# BlockSentinel Projesi İçin Anomali Senaryosu Raporu

## 11. Anomali: "UDS Security Access (Servis 0x27) Kaba Kuvvet ve Zamanlama Saldırısı" Senaryosu

### Senaryonun Amacı ve Kapsamı

***Anomali Nedir?***

UDS (Unified Diagnostic Services - ISO 14229), modern araçlardaki Elektronik Kontrol Üniteleri (ECU) ile harici tanılama cihazları (diagnostic testers) arasındaki iletişimi yöneten standart uygulama katmanı protokolüdür.2 Bu protokol, basit hata kodu (DTC) okumanın 3 çok ötesinde, ECU ayarlarını kalıcı olarak değiştirme 4 ve en kritik olarak, ECU'nun kendi yazılımını yeniden programlama (re-flashing) 2 gibi son derece güçlü işlevlere izin verir.

Bu kritik işlevler, "Security Access" (Güvenlik Erişimi) olarak bilinen ve UDS Servis ID 0x27 tarafından yönetilen bir kimlik doğrulama mekanizmasıyla korunmaktadır.4

***Senaryonun Amacı:***

Bu senaryonun amacı, bir saldırganın aracın fiziksel CAN-Bus ağına (genellikle kokpit altındaki OBD-II portu üzerinden) 8 erişerek, Servis 0x27'nin kullandığı "Seed/Key" (Tohum/Anahtar) kimlik doğrulama mekanizmasını 5 atlatmasını (bypass) simüle etmektir.

Kapsam:

Saldırı, BlockSentinel projesinin de odaklandığı Şarj Kontrol Ünitesi (CCU), Batarya Yönetim Sistemi (BMS) veya Araç Kontrol Ünitesi (VCU) gibi kritik ECU'ları hedefler. Başarılı bir saldırı, saldırgana bu ECU'ların korumalı parametrelerini (örneğin, maksimum şarj akım limitleri, batarya sağlık durumu (SoH) sayaçları) değiştirme 13 veya doğrudan zararlı/değiştirilmiş yazılım yükleme yetkisi verir.

Bu senaryo, dokümanda Anomali 13 olarak yüzeysel bahsedilen "Yetkisiz Tanılama İsteği" konusunu derinlemesine ele almaktadır. Tehdit, sadece "yetkisiz" bir istek göndermek değil, bu isteği "yetkili" hale getiren kimlik doğrulama mekanizmasının (Servis 0x27) kendisinin kırılmasıdır.

### Özet

Saldırgan, hedef aracın OBD-II portuna 8 bir CAN adaptörü (örn. PCAN-USB veya CANable) bağlar. Saldırı, standart bir UDS iletişim akışını taklit ederek başlar:

1. **Oturum Değişikliği (Servis 0x10):** Saldırgan, UDS Servis 0x10 (Diagnostic Session Control) 15 kullanarak ECU'yu sınırlı "Varsayılan Oturum"dan (Default Session, 0x01) 15, korumalı servislere erişimin denenebileceği "Genişletilmiş Tanılama Oturumu"na (Extended Diagnostic Session, 0x03) 15 geçirir.
2. **Tohum (Seed) Talebi (Servis 0x27, 0x01):** Ardından, Servis 0x27, Alt Fonksiyon 0x01 ('requestSeed') 5 komutunu göndererek ECU'dan bir "seed" (rastgele bir sorgu, örn. 4 byte'lık AA BB CC DD 11) talep eder.
3. **Anahtar (Key) Gönderimi (Servis 0x27, 0x02):** Saldırganın, bu "seed"i ve *sadece üretici tarafından bilinen gizli bir algoritmayı* 4 kullanarak doğru "key"i (anahtar) hesaplaması ve Servis 0x27, Alt Fonksiyon 0x02 ('sendKey') 5 ile geri göndermesi gerekir.

Zafiyet, ISO 14229 standardının bu "gizli algoritmayı" tanımlamayıp, üreticinin takdirine bırakmasıdır.4 Bu durum, genellikle zayıf (örn. basit XOR/rotasyon) 20, tersine mühendisliğe açık 22 veya çok kısa anahtar/seed alanlarına (örn. 2-byte) 24 sahip implementasyonlara yol açmıştır.

