SkullSecurity » Blog Archive » Stuffing Javascript into DNS names

SkullSecurity » Blog Archive » Stuffing Javascript into DNS names - Ron Bowes, skullsecurity.org.

- Published: date not stated
- Original: <https://web.archive.org/web/20170903113359/http://www.skullsecurity.org/blog/2010/stuffing-javascript-into-dns-names>
- Current location: <http://www.skullsecurity.org/blog/2010/stuffing-javascript-into-dns-names>
- Preserved from: http://www.skullsecurity.org/blog/2010/stuffing-javascript-into-dns-names (stored) on 2026-08-11
- Capture timestamp: 20121031001203
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

SkullSecurity » Blog Archive » Stuffing Javascript into DNS names

Stuffing Javascript into DNS names

Filed under: DNS, Hacking, Tools

Greetings!

Today seemed like a fun day to write about a really cool vector for cross-site scripting I found. In my testing, this attack is pretty specific and, in some ways, useless, but I strongly suspect that, with resources I don't have access to, this can trigger stored cross-site scripting in some pretty nasty places. But I'll get to that!

Interestingly enough, between the time that I wrote this blog/tool and published it, nCircle researchers have said almost the same thing (paper (pdf)). The major difference is, I released a tool to do it and demonstrate actual examples.

dnsxss

If you've installed nbtool, you may have noticed that, among other programs it comes with, one of them is called dnsxss. Take a look at the wiki page for more information, but what it does is, essentially, respond to DNS requests for CNAME, MX, TXT, and NS records with Javascript code. Unless you want some specific code, all you have to do is run it:

```
# dnsxss
Listening for requests on 0.0.0.0:53
Will response to queries with:
<script/src='http://www.skullsecurity.org/test-js.js'></script>
```

Pass -h or --help for a list of arguments.

Now, an observant reader will realize that this isn't valid HTML. Unfortunately, as far as I can tell, there's no way to send a space through DNS, so you have to come up with some space-free code. The best I could find was replacing spaces with a '/', which will work on Firefox but not IE (I haven't tested anything else). If anybody can think of a better way to write HTML without spaces, let me know. The next best solution is using the TXT query, which DOES allow spaces. dnsxss will reply to TXT queries with well formed Javascript.

For what it's worth, dnsxss will answer A or AAAA requests with localhost (127.0.0.1 or ::1) -- if somebody does a lookup, they'll get an odd answer that won't immediately lead back to you.

Let's break stuff!

So, what can you do with this?

Well, fortunately (or unfortunately? depends who you are), most sites don't echo back DNS records. But, some do. I picked the first three sites from a Google query and tested them out. All three were vulnerable. And I very much doubt that any programmers even *consider* filtering DNS responses. I mean, who expects DNS responses to contain HTML?

And, furthermore, if I can sneak HTML code into pretty much any site that looks up DNS names due to lack of filtering, how about SQL injection? If the response is inserted into the database without filtering for SQL characters, which I would bet they are on at least some sites, you now have an avenue for SQL injection! And, better yet, there's a decent chance that the requests won't be logged because a) it's coming through a backchannel so it's not going to be in their Web server logs, b) the statements containing SQL injection won't be inserted to your logging table (since they wouldn't be valid queries), and c) you can turn off the DNS server whenever you want, and no trace will be left that you were ever doing it (except short-lived caches).

As I mentioned earlier, I took the first three sites from a google query I crafted, and all three were vulnerable. I emailed the administrators of all three sites, and two of them replied thanking me. Both told me that it was a really interesting vector, and that they would fix their sites as soon as possible. The third I haven't heard back from. But the point is, on three random sites, none had even considered implementing any defenses.

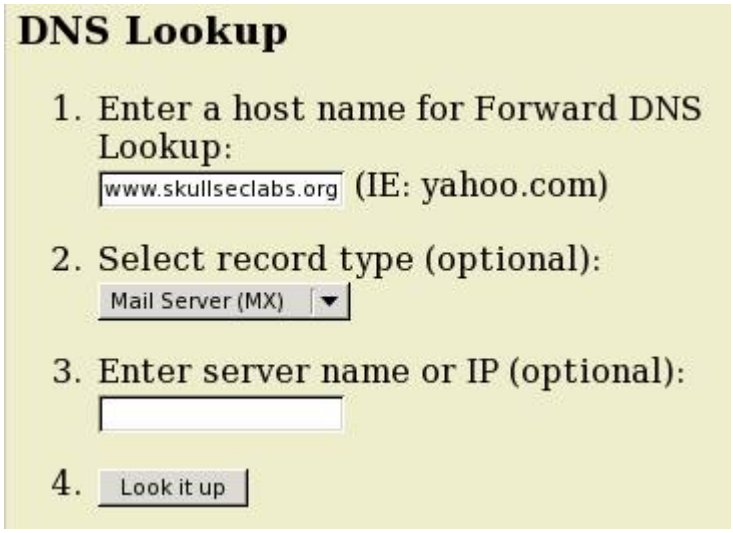
Let's take a look at the examples! But first, here are some notes on them:

- The examples use skullseclabs.org, which is the domain I use for all my testing -- if you plan on testing these yourself, you'll have to register your own domain
- The examples use "/* */" to conceal the space, but I realized later that a single "/" works just as well

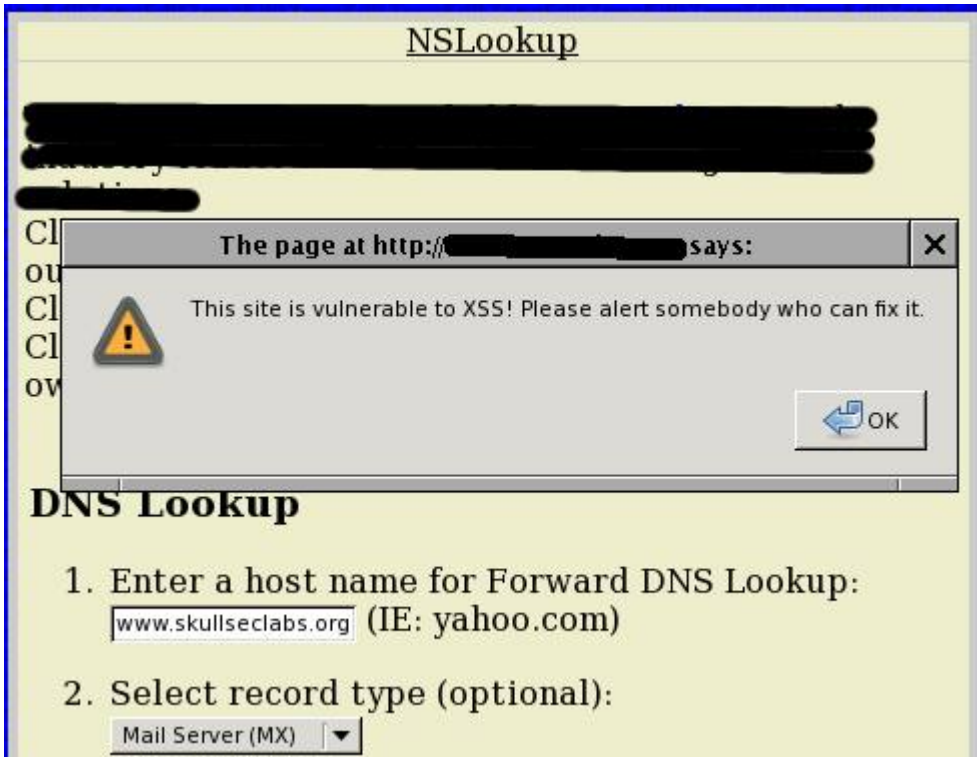
So, without further ado, here are some screenshots of the sites. I anonymized them a little, though a clever attacker could likely Google hack them.

Site 1

The form:



The result:

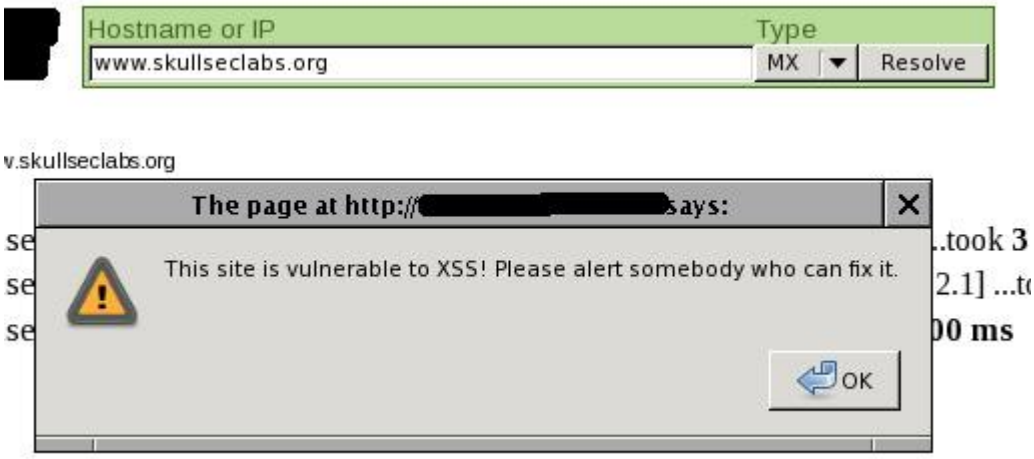


The source: [image: image]

