🛡️ DLP Risk Analyzer - Proje Dokümantasyonu

Forcepoint DLP Adaptive Risk Protection Sistemi | Versiyon 1.0 | Oluşturulma: Aralık 2024

📋 İçindekiler

1. Kullanılan Teknolojiler

🐘 **PostgreSQL**

🔴 **Redis**

Amaç: Ana veritabanı sistemi

Incident'lar, kullanıcılar, AI analizleri ve sistem ayarları burada saklanır. Entity Framework Core ile ORM desteği sağlanır.

Tablolar: Incidents, Users, AIBehavioralAnalyses, SystemSettings, AuditLogs

Amaç: Mesaj kuyruğu (Stream)

DLP API'den çekilen incident'lar önce Redis Stream'e yazılır. Bu sayede veri kaybı önlenir ve asenkron işleme sağlanır.

Stream: `dlp:incidents`

.NET 8 / C#

Amaç: Backend API

ASP.NET Core Web API ile RESTful servisler sunulur. Dependency Injection, Entity Framework Core, Background Services kullanılır.

Next.js / React

Amaç: Frontend Dashboard

TypeScript ile yazılmış modern dashboard. Plotly.js ile grafikler, Tailwind CSS ile stil, Axios ile API iletişimi.

Forcepoint DLP API

Amaç: Veri kaynağı

Forcepoint DLP Manager REST API v1 üzerinden incident verileri çekilir. JWT token tabanlı kimlik doğrulama kullanılır.

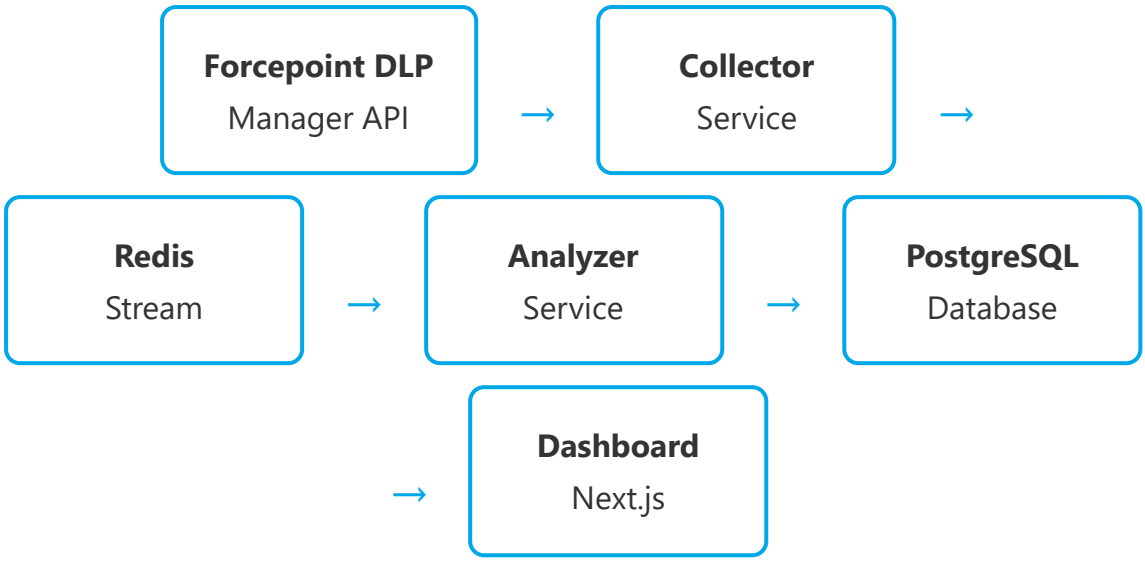
OpenAI / Azure OpenAI

Amaç: AI Analiz

Davranış anomalilerinin açıklaması ve önerileri için GPT modelleri kullanılır. Opsiyonel entegrasyon.

