HTML5 Güvenliği

Yeni Nesil Web Tehditleri

Emre Çakır

BEAM Teknoloji AŞ, Ankara

e-posta: emre@beamteknoloji.com

Özetçe

HTML5, süregelen tarayıcı savaşlarındaki "Kim daha fazla HTML5 destekliyor?" bölümünün etkisiyle ve özellikle sosyal paylaşım sitelerinin HTML5 standardının getirdiği yeni özellikleri kullanıcılarına sunmak için sabırsızlanmasıyla birlikte İnternet dünyasında har geçen gün daha fazla yer buluyor. Yarış ortamında geliştirilen ve piyasaya sürülen tarayıcılar ile henüz taslak halde bulunan bir standart kılavuzluğunda geliştirilen web uygulamalarının testlerini yapma yükümlülüğü de çoğu zaman durumdan habersiz İnternet kullanıcılarına kalıyor. Günümüzde birçok önemli verinin İnternet üzerinde depolandığı, taşındığı ve paylaşıldığı gerçeğini göz önünde bulundurursak, henüz taslak halde bulunan HTML5 standardının düzgün yorumlanmaması ve/veya uygulanmaması İnternet kullanıcılarının güvenliği konusunda ciddi bir tehdit oluşturuyor. Bu makale, güvenlik hassasiyeti olmadan geliştirilen HTML5 standardında tanımlanan yeni özelliklerin sebep olabileceği zafiyetleri incelemekte ve kullanıcıları bu zafiyetlere maruz bırakmamak için uyulması gereken güvenli web yazılımı geliştirme prensiplerini içermektedir.

1. Giriş

Halen Web uygulamalarını geliştirmek için kullanmakta olduğumuz HTML4 standardının geliştirilmesi 1997'de tamamlanmış olup W3C tarafından 1999 yılında geliştiricilerin kullanımı için önerilmiştir[1]. O günden bu bugüne gerek yazılım dünyasındaki gerekse donanım dünyasındaki gelişmeler göz önünde bulundurulduğunda kullanıcıların İnternetten beklentileri belirgin bir şekilde artmıştır. Yazılım geliştirme ortamlarının yeteneklerinin artması, internet bağlantı hızlarının belirgin bir şekilde yükselmesi, İnternet erişiminin cep telefonlarına kadar ulaşması bu dünyada yer sahibi olmak isteyen üreticilerin ortam gereksinimlerini arttırmış; bankacılık, eğitim, alışveriş, iletişim, paylaşım gibi birçok aktivitenin İnternet üzerine taşınması yeni ihtiyaçlar ortaya çıkarmıştır. İnternet dünyasında bu köklü değişimler yaşanırken, uygulamaların üzerine kurulduğu standart aynı kalmış ve yakın dönemlerde çoğu zaman geliştiricileri kısıtlayan bir kurallar bütünü haline gelmiştir. Yeni gelişen ihtiyaçlar Adobe® Flash, JAVA Applet, Microsoft® Silverlight gibi yan çözümlerle giderilmeye çalışılırken bu birbirinden bağımsız üretici çözümleri İnternet üzerinde standarttan uzaklaşmayı arttırarak, İnterneti birlikte çalışamayan ve de çalışamayacak olan uygulamalar yığını haline getirmeye başlamıştır. Son dönemde hızlıca yükselen sosyal paylaşım siteleri ve özü gereği bu sitelerin paylaşım ve birlikte çalışırlığa yönelik yapıları İnternet için yeni bir standart ihtiyacını tepe noktasına çıkartmıştır.

HTML standardın İnternetin yeni gereksinimlerini karşılaması amacıyla güncellenmesi için WHATWG (Web Hypertext Application Technology Working Group) 2004 yılında çalışmalarına başlamış ve çok geçmeden W3C de bu sürece dâhil olmuştur[2]. İnternetin yeni anayasası statüsündeki HTML5, Mayıs 2011 itibariyle yeni önerilere kapısını kapatmış ve 2014 yılı W3C tarafından standardın önerileceği tarih olarak belirlenmiştir[3].

HTML5 standardının 2014 yılına kadar önerilmeyeceği, 2014 yılına kadar HTML5 zafiyetlerinden kaynaklanacak tehditlere maruz kalmayacağımız manasına gelmemektedir. HTML5 WHATWG ve W3C önderliğinde ve koordinasyonunda geniş bir topluluk tarafından açık bir şekilde geliştirilmekte ve taslak standardın geçirdiği bütün evreler geliştiricilere sunulmaktadır. An itibariyle elimizdeki taslağın bazı kısımları olgunlaşmışken bazı kısımlarında geliştirme henüz devam etmektedir. Bu geliştirmeye aktif bir şekilde katılan tarayıcı üreticileri, bir yandan standardı gerçeklemeye devam ederken bir yandan da karşılaştıkları sorunları WHATWG ve W3C ile paylaşmakta, standart da bu şekilde birçok açıdan evrimleşmektedir. Bu süreç sırasında da daha önce bahsedilen sebeplerden dolayı, tarayıcı üreticileri taslak gerçeklemelerini yeni sürümlerinde piyasaya sürmekte ve HTML5 ile geliştirilmiş web uygulamaları bugün aktif olarak bütün popüler tarayıcılarda kısmi destekle çalışmaktadır.