Saldırgan, bu zayıf implementasyonu kırmak için iki temel yöntem kullanır:

1. **Kaba Kuvvet (Brute-Force):** Eğer anahtar alanı küçükse (örn. 2-byte key = 65,536 olasılık) 24, saldırgan tüm olasılıkları hızla deneyebilir.25
2. **Zamanlama ve Güç Kesintisi (Timing & Power Glitching) Saldırısı:** Bu, en gelişmiş siber-fiziksel yöntemdir. Birçok ECU, kaba kuvvet saldırılarını önlemek için yanlış anahtar denemelerini (lockout) sayar.5 Ancak bu sayacı, genellikle *yavaş* olan kalıcı belleğe (NVM) kaydederler.11 Bu NVM kaydı, CPU'nun "anahtar yanlış" kararını vermesinden (örn. $t\_1=5ms$) *sonra*, ancak "hata mesajı gönderme"den (örn. $t\_2=12ms$) *önce* bir gecikme (örn. 7ms) yaratır.11 Saldırgan, bu zaman penceresini (örn. $t\_x=8ms$) kullanarak, yanlış bir anahtar gönderdikten *hemen sonra* (NVM'ye yazma tamamlanmadan) ECU'nun gücünü aniden keser.11 ECU yeniden başladığında, başarısız deneme kaydedilmemiş olur ve deneme sayacı sıfırlanır. Bu, kaba kuvvet saldırılarını engelleyen kilitleme mekanizmasını etkisiz hale getirir ve saldırgana sonsuz deneme hakkı tanır.

Başarılı olduğunda (ECU'dan 67 02 21 pozitif yanıtı alındığında), saldırgan "root" seviyesinde yetki kazanır ve Batarya Yönetim Sistemi'nin (BMS) kritik şarj parametrelerini manipüle etmek 13 gibi eylemlere başlar.

### Hedef Varlıklar

* **Batarya Yönetim Sistemi (BMS):** En kritik hedeftir. Şarj/deşarj limitleri, hücre dengeleme algoritmaları, sıcaklık eşikleri ve "State of Health" (SoH - Batarya Sağlığı) raporlaması gibi hayati verileri barındırır.13
* **Şarj Kontrol Ünitesi (CCU):** BlockSentinel projesinin odaklandığı ve şarj oturumunu yöneten ana ECU.
* **ECU Yazılımı (Firmware):** Korumalı UDS servisleri, ECU'nun yazılımını yeniden programlama (re-flashing) yeteneği sağlar.2
* **OBD-II Portu:** Saldırının fiziksel giriş kapısıdır.8
* **Üreticiye Özel Anahtar Algoritması:** Saldırganın tersine mühendislik 22 veya kaba kuvvet 25 yoluyla elde etmeye çalıştığı gizli entelektüel mülkiyet.20

### İlgili Zafiyetler

* **Zayıf Kriptografi Uygulaması:** ISO 14229 standardının bir algoritma belirtmemesi 4, üreticilerin zayıf (örn. basit XOR veya rotasyon) 20 veya kısa anahtar/seed alanları (örn. 2-byte) 24 kullanmasına ("gizlilik yoluyla güvenlik") yol açar.
* **Güvensiz Kilitleme (Lockout) Mekanizmaları:** Başarısız deneme sayaçlarının NVM'ye yavaş yazılması, "Power Glitching" (Güç Kesintisi) saldırılarına 11 karşı savunmasız bir zaman penceresi yaratır.
* **Fiziksel Erişim Zafiyeti:** OBD-II portunun aracın içinde kolayca erişilebilir ve genellikle temel bir kimlik doğrulama katmanından yoksun olması.8
* **Düşük Seed Entropisi:** Bazı ECU'ların, her 'requestSeed' 11 talebinde yeterince rastgele (veya bazen hep aynı) "seed" üretememesi.11
* **Bilgi İfşası:** Tanılama araçlarının veya firmware güncellemelerinin, gizli algoritmayı 20 ortaya çıkaracak şekilde tersine mühendisliğe 22 karşı yeterince korunmaması.

### Tehdit Kategorisi (STRIDE)

Bu senaryo için STRIDE 29 modeli analizi:

