ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ
MÜHENDİSLİK MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİÇİMSEL DİLLER VE OTOMATA DERSİ

Konu: Security Information And Event Management (SIEM) vs XML Relation

Proje Adı: Log Analyzer

Grup No: 2

Hazırlayanlar:
Doğukan KIYIKLIK – 152120211104
Yusuf Eren HOŞGÖR -152120211123
Salih Enes ÜNAL – 152120211061 Safiullah
SEDİQİ - 152120211031

NİSAN 2024

İçindekiler

1. SIEM NEDİR? 2
1.1 SIEM'in Teknolojideki Yeri: 2
1.2 SIEM'in Kullanım Alanları: 3
2. XML NEDİR 3
2.1 XML'in Teknolojideki Yeri: 3
2.2 XML'in Kullanım Alanları: 3
2.3 XML ve SIEM İlişkisi 3
3. PROBLEM ÖZELLİKLERİ 4
3.1 Probleme Çözüm Yaklaşımı 4
3.2 Çözüm Adımları 5
3.3 Tasarlanan DFA 6
3.4 XML formatını doğrulamak için tasarlanan DTD: 7
4.Projenin Biçimsel Diller ve Otomata ile İlişkisi 7
5.Projede Kullanılacak Araç: C# 8
5.1 Arayüz Tasarımı 8
5.2 Veri Loglarının İncelenmesi 8
6. UYGULAMA YAZILIMI 8
6.1 Log Sınıfı: 8
6.2 LogManager Sınıfı: 9
6.3 Dfa Sınıfı: 10
7.PROJE EKİBİ DEĞERLENDİRME 11
8.KAYNAKÇA 11

1. SIEM NEDİR?

SIEM sisteminin açılımı "Security Information And Event Management"dır. Türkçe karşılığı ise "Güvenlik Bilgisi ve Onay İzleme Sistemi"dir. SIM (Security Information Management) ve SEM (Security Event Management) güvenlik teknolojilerinin birleşiminden ortaya çıkan sistemlerdir. Bu sistemler ile her iki güvenlik teknolojisi tek bir yerde toplanmıştır. Böylelikle güvenlik bilgisi ve onay izleme sistemi olan SIEM ortaya çıkmıştır. SIEM terimi ve baş harfi, 2005 yılında Gartner'dan Mark Nicolett ve Amrit Williams tarafından icat edilmiştir. [1]

1.1 SIEM'in Teknolojideki Yeri:

SIEM, kuruluşların güvenlik altyapılarını korumak için kritik bir rol oynar. Bilgisayar korsanları ve kötü niyetli yazılım gibi tehditlerin artmasıyla birlikte, organizasyonlar güvenlik olaylarını izlemek ve bunlara yanıt vermek için SIEM çözümlerine yönelmektedir. [2]

1.2 SIEM'in Kullanım Alanları:

1.2.1 Günlük Yönetimi: SIEM sistemleri devasa boyutlardaki verileri tek bir yerde toplayıp düzenleyerek bir tehdit, saldırı veya ihlal belirtisi olup olmadığını inceler.

1.2.2 Olaylar Arasındaki Bağıntılar: SIEM, olası tehditlerin hızlı bir şekilde algılanıp giderilmesi için veriler düzenlenerek aralarındaki ilişki ve bağlantılar belirlenir.

1.2.3 Olay İzleme ve Olay Yanıtı: SIEM teknolojisi, bir kurumun ağındaki güvenlik olaylarını izler ve bir olayla ilgili tüm etkinlikler için uyarı ve denetim seçenekleri sunar.

2. XML NEDİR

Genişletilebilir İşaretleme Dili (XML), verileri paylaşılabilir bir şekilde tanımlamanıza ve saklamanıza olanak tanır. XML, web siteleri, veri tabanları ve üçüncü taraf uygulamaları gibi bilgisayar sistemleri arasında bilgi alışverişini destekler. Önceden tanımlanmış kurallar, verileri herhangi bir ağ üzerinden XML dosyaları olarak iletmeyi kolaylaştırır, böylece alıcı verileri doğru ve verimli bir şekilde okumak için bu kuralları kullanabilir.[3]

2.1 XML'in Teknolojideki Yeri:

XML, geniş bir teknoloji yelpazesinde kullanılır. Web uygulamalarından veri alışverişine, belge yönetiminden veri depolamaya kadar birçok alanda XML'in kullanımı yaygındır.

