JavaScript Kütüphanelerinin Güvenilir Olmayan Verilere Karşı Hazır Olma Durumlarının Bayesian Ağları ile Ölçülmesi

E. Ufuktepe, T. Tuğlular

***Özet*—Yazılımın güvenilir olmayan verilere karşı hazır olma durumunun ölçülmesi zordur ve bu alanda çalışmalar halen devam etmektedir. Bunun nedeni yazılım geliştirme ve tasarlama aşamasında geliştiricilerin girdileri doğrulamada başarısız olmasıdır. Bu bildiride, bir yazılımın güvenilir olmayan verilere karşı hazır olma durumunun ölçülmesinde Bayesian Ağlarını kullanan bir yöntem sunulmaktadır. Önerilen yöntemde, seçilen yazılıma ait fonksiyonlar veya metotlar üzerinde çalışılmaktadır. Seçilen fonksiyonların girdi doğrulama bilgileri kullanılmaktadır. Böylece, Bayesian Ağları ile yazılımın güvenilir olmayan verilere karşı hazır olma durumunun ölçümü sağlanmaktadır. Örnek çalışmada, değerlendirme için artan popülaritesinden dolayı JavaScript dilinde geliştirilmiş ve tanınmış olan bir yazılım kütüphanesi olan Jquery seçilmiştir. Önerilen yöntem ile yazılım geliştirme takımları, geliştirdikleri yazılımın güvenilir olmayan verilere karşı önlem almayı izleyebilir.**

***Index Terms*—Yazılım Güvenliği, Girdi Zafiyetleri, Bayesian Ağları, Hazırlığının Ölçümü**

I. GİRİŞ

GİRDİ zafiyetleri, Open Web Application Security Project (OWASP)[1] ve National Vulnerability Database (NVD) [2] tarafından en tehlikeli ve sık karşılaşılan zafiyetlerden biri olarak tanımlanmıştır. Geliştirici tasarım, kodlama ve girdi zafiyetlerini doğrulamasında test yetersiz kaldığında, yazılım geliştirme yaşam döngüsü içerisinde tasarım ve geliştirme aşamalarında gerçekleşmektedir. Bu bildiride, ele alınan bir yazılımın güvenilir olmayan verilere karşı hazır olma durumunun ölçülmesine yönelik bir yöntem sunulmuştur. Yöntem, ele alınan yazılımdaki hazır olma durumunun ölçülmesinde girdi olarak kabul ettiği her fonksiyonun parametrelerinin sebep ve sonuç ilişkilerini dikkate alarak Bayesian Ağlarını (BA) kullanmaktadır. Wagner'ın[3] yazılım kalite tahmini ve değerlendirmesinde kullanılan etkinliğe dayalı kalite modeli takip edilerek BA oluşturulmuştur.

OWASP 2013 yılında ilk on zafiyeti tanımlamıştır [1]. Birçok girdi zafiyeti ilk on içerisinde yer almıştır. Bu çalışmada girdi zafiyetleri hedef alınmış ve XSS, SQL Injection, OS Command Injection, Input Validation, Code Injection ve Path Traversal zafiyetleri seçilmiştir. Christey [4] yazılım zafiyetlerini ağaç yapısı ile sınıflandırmıştır. Bu çalışmada benzer bir yöntem kullanılarak, girdi doğrulama zafiyetlerinin karar ağacını oluşturulmuştur.

Tablo I'de görüldüğü gibi Ocak 2000'de Ocak 2014'e kadar NVD [2] üzerinden rapor edilmiş girdi doğrulama zafiyetlerin verileri toplanmıştır. Böylece bu değerleri ileride oluşturulacak Bayesian Ağında ağırlık olarak verilecektir.

Bildiride 2. Bölüm'de benzer çalışmaların özetleri verilmiştir. 3. Bölüm'de yazılımın güvenilir olmayan verilere karşı hazır olma durumunun ölçülmesi için önerilen Baysian Ağının oluşturulma yöntemi açıklanmıştır. 4. Bölüm'de ise örnek olayın incelenmesi sunulmuş ve sonuçlar 5. Bölüm'de açıklanmıştır.

Tablo I. Ocak 2000'den Ocak 2014'e kadar rapor edilmiş girdi zafiyetlerinin dağılımı (Kaynak: NVD[2])

Girdi Doğrulama Zafiyetleri	Toplam Rapor Edilmiş Zafiyetler	Yüzde
XSS	4584	32%
SQL Injection	3652	25%
Input Validation	2665	19%
Code Injection	1818	13%
Path Traversal	1511	10%
OS Command Injection	108	1%
TOTAL	14338	100%

II. İLGİLİ ÇALIŞMALAR

Frigault ve Wang [5] bir saldırı çizgesinde kodlanmış zafiyetler arasındaki nedensel ilişkileri araştırmıştır. Ancak, zafiyet ve ağların gelişen doğası gereği büyük ölçüde göz ardı edilmiştir. İstismar edilmiş kodları veya yamaların geçerliliği gibi zamansal faktörleri birleştirmek için bir Dinamik Bayesian Ağ tabanlı model önermişlerdir. Modelden başlayarak, potansiyel uygulamaları göstermek için iki somut olgu üzerinde çalışmışlardır. Bu yeni model sürekli dinamik bir ortamda ağ güvenliğini ölçmek için teorik bir temel ve pratik bir yapı oluşturmuşlardır.