* **S - Spoofing (Sahtecilik):** Saldırgan, UDS protokolü aracılığıyla kendisini meşru bir tanılama cihazı (örn. yetkili servis teknisyeni aracı) 3 olarak tanıtır.
* **T - Tampering (Kurcalama):** *Saldırının ana hedefidir.* Yetki elde edildikten sonra, saldırgan kritik BMS parametrelerini (şarj limitleri, sıcaklık eşikleri) 13, Araç Kimlik Numarasını (VIN) veya ECU yazılımının kendisini 2 değiştirir (kurcalar).
* **R - Repudiation (İnkar Etme):** *BlockSentinel'in çözdüğü ana sorun.* Başarılı bir saldırgan, UDS Servis 0x14'ü (Clear Diagnostic Information) 3 kullanarak ECU üzerindeki yerel hata günlüklerini (DTC) siler. Bu, saldırının "inkar edilmesine" olanak tanır.
* **I - Information Disclosure (Bilgi İfşası):** Yetki elde edildikten sonra, saldırgan korumalı "Read Data By Identifier" (Servis 0x22) 4 komutlarını kullanarak hassas telemetri verilerini, kullanıcı davranışlarını 33 veya bellekte saklanan diğer kriptografik anahtarları 25 çalabilir.
* **D - Denial of Service (Hizmet Reddi):** Yetkili saldırgan, "ECU Reset" (Servis 0x11) 4 komutunu sürekli bir döngü içinde göndererek veya kritik parametreleri geçersiz değerlere ayarlayarak BMS'i 13 ya da şarj istasyonunu hizmet dışı bırakabilir.
* **E - Elevation of Privilege (Yetki Yükseltme):** *Saldırının kalbidir.* Saldırgan, "Default Session" 15 (yetkisiz) seviyesinden, "Security Access" 34 (tam yetkili/üretici) seviyesine yükselir.

### SWOT Analizi

Bu analiz, UDS Security Access Bypass senaryosunun kendisini (saldırgan ve savunmacı bakış açısıyla) inceler.

| **Kategori** | **Analiz** |
| --- | --- |
| **Güçlü Yönler (Strengths)** | (Senaryonun / Saldırının güçlü yanları)  1. **Yüksek Etki:** Başarılı bir saldırı, saldırgana ECU üzerinde tam (root-benzeri) kontrol sağlar ve siber-fiziksel hasara (örn. yangın) yol açabilir.13  2. **Standartlaşma:** Saldırı, üreticiden bağımsız olarak 2 UDS (ISO 14229) kullanan tüm araçlar için geçerli bir vektördür.  3. **Belgelenmiş Protokol:** Saldırganın protokolü (servisler, alt fonksiyonlar) öğrenmesi kolaydır.4 |
| **Zayıf Yönler (Weaknesses)** | (Saldırganın / Saldırının zayıf yanları)  1. **Fiziksel Erişim Zorunluluğu:** Senaryonun en yaygın varyantı, aracın OBD-II portuna 8 fiziksel erişim gerektirir, bu da yakalanma riskini artırır.  2. **Üreticiye Bağımlılık:** Standart (Servis 0x27) 4 olmasına rağmen, *algoritma* 4 üreticiye özeldir. Bir üretici için geliştirilen exploit (örn. Power Glitch zamanlaması 11 veya tersine mühendislik 22) diğeri için çalışmaz.  3. **Zaman Tüketimi:** Kaba kuvvet 25 veya Power Glitching 11 saldırıları, basit bir 'spoofing' saldırısından çok daha fazla zaman, uzmanlık ve potansiyel olarak özel donanım (güç kontrol rölesi) gerektirir. |
| **Fırsatlar (Opportunities)** | (BlockSentinel / Savunmacı için Fırsatlar)  1. **Durum Bazlı (Stateful) IDS Geliştirme:** Bu senaryo, basit, durumsuz (stateless) kuralların (örn. sadece ID'ye bakmak) yetersizliğini kanıtlar ve BlockSentinel'in "stateful monitoring" (durumsal izleme) 35 özelliğini geliştirmesi için net bir gerekçe sunar.  2. **Adli Kanıt (Forensics) Değeri:** Saldırganın saldırı sonrası izlerini (DTC) silme (Repudiation 31) yeteneği 3, BlockSentinel'in "değişmez blockchain loglaması" 1 özelliğini pazarlamak için mükemmel bir fırsattır.  3. **Yapay Zeka (AI) Farklılaştırması:** AI, meşru (bakım) UDS oturumlarının 3 desenleri ile anomali (saldırı) UDS oturumlarının (örn. Fuzzing 38 veya Brute-Force) desenlerini ayırt etmek için kullanılabilir. |
| **Tehditler (Threats)** | (BlockSentinel / Savunmacı için Tehditler)  1. **Meşru Aracın Kötüye Kullanımı:** Saldırgan, kaba kuvvet denemek yerine, içinde geçerli anahtar algoritması bulunan *meşru bir tanılama aracını* çalar veya ödünç alır. Bu durumda, trafik IDS için %100 "normal" ve "meşru" görünür.  2. **Gelecek Protokoller (DoIP):** Yeni araçlar, UDS'yi şifreli Ethernet (DoIP - Diagnostics over IP) 11 üzerinden taşımaya başlıyor. Bu, ağ tabanlı bir IDS'in (BlockSentinel gibi) trafiği "kör" hale gelmesine ve şifreli paketleri analiz edememesine neden olabilir.  3. **Güçlü Kriptografi:** Üreticiler zayıf algoritmaları 20 bırakıp, donanım güvenlik modülleri (HSM) veya AES 20 gibi standart kriptografi kullanmaya başlarsa, bu spesifik (brute-force) saldırı vektörü tamamen kapanır. |