Bu makale henüz taslak halde bulunmasına rağmen İnternette aktif olarak kullanılabilen HTML5 standardının düzgün yorumlanmadığında ve/veya düzgün gerçeklenmediğinde yol açabileceği zafiyetleri ve İnternet kullanıcılarını bu zafiyetlere maruz bırakmamak için uyulması gereken güvenlik prensiplerini sunmaktadır. Makalenin 2. kısmında HTML5 standardında tanımlanan belli başlı yenilikler, bu yeniliklerin gereksinimleri ve yol açabileceği olası zafiyetlerle güvenli web yazılımı geliştirme prensipleri sunulacaktır. 3. kısımda bu

güvenlik prensiplerinin bir özeti ve HTML5 güvenliği üzerine çalışacaklar için öneriler bulunacak ve son kısımda HTML5 güvenliği çalışmasından çıkarılan sonuçlara yer verilecektir.

2. HTML5 Yenilikleri

2004 yılında çalışmalarına başlanan HTML5 standardı, İnternetin var olan ve yakın gelecekte var olacak ihtiyaçlarını karşılamak üzere geniş bir topluluğun önerileri doğrultusunda geliştirilmiş ve bu geliştirme süreci 2011 yılı Mayıs ayında son halini almıştır. Bu süreç sırasında Web 2.0 olarak da bilinen ve temelde enteraktif ve birlikte çalışabilir bir ağ inşa edilmesi için gereken değişiklikler HTML standardına eklenmiştir. Bu temel değişiklikler ve karşıladığı gereksinimler aşağıdaki konu başlıklarında incelenmektedir.

2.1. Alanlar Arası İletişim (Cross Domain Messaging)

Web sitelerinin güvenlikli bir şekilde birlikte çalışabilirliği Web 2.0 gelişmeleri ile ihtiyaç duyulmaya başlanan en temel gereksinimlerinden biridir. Bu süreçte tarayıcılar, gizlilik ve güvenlik gereksinimlerinden dolayı farklı alanlardaki kaynakların birbirleriyle kuralsız bir şekilde haberleşmelerini engellemiştir[4,5]. HTML4 standardında bahsedilmeyen bu ihtiyaç HTML5 desteklenene kadar yan yollarla çözülmeye çalışılmış; bu da ya genel-geçer çözümler üretmeyi zorlaştırmış ya da çok ciddi güvenlik zafiyetlerine sebep olmuştur. HTML5 standardında tanımlanana kadar kullanılan çözümler aşağıdaki gibi özetlenebilir:

- **Flash Alanlar-Arası İletişim Politikası:** Adobe® Flash, bu ihtiyacı güvenli bir şekilde karşılayabilmek için bir kaynaklar-arası iletişim politikası dosyasında güvenilir kaynakları tanımlamış ve kullanıcıya gönderilen bu dosya sayesinde tarayıcıların bağlantı yapabilecekleri alanları sınırlamıştır[6]. Bu protokol DNS-yeniden sorgulama saldırılarına karşı zafiyet göstermektedir[7].

- **JavaScript Alanlar-Arası İletişim Protokolü:** XSS saldırılarına sebebiyet verebileceğinden dolayı alanlar arası gönderilen AJAX istekleri tarayıcılar tarafından uzunca bir süre engellenmiştir. W3C tarafında bu iletişimi sağlayabilecek protokol[8] tanımlandıktan sonra web uygulamaları belirlenen protokoller doğrultusunda tarayıcılar aracılığıyla birbirleriyle iletişim kurmaya başlamıştır.

Barth ve arkadaşları bu yukarıda belirtilen metotların gerçeklemeleri üzerinde yaptıkları çalışmalarda çeşitli zafiyetler tespit etmiş ve sağlayabileceği güvenlik seviyesinden dolayı HTML5 mesajlaşma modeli olan **postMessage** ara yüzünün kullanılmasını tavsiye etmiştir[9].

postMessage ara yüzü, alanlar-arası iletişim ihtiyacını karşılamak üzere HTML5 standardına eklenmiş bir ara yüzdür. Tarayıcılar tarafından gerçeklenmesi gereken bu ara yüz sayesinde, istemci tarafında çalışan JavaScript kodu, sunucuya gitmeden başka alanlarla iletişim kurabileceklerdir. Basit bir şekilde tanımlanan bu ara yüzde doğrulanması gereken gereksinimler;

- **Gizlilik:** Ara yüzde mesaj gönderen taraf mesajın alıcısı olan (targetOrigin) alanları belirtebilmelidir.

- **Kimlik Doğrulama:** Ara yüz ile gelen mesajda mesajın göndericisi olan alan bildirilmelidir.

