Domain Name Chaining Attacks

Master Thesis
Author: Johannes Barnickel
Supervisor: Prof. Dr. Jörg Schwenk
Lehrstuhl für Netz- und Datensicherheit
Ruhr-Universität Bochum

October 22, 2008

Abstract

This document is an overview over DNS protocol basics, its history, past and current threats on DNS infrastructure, and means to prevent or hinder known and unknown attacks. It reviews old and new measures to mitigate various attacks on DNS. The name chaining attack model is discussed regarding current DNS security measures.

Contents

Chapter 1

Introduction

1.1 How DNS Works

DNS facilitates translation of human-readable names to IP route able addresses. Queries are of the kind "which IP corresponds to www.example.com?" They are sent from the resolver to the recursor. Typically, the recursor is a central server on the network or in the upstream network, i.e. at the Internet service provider. Almost all recursors use caching techniques to save bandwidth.

When a recursor does not have an authoritative entry for a name (i.e. it does not own this name) and no cached entry for this name, it will itself become a resolver and issue a query to another recursor - typically, less one subdomain, e.g. example.com instead of www.example.com. The name server of example.com is authorized to provide the IP address of www.example.com. So, our recursor needs to find the name server of example.com. This is done by asking the authoritative .com name server, which our recursor will probably know, or ask the name server authoritative for .com, which is root. The IP addresses of the root servers are pre-configured in all DNS recursor implementations and change very rarely.

A nonce is used to identify and authenticate answers. All queries are accompanied by a 16 bit transaction ID that will also be included in the answer.

DNS servers listen on port 53 UDP and TCP and will send the reply packets to the any source port that was used by the client to send the packet.

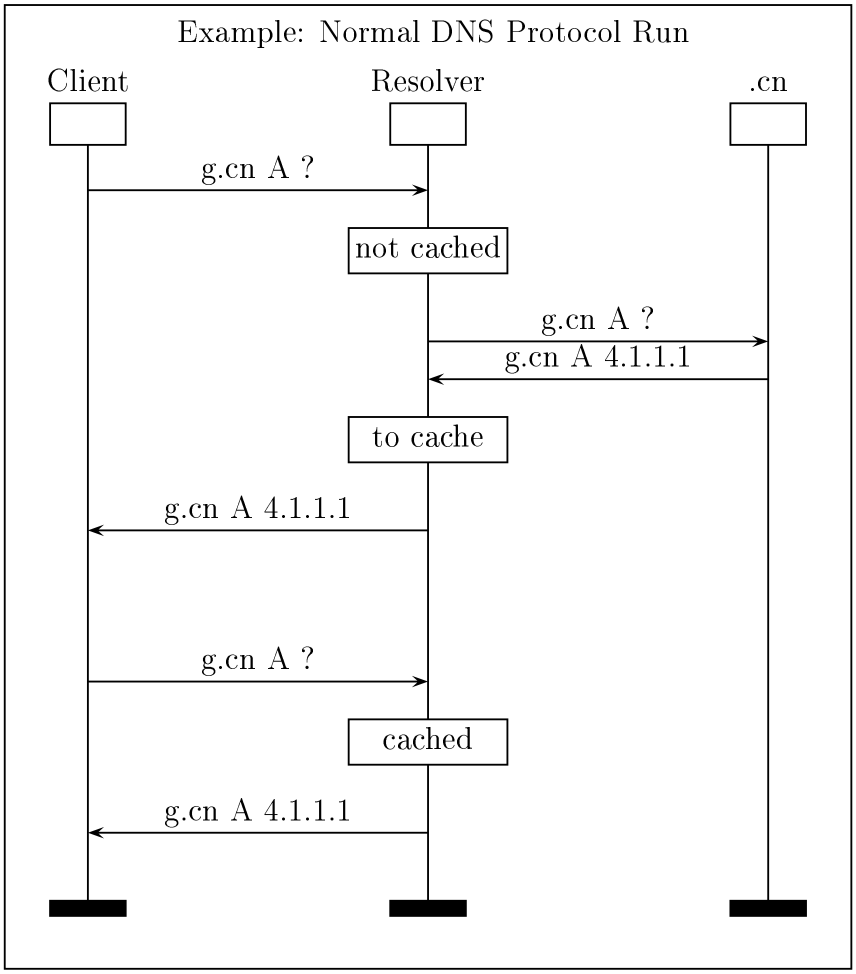
1.2 Terminology

Name servers that actively try to resolve names instead of only checking for queried names in their caches are called resolvers. When a name server recurses a query to another server, the first server is called recursor and the second server is called upstream server or upstream resolver, with the root servers at the top.

Some DNS protocol messages and their meaning:

message	meaning
g.cn A ?	what is the IP address of g.cn?
g.cn A 4.1.1.1	the IP address of g.cn is 4.1.1.1
g.cn NS ?	what is the name server of g.cn?
g.cn NS 4.1.1.1	the name server of g.cn is 4.1.1.1
g.cn NS ns.g.cn, ns.g.cn A 4.1.1.1	the name server of g.cn is ns.g.cn, and the IP address of ns.g.cn is 4.1.1.1

In the following examples, an attacker will often try to attack a domain called g.cn. This is a valid domain (Google China) and was chosen because of space considerations, especially in diagrams. The IP address 6.6.6.6 is often used for the attacker himself, his DNS server or the target in forged DNS packets which will be accepted by the server. The authoritative name server for .cn will simply be called .cn and is often shown as an upstream resolver in diagrams. In the diagrams, clients are end users who want to resolve names. Transaction IDs of queries and replies are omitted in diagrams when they are not attacked.



1.3 DNS Hierarchy

Although usually left out of display, every qualified domain name ends with a dot, e.g. "www.example.com." is the same as "www.example.com", because DNS software will automatically append the final dot. Authority zones are separated by dots, with the top being on the right hand side.

zone	note
.	DNS root
.com.	zone for commercial entities, free for registration
example.com.	registered by someone within the .com zone
www.example.com.	a server within the private owned domain

DNS generates a tree structure with the final dot being root and the individual machines being leafs.

1.4 Resolution Path

DNS is a distributed database, which is queried recursively. A typical query by a client will be processed by the following instances (in this order) until the first one provides an answer either from its cache or because the name is within its authority.

- application caches: use their own tiny caches, often limited to less than 30 minutes, all other queries done by operating system
- local stub resolver in operating system: has its town small cache and some static name definitions (e.g. localhost), queries DNS server as configured by user or DHCP
- Router/Gateway: resolves names within the LAN, queries ISP for all other names
- ISP/Carrier: knows names of own machines, queries authoritative server in own top level domain
- Authoritative Registries: control all second-level domains, e.g. VeriSign controls *.com and *.net, DENIC *.de, etc
- Root Servers: control top-level domains

1.5 Top Level Domains

Including test zones, there are 280 top level domains as of September 5, 2008. Most of these are country code top level domains. The most recent additions among the others are .aero, .asia, .biz, .cat, .coop, .edu, .gov, .info, .int, .jobs, .mil, .mobi, .museum, .name, .pro, .tel, .travel and the internationalized domain name (IDN) top level domains for Arabic, simplified and traditional Chinese, Greek, Devanagari (Hindi), Kanji, Hiragana and Katakana (Japanese), Hangul (Korean), Perso-Arabic (Persian), Cyrillic (Russian), Hebrew (Yiddish) that look like .xn–0zwm56d, .xn–zckzah, etc. in ASCII.

The list of top level domains and the root servers that delegate them are adminstered by IANA. Most top level domains are run by individual bodies. Many of them are national cooperations of providers in the respective country.