### Simülasyon & Çözüm Yaklaşımı (BlockSentinel IDS Tespiti)

**Amaç**

python-can 39 ve udsoncan 17 kütüphanelerini kullanarak sanal bir CAN arayüzü (vcan) üzerinde, zayıf bir UDS (0x27) implementasyonuna yönelik bir kaba kuvvet (brute-force) saldırısını simüle etmek. Eş zamanlı olarak, bu saldırıyı Kural IDS-UDS-01 ("Kaba Kuvvet Tespiti") kullanarak tespit eden ve BlockSentinel'e bir uyarı (alert) ve değişmez log (immutable log) gönderen bir "BlockSentinel IDS" monitör script'i çalıştırmak.

**Gereksinimler / Ortam**

* **Arayüz:** Sanal CAN (vcan). (Kurulum: sudo modprobe vcan && sudo ip link add dev vcan0 type vcan && sudo ip link set up vcan0)
* **Kütüphaneler:** python-can, udsoncan (Python kütüphaneleri).
* **Sunucu (ECU):** server\_ecu.py (Zayıf bir anahtar algoritmasını simüle eder, örn. key = (seed ^ 0x1234) 40).
* **İstemci (Saldırgan):** attacker.py (Tüm anahtar olasılıklarını dener).17
* **Monitör (BlockSentinel IDS):** ids\_monitor.py (Trafiği pasif olarak dinler ve saldırıyı tespit eder).

**Simülasyon Akışı (Adım Adım)**

1. **Baseline Oluşturma (Terminal 1 - ECU):** python3 server\_ecu.py çalıştırılır. Bu script, vcan0 üzerinde 0x7E0 ID'sini dinler ve 0x27 servisi için 2-byte'lık zayıf bir algoritma tanımlar.
2. **IDS'i Başlatma (Terminal 2 - BlockSentinel):** python3 ids\_monitor.py çalıştırılır. Bu script, vcan0 arayüzündeki *tüm* trafiği pasif olarak dinler ve Bölüm 2.7'deki Kural IDS-UDS-01'i uygulamak için 0x7F 27 35 (Invalid Key) yanıtlarını sayar.25
3. **Saldırı Senaryosu Başlatma (Terminal 3 - Saldırgan):** python3 attacker.py çalıştırılır. Bu script, client.change\_session(3) (Extended Session) 17 gönderir, ardından client.request\_seed(1) 17 ile seed'i alır.
4. **Monitoring (IDS):** attacker.py, 0'dan 65535'e kadar (2-byte 24) anahtarları denemeye başlar.
   * *Saldırgan (Terminal 3) Çıktısı:* Denendi: 0000 -> BAŞARISIZ, Denendi: 0001 -> BAŞARISIZ...
   * *IDS (Terminal 2) Çıktısı:* Bu denemelere ECU'nun verdiği 0x7E8 # 7F 27 35 yanıtlarını görür ve sayacını artırır: Geçersiz Anahtar Denemesi Tespit Edildi. Toplam: 1, Geçersiz Anahtar Denemesi Tespit Edildi. Toplam: 2...
5. **Tespit Algoritmaları (IDS):** Sayaç önceden tanımlanmış eşiği (örn. 10) aştığında (Kural IDS-UDS-01), ids\_monitor.py tetiklenir:
   * UDS KABA KUVVET (BRUTE-FORCE) SALDIRISI TESPİT EDİLDİ!
   * Olay blockchain'e kaydediliyor...
   * Hash: 0x...
6. **Saldırı Sonucu:** attacker.py (Terminal 3) doğru anahtarı bulur ve BAŞARILI! Anahtar bulundu:... çıktısını verir.
7. **Raporlama:** BlockSentinel IDS, saldırıyı *saldırgan başarılı olmadan çok önce* (ilk 10 denemede) tespit etmiş ve BlockSentinel'e değişmez bir adli kanıt kaydetmiştir.