Kondakçı [6] Bayesian Ağlarını kullanarak bir ağ güvenliği risk değerlendirme modeli önermiştir. Çeşitli Bilgi Teknolojileri (BT) varlıklarını ve güvenilir hesaplamalı ortamlarda risk hesaplamalarının uygulanılabileceği genel bir tehdit modeli ortaya atmıştır. Bilgi güvenliği tehditlerini (insan-ilişkili, iç ve dış) sınıflandırarak dört güvenilir parametre ile birleşik bir yapı oluşturup modellemiştir. Ayrıca

hangi risk düzeylerini kolayca tahmin ve farklı değerlendirme sistemler için puan ortalaması düzeni ve koşullu olasılık metotlarını kullanarak yeni bir risk yayılım modeli geliştirmiştir. Öte yandan Bayesian Ağların bilgiyi ifade ediş şekli ile otomatik olarak akıl yürütme yapısından dolayı çalışmasında kullandığını, BT ortamlarının dinamik ve değişen yapısı gereği geleneksel yorumlama metotların sonraki risk faktörleri dağılımında uygulanmasının daha güç olduğunu belirtmiştir. Bu nedenle birbirine bağımlı analizlerde Bayesian yaklaşımının geniş ölçekli ağlarda uygulanabilirliğini kolay olduğunu ifade etmiştir.

Wagner [3] faaliyet temelli kalite modelleri üzerinden yazılım değerlendirmesi ve kalite tahmininde Bayesian Ağlarını kullanmıştır. Bayesian Ağını oluştururken üç tip düğüm tanımlamıştır ve dört temel adımda belirtmiştir. İlk adımda ilgili faaliyetlerin ve göstergelerin hedef tabanlı türetilmesidir. İkinci adımda faktörlerimn ve alt faaliyetlerin belirlenmesidir. Üçüncü adımdafaktörler için uygun göstergeler içerilirmiştir. Dördüncü adımdadüğüm olasılık tabloları tanımlanıp kantitatif ilişkiler gösterilmiştir. Düğümler yazılımın "Durumlar ve Faaliyetler" haritasına göre oluşturulmuştur. "Faaliyetler" sistem üzerinde etkisinin bilgilerinin tutar. Örneğin; bakım ve kullanım yüksek dereceli birer faaliyettir. "Durumlar" ise bir sistemin ortamı ve geliştirme organizasyonunu içermektedir. Faaliyet üzerinde pozitif ya da negatif etkisi olan her durum için faktörlere bir düğüm eklenir. Eğer durumun faaliyet üzerinde hiçbir etkisi yoksa düğüm eklenmez. Ancak unutmamalıyız ki faaliyet üzerinde etkisi olan her alt faaliyet ve faktör için faaliyet düğümü eklenmelidir.

Guarnieri ve Livshits [7] statik analizin program optimizasyonundan hata tespit etmeye kadar kadar kullanışlı bir teknik olduğunu dile getirmiştir. Bu çalışmaları JavaScript programlarındaki akışı aşamalı statik analizi incelemişlerdir. Çalışmalarında hızdan dolayı çevirim içi ve çevirim dışı statik analizlerini birleştirerek kullanımını savunmuşlardır. Çevirim dışı kullanımı sunucunun ötesinde bir zamanda kullanılabilir, oysaki çevirim içi analizi web tarayıcısına entegre edilebilir olduğunu düşünmüşlerdir. Gerçek hayattan ve sentetik JavaScript kodu üzerinden gerçekleştirilen geniş kapsamlı deneylerden sonra, koddaki güncellemelerin az olduğu yerlerde kullanımı daha normal görmüşlerdir. Güncel statik analiz sonuçlarının her gün kullanımı için tarayıcıda kullanımını hızını kabul edilebilir olarak görmüşlerdir. Bu formdaki aşamalı statik analizi yaklaşımının birçok alanda, özellikle mobil cihazlarda avantajlı olduğunu iddia etmişlerdir.

Jensen ve arkadaşları [8] TAJS adında geliştirdikleri statik analiz aracı ile JavaScript dilinde veri tipi analizini gerçekleştirmiştir. Tip analizini "Değer" örgüsü üzerinden ve transfer fonksiyonların veri akışı analizi ve akış çizgeleri üzerinden gerçekleştirmişlerdir. Örneğin; eğer bir fonksiyon "*toString()*" fonksiyonu ile çağırılıyorsa, veri tipini "String" olarak yorumlanmıştır. Bu çalışmalarında, tip analizcilerinin JavaScript için ilk ses bazlı ve detaylandırılmış araç olduğunu ifade etmişlerdir. Monoton yapının ayrıntılı örgü yapısı ile kullanımı yeni soyutlama ile birleştirildiğinde istenilen ölçütte analiz sonuçların hassasiyetini iyileştiğini gözlemlemişlerdir.