1.6 Root Servers

There are 13 root servers, six of them are distributed, which makes a total of 123 root servers as of 2006. The root zone includes all top level domains (.com, .net, .de, etc) with their name servers and provides glue records in the additional info section to provide IP addresses to these servers.

Although top level domains change rarely, the root servers are exposed to high traffic, around 15,000 queries per second [20]. Most of this traffic comes from denial of service attacks or misconfigured clients with only 2% of the queries being legitimate [19].

1.7 Importance of DNS

Because DNS facilitates name lookups for all network services it is one of the most important network services. Many applications rely on DNS beyond name resolving, e.g. e-mail uses a special resource record in DNS to facilitate message transport. As Amit Klein pointed out in his 2008 Black Hat presentation [8], forged DNS replies can be used to acquire certificates with forged identities, to receive others' mails and many more vulnerabilities all based on DNS or any service that uses DNS in a business process.

1.8 DNS Protocol History

Before the deployment of DNS, at first static lists of host names were used to map names to IP addresses. These static lists are still in use for special applications in modern systems, e.g. the hosts file in Windows. Later, the ARPA Host Name Server Protocol (NAMESERVER) specified in 1979 as IEN-116 [10] was used. Its flat topology proved unapt for the growth and changes of the Internet. DNS was its predecessor.

DNS was first specified (RFC 882, 883) and implemented in 1983. BIND is the most prominent implementation of DNS and was first released in 1985. It was not rewritten until BIND 9 in September 2000.

The DNS specification has been revised a number of times, but the general principle has not been changed since 1983. A security extension to DNS called DNSSEC was first specified in 1999, revised several times and is still not in general use today, although the deployment progresses slowly.

Cryptographic security extensions for DNS were first introduced in 1999 but are still not in general use.

1.9 Caching Name Servers

Virtually every single resolver has its own cache, so as not to query the same names in short intervals, thereby reducing load on the servers above him. All authoritative replies contain a time to live field (TTL), telling the caching servers how long to cache this entry before querying again. This facilitates the ability to frequently change some entries while preserving low computational cost and network load for other entries that do not change frequently.

Short lived names with a low time to live and frequent update of IP addresses are common for mobile services, where a device may change IP when roaming between different wireless cells. There are also dynamic DNS services for home users who want to get globally accessible host names for their home PCs connected via dynamic IP connections such dial-up, DSL or cable modem. These types of end user Internet connections often enforce disconnects every 12 or 24 hours which often causes the IP to change with each reconnect.

Seldom IP address updates are common for most servers, especially gateways and DNS servers.

1.10 Network Neutrality

Net neutrality in general is the principle of treating all devices, parties and connections and all types of content on a network equally and without discrimination.

In 2003, three American home user Internet access providers delivered their own search engine website with sponsored ads to their users when they entered a non-existent domain name into their web browsers. This was done by having a *.com entry in the resolvers for their customers. One of the goals of this service was generating revenue from the ads displayed on the search engine website. This type of service is sometimes called provider-in-the-middle-attack. After three weeks this service was abandoned altogether after protests from ICANN, but appeared again later in less prominent forms.

Net neutrality goes much further. There are more profitable ways to modify their customers' Internet access for the providers, such as including additional own ads, replacing original ads with their own, slowing down or denying access to websites of competing organizations, websites that criticize the provider or especially bandwidth demanding services, such as filesharing. Filtering all ads could be offered as a premium service.

There is an ongoing discussion about net neutrality with prominent supporters and objectors. In many countries there are laws or other regulations to guarantee a minimum of net neutrality. Another example of lacking net neutrality is the blocking or obstruction of voice over IP telephony and instant messaging on Internet capable mobile phones. Cellular network operators want to prevent the use of these services so as not to lose revenue from telephone calls made and short messages sent, because these traditional services are often billed much higher in terms of traffic cost than mobile IP Internet access.

1.11 Notable DNS Big Players

IANA (Internet Assigned Numbers Authority) is the governing body of IP address allocation and DNS root zone management. IANA is operated by ICANN (Internet Corporation for Assigned Names and Numbers) under contract to the United States Department of Commerce. Among other tasks, IANA manages the root name servers' data. IANA operates two top level domains (.arpa for reverse DNS lookups and .int) and the zone root-servers.net itself, denoting the root name servers. IANA also negotiates with other top-level domain operators over DNS issues. There have been some politically motivated proposals to decouple IANA from ICANN, but to no avail.

VeriSign is the registry of the two most popular generic top level domains, .com and .net. VeriSign is a for-profit company in private hands. DENIC is the registry of the most popular country code top level domain, .de. DENIC is a non-profit cooperative.

ISC is the developer of BIND, which is the defacto standard implementation of DNS. ISC is a public charity non-profit corporation and runs the "F" root server. ISC also maintains the central Usenet moderators list and relays for moderated groups. BIND is released under the BSD license (open source).

Dan Bernstein is a professor at the University of Illinois at Chicago. Bernstein is a security researcher and the developer of djbdns, the second most popular DNS implementation which has a strong focus on security. Bernstein has not updated djbdns since 2001 but has only recently published a new standard proposal for DNS security called DNSCurve. djbdns was placed in the pulic domain in December 2007 after being released freely and open source before, but explicitly without license.

Dan Kaminsky is a security researcher for IOActive and has discovered a powerful new attack model on DNS in 2008. He coordinated a simultaneous patch effort by many developers. Kaminsky is also known for his talks at the Black Hat IT security conference.

1.12 DNS Best Practices

The following is a quick guide to operating a DNS server in a secure and convenient manner.

There should be at least two DNS servers in a DMZ network segment. Using two different implementations is a good idea. Many implementations can easily be set up as a secondary server to a primary server running a different implementation, e.g. djbdns can be easily set up to act as a secondary server to BIND.

There should be an own machine for each DNS server on which no other services run. This allows for more available source ports, easier maintenance and predictable load behavior. These machines should run hardened operating system with default accounts and shares disabled, a strongly secured administration account, a reasonably setup intrusion detection software and a firewall in the same network segment. This helps mitigating denial of service attacks that exhibit a specific pattern on the DNS servers.

The e-mail address entry for each zone needs to be correct for applications to notify the administrator of irregularities. Caution must be taken when adding CNAME entries. It's best to avoid them whenever possible. CNAME entries must not have names used by other RRs.

Split-horizon DNS (also called split-view DNS or split-brain DNS) is a concept of resolving names in a different manner depending on the client's source address. It can be used to keep certain elements of the zone private, i.e. to prevent resolving of internal names to outside clients. Split-horizon DNS is a feature in most name server implementations but it can also can be implemented by running separate servers on separate hardware with corresponding firewalls, or by running separate processes on a single machine for separate client classes. Queries should be restricted by client IP whenever possible.

In other setups it is prudent to allow access to the caching name server only from within the own network but not from the Internet. Users from the Internet only need to connect to the authoritative server which should have recursion and caching deactivated. Access to the caching server can be restricted by using network address ranges and other firewall rules. In such a setup, Internet users cannot abuse the caching server, e.g. for cache poisoning attacks.

Zone transfers are used to synchronize secondary resolvers to primary servers. Although zone data is not generally considered private, zone transfers should be limited so that they can be done only by other servers and clients who actually need to do them, i.e. the secondary servers.