Site 2

The form: [image: image]

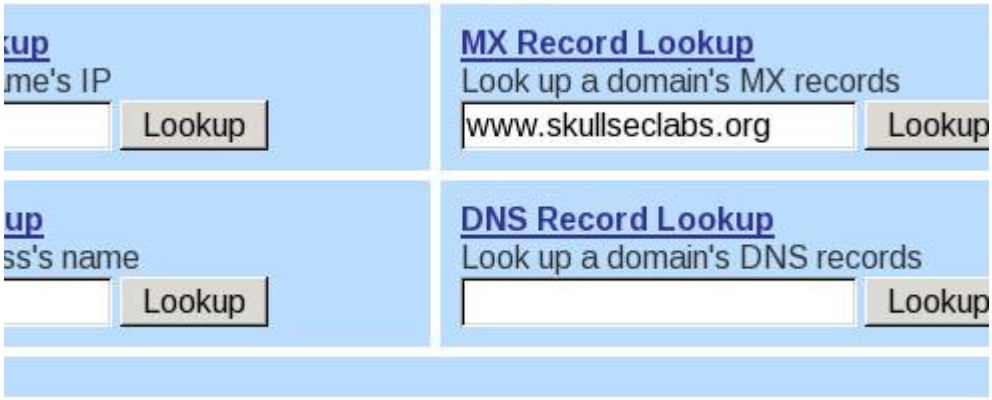
The result:



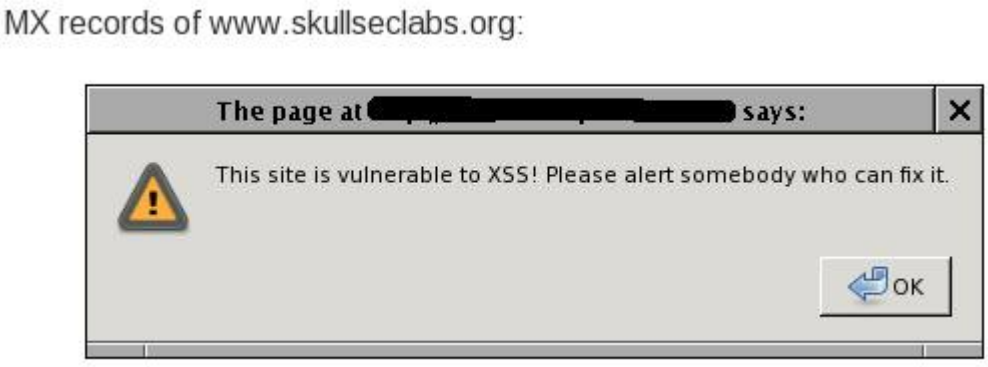
The source: [image: image]

Site 3

The form:



The result:



The source: [image: image]

So there we go...

Three sites, none of which filter out my cross-site scripting attempts. Fun!

Weaponization

The problem is, this only affects a small percentage of sites -- those that will look up domains and display them for you. How can this be used against more targets?

Well, I have two ideas:

- As I mentioned earlier, I'd bet money that there are other forms of attacks through these avenues -- I'd be surprised if SQL injection didn't exist
- Can you stuff javascript into **reverse** DNS entries?

The second point, I suspect, is where we're going to have fun. I can think of countless security devices, from firewalls to vulnerability management tools to proxy servers, with Web interfaces that display reverse DNS records. Not to mention tools where administrators are shown reverse lookups -- forums, for example. Another avenue is logfiles, which are normally visible to administrators.

In all of these cases, if you can stuff Javascript into reverse DNS lookups, you will likely find some very interesting vulnerabilities. Plus, you can instantly see when somebody hits one, and, more often than not, you can clean up your tracks quite well.

I don't have access to any domains where I control the reverse DNS records, but if anybody does I'd love to test this out!