

NotebookLM - Ultimate AI Görsel, Kodlama ve Güvenlik Yetenekleri

1. Popüler Kodlama Dilleri ve Kapsamlar

- Python (otomasyon, veri analizi, AI)
- JavaScript (web geliştirme, Node.js)
- Java (Android geliştirme)
- C / C++ (sistem yazılımlar, gömülü sistemler)
- C# (Unity, masaüstü uygulamalar)
- PHP (web backend)
- Go (yüksek performanslı backend)
- Rust (sistemsel güvenlik ve performans)

2. Web Geliştirme 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. Veritabanı Bilgisi

- MySQL, PostgreSQL, SQLite
- MongoDB, Redis

5. Mobil Geliştirme

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

6. Otomasyon & Betikler

- Python betikleri
- Bash Shell scriptleri
- PowerShell

7. Yardımcı Araçlar ve Sistem Bilgisi

- Git & GitHub
- Docker & Kubernetes (temel)
- Linux komutları

8. NotebookLM Yetenek Geniştirme Formatları

- Bilgi öbekleri (prompt tabanlı eğitim)
- Kod yorumlama ve analiz yeteneği
- Hata ayıklama (debug) algoritmaları
- Gerçek zamanlı sorgulama için döküman veri işleyici betikler (opsiyonel olarak)

9. Geni Kapsamlı Tersine Mühendislik Yetenekleri ve Kod Örnekleri

- **APK Decompile**:
 - * Araçlar: apktool, jadx, dex2jar
 - * Örnek kullanım:

```
```bash
apktool d app.apk -o decoded_app
jadx -d src app.apk
```

...

- **\*\*Binary Analizi\*\*:**

- \* Araçlar: Ghidra, IDA Pro, Radare2
- \* Ghidra örnek analiz akışı:
  - Binary içe aktarılır
  - Otomatik disassembler çalıştırılır
  - 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 örneği:

```
```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 ortamı temiz.')
```
```

- **\*\*Patchleme ve Crackleme Teknikleri\*\*:**

- \* Binary içindeki string/method patchleme (hex editör, python)
- \* Lisans kontrollerinin kaldırılması

## 10. Exploit Geliştirme, Shellcode ve Kernel Debugging

- **\*\*Buffer Overflow Temelleri\*\*:**

- \* Exploit yapısı: 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 Yazımı (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 tabanlı 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\*\*:

- \* İçerik tipi ve uzantı kontrollerinin atlatılması

## 12. Mobil Güvenlik ve Tersine Mühendislik

- **Android Güvenlik Testleri**:
  - \* Apktool, jadx, Frida ile analiz
  - \* AndroidManifest.xml'de izin analizi
- **iOS Güvenlik Testleri**:
  - \* Jailbreak sonrası uygulama içi dinleme (cycrypt, frida)
- **OWASP MASVS Kapsamı**:
  - \* Uygulama kod güvenliği
  - \* Veri güvenliği
  - \* API iletişimi güvenliği

## 13. Yapay Zeka Güvenliği ve Saldırı Senaryoları

- **Adversarial Examples**:
  - \* Küçük görsel değişikliklerle modelin yanıltılmasına sebep olunabilir
  - \* Örnek kod (FGSM saldırısı):

```
python
perturbed_image = image + epsilon * image.grad.sign()

```
- **Model Eziği (Model Stealing)**:
  - \* Sorgu-temelli yanıtlarla bir modeli yeniden eğitme
- **Prompt Injection**:
  - \* LLM'lere özel komut enjekte etme saldırıları
- **Veri Seti Zehirlenme (Data Poisoning)**:
  - \* Eğitim verisine zararlı örnekler eklenerek model manipüle edilir

## 14. Görsel ve Video Analizi, Metinden Görsele ve Görüntü Tabanlı Yapay Zeka Yetenekleri

- **Görsel (Image) Analizi**:
  - \* Resim sınıflandırma: CNN (Convolutional Neural Network)
  - \* Nesne tespiti: YOLOv8, Detectron2, OpenCV
  - \* OCR: Tesseract ile metin tanıma

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

```
- **Video Analizi**:
  - \* Hareket algılama, yüz tanıma, nesne takibi (OpenCV, Mediapipe)
  - \* Frame bazlı analiz ve özet çıkarma

```
python
import cv2
cap = cv2.VideoCapture('video.mp4')
while cap.isOpened():
 ret, frame = cap.read()
 if not ret:
 break
 # analiz yapılır

```

- **\*\*Metinden Görsele Üretim (Text-to-Image)\*\*:**
  - \* Diffusion modelleri (Stable Diffusion, DALL·E)
  - \* Örnek prompt: "Bir ormanda kükürcü taştan bir aslan, sinematik klandırma"
- **\*\*Görselden Görsel Üretim (Image-to-Image)\*\*:**
  - \* Stil aktarm, restorasyon, yüz değiştirme
- **\*\*Fotoğraftan Animasyona (Image-to-Animation)\*\*:**
  - \* Canlandırma: DeepMotion, D-ID, Kaiber AI
  - \* Yüz hareketi canlandırması:

```
```bash
python animate.py --input face.jpg --motion driving.mp4
```
```