GÜVENLİK TARAMASI YÖNETİM PANELI

PROJE INCELEMESI VE GELIŞTİRME RAPORU

İÇİNDEKİLER

1. ÖZET

Bu proje çalışması, Kali Linux üzerinde çalışan klasik güvenlik araçlarını kullanıcı dostu bir grafik arayüzle birleştiren, yapay zeka destekli ve not tabanlı bir analiz paneli geliştirmeyi hedeflemiştir. Geliştirilen sistem, terminal bilgisi gerektirerek ağ taramaları, zafiyet tespiti ve teknik çıktıların AI tarafından analiz edilerek bulguya dönüştürülmesini sağlar. Ancak yapay zeka entegrasyonu sayesinde, sistem bu bilgileri otomatik analiz eder ve kullanıcıya açıklayıcı öneriler sunarak süreci kolaylaştırır.

Yapay zeka motoru olarak Ollama platformu üzerinden özelleştirilmiş LLaMA modeli kullanılmıştır. Bu motor, çalıştırılan komutları ve çıktıları anlamlandırarak kullanıcıya açıklamalı öneriler sunar. Kullanıcı bu çıktıları isterse tek tuşla bulgu olarak kaydedebilir.

Panelin sahip olduğu özellikler:

- Proje bazlı analiz ve hedef IP yönetimi
- Dinamik komut üretimi ve AI destekli öneri sistemi
- Komut geçmişi, açık portlar ve notların hafızada tutulması (kaiMemory)
- SVG destekli ağaç görselleştirme sistemiyle notların organizasyonu
- Notların ebeveyn ilişkisi ve düzenleme/silme fonksiyonları
- Açık kaynak kodlu ve topluluk destekli geliştirilebilir yapı

Bu panel sayesinde hem yeni başlayanlar hem de ileri düzey güvenlik analistleri, görsel ve etkileşimli bir ortamda siber güvenlik testleri gerçekleştirebilir. Kullanılan açık kaynaklı teknolojiler ve özelleştirme olanakları sayesinde sistem, yerel ve kurumsal ihtiyaçlara kolayca uyarlanabilir hale getirilmiştir.

2.GİRİŞ

Bu tez çalışmasında, siber güvenlik alanında yapay zeka destekli bir güvenlik paneli geliştirilmiştir. Amaç, Kali Linux ortamındaki geleneksel güvenlik araçlarının kullanıcı dostu bir arayüzle entegre edilmesi ve yapay zeka öneri sistemiyle desteklenmesidir. Panel, hedef tespiti, araç yönetimi, terminal çıktısı, bulgu takibi ve not sistemi gibi birçok modülü entegre olarak sunmaktadır.

2.1 Problem

Kali Linux üzerindeki güvenlik araçları genellikle yalnızca terminal üzerinden kullanılmakta ve bu durum, teknik yeterliliği düşük kullanıcılar için ciddi bir bariyer oluşturmaktadır. Aynı zamanda, yapılan güvenlik taramaları, elde edilen bulgular ve yorumlar sistematik bir şekilde belgelenememektedir. Bu da elde edilen çıktıların anlamlandırılmasını ve sunulmasını zorlaştırmaktadır

2.2 Araştırmanın Amacı

Bu projenin temel amacı, Kali Linux güvenlik araçlarının grafiksel bir arayüz ile yönetilmesini sağlamak ve bu arayüzün yapay zeka ile desteklenerek kullanıcıya rehberlik etmesini sağlamaktır. Ayrıca, elde edilen çıktıların görselleştirilmesi ve belgelenmesi de hedeflenmiştir.

2.3 Araştırmanın Önemi

Siber güvenlik analizlerinde komut satırı bilmeyen kullanıcıların da aktif rol alabilmesi için bu tür yapılar kritik öneme sahiptir. Eğitim amaçlı ortamlarda veya hızlı analizlerde, sistematik bir not ve görsel çıktı altyapısına sahip olmak, analiz kalitesini artırır. Yapay zekâ desteği ile bu sistem yalnızca veri toplamakla kalmaz, aynı zamanda rehberlik sunar.

2.4 Varsayımlar

- Kullanıcılar temel ağ bilgisine sahiptir.
- Kali Linux ortamı hazır kurulu ve internet erişimi mevcuttur.
- Ollama modeli daha önceden sisteme kurulmuştur.
- Kullanıcılar teknik çıktılara anlam verebilecek düzeydedir

2.5 Sınırlılıklar

- Yapay zekâ yorumları her zaman %100 doğru olmayabilir, denetim gerekir.
- Komut çıktılarının karmaşıklığı bazı AI modelleri tarafından tam anlaşılamayabilir.
- Uygulama yerel sistemde çalıştığı için bazı sistemlerde performans sorunları yaşanabilir.
- Sunucu bağlantı hataları durumunda terminal-komut etkileşimi kopabilir

2.6 Proje Kapsamı

Bu projenin temel amacı; Nmap, Dirb, FFUF, WPScan gibi Kali Linux araçlarını grafiksel bir ortamda yönetmek, kullanıcıların teknik bilgi gereksinimini azaltmak ve yapay zeka ile desteklenen otomatik bulgu yorumlama sistemini entegre etmektir.

Proje şu kapsamları içerir:

- Proje oluşturma ve hedef IP belirleme
- Port tarama ve terminal çıktısı işleme
- Yapay zeka (Ollama modeli) ile komut analiz ve öneri sistemi
- Not alma, bulgu ağacı (TreeView) ve SVG çıktısı
- Web tabanlı Gradio terminal chatbot sistemi entegrasyonu

2.7 Kullanılan Teknolojiler

- React.js: Kullanıcı arayüzü için
- Flask: Backend API servisi ve terminal yönetimi için
- Gradio: AI terminal sohbeti için
- Ollama + LLaMA 3 modeli: Yapay zeka komut analiz sistemi
- D3.js: Ağaç görselleştirme için
- Nmap, Dirb, FFUF, WPScan: Güvenlik tarama araçları

2.8 Genel Mimari

Bu projede kullanılan güvenlik paneli; frontend (kullanıcı arayüzü), backend (sunucu tarafı) ve yapay zeka destekli analiz motorundan oluşan üç katmanlı bir mimari yapıya sahiptir. Amaç, siber güvenlik araçlarını entegre bir şekilde çalıştırmak, çıktılarını yorumlamak ve tüm süreci kullanıcı dostu bir arayüzle yönetilebilir hale getirmektir.

Genel mimari şu üç ana katmandan oluşur:

1. **Frontend Katmanı (React.js)**

   Kullanıcı ile doğrudan etkileşime giren katmandır. React.js ile geliştirilmiştir. Kullanıcılar bu arayüz üzerinden:
   - Yeni projeler oluşturabilir (NewProject.jsx)
   - Hedef IP taraması yapabilir (TargetSelection.jsx)
   - Güvenlik araçlarını kullanabilir (ScanDashboard.jsx)
   - Terminal üzerinden komut çalıştırabilir
   - Yapay zekadan öneri alabilir
   - Not ve bulgu yönetimi yapabilir (NotePanel, TreeView.jsx)

   Kullanıcıdan alınan her işlem, arka planda Flask API'leri ile haberleşerek işlenir.

2. **Backend Katmanı (Flask)**

   Sunucu tarafında çalışır ve `app.py` dosyası ile yönetilir. Frontend'den gelen istekleri işler ve:
   - Komutları terminalde çalıştırır
   - Notları kaydeder ve listeler
   - Hedef IP'yi ayarlar
   - Port taraması yapar (nmap)
   - Projeleri oluşturur/siler
   - Yapay zeka API'sini çağırarak komutları yorumlatır
   - `chatbot_interface.py` dosyası ile AI + Hafıza yapısını çalıştırır

   Tüm veriler `data/` klasöründe proje bazlı olarak tutulur.

3. **Yapay Zeka Katmanı (Ollama + Gradio)**

   Bu katman, çalıştırılan komutların anlamlandırılması ve çıktılardan potansiyel bulguların tespiti için tasarlanmıştır. Kullanılan teknoloji şunlardır:
   - Ollama: Yerel olarak çalışan LLaMA3 tabanlı model
   - `kali-fix`: Sadece Kali Linux komutlarını analiz eden özel model profili
   - Gradio: Kullanıcının komut girebildiği sohbet paneli

   `chatbot_interface.py` dosyası, Gradio arayüzünü başlatır ve kullanıcı mesajlarını /ai-suggest API'si aracılığıyla Ollama'ya iletir. AI yorumlarını `kaiMemory.json` adlı hafıza yapısında saklar.

Bu üç katmanın uyumlu çalışması sayesinde kullanıcı terminal bilgisi olmadan Kali Linux araçlarıyla güvenlik analizi yapabilir, komut çıktılarından anlamlı yorumlar alabilir ve süreci belgeleyebilir.

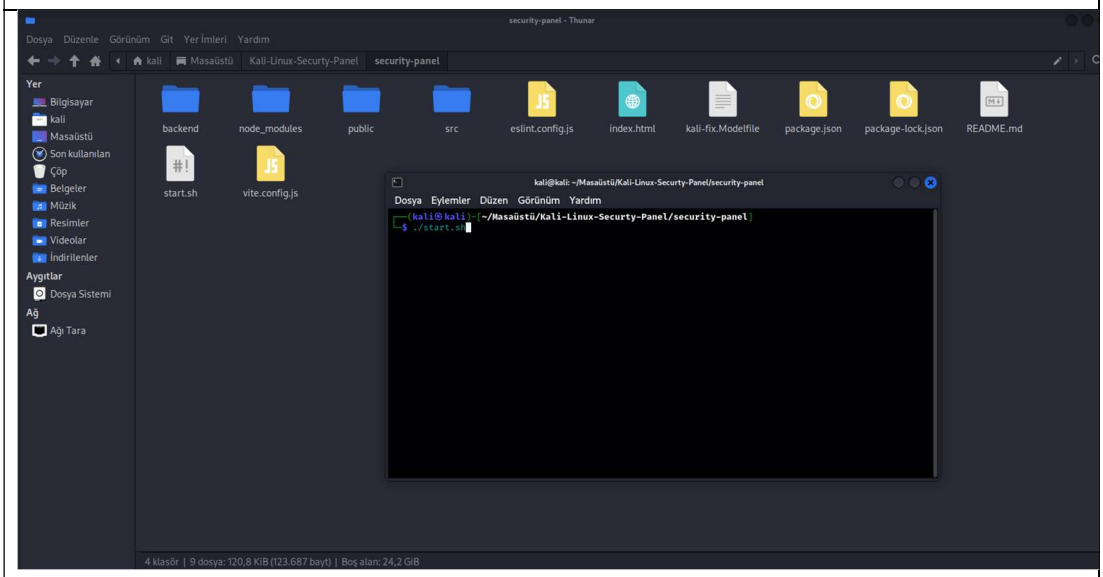
2.9 PROJE RESİMLERİ

Security-Panel projesine ait ekran görüntüleri, sistemin kullanım akışını, işlevsel yapısını ve kullanıcı arayüzünü detaylı bir şekilde yansıtmaktadır. Bu görseller, projenin görsel dökümantasyonu niteliğinde olup, kullanıcıların kurulumu ve kullanımı daha kolay kavrayabilmesi için hazırlanmıştır.