1.12.1 Gluelessness

Although referrals to name servers in foreign domains are allowed, this can cause a circular dependency:

example1.com NS ns.example2.com
example2.com NS ns.example1.com

RFC 1034, 1537 and 1912 specify that glue records are unnecessary in this situation. But without glue records, none of these domain names (example1.com, example2.com and all its subdomains) can be resolved, thus rendering all services unreachable. This problem might appear suddenly when one of these entries survived in the cache long enough and is later dropped from the cache. With glue records, this problem can be avoided:

example1.com NS ns.example2.com, ns.example2.com A 43.21.1.1
example2.com NS ns.example1.com, ns.example1.com A 43.21.1.2

Another obstacle in reaching the correct name server can be chains:

example1.com NS ns.example2.com
example2.com NS ns.example3.com
example3.com NS ns.example4.com
example4.com NS 12.31.23.123

In this scenario, a large number of queries is required for name resolution, delaying the query. Also, some resolvers may give up because of the large number of queries and memory required to resolve such a name. Care must be taken by DNS administrators to avoid such setups.

This is also true for CNAME entries (alias names) which should not point to other CNAME entries. Therefore, when setting up a CNAME to another name, the administrator should make sure the target name is not of the CNAME type.

1.12.2 Trusted servers

A name server that as a name on a deep subdomain level, i.e. is a subdomain to a subdomain etc., must trust all its parent domains. All parent domain name owners are able to influence to way the name server works up to the point of easy denial of service, e.g. w3.org used to have a name server w3csun1.cis.rl.ac.uk. Each of the owners of ac.uk, rl.ac.uk, and cis.rl.ac.uk were able to change the way w3.org would be reached.

Chapter 2

Attacking DNS

2.1 Intentions

There are a number of motives behind attacks on DNS infrastructure. Most of them do not aim at DNS itself but are means to accomplish other goals. Some of these are discussed here.

2.1.1 Denial of Service

By sending a forged NXDOMAIN reply indicating that a domain does not exist or by returning a wrong IP address for the domain or its web server the attacker is able to prevent users from reaching a host name including all services hosted there, e.g. websites and mail. Preventing visitors from connecting to the site may have various motivations:

- blackmail, especially against event sites depending on availability over a short period of time such as live coverage of sports or music, betting sites, voting sites, auctions and any kind of event coverage in general
- sabotage, especially against competing websites or organizations
- vandalism, sometimes politically or culturally motivated

2.1.2 Redirecting Web Traffic

Answering to queries about popular websites with an A record that points to an own web server can be used to redirect visitors of these websites. Attackers may try to redirect traffic to an ad site to directly generate profit from the large number of visitors.

It can also be used for phising, i.e. redirecting to a website with a login function that looks similar to the real website and recording the login data, possibly even asking for transaction authentication numbers.

Another use of redirecting web traffic is disinformation. This can be accomplished by redirecting traffic to a news site to a similar looking own website with false stories or manipulated stock quotes. These techniques can be used to manipulate stock quotes and votes.

2.1.3 Man-In-The-Middle-Attack with DNS

A man-in-the-middle attack with DNS means redirecting traffic to websites (or other services) over a relay that allows monitoring traffic. This can be used to tap personal information, login data and so on. Also, manipulation of data is possible, e.g. for fraud in online banking.

2.1.4 Censorship via DNS

Censorship via DNS was not intended by the inventors of DNS, but facilitates an easy way to prevent access to certain sites, especially for less skilled users. Countermeasures include directly visiting the site's IP address if it is known and doesn't change too often, choosing a different DNS server instead of the one suggested by the ISP, using web proxies, proxy networks such as TOR or even a VPN into an uncensored network.

Internet censorship has been implemented via DNS by a number of jurisdictions, the most famous being China. The Golden Shield Project (sometimes also referred to as Great Firewall of China) employs DNS servers that intentionally do not correctly resolve certain websites. Also, IP access to non-censoring DNS servers is blocked.

DNS based censorship is also in use in Denmark, Finland, Germany, Italy, Netherlands, Norway, Sweden and Thailand. The most common targets are websites displaying child pornography, but in some countries also websites containing pirated copyrighted material, general pornography (Denmark, Germany), information about censorship (Finland), Nazi propaganda (Germany) and foreign betting shops (Italy). Also, websites mocking the Crown or state founder are prone to banning in their respective countries.

2.2 Targets

With DNS being a distributed database, attacks on DNS infrastructure can aim at any element of this distributed database. Because DNS is a protocol from the days when other network stations where not considered as possibly hostile, there are many feasible ways to achieve goals associated with attacks on DNS.

2.2.1 Client Name Resolution: Stub Resolvers

The pre-configured DNS resolvers in operating systems are called stub resolvers because they usually do not offer many features. Their only job is forwarding application DNS requests to the DNS resolver at the upstream connection. This is typically a server on the LAN, a hardware router or the ISP's DNS resolver. Some stub resolvers use short-lived caching.

Attacking a stub resolver by sending forged replies requires an attack scenario where the attacker can cause the victim to issue a request. This can be done by embedding code in a website or HTML e-mail message. Therefore, this kind of attack has a very specific target, but little to no effect: There is only one victim per successful attack and the effect is short-lived. Also, it is difficult to find a scenario where an attacker can cause a request for a site he does not control while at the same time ensuring the victim actually tries to connect to the forged domain within a certain frame of time. Thus, direct attacks on single resolving attempts on stub resolvers are rare.

Attacks on a stub resolver are more feasible on a LAN or in other situations where the attacker is able to monitor traffic sent by the victim. This kind of attack run by a gateway can be used for censorship to block certain host names. This can done by simply issuing a fake NXDOMAIN reply. The Golden Shield Project (great firewall of China) uses this technique.

2.2.2 Client Visualization: IDN Homographs

Internationalized domain names (IDN) can be used to trick a user into trusting an unknown host name. This is achieved by registering domains that look like well-known domains, but use one or more lookalike Unicode characters and are really different names. Although no user will visit these sites by typing the domain or by visiting links on trusted websites, it can be used to make links to untrusted websites look legitimate. The most popular attack scenario would be fake e-mail messages from banking houses containing a link that looks like the real bank's website, but is actually a phising website.

Countermeasures include displaying international characters in puny code when they are from a different language than the top-level domain, only displaying IDN characters when they aren't composed of multiple languages or marking them with special background colors to allow the user to notice any difference to the legit name. All of these countermeasures must be implemented on end user client software and they all sacrifice some functionality or convenience.

