Predictability of Windows DNS resolver

ing. Roberto Larcher - http://webteca.altervista.org - robertolarcher@hotmail.com

rev. 1 - March 11, 2004

Abstract

The main DNS security issues have very often focused on server side[1] problems and vulnerabilities. This paper focuses on Windows client DNS service, also called DNS resolver.
This paper explains how it is often possible to predict the "Transaction ID" and the "UDP port number" used by Windows' DNS Resolver. With this information it will be shown how it is possible, under certain conditions, to win the race against the regular DNS server and hijack, for example, a TCP/IP session.
Even if this problem has been reported to Microsoft's security experts and we both agreed that there is no immediate threat or security vulnerability, it may be used to attack Windows LAN and WAN clients for example at startup. In WLAN too, which shares the medium and then is subjected to the well-known DNS attacks based on sniffing, this predictability increases the chances of being effectively attacked.
Microsoft informed me that the concerns mentioned in this paper will be addressed in future versions of its products.

The problem

The problem is that the "Transaction ID" and the "UDP port number" used in client DNS queries are predictable.
According to [2] RFC 1035 "Domain Names - Implementation And Specification", "Transaction ID" is: "a 16 bit identifier assigned by the program that generates any kind of query. This identifier is copied in the corresponding reply and can be used by the requester to match up replies to outstanding queries".
Since we are going to send fake replies to regular queries we also need to know the UPD port used by the client to send datagrams. According to [3] RFC 768 "User Datagram Protocol"," UDP source port" "indicates the port of the sending process".

To test it you can simply flush the DNS client cache –if enabled– and do some pings to whichever internet site you want. With a network analyzer[2], for instance Politecnico di Torino's Analyzer, it is easy to intercept the packets needed for this brief analysis.

Different OS versions behave differently:

WinXP sp1 with DNS client service enabled:

```
1   09:28:45.535209   IP: 10.36.0.35 => 10.36.0.14 (63) UDP: Length= 43, Port (3453 => 53) DNS: Query: Name to Address
2   09:28:46.755663   IP: 10.36.0.35 => 10.36.0.14 (62) UDP: Length= 42, Port (3453 => 53) DNS: Query: Name to Address
3   09:28:53.313006   IP: 10.36.0.35 => 10.36.0.14 (60) UDP: Length= 40, Port (3453 => 53) DNS: Query: Name to Address
4   09:28:56.514783   IP: 10.36.0.35 => 10.36.0.14 (59) UDP: Length= 39, Port (3453 => 53) DNS: Query: Name to Address
5   09:29:00.673814   IP: 10.36.0.35 => 10.36.0.14 (63) UDP: Length= 43, Port (3453 => 53) DNS: Query: Name to Address
```

As far as my tests are concerned, it seems that all the queries are sent with the same UDP port number (in bold in this example).
Moreover, when the Client DNS service is (re)started, the transaction ID (not shown in the example) is set to 1 and it is then incremented on each query.

[1] For example: [1] "Security Issue with DNS" Florent Carli – SANS Institute, 2003

[2] The analyzer has to run on the same client if you are in a switched network and you cannot monitor it. If you try to test your DNS server with the nslookup command-line tool, you may get a different behavior. It seems that nslookup does not use the DNS resolver (it increments both the "Transaction ID" and the "UDP port number"). This is comprehensible: if it used the DSN revolver, it would get the information from its cache and not from the server.

WinXP sp1 with DNS client service disabled:

```
1   09:00:59.350310  IP:  10.36.0.35 => 10.36.0.14  (63) UDP:  Length= 43,  Port (3440 => 53)  DNS:  Query:  Name to Address
2   09:01:00.788318  IP:  10.36.0.35 => 10.36.0.14  (62) UDP:  Length= 42,  Port (3441 => 53)  DNS:  Query:  Name to Address
3   09:01:07.143486  IP:  10.36.0.35 => 10.36.0.14  (60) UDP:  Length= 40,  Port (3442 => 53)  DNS:  Query:  Name to Address
4   09:01:10.448216  IP:  10.36.0.35 => 10.36.0.14  (59) UDP:  Length= 39,  Port (3443 => 53)  DNS:  Query:  Name to Address
5   09:01:14.744673  IP:  10.36.0.35 => 10.36.0.14  (63) UDP:  Length= 43,  Port (3444 => 53)  DNS:  Query:  Name to Address
```

In this case the UDP port is incremented in each query and the transaction ID is always set to 1.