Resimler; proje klasör yapısını, karşılanma ekranını, proje oluşturma sürecini, hedef IP taraması ekranını, komut çalışma stüdyosunu, yapay zeka destekli terminal panelini, notların SVG olarak görselleştirildiği ağaç yapısını ve proje klasörüne ait dosya sistemini kapsamaktadır.

Her bir ekran görüntüsü, uygulamanın belirli bir işlevini belgelemekte ve bu sayede teknik raporun destekleyici görsel bileşeni olarak kullanılmaktadır. Projeye ait bu resimler, aynı zamanda GitHub, LinkedIn veya YouTube gibi platformlarda yapılan tanıtımların temel içerik malzemesidir.

1. Proje Klasör Yapısı – Ana Dizinin Görünümü



Bu klasör yapısı, "security-panel" adlı proje dizinini göstermektedir. Proje, hem frontend (React) hem de backend (Flask) bileşenlerini içinde barındırır.

backend/ → Python ile yazılmış API ve terminal köprüsü buradadır (örnek: app.py)

src/ → React tabanlı kullanıcı arayüzü bileşenleri (örnek: ScanDashboard.jsx, TreeView.jsx)

public/ → Genel HTML ve favicon dosyalarının bulunduğu dizindir

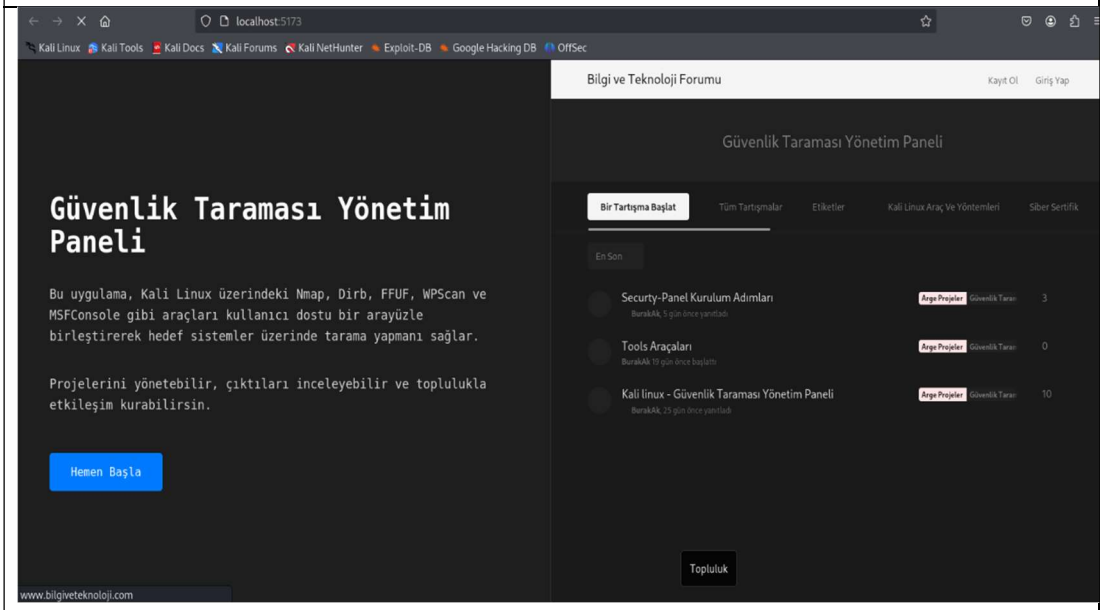
start.sh → Projeyi başlatmak için kullanılan bash betiği

kali-fix.Modelfile → Ollama modeli için özel profil yapılandırması

package.json ve vite.config.js → Frontend bağımlılıkları ve geliştirme sunucusu yapılandırması

Bu yapı, hem geliştiricinin hem de kullanıcıların sistemi yönetmesini ve özelleştirmesini kolaylaştırmak için modüler olarak tasarlanmıştır. Proje doğrudan start.sh betiği ile başlatılabilir.

2. Karşılama Sayfası – Ana Giriş ve Topluluk Erişimi

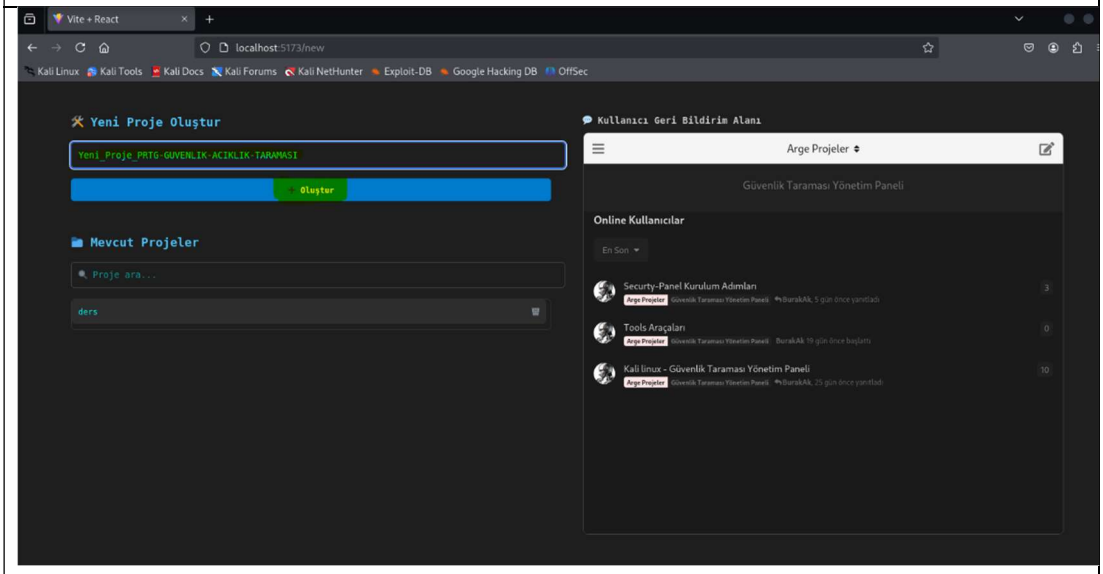


Kullanıcı, uygulamayı açtığında “Güvenlik Taraması Yönetim Paneli” başlığıyla karşılaşır. Sol bölümde, bu panelin amacı kısa ve öz şekilde anlatılmıştır. Kullanıcı burada sistemin Nmap, Dirb, FFUF, WPScan ve MSFConsole gibi araçları içerdiğini öğrenir.

- “Hemen Başla” butonu, kullanıcıyı yeni bir proje başlatma ekranına yönlendirir.
- Sağ bölümde ise “Bilgi ve Teknoloji Forumu” yer alır. Bu forumda kullanıcılar yardım alabilir, tartışmalara katılabilir veya yeni konular başlatabilir.

Bu sayfa, hem teknik hem topluluk açısından projeye güçlü bir başlangıç yapmanızı sağlar. Arayüz, terminal bilmeyen kullanıcılar için bilgilendirici olacak şekilde düzenlenmiştir.

3. Proje Oluşturma Sayfası – Yeni Tarama Sürecine Başlangıç



Bu sayfa, kullanıcıların yeni bir tarama projesi oluşturabileceği ve mevcut projelere erişebileceği alandır.

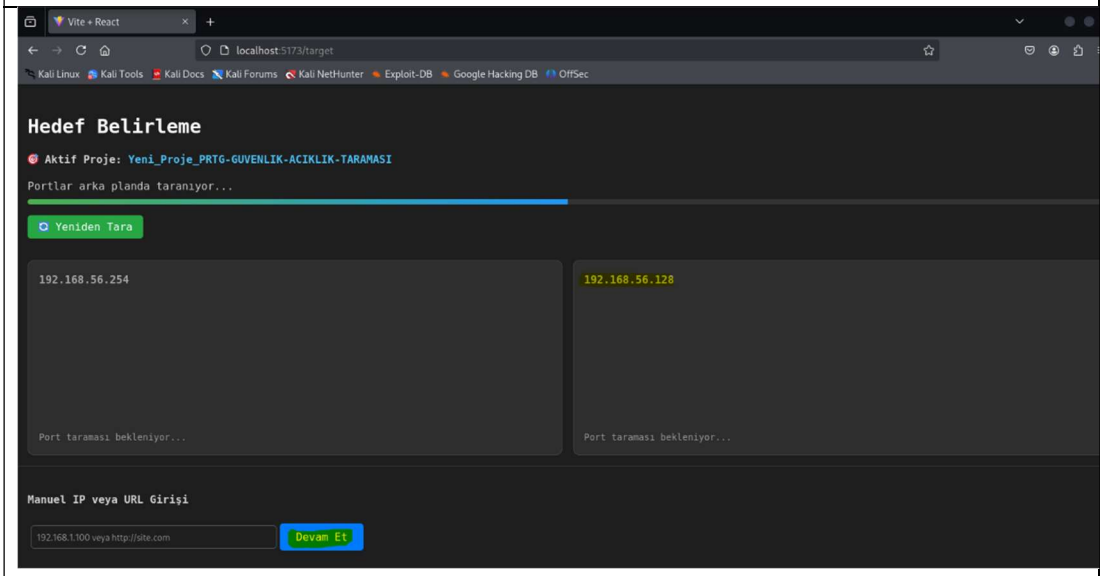
🔧 Yeni Proje Oluştur bölümü:

- Kullanıcı, benzersiz bir proje adı girerek (örneğin: Yeni_Proje_PRTG-GUVENLIK-ACIKLIK-TARAMASI) "+ Oluştur" butonuna tıklar.
- Sistem, bu projeyi yerel belleğe ve backend'e kayıt eder.
- Başarılı oluşturulan projeler, "Mevcut Projeler" altında listelenir.
- 📁 Daha önce oluşturulan projeler burada listelenir.

