# NotebookLM - Yaz■l■m & Kodlama Yetenekleri

## 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 tabanl■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zamanl■ sorgulama için d■■ veri i■leyici betikler (opsiyonel olarak)

# NotebookLM - Yaz■l■m, Kodlama ve Tersine Mühendislik (Geni■ Kapsaml■)

## 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 tabanl■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zamanl■ 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■
    ```

# NotebookLM - Yaz■l■m, Kodlama, Tersine Mühendislik ve Exploit Geli■tirme

### 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 tabanl■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zamanl■ 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

# NotebookLM Ultimate - Yaz■l■m, Güvenlik ve Tersine Mühendislik

## 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 tabanl■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zamanl■ 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 zararl■ örnekler eklenerek model manipüle edilir

# NotebookLM - Ultimate AI Görsel, Kodlama ve Güvenlik Yetenekleri

## 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 tabanl■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zamanl■ 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 zararl■ 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
    ```

# NotebookLM AI + Güvenlik: Ultimate Kodlama ve Algoritmalar

## 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 zararl■ 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

# NotebookLM - Yapay Zeka Destekli Bahis Algoritmalar■ Toplulu■u

### 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 tabanl■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zamanl■ 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 zararl■ 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

## 16. Geli■mi■ Bahis Analiz ve Tahmin Algoritmalar■ Toplulu■u

- **Genel Tan■m**:
  Bu algoritma toplulu■u, bahis sistemlerinde yasal s■n■rlar dahilinde çal■■arak yüksek do■rulukla tahminlerd
  Ana hedef, yapay zeka destekli veri toplama, analiz etme ve olas■l■k tabanl■ karar üretmektir.

- **Veri Toplama Kaynaklar■**:
  * Spor API'leri (Soccer API, Odds API, RapidAPI)
  * Resmi maç istatistik portallar■ (FIFA, UEFA, NBA)
  * Bahis sitelerinin aç■k verileri (JSON/XML)
  * Sosyal medya ve haber kaynaklar■ndan oyuncu form analizleri

- **Algoritma Yap■s■**:
  1. **Veri Toplay■c■**: API ve haber kaynaklar■ndan veri çeker
  2. **Ön ■■leyici**: Verileri normalize eder, eksikleri giderir
  3. **Model E■itici**: ML algoritmalar■ ile geçmi■ veriden ö■renir

4. **Skorlay■c■**: Yeni maç için olas■l■k puanlar■ üretir
5. **Sesli Yorumlay■c■ (Opsiyonel)**: Komutla konu■ur, tahminleri aç■klar

- **Kullan■lan Algoritmalar**:
  * Random Forest / XGBoost (skor ve galibiyet tahmini)
  * Logistic Regression (oran + oyuncu performans■ e■le■mesi)
  * LSTM (zaman serisi form ve sakatl■k analizi)
  * Ensemble Voting (birden çok modelin a■■rl■kl■ ortalamas■)

- **Veri ■■leme ve Özellik Mühendisli■i**:
  * Sakat/cezal■ oyuncu etkisi
  * Savunma/forvet istatistiklerinin çak■■t■r■lmas■
  * Teknik direktör ve saha ko■ullar■n■n skora etkisi

- **Etik Kurallar ve S■n■rlamalar**:
  * Hiçbir ■ekilde sistemsel izinsiz eri■im yap■lmaz
  * Tahminler %100 kesinlik garantisi vermez, istatistiksel desteklidir
  * Kullan■c■, verileri kendi kaynaklar■ndan temin etmekle yükümlüdür

# NotebookLM - Wi-Fi Cihaz Kontrol & Eri■im S■n■rland■rma Algoritmalar■

### 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 tabanl■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zamanl■ 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 zararl■ 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

## 16. Geli■mi■ Bahis Analiz ve Tahmin Algoritmalar■ Toplulu■u

- **Genel Tan■m**:
  Bu algoritma toplulu■u, bahis sistemlerinde yasal s■n■rlar dahilinde çal■■arak yüksek do■rulukla tahminlerd
  Ana hedef, yapay zeka destekli veri toplama, analiz etme ve olas■l■k tabanl■ karar üretmektir.