III. GÜVENİLİR OLMAYAN VERİLERE KARŞI HAZIRLIĞININ ÖLÇÜLMESİ İÇİN BAYESİAN AĞININ OLUŞTURULMASI

Çalışmamızın temelinden Bayesian Ağlarının kullanılma nedeni, BA'ların belirsizliklerin açıklanmasında tutarlı anlamlar sunması ve çeşitli etkileşimlerin sebep ve sonuç ilişkilerine dayalı sezgisel çizgelerle ifade edilebilmesidir [9].

İlk adımda faaliyetler tanımlanmıştır. Faaliyetler tanımlamış olduğumuz altı girdi doğrulama zafiyetine karşılık gelmektedir; XSS, SQL Injection, OS Command Injection, Input Validation, Code Injection ve Path Traversal. Daha sonra tanımlanmış faaliyetlere ilişkin faktörler tanımlanmıştır. Bu durumda Bayesian Ağında bu çalışma kapsamında tanımlanan fakörler, ele alınan yazılımda tespit etmiş olduğumuz zedelenilebilir fonksiyonlara karşılık gelmektedir. Son adımda indikatörler tanımlanmıştır. Özel olarak ölçmek istenilen faaliyet veya faktörden ek bir düğüm çıkartılılarak indikatörler tanımlanır. Wagner [3] indikatörlerin Bayesian Ağlarında yerini şöyle tanımlar, "faaliyet veya faktörlerden indikatöre bir ayrıt ile tanımlanır. İndikatörler faaliyetler ve faktörlere bağlıdır. İndikatörler sadece altta yatan anlatımların ifade ediliş şeklidir".

Bayesian Ağını oluşturuken, düğümler arasındaki ilişkiler gösterilmiştir (bknz Şekil 3. Her öncel düğüm takipçi düğümün Olasılıksal Düğüm Tablosunu (ODT) etkiler. BA'da toplamda oluşacak ODT sayısı BA'da bulunan düğüm sayısı kadar olacaktır. Toplamda bulunacak ODT sayısı formülü:

$$1+(zafiyet\_sayısı)+(fonksiyon\_sayısı)=(toplam\_ODT'ler) \quad (1)$$

Formül 1'deki (1) 1 değeri "Uygulama düğümüne" karşılık gelmektedir. Çalışmada 6 girdi doğrulama zafiyeti tanımlandığından zafiyet sayısı 6 olacaktır. Fonkisyon sayısı ise yazılımdan seçilen fonksiyonların sayısına denk gelmektedir. Yazılımda bulanan fonksiyonlar fazla olacağından, seçilen fonksiyonlar sık kullanılan ve yazılımda yayılmış olan fonksiyonlar tercih edilmiştir. ODT'leri üç kategoride incelenmiştir; Uygulama ODT, (girdi doğrulama) Zafiyet ODT ve Fonksiyon ODT'leri. Şekil 3'de görüldüğü gibi, en üstteki tek düğüm uygulama düğümü olup Uygulama ODT'si içermektedir, altı zafiyet düğümlerinin bulunduğu ikinci katmanda Zafiyet ODT'lerini ve fonksiyon düğümlerinin bulunduğu üçüncü katmanda Fonksiyon ODT'lerini içermektedir.

A. Uygulama Düğümünün ODT Hesaplaması

Her zafiyetin uygulama üzerinde farklı etkisi olmaktadır. OWASP [1] üzerinde zafiyetlerin risk sıralaması Bölüm 1'de gösterilmişti. Ancak risk bilgileri sadece sıralama olarak verilmiştir (bknz. Tablo II). Zafiyetlerin risk yüzdelerini elde etmek için sıralamalarına ilişkili olarak ağrılık atanarak "Ağırlıklı Yüzde Formül'ünü/Weighted Percentile Formula"(2) uygulanarak zafiyetlerin risk yüzdeleri elde edilmiştir.

$$S_n = \sum_{k=1}^{n} w_n, P_n = \frac{100}{S_n}\left(S_n - \frac{w_n}{2}\right) \quad (2)$$

Girdi zafiyetleri üzerinde "Ağırlıklı Yüzde Formül'ü" uyguladıktan sonra Tablo II'deki değerler elde edilmiştir.

Tablo II. Risk yüzdeleri ile beraber girdi doğrulama zafiyetleri

Risk Sıralaması	Girdi Doğrulama Zafiyeti	Ağırlık	Risk %
1	SQL Injection	4	80%
1	Code Injection	4	80%
1	OS Command Injection	4	80%
2	XSS	3	75%
3	Path Traversal	2	66.7%
4	Input Validation	1	50%

Zafiyetlerin riskleri tanımlandıktan sonraki aşamada Uygulama düğümünün ODT'si hesaplanır. Uygulama düğümünün içerdiği değer sayısını 2^{n+1} olarak tanımlanmıştır, *n* değerini Uygulama düğümüne bağlı olan öncel düğümlerin (zafiyet kategorilerin) sayısı olarak belirlenmiştir. 1 değeri ise yanlış değerlerin hesaplanmaya katılması için eklenmiştir.