WinNT sp5, WinNT sp6 and WinNT Server sp6:

```
1   07:38:58.881531  IP:  10.36.0.31 => 10.36.0.14  (63) UDP:  Length= 43,  Port (1031 => 53)  DNS:  Query:  Name to Address
2   07:38:59.044741  IP:  10.36.0.31 => 10.36.0.14  (62) UDP:  Length= 42,  Port (1032 => 53)  DNS:  Query:  Name to Address
3   07:39:00.346721  IP:  10.36.0.31 => 10.36.0.14  (60) UDP:  Length= 40,  Port (1033 => 53)  DNS:  Query:  Name to Address
4   07:39:00.651171  IP:  10.36.0.31 => 10.36.0.14  (59) UDP:  Length= 39,  Port (1034 => 53)  DNS:  Query:  Name to Address
5   07:39:00.813969  IP:  10.36.0.31 => 10.36.0.14  (63) UDP:  Length= 43,  Port (1035 => 53)  DNS:  Query:  Name to Address
```

In this case the UDP port is incremented in each query and the transaction ID is always set to 1. Please note that WinNT has no client DNS cache.

Win2000 sp3 with DNS client service enabled:

```
1   10:03:46.152742  IP:  10.36.0.50 => 10.36.0.14  (63) UDP:  Length= 43,  Port (2579 => 53)  DNS:  Query:  Name to Address
2   10:03:46.456406  IP:  10.36.0.50 => 10.36.0.14  (62) UDP:  Length= 42,  Port (2580 => 53)  DNS:  Query:  Name to Address
3   10:03:48.060385  IP:  10.36.0.50 => 10.36.0.14  (60) UDP:  Length= 40,  Port (2581 => 53)  DNS:  Query:  Name to Address
4   10:03:48.249590  IP:  10.36.0.50 => 10.36.0.14  (59) UDP:  Length= 39,  Port (2582 => 53)  DNS:  Query:  Name to Address
5   10:03:48.425597  IP:  10.36.0.50 => 10.36.0.14  (63) UDP:  Length= 43,  Port (2583 => 53)  DNS:  Query:  Name to Address
```

In this case the UDP port is incremented in each query and the transaction ID is always set to 1.

Win2000 sp4 with DNS client service enabled:

```
1   07:31:27.221489  IP:  10.36.0.15 => 10.36.0.14  (63) UDP:  Length= 43,  Port (1041 => 53)  DNS:  Query:  Name to Address
2   07:31:27.448766  IP:  10.36.0.15 => 10.36.0.14  (62) UDP:  Length= 42,  Port (1042 => 53)  DNS:  Query:  Name to Address
3   07:31:28.608191  IP:  10.36.0.15 => 10.36.0.14  (60) UDP:  Length= 40,  Port (1043 => 53)  DNS:  Query:  Name to Address
4   07:31:28.784089  IP:  10.36.0.15 => 10.36.0.14  (59) UDP:  Length= 39,  Port (1044 => 53)  DNS:  Query:  Name to Address
5   07:31:28.917098  IP:  10.36.0.15 => 10.36.0.14  (63) UDP:  Length= 43,  Port (1045 => 53)  DNS:  Query:  Name to Address
```

In this case the UDP port is incremented in each query and the transaction ID seems to be random.