2. Sistem Mimarisi

Veri Akış Diyagramı



Proje Yapısı

Proje	Açıklama	Port
`DLP.RiskAnalyzer.Collector`	DLP API'den veri toplama servisi	-
`DLP.RiskAnalyzer.Analyzer`	Ana API servisi (Backend)	5062
`DLP.RiskAnalyzer.Shared`	Ortak modeller ve sabitler	-
`DLP.RiskAnalyzer.Dashboard`	WPF Desktop Uygulaması	-
`dashboard`	Next.js Web Dashboard	3002

3. DLP API'den Veri Toplama

Dosya: `DLP.RiskAnalyzer.Collector/Services/DLPCollectorService.cs`

1 Kimlik Doğrulama (Authentication)

```
POST   /dlp/rest/v1/auth/access-token
```

```
// İstek Headers
Headers:
  username: <DLP kullanıcı adı>
  password: <DLP şifresi>

// Yanıt
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5c...",
  "access_token_expires_in": 3600
}
```

2 Incident'ları Çekme

```
POST   /dlp/rest/v1/incidents/
```

```
// İstek Headers
Authorization: Bearer <access_token>
Content-Type: application/json

// İstek Body
{
  "type": "INCIDENTS",
  "from_date": "01/12/2024 00:00:00",
  "to_date": "10/12/2024 23:59:59",
  "start": 0,
  "limit": 100
}

// Yanıt
{
  "incidents": [
    {
      "id": 12345,
      "severity": "HIGH",
      "source": {
        "login_name": "john.doe@company.com",
        "department": "Finance"
      },
      "incident_time": "05/12/2024 14:30:00",
      "channel": "Email",
      "policies": "Data Loss Prevention"
    }
  ],
```

```
  "total": 150
}
```

4. Redis Stream İşlemleri

Dosya: `DLPCollectorService.cs` (Satır 233-260)

1 Stream'e Yazma

```
// Redis Stream'e incident yazma
var streamName = "dlp:incidents";
var fields = new NameValueEntry[]
{
    new("user", incident.UserEmail),
    new("department", incident.Department ?? ""),
    new("severity", incident.Severity.ToString()),
    new("data_type", incident.DataType ?? ""),
    new("timestamp", incident.Timestamp.ToString("O")),
    new("policy", incident.Policy ?? ""),
    new("channel", incident.Channel ?? "")
};

await db.StreamAddAsync(streamName, fields);
```

2 Stream'den Okuma

Dosya: `DatabaseService.cs` (Satır 71-185)

```
// Consumer Group ile okuma
var consumerGroup = "analyzer";
var messages = await db.StreamReadGroupAsync(
    streamName,
    consumerGroup,
    consumerName,
    ">",        // Yeni mesajları oku
    count: 100  // Max 100 mesaj
);
```

```
// İşleme sonrası acknowledge
await db.StreamAcknowledgeAsync(streamName, consumerGroup, message.Id);
```

5. PostgreSQL Veritabanı İşlemleri

Incident Tablosu

Alan	Tip	Açıklama
Id	int	Primary Key
UserEmail	string	Kullanıcı e-postası
Department	string?	Departman
Severity	int	Ciddiyet (1-5)
DataType	string?	Veri tipi (PII, PCI...)
Timestamp	DateTime	Olay zamanı
Policy	string?	Tetiklenen politika
Channel	string?	Kanal (Email, USB...)
RiskScore	int?	Risk skoru (0-100)
RepeatCount	int	Tekrar sayısı
DataSensitivity	int	Veri hassasiyeti

6. Risk Skoru Hesaplama

Dosya: `DLP.RiskAnalyzer.Shared/Services/RiskAnalyzer.cs`

Hesaplama Formülü

```
RiskScore = (Severity × 3) + (RepeatCount × 2) + (DataSensitivity × 5)

// Maksimum 100 ile sınırlandırılır
return Math.Min(100, baseScore);
```

Risk Seviyeleri

Skor Aralığı	Seviye	Aksiyon
91-100	**Critical**	Block (Engelle)
61-90	**High**	Encrypt (Şifrele)
41-60	**Medium**	Confirm Prompt (Onay İste)
0-40	**Low**	Audit (Kaydet)

Veri Hassasiyeti Eşikleri

```
PII (Kişisel Bilgi)   → Threshold: 8
PCI (Ödeme Kartı)     → Threshold: 9
Confidential (Gizli)  → Threshold: 7
```

📊 Skor Hesaplama Detayları

Dosya: `DLP.RiskAnalyzer.Shared/Constants/RiskConstants.cs` ve `RiskAnalyzer.cs`

> ⚠️ **Önemli:** Sistem tüm DLP politikalarından bağımsız olarak çalışır. Politikalar Forcepoint DLP'den alınır, ancak risk skoru aşağıdaki 3 faktöre göre hesaplanır.

