Sensör Verisi Zehirleme (SDP) Senaryosu için Simülasyon Ortamı Raporu

1. Giriş ve Amaç

Bu rapor, "Sensör Verisi Zehirleme (SDP)" olarak tanımlanan spesifik anomali senaryosunu test etmek amacıyla kurulacak simülasyon ortamını detaylandırmaktadır.

Senaryonun temel hedefi, şarj istasyonu sensör verilerinin (akım, gerilim vb.) OCPP telemetri akışı üzerinden "Ortadaki Adam" (Man-in-the-Middle) saldırısı ile manipüle edilmesidir. Amaç, sistemi çökertmek değil, yapay zekâ tabanlı anomali tespit sistemlerinin (IDS) öğrenme sürecine istatistiksel olarak zehirlenmiş (poisoned) sahte veriler sokmaktır.

Bu simülasyon ile hedeflenen, uzun vadede küçük ve periyodik sapmalar (örn. ±%2-5) enjekte ederek merkezi modelin "normal" profilini kaydırmak (model drift / dejenerasyon) ve gelecekteki gerçek saldırıları (örn. enerji hırsızlığı) tespit edilemez hale getirmektir.

2. Simülasyon Ortamı Mimarisi

Simülasyon ortamı, bir "Gateway (Ağ Geçidi)" modeli üzerine kurulacaktır. Bu "Gateway", Şarj İstasyonu (CP) ana kontrolcüsünü temsil edecek ve iki farklı ağı birbirine bağlayacaktır:

1. **Lokal Ağ (CAN-bus):** CP içindeki sensörler, röleler ve güç elektroniği arasındaki yerel haberleşme. Bu, fiziksel donanım olmadan `vcan` (sanal CAN) arayüzü ile simüle edilecektir.
2. **Geniş Alan Ağı (OCPP):** CP'nin Merkezi Yönetim Sistemine (CSMS) bağlandığı internet tabanlı ağ protokolü.

Saldırı, OCPP kanalı üzerinde "Ortadaki Adam" (Man-in-the-Middle) olarak konumlanarak gerçekleştirilecektir.

3. Gerekli Bileşenler

Simülasyon, fiziksel donanım gerektirmeden aşağıdaki yazılım bileşenleri ile kurulacaktır:

Bileşen	Araç/Kütüphane	Görevi ve Gerekçesi
İşletim Sistemi	Ubuntu 22.04 LTS (VM)	Gerekli ağ ve çekirdek modüllerini (`vcan`) sağlar.
Sanal Sensör Ağı	`vcan` (`can-utils`)	Fiziksel CAN-bus ağını simüle eden sanal arayüz.
Programlama	Python 3, pip, venv	Simülatör script'lerini çalıştırmak için kullanılır.
Lokal Ağ İletişimi	`python-can` kütüphanesi	`vcan0` arayüzüne sensör verisi göndermek ve almak.
Geniş Alan Ağı	`ocpp` kütüphanesi	CP ve CSMS simülatörlerini Python'da hızla oluşturmak.
Saldırı Vektörü	`mitmproxy`	OCPP (WebSocket) trafiğini yakalayan, değiştiren ve ileten MitM proxy aracı.

4. SDP Senaryosu Konfigürasyonu ve Parametre Değişiklikleri

Bu bölüm, SDP senaryosunu uygulamak için `mitmproxy` üzerinde yapılacak spesifik konfigürasyonları ve değiştirilecek parametreleri detaylandırır.

Temel Konfigürasyon:

1. **Zafiyet Oluşturma:** SDP senaryosu "veri doğrulama/etiketleme süreçlerinin zayıf olmasını" gerektirir. Bu, simülasyonda CP-CSMS bağlantısı şifresiz (`ws://` - plain websocket) olarak yapılandırılarak sağlanacaktır.
2. **Ağın Yönlendirilmesi:** CP simülatörü, CSMS'ye doğrudan bağlanmak yerine, `mitmproxy`'nin çalıştığı porta (örn. `localhost:8080`) bağlanmaya zorlanacaktır.

Parametre Varyasyonları (SDP Saldırısı):

`mitmproxy`'ye yüklenecek olan saldırı script'i, aşağıdaki parametrelere göre yapılandırılmalıdır:

1. Hedef Veri: `MeterValues`

- **Hedef OCPP Mesajı:** `MeterValues`.
- **Hedef CAN ID:** Bu mesajı tetikleyen lokal CAN mesajı (örn. `ID 0x300`).