HTML5 standardını desteklediğini iddia eden tarayıcı tarafından garanti edilmelidir. Bu şekilde alanlar-arası güven probleminde tarayıcı garantör olarak devreye girerek güven sorununu çözecektir.

2.1.1. postMessage Arayüzü

- **Mesaj Gönderme:** window.postMessage ara yüzü standartta şöyle tanımlanmıştır:

```
postMessage(message, targetOrigin);
```

Bu fonksiyonun çağrılması sonucunda tarayıcı *targetOrigin* ile belirtilmiş alanlara bu mesajı ulaştırmakla yükümlüdür. *targetOrigin* parametresi "*" karakteri kabul etmekte ve bu karakter verildiğinde tarayıcı mesajı bütün pencerelere iletmektedir.

- **Mesaj Alma:** postMessage ara yüzünden gelecek mesajları almak için pencerenin mesaj olay dinleyicisi tanımlanmalıdır.

```
window.addEventListener('message',receive,false)
```

receive fonksiyonu içerisinde gelen mesajlar işlenebilir ve gerekli aksiyonlar gerçekleştirilebilir.

2.1.2. HTML5 Alanlar-arası İletişim Zafiyetleri

HTML5 alanlar-arası haberleşme problemini tarayıcıyı garantör olarak basit bir şekilde çözmeyi başarmış fakat çözerken web uygulaması geliştiricilere "giden mesajın gizliliğini sağlamak" ve "gelen mesajda göndericinin kimlik doğrulamasını yapmak" sorumluluklarını yüklemiştir. Bu doğrulamaların önceki uygulamalarda olduğu gibi merkezi bir şekilde yönetilememesi, her mesajda tekrar edilmesi gerekliliği de güvenlik zafiyetine sebebiyet verebilecek bir eksiklik olarak görünmektedir. Protokolün *targetOrigin*

tanımlanırken "*" joker karakterine izin vermesi yine geliştiriciler tarafından yanlış uygulanacak ve veri gizliliğine yönelik sızmalara sebebiyet verecektir. Bu iletişimin düzgün bir şekilde gerçeklenmiş olmaması durumunda iletişim üzerinde aşağıdaki saldırı senaryolarının uygulanması mümkün olacaktır:

- **Mesaj Bütünlüğüne Yönelik Saldırılar:** Gönderici kimliğinin kontrol edilmemesi durumunda postMessage ara yüzü kullanılarak yapılacak sorgulara asıl gönderici yerine kötü niyetli bir saldırgan tarafından cevap verilebilir. Bu dönen cevabın alıcı tarafında işlenişine göre istemci makinede rastgele kod çalıştırılmasına kadar varabilecek bir saldırı senaryosunun uygulanması mümkündür.

- **Gizliliğin İhlaline Yönelik Saldırılar:** postMessage fonksiyonu "*" joker karakteri ile birlikte çağrıldığında tarayıcı gönderilen mesajı yetkili alıcıya değil de bütün dinleyicilere ulaştıracağından, mesajın içeriği kötü niyetli saldırganlar tarafından açık edilebilecektir.

2.1.3. HTML5 ile Alanlar-arası İletişim Güvenliği

postMessage ara yüzü kullanılırken mesajın alıcısı düzgünce tanımlanmalı ve pencereye iletilen mesajlar işlenmeden önce gönderici mutlaka kontrol edilmelidir. Bu mesajlarda girdi doğrulaması mutlaka yapılmalı, "*" joker kartı kesinlikle kullanılmamalıdır.

Protokolün bir başka zafiyeti olan bu kontrollerin her mesajlaşmada uygulanması gerekliliği konusunda da gerekli önlemler alınmalı, *postMessage* fonksiyon çağrılarının güvenliği mutlaka kod gözden geçirme aktivitesine eklenmelidir.

Hanna ve arkadaşları bu zafiyet için "mesuliyet ekonomisi" önerisinde bulunmuşlar[10], fakat bu öneri henüz HTML5 standardında yer bulmamıştır. "Mesuliyet ekonomisi" bu *postMessage* doğrulamalarının geliştirici bırakılmasındansa tarayıcı tarafından yapılmasını önermektedir. Bu şekilde unutkanlık, bilgisizlik, dalgınlık sebebiyle geliştiricilerin zafiyet içeren kod geliştirmelerine engel olmayı hedeflemektedir. Şu anki haliyle bu mümkün görünmemektedir, fakat *postMessage* ara yüzünü kullanılarak hazırlanabilecek kütüphanelerle bu ekonomi sağlanabilir. Her durumda *postMessage* içeren çıktılar ikinci bir gözün gözden geçirme faaliyetine tabi tutulmalıdır.

2.2. Yerel Depolama/Veritabanı

İnternet için çok yeni olmayan fakat HTML5 ile standarda giren yeniliklerden bir diğeri yerel depolama mekanizmalarıdır. Standartta iki şekilde tanımlanan yerel depolama tarayıcılar tarafından anahtar-değer ikilisi yazıp okuma ve SQL veritabanı servisi sağlama şeklinde sağlanacaktır. Bu okuma/yazma ve sorgulama işlemleri sırasında tarayıcılar "Aynı Orijin Politikası"nı uygulayacak ve alan dışı veriye erişime izin vermeyecektir. HTML5 ile yerel depolama mekanizması tanımlanmadan önce de kullanıcı makinelerine veri saklamak için kullanılan metotlar aşağıdaki gibi özetlenebilir.