2.2.3 Client Policies and Firewalls: Rebinding Attack

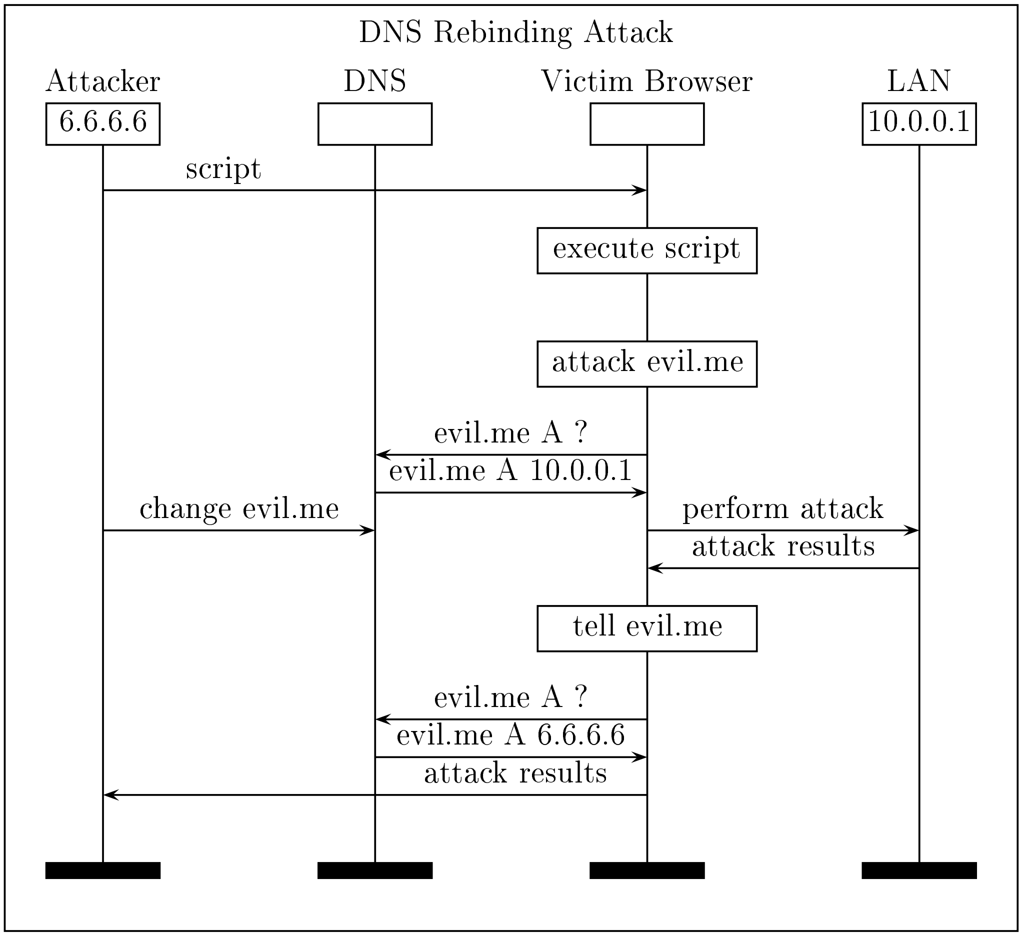
Rebinding is an attack that employs legit DNS uses to circumvent the same origin policy in web browsers [7]. The victim client must execute a script made by the attacker for the attack to work, which is typically achieved by iframes in websites, e.g. by an ad server. The scripts executed by the victim client are typically written in JavaScript, Java or Flash. Also, the attacker must be able to change the A resource record for a domain name quickly and frequently. The name may be a name he legitimately owns.

The attack works by having the client resolve the name in control of the attacker to different IPs in consecutive queries with a short (or zero) time to live entry. The script will read and write data to and from the same name, but the name will resolve to different IPs. These IPs may be internal IPs of the victim's LAN or external Internet IPs.

This can be harmful because it can change the scope of the IP without violating the same origin policy of the web browser. Same origin policies ensure that scripts within a frame in the browser can only read and send data to themselves. This is achieved by verifying host names, not IP addresses. Therefore, when the host name stays constant but the IP changes, the same-origin policy is bypassed. By circumventing the same origin policy, it is e.g. possible to steal cookies or to use the web browser as a proxy for scanning its local network, because the attacker can also return internal IPs (e.g. 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) of the victim's network and use rebinding to access internal network resources from outside.

There are proof-of-concept attacks that turn a web browser into a proxy and enable network scanning from outside the network by passing code to the browser and using DNS to bypass same origin policies.

In the following example, DNS rebinding is used to bypass same origin policies in a web browser to forward the results of an attack on a host on the LAN (IP 10.0.0.1, i.e. a file server) to an external attacker (IP 6.6.6.6). The web browser executes a script sent by the attacker. The attacker needs to control the name evil.me and must be able to change its A entry quickly.



Countermeasures against rebinding attacks include defining a minimum setting for the time to live (called DNS pinning), not allowing internal IPs to be passed through the network's name server and rejecting HTTP requests with unknown host headers [11].

2.2.4 Gateways: Pharming

Pharming is an attack to modify the client's networking settings to query the attacker's DNS server instead of the client's regular server. The attacker may set his DNS server up in any way he likes, blacklisting or forging host names as he sees fit. This is not an attack on the DNS protocol, but on insecure clients. DNS is used to achieve the attacker's goals.

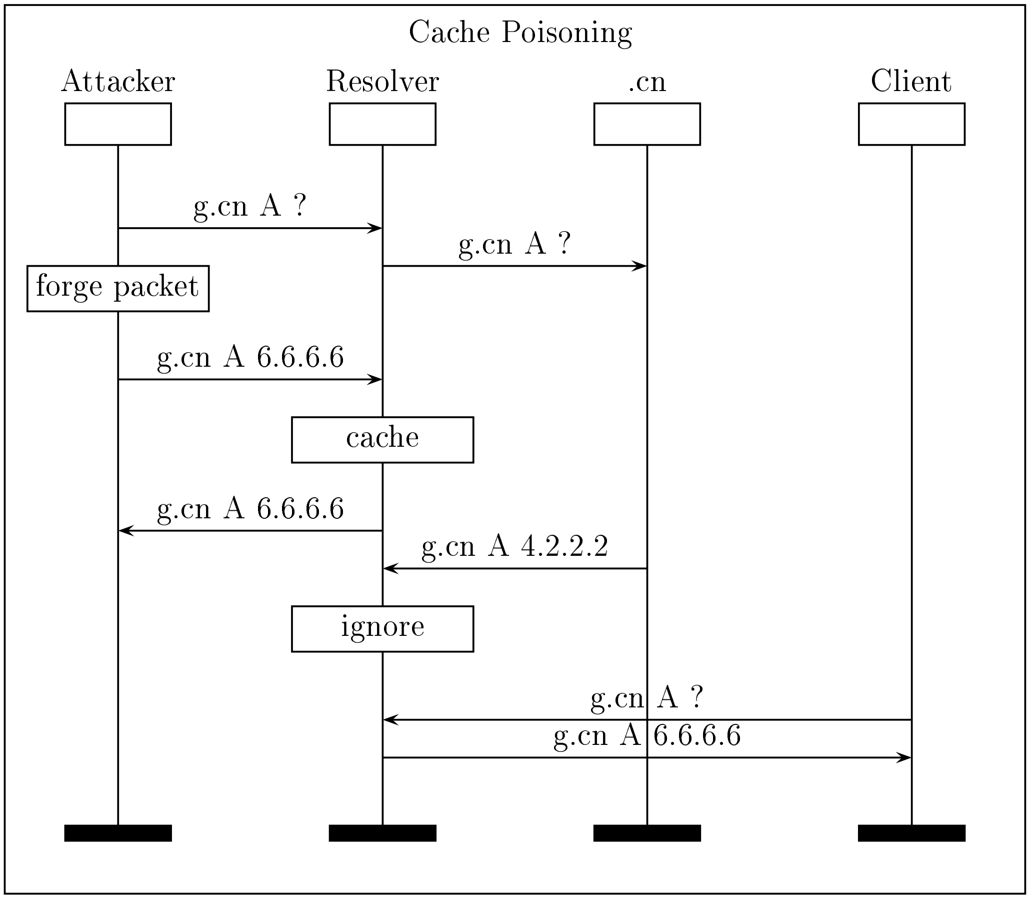
One common variant of this attack includes trojan software changing the localhosts file so that certain names will be directly mapped to fixed IPs without DNS querying, e.g. to prevent updating anti-virus software or to redirect traffic from popular websites to others. Another variant is drive-by pharming, which is scanning of WLANs from home Internet access routers for default administrative passwords and changing their DNS server settings to a server the attacker controls.

These attacks are serious because they have a long lasting effect on the client and are hard to detect with common antivirus software. When not checked carefully, the wrong IP for the client's DNS server entries will remain even after the trojan software was removed or after the router's administration password was changed.