(hesaplanmış doğru değer)+(hesaplanmış yanlış değer)=1 (3)

Aksi takdirde Bayesian Ağına sadece doğru değerleri tanımlanacaktır ve eksi yöndeki değişmleri görülemeyecektir. Bu durumda, Uygulama düğümünün ODT'si 2^{6+1}= 128 içerecektir. Uygulama düğümünün ODT değerleri hesaplandıktan sonra Bayesian Ağlarına uygun formatta değerler sunabilmek için değerler üzerinde 1-0 normalizasyonunu (4) uygulanmıştır.

$$Normalized(e_i) = \frac{e_i - E_{min}}{E_{max} - E_{min}} \quad (4)$$

B. Zafiyet Düğümlerinin ODT'lerinin Hesaplanması

Bayesian Ağındaki bütün fonksiyon düğümleri her zafiyet düğümüne yönelik ayrıt ile bağlanmıştır. Böylece zafiyetlerin ODT değerlerini fonksiyon düğümleri belirleyecek anlamına gelmektedir. Zafiyetlerin ODT değerlerini 3 tip fonksiyon bilgisi ile değerlendirilmiştir.

İlk olarak Tablo I'de görüldüğü gibi NVD [2] üzerinden toplanan Ocak 2000'den Ocak 2014'e kadar zafiyetlerin sıklıkları hesaplanmış, fonksiyonları hangi olasılıkla karşılacağı bilgisini çıkarılmıştır. Bu bilgiyi fonksiyon bilgisi olarak kullanılmanın nedeni, rapor edilen zafiyetlerin kod üzerinden yapılmış olmasından kaynaklanmaktadır. Ek olarak, Bayesian Ağlarının doğası gereği öncel (fonksiyon) düğümlerin bilgilerini takipçi (zafiyet) düğümlerine kalıtsal olarak aktarır. Böylece takipçi düğümler zafiyet sıklığı bilgisinden mahrum kalmayacaktır.

İkinci bilgi olarak, fonksiyonları derinlik bilgisi kullanılmıştır. Derinlik terimini detaylandırıldığında, bir fonksiyonun çağrı çizgesi çıkarıldığında, bir fonksiyonun çağrıma ilişkilerini bir yönlendirilmiş çizge ile elde edilir. Çağrı çizgesindeki döngüleri görmezden gelerek, fonksiyonun çağırılma anından yaprak düğümüne kadar en uzun mesafesi alınarak derinliği hesaplanır. Bir fonksiyonun çağrı çizgesinde bulunan diğer fonksiyonları da göz önünde bulundurarak, bir fonksiyonun derinliği arttıkça doğru orantılı olarak potansiyel zararının da artacağı varsayılmaktadır.

Üçüncü ve son bilgi olarak, belli bir fonksiyonun kod içerisinde kaç kez çağırıldığının bilgisidir. Eğer bir fonksiyon kod içerisinde birçok kez çağırlıyorsa, çağırılan fonksiyonun kod içerisindeki kapsanması da artacaktır.

C. Fonksiyon Düğümlerinin ODT'lerinin Hesaplanması

Bayesian yaklaşımı kanı temellidir ve belirsiz sonuçlar sunmaktadır. Belirsizlik, bir durumdan %100 emin olmama anlamına gelmektedir. Bu yüzden BA'larda olasılık değerlerinde küçük bir standart sapma verilmektedir. Aksi takdirde 1 ve 0 gibi kesin değerlerin verilmesi ile Bayesian yaklaşımdan ziyade bir Sıklıkçı/Frequentist yaklaşım olmaktadır. Bir Sıklıkçı yaklaşım, bir olayların frekanslarına bağlı olarak rastgele durumlara değer atamaktadır. Bir Bayesian yaklaşım ise belirsiz durumlara göre olasılık değerleri atamaktadır.

Bu sebeple fonksiyon düğümlerin ODT'da *doğru* ve *yanlış* durumları vardır. *Doğru* durumuna "0.95" değeri atanarak, fonksiyonun herhangi bir doğrulama kodu kullanmadığını, *yanlış* durumu için "0.05" değeri atanarak, fonksiyonun doğrulama kodu kullandığını belirtilmiştir. Bayesian yaklaşımına sadık kalmak için 1 ve 0 gibi kesin değerler verilmemiştir. Ayrıca kesin değerler verilmeyerek bazı özel durumlar da katmıştır. Örneğin; dil kodlanması atakları ve tip bazlı doğrulamaların atlatılması vb.

D. Otomatik Bayesian Ağ Oluşturucusunun Geliştirilmesi

Bayesian Ağın oluşturulmasında gerekli bilgilerin tutulması için bir araç önemli bir faktör. BA'nın düğüm bağlantılarını ve her düğümün ODT değerlerini tutacak bir dosya olmasıdır. Bu nedenle Java ile yazılmış açık kaynak bir yazılım aracı olan OpenMarkov [11] kullanılmıştır. Okunması ve yazması kolay olan XML fomatına yakın bir dosya sağlamaktadır. Böylece OpenMarkov'un dosya formatı ile Bayesian Ağı oluşturulabilmektedir. OpenMarkov Java dilinde yazıldığından Bayesian Ağ Oluşturucusu Java dilinde geliştirilmiştir. BA'da ODT'lerin hesaplanmasında bağlantıların sağlanması ve dosya operasyon yapılması kolaylaştırıldı.