- **Veri Toplama Kaynaklar■**:
  * Spor API'leri (Soccer API, Odds API, RapidAPI)
  * Resmi maç istatistik portallar■ (FIFA, UEFA, NBA)
  * Bahis sitelerinin aç■k verileri (JSON/XML)
  * Sosyal medya ve haber kaynaklar■ndan oyuncu form analizleri

- **Algoritma Yap■s■**:
  1. **Veri Toplay■c■**: API ve haber kaynaklar■ndan veri çeker
  2. **Ön ■■leyici**: Verileri normalize eder, eksikleri giderir
  3. **Model E■itici**: ML algoritmalar■ ile geçmi■ veriden ö■renir

4. **Skorlay■c■**: Yeni maç için olas■l■k puanlar■ üretir
5. **Sesli Yorumlay■c■ (Opsiyonel)**: Komutla konu■ur, tahminleri aç■klar

- **Kullan■lan Algoritmalar**:
  * Random Forest / XGBoost (skor ve galibiyet tahmini)
  * Logistic Regression (oran + oyuncu performans■ e■le■mesi)
  * LSTM (zaman serisi form ve sakatl■k analizi)
  * Ensemble Voting (birden çok modelin a■■rl■kl■ ortalamas■)

- **Veri ■■leme ve Özellik Mühendisli■i**:
  * Sakat/cezal■ oyuncu etkisi
  * Savunma/forvet istatistiklerinin çak■■t■r■lmas■
  * Teknik direktör ve saha ko■ullar■n■n skora etkisi

- **Etik Kurallar ve S■n■rlamalar**:
  * Hiçbir ■ekilde sistemsel izinsiz eri■im yap■lmaz
  * Tahminler %100 kesinlik garantisi vermez, istatistiksel desteklidir
  * Kullan■c■, verileri kendi kaynaklar■ndan temin etmekle yükümlüdür

## 17. Wi-Fi A■■nda Cihaz Kontrolü, Trafik Analizi ve Eri■im S■n■rland■rma Algoritmalar■

- **Cihaz Ke■if Algoritmalar■**:
  * A■ tarama araçlar■: `nmap`, `arp-scan`, `fing`
  * MAC/IP e■le■tirme tablosu ile cihaz s■n■fland■rmas■
    ```bash
    sudo nmap -sn 192.168.1.0/24
    arp -a
    ```

- **Trafik Analizi**:
  * Paket dinleme: `Wireshark`, `tcpdump`, `tshark`
  * Python + Scapy ile özel trafik yakalama:
    ```python
    from scapy.all import sniff
    sniff(prn=lambda x: x.summary(), count=10)
    ```

- **Eri■im S■n■rland■rma Yöntemleri**:
  * Zaman bazl■ k■s■tlama (Router parental controls)
  * IP/MAC filtreleme (whitelist/blacklist)
  * OpenWRT ile port ve zaman bazl■ s■n■rlama

- **Anomali Tespiti ve Uyar■ Sistemleri**:
  * AI tabanl■ trafik analizi (IsolationForest, DBSCAN)
  * ■üpheli cihaz eri■im takibi ve otomatik loglama

- **Eri■im Loglama ve Görselle■tirme**:
  * Her cihaza özel ba■lant■ geçmi■i tutulur
  * Grafana/Prometheus ile ba■lant■ istatistik raporu ç■kar■l■r

- **Uygulama Senaryosu (Legal)**:

* Ev a■■nda çocuklar için saat bazl■ k■s■tlama
* Ofiste sadece yetkili cihazlara port 22 eri■imi
* ■üpheli MAC adreslerini tespit edip uyar■ gönderme

# NotebookLM - Geli■mi■ Android Yapay Zekâ Asistan■ (OneKanki Altyap■s■)

### 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 tabanl■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zamanl■ 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 zararl■ 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

## 16. Geli■mi■ Bahis Analiz ve Tahmin Algoritmalar■ Toplulu■u

- **Genel Tan■m**:
  Bu algoritma toplulu■u, bahis sistemlerinde yasal s■n■rlar dahilinde çal■■arak yüksek do■rulukla tahminlerd
  Ana hedef, yapay zeka destekli veri toplama, analiz etme ve olas■l■k tabanl■ karar üretmektir.