2.2.5 Resolvers: Cache Poisoning Attack

The most serious attacks on DNS are cache poisoning attacks. In this attack model, the target is a caching resolver, usually one for a network, e.g. at an ISP. Success in this attack model means having the resolver accept forged replies and absorbing them into his cache. All queries to this name arriving later will be answered with the forged reply initially sent by the attacker.

The following diagram is an example of a cache poisoning attack on the domain g.cn. It is meant to illustrate the principle of cache poisoning attacks in general. Therefore, it is left open how attacker is able to convince the resolver that his reply is legit. In a real attack, he would have to exploit a certain implementation weakness or he would have to guess or predict the transaction ID, the port number of the query done by the resolver and he would need to know the source IP address of the upstream resolver queried by the resolver.



The attack is successful when the resolver accepts the attacker's forged reply and adds the entry into his cache. The real reply from the upstream resolver arrives too late and is ignored.

There have been various scenarios to achieve cache poisoning. Usually, a query is sent to the victim server querying him for the domain to be impersonated. This will cause the server to send a query and to listen for replies. The attacker must answer before the real reply reaches the attacked server. This is a race condition. When the real reply arrives first, the attacker has lost and cannot try again on the same name because it is already in the cache. The attack cannot be repeated on the same name for as long as the time to live in the real reply specifies.

To shorten this delay the attacker might try to crash the target server so that it will restart and clear its cache, which is only possible in some scenarios.

The attacker may also issue a large number of queries to a series of names until the cache is full and the oldest entries are dropped, which is difficult to predict. The attacker can try to improve his chances to win the race condition by running a denial-of-service-attack against the upstream resolvers along the legitimate resolving path, thus delaying the real reply.

Other improved cache poisoning attacks detailed later use collision attacks (birthday paradox) or exploit predictable transaction IDs.

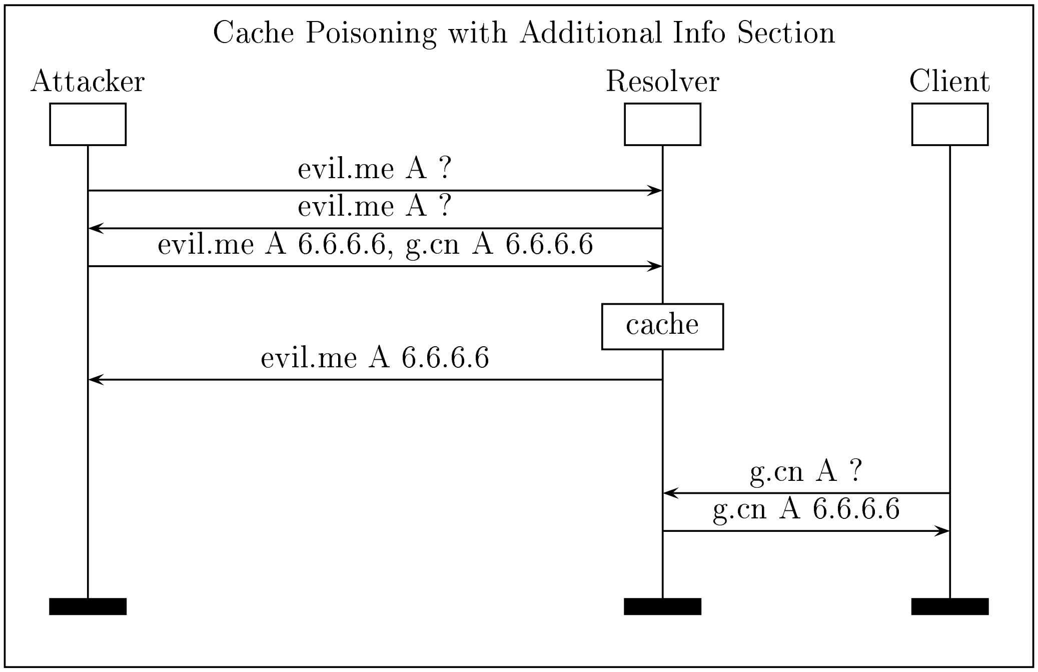
Chapter 3

DNS Security History

With DNS around since 1983 and the general complexity of the distributed database that composes DNS, there have been many ways to attack DNS. Some of these are exploits of disregardful implementations, some apply general cryptographic attacks, others exploit protocol design flaws.

3.1 Cache Poisoning via Additional Info Section

In 1993, Christoph Schuba discovered that entries in the additional information section in DNS replies can be used for cache poisoning in BIND. In this scenario, an attacker would query the victims' resolver for an IP of a domain he controls. The reply from his own server would contain an additional info section with an entry such as www.examplevictim.com A 12.34.56.78. This would cause the victims' server to add 12.34.56.78 as the IP for www.examplevictim.com in his cache and deliver 12.34.56.78 as IP for www.examplevictim.com to all his users. This effect is called cache poisoning, because unauthorized entries are updated into the cache. The effect is much greater than forging replies from a resolver to a single user because it affects all users of the poisoned resolver, e.g. all clients of the ISP. The poison remains on the server for as long as the TTL specified while the attack itself only takes two packets.



This attack was fixed by the introduction of bailiwick checking for the additional info section. Entries that are in-bailiwick, i.e. belong to the same domain as the authoritative answer in the answer section would be adopted into the cache, all others are considered out-of-bailiwick and would be ignored.

Therefore, an authoritative answer to www.example.com that also contains additional info for mail.example.com and ns2.example.com would have all its information added to the cache, but an entry to www.other.com would be ignored. However, the additional info section has later been used for more sophisticated attacks.

The most prominent of these attacks occurred in July 1997 when Eugene Kashpureff, President and chief technical officer of AlterNIC, used these cache poisoning techniques to redirect all visitors to www.InterNIC.com to AlterNIC's website. This attack was a political statement. Eugene Kashpureff was arrested for it and fined with criminal charges. The attack targeted BIND resolvers that where not yet patched to check bailiwicks in the additional info section as mentioned above. The attack gained some notoriety because of its large impact and the strict law enforcement against Kashpureff.

3.2 BIND Sequential Transaction IDs

In the second half of 1997 DNS security was becoming a more interesting issue with the Kashpureff case in the news. CERT discovered sequential transaction IDs in BIND in August 1997. BIND would issue transaction IDs sequentially (1, 2, 3, 4, ...). This was considered insecure, because an attacker could assemble and send a UDP packet with forged sender IP and correctly calculated transaction ID, which would cause a resolver to accept forged authoritative answers. Thus, cache poisoning was possible again.