Otomatik Bayesian Ağının dinamik, otomatik ve esnek bir BA sağlaması hedeflenmektedir. Böylece geliştiriciler BA'larını JavaScript uygulamalarına göre dilediği gibi modifiye edebilecektir. Bunu sağlamak için, öncelikle geliştirilen aracın okuyacağı bir kurulum dosyası hazırlanmıştır. Geliştiricilere zafiyet ve fonksiyon eklemeleri ve çıkarmaları için imkan sağlamaktadır. Bir başka avantaj ise, kurulum dosyasının fonksiyon ve zafiyet bilgilerini içermesidir. Bu da geliştirilen aracın güncel kalmasını sağlamaktadır. Fonksiyon ya da zafiyet istatistiklerinde herhangi bir değişiklik ile karşılaşıldığında, kurulum dosyası değiştirilip şimdiki zaman değerlerine adapte edilebilmektedir.

İlerleyen aşamalarda algoritmalar (her zafiyet ve fonksiyonun ODT'lerin değerlendirilmesi) araca entegre edilmektedir. Daha sonra düğümler tanımlanarak BA oluşturulmaktadır. Bir sonraki aşamada düğümlerin ilişkileri ve bağlantıları sağlanmaktadır ve BA'nın mimarisinin nasıl olacağı tanımlanmaktadır. BA üç katmanla tanımlanmaktadır; Uygulama, Zafiyet ve Fonksiyon. Özetle, düğümlerin ilişkilerini araca tanımlarken, bütün zafiyet düğümlerini Uygulama düğümüne yöneltilmektedir ve bütün fonksiyon düğümleri her bir zafiyete yönlendirerek tam ikili parçalı bir çizge elde edilmektedir. Son olarak, hesaplanmış ODT'leri düğümler atanmaktadır.

IV. ÖRNEK ÇALIŞMA

Örnek çalışma olarak jQuery'nin sürüm 1.9.1 kütüphanesi seçilmiştir. jQuery sık kullanılan ve web uygulama geliştiricilerin tercih ettiği bir kütüphanedir. Geliştiricilerin jQuery'i tercih etmelerindeki en büyük etken hızlı, küçük ve zengin içerikli olmasıdır. Geliştiricilere yardımcı olabilecek birçok fonksiyon içermektedir, örneğin; olay işleme, HTML doküman gezinmeleri, manipülasyon ve animasyon vb. Bunula beraber birçok popüler web sayfalarında kullanılmaktadır; Amazon, Microsoft, WordPress, Reddit, Instagram, Stack Overflow, the Guardian, Fox New vb. Öncelik olarak fonksiyonların bilgilerinin toplanması üzerine yoğunlaşılmaktadır.

A. *Seçilen jQuery Fonksiyonları*

İlk adımda 10 tane en sık kullanılan ve geliştiriciler tarafından en çok tercih edilen fonksiyonlar seçilmektedir. Ek olarak, Cooper [12] Bayesian Ağlarının sonuç çıkarımlarının NP-hard problem olarak tanımladığı için fonksiyon seçimi 10 ile kısıtlanmıştır. Seçilen fonksiyonlar üzerinden fonksiyonların parametrelerinin kullanılmadan önce doğrulanıp doğrulanmadığını kontrol edilir. Parametrelerin doğrulanıp doğrulanmadığını öğrenmek için TAJS'nin [8] akış çizgesi özelliği üzerinden herhangi bir doğrulama bloğun olup olmadığı elle de kontrol edilmiştir.

Şekil 1'de, TAJS ile "attr()" fonksiyonunun akış çizgesi çıkartılmıştır. Herhangi bir girdi doğrulama işleminin yapılıp yapılmadığını kontrol edilmeden önce her parametreyi birer girdi olarak kabul edilmiştir. Daha sonra parametrelerin kullanılmadan önce doğrulanıp doğrulanmadığını kontrol edilmiştir. Şekil 1'de kırmızı çerçeve içerisinde fonksiyonun parametreleri görülebilmektedir. Kırmızı çerçeveye bakıldığında parametreler üzerinde "*okuma/read*" operasyonu uygulanmaktadır, bir başka değişle parametrelerin kullanılmaya başladığının bir göstergesidir. Bu yüzden kırmızı çerçevenin üstüne bakıp öncesinde parametreler üstünde herhangi bir doğrulama yapılmış mı kontrol edilmektedir. Akış çizgesine bakıldığında parametreler kullanılmadan önce herhangi bir doğrulama işlemi ile karşılaşılmamaktadır. Şekil 2'ye bakıldığında, aynı fonksiyonun çizgesinin bu sefer bir doğrulama işleminin yapıldığı sürümü incelenmektedir. Bu incelendiğinde parametrelerin kullanıldığı kırmızı ve yeşil çerçeveler görülmektedir. Ancak, yeşil çerçevedeki "*okunma/read*" operasyonu doğrulama için bir "if/sorgu-blok'u" için uygulanmaktadır. Eğer parametre koşulları sağlamıyorsa, akışı bir başka duruma yönlendirerek yazılımın zarara uğramısını engellemektedir.