Win2000 sp4 with DNS client service disabled:

```
1   10:14:33.811720  IP:  10.36.0.15 => 10.36.0.14  (63) UDP:  Length= 43,  Port (1057 => 53)  DNS:  Query:  Name to Address
2   10:14:34.121979  IP:  10.36.0.15 => 10.36.0.14  (62) UDP:  Length= 42,  Port (1058 => 53)  DNS:  Query:  Name to Address
3   10:14:35.255329  IP:  10.36.0.15 => 10.36.0.14  (60) UDP:  Length= 40,  Port (1059 => 53)  DNS:  Query:  Name to Address
4   10:14:35.394092  IP:  10.36.0.15 => 10.36.0.14  (59) UDP:  Length= 39,  Port (1060 => 53)  DNS:  Query:  Name to Address
5   10:14:35.529493  IP:  10.36.0.15 => 10.36.0.14  (63) UDP:  Length= 43,  Port (1061 => 53)  DNS:  Query:  Name to Address
```

In this case the UDP port is incremented in each query and the transaction ID seems to be random.

Real-life network monitoring sessions tracking all the DNS client requests [alternative: every/each DNS client request] were made and, when DNS cache is enabled, both transaction ID and UDP port number are low-ranged.

More on WinXP SP1

More extended tests were made on WinXP SP1 (the latest Windows "client" version with the latest Service Pack at the moment of writing) and it seems very easy to guess the UDP port number assigned to the DNS resolver. It seems actually to be the first free UDP port at the moment of the first DNS request.
Moreover, in the standard configuration, i.e. with Internet Time Server enabled and set to time.windows.com, at startup before the logon WinXP tries to synchronize and makes a DNS request. So the DNS resolver always gets a low UDP port number – in my tests 1027 is the more frequent, followed by 1028 or 1029.

The sample application

In order to make real-life tests, an application has been developed. This application, dnspoison[3], tries to corrupt a client DNS cache by flooding it with fake DNS replies.

The application takes this set of parameters:

dnspoison [-i interface]
[-t target]
[-s real_DNS_ip]
[-f fake_ip]
[-p port_number]
[-r id_range]
[-q question_name]

By flooding a client with a forged DNS response on UDP port 1027, a specific FQDN[4] and transaction ID cyclically ranging from 0 to 20, the DNS cache has been corrupted:

```
www.test.net
----------------------------------------
Record name . . . . . : www.test.net
Record  type. . . . . : 1
Time To Live. . . . . : 3588
Data Length . . . . . : 4
Section . . . . . . . : Answer
Record A (Host) . . . : 192.168.0.11
```

As you can see, a Internet FQDN has been mapped to a local LAN address. Obviously a lot of attacks can be done once the client has been fooled about the IP[5]. For instance I hijacked the internet browser traffic to a fake proxy without any problem. See later for a complete example.

[3] You can download it here http://webteca.altervista.org/dnspoison.htm
[4] FQDN: A *fully qualified domain name* consists of a host and domain name, including top-level domain. For example, webteca.altervista.org is a fully qualified domain name; webteca is the host, altervista is the second-level domain, and .org is the top level domain. (http://www.webopedia.com/TERM/F/FQDN.html)
[5] A compete set of attacks can be found in : [5] "DNS ID prediction and exploitation", Gamma - August 1998