- **Veri Toplama Kaynaklar■**:
  * Spor API'leri (Soccer API, Odds API, RapidAPI)
  * Resmi maç istatistik portallar■ (FIFA, UEFA, NBA)
  * Bahis sitelerinin aç■k verileri (JSON/XML)
  * Sosyal medya ve haber kaynaklar■ndan oyuncu form analizleri

- **Algoritma Yap■s■**:
  1. **Veri Toplay■c■**: API ve haber kaynaklar■ndan veri çeker
  2. **Ön ■■leyici**: Verileri normalize eder, eksikleri giderir
  3. **Model E■itici**: ML algoritmalar■ ile geçmi■ veriden ö■renir

4. **Skorlay■c■**: Yeni maç için olas■l■k puanlar■ üretir
5. **Sesli Yorumlay■c■ (Opsiyonel)**: Komutla konu■ur, tahminleri aç■klar

- **Kullan■lan Algoritmalar**:
  * Random Forest / XGBoost (skor ve galibiyet tahmini)
  * Logistic Regression (oran + oyuncu performans■ e■le■mesi)
  * LSTM (zaman serisi form ve sakatl■k analizi)
  * Ensemble Voting (birden çok modelin a■■rl■kl■ ortalamas■)

- **Veri ■■leme ve Özellik Mühendisli■i**:
  * Sakat/cezal■ oyuncu etkisi
  * Savunma/forvet istatistiklerinin çak■■t■r■lmas■
  * Teknik direktör ve saha ko■ullar■n■n skora etkisi

- **Etik Kurallar ve S■n■rlamalar**:
  * Hiçbir ■ekilde sistemsel izinsiz eri■im yap■lmaz
  * Tahminler %100 kesinlik garantisi vermez, istatistiksel desteklidir
  * Kullan■c■, verileri kendi kaynaklar■ndan temin etmekle yükümlüdür

## 17. Wi-Fi A■■nda Cihaz Kontrolü, Trafik Analizi ve Eri■im S■n■rland■rma Algoritmalar■

- **Cihaz Ke■if Algoritmalar■**:
  * A■ tarama araçlar■: `nmap`, `arp-scan`, `fing`
  * MAC/IP e■le■tirme tablosu ile cihaz s■n■fland■rmas■
    ```bash
    sudo nmap -sn 192.168.1.0/24
    arp -a
    ```

- **Trafik Analizi**:
  * Paket dinleme: `Wireshark`, `tcpdump`, `tshark`
  * Python + Scapy ile özel trafik yakalama:
    ```python
    from scapy.all import sniff
    sniff(prn=lambda x: x.summary(), count=10)
    ```

- **Eri■im S■n■rland■rma Yöntemleri**:
  * Zaman bazl■ k■s■tlama (Router parental controls)
  * IP/MAC filtreleme (whitelist/blacklist)
  * OpenWRT ile port ve zaman bazl■ s■n■rlama

- **Anomali Tespiti ve Uyar■ Sistemleri**:
  * AI tabanl■ trafik analizi (IsolationForest, DBSCAN)
  * ■üpheli cihaz eri■im takibi ve otomatik loglama

- **Eri■im Loglama ve Görselle■tirme**:
  * Her cihaza özel ba■lant■ geçmi■i tutulur
  * Grafana/Prometheus ile ba■lant■ istatistik raporu ç■kar■l■r

- **Uygulama Senaryosu (Legal)**:

* Ev a■■nda çocuklar için saat bazl■ k■s■tlama
* Ofiste sadece yetkili cihazlara port 22 eri■imi
* ■üpheli MAC adreslerini tespit edip uyar■ gönderme

## 18. Geli■mi■ Android Yapay Zekâ Asistan■ - OneKanki Altyap■s■

- **Genel Amaç**:
  Bu sistem, kullan■c■n■n tam kontrolüyle çal■■an, sürekli sesli komut alg■layan, internette do■rulama yapabilen, dosya-indirme ve güvenlik izleme görevleri üstlenen bir yapay zekâ asistan■d■r.

- **Sesli Komut Alg■lama**:
  * Wake Word ile ba■latma: örn. “Hey Kanki”
  * Python modülleri: `speech_recognition`, `pyaudio`, `vosk` (offline destekli)
    ```python
    import speech_recognition as sr
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Dinleniyor...")
        audio = r.listen(source)
        command = r.recognize_google(audio, language='tr-TR')
    ```

- **Konu■madan Arama ve Sesli Yan■t**:
  * `DuckDuckGo` veya `Google Custom Search API` ile sorgulama
  * `pyttsx3` veya `gTTS` ile sesli okuma

- **Dosya Açma & APK Kurma**:
  * Termux destekli `pm install` ile APK yükleme
    ```bash
    pm install /sdcard/Download/uygulama.apk
    ```

- **Risk ve Virüs Taramas■**:
  * `virustotal` API ile dosya hash sorgulama
  * Alternatif: `clamav` terminal virüs taray■c■

- **Eri■im Yetkileri & ■zinler**:
  * AndroidManifest.xml'de izin tan■mlama
  * Kullan■c■ yaln■zca izin verdi■i i■lemlere onay verir

- **Kendi Kendini Geli■tirme (Ö■renen Sistem)**:
  * Kullan■c■n■n önceki komutlar■n■ haf■zaya alma (`notebook.json`)
  * Kullan■c■n■n davran■■■na göre yeni önerilerde bulunma
  * Basit örnek:
    ```python
    memory.append({"komut": komut, "tarih": datetime.now().isoformat()})
    ```

- **Opsiyonel Geni■letmeler**:
  * Telegram üzerinden sesli/komutlu kontrol

* Android üzeri GUI entegrasyonu
* QR ile cihazlar aras■ komut al■■veri■i

# NotebookLM - Offline APK Geli■tirme ve Yapay Zeka Algoritmalar■

## 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 tabanl■ e■itim)
- Kod yorumlama ve analiz yetene■i
- Hata ay■klama (debug) algoritmalar■
- Gerçek zamanl■ 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 zararl■ 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

## 16. Geli■mi■ Bahis Analiz ve Tahmin Algoritmalar■ Toplulu■u

- **Genel Tan■m**:
  Bu algoritma toplulu■u, bahis sistemlerinde yasal s■n■rlar dahilinde çal■■arak yüksek do■rulukla tahminlerd
  Ana hedef, yapay zeka destekli veri toplama, analiz etme ve olas■l■k tabanl■ karar üretmektir.

- **Veri Toplama Kaynaklar■**:
  * Spor API'leri (Soccer API, Odds API, RapidAPI)
  * Resmi maç istatistik portallar■ (FIFA, UEFA, NBA)
  * Bahis sitelerinin aç■k verileri (JSON/XML)
  * Sosyal medya ve haber kaynaklar■ndan oyuncu form analizleri

- **Algoritma Yap■s■**:
  1. **Veri Toplay■c■**: API ve haber kaynaklar■ndan veri çeker
  2. **Ön ■■leyici**: Verileri normalize eder, eksikleri giderir
  3. **Model E■itici**: ML algoritmalar■ ile geçmi■ veriden ö■renir

4. **Skorlay■c■**: Yeni maç için olas■l■k puanlar■ üretir
5. **Sesli Yorumlay■c■ (Opsiyonel)**: Komutla konu■ur, tahminleri aç■klar

