UTM DASHBOARD

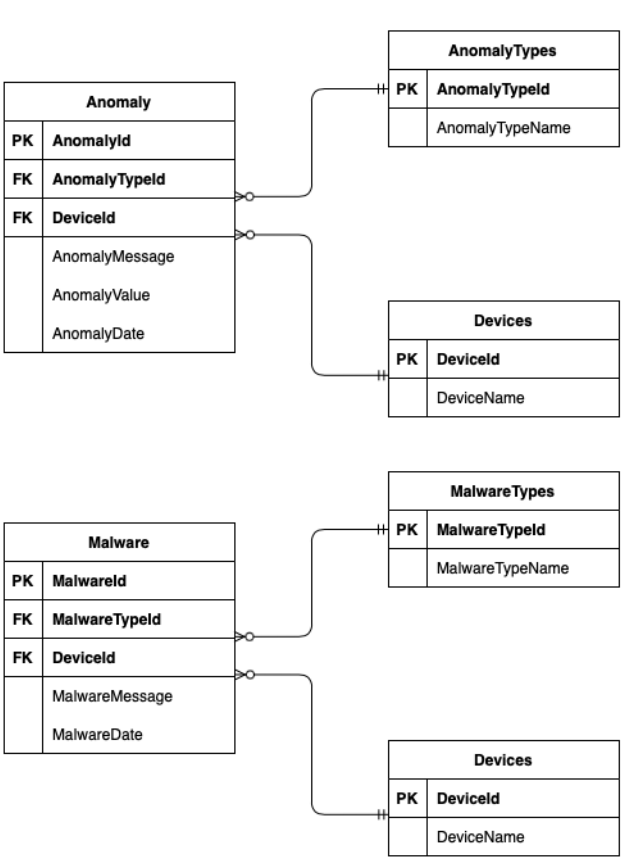
Projede son düzenlemelerin üzerine ChatGPT'den sahadan alınmış olabilecek malware ve anomali log mesajlarına benzer her tip için 4'er mesaj taslağı alarak C++ ile mini bir program yazarak sahadan alınmış loğların düzeninde random tip ve mesajlarla doldurarak benzer, 400'er adet log kaydı bulunan log dosyaları oluşturduk ve bunları projeye import ettik. Program başlatıldığında veritabanında otomatik olarak tip tablolarına tipler ve ilgili tablolarına ise loglar kaydedilmektedir. Anomaly grafiğini daha anlamlı olacağını düşünerek farklı bir grafikle değiştirdik. Anomaly logları görünümü aşağıdaki şekilde metin belgesinde bulunmaktadır.

```
2022-09-03 09:10:39 - 7UT85 - 28 - Point - Beklenenden daha düşük
```

Malware logları ise şu şekilde bulunmaktadır.

```
2022-09-01 00:11:16 - 7UT85 - Rootkit - Sayaç sistemine yerleşen bir rootkit,
```

Dosyalardan loglar çekilerek database'e aktarılmış ve sonrasında grafiklere aktarılmıştır. ERD diyagramları ise aşağıdaki şekildedir.