Özellikle veri entegrasyonu ve veri paylaşımı gerektiren sistemlerde XML tercih edilir.

2.2 XML'in Kullanım Alanları:

İşletmeler arası işlemleri destekleme: Bir şirket başka bir şirkete mal veya hizmet sattığında, iki işletmenin maliyet, teknik özellikler ve teslimat programları gibi bilgileri paylaşması gerekir. Genişletilebilir İşaretleme Dili (XML) ile gerekli tüm bilgileri elektronik olarak paylaşabilir ve herhangi bir insan müdahalesi olmaksızın karmaşık anlaşmaları otomatik olarak gerçekleştirebilirler.

Veri bütünlüğünü koruma: XML, verilerin açıklamasıyla birlikte verileri aktarmanıza izin vererek veri bütünlüğü kaybını önler. Aşağıdakileri yapmak için bu açıklayıcı bilgileri kullanabilirsiniz:

- Veri doğruluğunu doğrulama
- Farklı kullanıcılar için veri sunumunu otomatik olarak özelleştirme
- Verileri birden çok platformda tutarlı bir şekilde depolama

Arama verimliliğini artırma: Arama motorları gibi bilgisayar programları, XML dosyalarını diğer belge türlerinden daha verimli ve hassas bir şekilde sıralayabilir ve kategorize edebilir. Örneğin, kelime işareti bir isim veya fiil olabilir. Arama motorları, XML etiketlerine dayanarak ilgili arama sonuçları için işareti doğru bir şekilde kategorilere ayırabilir. Böylece XML, bilgisayarların doğal dili daha verimli bir şekilde yorumlamasına yardımcı olur.[4],

2.3 XML ve SIEM İlişkisi

XML (Genişletilebilir İşaretleme Dili) ve SIEM (Güvenlik Bilgi ve Olay Yönetimi) arasında birkaç açıdan ilişki bulunmaktadır, özellikle güvenlik bilgileri ve olaylarını yönetme bağlamında. İşte XML ve SIEM arasındaki ilişki hakkında bazı temel noktalar:

Veri Temsili: XML genellikle yapılandırılmış verileri temsil etmek için kullanılır, bunlar arasında güvenlikle ilgili bilgiler de bulunmaktadır. SIEM sistemleri, günlükler, uyarılar ve yapılandırma bilgileri gibi güvenlik olayı verilerini depolamak ve değiştirmek için XML'i kullanabilir.

Etkileşim: XML, veri değişimi için standart bir format sağlar, bu da farklı SIEM sistemlerinin ve güvenlik araçlarının iletişim kurmasını ve bilgi paylaşmasını kolaylaştırır. Bu etkileşim, karmaşık ortamlarda etkili güvenlik yönetimi için önemlidir.

Günlük Yönetimi: XML, çeşitli sistemler ve uygulamalar tarafından üretilen günlük dosyalarını yapılandırmak için kullanılabilir. SIEM sistemleri, bu XML biçimindeki günlükleri ayrıntılı bilgi çıkarmak ve güvenlik olaylarını analiz etmek için analiz edebilir.

Normalleştirme: SIEM sistemleri genellikle güvenlik olayı verilerini standart hale getirme ve kategorize etme amacıyla normalleştirme kullanır. XML, verilerin nasıl temsil edileceği ve işleneceği konusunda tutarlılık sağlamak için normalleştirme sürecinin bir parçası olarak kullanılabilir.

Özelleştirme: XML'in genişletilebilir olması, SIEM sistemlerinin veri biçimlerini ve yapılarını özelleştirerek belirli güvenlik gereksinimlerini karşılamasına olanak tanır. Bu esneklik, farklı ortamlara ve güvenlik ihtiyaçlarına uyum sağlamak için önemlidir.