- **Çerez**: Yeteneği itibariyle herhangi bir veriyi kullanıcı makinesinde saklamak için kullanılabilecek çerezler, zamanla kullanıcı tanımlayıcı değerlerin saklanması için kullanılır olmuştur. Anahtar-değer ikilisi şeklinde yazılabilen ve okunabilen çerezler de "Aynı Orijin Politikası"na tabi olup içerdiği verinin önemi nedeniyle sıkı güvenlik önlemlerine de tabidir. Bu yüzden yanında başka verinin saklanması ve tutulması çok uygulanabilir bir yöntem değildir. Çerezlerin bir başka dezavantajı da bütün istek ve cevaplarla birlikte taşınıyor olmasıdır. Bu durum her istekte sunucuya gitmesi gerekmeyen veriler için gereksiz ağ kaynakları kullanımı manasına geldiğinden çerezler bu tür verileri saklamak için tercih edilebilir bir mekanizma değildir.

- **Adobe® Flash Objeleri:** İstemcide veri depolama ihtiyacına Adobe'nin 2002 yılında geliştirdiği çözümdür. Flash çerezleri olarak tanınmasına rağmen asıl amacı veri depolamaktır.

- **Google Gears:** 2007 yılında Google, açık kaynak kodlu Gears eklentisi ile tarayıcıların veri depolayabilme yeteneği kazanmasını amaçlamıştır. Gears, özetle, SQLite veritabanı ve bu veritabanına erişebilmek için gerekli bir ara yüzden oluşmaktadır.

WHATWG ve W3C veri depolama konusundaki bu çok başlılığı ortadan kaldırmak için standarda yerel depolama ve veritabanını eklemiştir. Yakın zamanda Google, Gears değil de HTML5 standardını takip edeceklerini açıklamıştır[11].

2.2.1. HTML5 Standardında Yerel Depolama

HTML5 standardında tanımlanana göre tarayıcılar JavaScript kodu aracılığıyla yerel depolama alanlarına anahtar-değer ikilileri yazabilir:

localStorage.setItem("key", value);

yazılı olan bu değerleri okuyabilirler:

localStorage.getItem("key");

veya bu değerleri silebilirler:

localStorage.removeItem("key")

Standartta yerel veritabanı olarak SQLite veritabanı belirtilmiş ve tarayıcıların bu veritabanına olan ara yüzü tanımlanmıştır. Yerel veritabanı üzerinde web uygulamaları temel SQL işlemlerinin tamamını gerçekleştirebileceklerdir. Aşağıdaki kod parçası yerel veritabanı açıp içerisinde bir tablo oluşturmaktadır.

```
openDatabase('test', '1.0', 'Yerel Veritabanı', 5*1024*1024, function(db){

    db.ChangeVersion('', '1.0', function(t){

        t.ExecuteSql('CREATE TABLE settings(id, name, value)');}, error);

});
```

2.2.2. HTML5 Yerel Depolama Zafiyetleri

Yerel depolama mekanizmaları web uygulamalarının kabiliyetini oldukça arttırırken sunucu ve istemci arasında akan trafiğin azaltılması konusunda da büyük etkisi olacaktır. Bununla birlikte yerel depolama mekanizmaları kullanılırken alınması gereken güvenlik önlemleri gözden kaçırılmamalıdır. Kullanıcı makinelerinde veri saklamak aşağıda sıralanan bir dizi güvenlik zafiyetine sebep olabilecektir:

- **Hassas Verilerin Sızması:** Kullanıcı makinelerine kaydedilebilecek herhangi bir verinin yetkisiz erişim, veri bütünlüğü, girdi geçerliliği gibi problemlerin oluşabilecektir. İstemci depolama alanları sunucunun hâkimiyeti olan alanlar olmadığından buraya yazılan veriler başkaları tarafından okunabilir, değiştirilebilir ve/veya silinebilir. Özellikle ortak kullanımdaki bilgisayarlarda bu tehdidin gerçekleşme ihtimali daha yüksektir.

- **XSS ile SQL Enjeksiyonu:** OWASP Web Zafiyet Top 10 listesinin ilk iki sırasını uzunca bir süredir işgal eden XSS ve SQL Enjeksiyonu, HTML5 standardında tanımlanan ve JavaScript ile veritabanına erişimi sağlayan ara yüz ile bir arada uygulanabilecek iki saldırı olacaktır. XSS veya herhangi bir başka şekilde istemci makinesine erişen saldırganlar burada saklanan verileri değiştirdiği durumda sunucunun haberi dahi olmadan bu veritabanını kötüye kullanabileceklerdir.