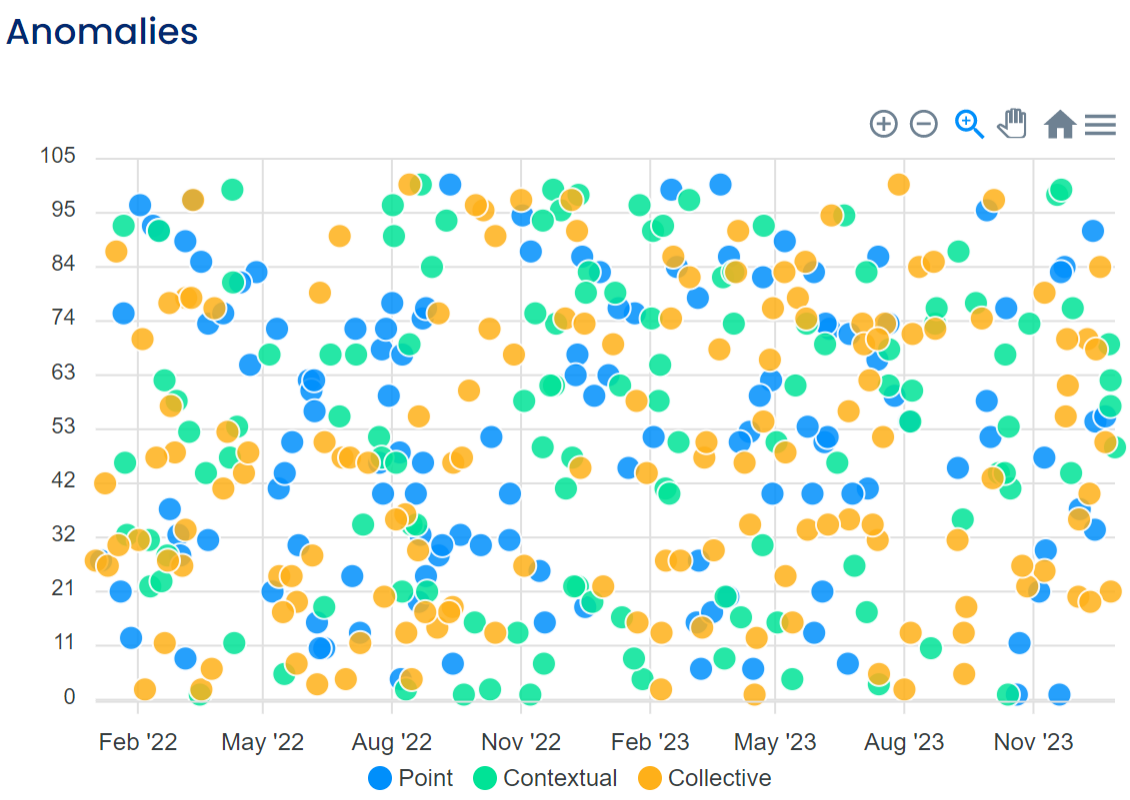
AnomalyTypes Table	
AnomalyTypeId	**AnomalyTypeName**
1	Point
2	Contextual
3	Collective

Devices Table	
DeviceId	**DeviceName**
1	7UT85
2	6MD85

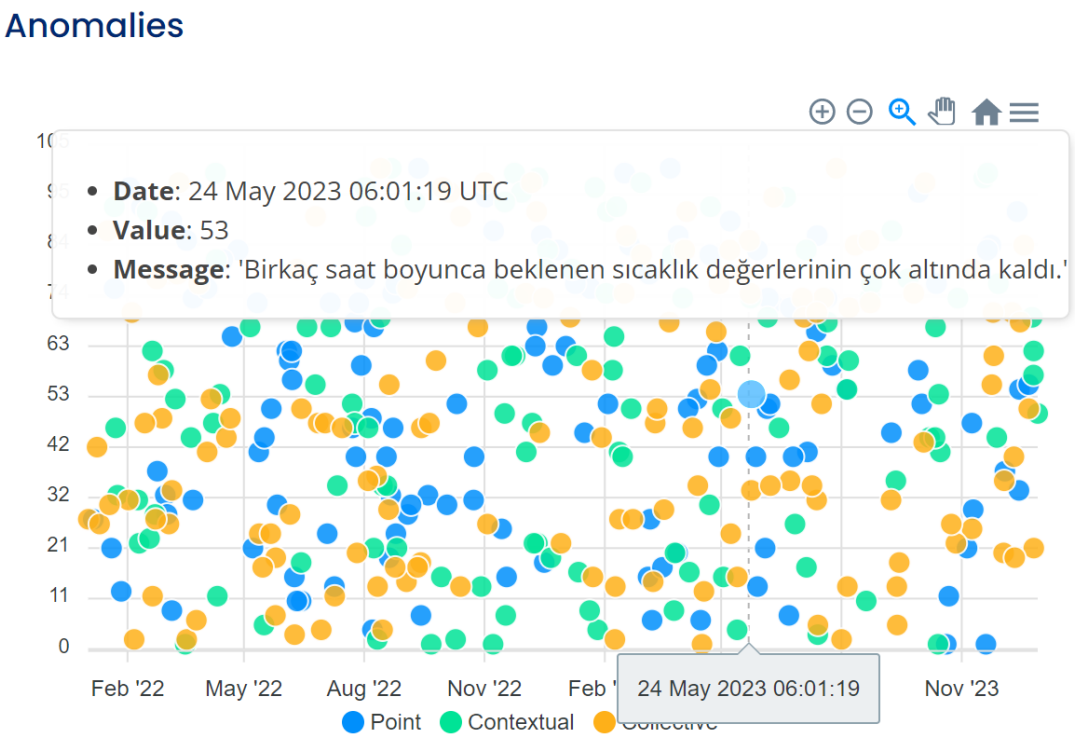
MalwareTypes Table	
MalwareTypeId	**MalwareTypeName**
1	Worm
2	Rootkit
3	Spyware
4	Adware