3. PROBLEM ÖZELLİKLERİ

Günümüz dijital çağında, bilgisayar ile harici cihazlar arasında veri transferini yönetmek ve güvence altına almak büyük önem taşımaktadır. Yetkisiz veya şüpheli USB faaliyetleri, veri ihlalleri veya kötü amaçlı yazılım bulaşmaları gibi ciddi güvenlik riskleri oluşturabilir. Bu sorunları gidermek amacıyla, basit bir Güvenlik Bilgi ve Olay Yönetimi (SIEM) uygulaması tasarlanmıştır. Bu uygulama, USB faaliyetlerinin log kayıtlarını alarak bu logları bir Deterministik Sonlu Otomat (DFA) ile analiz eder. Uygulama, bilgisayardan büyük dosya transferleri, bilgisayara aynı anda bağlanan cihaz sayısı, USB'ye dosya kopyalandıktan sonra dosyanın bilgisayardan silinmesi ve USB'den bilgisayara dosya aktarımı gibi kritik olayları tespit ederek raporlamayı amaçlamaktadır.

3.1 Probleme Çözüm Yaklaşımı

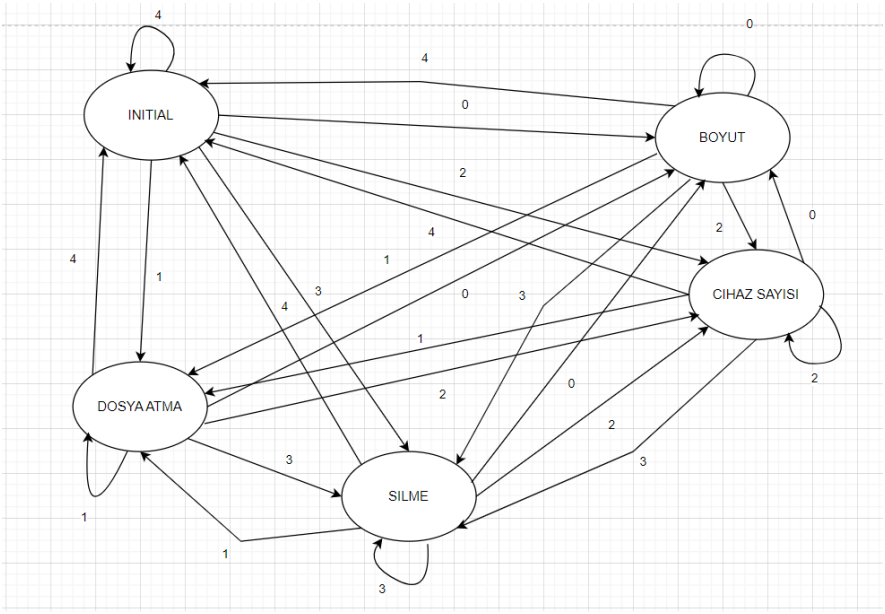
Projede, Windows işletim sisteminde USB ile ilgili log kayıtlarını elde edemediğimiz 'USB Radar' isimli bir uygulama kullanarak bilgisayara bağlı USB cihazlarının log kayıtlarını elde ettik. Daha sonra, belirli kritik durumları kontrol etmek amacıyla bir Deterministik Sonlu Otomat (DFA) tasarladık. Bu kritik durumlar arasında bilgisayardan büyük dosya transferleri, bilgisayara aynı anda bağlanan cihaz sayısı, USB'ye dosya kopyalandıktan sonra dosyanın bilgisayardan silinmesi ve USB'den bilgisayara dosya aktarımı gibi durumlar yer alıyor. Tasarladığımız DFA'i XML formatında depolayarak, farklı DFA'lerin de kolayca koda entegre edilmesini sağladık. Böylece, yeni bir DFA tasarlandığında kodda büyük değişiklikler yapmaya gerek kalmıyor. Ayrıca, bu XML'in belirli bir formatta yazılması için bir DTD (Document Type Definition) tasarlandı. Ardından, bu DFA koda entegre edilerek çalıştırıldı ve çeşitli kritik uyarıların gerçekleşmesi durumunda kullanıcı bilgilendirildi. Bu sayede, güvenlik açısından önemli olan USB faaliyetlerini etkin bir şekilde izlenildi ve gerekli önlemleri alabilme imkanı sağlandı.