2.2.3. HTML5 Yerel Depolama Güvenliği

HTML5 standardı ile tanımlanan yerel depolama mekanizmalarının güvenlik konusundaki temel prensibi hiç kuşkusuz gizlilik üzerine olacaktır. Bu veritabanları sunucu hâkimiyeti dışında kaldıkları için hiçbir surette hassasiyet taşıyan veriler buralarda saklanmamalıdır. Bu konudaki geliştiricilerin benimseyebileceği prensip "sunucu tarafında açık bir şekilde saklanamayacak veriler, istemci tarafında hiçbir surette saklanmamalıdır". Her ne kadar tarayıcılar yerel depolama mekanizmalarında "Aynı Alan Prensibi"ne uygun hareket etseler de, bu yerel makinelerin güvenliğinin sağlanamayacağı, ortak kullanıma açık olabilecekleri gibi sebeplerden dolayı hassas verileri buralarda saklamamalıdırlar. En nihayetinde tarayıcılar alan uyumuna bakmakta, aynı alan için kullanıcıları birbirinden ayırt edemezken, kullanıcının kendisinin mi yoksa kullanıcı tarafından çalıştırılan kötü niyetli kod parçalarının mı veritabanına erişmek istediğini kontrol edemezler. Başka bir deyişle, hassasiyet içermeyen ve yetkisiz kullanıcıların da erişebildiği veriler ancak istemci makinelerinde saklanabilirler.

Yerelde depolanan verilerin bütünlüğü de tehlike altında olacağından web uygulamaları bu kaynaktan okudukları veriye; genelde hiçbir yerden okudukları verilere, hiçbir şekilde güvenmemeli, verinin bütünlüğü kullanılmadan önce mutlaka doğrulanmalıdır.

HTML5 standardında herhangi bir alanın oluşturduğu veritabanlarını birbirinden ayırt edecek özellik veritabanı ismidir. Aynı alanın farklı kullanıcıları arasında ayrım yapabilecek bu isim belirlenirken mutlaka tekil bir isim seçilmeli, bu isim tahmin edilebilir olmamalıdır. Aksi takdirde saklanan veri yetkisiz erişime kolayca izin verecek ve veri güvenliği tehlikeye girecektir.

Veritabanı erişimi güvenliği konusunda yapılan çalışmalar HTML5 yerel veritabanları için de geçerlidir. Bu veritabanı iletişimi için obje-ilişkisel eşleştirim kütüphaneleri piyasaya çıkmıştır. Bu kütüphaneler veritabanı iletişiminde kullanıcı girdilerini temizlemeli, olası enjeksiyon saldırılarına karşı veritabanını korumalıdırlar.

Kullanıcılar yerel depolama işleminden önce uyarılmalı, onayladıkları takdirde istemcilere veri kaydedilmelidir. Ayrıca, kullanıcıların farklı istemcilerden erişim gerçekleştirmeleri durumunda bu onaylama mekanizması tekrar devreye alınmalıdır.

2.3. HTML5 Web Soket

HTML5 standardı ile web dünyasına gelen bir diğer devrimsel nitelikteki yenilik de web uygulamalarının istemcilerin alt seviye kaynaklarına ulaşıp ağ iletişimi için soket açmalarını sağlayacak ara yüzün standarda eklenmesidir. Böylelikle "bağlantısız protokol" olarak tanımlanan http üzerinde çalışan web uygulamaları "bağlantılı" bir alternatife sahip olmuştur. Bu soket ara yüzü de "Aynı Alan Prensibi" ile çalışmakta ve tarayıcılar soket açtıkları alan ile soketi açan uygulamanın alanını eşleştirmektedir. Böylelikle daha önce tarayıcılara

yüklenen Adobe® Flash ve JAVA Applet gibi eklentilerle erişilen alt seviye kaynaklara JavaScript ile erişilebilecek, istemci ve sunucu arasında alternatif bir kanal açılabilecektir.

Yapısı itibariyle alanlar arası iletişim ile benzerlik gösteren web soket ara yüzü, yine üzerinde uygulanabilecek saldırı senaryoları açısından da benzerlik göstermektedir. Barth ve arkadaşları [7] bu saldırı senaryolarını uygulamış ve bulgularını standart geliştiricileriyle paylaşmışlardır. Soket ara yüzü özellikle güvenlik açısından henüz olgunlaşmamış bir yenilik olduğundan daha önceden destekledikleri halde Opera ve Firefox yeni sürümlerinden soket desteğini kaldırmışlardır.

2.3.1. HTML5 Web Soket Ara Yüzü

Standartta tanımlanan web soket ara yüzünü kullanarak bir bağlantı oluşturabilmek için:

```
var conn=new
WebSocket("ws://html5.websocket.com");
```

cümlesi kullanılabilir. Kanal güvenliği gerekli olduğu durumlarda soket adresinde *ws://* yerine *wss://* kullanılmalıdır. Daha sonra bu bağlantı objesinin açma, kapama ve mesaj alma ara yüzlerine ilgili fonksiyonlar bağlanarak soket ara yüzü kullanılabilir. Soket ara yüzü kullanarak sunucuya mesaj göndermek için de:

```
conn.postMessage("Hello World!");
```

fonksiyonu kullanılmalıdır. İstemci ile sunucu arasında bir soket açılmak istendiğinde, önce 80inci portta HTTP üzerinden bir anlaşma sağlanır, daha sonra bu bağlantı istemci tarafından soket bağlantısı olarak başka bir porta yükseltilir. Bu mekanizma bağlantının güvenlik duvarı engelini aşması için gereklidir.