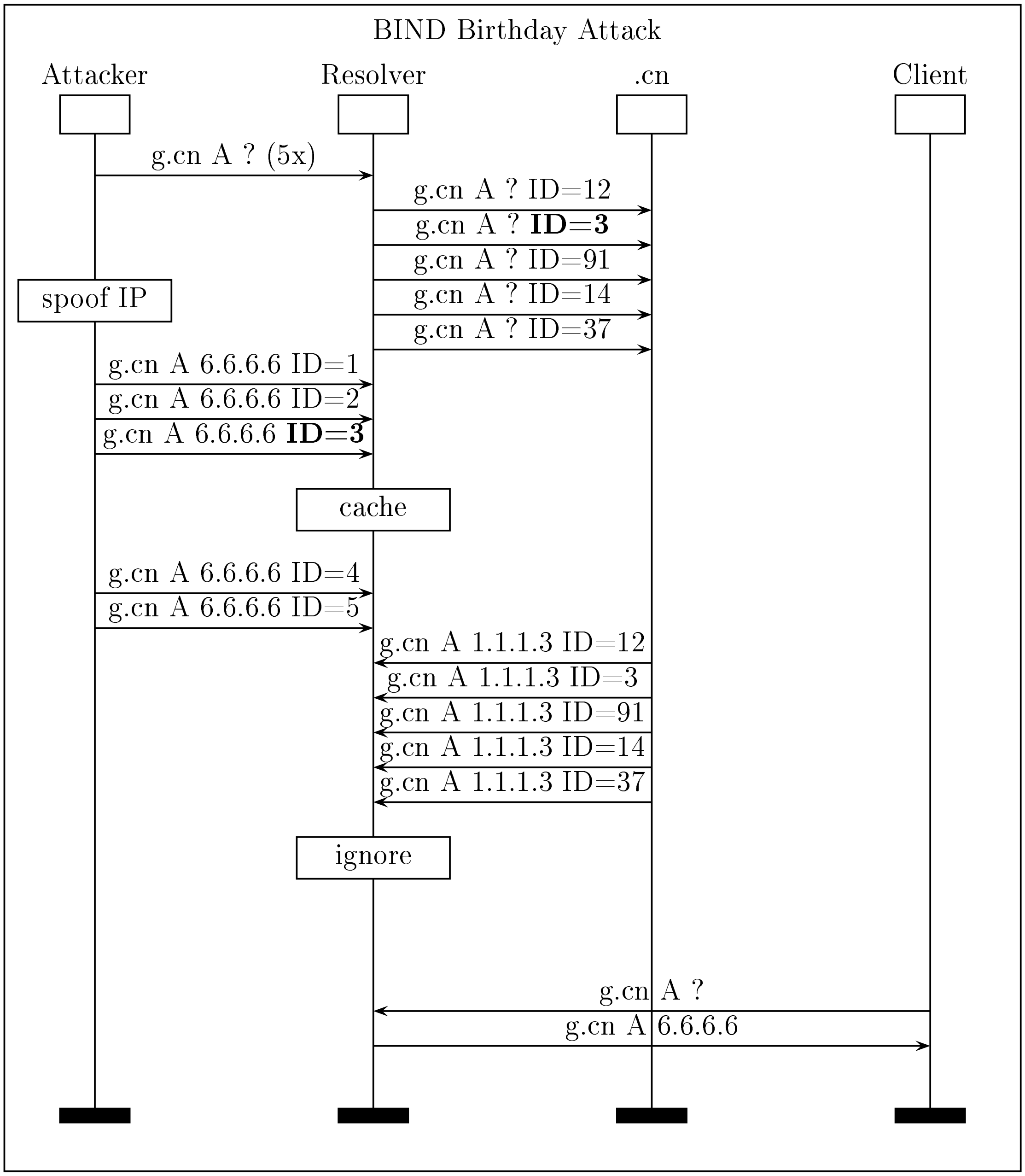
This issue was quickly fixed in many implementations with random numbers introduced into DNS resolvers. Some used the operating systems' functions for generating random numbers, others used own implementations.

3.3 BIND Birthday Attack

A birthday attack on DNS was first mentioned in July 2001 by Dan Bernstein and detailed in 2002 by Vagner Sacramento. Sacramento discovered that BIND sends multiple recursive queries for the same name simultaneously when asked for the same domain a number times simultaneously. Sending 100 queries about www.example.com would cause BIND to send 100 queries about www.example.com to its upstream resolver, each with different transaction IDs but otherwise identical. When sending a flood of forged replies the high number of possible legitimate replies increased the attacker's chances. This attack is one of the birthday paradox type: It is not important which of the questions asked by BIND is answered, as one single corresponding reply is enough for the resolver to accept the forged packet.

The attacker must send a sufficient number of forged replies before the first legit reply arrives. Success probability with this type of attack is 50% after

300 packets and almost 100% after 700 packets. Conventional attacks achieve only 1.1% after 700 packets.



In the example, the attacker sends five queries to a DNS resolver about the domain g.cn which it wants to poison. The resolver issues five queries to his upstream server, which is the .cn authoritative server. Each of these five queries has a different transaction ID. The attacker immediately starts sending forged reply packets with increasing transaction IDs and an IP he would like to add on the resolver's cache for g.cn, which is 6.6.6.6. The real upstream resolver replies too slow and the cache poisoning succeeds. Queries issued later by clients of the victim resolver will receive the forged reply.

BIND was fixed so as not to send multiple queries at the same time for the same query. The lesson learned from this attack was not to use multiple random numbers for a single transaction, a principle that applies in general software engineering.

3.4 Buffer Overflow in Resolvers

In June 2002, Joost Pol discovered a buffer overflow vulnerability in the gethostbyname() function in libresolv. The libresolv library is based on BIND and is used by Unix C programs to resolve host names into IP addresses. Buffer overflow vulnerabilities are typical for C programs and are often caused by careless string handling. Exploiting such a vulnerability achieves the ability for an attacker to crash the application or to cause the execution of code injected by the attacker on the vulnerable machine with privileges of the vulnerable application process. Because resolving names is a feature used by almost all network services, the buffer overflow vulnerability in the library affected all of them.

This vulnerability was fixed in BIND 4.9.9, 8.2.6, 8.3.3, and 9.2.2. Before releasing the update, ISC (developers of BIND and libresolv) claimed that a caching server between client and recursing server would be a workaround against this issue, which was disputed by Dan Bernstein. Two months later, Bernstein's claims were acknowledged by ISC. [2]

In April 2007, a similar vulnerabily in Microsoft DNS server was reported [17]. A patch was released one month later after exploits appeared.

3.5 Weak Pseudo-Random Numbers First Discovered

In his 2001 paper "Strange Attractors and TCP/IP Sequence Number Analysis" [21], Michal Zalewski of Bindview analyzed and compared the output of random number generators of different operating systems by observing the distribution of TCP sequence numbers in 3D space and applying some mathematical analysis. His conclusion was that most operating systems use weak random number generators and that prediction of the next number in a sequence is often possible with feasible effort, i.e. less that 1,000 monitored numbers.

Although primary analyzing 32 bit TCP sequence numbers, his results also apply on DNS transaction IDs, which are only 16 bit in length and therefore much more vulnerable. Transaction IDs are used as the single means of authentication in the DNS protocol. An attacker who can predict transaction IDs can force a DNS recursor to accept forged replies, because he cannot distinguish between real and forged replies. Zalewski also analyzed the transaction IDs of glibc 2.1.9x, Microsoft DNS server and Solaris 7 libc resolver, all of which expose vulnerable pseudo random numbers. Zalewski could not find weaknesses of this kind in BIND.

The results of Zalewski's research and the tools he released are applicable to any network service using random numbers.

In 2003, Joe Stewart used Zalewski's tools to test popular DNS server software. He found BIND 8.4.3 to be vulnerable with a 100% chance to correctly guess the next transaction ID after observing three transaction IDs. BIND 9.x relies on the host operating system, which was Linux 2.4.19 at the time of first testing. A 20% chance after 5,000 observed packets is considered to be a fairly good, but not perfect result. Testing djbdns showed a 30% chance after 5,000 observed packets, slightly worse than BIND 9, but nullified by the fact that djbdns uses source port randomization, so an attacker has to guess not only the transaciton ID, but a pair of both transaction ID and source port, which would considerably prolong an attack.