3.2 Çözüm Adımları

1. Log Dosyalarının Elde Edilmesi:
    - USB Radar uygulaması üzerinden elde edilen log dosyası program tarafından okunur.
2. Log Dosyasının Analizi:
    - Okunan log dosyası, oluşturulan ayrıştırma metodu ile ayrıştırılır. Her bir log, log listesine eklenir. Bu sayede, her bir log nesnesi gerekli durumları kontrol aşamasında kolaylık sağlar.
3. DFA'nın XML Formatında Depolanması:
    - DFA, belirli taglar altında tanımlanır. Bu sayede, log dosyalarını kontrol edecek mekanizma daha esnek bir yapı haline gelir.
4. DFA'nın XML'den Okunması:
    - XMLReader sınıfı tarafından, XML formatı program için uygun forma dönüştürülür.
5. DFA'nın Çalıştırılması:
    - Process_Log metodu, DFA yoluyla çalıştırılır.
6. Arayüz Tasarımı:
    - Yapılan işlemler ve uyarı mesajları, düzgün bir arayüz içerisinde kullanıcıya sunulur.

Bu adımlar sayesinde, log dosyalarının elde edilmesinden analizine, DFA'nın XML formatında depolanmasından çalıştırılmasına ve kullanıcıya sunulmasına kadar tüm süreç etkin ve verimli bir şekilde yönetilir.

3.3 Tasarlanan DFA



Tasarlanan DFA'de bilgisayara USB cihaz bağlanamsı halinde bazı kritik olayları kontrol ediyoruz.

Kontrol edilen olaylar:

- **Initial** = Herhangi bir kritik olayın gerçekleşmediği durum.
- **Boyut** = Bilgisayardan alınan dosyanın belirli bir boyuttan fazla olması halinde gerçekleşen durum.
- **Cihaz Sayısı** = Bilgisayara aynı anda bağlanan cihaz sayısı belirli bir sınırın üzerindeyse gerçekleşen durum.
- **Silme** = Bilgisayardan USB'ye alınan dosyanın masaüstünden silinemsi halinde gerçekleşen durum.
- **Dosya Atma** = USB'den bilgisayara herhangi bir dosya atma durumunda gerçekleşen durum.

DFA'deki alfabe:

- **0** = Eğer USB'ye atılan dosya boyutu belirtilen sınırın üzerinde olması
- **1** = USB'den bilgisayara dosya atılması
- **2** = Bilgisayara aynı anda bağlanan cihaz sayısının belirli bir sayının üzerinde olması
- **3** = USB'ye atılan dosaynın masaüstünen silinmesi
- **4** = Normal

3.4 XML formatını doğrulamak için tasarlanan DTD:

```
<!ELEMENT DFA (States, Alphabet, Transitions)>
<!ELEMENT States (state+)>
<!ELEMENT state (state_id, name, initial, final)>
<!ELEMENT state_id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT initial (#PCDATA)>
<!ELEMENT final (#PCDATA)>

<!ELEMENT Alphabet (condition+)>
<!ELEMENT condition (alphabet_id, message, sınır?)>
<!ELEMENT alphabet_id (#PCDATA)>
<!ELEMENT message (#PCDATA)>
<!ELEMENT sınır (#PCDATA)>

<!ELEMENT Transitions (transition+)>
<!ELEMENT transition (from, to, condition_id)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT condition_id (#PCDATA)>
```