How this problem can be used: some topology scenarios

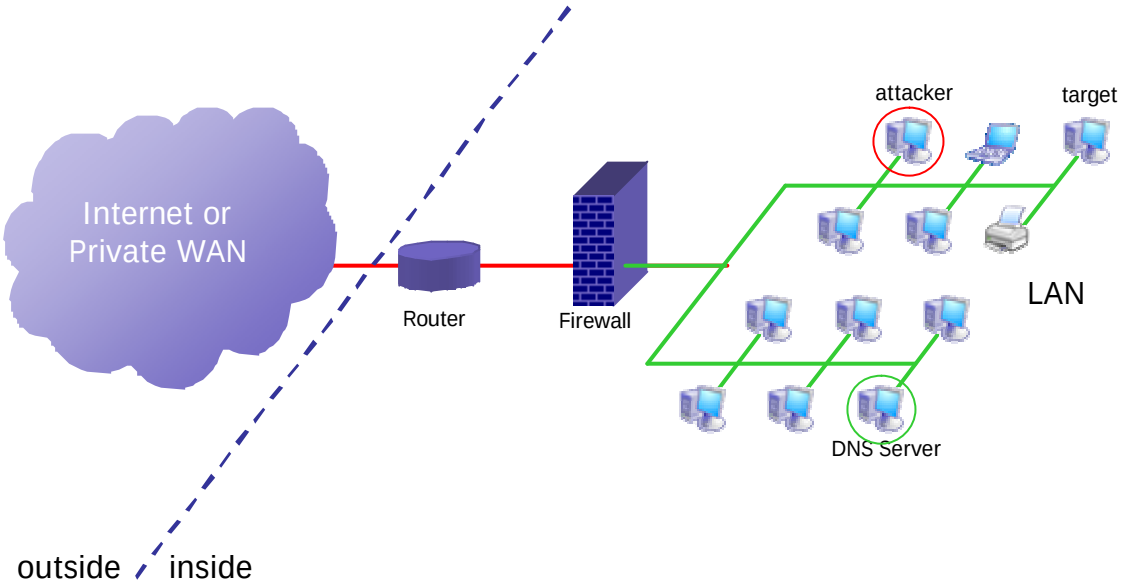
This flaw can be exploited to try to hijack TCP/IP sessions. Any time a client requests an IP address to a DNS server you can use this predictability to increase your chances to win the race against the server itself. Suppose that an attacker wants to hijack the TCP/IP session to a specific FQDN. Different scenarios are vulnerable[6] in different ways.

Switched Ethernet

In this case an attacker cannot see any packet form the net[7], so he have to:

- guess the "Transaction ID",
- guess the "UDP port number",
- be faster than the real DNS.

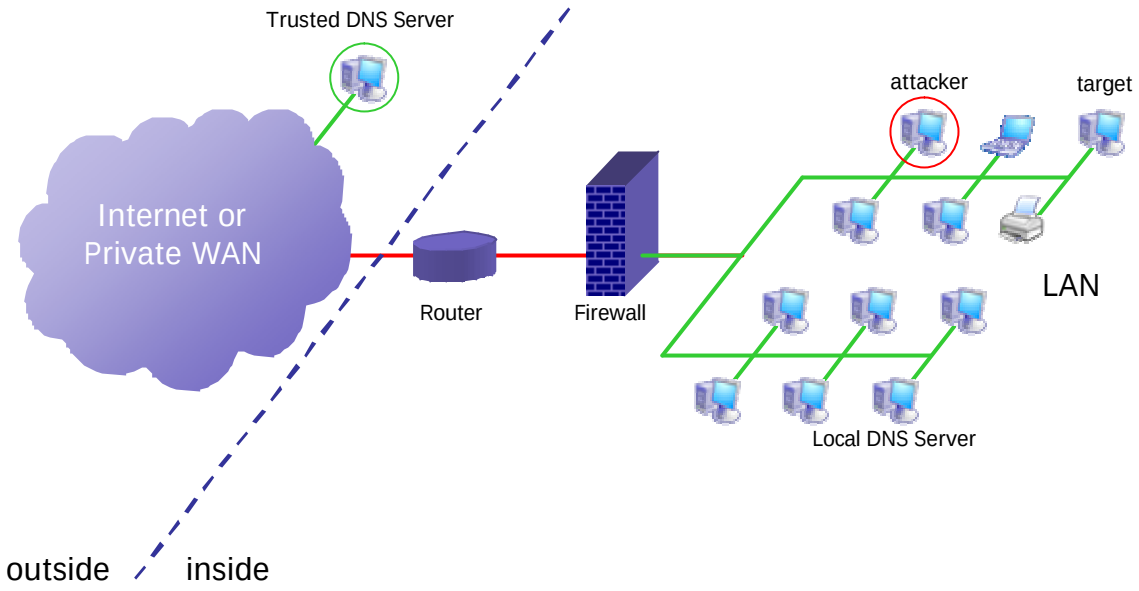
The first two points are covered by the reported predictability. The third is deeply dependant on the layout of the net.
Consider this picture:



In this case, the DNS server, the target and the attacker are on the same net. So the attacker has to be quicker than the DNS Server. If we consider an application like dnspoison that cycles the transaction ID, for example, from 0 to 200, the chances to win the race are few. Other techniques, such as DNS Server flooding to try to slow it down, may help.