İkinci aşamada, fonksiyonun jQuery içerisinde kaç kez çağırıldığı sayılmaktadır. Sonuçları el ile de kontrol ederek bir fonksiyonu bulmak ve kaç kez çağırıldığını saymak için Eclipse'in Refactor özelliği ile JSRefactor [13] kullanılmaktadır. Üçüncü aşamada, fonksiyonların derinliklerini hesaplamak için çağrı çizgeleri çıkartılarak, çağırıldığı andan en uzun yaprak düğümüne kadar olan mesafe alınmaktadır. Fonksiyonların çağrı çizgelerini elde etmek için TAJS'nin çağrı çizglerini çıkartma özelliğinden faydalanılmaktadır. Seçilen 10 fonksiyon hakkında elde edilen bilgiler Tablo III'de görülmektedir.

Tablo III. Program fonksiyon bilgileri.

Fonksiyonun Adı	Fonksiyonun jQuery içerisinde kaç kez çağırıldığı	Derinlik
html()	2	2
text()	1	2
css()	32	4
attr()	3	2
val()	5	0
each()	59	0
data()	5	2
hasData()	2	0
removeData()	1	2
find()	19	0

B. *jQuery'nin Güvenilir Olmayan Verilere Karşı Hazırlığı*

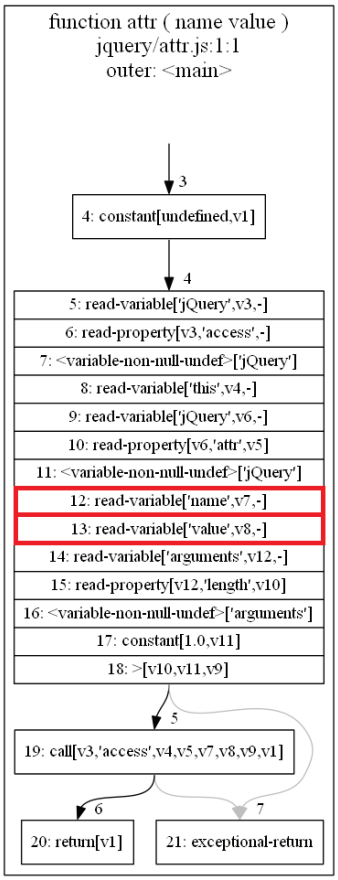
Bayesian Ağının Bayesian Ağ Oluşturucusu ile çalıştırıldığında sadece hangi tip zafiyetler ile karşılaşılabileceğin bilgisi değil aynı zamanda uygulamanın güvenilir olmayan verilere karşı hazırlığını da incelenmektedir. Tablo IV'de BA'nın verdiği değerler görülmektedir.

Tablo IV. Uygulamanın girdi doğrulama zafiyeti içerme olasılıkları

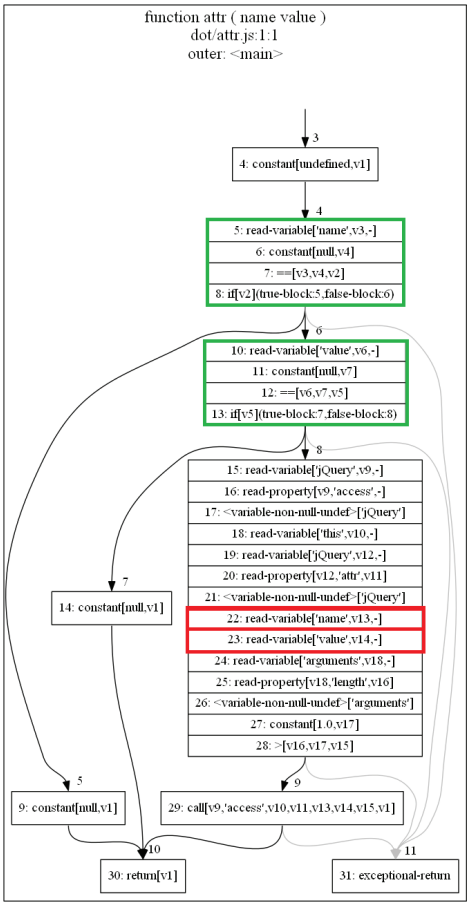
Zafiyet	Zafiyetin uygulama içerisinde karşılaşılma olasılığı
XSS	0.95
SQL Injection	0.74
Input Validation	0.56
Code Injection	0.39
Path Traversal	0.29
OS Command Injection	0.03

Bayesian Ağ üzerinde herhangi bir değişiklik yapılmadığı takdirde jQuery'nin seçilmiş 10 fonksiyonuna göre, güvenilir olmayan verilere karşı hazırlığını 0.51 olarak ölçülmektedir. Bu sonuç üzerinden jQuery'nin güvenilir olmayan verilere karşı %51 hazır olarak yorumlanmaktadır.

Ancak, fonksiyonları "Not Used/Kullanılmamış" durumları seçerek hazırlık değerleri değiştirilebilmektedir. Bunula beraber, eğer bir uygulama herhangi bir veritabanı kullanmamış ise, geliştirici "SQL Injection" zafiyet düğümünün durumunu "Not Contains/İçermiyor'a" çevirip hazırlık değerini değiştirebilmektedir, çünkü veritabanı olmayan bir uygulamanın SQL Injection saldırılarına karşı bir etkisi olmayacaktır. Şekil 3'de Bayesian Ağ Oluşturucusunun



Şekil 1. Doğrulanmamış parametrelerin akış çizgesi örneği.