- **Kullan■lan Algoritmalar**:
  * Random Forest / XGBoost (skor ve galibiyet tahmini)
  * Logistic Regression (oran + oyuncu performans■ e■le■mesi)
  * LSTM (zaman serisi form ve sakatl■k analizi)
  * Ensemble Voting (birden çok modelin a■■rl■kl■ ortalamas■)

- **Veri ■■leme ve Özellik Mühendisli■i**:
  * Sakat/cezal■ oyuncu etkisi
  * Savunma/forvet istatistiklerinin çak■■t■r■lmas■
  * Teknik direktör ve saha ko■ullar■n■n skora etkisi

- **Etik Kurallar ve S■n■rlamalar**:
  * Hiçbir ■ekilde sistemsel izinsiz eri■im yap■lmaz
  * Tahminler %100 kesinlik garantisi vermez, istatistiksel desteklidir
  * Kullan■c■, verileri kendi kaynaklar■ndan temin etmekle yükümlüdür

## 17. Wi-Fi A■■nda Cihaz Kontrolü, Trafik Analizi ve Eri■im S■n■rland■rma Algoritmalar■

- **Cihaz Ke■if Algoritmalar■**:
  * A■ tarama araçlar■: `nmap`, `arp-scan`, `fing`
  * MAC/IP e■le■tirme tablosu ile cihaz s■n■fland■rmas■
    ```bash
    sudo nmap -sn 192.168.1.0/24
    arp -a
    ```

- **Trafik Analizi**:
  * Paket dinleme: `Wireshark`, `tcpdump`, `tshark`
  * Python + Scapy ile özel trafik yakalama:
    ```python
    from scapy.all import sniff
    sniff(prn=lambda x: x.summary(), count=10)
    ```

- **Eri■im S■n■rland■rma Yöntemleri**:
  * Zaman bazl■ k■s■tlama (Router parental controls)
  * IP/MAC filtreleme (whitelist/blacklist)
  * OpenWRT ile port ve zaman bazl■ s■n■rlama

- **Anomali Tespiti ve Uyar■ Sistemleri**:
  * AI tabanl■ trafik analizi (IsolationForest, DBSCAN)
  * ■üpheli cihaz eri■im takibi ve otomatik loglama

- **Eri■im Loglama ve Görselle■tirme**:
  * Her cihaza özel ba■lant■ geçmi■i tutulur
  * Grafana/Prometheus ile ba■lant■ istatistik raporu ç■kar■l■r

- **Uygulama Senaryosu (Legal)**:

* Ev a■■nda çocuklar için saat bazl■ k■s■tlama
* Ofiste sadece yetkili cihazlara port 22 eri■imi
* ■üpheli MAC adreslerini tespit edip uyar■ gönderme

## 18. Geli■mi■ Android Yapay Zekâ Asistan■ - OneKanki Altyap■s■

- **Genel Amaç**:
  Bu sistem, kullan■c■n■n tam kontrolüyle çal■■an, sürekli sesli komut alg■layan, internette do■rulama yapabilen, dosya-indirme ve güvenlik izleme görevleri üstlenen bir yapay zekâ asistan■d■r.

- **Sesli Komut Alg■lama**:
  * Wake Word ile ba■latma: örn. “Hey Kanki”
  * Python modülleri: `speech_recognition`, `pyaudio`, `vosk` (offline destekli)
    ```python
    import speech_recognition as sr
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Dinleniyor...")
        audio = r.listen(source)
        command = r.recognize_google(audio, language='tr-TR')
    ```

- **Konu■madan Arama ve Sesli Yan■t**:
  * `DuckDuckGo` veya `Google Custom Search API` ile sorgulama
  * `pyttsx3` veya `gTTS` ile sesli okuma

- **Dosya Açma & APK Kurma**:
  * Termux destekli `pm install` ile APK yükleme
    ```bash
    pm install /sdcard/Download/uygulama.apk
    ```