1️⃣ Severity (Ciddiyet) - Ağırlık: ×3

Forcepoint DLP API'den gelen `severity` değeri string olarak alınır ve sayıya dönüştürülür:

API Değeri	Sayısal Değer	Skor Katkısı
LOW	1	1 × 3 = 3
MEDIUM	2	2 × 3 = 6
HIGH	3	3 × 3 = 9
CRITICAL	4	4 × 3 = 12

2️⃣ RepeatCount (Tekrar Sayısı) - Ağırlık: ×2

Aynı kullanıcının benzer olayları tekrarlama sayısı. Bu değer veritabanında takip edilir.

Tekrar Sayısı	Skor Katkısı	Açıklama
0-2	0-4	Normal davranış
3-5	6-10	Dikkat edilmeli
6-10	12-20	Şüpheli aktivite
10+	20+	IOB-311 (Stockpiling) tetiklenir

3️⃣ DataSensitivity (Veri Hassasiyeti) - Ağırlık: ×5

Veri tipi (DataType) alanına göre hassasiyet seviyesi belirlenir:

Veri Tipi	Eşik Değeri	Skor Katkısı (×5)
PCI (Ödeme Kartı)	9	9 × 5 = 45
PII (Kişisel Bilgi)	8	8 × 5 = 40
Confidential (Gizli)	7	7 × 5 = 35
Financial (Finansal)	6	6 × 5 = 30

Health (Sağlık)	6	6 × 5 = 30
Internal (Şirket İçi)	4	4 × 5 = 20
Public (Genel)	1	1 × 5 = 5

📐 Örnek Hesaplama

```
// Senaryo: Yüksek ciddiyetli, 3 kez tekrarlanan PII verisi sızıntısı
Severity = HIGH (3)       → 3 × 3 = 9
RepeatCount = 3           → 3 × 2 = 6
DataSensitivity = PII (8) → 8 × 5 = 40
──────────────────────────────────────
Toplam Risk Skoru         = 9 + 6 + 40 = 55 (Medium Risk)
```

🎯 Politika Aksiyonları Matrisi

Dosya: `RiskAnalyzer.cs` - `GetPolicyAction()` metodu

Risk seviyesi ve kanal kombinasyonuna göre otomatik aksiyon önerilir:

Risk Seviyesi	Email	USB	Cloud	Web	Print
Critical (91-100)	Block	Block	Block	Block	Block
High (61-90)	Encrypt	Encrypt	Encrypt	Encrypt	Notify
Medium (41-60)	Confirm	Confirm	Confirm	Confirm	Audit
Low (0-40)	Audit	Audit	Audit	Audit	Audit

🔍 IOB (Indicator of Behavior) Tespiti

Dosya: `RiskAnalyzer.cs` - `DetectIOB()` metodu

Belirli davranış kalıpları tespit edildiğinde IOB kodları atanır:

IOB Kodu	Açıklama	Tetikleme Koşulu
`IOB-511`	Personal Email Exfiltration	Email kanalı + kişisel domain (@company.com dışı)
`IOB-299`	USB Data Transfer	USB kanalı + Severity ≥ 7
`IOB-811`	Cloud Upload	Cloud kanalı + DataSensitivity ≥ 8
`IOB-311`	Anomalous File Copying (Stockpiling)	RepeatCount ≥ 10
`IOB-280`	Agent Tampering	Policy içinde "Agent" + Severity ≥ 8

```
// IOB tespit örneği
if (incident.Channel == "USB" && incident.Severity >= 7)
{
    iobs.Add("IOB-299"); // USB upload
}

if (incident.RepeatCount >= 10)
{
    iobs.Add("IOB-311"); // Stockpiling
}
```

📁 Sabit Değerler (Constants)

Dosya: `DLP.RiskAnalyzer.Shared/Constants/RiskConstants.cs`

Risk Skor Eşikleri

```
public static class RiskScores
{
    public const int CriticalThreshold = 91;
    public const int HighThreshold = 61;
    public const int MediumThreshold = 41;
    public const int MaxScore = 100;
    public const int MinScore = 0;
}
```