💬 Sağ Panel – Kullanıcı Geri Bildirim Alanı:

- Bilgi ve Teknoloji Forumu'na entegre edilmiş gerçek zamanlı etkileşim paneli.
- Kullanıcılar aktif tartışmalara katılabilir, yeni fikirler veya hatalar paylaşabilir.

4. Hedef Belirleme Sayfası – IP Keşfi ve Port Taraması



Bu sayfa, aktif proje kapsamında hedef IP adreslerinin otomatik ya da manuel olarak belirlenmesini sağlar.

🎯 Aktif Proje: Sayfanın üst kısmında, hangi proje üzerinde işlem yapıldığı vurgulanır. Örneğin: Yeni_Proje_PRTG-GUVENLIK-ACIKLIK-TARAMASI.

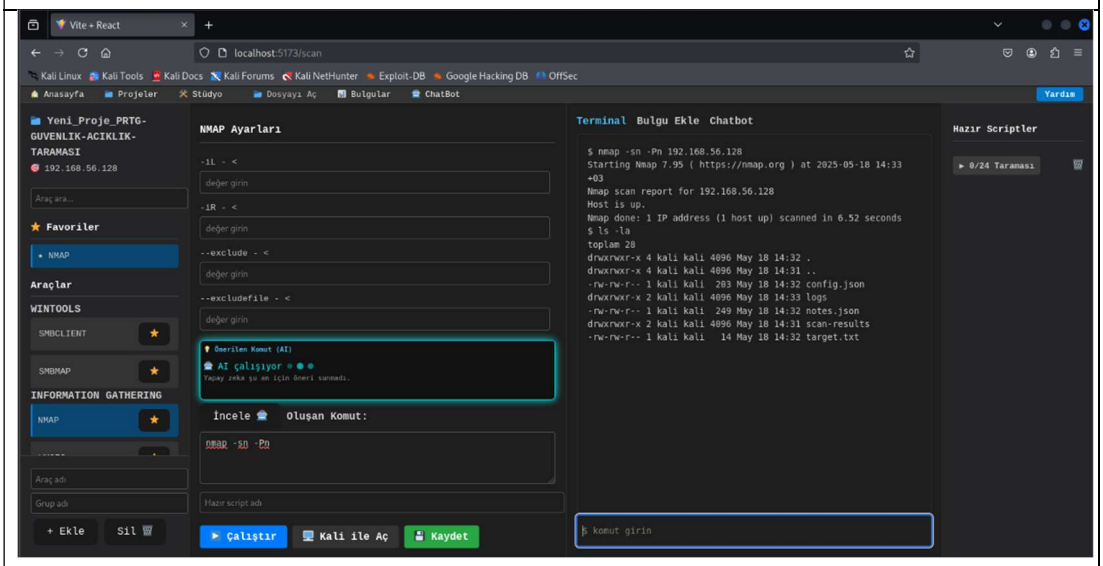
📡 IP Keşfi:

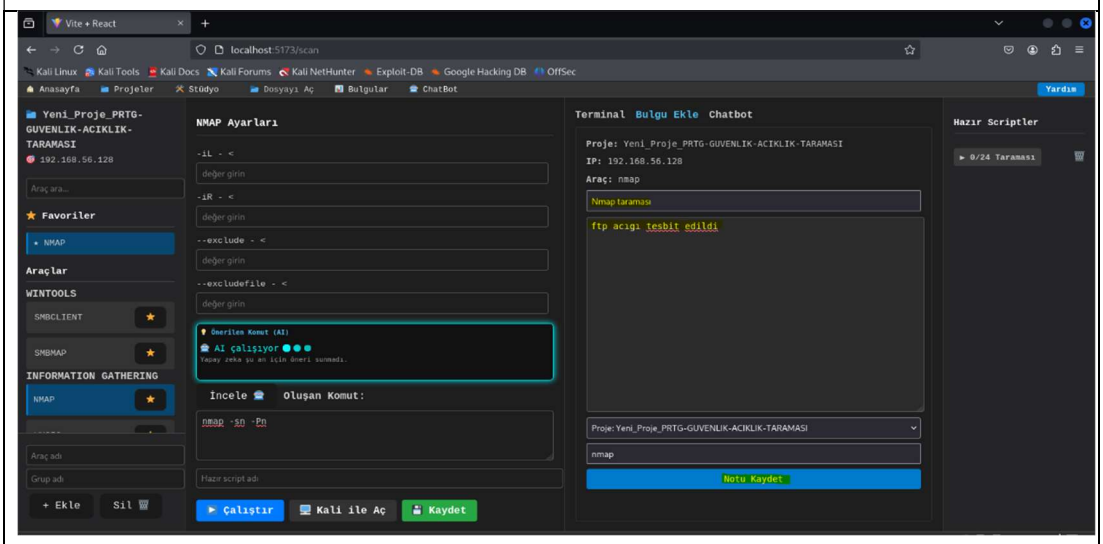
- Sistem, yerel ağda Nmap ile IP taraması yapar.
- Bulunan cihazlar kutucuklar hâlinde listelenir.
- Her IP'nin altında port taraması ilerlemesi gösterilir.
- “Yeniden Tara” butonuyla ağ keşfi tekrar başlatılabilir.

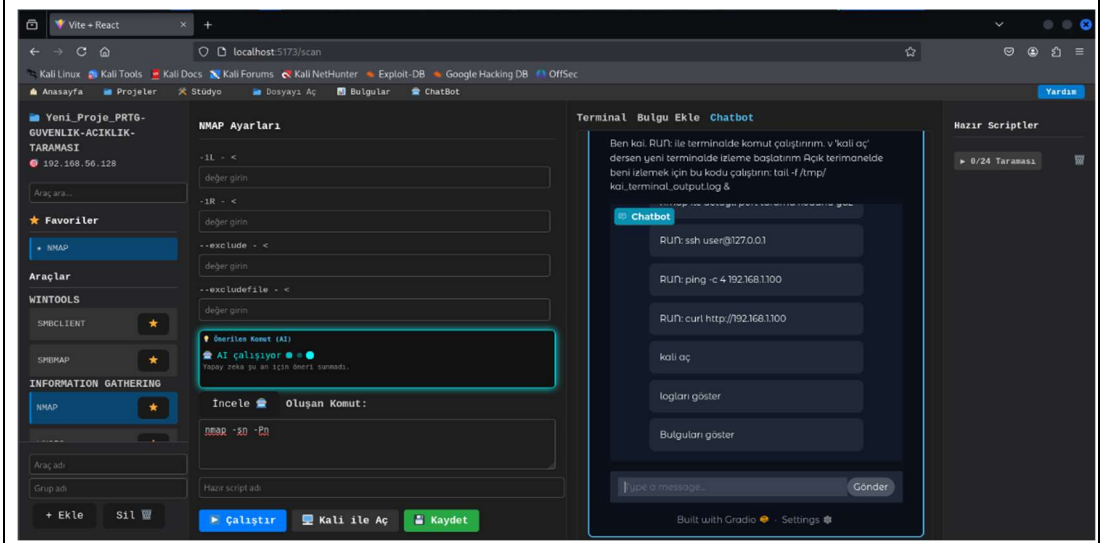
⌨️ Manuel IP Girişi:

- Kullanıcı belirli bir IP ya da web adresini elle girebilir.
- Örneğin: 192.168.1.100 veya http://hedef.com
- “Devam Et” butonuna basıldığında bu IP sistem tarafından hedef olarak belirlenir ve bir sonraki adıma geçilir.

5. Stüdyo (Tarama ve Terminal) Alanı – Tüm İşlemlerin Kalbi







Bu ekran, projenin merkezidir. Araç seçimi, komut oluşturma, AI öneri sistemi, terminal çıktıları, script kaydetme ve bulgu oluşturma işlemleri burada yapılır.

🧭 Araç Paneli (Sol):

- Sol tarafta favori araçlar listelenmiştir (örn: NMAP).
- Alt kategoriler bilgi toplama (INFORMATION GATHERING) ve Windows araçları (WINTools) olarak gruplanmıştır.
- Kullanıcı araç ekleyebilir, silebilir, favorilere alabilir.
- Arama çubuğu ile hızlı filtreleme yapılabilir.

🧠 Yapay Zeka Komut Öneri Kutusu (Orta):

- Seçilen araca göre form alanları belirir.
- Yapay zeka, bu form girdilerine göre önerilen komutu üretir.
- “AI çalışıyor…” animasyonu öneri geldiğinde durur.
- Komutun altında çalıştırılacak tam komut görülür: örn. nmap -sV -Pn

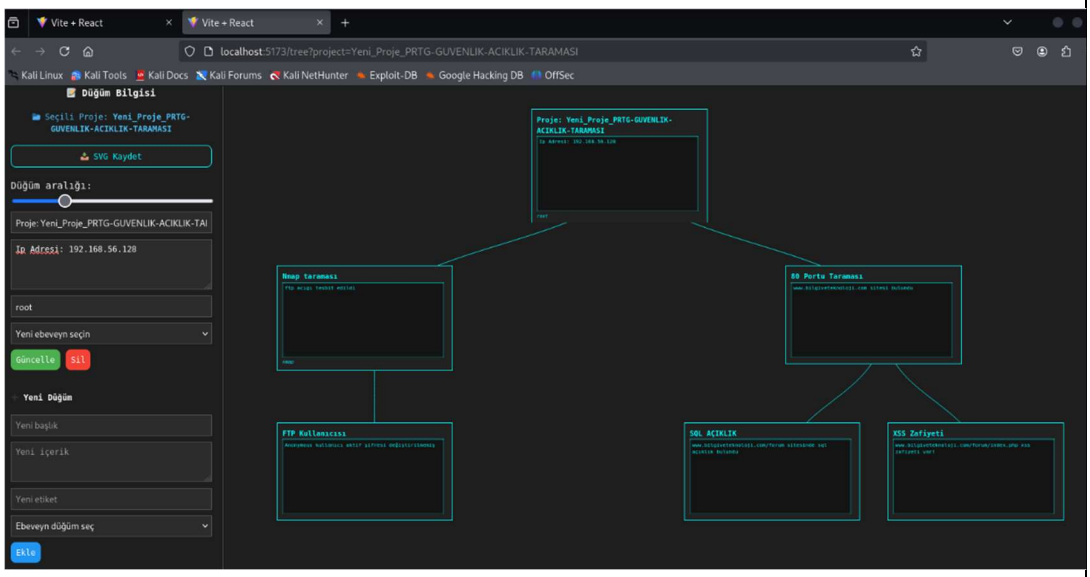
🖥️ Terminal Alanı (Sağ):

- Gerçek zamanlı terminal çıktısı burada görüntülenir.
- Komutlar doğrudan terminale gönderilir ve çıktı satır satır gösterilir.
- AI, bu çıktıyı okuyarak bulguya dönüşebilecek yorumlar sunar.