2.3.2. Web Soket Zafiyetleri

Web soket ara yüzünün *postMessage* ara yüzünde bahsedilen saldırılara karşı zafiyeti bulunduğu yine Barth ve arkadaşları tarafından gösterilmiştir[7]. Web soketin *postMessage* ara yüzünden farklı olarak alt seviye kaynaklara erişiminin olması onu daha güçlü ve aynı zamanda daha tehlikeli bir hale getirmektedir. Her ne kadar tarayıcı tarafından "Aynı Alan Prensibi"ne göre çalıştırılsa da, bahsedilen senaryodaki DNS-yeniden sorgulama saldırıları ile bu kontrol aşılabilmekte ve bundan sonrası saldırganın hayal gücüne kalmaktadır. Saldırgan bu noktadan sonra, ağ üzerinde makine ve port taraması yaparak başlayacağı saldırı işlemini bulduğu bir açıklık aracılığıyla ağ üzerindeki bir makineyi ele geçirmeye kadar varabilecek saldırıları da yine bu soket ara yüzü üzerinden gerçekleştirebilmektedir. Barth aynı makalesinde soket bağlantısı için daha güvenli bir protokol önermiş ve önerisi HTML5 geliştiricileri tarafından incelemeye alınmıştır.

2.3.3. HTML5 Web Soket Güvenliği

HTML5 soket ara yüzünde gerekli olan güvenlik kontrolleri tarayıcı tarafından yapılmaktadır. Tarayıcı uygulamaların yalnızca istemci tarafından uygulama alanına soket açmasına izin vererek olası saldırıların önüne geçmektedir.

Bu kontrollere rağmen geliştiricilerin dikkat etmesi gereken hususlar bulunmaktadır. Bunlardan en önemlisi hassas verilerin soket ara yüzü üzerinde taşınacağı durumda güvenli soket ara yüzü kullanılması gerektiğidir. Bu şekilde veriler kanal üzerinde şifrelenmiş bir şekilde taşınacak ve uçtan uca güvenlik sağlanmış olacaktır.

HTML5 standardında web soket ara yüzü tanımı henüz olgunlaşmamış ve yaygın bir şekilde kullanılmaya başlanmamıştır. Firefox ve Opera bu ara yüzdeki zafiyetlerden dolayı yeni sürümlerinde soket ara yüzünden desteğini çekmiş ve standardın güvenli bir şekilde tanımlanmasını beklemeye başlamıştır.

JavaScript üzerinden soket seviyesinde istemci kaynaklarına erişim İnternet dünyası için yeni sayılabilecek bir uygulamadır. Geliştiricilere alternatif kanal ile sunucu iletişimi sağlayabilecekleri bir altyapı sunan soket ara yüzü, kullanıcıların da kullanım kalitesini arttıracağı konusunda bir şüphe bulunmamaktadır. Fakat, soket ara yüzü henüz tam manasıyla yaygınlaşmamış güvenliği açısından derinlemesine incelenememiştir. Kullanılmaya başlandıkça gerek tarayıcı gerçeklemeleri gerekse protokolün kendisi güvenlik açısından daha detaylı bir şekilde incelenebilecek ve olası zafiyetleri görülebilecektir. Güvenlik açısından belirli bir olgunluğa erişmeden soket ara yüzünün kullanılması hassas verilerle çalışan uygulamalar için pek uygulanabilir durmamaktadır. Bunun yerine önce standardın olgunlaşması, daha sonra İnternette kullanımının yaygınlaşması beklenmelidir.

2.4. Öğe Özelliklerin Kötüye Kullanımı

Web üzerinde gerçekleştirilen saldırı senaryolarının bir kısmında HTML öğelerinin özellikleri kullanılmaktadır. Örneğin CSRF olarak bilinen Alanlar Arası İstek Düzme saldırılarında URI verilen öğeler sonucu yapılan GET istekleri kullanılarak tarayıcılara kötü amaçlı istekler yaptırılabilir[12]. Bu saldırılar hali hazırda İnternet üzerinde gerçekleştirilmekte, geliştiriciler ve güvenlikçiler tarafından da yoğun bir şekilde incelenmektedir.

HTML5 standardının tanımında web dünyasına yeni giren öğeler kadar eski öğeler için yeni tanımlanan özellikler de standardı oldukça genişletmekte ve güçlendirmektedir. Fakat, bu genişletme ve güçlendirme aynı zamanda kötü niyetli saldırganlar için de yeni saldırı yüzeyleri manasına gelmektedir. Aşağıda bu türde kötüye kullanılabilecek yeni özelliklerden birkaçı listelenmiştir:

- **formaction:** formlar üzerinde HTML öğelerinin işlem yapmasını sağlayan bu özellik XSS saldırılarına açıktır.

