IMPROVED DNS SPOOFING USING NODE RE-DELEGATION

Bernhard Müller
SEC Consult Vulnerability Lab, Vienna, 07/14/2008

Abstract

Due to the nature of UDP and the DNS protocol, it is always possible to inject poison RRs into the cache of a caching DNS server, given that the attacker is able to reach the target server with DNS response packets before the real authoritative name server does (and the target server does not use additional security features, such as DNSSEC transaction signatures [1]). However, classical DNS spoofing attacks may not be feasible in many cases due to the long time (hours to days) that valid RRs stay in the target server's cache. This paper demonstrates a method that eliminates the TTL variable and therefore greatly reduces the difficulty of a successful attack.

Introduction

DNS spoofing and cache poisoning attacks have been around for many years (e.g. 2]). Since then, many variants of these attacks have been described by various researchers, with much of the work focusing on transaction ID prediction [3][4]. The basics of DNS spoofing attacks are well known and will not be discussed in this whitepaper.

An Internet-Draft by Hubert and van Mook [5] discusses the time and bandwidth requirements needed for successful DNS spoofing attacks. In this draft, the following equation is used:

$$P_{cs} = 1 - \left(1 - \frac{D * R * W}{N * P * I}\right)^{(T / TTL)}$$

I: Number distinct IDs available (maximum 65536)
P: Number of ports used (maximum around 64000, but often 1)
N: Number of authoritative nameservers for a domain
R: Number of packets sent per second by the attacker
W: Window of opportunity, in seconds. Bounded by the response time of the authoritative servers (often 0.1s)
D: Average number of identical outstanding questions of a resolver (typically 1)
T: Time period in which the attack occurs
TTL: Time to live of the legitimate resource record

Using this formula, the authors come to the conclusion that for example, "(for) a domain with a 3600 second TTL, an attacker sending 7000 fake answer packets/s (a rate of 4.5Mb/s), stands a 10% chance of spoofing a record in the first 24 hours, which rises to 50% after a week." In this calculation, it is assumed that the destination port for the spoofed packets is known.

If we take into account that the TTLs for most domains are significantly higher, we can conclude that a typical cache poisoning attack will, in most cases, only be successful if it is conducted for several weeks or months.

Time requirements of a classical cache poisoning attack

We will use an example to demonstrate our attack. At first, we will calculate the time and bandwidth requirements needed for a classical cache poisoning attack, in which the attacker would try to directly poison the A record for the target node.

We are attacking a caching server named `ns1.target.com`, and want to insert a poison cache entry for `www.example.com`. First, we look up the authoritative name servers for the zone `example.com`:

```
h4l@b4byl0n:~$ dig -t ns example.com

;; QUESTION SECTION:
;example.com.          IN    NS

;; ANSWER SECTION:
example.com.     172800     IN    NS    dns1.example.com.
example.com.     172800     IN    NS    dns2.example.com.

;; ADDITIONAL SECTION:
dns2.example.com.     172800     IN    A     192.168.100.2
dns1.example.com.     172800     IN    A     192.168.100.1
```

As we can see, 2 authoritative name servers for example.com exist. For our calculation, we will also need the TTL given for the address record www.example.com:

```
;; ANSWER SECTION:
www.example.com. 86400 IN    CNAME  example.com.
example.com.     86400 IN    A      192.168.100.99
```

As we see here, the TTL for the `www.example.com` RR is 86400 seconds (24 hours). This means that the window of opportunity for the attack only opens once a day.

If we assume that the attacker has a bandwidth of 10 Mb/s, and a single DNS response packet consists of 80 byte, the attacker will (at least theoretically) be able to send about ~15000 spoofed response packets per second. The window of opportunity is assumed to be 100 milliseconds. Our formula now looks as follows:

$$P_{cs} = 1 - \left(1 - \frac{1 * 15000 * 0.1}{2 * 1 * 65535}\right)^{\left(\frac{T}{86400}\right)}$$

We can now calculate the following probabilities of a successful attack for different timeframes:

- 1 day: 1.1 %
- 7 days: 7.7 %
- 30 days: 29.1 %
- 6 month (180 days): 87.3 %
- 1 year (365 days): 98.5 %

We can conclude that for a reasonable success rate, the attack must be conducted for at least a few months. In most cases, this would not be feasible.