🗂️ Hazır Scriptler ve Kaydetme:

- Kullanıcı oluşturduğu komutu kaydedebilir.
- Daha sonra bu komutu bir tıkla yeniden çalıştırabilir.
- Script silme veya favorilere ekleme seçenekleri mevcuttur.

6. Bulgular Ağaç Yapısı – Düğümlerle İlişkilendirilmiş Not Sistemi



Bu ekran, yapılan güvenlik taramaları sonucunda elde edilen bulguların görsel olarak hiyerarşik şekilde temsil edildiği not ağacını gösterir. Kullanıcı hem elle düğüm ekleyebilir hem de AI destekli çıktıları buraya otomatik olarak kaydedebilir.

🌳 Ağaç Görselleştirme (Sağ):

- Her kutu bir düğümdür ve bir notu temsil eder.
- Düğümler arasındaki çizgiler, ebeveyn–çocuk ilişkisini gösterir.
- Örnek: “Nmap Taraması” üst düğüm → “FTP Kullanıcısı” alt düğüm
- Notlar, araç adı ve içerik ile birlikte gösterilir (örneğin: “FTP anonim kullanıcı aktif”).

🧩 Sol Panel – Düğüm Bilgisi:

- Aktif proje, IP adresi ve kök not bilgisi gösterilir.
- Düğüm aralığı (mesafe/yoğunluk) slider ile ayarlanabilir.
- Seçili düğüm düzenlenebilir (başlık, açıklama, etiket değiştirilebilir).
- Düğüm silme ve güncelleme fonksiyonları kullanılabilir.

➕ Yeni Düğüm Ekleme:

- Alt bölümde başlık, içerik ve etiket girilerek yeni not eklenebilir.
- Notlar bir üst düğüme bağlanabilir (örn. root → Nmap taraması → FTP).

📥 SVG Kaydet:

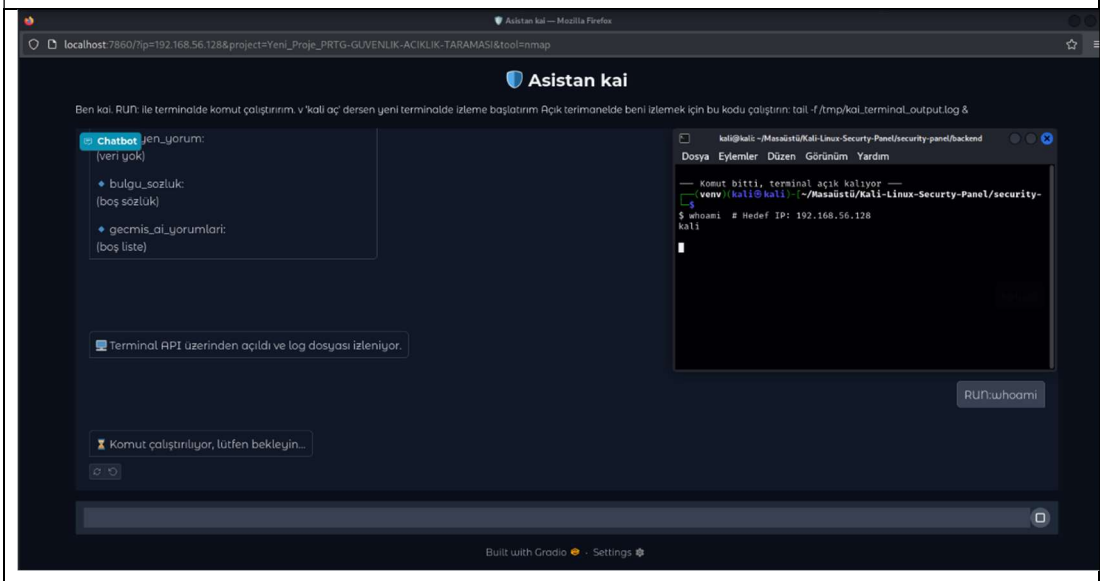
- Grafik çıktısı “SVG” olarak indirilebilir.
- Sunumlarda, raporlarda veya analiz belgelerinde kullanılabilir.

🧠 Yapay Zeka ile Entegrasyon:

- AI tarafından yapılan analiz sonuçları (örneğin: “XSS zafiyeti var”) otomatik olarak ağaç yapısına eklenebilir.

Bu yapı sayesinde tarama süreci sadece komut–çıktı zincirinden ibaret kalmaz; aynı zamanda ilişkisel olarak belgelenir ve tüm analiz görselleştirilmiş bir biçimde sunulur.

7. Yapay Zekâ Asistanı KAI – Terminal ile Etkileşimli Komut Çalıştırma



Bu ekran, projeye entegre edilen Gradio tabanlı yapay zekâ asistanı “Kai” ile terminal üzerinden doğal dilde etkileşim kurulan yapıdır. Kullanıcı komutlarını yazılı şekilde iletir, Kai terminalde çalıştırır ve çıktıları analiz eder. 🤖 Kai'nin Görevi:

- RUN: ile başlayan komutları terminalde çalıştırır.
- Komutlar subprocess ile arka planda yürütülür.
- Terminal çıktısı, /tmp/kai_terminal_output.log üzerinden canlı izlenir.
- Kullanıcıdan gelen komutları AI hafızasına da kaydeder (kaiMemory.json).

🖥️ Terminal Entegrasyonu:

- Sağda gerçek terminal penceresi yer alır (örnek: $ whoami → kali).
- Kai, bu çıktıyı okur ve potansiyel olarak anlamlandırır.
- Gelişmiş sürümlerde bu çıktıdan zafiyet, dosya izi, kullanıcı profili gibi detaylar çıkarılır.

🧠 Kai Hafızası (Bellek Sistemi):

- bulgu_sozluk, gecmis_ai_yorumlari, komut_gecmisi gibi alanlar içerir.
- Örneğin: “whoami” komutu çalıştırıldığında sonuç hafızaya eklenir.
- AI yorumları ve önerileri bu hafızaya göre bağlamsal olarak geliştirilir.

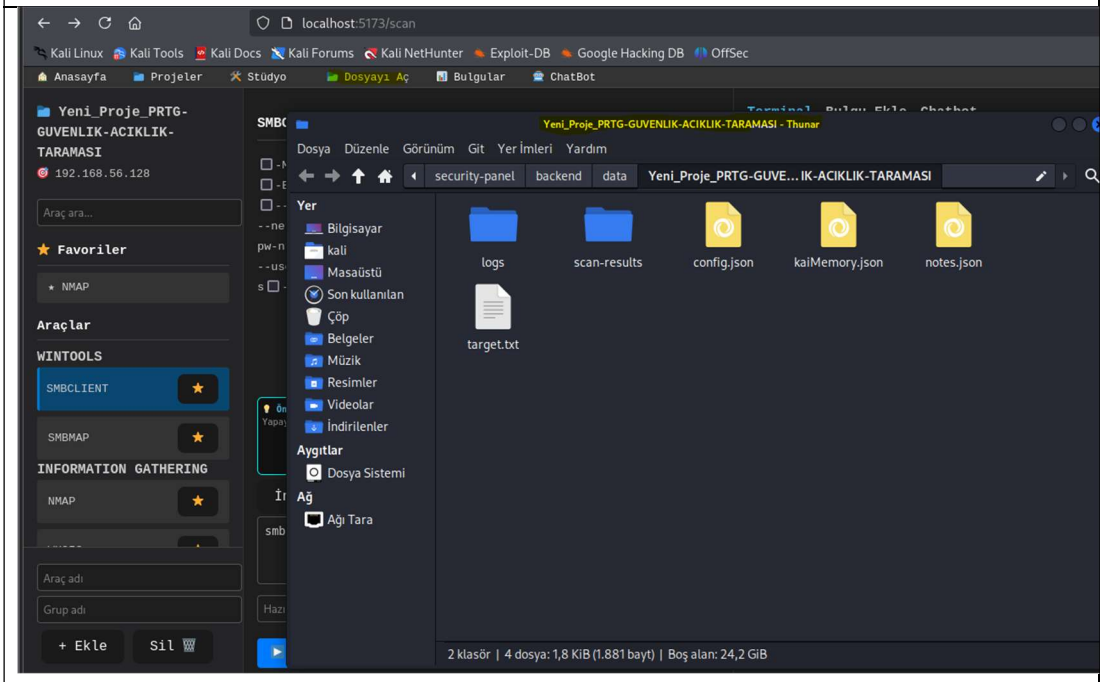
💬 Kullanıcı Deneyimi:

- Kullanıcı doğrudan doğal dille komut yazabilir: RUN: ifconfig, RUN: ls -la
- Kai çalıştırma, yorumlama ve kaydetme süreçlerini ardışık olarak yönetir.
- Terminal hatalarını da yakalayarak kullanıcıyı bilgilendirir.

⚙️ Gelişmiş İzleme:

- Sistem, terminal log dosyasını canlı takip eder:
  tail -f /tmp/kai_terminal_output.log

8. Proje Dosya Dizini – Hedefe Özgü Klasör Yapısı



Bu ekran, her proje için sistem tarafından otomatik olarak oluşturulan dosya yapısını göstermektedir. Her tarama çıktısı, hedef IP, not ve yapay zekâ hafızası ayrı ayrı kaydedilerek veri karışıklığının önüne geçilir.

