

# Meta-Information Cross Site Scripting

---

*Tyler Reguly*  
*Lead Security Research Engineer, nCircle*

As more and more of our daily activities move online, we find ourselves inundated with web-based tools. Step back and forget about Twitter, Facebook, Google Wave and even Google Docs. Think about those network administration utilities that so many web masters and SMB administrators rely on. Tools that perform a whois lookup, resolve DNS records, or simply query the headers of a web server. They're taking the meta-information provided by various services and displaying it within the rendered website. These web-based services introduce a class of Cross Site Scripting (XSS) that can't be captured by the current categories of XSS.

Before we delve deeper into the world of Meta-Information XSS (miXSS), we need to first understand why the current XSS attacks do not work to classify this instance. The three current types are: Reflected, Persistent and DOM Based.

Reflected XSS refers to an attack that occurs when user input is reflected back at the user. This means that you provide the malicious data as user input and the web application simply echoes the data back to you.

Persistent XSS refers to an attack that stores user input, allowing it to affect a much broader scope of visitors. An attack may be stored in the database and displayed to all visitors, rather than just the visitor that provided the malicious input.

The final type, DOM Based XSS, refers to attacks that modify the Document Object Model directly and don't require data in the HTTP response.

None of these really capture the process that occurs when you are dealing with Meta-Information Cross Site Scripting. With miXSS, the input that the user provides is completely valid, and properly sanitized, this rules out Reflected XSS and since we aren't storing the user input, Persistent XSS can also be disregarded. Finally, since we're not interacting with the DOM, we can eliminate this type of attack.

So what is miXSS? It is a form of attack that represents aspects of both Reflected and Persistent attacks, yet is defined by neither. It is valid user input provided to a service, the service then utilizes the user provided data to gather metadata and display it for the user. It is in this data that the Cross Site Scripting occurs.

Consider this example. A DNS TXT record which contains the value "`<script>alert(1)</script>`" and a service designed to gather DNS TXT records for the purpose of testing SPF (sender policy framework) records. The user provides the domain name pointing to the TXT record; while the service resolves the TXT data and displays the data to the user. Since the data contains JavaScript, the returned data is processed and successful Cross Site Scripting has occurred.

### Affected services include:

- GET/POST Based DNS Resolution Services
- GET Based SSL Verification Services
- GET/POST Based HTTP Server Header Retrieval Services
- Client Based HTTP Header Display Services
- GET/POST Based Whois Clients
- POST Based SMTP Verification Services

### Example Server Meta-Information:

#### DNS

```
C:\>nslookup -q=TXT redirect.sslfail.com ns.slashconslashcon.com
```

```
Server: ns.slashconslashcon.com
```

```
Address: 74.208.78.200
```

```
redirect.sslfail.com text =
```

```
"<script language='JavaScript'>window.location='http://www.sslfail.com';</script>"
```

#### SSL

```
Loading 'screen' into random state - done
```

```
CONNECTED(00000754)
```

```
depth=0
```

```
/C=CA/ST=<script>alert(1)</script>/L=<script>alert(1)</script>/O=<script>alert(1)</script>/OU=<script>
```

```
alert(1)</script>/CN=<script>alert(1)</script>/emailAddress=<script>alert(1)</script>
```

```
verify error:num=18:self signed certificate
```

```
verify return:1
```

```
depth=0
```

```
/C=CA/ST=<script>alert(1)</script>/L=<script>alert(1)</script>/O=<script>alert(1)</script>/OU=<script>
```

```
alert(1)</script>/CN=<script>alert(1)</script>/emailAddress=<script>alert(1)</script>
```

```
verify return:1
```

### *HTTP*

HTTP/1.0 200 OK

Date: Wed, 02 Dec 2009 04:39:33 GMT

Server: Apache/2

X-Powered-By: PHP/5.2.4-2ubuntu5.9

X-XSS: <script>alert(1)</script>

Connection: close

Content-Type: text/html; charset=UTF-8

### *SMTP*

220 [redacted] ESMTP Postfix <script>alert(1)</script>

### *Whois*

registrant-firstname: [redacted]  
registrant-lastname: [redacted]  
registrant-organization: <script>alert(1)</script>  
registrant-street1: [redacted]  
registrant-street2: <script>alert(1)</script>  
registrant-pcode: [redacted]  
registrant-state: [redacted]  
registrant-city: [redacted]  
registrant-ccode: [redacted]  
registrant-phone: [redacted]  
registrant-email: [redacted]