Şekil 2. Doğrulanmış parametrelerin akış çizgesi örneği.

fonksiyonların "Not Used/Kullanılmamış" ve zafiyetlerin "Not Contains/İçermiyor" ekran alıntısı görülmektedir. Yapılan seçimlerle ve ayarlarla hazırlık değerinin 0.41 olduğu görülmektedir. Gri düğümler geliştiricinin seçilmiş düğümlerini ifade etmektedir. Fonksiyon düğümlerinde ise seçim için iki durum tanımlanmıştır; doğrulanmış parametreler/girdiler ve doğrulanmamış parametreler/girdiler. Geliştirici "Not Used/Kullanılmamış" durumunu işaretlediğinde, otomatik olarak zafiyet düğümlerini içerme yüzdeleri yükselecek ve uygulama düğümünün hazırlık yüzdesi azalacaktır. Çünkü "Not Used/Kullanılmamış" durumu seçilerek, geliştirici fonksiyonun parametreleri için gerekli doğrulama işlemlerini yapmadığını onaylamaktadır. Ancak geliştirici fonksiyon parametreleri için gerekli doğrulama operasyonlarını tamamlamışsa "Used/Kullanılmış" durumunu seçerek zafiyet düğümlerinin yüzdelerini düşürebilir ve uygulama düğümünün hazırlık yüzdesini yükseltebilmektedir.

V. SONUÇ

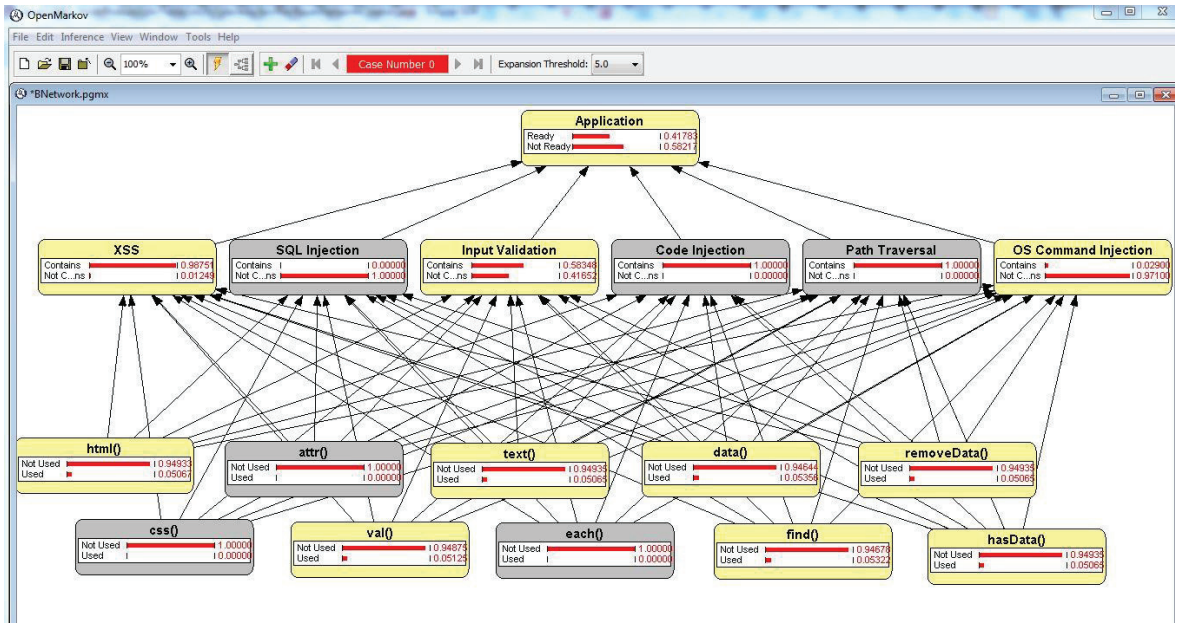
Bu bildiride Bayesian Ağları ile JavaScript uygulamalarında güvenilir olmayan verilere karşı hazırlığının ölçülmesi için bir yöntem ve bunu yapabilecek otomatize edilmiş Bayesian Ağ Oluşturucusu geliştirilmiştir. Güvenilir olmayan verileri tanımlandığında, toplamış olduğumuz verilerden (14 yıllık rapor edilmiş zafiyetlerin istatistikleri) girdi doğrulama zafiyetlerin davranışları gözlemlenmiştir. Her zafiyetin yazılım geliştirme yaşam döngüsü içerisindeki zarar azaltma istatistikleri incelenmiştir. İncelemeler sonucunda 6 tane girdi zafiyeti bir karar ağacı ile sınıflandırılmıştır. Yazılım geliştirme yaşam döngüsünün tasarım ve geliştirme aşamalarında, girdi doğrulama zafiyetleri için önerilen zarar hafifletme işlemi girdilerin doğrulanmasıdır. Bu nedenle statik analizi sadece potansiyel girdi doğrulama zafiyetlerini tespit etmek için değil, aynı zamanda uygulamanın fonksiyonları analiz ederek derinliklerinin çıkarılması için de kullanılmıştır. Ek olarak, fonksiyonların kaç kez çağırıldıklarını bulabilmek için refactoring kullanılmıştır.