4.Projenin Biçimsel Diller ve Otomata ile İlişkisi

Projemizde geliştirilecek olan SIEM uygulaması, biçimsel diller ve otomata teorisi konseptleri ile doğrudan bir ilişkiye sahiptir. Özellikle, log verilerinin analizi ve işlenmesi sürecinde, bu disiplinlerden gelen metodolojilerin ve algoritmaların uygulanması söz konusudur. Log verileri, farklı formatlar ve yapılar içerebildiğinden, bu verileri standart bir formata dönüştürmek ve anlamlı bilgilere çevirmek için düzenli ifadeler (regular expressions) ve ayrıştırma (parsing) teknikleri kullanılır. Düzenli ifadeler, biçimsel dillerin bir alt kümesi olarak kabul edilir ve metin tabanlı verileri işlemede son derece etkilidir. Ayrıca, otomata teorisi, sistemimizdeki olay tanıma ve tehdit tespiti mekanizmalarında da rol oynar. Örneğin, bir güvenlik ihlalinin belirli bir desene göre tekrar eden olaylar zinciri olarak modellenebileceğini düşünelim. Bu tür desenler, sonlu durum makineleri (FSM - Finite State Machines) kullanılarak tanımlanabilir ve tespit edilebilir. FSM'ler, otomata teorisinin temel yapı taşlarından biri olup, sistemimizin olayları sınıflandırma ve tehdit tespiti yapısında kritik bir rol oynar.

Bu bağlamda, projemizdeki SIEM uygulamasının geliştirilmesi, "Biçimsel Diller ve Otomata" dersinde öğrenilen teorik bilgilerin ve konseptlerin pratikte nasıl uygulanabileceğinin bir örneğini oluşturur. Bu sayede, teorik bilgilerin gerçek dünya problemlerini çözmede nasıl kullanılabileceği somut bir şekilde gösterilmiş olur.[7]

5.Projede Kullanılacak Araç: C#

5.1 Arayüz Tasarımı

Projemizin kullanıcı arayüzü, kullanıcı dostu ve etkileşimli bir deneyim sunmak üzere C# programlama dili kullanılarak yapılacaktır. C#'ın zengin widget'lar ve araçlar sunması, modern ve fonksiyonel kullanıcı arayüzleri oluşturma konusunda büyük esneklik sağlar. Örneğin, Windows Forms veya WPF kullanarak, görsel arayüz tasarımını kolayca gerçekleştirebiliriz. Bu kütüphaneler sayesinde, kullanıcılarımızın log verilerini kolayca gözden geçirebilecekleri, filtreleyebilecekleri ve analiz sonuçlarını etkili bir şekilde görselleştirebilecekleri bir arayüz geliştireceğiz.

5.2 Veri Loglarının İncelenmesi

Log verilerinin işlenmesi ve analizi konusunda C#, LINQ (Language Integrated Query) gibi güçlü özelliklere sahiptir. Ayrıca, .NET Framework ile birlikte gelen System.IO ve
sağlar. LINQ, veri temizleme, dönüştürme ve analizi için yüksek performanslı, kullanımı kolay yöntemler sunar. System.IO, dosya okuma ve yazma işlemleri için kullanılırken, System.Text.RegularExpressions, düzenli ifadeler kullanarak metin tabanlı log verilerinin analizini kolaylaştırır.

Ayrıca, log verilerinin anlamlandırılması ve olay tanıma için C#'ın doğal dil işleme (NLP) yeteneklerinden yararlanabiliriz. Örneğin, metin sınıflandırma teknikleri kullanarak, log verileri içerisinden önemli bilgilerin çıkarılması ve tehditlerin tespiti mümkün olacaktır.

6. UYGULAMA YAZILIMI

6.1 Log Sınıfı:

```
9 references
public class Log
{
    0 references
    public string LogID { get; set; }
    5 references
    public string Event { get; set; }
    2 references
    public string DateTime { get; set; }
    2 references
    public string Process { get; set; }
    2 references
    public string UsernameDomain { get; set; }
    6 references
    public string Files { get; set; }
    2 references
    public string DeviceDrive { get; set; }
    2 references
    public string DeviceName { get; set; }
    2 references
    public string DeviceDescription { get; set; }
    2 references
    public string DeviceType { get; set; }
    4 references
    public string DeviceID { get; set; }
    2 references
    public string DeviceGUID { get; set; }
}
```