Using node re-delegation to eliminate the TTL problem

For a more effective TXID guessing attack, we will need to generate queries that do not elicit a valid, cacheable response record from the authoritative name server. We can accomplish this by avoiding direct queries for our target node, but rather requesting addresses for non-existent nodes that seemingly belong to a zone deeper down the DNS hierarchy. We can then try to spoof a response that delegates the zone to a name server of our choice.

This attack is based on the fact that resolvers will cache any delegations that are closer to the requested node. From RFC 1034 [5]:

"If the response shows a delegation, the resolver should check to see that the delegation is "closer" to the answer than the servers in SLIST are. This can be done by comparing the match count in SLIST with that computed from SNAME and the NS RRs in the delegation. If not, the reply is bogus and should be ignored. If the delegation is valid the NS delegation RRs and any address RRs for the servers should be cached. The name servers are entered in the SLIST, and the search is restarted."

For this attack, we send queries for randomly generated node names, in the format (`[rand].www.example.com`).

- Query 1: `IN A ASJSDHASASSMXOEUWIUEUXMID.www.example.com`
- Query 2: `IN A HFUEFNDKHUEHJDHFKJAFDHKJD.www.example.com`
- ...

For each query, send the maximum possible amount of responses within the window of opportunity. These responses contain an NS record for `www.example.com`.

- Spoofed rsp.: `www.example.com IN NS ns1.attacker.com`

As soon as one of our responses hits the correct transaction ID, `ns1.attacker.com` will be cached as the authoritative name server for `www.example.com`. Note that the actual RR for `www.example.com` will never be cached at the target server, which means that we do not have to wait for its TTL to expire.

To calculate our success rate, we can now use a modified version of the probability function:

$$P_{cs} = 1 - \left(1 - \frac{1 * 15000 * 0.1}{2 * 1 * 65535}\right)^{T}$$

The new probability values are:

- 1 minute: 49.7 %
- 2 minutes: 74.3 %
- 5 minutes: 96.7%
- 10 minutes: 99.9 %

The results show that the attack will, in the most cases, succeed within the first 5 minutes.

Defending against the node re-delegation attack

The attack described in this paper is largely based on the fact that DNS transaction IDs are easily guessable (with a chance of 1/65535). By introducing source port randomization, the problem space for a correct guess expands by a factor of 64000, which effectively defeats all blind DNS spoofing attacks (i.e., DNS spoofing without prior packet sniffing).

Recently, security updates that implement source port randomization have been released by all important DNS vendors [7] [8] . SEC Consult highly recommends an upgrade of vulnerable DNS servers.

REFERENCES

1. **Gudmundsson , Lewis , Nordmark, Wellington.** RFC 2931: DNS Request and Transaction Signatures (SIG(0)s). [Online] 2000. http://www.ietf.org/rfc/rfc2931.txt
2. **Schuba , Christophj,** ADDRESSING WEAKNESSES IN THE DOMAIN NAME SYSTEM PROTOCOL (M.Sc. Thesis). 1993. http://ftp.cerias.purdue.edu/pub/papers/christoph-schuba/schuba-DNS-msthesis.pdf
3. **LURHQ Threat Intelligence Group**. DNS Cache Poisoning - The Next Generation. [Online] 2003. http://www.lurhq.com/cachepoisoning.html
4. **Klein , Amit** . BIND 9 DNS Cache Poisoning. [Online] 2007. http://www.trusteer.com/bind9dns

5. **Hubert, van Mook.** Measures to prevent DNS spoofing (Internet-Draft) [Online] 2006 http://tools.ietf.org/html/draft-hubert-dns-anti-spoofing-00

6. **Mockapetris**. DOMAIN NAMES - CONCEPTS AND FACILITIES (RFC 1034). (Online) 1987. http://www.ietf.org/rfc/rfc1034.txt

7. BIND Vulnerabilities: CERT VU#800113 DNS Cache Poisoning Issue http://www.isc.org/index.pl?/sw/bind/index.php

8. Microsoft Security Bulletin MS08-037, Vulnerabilities in DNS Could Allow Spoofing (953230) http://www.microsoft.com/technet/security/Bulletin/ms08-037.mspx

About the Vulnerability Lab

Members of the SEC Consult Vulnerability Lab perform security research in various topics of technical information security. Projects include vulnerability research and the development of cutting edge security tools and methodologies, and are supported by partners like the Technical University of Vienna. The lab has published security vulnerabilities in many high-profile software products, and selected work has been presented at top security conferences like Blackhat and DeepSec.

For more information, see http://www.sec-consult.com/