Bayesian Ağını oluşturulurken üç aşama ile tanımlanmıştır; faktörler, aktiviteler ve uygulamanın hazırlığı. Faktörler, girdi zafiyetlerin sıklığını, fonksiyonun derinliği ve fonksiyonun kaç kez çağırıldığının bilgilerinin tutan fonksiyon düğümlerine karşılık gelmektedir. Faaliyet aşaması, zafiyetlerin zarar verme yüzdelerini tutan girdi zafiyetleri düğümlerine denk gelmektedir. Son aşama ise, uygulamanın güvenilir olmayan verilere karşı hazır olma durumunun ölçümünü veren çıkış düğümüdür.

Örnek çalışmada, testler JavaScript geliştiricilerinin kullanımından ve sık tercih etmelerinden dolayı JavaScript'in en popüler kütüphanesi olan jQuery üzerinde

gerçekleştirilmiştir. Son olarak, jQuery'nin en sık kullanılan 10 fonksiyonu üzerinden deneyler gerçekleştirilerek, Bayesian Ağındaki düğümlerindeki durumlarda hiçbir değişiklik yapmadan, jQuery'nin güvenilir olmayan verilere karşı hazırlığı %51 olarak ölçülmüştür. Sadece güvenilir olmayan verilere karşı hazırlığının ölçülmesi ile kalmayıp, uygulamanın içerisinde hangi zafiyetin ne kadar olasılıkla bulunma durumlarını da gözlemlemiştir.

Netice olarak, Bayesian Ağlarını kullanan birçok çalışmada bilgi eşit olarak dağıtılmıştır. Ancak önerilen model ve geliştirilen araçta gerçek bilgi ve istatistikler kullanıldığından sunulan ölçüm ve çalışma diğer çalışmalardan ayrılmaktadır. Bir başka avantaj ise sunulan aracın esnek ve dinamik olmasıdır. Böylece geliştirici kolaylıkla kendi fonksiyon, zafiyet ve bilgilerini tanıtabilmektedir. Bu bildiride geliştiricilerin yazılımlarının güvenilir olmayan verilere karşı ne kadar hazır olduğu ölçen bir model ve bir araç geliştirilmiştir. Böylece geliştiriciler ölçüm sonuçlarını gözlemleyip uygulamalarını buna göre geliştirebilmektedir.



Şekil 3. Bayesian Ağ Oluştucusu Ekran Alıntısı.

KAYNAKLAR

[1] N. Smithline, OWASP Top 10 2013, https://www.owasp.org/index.php/Top_10_2013-Top_10, 2013.
[2] National Vulnerability Database, http://web.nvd.nist.gov/view/vuln/statistics, 2014.
[3] S. Wagner, A Bayesian Network Approach to Assess and Predict Software Quality using Activity-based Quality Models. Information and Software Technology, 52, pp.1230-1241, 2010.
[4] S. Christey, Preliminary List Of Vulnerability Examples for Researchers, NIST Workshop Defining the State of the Art of Software Security Tools, Gaithersburg, MD, August 2005.
[5] M. Frigault and L. Wang . Measuring network security using bayesian network-based attack graphs (pp. 698-703). IEEE, 2008.
[6] S. Kondakçı. Network Security Risk Assessment Using Bayesian Belief Networks, IEEE International Conference on Privacy, Security, Risk and Trust, 978-0-7695-4211-9/10, 2010.
[7] S. Guarnieri and V.B. Livshits. Gulfstream: Incremental static analysis for streaming JavaScript applications. In Proceedings of the USENIX Conference on Web Application Development, 2010.
[8] S. H. Jensen, A. Møller, and P. Thiemann. Type analysis for JavaScript. In Proc. 16th International Static Analysis Symposium, SAS '09, volume 5673 of LNCS, pages 238–255. Springer-Verlag, August 2009, 2009.
[9] D. Heckerman. A Tutorial on Learning With Bayesian Networks, Technical Report, Microsoft Research, no. MSR-TR-96-06, http://research.microsoft.com/apps/pubs/?id=69588, 1996
[10] K.B. Korb and A.E. Nicholson. Bayesian artificial intelligence. cRc Press, 2003.
[11] OpenMarkov, http://www.openmarkov.org/, 2014.
[12] G.F. Cooper. "The computational complexity of probabilistic inference using Bayesian belief networks." Artificial intelligence 42, no. 2: 393-405, 1990.
[13] JSRefactor, http://www.brics.dk/jsrefactor/, 2014.

Ekincan UFUKTEPE İzmir Ekonomi Üniversitesi Bilgisayar Mühendisliği Bölümü'nden 2011 yılında mezun oldu. İzmir Yüksek Teknoloji Enstitüsü'nde Bilgisayar Mühendisliği Bölümünden 2014 Eylül ayında Yüksek Mühendis olarak mezun oldu. Halen İzmir Yüksek Teknoloji Enstitüsü'nde Araştırma Görevlisi olarak görevine devam etmektedir.
Tuğkan TUĞLULAR Ege Üniversitesi Bilgisayar Mühendisliği Bölümü'nden doktorasını 1999 yılında aldı. 2000 yılında İzmir Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği Bölümü'ne katılan Tuğlular aynı bölümde Öğretim Üyesi olarak görevine devam etmektedir.