DNS bug leaks by matasano
Matasano security leaked out the bug and then tried to get the cat back into the bag.. pretty funny! :)
Anyway, here's the copy of their post, I am eager to experiment with some actual code now ;-)
Originally was posted at www.matasano.com/log/1103/reliable-dns-forgery-in-2008-kaminskys-discovery/

Reliable DNS Forgery in 2008: Kaminsky's Discovery
from Matasano Chargen by ecopeland
0.

The cat is out of the bag. Yes, Halvar Flake figured out the flaw Dan Kaminsky will announce at Black Hat.
1.

Pretend for the moment that you know only the basic function of DNS — that it translates WWW.VICTIM.COM into 1.2.3.4. The code that does this is called a resolver. Each time the resolver contacts the DNS to translate names to addresses, it creates a packet called a query. The exchange of packets is called a transaction. Since the number of packets flying about on the internet requires scientific notation to express, you can imagine there has to be some way of not mixing them up.

Bob goes to to a deli, to get a sandwich. Bob walks up to the counter, takes a pointy ticket from a round red dispenser. The ticket has a number on it. This will be Bob's unique identifier for his sandwich acquisition transaction. Note that the number will probably be used twice — once when he is called to the counter to place his order and again when he's called back to get his sandwich. If you're wondering, Bob likes ham on rye with no onions.

If you've got this, you have the concept of transaction IDs, which are numbers assigned to keep different transactions in order. Conveniently, the first sixteen bits of a DNS packet is just such a unique identifier. It's called a query id (QID). And with the efficiency of the deli, the QID is used for multiple transactions.
2.

Until very recently, there were two basic classes of DNS vulnerabilities. One of them involves mucking about with the QID in DNS packets and the other requires you to know the Deep Magic.

First, QIDs.

Bob's a resolver and Alice is a content DNS server. Bob asks Alice for the address of WWW.VICTIM.COM. The answer is 1.2.3.4. Mallory would like the answer to be 6.6.6.0.

It is a (now not) secret shame of mine that for a great deal of my career, creating and sending packets was, to me, Deep Magic. Then it became part of my job, and I learned that it is surprisingly trivial. So put aside the idea that forging IP packets is the hard part of poisoning DNS. If I'm Mallory and I'm attacking Bob, how can he distinguish my packets from Alice's? Because I can't see the QID in his request, and the QID in my response won't match. The QID is the only thing protecting the DNS from Mallory (me).

QID attacks began in the olden days, when BIND simply incremented the QID with every query response. If you can remember 1995, here's a workable DNS attack. Think fast: 9372 + 1. Did you get 9372, or even miss and get 9373? You win, Alice loses. Mallory sends a constant stream of DNS responses for WWW.VICTIM.COM. All are quietly discarded —- until Mallory gets Bob to query for WWW.VICTIM.COM. If Mallory's response gets to your computer before the legitimate response arrives from your ISP's name server, you will be redirected where Mallory tells you you're going.

Obvious fix: you want the QID be randomly generated. Now Alice and Mallory are in a race. Alice sees Bob's request and knows the QID. Mallory has to guess it. The first one to land a packet with the correct QID wins. Randomized QIDs give Alice a big advantage in this race.

But there's a bunch more problems here:

*

If you convince Bob to ask Alice the same question 1000 times all at once, and Bob uses a different QID for each packet, you made the race 1000 times easier for Mallory to win.
*

If Bob uses a crappy random number generator, Mallory can get Bob to ask for names she controls, like WWW.EVIL.COM, and watch how the QIDs bounce around; eventually, she'll break the RNG and be able to predict its outputs.
*

16 bits just isn't big enough to provide real security at the traffic rates we deal with in 2008.

Your computer's resolver is probably a stub. Which means it won't really save the response. You don't want it to. The stub asks a real DNS server, probably run by your ISP. That server doesn't know everything. It can't, and shouldn't, because the whole idea of DNS is to compensate for the organic and shifting nature of internet naming and addressing. Frequently, that server has to go ask another, and so on. The cool kids call this "recursion".

Responses carry another value, too, called a time to live (TTL). This number tells your name server how long to cache the answer. Why? Because they deal with zillions of queries. Whoever wins the race between Alice and Mallory, their answer gets cached. All subsequent responses will be dropped. All future requests for that same data, within the TTL, come from that answer. This is good for whoever wins the race. If Alice wins, it means Mallory can't poison the cache for that name. If Mallory wins, the next 10,000 or so people that ask that cache where WWW.VICTIM.COM is go to 6.6.6.0.
3.

Then there's that other set of DNS vulnerabilities. These require you to pay attention in class. They haven't really been talked about since 1997. And they're hard to find, because you have to understand how DNS works. In other words, you have to be completely crazy. Lazlo Hollyfeld crazy. I'm speaking of course of RRset poisoning.

