

Yazar: Sena Ateş

Tarih: 02.11.2025

OCPP Protokolünde Fuzzing Tabanlı Zafiyet Tespiti: Protokol Sağlamlık Analizi

1. Giriş ve Problem Tanımı

Konu: Elektrikli Araç (EV) şarj altyapısının temel iletişim protokolü olan Open Charge Point Protocol (OCPP) implementasyonlarında protokol sağlamlık (robustness) ve güvenlik zafiyetlerinin tespiti.

Problem Tanımı: EV şarj istasyonları (EVSE) ile Merkezi Şarj Yönetim Sistemleri (CSMS) arasındaki iletişimi standartlaştıran OCPP, enerji dağıtımının ve finansal işlemlerin kritik bir parçasıdır. Protokolün düzgün çalışması hayati önem taşırken, implementasyon hataları veya zayıf hata işleme mekanizmaları, ciddi güvenlik zafiyetlerine yol açabilir. Genellikle, protokol testleri sadece geçerli mesajların doğru işlenmesine odaklanır. Ancak, saldırganlar sıklıkla standart dışı, bozuk veya beklenmedik mesajlar göndererek sistemleri istismar etmeye çalışır.

Seçilen Senaryo: "Fuzzing" ile Protokol Zafiyeti Tespiti Bu proje, "Fuzzing" adı verilen bir güvenlik testi tekniğini kullanarak OCPP implementasyonlarının beklenmedik veya hatalı girdilere karşı nasıl davrandığını analiz etmeyi amaçlamaktadır. Fuzzing, protokole kasıtlı olarak bozuk, rastgele veya geçersiz veri göndererek, protokolün tasarımı veya implementasyonundaki zafiyetleri (örn: çökme, hafıza sızıntısı, yetkisiz durum değişiklikleri) ortaya çıkarır.

Zafiyetin Kök Nedeni: Zafiyetin kök nedeni, OCPP implementasyonlarının (hem EVSE hem de CSMS tarafında) **yetersiz girdi doğrulaması, hatalı durum yönetimi veya beklenmedik mesaj tiplerini/değerlerini işlerken oluşan yazılımsal hatalardır (bug'lar).**

2. Senaryo Analizi: Tehdit Modeli ve Zafiyetin Kök Nedeni

Tehdit Modeli (Protokol Sahteciliği): Saldırgan (fuzzer), doğrudan OCPP protokolünün iletişim kanalına erişim sağlayarak veya meşru bir istemci gibi davranarak, CSMS'e veya EVSE'ye hedeflenen bozuk mesajlar gönderir.

Zafiyetin Kök Nedeni: Yazılımsal Hatalar ve Güvenilir Olmayan Girdi İşleme EV şarj sistemleri karmaşık yazılımlar içerir. Bu yazılımların, her türlü beklenmedik senaryoya (ağ hataları, kasıtlı saldırılar) karşı sağlam olması gerekir. Protokol implementasyonlarında görülen başlıca zafiyet nedenleri şunlardır:

- **Yetersiz Girdi Doğrulaması:** CSMS/EVSE, gelen OCPP mesajlarındaki alanların (örn. transactionId, meterValue) tipini, uzunluğunu veya değer aralığını yeterince kontrol etmez.
- **Hafıza Güvenliği Hataları:** Kasıtlı olarak büyük veri gönderimi (buffer overflow), sisteme enjekte edilen kodun çalışmasına veya sistemin çökmesine neden olabilir.
- **Hatalı Durum Yönetimi:** Beklenmedik bir mesaj alındığında sistemin kararsız bir duruma girmesi veya beklenmedik bir işlem yapması.
- **Kaynak Tükenmesi:** Yetersiz kaynak yönetimi nedeniyle, belirli mesaj tiplerinin aşırı işlenmesi sistem kaynaklarını tüketebilir.

3. Saldırı Vektörü: DoS, Veri Bütünlüğü Bozulması ve Potansiyel Kontrol Kaybı

Aksiyon: Bir saldırgan (fuzzer), OCPP istemcisi gibi davranarak hedef CSMS sunucusuna (veya bir EVSE'ye) çeşitli bozuk OCPP mesajları gönderir.

1. **Mesaj Oluşturma:** Fuzzer, StartTransaction, StopTransaction, Authorize, MeterValues gibi OCPP mesajlarını alır. Bu mesajların içeriklerini rastgele veya önceden tanımlanmış kurallara göre (örn. geçerli bir alana çok uzun metin, geçersiz karakterler, eksik alanlar) bozar.
2. **Mesaj Gönderimi:** Bozulan bu mesajlar, hedef CSMS'e art arda ve yüksek hızda gönderilir.
3. **Sistem Tepkisi:** CSMS, bu beklenmedik mesajları işlerken:
 - a. **Çökebilir (Crash):** Hizmet Reddi (DoS) durumuna yol açar.
 - b. **Yanıt Vermeyi Durdurabilir:** Yine bir DoS şekli.
 - c. **Hafıza Sızıntısı Yaşayabilir:** Hassas bilgilerin açığa çıkmasına neden olabilir.
 - d. **Beklenmedik Durum Değişiklikleri Yapabilir:** Örneğin, ücretsiz şarj oturumları başlatabilir veya mevcut şarjları yanlış şekilde sonlandırabilir.
 - e. **Anormal Loglama:** Güvenlik tespitini zorlaştıran garip log girdileri oluşturabilir.

