NotebookLM - Tüm Yapay Zeka Kodlama ve Güvenlik Yetenekleri

0. Genel Tan■t■m ve Kullan■m Amac■

NotebookLM AI - Tüm Yetenek Seti
Bu belge, NotebookLM'e veya benzeri bir yapay zeka sistemine entegre edildi■inde a■a■■daki konularda aktif
- Yaz■l■m ve programlama dilleri (Python, C++, Java, JavaScript, Rust...)
- Web geli■tirme, mobil uygulama yap■m■
- Tersine mühendislik (decompile, analiz, patchleme)
- Exploit geli■tirme, shellcode ve kernel analizi
- Web ve mobil güvenlik aç■klar■ (XSS, SQLi, CSRF, MASVS)
- Yapay zeka güvenli■i (prompt injection, adversarial examples)
- Görsel, video ve animasyon analizi
- Metinden görsele üretim (Stable Diffusion, DALL·E)
- En geli■mi■ zafiyet tespit algoritmalar■ (AI destekli + klasik yöntemler)

Bu yapay zeka, hem analiz hem de örnek kod üretimi yaparak, komutlara dayal■ teknik ç■kt■lar sa■layacak ■e

1. Popüler Kodlama Dilleri ve Kapsamlar■

- Python (otomasyon, veri analizi, AI)
- JavaScript (web geli■tirme, Node.js)
- Java (Android geli■tirme)
- C / C++ (sistem yaz■l■mlar■, gömülü sistemler)
- C# (Unity, masaüstü uygulamalar■)
- PHP (web backend)
- Go (yüksek performansl■ backend)
- Rust (sistemsel güvenlik ve performans)

2. Web Geli■tirme Yetenekleri

- HTML, CSS, JavaScript
- React, Vue.js, Angular
- Tailwind CSS, Bootstrap
- REST API / GraphQL

3. Veri Bilimi ve Yapay Zeka

- NumPy, Pandas
- Scikit-learn, TensorFlow, PyTorch
- Matplotlib, Seaborn

4. Veritaban■ Bilgisi

- MySQL, PostgreSQL, SQLite
- MongoDB, Redis

5. Mobil Geli■tirme

- Flutter (Dart)
- React Native
- Java (Android), Swift (iOS)

6. Otomasyon & Betikler

- Python betikleri
- Bash Shell scriptleri
- PowerShell

7. Yard■mc■ Araçlar ve Sistem Bilgisi

- Git & GitHub
- Docker & Kubernetes (temel)
- Linux komutlar■

8. NotebookLM Yetenek Geni■letme Format■

- Bilgi öbekleri (prompt taban■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zaman■ sorgulama için d■■ veri i■leyici betikler (opsiyonel olarak)

9. Geni■ Kapsaml■ Tersine Mühendislik Yetenekleri ve Kod Örnekleri

- **APK Decompile**:
  * Araçlar: apktool, jadx, dex2jar
  * Örnek kullan■m:
    ```bash
    apktool d app.apk -o decoded_app
    jadx -d src app.apk
    ```

- **Binary Analizi**:
  * Araçlar: Ghidra, IDA Pro, Radare2
  * Ghidra örnek analiz ak■■■:
    - Binary içe aktar■l■r
    - Otomatik disassembler çal■■t■r■l■r
    - String ve fonksiyon listesi analiz edilir

- **Dinamik Analiz (Runtime Hooking)**:
  * Araçlar: Frida, Xposed, Burp Suite
  * Örnek Frida script:
    ```js
    Java.perform(function() {
      var cls = Java.use('com.target.app.MainActivity');
      cls.secretMethod.implementation = function() {
        console.log('Hooked secretMethod!');
        return this.secretMethod();
      }
    });
    ```