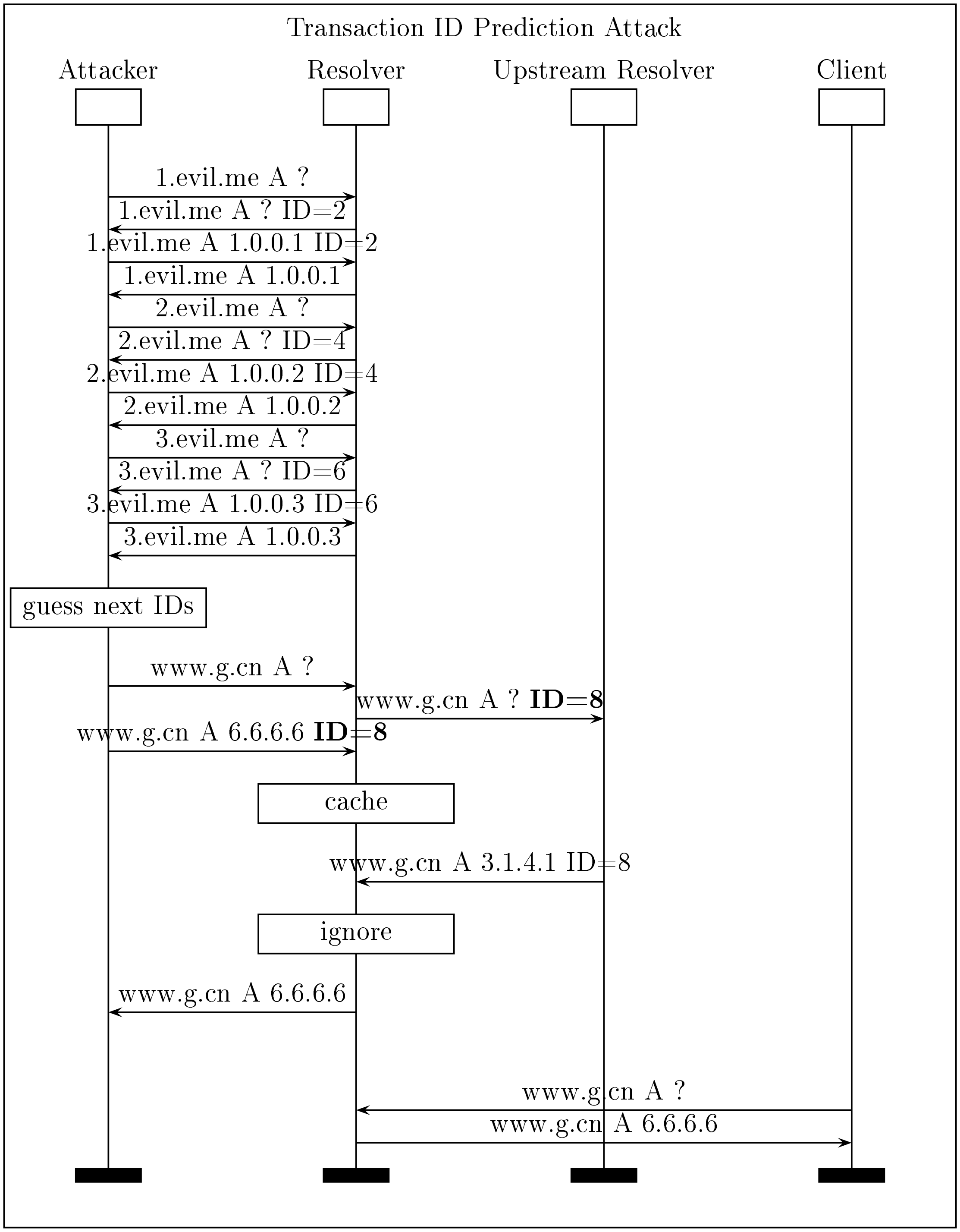
3.6 Amit Klein's Transaction ID Prediction in Resolvers

In 2007, Amit Klein analyzed the distribution of various DNS resolvers' transaction IDs in a manner similar to Michal Zalewski but also with source code analysis. Klein found out that there are weak random number generators in BIND, Windows DNS [13] and many other DNS resolver implementations.

To generate random numbers, BIND 9 used two LFSRs with 32 bit length each, constant feedback taps for all installations and 16 output bits in each cycle. LFSRs in general are not too bad of a choice for pseudo random numbers when a single output bit is used every round. However, BIND used 16 output bits per round. This was such a big mistake that different clocking of the LFSRs could not prevent calculation of the full internal state of both LFSRs after observing a sequence of outputs, thus making the random numbers completely predictable. Therefore, the transaction IDs were predictable after observing a small number of transaction IDs [14].

In the attack, the attacker first queries the victim resolver about a number of domains the attacker controls to record the sequence of transaction IDs from the queries to the attacker's name server. Then he uses this data to calculate the next few transaction IDs, queries the victim server for the domain name he wants to forge and immediately sends forged replies with the calculated transaction IDs and a forged sender IP matching that of the upstream resolver.

In a real attack, the attacker's DNS would send more than one forged packet to increase his chances because he cannot prevent other users from querying at the same time, e.g. packets with IDs 8, 10, 12 and so on. In this scenario, the attacker and his DNS server may be a single station as shown in the chart. It is crucial that the attacker can observe transaction IDs issued by the target resolver. The easiest way for him to achieve this is querying for names he controls.



This attack does not require packet interception. There is one race condition where the attacker's forged packet must arrive at the resolver before the legit packet from the upstream resolver. The attacker must be able to inject packets with forged sender IP, which many ISPs allow. The attacker needs to know the IP address of the upstream server, which would typically be that of the top level domain authoritative server. Like all cache poisoning attacks so far, this attack is only feasible with a host name that is not already in the cache.

Patches to improve the pseudo random number generator in BIND and other implementatios were released shortly thereafter, although cryptographically secure pseudo random number generators were already known and standardized years before. Some researchers even claim that the problem was known for ten years before it was fixed e.g. in Microsoft DNS Server [12].

3.7 Weak Pseudo-Random Numbers in Windows Stub Resolvers

In March-May 2007, Amit Klein discovered predictable transaction ID generators in the stub resolvers of the latest Microsoft Windows 2000/XP/Vista version [15]. Microsoft issued fixes one full year later in March 2008 for Windows Vista and in April 2008 for Windows 2000 and XP between the regular hotfix release dates (so-called patch days), only days before Amit Klein would disclose full information about the weakness to the public.