6.2 LogManager Sınıfı:

```
3 references
public class LogManager
{
    /* EKLEME */
    public List<Log> newLogs = new List<Log>();
    public int lastProcessedLogIndex;
    //***************
    7 references
    public List<Log> Logs { get; private set; }
    public string logFolderPath = @"C:\Program Files\NoVirusThanks\USB Radar\Logs";
    public string logFileExtension = ".log";
    public string logFileNamePattern = "d.MM.yyyy";
    public string[] logHeaders = ...;

    1 reference
    public LogManager()...
    1 reference
    public void ParseLogFile(string filePath)...
    1 reference
    public void WriteLogsToDataGridView(DataGridView dataGridView)...
    1 reference
    public void StartLogMonitoring(string logFilePath, DataGridView dataGridView, int interval)...
}
```

LogManager sınıfı, USB Radar log dosyalarının okunması, analiz edilmesi ve görselleştirilmesi için kullanılır.

Alanlar ve Özellikler:

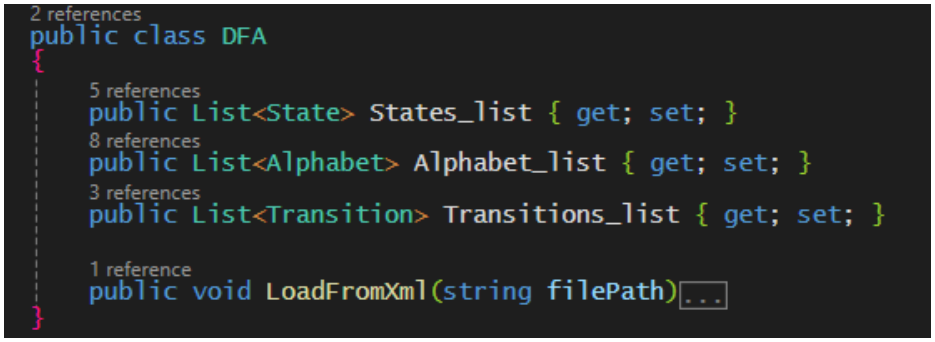
- **newLogs**: Yeni eklenen logları tutar.
- **lastProcessedLogIndex**: Son işlenen log indeksini tutar.
- **Logs**: Tüm log nesnelerini içeren liste.
- **logFolderPath**: Log dosyalarının bulunduğu klasör yolu.
- **logFileExtension**: Log dosyalarının uzantısı.
- **logFileNamePattern**: Log dosyalarının isim deseni.
- **logHeaders**: Log dosyalarının başlıkları.

Metodlar:

1. **LogManager()**:
   - Logs listesini başlatır.
2. **ParseLogFile(string filePath)**:
   - Dosya yolundaki log dosyasını okur ve analiz eder.
   - Her bir log nesnesini Logs listesine ekler.
   - Yeni logları `newLogs` listesine ekler ve `lastProcessedLogIndex`i günceller.
3. **WriteLogsToDataGridView(DataGridView dataGridView)**:
   - Log verilerini DataGridView'e yazar.
4. **StartLogMonitoring(string logFilePath, DataGridView dataGridView, int interval)**:
   - Belirtilen log dosyasını belirli aralıklarla izler.
   - Yeni logları tespit eder ve DataGridView'e yazar.
   - Bu işlem arka planda ayrı bir iş parçacığında yapılır.

Bu sınıf, log dosyalarını analiz ederek, sonuçları kullanıcı arayüzünde görselleştirir ve sürekli izleme sağlar.

6.3 Dfa Sınıfı:



DFA sınıfı, XML formatında tanımlanmış bir DFA'nın yüklenmesi ve yönetilmesi için kullanılır.

Alanlar ve Özellikler:

- **States_list**: DFA'daki durumları tutan liste.
- **Alphabet_list**: DFA'nın alfabesini (geçiş koşulları) tutan liste.
- **Transitions_list**: DFA'daki geçişleri tutan liste.