4. Simülasyon Ortamı ve Metodoloji

Projenin temelinde, "**OCPPStorm**" makalesinde bahsedilen metodolojiye benzer bir yaklaşım olacaktır.

- **Hedef Sistem:** Açık kaynaklı bir OCPP CSMS implementasyonu (örneğin, Python tabanlı bir OCPP sunucu kütüphanesi) veya basitleştirilmiş bir simülasyon sunucusu.
- **Fuzzer Geliştirme:** Python dilinde basit bir fuzzer script'i yazılır. Bu fuzzer:
 - OCPP mesaj formatlarını temel alır.
 - Mesaj payload'ını (JSON içeriğini) değiştirir (rastgele veri, çok uzun stringler, geçersiz tipler).
 - Hazırlanan bozuk mesajları hedef CSMS'e gönderir.
- **Tespit ve Analiz:**
 - CSMS sunucusunun logları izlenir.
 - Fuzzer'ın gönderdiği her mesaja karşılık CSMS'ten gelen yanıtlar kaydedilir.
 - Sunucunun çökme, donma veya hatalı yanıt verme durumları analiz edilir.

5. Simülasyon Sonuçları ve Anomali Tespiti

Fuzzer Tarafı:

Fuzzing başlatıldı... Mesaj ID: 1, Gönderilen: {"command": "BootNotification", "vendor": "VERY_LONG_STRING..."} Mesaj ID: 2, Gönderilen: {"command": "InvalidCommand", "data": "random_data"} ... Mesaj ID: 1234, Sunucu yanıt vermiyor. Tespit: Sunucu Çöktü (DoS).

CSMS Sunucu Logları (Örnek):

```
ERROR: [2025-11-02 14:35:12] Unhandled exception in handler: ValueError: invalid literal for int() with base 10: 'VERY_LONG_STRING...' CRITICAL: [2025-11-02 14:35:13] Server crashed due to unhandled payload.
```

Bulguların Yorumlanması: Simülasyon, fuzzing tekniklerinin OCPP implementasyonlarındaki gizli zafiyetleri (örn. beklenmedik veri tipleri veya uzunluklarına karşı zayıf savunma) nasıl etkili bir şekilde ortaya çıkarabileceğini gösterir. Tespit edilen anomali, genellikle bir DoS durumu veya beklenmedik bir hata mesajı ile kendini gösterir.

6. Savunma Stratejileri ve Çözüm Önerileri

Bu tür zafiyetlere karşı geliştirilecek çözümler, sağlam yazılım geliştirme prensiplerine dayanır.

1. Çözüm 1: Kapsamlı Girdi Doğrulama (Input Validation):

- Tüm gelen OCPP mesaj alanları için katı tip, uzunluk ve değer aralığı doğrulaması yapılmalıdır. JSON Schema validation, bu konuda etkili bir araçtır.
- Beklenmeyen veya geçersiz karakterler temizlenmeli veya reddedilmelidir.

2. Çözüm 2: Sağlam Hata İşleme (Robust Error Handling):

- Hatalı veya beklenmedik mesajlar alındığında sunucunun çökmesi yerine, uygun bir hata mesajı (örn. ProtocolError) ile yanıt vermesi ve çalışmaya devam etmesi sağlanmalıdır.
- Uncaught (yakalanmamış) istisnalar (exceptions) olmamalıdır.

3. Çözüm 3: Güvenli Kodlama Pratikleri ve Fuzzing Testleri:

- Geliştirme sürecine "OCPPStorm" gibi araçlarla otomatik fuzzing testleri entegre edilmelidir.
- Hafıza güvenliği dillerinin (Rust gibi) kullanılması veya hafıza güvenliğine odaklanan statik/dinamik analiz araçlarının kullanılması.

7. Sonuç ve Değerlendirme

Özet: Bu proje, fuzzing tekniklerinin OCPP implementasyonlarındaki gizli protokol zafiyetlerini (yetersiz girdi doğrulaması ve hatalı hata işleme) ortaya çıkarmada ne kadar etkili olduğunu göstermiştir. Bu tür zafiyetler, EV şarj altyapısında Hizmet Reddi (DoS) ve diğer operasyonel veya finansal risklere yol açabilir.