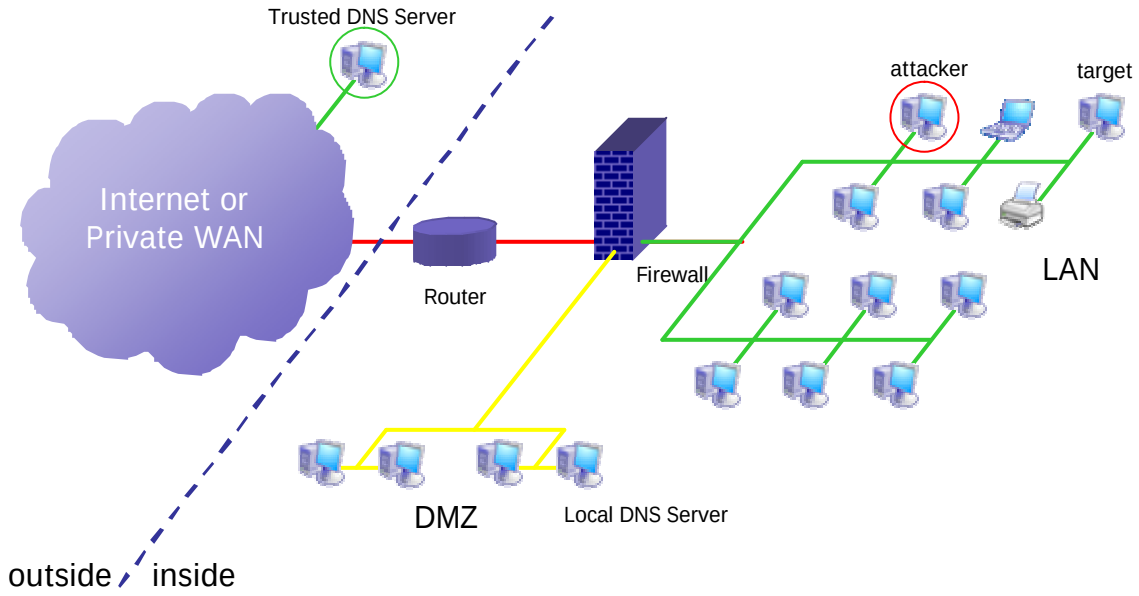
[6] Strictly speaking is not any mayor vulnerability, but it is possible that the predictability reported in this paper could issue security concerns for some users, such as corporate ones.
[7] Of course he can be on a monitoring port, or use ARP poisoning or port stealing, but in this cases, why should he use other techniques?

But what if the requested FQDN is not in the local DNS Server cache? Often the DNS is configured to ask some other trusted DNS servers located in Internet or in a private network.



In such cases a hop (or even more) is needed to retrieve the required data to the client. So the attacker has much more time to deliver its attack, and consequently, its chances of success will be higher.

It may also happen that, for whatever reason, the DNS is located in the DMZ:

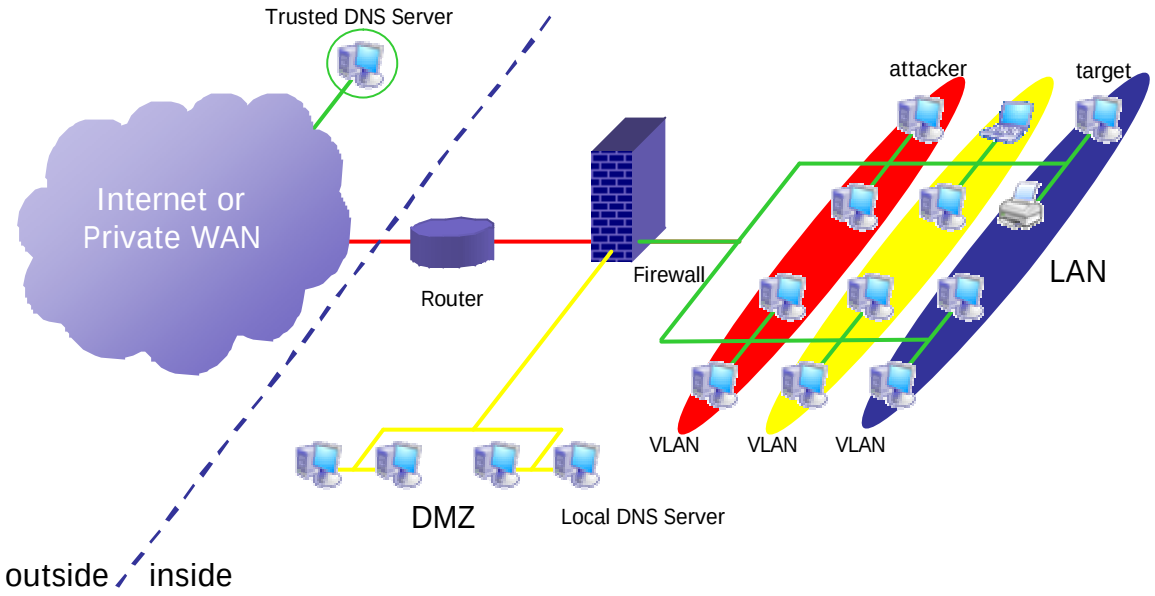


In this case the firewall introduces a delay in the packet dispatching, which, again, delays the DNS response.

Switched Ethernet – cross VLANs cache corruption

In this case we suppose that the attacker and the target are on different VLANs. In this scenario, the VLANs can of course see each other via a routing device (for example a router or a multilayer switch – not shown in the picture). The attacker cannot see any packet form the target, either using techniques like ARP spoofing or using port stealing. As in the above scenario, he needs to:

- guess the "Transaction ID",
- guess the "UDP port number",
- be faster than the real DNS.



According to my test, a client's DNS cache can be effectively corrupted over VLANs. In this case, the predictability of the "Transaction ID" and the "UDP port number" opens/causes more severe problems to the security of the net.

In fact, it can be regarded as a "less efficient arp-spoofing-like technique over VLANs".

Let's examine this definition:

- "Less efficient", since you still have to guess the right "Transaction ID" and the "UDP port Number",
- "arp-spoofing-like", since the final effect is quite similar to arp-spoofing: here you fool a LAN client into believing that a FQDN has a fake IP address, there you fool a LAN client into believing that a particular IP address has a fake MAC address,
- "over VLANs" since the attacker and the target are in two different VLANs.

Shared environments

Obviously in shared environments, such as Ethernet on hubs, Token ring or 802.11 wireless LANs, it is easier to hijack TCP/IP sessions. In this case you can simply monitor the medium in order to get a DNS request and once you have found a suitable request, you simply forge the right answer[8]. You do not need to guess anything: you need only to win the race with the regular server.

But if you have a DNS request packet, you will know the client UDP port number, and since the "Transaction ID" is predictable, you will know all the parameters for the *next* request and the next reply too. In this case, you can flood the client with only the *right next* answer, and your chances to win the race increase a lot[9].

How this problem can be used: some corporate scenarios

Corporate scenarios – Browser Settings

The typical corporate PC –especially in large companies– comes with fixed configuration. This helps administrators to manage huge numbers of computers. But it also may help an attacker to set up some attacks. Think, for example, about a corporate PC with WinXP sp1 that:

- has fixed, armored configuration,
- opens the corporate Intranet site at startup,
- uses a FQDN to specify the proxy to connect to Internet.