Devices Table	
DeviceId	**DeviceName**
1	7UT85
2	6MD85

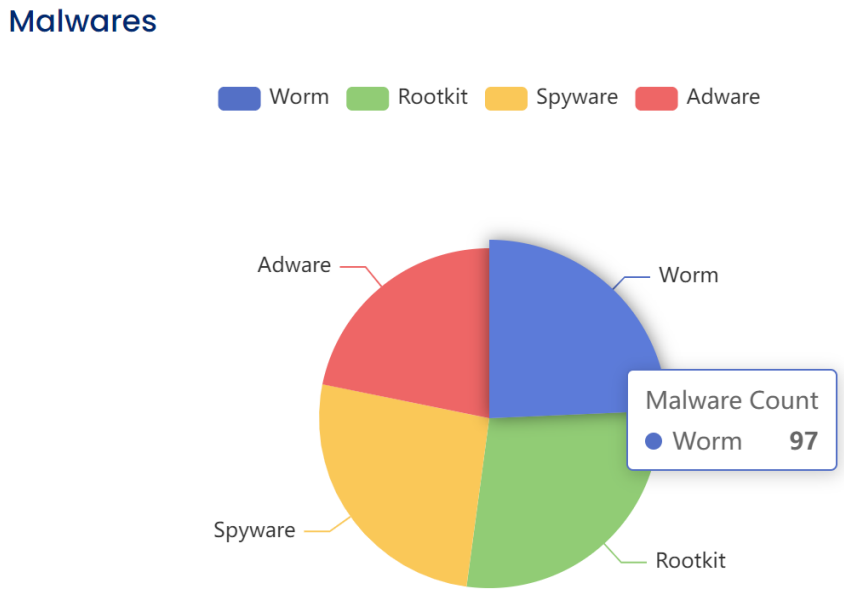
Aktarılan loglarla oluşan yeni anomali grafiği görünümü aşağıdaki şekildedir. Grafik üzerinde tarihsel yakınlaştırma uzaklaştırma gibi işlemler veya herhangi bir tipi veya tipleri kapatarak yalnızca diğer tipteki anomalileri görüntüleyebilme işlemleri yapılabilmektedir.



Bunun yanı sıra aktivitelerden yani grafikteki baloncukların üzerine gelindiğinde ise anomali ile ilgili bilgiler verilmektedir.



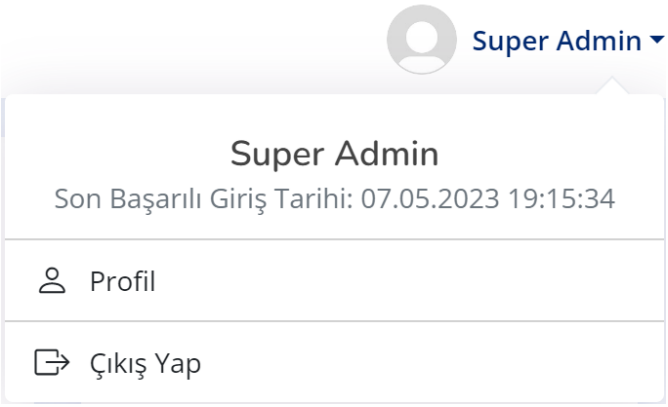
Aktarılan loglarla oluşan malware grafiği görünümü ise aşağıdaki şekildedir. Grafikte ilgili malware tiplerine ait toplam malware adetleri görünmektedir.