📁 Klasör Yolu:
/security-panel/backend/data/Yeni_Proje_PRTG-GUVENLIK-ACIKLIK-TARAMASI

🔒 Avantajları:

- Her proje izole bir dizinde tutulur, dosya karmaşası yaşanmaz.
- Dosyalar kolay yedeklenebilir, taşınabilir veya arşivlenebilir.
- JSON formatı sayesinde sistemler arası veri uyumluluğu yüksektir.
- Geliştiriciler bu yapıyı kullanarak ileri düzey analiz araçlarına entegrasyon yapabilir.

Bu yapı, güvenlik testlerini yalnızca bir oturumda bırakmak yerine, kalıcı ve tekrar erişilebilir bir veri havuzuna dönüştürür.

3. ANA BİLEŞENLER

3.1 app.py – Flask API ve Terminal Yönetimi

`app.py`, Security Panel yazılımının en temel yapı taşı olan sunucu tarafı (backend) uygulamasını oluşturmaktadır. Flask mikroframework'ü üzerine inşa edilen bu bileşen, frontend tarafındaki React bileşenlerinden gelen istekleri karşılar, terminalde komut çalıştırır, verileri kaydeder ve yapay zeka ile iletişimi sağlar. Aynı zamanda Gradio tabanlı yapay zeka arayüzünü de başlatan tek merkezdir.

Bu modül, siber güvenlik araçlarını terminalde komut olarak çalıştırmakla kalmaz, aynı zamanda kullanıcı notlarını saklar, IP keşfi yapar, port taraması gerçekleştirir, logları işler ve hafıza temelli yapay zeka yanıtlarını sistemde tutar.

Başlıca bileşenleri aşağıdaki gibi özetlenebilir:

Flask Sunucusunun Başlatılması

`Flask` uygulaması `CORS` (Cross-Origin Resource Sharing) desteğiyle birlikte başlatılır. Sunucu hem API uç noktalarını hem de Gradio servisini barındırır.

Kod parçası:

```
app = Flask(__name__)
CORS(app)
```

Terminal Köprüsü – start_terminal_bridge()

Bu fonksiyon, `/tmp/kai_terminal_input.sh` üzerinden gelen komutları `bash` ile çalıştırır ve çıktısını `/tmp/kai_terminal_output.log` dosyasına kaydeder. Kullanıcı bu terminal üzerinden adım adım komut çalıştırabilir.

Kullanımı:

- input.sh → bash'e akar

- output.log → terminal çıktıları burada birikir

Yapay Zeka Komut Önerisi – /ai-suggest

Bu endpoint, gelen komutu `ollama run kali-fix <komut>` şeklinde çalıştırır. Kullanıcıdan gelen komut + IP bilgisi birleştirilerek yapay zeka motoruna gönderilir.

Koddan örnek:

```
cmd = ['ollama', 'run', 'kali-fix', full_command]

stdout, stderr = proc.communicate()
```

Proje ve Hedef Yönetimi

Proje oluşturma (`/create-project`), silme (`/delete-project`), hedef IP kaydı (`/set-target`), hedef IP okuma (`/get-target`) gibi uç noktalar proje yapısının yönetimini sağlar. Tüm veriler `data/` klasörü altında saklanır.

IP Keşfi ve Port Taraması

`/nmap-discover` endpoint'i yerel IP aralığını kullanarak `nmap -sn` komutu ile canlı makineleri tespit eder. `/scan-ports` endpoint'i ise tespit edilen her IP için port taraması yapar ve açık portları analiz eder.

Taramadan sonra sonuçlar geçmişle karşılaştırılır, önceki açık portlar kapalıysa `status: closed` şeklinde kayıt edilir.

Not Sistemi – /notes ve /tree

Kullanıcıların eklediği notlar JSON formatında `notes.json` adlı dosyada saklanır. `/notes` endpoint'i ile notlar eklenebilir veya düzenlenebilir. `/tree` endpoint'i ile bu notlar hiyerarşik bir ağaç yapısı şeklinde frontend'e gönderilir.

Terminal Çıktısı İzleme

`monitor_terminal_output()` fonksiyonu, terminal çıktılarındaki her satırı `stdout` üzerinden okur. Bu fonksiyon `threading` ile arka planda sürekli çalışır. Amaç, yapay zeka ve kullanıcı için canlı terminal izleme imkânı sunmaktır.

Sonuç olarak, `app.py` dosyası sadece bir sunucu uygulaması değil; aynı zamanda sistemin merkezi koordinasyon birimidir. Komut akışları, veri kaydı, yapay zeka köprüsü, terminal yönetimi ve proje denetimi bu dosya üzerinden yürütülmektedir.

3.2 Chatbot_interface.py – Gradio Chatbot ve Hafıza Sistemi

Bu dosya, yapay zeka destekli terminal asistanının kalbini oluşturur. İşlevi, kullanıcıdan gelen komutları hem terminalde çalıştırmak hem de Ollama tabanlı yapay zeka modeli aracılığıyla yorumlamaktır. Ayrıca, tüm bu süreci proje bazlı hafıza dosyasında saklayarak "kaiMemory.json" ile kalıcı bir bağlam yaratır.

Ana Sorumluluklar:

- Gradio arayüzü ile sohbet ekranı oluşturmak
- Komutları terminalde çalıştırmak ("RUN:" komutu ile)
- Terminal çıktılarını oku, son 20 satırı göster
- Yapay zekaya yorumlatmak ve "evet/hayır" cevabına göre not sistemine entegre etmek
- Komuttan port bilgisi, ftp anonim erişimi gibi öğeleri ayrıştırmak
- Hafıza yapısına son çıktıyı ve yorumları eklemek

Kai Hafıza Yapısı:

- hedef_ip
- local_ip
- komut_gecmisi
- acik_portlar
- notlar
- bekleyen_yorum
- suggested_commands
- gecmis_ai_yorumlari
- bulgu_sozluk

Bu mimari sayesinde, sadece tek satırlık bir terminal botu değil, aynı zamanda öğrenebilen, anlam çıkarabilen ve tespit edilen zafiyetleri sözülü komutlarla not sistemine entegre edebilen bir dijital asistan ortaya çıkmaktadır.

Örnek Akış:

- Kullanıcı: "RUN: nmap -sV 192.168.1.1"
- Terminal çıktısı: çalıştırılır, kaydedilir
- AI: çıktıya yorum yapar, zafiyet varsa belirler
- Kullanıcı: "evet" derse, yorum not olarak kaydedilir

Tüm bu yapı, Gradio arayüzü içinde sohbet benzeri bir formatta gerçekleşir.

3.3 ScanDashboard.jsx – Komut ve Terminal Paneli

ScanDashboard bileşeni, güvenlik tarama sürecinin merkezidir. Kullanıcıların güvenlik araçlarını seçmesini, bu araçlar için komut üretmesini, komutları terminalde çalıştırmasını ve sonuçları analiz etmesini sağlar. React ile geliştirilen bu bileşen çok sayıda modül ve state yönetimi içerir.

Temel Özellikler:

- Araç gruplarını ve araçları listeleme (toolsByGroup)
- Yardım seçeneklerine göre form üretimi (fetchToolHelp)
- Komut oluşturma (buildCommand)
- Terminal çıktısını gösterme ve çalıştırma (handleRun)
- Yapay zeka destekli komut öneri kutusu (getAiSuggestion)
- Önceden kaydedilmiş scriptleri çağırma ve kaydetme
- Not sistemi entegrasyonu (NotePanel)
- Gradio chatbot iframe entegrasyonu

Bileşen içinde üç ana sekme bulunur:

1. Terminal: Komut çalıştırma ve çıktıları görüntüleme
2. Bulgu Ekle: Not alma ve bulguları kaydetme
3. Chatbot: Yapay zeka paneli ile doğal dilde etkileşim

Örnek Akış:

- Kullanıcı bir araç seçer (örneğin: nmap)
- Form üzerinden parametreleri girer (örneğin -sV, -p 80)
- Komut oluşturulur: "nmap -sV -p 80 192.168.1.1"
- Komut çalıştırılır, çıktı ekrana yazılır
- AI incelemesi yapılır, öneri kutusunda açıklama görünür
- Kullanıcı isterse komutu tekrar düzenleyip çalıştırabilir

Bu bileşen, sistemin terminal ve görsel kullanıcı arayüzü arasında köprü kurar. Kullanıcı dostu yapısıyla karmaşık komutlar sadeleştirilmiş bir şekilde yürütülür.

3.4 TargetSelection.jsx – Hedef IP Seçimi ve Port Taraması

Bu bileşen, kullanıcının hedef IP adresini belirlemesini ve ağdaki cihazların otomatik olarak taranmasını sağlar. Kullanıcı arayüzü modern bir grid sistemiyle tasarlanmış, her IP adresi ve port bilgisi kutucuklar şeklinde gösterilmiştir. React hook'ları ve Axios kullanılarak gerçek zamanlı IP keşfi ve port taraması yapılır.

Başlıca Özellikler:

- Yerel ağda IP keşfi yapmak (nmap -sn komutu üzerinden)
- Seçilen IP için port taraması başlatmak (nmap -p 1-1000)
- Port bilgilerini geçmiş taramalarla karşılaştırmak
- IP'yi seçilen projeye kaydetmek ve yönlendirme yapmak
- Manuel IP girişi seçeneği sunmak

İş Akışı:

1. Bileşen yüklendiğinde localStorage'dan aktif proje alınır
2. IP keşfi başlatılır (/nmap-discover endpointi)
3. Bulunan IP'ler frontend'de kutularla gösterilir
4. Seçilen IP adresi /set-target endpointine kaydedilir
5. İlk not olarak bu hedef IP "root" düğüm olarak notes.json dosyasına yazılır

6. Kullanıcı "/scan" sayfasına yönlendirilir

Örnek Kod: await axios.post('http://localhost:5050/set-target', { project_name: activeProject, target_ip: finalTarget })

Kullanıcı Deneyimi:

- Yüklenme süreci boyunca ilerleme çubuğu gösterilir
- Açık portlar yeşil, kapalı portlar gri renkle ayırt edilir
- Manuel IP giriş kutusu daima görünür durumdadır

Bu yapı sayesinde kullanıcı, ağ üzerindeki sistemleri hızlıca keşfedebilir ve tarama yapacağı hedefi tek tıkla belirleyebilir. Sistem otomatik olarak bu IP'yi ilgili projeye bağlayarak not sistemine ilk adımı atar.