In this case it is quite easy for an internal attacker to get the information needed to fool the browser about the proxy IP address and hijack all the Internet traffic. He actually has:

- the FQDN to impersonate,
- the UPD port number, low-numbered since Internet Time is enabled by default,
- the transaction ID, low-numbered since the corporate Intranet opens at startup

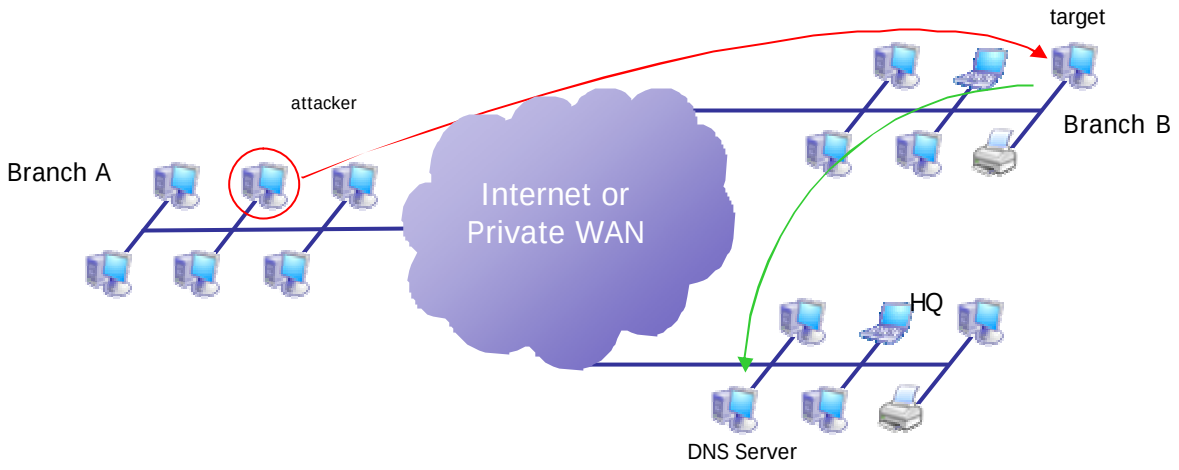
I tried to accomplish this –with the DNS one hop far from the target– and I got the client fooled.

[8] This is a well-known attack. Again, for example: [5] "DNS ID prediction and exploitation"

[9] For example consider the second or the third switched Ethernet pictures and apply them to shared Ethernet.

Corporate scenarios – WAN

Large companies may have branches that are connected through WAN (or VPNs). In this case it is quite frequent that the DNS is located in the corporate main site or HQ. A branch's PC can be more than a hop far from the server. So an attacker can flood a target locally and even remotely.



Conclusions

In this paper it has been shown that some versions of Windows DNS resolver (client side) present some problems about predictability of the "Transaction ID" and the "UDP port Number". Even if I do not think that this kind of problems is a vulnerability and there is no immediate threat, there are (however) some scenarios in which some security issues can arise. Tests were made in order to check if these problems can lead to client side cache corruption in real life situations. This can be done under certain conditions.

Acknowledgements

I am greatly thankful to Marco Favaretto who revised this paper. Thanks also to my sister Cinzia for additional corrections.
Of course any mistake in this paper is only my responsibility.

References

[1] Security Issue with DNS
Florent Carli – SANS Institute, 2003

http://www.sans.org/rr/papers/17/1069.pdf

[2] RFC 1035: "Domain Names - Implementation And Specification"
P. Mockapetris – ISI, November 1987

http://www.faqs.org/rfcs/rfc1035.html

[3] RFC 768: "User Datagram Protocol"
J. Postel- ISI, 28 August 1980

http://www.faqs.org/rfcs/rfc768.html

[4] "DNS ID prediction and exploitation
Gamma, August 1998

http://c0vertl.tripod.com/text/dnsattack.txt

[5] DNS Cache Poisoning - The Next Generation
Joe Stewart, Jan 27 2003

http://www.securityfocus.com/guest/17905

[6] Strange Attractors and TCP/IP Sequence Number Analysis
Michal Zalewski, 2001

http://razor.bindview.com/publish/papers/tcpseq.html

[7] The dsniff suite
Dug Song

http://naughty.monkey.org/~dugsong/dsniff/faq.html