- **autofocus:** HTML5 ile web dünyasına giren autofocus, elemanların otomatik olarak focus almasını sağlar. onfocus veya onblur olaylarıyla birlikte kullanıldığında kullanıcı hareketi olmaksızın saldırı gerçekleştirilebilir.

- **onerror, onscroll:** Yeni tanımlanan bu olaylar ile kullanıcı farkında olmadan tarayıcıda kod çalıştırılabilir.

HTML öğelerinin özelliklerinin kötüye kullanılmasını inceleyen bu liste HTML5 ile artık daha da kabarık bir hale gelmiştir[13]. Bu listede verilen zafiyet durumlarının oluşmasına engel olmak için HTML kodlarının oluşturulmasında kullanılan kullanıcı girdileri mutlaka doğrulanmalıdır. Yapılması gereken doğrulamalar ve dikkat edilmesi gereken öğler ve/veya öğe özellikleri için HTML5 güvenlik listeleri kontrol edilebilir[13].

Bu noktada altı çizilmesi gereken bir başka husus da HTML5 özelliklerinin tarayıcıların desteklemesiyle birlikte artık uygulamaların kullanımına sunulduğu gerçeğidir. Bu uygulamalar HTML5 ile geliştirilmemiş dahi olsa tarayıcılar HTML5 öğelerini ve özelliklerini tanırlar. Bu durum geliştiricilerin ve güvenlikçilerin hali hazırda çalışan uygulamaları HTML5 kontrolleri ile birlikte test etmeleri gereksinimini doğurmuştur. Hali hazırda web uygulamaları için geliştirilen ve kullanılan test araçlarında HTML5 öğeleri ve özellikleri dikkate alınmadığı gibi, bu uygulamaların güvenlik açısından test edilmelerinde ve sıkılaştırılmalarında da bu özellikler göz önünde bulundurulmamıştır. Aynı şekilde web uygulamalarının otomatik güvenlik testlerini yapan araçlarda da bu hassasiyet eklenmemiş olabilir. Geliştiricilerin ve güvenlikçilerin bu duruma dikkat etmesi gerekmektedir. Özet olarak, HTML4 standardı için geliştirilen ve güvenli hale getirilen uygulamalar HTML5 için de güvenlidir önermesi tehlikeli ve yanlıştır. Geliştiriciler ve güvenlikçilerin web üzerinde servis ettikleri uygulamaları HTML5 öğelerini dikkate alarak tekrar test etmesi ve güvenlik hedeflerini doğrulaması gerekmektedir. Web uygulaması güvenlik testi gerçekleştiren otomatik araçlar da en kısa zamanda yeni öğeleri ve özellikleri desteklemeli, testlerini bu yeni saldırı yüzeyinde gerçekleştirmelidir.

Web uygulaması güvenliği denildiğinde es geçilemeyecek bir konu da girdi doğrulamasıdır. Girdi doğrulaması temelde hiçbir kaynaktan gelen girdiye güvenilmemesi, girdinin doğruluğunun teyit edilmesi demektir. Bu doğrulamayı web uygulamalarının dışarıya açık bütün ara yüzlerinde, yeterli ve doğru biçimde, aynı zamanda HTML5 öğelerini de hesaba katarak yapan uygulamalar güvenli olduklarını iddia edebilirler.

3. HTML5 Güvenliği

Şüphesiz HTML5 İnternet dünyasında yeni bir çığır açacak ve hem geliştiriciler için hem de kullanıcılar için vazgeçilmez uygulamalara imkân tanıyacaktır. Burada gözden kaçırılmaması gereken HTML5 standardının henüz taslak halde bulunduğu gerçeğidir. Taslak halde bulunan bir standarttan yola çıkılarak hızlı bir şekilde tarayıcılar tarafından desteklenen bu yeni özellikler bugüne kadar karşılaşılmamış, bilinmeyen zafiyetlere sebebiyet verebileceği gibi, bilinen zafiyetlerin farklı biçimlerde tekrar ortaya çıkartılmasına da sebep olabilirler.

Burada tekrar altını çizmemiz gereken bir başka nokta tarayıcıların desteklemesiyle birlikte bütün İnternet kullanıcılarının HTML5 kullanmaya başlamasıdır. Bu geçiş sırasında kullanıcının onayı alınmadığı gibi, otomatik güncellemeyle tarayıcısını güncelleyen kullanıcılar HTML5 zafiyetlerinin oluşturabileceği tehditlerin hedefi haline gelmektedir. Burada geliştiricilere önemli bir sorumluluk düşmektedir. Bugüne kadar İnternet üzerinden servis ettikleri yazılımlar HTML5 standardının tarayıcılarla desteklenmesinden sonra güvenlik zafiyeti gösterebiliriler. Nispeten yeni ve tamamlanmamış olan bu zafiyetlere karşı güvenlik testleri derinlemesine ve eksiksiz bir şekilde bir an evvel yapılmalı, yeni geliştirilen uygulamalarda da bu yazıda bir kısmı verilen güvenlik önerilerine uygun bir şekilde geliştirme yapılmalıdır.