Metodlar:

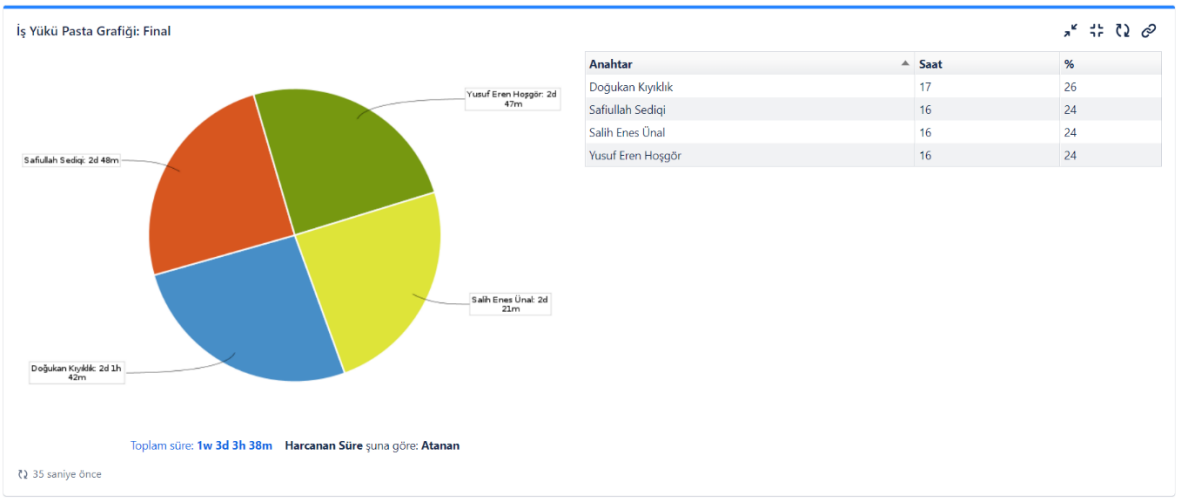
1. **LoadFromXml(string filePath)**:
    - Belirtilen XML dosyasından DFA'nın durumlarını, alfabesini ve geçişlerini yükler.
    - **Durumlar**:
        - XML'deki `<States>` elemanlarını okur ve her bir durumu `States_list` listesine ekler.
    - **Alfabe**:
        - XML'deki `<Alphabet>` elemanlarını okur ve her bir alfabeyi `Alphabet_list` listesine ekler.
    - **Geçişler**:
        - XML'deki `<Transitions>` elemanlarını okur ve her bir geçişi `Transitions_list` listesine ekler.

Bu sınıf, DFA'yı XML dosyasından yükleyerek durumları, alfabeyi ve geçişleri içeren listelere ayrıştırır ve bu verilerin yönetilmesini sağlar.

7.PROJE EKİBİ DEĞERLENDİRME

Grup Koordinatörü: Doğukan Kıyıklık – 152120211104
Toplantı Sayısı: 4 - **Alınan Kararların Uygulanma Oranı:** %90
Toplantıda Alınan Kararlar

- Log dosyalarının nereden elde edileceğine karar verilmesi
- DFA tasarımı
- Kontrol edilecek durumların kararlaştırılması
- Uygulamanın performansının ve verimliliğinin artırılması
- Programlama dili olarak C# kullanılması kararlaştırıldı



Anahtar	Saat	%
Doğukan Kıyıklık	17	26
Safiullah Sediqi	16	24
Salih Enes Ünal	16	24
Yusuf Eren Hoşgör	16	24

8.KAYNAKÇA

[1] https://turk.net/blog/siem-nedir/

[2] https://www.microsoft.com/tr-tr/security/business/security-101/what-is-siem

[3] https://aws.amazon.com/tr/what-is/xml/

[4] https://aws.amazon.com/tr/what-is/xml/

[5] https://www.linkedin.com/pulse/siem-neden-gerekli-ufuk-dilek/

[6] https://www.ozztech.net/genel/siem-security-information-and-event-management-nedir/

[7] https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/automata-theory/basics.html