### Pseudocode (BlockSentinel IDS Tespit Mekanizması)

Python

import can  
import time  
  
# BlockSentinel IDS için Kural IDS-UDS-01  
# Zafiyet: NRC 0x35 (Invalid Key) veya 0x36 (ExceedNumberOfAttempts) [5] spam'i.  
# Tespit: Belirli bir sürede 'n' adetten fazla başarısız girişimi izleme (Rate-based analysis).  
  
# --- İzleme Parametreleri ---  
INVALID\_KEY\_ATTEMPTS = 0  
WINDOW\_START\_TIME = time.time()  
TIME\_WINDOW\_SECONDS = 30 # 30 saniyelik zaman penceresi  
ATTEMPT\_THRESHOLD = 10 # 10 denemeden fazlası alarmı tetikler  
  
# --- BlockSentinel Entegrasyonu (Simüle edilmiş) ---  
def log\_event\_to\_blockchain(event\_data):  
 """  
 Bu fonksiyon, tespit edilen olayı BlockSentinel'in değişmez defterine kaydeder.  
 Gerçek implementasyonda:  
 1. event\_data'yı JSON olarak formatlar.  
 2. Verinin kriptografik hash'ini (SHA-256) alır.  
 3. Bu hash'i bir Akıllı Kontrata (Smart Contract) gönderir.  
 4. Blockchain'e değiştirilemez bir adli kanıt olarak kaydeder.[36, 37, 43]  
 """  
 print(f"\n Değişmez Kayıt Oluşturuldu. Veri: {event\_data}")  
  
# --- Ana IDS Monitör Döngüsü ---  
bus = can.interface.Bus(channel='vcan0', bustype='socketcan')  
print(" BlockSentinel UDS Monitörü Başlatıldı. vcan0 dinleniyor...")  
  
try:  
 for msg in bus:  
 # Sadece UDS Negatif Yanıtlarını (SID 0x7F) [42] filtrele  
 # Gerçekte, yanıt ID'si (örn. 0x7E8) dinamik olarak öğrenilmelidir.  
 if msg.data == 0x7F:  
   
 # Negatif yanıtın UDS 0x27 (Security Access) için olup olmadığını kontrol et  
 if msg.data == 0x27:  
   
 # NRC'nin 0x35 (Invalid Key) olup olmadığını kontrol et  
 if msg.data == 0x35:  
 INVALID\_KEY\_ATTEMPTS += 1  
 print(f" Geçersiz Anahtar Denemesi (0x35) Tespit Edildi. Toplam: {INVALID\_KEY\_ATTEMPTS}")  
   
 # Zaman penceresi dolduysa sayacı sıfırla  
 current\_time = time.time()  
 if (current\_time - WINDOW\_START\_TIME) > TIME\_WINDOW\_SECONDS:  
 INVALID\_KEY\_ATTEMPTS = 1  
 WINDOW\_START\_TIME = current\_time  
 print(" Zaman penceresi sıfırlandı.")  
  
 # Kuralı kontrol et: Eşiği aştık mı?   
 if INVALID\_KEY\_ATTEMPTS > ATTEMPT\_THRESHOLD:  
 current\_time = time.time()  
 print(f"\n KURAL IDS-UDS-01 TETİKLENDİ: KABA KUVVET (BRUTE-FORCE) SALDIRISI TESİP EDİLDİ!")  
   
 event = {  
 "timestamp": int(current\_time),  
 "attack\_type": "UDS\_Brute\_Force\_0x27",  
 "source\_can\_id": hex(msg.arbitration\_id),  
 "nrc\_code": "0x35 (InvalidKey)",  
 "attempts\_in\_window": INVALID\_KEY\_ATTEMPTS  
 }  
   
 # Saldırgan ECU loglarını silse bile (Repudiation) , biz kanıtı kaydederiz  
 log\_event\_to\_blockchain(event)  
   
 # Alarmdan sonra sayacı sıfırla (log spam'ini önle)  
 INVALID\_KEY\_ATTEMPTS = 0  
 WINDOW\_START\_TIME = time.time()  
  
except KeyboardInterrupt:  
 print(" Monitör durduruldu.")  
 bus.shutdown()

Kodun Genel Amacı (BlockSentinel Çözüm Yaklaşımı):

Bu pseudocode, BlockSentinel IDS'in temel "durum bazlı" (stateful) kural motorunu 35 temsil eder. vcan0 hattını pasif olarak dinler. Saldırganın attacker.py script'i 17 client.send\_key() komutunu her çağırdığında ve bu başarısız olduğunda, server\_ecu.py bir 7F 27 35 (Invalid Key) 5 yanıtı gönderecektir. Monitörümüz bu yanıtları sayar. 30 saniye içinde 10'dan fazla 25 başarısız deneme tespit ettiğinde, bunu bir "kaba kuvvet" saldırısı olarak tanımlar ve bu olayı BlockSentinel'in değişmez (immutable) 37 defterine (blockchain) adli bir kanıt 43 olarak kaydeder. Bu, saldırganın daha sonra Servis 0x14 32 ile ECU loglarını silerek saldırıyı inkar etme (Repudiation) 31 girişimini boşa çıkarır.