3.5 TreeView.jsx – Not Ağaç Görselleştirme ve SVG Çıktı

TreeView bileşeni, kullanıcıların eklediği notları hiyerarşik bir ağaç yapısı olarak görselleştiren güçlü ve etkileşimli bir arayüz sunar. D3.js kütüphanesi kullanılarak geliştirilmiş bu bileşen, her bir notu bir düğüm olarak ele alır ve bu düğümler arasındaki ebeveyn-çocuk ilişkisini görsel olarak sunar.

Temel Özellikler:

- Hiyerarşik not yapısını D3.js ile ağaç diyagramına dönüştürme
- Her düğümü kart formatında (foreignObject) HTML içeriğiyle gösterme
- Seçilen düğümün içeriğini düzenleme (başlık, açıklama, etiket)
- Ebeveyn değiştirme fonksiyonu
- Düğüm silme (alt düğümleri root'a taşıyarak)
- Yeni düğüm ekleme
- SVG olarak ağaç yapısını dışa aktarma (Kaydet butonu ile)
- Zoom, pan ve yeniden konumlandırma desteği

Düğüm içeriği aşağıdaki gibi görselleştirilir:

- Başlık (bold yazı)
- Açıklama metni (readonly textarea)
- Etiket bilgisi (italic yazı)

Koddan Örnek:

```
<foreignObject>
  <xhtml:div>
    <strong>{d.data.title}</strong>
    <textarea>{d.data.content}</textarea>
    <em>{d.data.tool}</em>
  </xhtml:div>
</foreignObject>
```

İş Akışı:

1. http://localhost:5050/tree/{project} API'sinden notlar çekilir
2. D3 hiyerarşi yapısına çevrilir
3. Görselleştirme yapılır, tıklanan düğüm düzenlenebilir
4. Yeni düğüm eklendiğinde API'ye post edilir
5. SVG çıktısı alınabilir

Bu bileşen, not sistemini yalnızca metin bazlı bir liste olarak değil, anlamlı bir ağaç yapısı içinde sunar. Böylece kullanıcılar hangi notun hangi bulguya bağlı olduğunu, hangi araçla elde edildiğini, hangi adımın sonucunda geldiğini görsel olarak takip edebilir.

Özetle TreeView.jsx, teknik bulgu akışını belgeleyen ve organize eden kritik bir görsel analiz modülüdür.

3.6 NewProject.jsx – Proje Oluşturma ve Yönetimi

Bu bileşen, kullanıcının yeni projeler oluşturmasını, var olanları görüntülemesini ve seçilen bir projeye geçiş yapmasını sağlar. Projenin hem isim doğrulaması yapılır hem de IP hedefi tanımlanmışsa tarama sayfasına otomatik yönlendirme gerçekleştirilir.

Temel Özellikler:

- Proje ismi girilerek yeni proje oluşturma (/create-project endpoint)
- Mevcut projelerin listelenmesi (/projects endpoint)
- Her projeye özel hedef IP kontrolü (/get-target)
- Hedef IP varsa: /scan sayfasına yönlendirme
- Yoksa: /target sayfasına yönlendirme
- Proje silme (/delete-project/:name)

Kullanıcı Deneyimi:

- Yan panelde oluşturulan projeler listelenir
- Silme işlemi için onay kutusu çıkar
- Sağ sütunda iframe aracılığıyla topluluk ve kullanıcı geri bildirim sayfası görüntülenir
- Proje adı boş bırakılırsa veya geçersiz karakter içerirse hata mesajı gösterilir

Koddan Örnek:

```
await axios.post('http://localhost:5050/create-project', { name: trimmed })
```

İş Akışı:

1. Kullanıcı proje adını girer
2. Sistem, proje adının uygunluğunu kontrol eder
3. Proje oluşturulur ve localStorage'a kaydedilir
4. Kullanıcı, hedef IP atanmışsa doğrudan tarama sayfasına gider

Bu bileşen, sistemin başlangıç noktasıdır. Tüm tarama ve analiz süreci bu bileşende oluşturulan proje üzerinden yönetilir

3.7 Home.jsx – Giriş Ekranı ve Tanıtım Sayfası

Home.jsx bileşeni, kullanıcıyı karşılayan ve uygulamanın genel amacını tanıtan ana giriş ekranıdır. Basit ama etkili bir şekilde tasarlanmış olan bu bileşen, kullanıcıya projenin neler yapabileceğini açıklar ve onu hızlıca yeni bir proje başlatmaya yönlendirir.

Temel Özellikler:

- Sol panelde uygulama tanıtımı
- Sağ panelde topluluk forum iframe bileşeni
- "Hemen Başla" butonu ile kullanıcıyı /new rotasına yönlendirme
- Duyarlı (responsive) tasarım ve sade görsel düzen

Sol Panel İçeriği:

- Uygulamanın adını ve amacını içeren başlık
- Kali Linux araçlarını (Nmap, Dirb, FFUF, WPScan, MSFConsole) grafik arayüzle birleştirdiğini anlatan açıklamalar
- Proje yönetimi, çıktı görüntüleme ve toplulukla etkileşim özelliklerini özetleyen bilgi metinleri
- Başlat butonu: Kullanıcıyı proje oluşturma ekranına taşır

Koddan Örnek:

```
<Link to="/new">
  <button>
    Hemen Başla
  </button>
</Link>
```

Sağ Panel İçeriği:

- Topluluk sayfası iframe ile gösterilir
- Forumdan doğrudan destek veya yorum alınabilir

Kullanıcı Deneyimi:

- Anlaşılır ve hızlı giriş sağlar
- Kullanıcıdan herhangi bir işlem beklemeden yönlendirme yapar
- Arka plan koyu tema ile bütünleşir ve uygulama kimliğini vurgular

Home.jsx bileşeni, teknik detay içermeyen ancak kullanıcıyı uygulama akışına dahil eden ilk izlenim noktasıdır. Kullanıcının projeye hızlı bir şekilde adapte olması açısından kritik öneme sahiptir.

4. YAPAY ZEKA

4.1 Hugging Face – Teorik Arka Plan ve Kullanım Amacı

Hugging Face, doğal dil işleme (NLP), görsel işleme, konuşma analizi ve çok modlu yapay zeka alanlarında açık kaynaklı modeller, veri setleri ve araçlar sunan öncü bir platformdur. Geliştiriciler, araştırmacılar ve endüstriyel kullanıcılar için büyük bir AI ekosistemi sağlar. En bilinen projesi "Transformers" kütüphanesidir.

Neden Kullanılır:

- Yaygın olarak kullanılan hazır modeller (GPT, BERT, RoBERTa, T5, vs.)
- Tek satırlık kod ile model çağırma ve kullanma
- API erişimi veya yerel model barındırma seçenekleri
- Eğitimli modellerin paylaşımı ve topluluk katkısı
- Metin sınıflandırma, çeviri, soru-cevap, özetleme gibi görevlerde kullanılabilir

Proje Kapsamında Kullanımı:
Bu panelde Hugging Face API yerine, benzer şekilde çalışan ancak offline yetenek sağlayan Ollama tercih edilmiştir. Hugging Face hakkında bilgi verilmesinin amacı,

bu projenin AI alt yapısının neye dayandığını açıklamak ve alternatif teknolojileri göstermektir.

4.2 Ollama – Yerel AI Motoru ve Kurulum Kılavuzu

Ollama, yerel olarak çalışan hafif ve optimize edilmiş yapay zeka modellerini çalıştırmak için geliştirilen bir AI runtime platformudur. Hugging Face gibi büyük sunuculara ihtiyaç duymadan, offline ortamda LLM (large language model) çalıştırmaya olanak tanır.

Neden Ollama Kullanıldı:

- İnternet bağlantısı olmadan yerel çalışabilir
- Kaynak dostu, sistem performansına göre optimize edilebilir
- Model profili (Modelfile) tanımlanarak belirli görevler için özelleştirilebilir
- ollama run komutu ile AI çağırma kolaylığı

Kullanım Örneği:

```
ollama run kali-fix "nmap -sV 192.168.1.1 çıktısını analiz et"
```

Kurulum Kılavuzu (Linux):

1. curl komutu ile Ollama'nın kurulumu yapılır:

```
curl -fsSL https://ollama.com/install.sh | sh
```

2. Sistem PATH ayarları yapılır (gerekirse):

```
export PATH="$PATH:/root/.ollama/bin"
```

3. Ardından servis başlatılır:

```
ollama serve
```

4. Model indirilir (örn. LLaMA3):

```
ollama pull llama3
```

5. Özel model tanımı yapılır (Modelfile):

```
ollama create kali-fix -f Modelfile
```

6. Kullanım:

ollama run kali-fix "nmap çıktısını analiz et"

Bu yapının en büyük avantajı, AI sisteminin internet erişimi gerektirmeden terminal ortamında doğrudan kullanılabilmesidir. Ayrıca bu sistem app.py üzerinden tanımlı /ai-suggest endpoint'i ile entegre olarak çalışır.

7. Komut Öneri ve AI Yorumlama Süreci

Bu proje kapsamında geliştirilen sistem, yalnızca kullanıcıdan gelen komutları terminalde çalıştırmakla kalmaz, aynı zamanda bu komutların çıktısını yapay zeka motoruna göndererek anlamlı yorumlar üretir. Bu yorumlar üzerinden zafiyet tespiti, bulgu analizi ve not oluşturma işlemleri gerçekleştirilir.

Yapay zeka destekli yorumlama süreci üç temel bileşenden oluşur:

1. Kullanıcıdan gelen komutun toplanması
2. Komutun terminalde çalıştırılması ve çıktının yakalanması
3. Çıktının AI modeline gönderilerek analiz edilmesi

Bu işlem döngüsü şu dosyalar arasında gerçekleşir:

- ScanDashboard.jsx (komut oluşturma ve gönderme)
- app.py (komutu çalıştırma ve AI API'ye iletme)
- chatbot_interface.py (Gradio üzerinden etkileşimli AI sohbeti)

Komut Önerisi:
Kullanıcı, herhangi bir araç seçip parametrelerini girdiğinde sistem otomatik olarak komut önerisi üretir. Bu komut, kullanıcı değişiklik yapmadığı sürece hazır olarak gelir ve AI tarafından incelenmek üzere arka planda ollama motoruna gönderilir.