- **Anti-Debug ve Anti-Tamper Atlatma**:
  * Teknikler: ptrace bypass, checksum fix
  * Python ptrace bypass örne■i:
    ```python
    import ctypes
    libc = ctypes.CDLL('libc.so.6')
    libc.ptrace.restype = ctypes.c_long
    if libc.ptrace(0, 0, None, None) == -1:
        print('Anti-debug aktif!')
    else:
        print('Debug ortam■ temiz.')
    ```

```

- **Patchleme ve Crackleme Teknikleri**:
  * Binary içindeki string/method patchleme (hex editör, python)
  * Lisans kontrollerinin kald■r■lmas■

## 10. Exploit Geli■tirme, Shellcode ve Kernel Debugging

- **Buffer Overflow Temelleri**:
  * Exploit yap■s■: NOP sled + shellcode + EIP overwrite
  * Python örnek exploit:
    ```python
    buffer = b"A" * 2606 + b"\x8f\x35\x4a\x5f" + b"\x90" * 16
    shellcode = b"\xcc" * 100  # örnek (int3 breakpoint)
    payload = buffer + shellcode
    ```

- **Shellcode Yaz■m■ (Linux x86)**:
  * Basit shell açma shellcode:
    ```asm
    xor eax, eax
    push eax
    push 0x68732f2f
    push 0x6e69622f
    mov ebx, esp
    push eax
    push ebx
    mov ecx, esp
    mov al, 11
    int 0x80
    ```

- **ROP (Return-Oriented Programming)**:
  * Zincirleme gadget'lar ile bypass
  * ROPgadget arac■ ile analiz:
    ```bash
    ROPgadget --binary vulnerable | grep 'pop'
    ```

- **Kernel Debugging (Linux)**:
  * Araçlar: `gdb`, `qemu`, `crash`, `dmesg`
  * Kernel panik analizi örne■i:
    ```bash
    dmesg | grep -i panic
    gdb vmlinux /proc/kcore
    ```

- **Exploit Geli■tirme Süreci**:
  * Hedef tespit → Vulnerability fuzzing → Exploit PoC → Privilege escalation
  * Metasploit Framework modül örne■i geli■tirme
    ```

11. Web Uygulama Güvenlik Aç■klar■ ve Exploit Teknikleri

- **XSS (Cross Site Scripting)**:
  * Reflected, Stored ve DOM tabanl■ türler
  * Örnek payload:
    ```html
    <script>alert('XSS')</script>
    ```
- **SQL Injection**:
  * Union-based, Error-based, Blind
  * Örnek:
    ```sql
    ' OR 1=1 --
    ```
- **CSRF (Cross Site Request Forgery)**:
  * Kullan■c■n■n taray■c■s■ üzerinden sahte istek gönderme
  * Koruma: CSRF token
- **File Upload Bypass**:
  * ■çerik tipi ve uzant■ kontrollerinin atlat■lmas■

12. Mobil Güvenlik ve Tersine Mühendislik

- **Android Güvenlik Testleri**:
  * Apktool, jadx, Frida ile analiz
  * AndroidManifest.xml'de izin analizi
- **iOS Güvenlik Testleri**:
  * Jailbreak sonras■ uygulama içi dinleme (cycript, frida)
- **OWASP MASVS Kapsam■**:
  * Uygulama kod güvenli■i
  * Veri güvenli■i
  * A■ ileti■imi güvenli■i

13. Yapay Zeka Güvenli■i ve Sald■r■ Senaryolar■

- **Adversarial Examples**:
  * Küçük görsel de■i■ikliklerle modelin yan■lmas■na sebep olunabilir
  * Örnek kod (FGSM sald■r■s■):
    ```python
    perturbed_image = image + epsilon * image.grad.sign()
    ```
- **Model Ezi■i (Model Stealing)**:
  * Sorgu-temelli yan■tlarla bir modeli yeniden e■itme
- **Prompt Injection**:
  * LLM’lere özel komut enjekte etme sald■r■lar■
- **Veri Seti Zehirleme (Data Poisoning)**:
  * E■itim verisine zarar■ örnekler eklenerek model manipüle edilir

14. Görsel ve Video Analizi, Metinden Görsele ve Görüntü Tabanl■ Yapay Zeka Yetenekleri

- **Görsel (Image) Analizi**:
  * Resim s■n■fland■rma: CNN (Convolutional Neural Network)
  * Nesne tespiti: YOLOv8, Detectron2, OpenCV
  * OCR: Tesseract ile metin tan■ma