### Savunma / Hafifletme Önerileri

Bu saldırı vektörüne karşı savunma çok katmanlı olmalıdır:

1. **ECU Tasarım Seviyesi (Üretici):**
   * **Güçlü Kriptografi:** Zayıf, özel (proprietary) algoritmalar 20 yerine, endüstri standardı (örn. AES-128) 20 ve uzun (örn. 16-byte) seed/key kullanan bir mekanizma uygulanmalıdır.
   * **Güvenli Kilitleme (Lockout):** "Power Glitching" 11 saldırısını önlemek için, başarısız deneme sayacının NVM'ye yazılmasının atomik ve hızlı olması veya güç kesintisini algılayan donanımsal korumalar (örn. Brown-out Detectors) eklenmesi gerekir. Başarısız denemeler arası gecikme, sadece yazılımsal değil, donanımsal veya exponansiyel (katlanarak artan) olmalıdır.27
2. **Araç Mimarisi Seviyesi (Entegratör):**
   * **Fiziksel Güvenlik:** OBD-II portu 8 kilitli bir kapak veya yetkilendirme (örn. birincil ekran üzerinden yazılımsal olarak "kilidi açma") gerektirmelidir.10
   * **Ağ Segmentasyonu (Gateway Filtreleme):** Kritik güvenlik ECU'ları (BMS, CCU), düşük güvenlikli ECU'lardan (Infotainment, Telematik) 45 bir ağ geçidi (Gateway) ile ayrılmalıdır. Bu gateway, UDS tanılama mesajlarını (örn. 0x7E0) filtrelemeli ve aracın "Sürüş" veya "Şarj" modlarında kritik ağa yönlendirilmesini engellemelidir.46
3. **İzleme Seviyesi (BlockSentinel):**
   * **Durumsal Tespit (Stateful Detection):** Yukarıdaki (2.10) pseudocode'da detaylandırıldığı gibi, UDS protokolünün durumunu (oturum, yetki) takip eden ve Kural IDS-UDS-01, 02, 03'ü uygulayan bir IDS zorunludur.25
   * **Değişmez Adli Kayıt (Immutable Forensics):** Tespit edilen her anomali, saldırganın silemeyeceği 44 bir blockchain üzerine kaydedilmelidir. Bu, saldırının inkarını (Repudiation) 31 engeller ve siber-fiziksel olaylarda yasal kanıt sağlar.43