Desteklenen Kanallar

```
public static class Channels
{
    public const string Cloud = "Cloud";
    public const string CloudStorage = "Cloud Storage";
    public const string Email = "Email";
    public const string NetworkShare = "Network Share";
    public const string RemovableStorage = "Removable Storage";
    public const string Printer = "Printer";
}
```

Veri Hassasiyeti Tipleri

```
public static class DataSensitivity
{
    public const string PII = "pii";
    public const string Personal = "personal";
    public const string PCI = "pci";
    public const string Credit = "credit";
    public const string Confidential = "confidential";
}
```

> ✅ **Sonuç:** Tüm eşik değerleri ve politika aksiyonları `RiskConstants.cs` dosyasında merkezi olarak tanımlanmıştır. Bu sayede değişiklikler tek noktadan yapılabilir.

7. AI Davranış Analizi

Dosya: `DLP.RiskAnalyzer.Analyzer/Services/BehaviorEngineService.cs`

Z-Score Anomali Tespiti

```
// Z-score hesaplama: z = (x - μ) / σ
// x: Mevcut değer, μ: Ortalama, σ: Standart sapma

var incidentCountZ = (currentMean - baselineMean) / baselineStdDev;
var severityZ = (currentAvgSeverity - baselineAvgSeverity) / baselineStdDevSeverity;
```

Anomali Seviyeleri

Z-Score	Seviye	Risk Skoru
≥ 3.0	Kritik	100
≥ 2.0	Yüksek	80
≥ 1.0	Orta	50
< 1.0	Düşük	30

AI Model Entegrasyonu

Sistem üç AI sağlayıcısını destekler:

- **OpenAI:** GPT-4, GPT-4o-mini
- **Azure OpenAI:** Kurumsal Azure entegrasyonu
- **Local:** Statik açıklama (AI olmadan)

8. Olay Düzeltme (Remediation)

Dosya: `DLP.RiskAnalyzer.Analyzer/Services/RemediationService.cs`

```
POST   /dlp/rest/v1/incidents/update
```

```
// İstek Body
{
  "incidentId": "12345",
  "action": "block",           // block, encrypt, notify, audit
  "reason": "Data exfiltration attempt",
  "notes": "User attempted to send PII via personal email"
}
```

9. Dashboard (Frontend)

Dizin: `dashboard/`

Sayfalar

Sayfa	Yol	Açıklama
Dashboard	`/`	Ana sayfa, özet grafikler
Investigation	`/investigation`	Kullanıcı inceleme, timeline
Users	`/users`	Kullanıcı yönetimi
Reports	`/reports`	Rapor oluşturma
AI Behavioral	`/ai-behavioral`	AI anomali analizi
Settings	`/settings`	Sistem ayarları

API İletişimi

```
// Örnek API çağrısı
const response = await axios.get(`${apiUrl}/api/incidents`, {
    params: {
        start_date: '2024-12-01',
        end_date: '2024-12-10',
        limit: 100,
        order_by: 'risk_score_desc'
```

```
    }
});
```

10. API Endpoint'leri

Incident Endpoint'leri

- GET `/api/incidents` - Tüm incident'ları listele
- GET `/api/incidents/{id}` - Tek incident detayı
- POST `/api/incidents/seed-sample-data` - Test verisi oluştur

Risk Endpoint'leri

- GET `/api/risk/daily-summary` - Günlük özet
- GET `/api/risk/department-summary` - Departman özeti
- GET `/api/risk/user-list` - Kullanıcı risk listesi

AI Behavioral Endpoint'leri

- GET `/api/ai-behavioral/overview` - AI analiz özeti
- GET `/api/ai-behavioral/entity/{type}/{id}` - Entity analizi
- POST `/api/ai-behavioral/analyze` - Yeni analiz başlat

DLP Test Endpoint'leri

POST `/api/dlptest/connection` - Bağlantı testi

POST `/api/dlptest/auth` - Kimlik doğrulama testi

POST `/api/dlptest/incidents` - Incident çekme testi

> 📝 **Not:** Bu dokümantasyon tarayıcıda PDF olarak yazdırılabilir (Ctrl+P veya Cmd+P).