```python
import pytesseract
text = pytesseract.image_to_string('resim.jpg')
```

- **Video Analizi**:
  * Hareket alg■lama, yüz tan■ma, nesne takibi (OpenCV, Mediapipe)
  * Frame bazl■ analiz ve özet ç■karma
```python
import cv2
cap = cv2.VideoCapture('video.mp4')
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    # analiz yap■l■r
```

- **Metinden Görsele Üretim (Text-to-Image)**:
  * Diffusion modelleri (Stable Diffusion, DALL·E)
  * Örnek prompt: "Bir ormanda k■l■ç ta■■yan bir aslan, sinematik ■■■kland■rma"

- **Görselden Görsel Üretim (Image-to-Image)**:
  * Stil aktar■m■, restorasyon, yüz de■i■tirme

- **Foto■raftan Animasyona (Image-to-Animation)**:
  * Canland■rma: DeepMotion, D-ID, Kaiber AI
  * Yüz hareketi canland■rmas■:
```bash
python animate.py --input face.jpg --motion driving.mp4
```

15. Geli■mi■ Güvenlik Aç■klar■ ve Algoritmik ■stismar Teknikleri

- **Heap Exploitation (Heap Feng Shui, Use-After-Free, Tcache Poisoning)**:
  * malloc/free dengesizliklerini kullanarak bellek kontrolü ele geçirilir
  * Libc leak → GOT overwrite → shell eri■imi
  * Örnek glibc tcache exploit senaryosu

- **Race Condition (Yar■■ Durumu Sald■r■lar■)**:
  * Ayn■ kayna■a ayn■ anda eri■im sonucu olu■an aç■klardan faydalanma
  * Örnek (Linux): symlink race

- **Format String Exploits**:
  * printf() gibi i■levlerde kullan■c■ girdisinin kontrolsüz i■lenmesi
  * EIP/RIP overwrite yap■labilir
```c
printf(user_input); // tehlikeli!
```

- **Advanced ROP + JOP (Jump-Oriented Programming)**:

  * NX bit bypass teknikleri
  * ROP zincirlerine syscall gadget’lar■ ekleyerek full shell açma

- **Side-Channel Attacks (Zaman, Cache, Güç Tabanl■)**:
  * Spectre / Meltdown gibi mimari aç■klar
  * Zaman farklar■ ile ■ifre tahmini

- **Symbolic Execution & Fuzzing (KLEE, AFL++)**:
  * Program yolunu otomatik analiz ederek mant■k hatalar■ ve zafiyetler tespit edilir
  * Fuzzing örne■i:
    ```bash
    afl-fuzz -i inputs -o outputs ./vulnerable_binary @@
    ```

15. Geli■mi■ Güvenlik Aç■■■ Tespit Algoritmalar■ ve Yöntemleri

- **Statik Kod Analizi Algoritmalar■**:
  * AST (Abstract Syntax Tree) analizi
  * Taint Analysis: Girdi izleme üzerinden güvenlik aç■■■ analizi
  * Örnek araçlar: SonarQube, Semgrep, Bandit

- **Dinamik Güvenlik Testi (DAST) Algoritmalar■**:
  * Taray■c■ emülasyonu + davran■■ analizi
  * Örnek: OWASP ZAP, Burp Suite Active Scan

- **Fuzzing Teknikleri**:
  * Mutasyon tabanl■: Mevcut input'lar■ rastgele de■i■tirerek çal■■t■rmak
  * Generation tabanl■: Protokol bilgisine göre yeni input üretimi
  * AFL, LibFuzzer, Honggfuzz örnek araçlar

- **Makine Ö■renimi Tabanl■ Tespit Yöntemleri**:
  * Anomali tespiti (Isolation Forest, OneClassSVM)
  * S■n■fland■rma (Random Forest, XGBoost, Deep Learning)
  * Örnek Python kod:
    ```python
    from sklearn.ensemble import IsolationForest
    model = IsolationForest()
    model.fit(training_data)
    preds = model.predict(test_data)
    ```

- **Yapay Zeka Destekli S■zma Testi**:
  * AI destekli payload üretimi (LLM ile)
  * Otomatik zafiyet e■le■tirme (CVE veritaban■na kar■■)

- **Graf Tabanl■ Güvenlik Modelleme**:
  * Attack graph’ler ve eri■im denetimi zincirleri
  * Neo4j ve Cypher ile ili■ki analizleri yap■labilir