Örnek:

nmap -sV -p 21,22 192.168.1.1

Bu komut AI'ya şu şekilde iletilir:

ollama run kali-fix "nmap -sV -p 21,22 192.168.1.1 çıktısını analiz et"

AI Yanıt Türleri:

- 🔧 Önerilen: Alternatif komut tavsiyesi
- 🛠️ Hata Giderildi: Hatalı komutun düzeltilmiş versiyonu
- ℹ️ Açıklama: Komutun yaptığı işlemler, güvenlik açığı riski, alınması gereken önlemler

Yorumlama Süreci:

1. Komut çıktısı alınır ve kai_terminal_output.log dosyasına kaydedilir
2. AI yorum üretir (örn: "21/tcp açık. FTP anonim erişim açık olabilir. Bu ciddi bir zafiyettir.")
3. Yorum kullanıcıya gösterilir: "Bu yorumu not olarak kaydetmek ister misiniz? (evet / hayır)"
4. Kullanıcı onaylarsa not olarak eklenir (tool, ip ve açıklama dahil)

Bu süreç şu faydaları sağlar:

- Kullanıcıların güvenlik analizini daha bilinçli yapması
- AI'nın geçmiş çıktılara göre zafiyet tespiti yapabilmesi
- Notların otomatik oluşturulması sayesinde belgeleme sürecinin hızlanması

Sonuç olarak, bu yapı klasik terminal kullanımının ötesine geçerek, kullanıcıya rehberlik eden ve karar sürecini destekleyen bir yapay zeka katmanı sunar.

5. Komut Öneri ve AI Yorumlama Süreci

Bu proje kapsamında geliştirilen sistem, yalnızca kullanıcıdan gelen komutları terminalde çalıştırmakla kalmaz, aynı zamanda bu komutların çıktısını yapay zeka motoruna göndererek anlamlı yorumlar üretir. Bu yorumlar üzerinden zafiyet tespiti, bulgu analizi ve not oluşturma işlemleri gerçekleştirilir.

Yapay zeka destekli yorumlama süreci üç temel bileşenden oluşur:

4. Kullanıcıdan gelen komutun toplanması
5. Komutun terminalde çalıştırılması ve çıktının yakalanması
6. Çıktının AI modeline gönderilerek analiz edilmesi

Bu işlem döngüsü şu dosyalar arasında gerçekleşir:

- ScanDashboard.jsx (komut oluşturma ve gönderme)
- app.py (komutu çalıştırma ve AI API'ye iletme)
- chatbot_interface.py (Gradio üzerinden etkileşimli AI sohbeti)

Komut Önerisi:
Kullanıcı, herhangi bir araç seçip parametrelerini girdiğinde sistem otomatik olarak komut önerisi üretir. Bu komut, kullanıcı değişiklik yapmadığı sürece hazır olarak gelir ve AI tarafından incelenmek üzere arka planda ollama motoruna gönderilir.

Örnek:

nmap -sV -p 21,22 192.168.1.1

Bu komut AI'ya şu şekilde iletilir:

ollama run kali-fix "nmap -sV -p 21,22 192.168.1.1 çıktısını analiz et"

AI Yanıt Türleri:

- 🔧 Önerilen: Alternatif komut tavsiyesi
- 🛠️ Hata Giderildi: Hatalı komutun düzeltilmiş versiyonu
- ℹ Açıklama: Komutun yaptığı işlemler, güvenlik açığı riski, alınması gereken önlemler

Yorumlama Süreci:

5. Komut çıktısı alınır ve kai_terminal_output.log dosyasına kaydedilir
6. AI yorum üretir (örn: "21/tcp açık. FTP anonim erişim açık olabilir. Bu ciddi bir zafiyettir.")
7. Yorum kullanıcıya gösterilir: "Bu yorumu not olarak kaydetmek ister misiniz? (evet / hayır)"
8. Kullanıcı onaylarsa not olarak eklenir (tool, ip ve açıklama dahil)

Bu süreç şu faydaları sağlar:

- Kullanıcıların güvenlik analizini daha bilinçli yapması

- AI'nın geçmiş çıktılara göre zafiyet tespiti yapabilmesi
- Notların otomatik oluşturulması sayesinde belgeleme sürecinin hızlanması

Sonuç olarak, bu yapı klasik terminal kullanımının ötesine geçerek, kullanıcıya rehberlik eden ve karar sürecini destekleyen bir yapay zeka katmanı sunar.

6. Güvenlik Araçları Entegrasyonu ve Kullanımı

Bu projede, Kali Linux dağıtımında yaygın olarak kullanılan siber güvenlik araçları grafik kullanıcı arayüzüne entegre edilmiştir. Amaç, bu araçları terminal bilgisi gerektirmeden parametrik olarak çalıştırmak ve çıktıları anlık olarak kullanıcıya sunmaktır. Araçlar gruplar halinde sunulur ve her bir araç için özelleştirilmiş parametre formu, komut üretici ve terminal çıktısı ekranı bulunmaktadır.

Entegre Edilen Araç Grupları:

- Ağ Taraması (örn: nmap)
- Web Tarayıcıları (örn: dirb, ffuf)
- Zafiyet Tarayıcıları (örn: wpscan)
- Sızma Testi Yardımcıları (örn: msfconsole, hydra)

Sistem Yapısı:

- Her aracın komut satırı --help çıktısı GET /tool-help/:tool ile çekilir.
- Bu çıktı analiz edilerek dinamik olarak parametre formu oluşturulur.
- Kullanıcı, checkbox veya text alanlarına parametre girer.
- Sistem otomatik olarak komutu oluşturur (örn: nmap -sV -p 21 TARGET)
- Bu komut AI öneri sistemine de gönderilerek doğruluğu test edilir.

Koddan Örnek (ScanDashboard.jsx):

```
const buildCommand = () => {
  let cmd = selectedTool
  helpOptions.forEach(param => {
    const value = formData[param.flag]
    if (param.type === 'checkbox' && value) cmd += ` ${param.flag}`
    else if (param.type === 'text' && value) cmd += ` ${param.flag} ${value}`
  })
  return cmd
}
```

Hazır Script Özelliği:

- Kullanıcı, belirli bir araç için tanımladığı komutu kaydedebilir.
- Script adı girilir, komut hazır şablon olarak saklanır.
- Daha sonra tek tıklamayla bu komut çağrılarak çalıştırılabilir.

Yürütme:

- Komut "Çalıştır" butonuyla terminalde yürütülür
- Terminal çıktısı satır satır arayüzde gösterilir
- Komut geçmişi hafızada (kaiMemory) tutulur

Favori Araçlar:

- Kullanıcılar sık kullandığı araçları yıldız simgesi ile favori olarak işaretleyebilir.
- Favoriler ayrı bir başlık altında listelenir.

Gelişmiş Arayüz Özellikleri:

- Araç grubu oluşturma ve araç ekleme/silme
- Arama kutusu ile araç filtreleme
- AI destekli öneri kutusu
- Komut oluşturucu + AI analiz paneli

Sonuç:

Geleneksel olarak CLI üzerinden çalıştırılan siber güvenlik araçları, bu panel ile parametrik formlar ve öneri sistemleri üzerinden çalıştırılabilir hale gelmiştir. Böylece, uzman olmayan kullanıcılar bile karmaşık komutlara ihtiyaç duymadan tarama ve analiz işlemlerini gerçekleştirebilir.

7. Not Sistemi ve Bulguların Kaydı

Bu projede geliştirilen not sistemi, kullanıcıların gerçekleştirdiği taramalara ait gözlemleri, AI tarafından yapılan yorumları ve analiz bulgularını düzenli bir şekilde kaydetmesini sağlar. Sistem; manuel not girişine, yapay zeka yorumlarının otomatik not olarak kaydedilmesine ve tüm bu notların görsel ağaç yapısı içinde takip edilmesine imkân tanır.

Ana Özellikler:

- Her not bir başlık, içerik, araç adı (tool) ve ebeveyn kimliği (parent) içerir
- İlk not, hedef IP'yi içeren kök (root) not olarak otomatik oluşturulur
- Notlar notes.json dosyasında proje bazlı olarak saklanır

- /notes/{project} endpoint'i ile notlar alınabilir veya gönderilebilir
- /tree/{project} endpoint'i ile notlar ağaç yapısına çevrilerek frontend'e aktarılır

Not Kaydetme Süreci:

1. Kullanıcı not formuna başlık, açıklama, etiket ve ebeveyn seçimi girer
2. Not, backend'e POST edilir
3. Backend, notu JSON formatında dosyaya ekler

AI Entegrasyonu ile Not Oluşturma:

- Kullanıcı bir komut çalıştırır
- AI çıktıyı analiz eder ve yorum üretir
- Sistem, kullanıcıya "Bu yorumu not olarak kaydedeyim mi?" şeklinde sorar
- Kullanıcı "evet" derse, yorum root notun altına bir çocuk not olarak eklenir

Koddan Örnek:

```
{
 "title": "FTP anon erişim",
 "content": "AI: 21/tcp açık. Anonim FTP erişimi etkin olabilir.",
 "tool": "nmap",
 "parent": "root"
}
```

Görsel Ağaç Yapısı:

- Tüm notlar hiyerarşik şekilde görselleştirilir (TreeView.jsx)
- Her not kart şeklinde başlık, içerik ve etiket bilgisiyle gösterilir
- Notlar sürüklenemez ancak ebeveyni değiştirilebilir
- SVG çıktısı alınabilir ve dışa aktarılabilir

Notlar Nerede Kullanılır?

- Bulguların düzenli saklanması
- Rapor oluşturma sürecinde görsel analiz
- Yapay zekanın tespit ettiği zafiyetlerin belgelenmesi
- Hedef sistem hakkında adım adım bilgi birikimi oluşturma

Sonuç:
Bu sistem sayesinde sadece teknik çıktılar değil, bunların yorumları ve analizleri de kalıcı ve görsel olarak takip edilebilir bir şekilde saklanmaktadır. Not sistemi, projenin belgelendirme yönünü güçlendiren en önemli modüllerden biridir.

8. Görselleştirme ve Bulguların SVG Çıktısı

Bu proje kapsamında geliştirilen görselleştirme sistemi, teknik bulguların sadece metinsel değil aynı zamanda görsel olarak da takip edilebilmesini mümkün kılar. Bu sayede kullanıcılar, bulgular arasındaki ilişkileri daha rahat anlayabilir, analiz sürecini yönetebilir ve belgeleri daha etkili sunabilir.

