per IP Address', fontsize=14)
plt.xlabel('IP Address', fontsize=12)
plt.ylabel('Request Count', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [9]:
```python
#Visualizing Endpoint Access Counts using seaborn
import seaborn as sns
import pandas as pd

# Sample data (replace this with actual data from log analysis)
data = {
    'endpoint': ['/home', '/login', '/about', '/contact', '/dashboard'],
    'access_count': [403, 50, 120, 75, 150]
}

df = pd.DataFrame(data)

# Create a bar plot for endpoint access counts
plt.figure(figsize=(10, 6))
sns.barplot(x='endpoint', y='access_count', data=df, palette='viridis')
plt.title('Access Count per Endpoint', fontsize=14)
plt.xlabel('Endpoint', fontsize=12)
plt.ylabel('Access Count', fontsize=12)
plt.tight_layout()
plt.show()
```

```
C:\Users\User\anaconda3\lib\site-packages\pandas\core\arrays\masked.py:60:
UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (ver
sion '1.3.5' currently installed).
  from pandas.core import (
```



In [10]:
```python
import ipywidgets as widgets
from IPython.display import display

# Create a slider widget for threshold of failed login attempts
threshold_slider = widgets.IntSlider(value=10, min=1, max=50, step=1, descri

# Function to simulate suspicious activity detection based on threshold
def detect_suspicious_activity(threshold):
    # Sample log data (replace with actual data)
    failed_logins = {
        '192.168.1.100': 56,
        '203.0.113.34': 12,
        '198.51.100.23': 4,
        '192.168.1.1': 5
    }

    # Find suspicious IPs
    suspicious_ips = {ip: count for ip, count in failed_logins.items() if co

    # Display suspicious IPs
    if suspicious_ips:
        print("Suspicious Activity Detected:")
        for ip, count in suspicious_ips.items():
            print(f"IP Address: {ip}, Failed Logins: {count}")
    else:
        print("No suspicious activity detected.")

# Link the slider to the function
widgets.interactive(detect_suspicious_activity, threshold=threshold_slider)

# Display the widget
display(threshold_slider)
```

Threshold: 10