HTML5 güvenlik testi yapmaya aday olan araçlar da var olan zafiyet ve tehdit veritabanlarını güncellemeli ve tarayıcılar tarafından desteklenen özellikler bütününü testlerine dâhil etmelidirler.

Son olarak vurgulamak istediğimiz bir diğer nokta da Firefox ve Opera'nın yeni sürümlerde zafiyet ve açıklıklara sebebiyet verebileceğinden dolayı soket ara yüzünden desteğini çekmesidir. Süregelen tarayıcı savaşlarında öne çıkma arzusunun yerine kullanıcılarının güvenliğini ön plana çıkarması bu iki tarayıcıyı güvenlik açısından diğerlerinin önüne geçirmeyi başarmıştır.

Bütün bu önlem ve kısıtlamalarla birlikte İnternet üzerinde birlikte çalışabilirliği önemli ölçüde arttıracak olan HTML5 standardının kullanımı ve yaygınlaşması desteklenmeli; kritik olmayan uygulamalar aracılığıyla standarttaki zafiyetler bir an evvel tespit edilip kapatılarak HTML5 standardı bütün geliştiricilerin kullanımına açılmalıdır.

4. Sonuçlar

HTML5 standardı henüz taslak halinde olmasına rağmen çözüm getirdiği sorunlar ve web üzerinde çalışan uygulamaların yeteneklerini genişletmesi açısından özellikle Facebook, Google, Apple gibi üreticiler tarafından çoktan benimsenmiş ve uygulamaya konmuştur. Popüler tarayıcılar da bu sürece yoğun bir şekilde destek vermekte ve standardın olgunlaşması için çaba sarf etmektedirler. Bu görünüme göre çok uzak olmayan bir zamanda HTML5 standardının yetenekleri kullanıcılar tarafından istenmeye başlayacak ve web üzerinde geliştirme yapan bütün üreticiler standardı programlarına alacaktır. Böyle bir ortamda web uygulamalarının HTML5 standardına kısa zamanda keskin bir geçiş yapacaklarını öngörmek çok da yanlış olmaz.

Bu noktada, özellikle olgun bir HTML4 ile geliştirme hayatlarına başlamış geliştiricilerin HTML5 zafiyetleri konusunda bilgi sahibi olmaları, geliştirme sırasında güvenlik gereksinimlerini de göz önünde bulundurmaları kullanıcılar için hayati önem taşımaktadır. Bu ihtiyaç da web geliştiricilerinin web güvenliği ile ilgili çalışmaları her zamankinden daha yakından ve daha çok takip etmelerini gerektirmektedir.

5. Kaynakça

[1] D.Raggett, A. Le Hors ve I. Jacobs. "HTML 4.01 Specification". http://www.w3.org/TR/html401/, 24.12.1999 [11.06.2011]

[2] Wikipedia. "HTML5". http://en.wikipedia.org/wiki/HTML5, 12.06.2011 [12.06.2011]

[3] W3C. "W3C Confirms May 2011 for HTML5 Last Call, Targets 2014 for HTML5 Standard". 14.02.2011 [12.06.2011]

[4] OWASP. "Cross-site Scripting (XSS)". https://www.owasp.org/index.php/Cross-site_Scripting_(XSS), 20.10.2010[11.06.2011]

[5] W3C. "Same-Origin Policy". http://www.w3.org/Security/wiki/Same_Origin_Policy, 06.06.2010 [15.05.2011]

[6] Adobe. "Cross-domain Policy File Specification". http://www.adobe.com/devnet/articles/crossdomain_policy_file_spec.html, 22.01.2010 [15.05.2011].

[7] L. Huang, E. Chen, A. Barth, E. Rescorla, ve C. Jackson. "Talking to Yourself for Fun and Profit". 5inci "Web 2.0 Security and Privacy" Konferansı, California, US, 2011.

[8] A.Van Kesteren. "Cross Origin Resource Sharing". 27.07.2010 [15.05.2011].

[9] A.Barth, C. Jackson, ve W.Li. "Attacks on JavaScript mahsup communication". 3üncü "Web 2.0 Security and Privacy" Konferansı, California, US, 2009.

[10] S. Hanna, E.C.R. Shin, D. Akhawe, A. Boehm, P. Saxena ve D. Song. "The Emperor's New API: On the (In)Secure Usage of New Client-side Primitives". 4üncü "Web 2.0 Security and Privacy Conference", California, US, 2010.

[11] I. Fette. "Hello HTML5". http://gearsblog.blogspot.com/2010/02/hello-html5.html, 19.02.2010 [25.05.2011].

[12] OWASP. "Cross-site Request Forgery (CSRF)". https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF), 09.04.2010 [11.06.2011].

[13] "HTML5 Security Cheatsheet". http://html5sec.org/. [17.05.2011]