#### Alıntılanan çalışmalar

1. BlockSentinel\_ANOMALİLER (1).pdf
2. UDS - Unified Diagnostic Services - ISO 14229 - Vector, erişim tarihi Kasım 8, 2025, <https://www.vector.com/at/en/products/solutions/diagnostic-standards/uds-unified-diagnostic-services-iso14229/>
3. UDS Explained - A Simple Intro (Unified Diagnostic Services) - CSS Electronics, erişim tarihi Kasım 8, 2025, <https://www.csselectronics.com/pages/uds-protocol-tutorial-unified-diagnostic-services>
4. Introduction to UDS — udsoncan 1.19.0 documentation, erişim tarihi Kasım 8, 2025, <https://udsoncan.readthedocs.io/en/v1.19.0/udsoncan/intro.html>
5. What is the use-case of security levels according to ISO 14229 Unified Diagnostic Services (UDS)? - Stack Overflow, erişim tarihi Kasım 8, 2025, <https://stackoverflow.com/questions/49491800/what-is-the-use-case-of-security-levels-according-to-iso-14229-unified-diagnosti>
6. The ECU killer: broken authentication in automotive security - Cydrill, erişim tarihi Kasım 8, 2025, <https://cydrill.com/cyber-security/the-unlikely-ecu-killer-broken-authentication-in-automotive-security/>
7. Services — python-uds documentation, erişim tarihi Kasım 8, 2025, <https://python-uds.readthedocs.io/en/latest/services.html>
8. Addressing IoT Security Vulnerabilities in Electric Vehicles and Charging Stations - Coalfire, erişim tarihi Kasım 8, 2025, <https://coalfire.com/the-coalfire-blog/iot-security-vulnerabilities-electric-vehicles>
9. Understanding CAN Bus Vulnerabilities and How Blockchain Can Amplify Security - Medium, erişim tarihi Kasım 8, 2025, <https://medium.com/@chaincom/understanding-can-bus-vulnerabilities-and-how-blockchain-can-amplify-security-a58388bf1fb4>
10. Vehicle Cybersecurity Threats and Mitigation Approaches - NREL, erişim tarihi Kasım 8, 2025, <https://docs.nrel.gov/docs/fy19osti/74247.pdf>
11. ANALYSIS OF SECURITY VULNERABILITIES IN ... - Semantic Scholar, erişim tarihi Kasım 8, 2025, <https://pdfs.semanticscholar.org/2a87/110372a6b3d774798eefc3685fbb9e36c560.pdf>
12. Overview of Unified Diagnostic Services Protocol - Adventure, erişim tarihi Kasım 8, 2025, <https://nvdungx.github.io/unified-diagnostic-protocol-overview/>
13. Cybersecurity for Battery Management Systems | eInfochips, erişim tarihi Kasım 8, 2025, <https://www.einfochips.com/blog/cybersecurity-in-battery-management-system/>
14. Cyber-Physical Cloud Battery Management Systems: Review of Security Aspects - MDPI, erişim tarihi Kasım 8, 2025, <https://www.mdpi.com/2313-0105/9/7/382>
15. Accessing diagnostic session control (uds service 0x10) - CanBusHack, erişim tarihi Kasım 8, 2025, <https://canbushack.com/web/accessing-diagnostic-session-control-uds-service-0x10/>
16. Diagnostics and Communication Management Function Group - UDS Protocol Tutorial Part 2 - EmbeTronicX, erişim tarihi Kasım 8, 2025, <https://embetronicx.com/tutorials/automotive/uds-protocol/diagnostics-and-communication-management/>
17. udsoncan - PyPI, erişim tarihi Kasım 8, 2025, <https://pypi.org/project/udsoncan/>
18. Simple combined write & read example? · Issue #2 · marcelbuesing/socketcan-isotp, erişim tarihi Kasım 8, 2025, <https://github.com/marcelbuesing/socketcan-isotp/issues/2>
19. S32K unified bootloader demo send key - NXP Community, erişim tarihi Kasım 8, 2025, <https://community.nxp.com/t5/S32K/S32K-unified-bootloader-demo-send-key/m-p/1263050>
20. Calculating key from seed for UDS service 27(Security access) - Stack Overflow, erişim tarihi Kasım 8, 2025, <https://stackoverflow.com/questions/46538665/calculating-key-from-seed-for-uds-service-27security-access>
21. UDS Part - 10 | Security Access (0x27) | Unified Diagnostic Services - YouTube, erişim tarihi Kasım 8, 2025, <https://www.youtube.com/watch?v=EiaGCaJLbws>
22. ECU Reverse Engineering & Bypassing UDS SecurityAccess (0x27 ..., erişim tarihi Kasım 8, 2025, <https://medium.com/autosecurityy/ecu-reverse-engineering-bypassing-uds-securityaccess-0x27-9afd88771991>
23. Exploiting Diagnostic Protocol Vulnerabilities on Embedded Networks in Commercial Vehicles, erişim tarihi Kasım 8, 2025, <https://www.ndss-symposium.org/wp-content/uploads/vehiclesec2024-46-paper.pdf>
24. Security Evaluation of the Electronic Control Unit Software Update Process - DiVA portal, erişim tarihi Kasım 8, 2025, <http://www.diva-portal.org/smash/get/diva2:934083/FULLTEXT01.pdf>