- **Amaç:** `mitmproxy` script'i, CP'den CSMS'ye giden tüm `MeterValues` JSON mesajlarını yakalamalı ve içindeki enerji/akım/gerilim verilerini manipüle etmelidir.

2. Manipülasyon Parametresi: Sapma Büyüklüğü (±%2-5)

- **SDP Senaryo Gerekçesi:** "Saldırganın sensör verisini küçük, periyodik sapmalarla değiştirebilme yeteneği (ör. ±2-5%)" ve "küçük ölçekli sapmalar ... modelin normal kabul edeceği toleriler içinde kalır".
- **Simülasyon Parametresi:** `mitmproxy` script'indeki `sapma_yuzdesi` değişkeni.
- **Varyasyonlar:**
  - `value = value * 1.03` (+%3 sapma)
  - `value = value * 0.98` (-%2 sapma)
- **Beklenen Sonuç:** CSMS loglarındaki veriler, CP loglarındaki (gerçek) verilerden bu küçük yüzde kadar farklı olacaktır.

3. Manipülasyon Parametresi: Model Kaydırma (Kümülatif Etki)

- **SDP Senaryo Gerekçesi:** "Zaman içinde bu küçük sapmalar birikir ve merkezi model 'normal' profilini yeniden tanımlar" (Kümülatif Etki).
- **Simülasyon Parametresi:** `kaydirma_yonu` (Drift Yönü).
- **Varyasyonlar:**
  - **Başarısız Senaryo (Rastgele Sapma):** `value = value * (1 + random.uniform(-0.05, 0.05))`

    Analiz: Bu, verinin ortalamasını değiştirmez. Modelin "normal" tanımı değişmez.
  - **Başarılı Senaryo (Tek Yönlü Sapma):** `value = value * 1.03`

    Analiz: Tüm sapmaların sürekli olarak *pozitif* yönde yapılması, modelin öğrendiği "normal" verinin ortalamasını yavaş yavaş ve kasten yukarı çeker. Bu, "model drift" veya "dejenerasyon" olarak adlandırılan durumdur.

4. Manipülasyon Parametresi: Tespit Edilemezlik (Periyodiklik)

- ***SDP Senaryo Gerekçesi:*** *Tespit sistemlerinden kaçınmak için "periyodik" ve "uzun süreli" manipülasyon.*
- ***Simülasyon Parametresi:*** `zehirleme_sikligi` *(Poisoning Frequency).*
- ***Varyasyonlar:***
  - ***Gürültülü (Tespit Edilebilir) Senaryo:*** `if True:` *(Her* `MeterValues` *mesajını zehirle).*

    - ***Sinsi (Gizli) Senaryo:*** *`if random.random() < 0.2:` (Mesajların sadece rastgele %20'sini zehirle).*
- ***Beklenen Sonuç:*** *Düşük sıklıkta zehirleme, basit istatistiksel alarmları tetiklemeden modelin yavaşça zehirlenmesini sağlar.*

5. Raporlama ve Göstergeler

Saldırının başarısını raporlamak için üç farklı noktadan log alınmalıdır:

1. ***Gerçek Veri (CP Logları):*** *`cp_simulator.py` script'inin `vcan0`'dan okuduğu ve gönderilmek üzere hazırladığı orijinal `MeterValues` verisi (örn: `Enerji: 1500Wh`).*
2. ***Manipülasyon Kanıtı (MitM Logları):*** *`mitmproxy`'nin arayüzünde gösterilen "Mesaj 1500Wh olarak alındı -> 1545Wh (%3 artırıldı) olarak değiştirildi" logu.*
3. ***Zehirlenmiş Veri (CSMS Logları):*** *`csms_server.py`'nin veritabanına kaydettiği veya "modelini eğitmek" için kullandığı nihai, zehirlenmiş veri (örn: `Enerji: 1545Wh`).*

Bu üç logun karşılaştırılması, SDP saldırısının başarıyla gerçekleştirildiğini ve model eğitim verisinin manipüle edildiğini kanıtlayacaktır.

6. Sonuç

Bu simülasyon ortamı, tanımlanan altyapıyı (Linux, `vcan`, `mitmproxy`) kullanarak, "Sensör Verisi Zehirleme" anomali senaryosunu test etmek için gerekli tüm bileşenleri ve parametrik kontrolü sağlamaktadır. Tüm testler, etik kurallar çerçevesinde, izole edilmiş sanal ağ üzerinde yapılacaktır.