Not sistemiyle bütünleşik olarak çalışan bu yapı, kullanıcı tarafından oluşturulan her notu bir düğüm olarak ele alır. Düğümler arasında ebeveyn-çocuk ilişkisi tanımlanır ve tüm yapı interaktif bir ağaç (tree) şemasıyla görselleştirilir.

Ana Özellikler:

- D3.js ile ağaç yapısının SVG tabanlı dinamik çizimi
- Her notun başlık, içerik ve etiket bilgileriyle kart formatında gösterimi
- Düğümlerin tıklanarak düzenlenebilir olması (başlık, açıklama, ebeveyn değişimi)
- Yeni düğüm ekleme ve mevcutları silme fonksiyonu
- Zoom ve pan (kaydırma/yakınlaştırma) desteği
- Görselleştirilmiş ağaç yapısının SVG formatında dışa aktarımı

SVG Çıktısı Alma:

- Kullanıcı "SVG Kaydet" butonuna tıkladığında, DOM'daki SVG element XML'e dönüştürülür

- Dosya proje-adı-tree.svg olarak indirilir
- Dışa aktarılan bu dosya raporlarda, sunumlarda veya arşivlemelerde kullanılabilir

Kullanım Senaryoları:

- Zafiyetlerin görsel olarak hangi adımlar sonucu ortaya çıktığını takip etmek
- Yapay zeka tarafından oluşturulan notların hangi araçlara dayandığını görmek
- Bulgular arasında zaman-mekan bağlamı kurmak
- Raporlara görsellik ve profesyonellik katmak

Koddan Örnek:

```
const serializer = new XMLSerializer()
const source = serializer.serializeToString(svgElement)
const blob = new Blob([source], { type: 'image/svg+xml;charset=utf-8' })
```

Sonuç:

Bu yapı sayesinde siber güvenlik analizi sadece teknik bir işlem olmaktan çıkar, aynı zamanda stratejik ve görsel bir izleme sürecine dönüşür. Notların grafiksel sunumu, karmaşık analizleri anlaşılır hale getirir ve sunumlarda etkileyici çıktılar elde edilmesini sağlar.

9. SONUÇ VE ÖNERİLER

Bu çalışmada geliştirilen Güvenlik Paneli; terminal bilgisine ihtiyaç duymadan Kali Linux tabanlı araçlarla siber güvenlik analizi yapabilen, yapay zeka destekli öneri sistemi barındıran ve tüm süreçleri görselleştiren entegre bir platform sunmaktadır. Her seviyeden kullanıcıya hitap eden bu sistem, AI motoru ile güçlendirilmiş etkileşimli altyapısı sayesinde siber güvenlikte yeni bir kullanım deneyimi yaratmıştır.

Öne çıkan sonuçlar:

- Kullanıcılar komut satırı bilgisi olmadan ileri düzey araçları çalıştırabildi
- Yapay zeka, port analizinden çıkan çıktılara anlam yükleyerek öneriler sundu
- Not sistemi ve SVG tabanlı ağaç görselleştirme sayesinde bilgi takibi maksimum düzeye çıkarıldı
- Proje dosyaları arasında akış kesilmeden sürdürülebilir bir analiz süreci sağlandı

Öneriler (ileri seviye ve yaratıcı perspektiflerle):

1. 🌐 Gerçek zamanlı bulut senkronizasyonu: Her not ve çıktı, kullanıcı hesabına özel olarak bulutta saklanabilir. Bu sayede farklı makinelerden erişim mümkün olur.

2. 🧠 Kendi kendine öğrenen AI modülü: Kullanıcının verdiği yanıtlarla ("evet", "hayır") AI model eğitilebilir, zamanla daha isabetli yorumlar üretilebilir.

3. 🛰️ Uydu simülasyonu: Görselleştirme ekranında SVG çıktılar üç boyutlu uzay grafiğine dönüştürülerek, bulgular galaksi haritası gibi gösterilebilir.

4. 🤖 Saldırı simülasyonu modülü: Girilen bulgulara göre olası saldırı senaryoları animasyonlu olarak simüle edilebilir.

5. 🧩 AI destekli otomatize exploit öneri sistemi: AI yalnızca açıklama yapmaz, aynı zamanda bulunduğu açıklığa göre metasploit exploit modülleri önerir ve hatta parametrelerini hazırlar.

6. 📊 Otomatik PDF rapor oluşturucu: Tüm çıktılar, notlar, bulgular ve görseller tek bir tıklama ile profesyonel bir rapora dönüştürülebilir.

7. 🗣️ Sesli Komut ile Etkileşim: "Nmap ile tarama yap" dediğinizde, sistem ilgili aracı seçer, AI ile parametre oluşturur ve başlatır.

8. 🔐 Otomatik CVSS skorlama: AI tarafından tanımlanan zafiyetler, CVSS v3.1'e göre otomatik olarak puanlanabilir.

9. 🛰️ OSINT Entegrasyonu: Hedef IP adresi için açık kaynak istihbarat analizi sunulabilir (Shodan, Censys, ZoomEye API kullanarak).

10. 📡 Gerçek zamanlı tehdit istihbaratı: Yapay zeka motoru, kullanıcı tarafından tanımlanan her bulguyu otomatik olarak MITRE ATT&CK matrisindeki tekniklerle karşılaştırabilir. Örneğin, bir FTP anonim erişim bulgusu tespit edildiğinde, AI bunu "T1078 – Valid Accounts" veya "T1046 – Network Service Scanning" teknikleriyle ilişkilendirir. Ayrıca, AI çıktıları anlık olarak bir tehdit haritasına işlenebilir ve sistemde dinamik bir 'threat heatmap' oluşabilir. Bu harita SVG veya WebGL tabanlı görsel arayüzde zamanla değişen tehdit yoğunluğunu simüle eder. Gelişmiş sürümlerde bu yapı SIEM sistemleriyle entegre edilerek gerçek zamanlı saldırı korelasyonu sağlanabilir.

10. KURULUMU

Security-Panel projesi, Kali Linux ortamında siber güvenlik araçlarının yönetimini kolaylaştırmak ve yapay zeka destekli komut önerileri sunmak amacıyla açık kaynaklı olarak geliştirilmiştir ve güncellemeler ile geliştirilmeye devam edecektir.

Proje, herkesin katkısına, geliştirme fikirlerine ve önerilerine açıktır.

Kurulum veya kullanım sırasında herhangi bir problemle karşılaşırsanız, yorumlar bölümünde geri bildirimde bulunarak destek alabilirsiniz.

Kurulum için url: **https://www.bilgiveteknoloji.com/forum/public/d/85-securty-panel-kurulum-adimlari/5**

Açıklama	**Kod**
Sistemi Güncelleyin	sudo apt update sudo apt upgrade -y
Python + pip + git	sudo apt install python3 python3-pip python3-venv git -y
GPU Desteği (opsiyonel ama önerilir)	sudo apt install nvidia-cuda-toolkit -y
Ollama Kurulumu	curl -fsSL https://ollama.com/install.sh | sh
Ollama Servisini Başlat	ollama serve
Model İndir (örnek: llama3:8b-instruct-q4_0)	ollama pull llama3:8b-instruct-q4_0
Test Komutu	ollama run llama3:8b-instruct-q4_0 "Merhaba! Kitaplarımı işlemeye hazır mısın?"
Node.js ve npm kurulumu	sudo apt install nodejs npm -y node -v npm -v
Projeyi Klonla	git clone https://github.com/acikburak/Kali-Linux-Securty-Panel.git
Frontend bağımlılıklarını yükle	cd ~/Masaüstü/Kali-Linux-Securty-Panel/security-panel/ npm install

	npm install axios react react-router-dom d3 vite --save
Backend kurulumu	cd backend python3 -m venv venv source venv/bin/activate pip install Flask Flask-Cors ollama gradio_client ve gradio kurulumu: pip install gradio_client gradio kurulumu ve güncellemesi: pip install gradio pip install --upgrade gradio
Modelfile yapılandırması	cd ~/Masaüstü/Kali-Linux-Securty-Panel/security-panel/ ollama rm kali-fix ollama create kali-fix -f kali-fix.Modelfile
Frontend başlat	npm run dev
Backend başlat	cd backend source venv/bin/activate python3 app.py
Sistemi sonraki seferde tek komutla başlatmak için	bash start.sh

KAYNAKÇA

A. Proje ve Topluluk Bağlantıları

[1] GitHub Kaynağı: Kali Linux Security Panel (açık kaynaklı proje deposu). https://github.com/acikburak/Kali-Linux-Securty-Panel/tree/master

[2] Güvenlik Paneli Topluluk Desteği: https://www.bilgiveteknoloji.com/forum/public/t/g-venlik-taramas-y-netim-paneli

B. Teknolojik Kaynaklar

[1] Flask Web Framework. https://flask.palletsprojects.com/

[2] React Resmi Belgeleri. https://react.dev/

[3] D3.js Görselleştirme Kütüphanesi. https://d3js.org/

[4] Hugging Face Transformers. https://huggingface.co/

[5] Ollama AI. https://ollama.com/

[6] Gradio Framework. https://www.gradio.app/

[7] Nmap Network Mapper. https://nmap.org/

[8] FFUF - Fuzz Faster U Fool. https://github.com/ffuf/ffuf

[9] Dirb Web Scanner. https://tools.kali.org/web-applications/dirb

[10] WPScan WordPress Security Scanner. https://wpscan.com/

[11] Metasploit Framework. https://www.metasploit.com/

[12] MITRE ATT&CK Framework. https://attack.mitre.org/

[13] KaiMemory JSON Yapısı – chatbot_interface.py içinde özel yapılandırma

[14] SVG Serialization Yapısı – TreeView.jsx içinde XML export mantığı

[15] Proje dosyaları: app.py, chatbot_interface.py, ScanDashboard.jsx, TargetSelection.jsx, TreeView.jsx, Home.jsx, NewProject.jsx (OpenAI tarafından analiz edildi)

[16] ChatGPT AI destekli öneriler ve içerik oluşturma süreci (OpenAI, 2025)

[17] Linux terminal komutları ve test çıktıları (Kali Linux ortamında test edilmiştir)

[20] Güvenlik Paneli Topluluk Desteği: https://www.bilgiveteknoloji.com/forum/public/t/g-venlik-taramas-y-netim-paneli