DNS has a complicated architecture. Not only that, but not all name servers run the same code. So not all of them implement DNS in exactly the same way. And not only that, but not all name servers are configured properly.

I just described a QID attack that poisons the name server's cache. This attack requires speed, agility and luck, because if the "real" answer happens to arrive before your spoofed one, you're locked out. Fortunately for those of you that have a time machine, some versions of DNS provide you with another way to poison the name server's cache anyway. To explain it, I will have to explain more about the format of a DNS packet.

DNS packets are variable in length and consist of a header, some flags and resource records (RRs). RRs are where the goods ride around. There are up to three sets of RRs in a DNS packet, along with the original query. These are:

*

Answer RR's, which contain the answer to whatever question you asked (such as the A record that says WWW.VICTIM.COM is 1.2.3.4)
*

Authority RR's, which tell resolvers which name servers to refer to to get the complete answer for a question
*

Additional RR's, sometimes called "glue", which contain any additional information needed to make the response effective.

A word about the Additional RR's. Think about an NS record, like the one that COM's name server uses to tell us that, to find out where WWW.VICTIM.COM is, you have to ask NS1.VICTIM.COM. That's good to know, but it's not going to help you unless you know where to find NS1.VICTIM.COM. Names are not addresses. This is a chicken and egg problem. The answer is, you provide both the NS record pointing VICTIM.COM to NS1.VICTIM.COM, and the A record pointing NS1.VICTIM.COM to 1.2.3.1.

Now, let's party like it's 1995.

Download the source code for a DNS implementation and hack it up such that every time it sends out a response, it also sends out a little bit of evil — an extra Additional RR with bad information. Then let's set up an evil server with it, and register it as EVIL.COM. Now get a bunch of web pages up with IMG tags pointing to names hosted at that server.

Bob innocently loads up a page with the malicious tags which coerces his browser resolve that name. Bob asks Alice to resolve that name. Here comes recursion: eventually the query arrives at our evil server. Which sends back a response with an unexpected (evil) Additional RR.

If Alice's cache honors the unexpected record, it's 1995 —- buy CSCO! —- and you just poisoned their cache. Worse, it will replace the "real" data already in the cache with the fake data. You asked where WWW.EVIL.COM was (or rather, the image tags did). But Alice also "found out" where WWW.VICTIM.COM was: 6.6.6.0. Every resolver that points to that name server will now gladly forward you to the website of the beast.
4.

It's not 1995. It's 2008. There are fixes for the attacks I have described.
Fix 1:

The QID race is fixed with random IDs, and by using a strong random number generator and being careful with the state you keep for queries. 16 bit query IDs are still too short, which fills us with dread. There are hacks to get around this. For instance, DJBDNS randomizes the source port on requests as well, and thus won't honor responses unless they come from someone who guesses the ~16 bit source port. This brings us close to 32 bits, which is much harder to guess.
Fix 2:

The RR set poisoning attack is fixed by bailiwick checking, which is a quirky way of saying that resolvers simply remember that if they're asking where WWW.VICTIM.COM is, they're not interested in caching a new address for WWW.GOOGLE.COM in the same transaction.

Remember how these fixes work. They're very important.

And so we arrive at the present day.
5.

Let's try again to convince Bob that WWW.VICTIM.COM is 6.6.6.0.

This time though, instead of getting Bob to look up WWW.VICTIM.COM and then beating Alice in the race, or getting Bob to look up WWW.EVIL.COM and slipping strychnine into his ham sandwich, we're going to be clever (sneaky).

Get Bob to look up AAAAA.VICTIM.COM. Race Alice. Alice's answer is NXDOMAIN, because there's no such name as AAAAA.VICTIM.COM. Mallory has an answer. We'll come back to it. Alice has an advantage in the race, and so she likely beats Mallory. NXDOMAIN for AAAAA.VICTIM.COM.

Alice's advantage is not insurmountable. Mallory repeats with AAAAB.VICTIM.COM. Then AAAAC.VICTIM.COM. And so on. Sometime, perhaps around CXOPQ.VICTIM.COM, Mallory wins! Bob believes CXOPQ.VICTIM.COM is 6.6.6.0!

Poisoning CXOPQ.VICTIM.COM is not super valuable to Mallory. But Mallory has another trick up her sleeve. Because her response didn't just say CXOPQ.VICTIM.COM was 6.6.6.0. It also contained Additional RRs pointing WWW.VICTIM.COM to 6.6.6.0. Those records are in-bailiwick: Bob is in fact interested in VICTIM.COM for this query. Mallory has combined attack #1 with attack #2, defeating fix #1 and fix #2. Mallory can conduct this attack in less than 10 seconds on a fast Internet link.