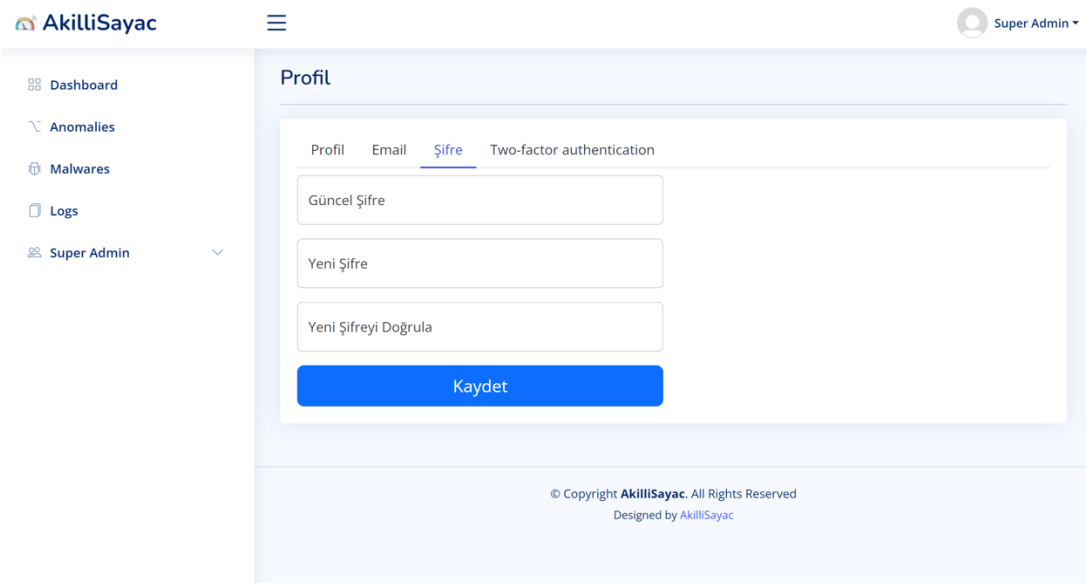
Bunlara ek olarak kullanıcıların son giriş ve son parola değiştirme tarihleri için veritabanında fieldlar oluşturduk.

```
[Required]
[DataType(DataType.Date)]
6 references
public DateTime LastPasswordChangedDate { get; set; }
[Required]
[DataType(DataType.Date)]
2 references
public DateTime LastLoginDate { get; set; }
```

Görüntülenebilmesi için ise son başarılı giriş tarihini kullanıcı paneline ekledik.



Son parola değiştirme tarihini kullanarak parola geçerlilik süresini 90 gün olarak ayarladık. Son parola değiştirme tarihinden 90 gün geçtikten sonra kullanıcı giriş yaptığında otomatik olarak şifre değiştirme ekranına yönlendirilmektedir.



Bunu da bir middleware ekleyerek yaptık. (İşlemleri gerçekleştirmek için oluşturduğumuz sınıfın ismi “ChangePasswordResourceFilter”)

```
builder.Services.AddScoped<ChangePasswordResourceFilter>();

builder.Services.AddMvc(o =>
{
    o.Filters.Add(typeof(ChangePasswordResourceFilter));
});
```

Tasarımda hazırlamış olduğumuz güvenli yazılım geliştirme ilkelerine ait hazırladığımız checklistte ise son rapordan sonra gelmiş olduğumuz durum aşağıdadır.

✓ Yazılımda kullanılan tüm varsayılan değerler güvenliği arttıracak şekilde seçilmeli ve her bir nesnenin varsayılan erişimi hiç olmalı

✓ Parolalar için bir en uzun geçerlilik süresi tanımlanmış olmalıdır.

✓ Uygulama kullanıcının son başarılı oturum açma tarih ve saatini görüntülemelidir.