

## **Anomali: Elektrikli Araç Şarj İstasyonu Kontrol Ünitesinde Bellek Sızıntısı (Memory Leak) Özet:**

Bu senaryoda, bir elektrikli araç şarj istasyonunun ana kontrol yazılımı, uzun süre kesintisiz çalıştığında, her şarj oturumu sonrası temizlemesi gereken geçici verileri (session logs, anlık akım grafikleri) RAM'de tutmaya devam ederek bellek sızıntısına (memory leak) yol açar.

Bu hata, istasyon yazılımında şarj oturumu nesnelerinin yaşam döngüsünün yanlış yönetilmesinden veya merkeze gönderilen verilerin yerel hafızadan silinmemesinden kaynaklanır.

### **1. Başlangıç: Normal Sistem Davranışı**

Normal koşullarda şarj istasyonu her işlem döngüsünde:

Yeni bir araç bağlandığında şarj oturumunu başlatır ve anlık voltaj/akım verilerini okur.

Veriyi işler, ekrana yansıtır ve kısa süreli bellekte (RAM) tutar.

Şarj tamamlandıktan sonra veri merkeze gönderildikten sonra bu geçici oturum nesnelerini serbest bırakır (örneğin `del current_session_data`).

İstasyon kontrolcüsünün RAM kullanımı zaman içinde küçük dalgalanmalar gösterir, ancak ortalama sabit kalır.

### **2. Anomali Oluşumu:**

Zaman içinde gözlenen anormal durum şudur: İstasyonun

RAM kullanımı sürekli artar.

Şarj hizmeti yoğunluğu artmasa bile cihaz daha fazla kaynak tüketir.

Birkaç gün/hafta sonra istasyon ekranı donar veya yeni gelen araçlara "Hizmet Dışı" hatası verir (sistem yanıt veremez hale gelir).

### Olası Teknik Nedenler (Şarj İstasyonu Bağlamında):

Neden	Amaç / Örnek Durum
Şarj oturum kayıtlarının (logs) temizlenmemesi	completed_sessions.append(data) kullanılıyor ama gönderilen veriler listeden asla silinmiyor.
Kapalı ağ soketlerinin unutulması	Merkez sunucu ile iletişim için socket.connect() yapılıyor ama bağlantı kopunca close() çağırılmıyor.
Arayüz (UI) nesnelerinin birikmesi	Her yeni araç için ekranda yeni bir grafik nesnesi (GraphWidget) oluşturuluyor ama eskiler yok edilmiyor.

### 3. Algılama Mantığı:

Sistem kontrolcüsü her 60 saniyede bir kendi bellek kullanımını ölçer (psutil vb. ile).

Aşağıdaki koşul gerçekleştiğinde anomali olarak loglanır: if

current\_memory > previous\_memory \* 1.20:

log("MEMORY LEAK DETECTED - CHARGING UNIT CRITICAL")

**Tespit Kriterleri:** ○ RAM kullanımı 5 ölçüm boyunca sürekli artış

gösteriyorsa, ○

Ortalama artış oranı %10'un üzerindeyse, ○ İşlemci (CPU) kullanımı sabitken RAM yükseliyorsa → "Memory Leak" alarmı oluşturulur.

### 4. Karar ve Tepki Mekanizması:

Anomali tespit edildiğinde istasyon:

Mevcut bellek kullanımını, aktif şarj durumunu ve çalışma süresini loglar. Eğer aktif bir şarj işlemi yoksa, istasyon yazılımını kontrollü şekilde yeniden başlatır (reboot).

Merkezi yönetim sistemine "Bakım Gerekli - Bellek Hatası" uyarısı gönderir.

## 5. Log Örneği (Uyarılanmış):

2025-11-10T20:45:19 | StationID=TR-38-A01 | MemUsage=850MB | Prev=708MB | Growth=+20.0% |  
ActiveSession=None | Event=MEMORY\_LEAK\_WARNING [cite: 35, 36]  
2025-11-10T20:50:21 | StationID=TR-38-A01 | MemUsage=995MB | Growth=+17.0% |  
Event=MEMORY\_LEAK\_CONFIRMED | Action=RebootStationSoft [cite: 37, 38]

## 6. Kaynaklara Dayalı Açıklama:

Bellek sızıntısı, Python veya Java gibi otomatik hafıza yönetimi olan dillerde yazılmış istasyon yazılımlarında bile görülebilir.

Özellikle 7/24 çalışan bu tip gömülü sistemlerde, küçük bir sızıntı bile zamanla cihazı kilitler.

## 7. Öneriler:

**Bellek Profileleme:** İstasyon yazılımının test aşamasında memory\_profiler ile uzun süreli yük testleri yapılmalı.

**Kaynak Yönetimi:** Her donanım erişimi (kart okuyucu, sayaç) with blokları veya güvenli try-finally yapıları ile kapatılmalı.

**Otomatik Bakım:** İstasyon, kullanımın olmadığı gece saatlerinde (örneğin 03:0004:00 arası) kendini otomatik olarak yeniden başlatacak şekilde programlanmalı.

## 8. Kaynaklar:

- Python Memory Management Documentation (Python.org, 2024)
- RedHat Developer – Tracing Memory Leaks in Python Services (2023)
- IEEE Access – Reliability Metrics for Continuous Monitoring Systems (2024)