- **Risk ve Virüs Taramas■**:
  * `virustotal` API ile dosya hash sorgulama
  * Alternatif: `clamav` terminal virüs taray■c■

- **Eri■im Yetkileri & ■zinler**:
  * AndroidManifest.xml'de izin tan■mlama
  * Kullan■c■ yaln■zca izin verdi■i i■lemlere onay verir

- **Kendi Kendini Geli■tirme (Ö■renen Sistem)**:
  * Kullan■c■n■n önceki komutlar■n■ haf■zaya alma (`notebook.json`)
  * Kullan■c■n■n davran■■■na göre yeni önerilerde bulunma
  * Basit örnek:
    ```python
    memory.append({"komut": komut, "tarih": datetime.now().isoformat()})
    ```

- **Opsiyonel Geni■letmeler**:
  * Telegram üzerinden sesli/komutlu kontrol

* Android üzeri GUI entegrasyonu
* QR ile cihazlar aras■ komut al■■veri■i

### 19. NotebookLM APK Geli■tirme (Tamamen Offline ve Cihaz Üzerinde)

- **Amaç**:
  ■nternete ba■l■ olmadan ve d■■ cihazdan yard■m almadan Android üzerinde çal■■an NotebookLM benzeri

- **Geli■tirme Ortam■**:
  * Android + Termux + proot-distro + Python 3
  * Ek: Vosk (offline ses tan■ma), transformers (yerel LLM), sqlite (haf■za)

- **Ana Bile■enler**:
  1. **LLM (Dil modeli)**: GGML tabanl■ küçük modeller (Mistral, Phi2, TinyLlama)
  2. **Veri Taban■**: sqlite tabanl■ kullan■c■ komut geçmi■i ve haf■za
  3. **Sorgu Motoru**: Komut analiz, e■le■tir, cevap üret
  4. **Ses Tan■ma**: vosk-offline ile kelime alg■lama
  5. **TTS**: Coqui TTS veya Piper (offline metinden ses)
  6. **Android Arayüzü (opsiyonel)**: Termux:API veya Kivy (grafik UI)

- **Kritik Kod Parçalar■**:
  * Sorgu e■le■tirme ve haf■zaya kaydetme:
    ```python
    import sqlite3
    conn = sqlite3.connect('hafiza.db')
    conn.execute("CREATE TABLE IF NOT EXISTS komutlar (soru TEXT, cevap TEXT)")
    conn.execute("INSERT INTO komutlar VALUES (?, ?)", (soru, cevap))
    ```
  * Offline LLM çal■■t■rma (ggml modeli ile):
    ```bash
    ./llm_binary -m ./models/mistral.ggml -p "Türkçeyi anlayan bir asistan m■s■n?"
    ```

- **Veri Kaynaklar■ ve Ö■renme**:
  * `offline_dataset/` klasöründen metin okuma
  * Kullan■c■n■n geçmi■ komutlar■ndan pattern ç■kar■m■

- **■leri Özellikler**:
  * Kaynak dosya arama (`find`, `grep`, `offline search`)
  * Komutlara göre dosya olu■turma/açma (`os.system`, `shutil`)
  * Termux üzerinden sistem yönetimi (`termux-battery-status`, `termux-toast`)

- **Geli■tirme Stratejisi (Offline)**:
  1. Modeli ve verileri `/sdcard/NotebookLM_Offline/` klasörüne yerle■tir
  2. Termux'ta Python beti■ini çal■■t■r: `python3 main.py`
  3. Sorulan komutlar■ analiz et, modelden cevap al, sqlite'a kaydet
  4. Mikrofon ile konu■ma ba■lat■ld■■■nda Vosk tetiklenir, sonuç modele aktar■l■r

- **Kapan■■**:
  Bu algoritmalar ile sistem, offline çal■■abilir, sesli etkile■im kurabilir, veri kaydedebilir ve komutlara cevap vere