25. From ECU to VSOC: UDS Security Monitoring StrategiesPreprint (arXiv). Accepted to SECURWARE 2025, The Nineteenth International Conference on Emerging Security Information, Systems and Technologies, Barcelona, Spain. Reference: https://www.thinkmind.org/library/SECURWARE/SECURWARE\_2025/securware\_2025\_1\_70\_30030.html, erişim tarihi Kasım 8, 2025, <https://arxiv.org/html/2510.25375v1>
26. ATM-T0033 Bypass UDS Security Access - Automotive Threat Matrix, erişim tarihi Kasım 8, 2025, <https://atm.automotiveisac.com/technique/ATM-TA0006/ATM-T0033>
27. Enhancing Security in UDS Seed-Key Exchange: Best Practices and Implementation Strategies | by Muhammet Kalaycı | Medium, erişim tarihi Kasım 8, 2025, <https://medium.com/@mkklyci/uds-seed-key-algorithm-764d5067d90c>
28. Security and Safety Considerations - Monolithic Power Systems, erişim tarihi Kasım 8, 2025, <https://www.monolithicpower.com/en/learning/mpscholar/battery-management-systems/bms-communication-interface/security-and-safety-considerations>
29. STRIDE model - Wikipedia, erişim tarihi Kasım 8, 2025, <https://en.wikipedia.org/wiki/STRIDE_model>
30. Threat Modeling Process - OWASP Foundation, erişim tarihi Kasım 8, 2025, <https://owasp.org/www-community/Threat_Modeling_Process>
31. Threat Modeling Methodology: STRIDE - IriusRisk, erişim tarihi Kasım 8, 2025, <https://www.iriusrisk.com/resources-blog/threat-modeling-methodology-stride>
32. Interacting with UDS — RAMN 1.0.0 documentation - Read the Docs, erişim tarihi Kasım 8, 2025, <https://ramn.readthedocs.io/en/latest/userguide/diag_tutorial.html>
33. Revolutionizing Security: The Impact of EV Fleet Management Solutions - SecuritySenses, erişim tarihi Kasım 8, 2025, <https://securitysenses.com/posts/revolutionizing-security-impact-ev-fleet-management-solutions>
34. Blind Wireless Seed Key Unlock | NMFTA, erişim tarihi Kasım 8, 2025, <https://nmfta.org/wp-content/media/2025/01/Blind-Wireless-Seed-Key-Unlock-Whitepaper-final.pdf>
35. Implementation of a Stateful Network Protocol Intrusion Detection Systems - SciTePress, erişim tarihi Kasım 8, 2025, <https://www.scitepress.org/Papers/2022/113274/113274.pdf>
36. How to use blockchain technology to trace database security incidents? - Tencent Cloud, erişim tarihi Kasım 8, 2025, <https://www.tencentcloud.com/techpedia/119267>
37. A Blockchain-Based Audit Trail Mechanism: Design and Implementation - MDPI, erişim tarihi Kasım 8, 2025, <https://www.mdpi.com/1999-4893/14/12/341>
38. Fuzz Testing on UDS over CAN - Presse-Blog, erişim tarihi Kasım 8, 2025, <https://www.presse-blog.com/2021/09/13/fuzz-testing-on-uds-over-can/>
39. Python-uds — python-uds documentation, erişim tarihi Kasım 8, 2025, <https://python-uds.readthedocs.io/en/latest/>
40. Examples — udsoncan 1.25.1 documentation, erişim tarihi Kasım 8, 2025, <https://udsoncan.readthedocs.io/en/latest/udsoncan/examples.html>
41. How to perform UDS session unlock using security dll from python program?, erişim tarihi Kasım 8, 2025, <https://stackoverflow.com/questions/67263096/how-to-perform-uds-session-unlock-using-security-dll-from-python-program>
42. Everything You Need to Know About Diagnostic Session Control (SID 0x10) in UDS Protocol, erişim tarihi Kasım 8, 2025, <https://pievcore.com/ideas/everything-you-need-to-know-about-diagnostic-session-control-sid-0x10-in-uds-protocol>
43. ForensicTransMonitor: A Comprehensive Blockchain Approach to Reinvent Digital Forensics and Evidence Management - MDPI, erişim tarihi Kasım 8, 2025, <https://www.mdpi.com/2078-2489/15/2/109>
44. Blockchain and Immutable Logging for Audit Integrity - LogZilla, erişim tarihi Kasım 8, 2025, <https://www.logzilla.net/blogs/blockchain-log-management-immutable-logging>
45. Can electric vehicles be hacked? | University of Nevada, Reno, erişim tarihi Kasım 8, 2025, <https://www.unr.edu/nevada-today/news/2023/atp-electric-cars-and-hacking>
46. Intrusion Detection in the Automotive Domain A Comprehensive Review - DTU Research Database, erişim tarihi Kasım 8, 2025, <https://orbit.dtu.dk/files/341095318/Intrusion_Detection_in_the_Automotive_Domain_A_Comprehensive_Review.pdf>
47. (PDF) Forensic Evidence Security System Using Blockchain Technology - ResearchGate, erişim tarihi Kasım 8, 2025, <https://www.researchgate.net/publication/373144792_Forensic_Evidence_Security_System_using_Blockchain_Technology>