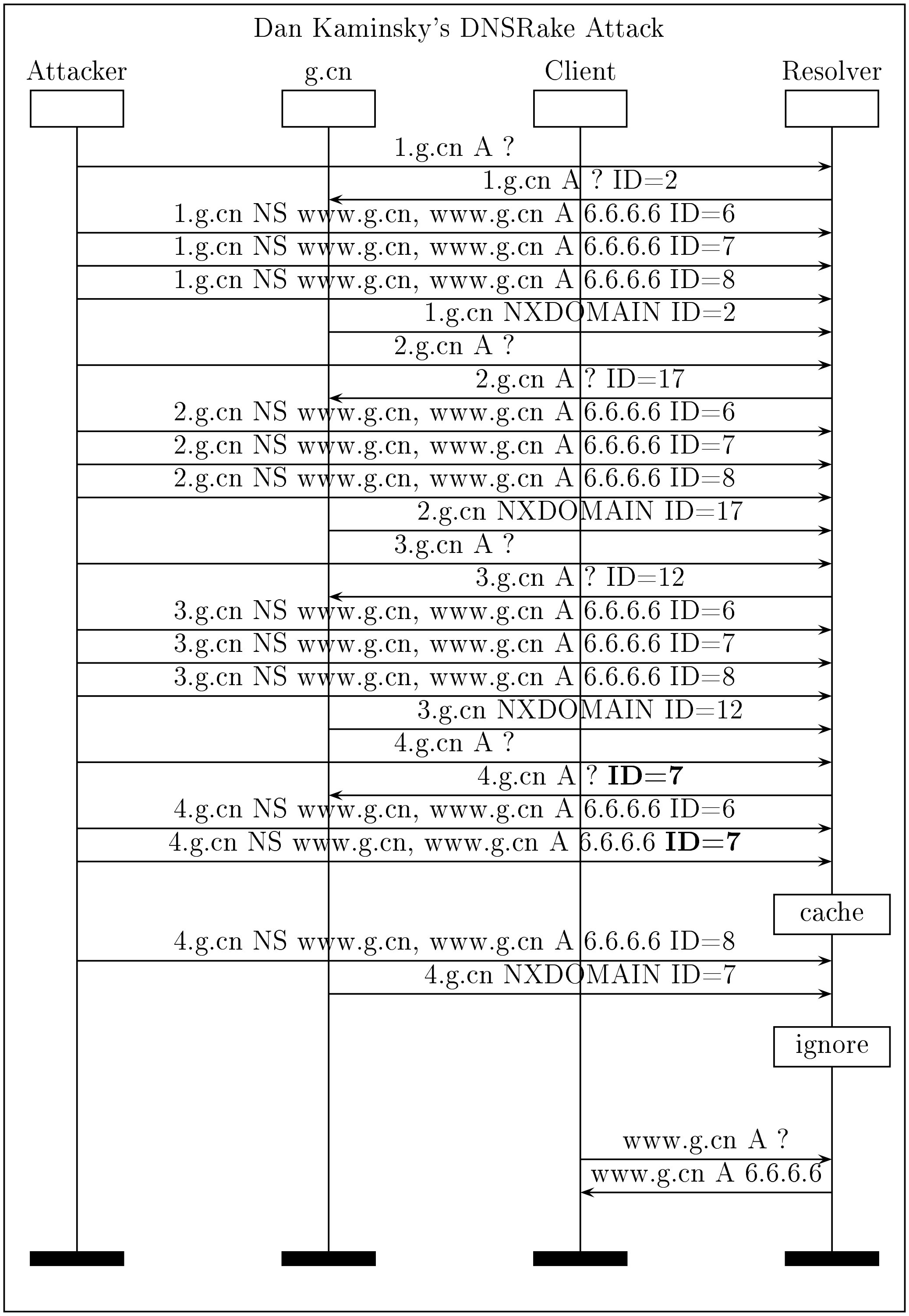
3.8 Dan Kaminsky's DNSRake Attack

In 2008 Dan Kaminsky developed a new attack model for a cache poisoning attack. His implementation DNSRake would not try guessing the correct transaction ID for a single query or a series of queries about www.example.com, but to a series of subdomains such as 1.example.com., 2.example.com, 3.example.com etc. One of the advantages of this approach is that these domains names will probably not already be in the cache, unlike www.example.com if it is a popular website. After not succeeding with guessing the transaction ID on the first attempt, the second attempt on 2.example.com can be started immediately, without having to wait for the time to live of the cached reply to www.example.com to expire. Also, queries for non-existent names take longer, thus delaying the real reply and giving the attacker better chances to win the race condition.

For each query sent by the attacker he will also send one or more forged reply packets with different transaction IDs. These packets redirect the recursor to a name server with the name to be poisoned and an A entry for this name pointing to the desired IP. When the transaction ID is eventually matched, the victim recursor will honor the A entry for the assumed name server, because it points to an entry within the same domain: 321.example.com is considered in-bailiwick with www.example.com. The recursor supposes that both names are controlled by the name server of example.com with witch it supposedly has just exchanged packets. This is the same process as any other name resolution.

The attacker's chances to succeed are only one in 65,535 per forged reply and query, but he can sequentially issue as many queries as he likes without having to wait for the time to live to expire. Also, with more than one forged reply packet per query, his chances increase per query. With 100 forged replies per query arriving quicker than the legit reply at the recursor, his chances increase to one in 655 per query. He will only need 328 queries with 100 replies each for 50% success probability. The attack can be successfully executed on a typical home user connection in less then ten seconds if the access provider allows forged sender IPs. As the attacker can sequentially try as many subdomains as he likes, his chances for successfully matching the transaction ID increase with every new subdomain.

In the following example, the attacker impersonates g.cn and convinces the resolver that www.g.cn is 6.6.6.6 and the resolver adds this information to his cache, serving it to clients who ask about www.g.cn later. Here, the attacker always uses the same transaction IDs in his forged replies. He might just as well use random numbers. There is no real difference in these variants because the attacker has to guess new numbers for each query. Here, three replies per query are used.


Dan Kaminsky's DNSRake Attack
Attacker
g.cn
Client
Resolver
1.g.cn A ?
1.g.cn A ? ID=2
1.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=6
1.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=7
1.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=8
1.g.cn NXDOMAIN ID=2
2.g.cn A ?
2.g.cn A ? ID=17
2.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=6
2.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=7
2.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=8
2.g.cn NXDOMAIN ID=17
3.g.cn A ?
3.g.cn A ? ID=12
3.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=6
3.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=7
3.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=8
3.g.cn NXDOMAIN ID=12
4.g.cn A ?
4.g.cn A ? **ID=7**
4.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=6
4.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 **ID=7**
cache
4.g.cn NS www.g.cn, www.g.cn A 6.6.6.6 ID=8
4.g.cn NXDOMAIN ID=7
ignore
www.g.cn A ?
www.g.cn A 6.6.6.6

Since handling the additional info section cannot be made more restrictive without breaking DNS in situations where glue records are required and because even a perfect random distribution of transaction IDs does not help, there has been an effort among DNS recursor developers to add source port randomization to their implementations. This is the only known means to delay this attack to a point where it is less threatening. With source port randomization, the attacker also has to guess the correct source port, which reduces his chances from one in 65,535 to one in 4 billion per packet.

Because of the severity of this new attack model, Kaminsky did not publish details and worked with vendors to release patches. All details of the attack were finally made public in August 2008, 30 days after the patches were released. There were some deployment issues with the patch when firewalls restricted or actively reduced the number of available source ports. Kaminsky himself considers source port randomization as a stopgap remedy with the DNS protocol still being in need of change, especially because of steadily growing bandwidth. So far, only DNSSEC is considered as a sustainable remedy.

An exploit named Evilgrade soon appeared which poisoned DNS entries to popular automatic software update URI's and delivers code of the attackers' choice instead of the real software updates. Many automatic software update functions in software download patches from sources only verified by their hostnames and then execute them automatically. With the poisoned entries and an according setup of own download servers, the attacker is able to execute code (i.e. install malware) on all clients who use both the poisoned name servers and the automatic updating software targeted by the attacker.

3.9 Notes on Top Level Domain Wildcards

Since 2003, so-called provider-in-the-middle attacks became popular: Home user Internet access providers deliver a own search engine website with sponsored ads to their users when they try to visit a non-existent domain name with their web browsers. This is done by having a *.com entry in the resolvers for their customers.

- VeriSign, the authoritative registry operator for .com and .net, redirected all queries to non-existent domains to their own website called site finder, which displayed ads for VeriSign and some ad partners. The catch-all registration in .com and .net broke several services such as automatic checking of domain existence which is e.g. used in fighting spam. Patches for DNS resolvers ignoring these wildcard domains were quickly released. Because of the protest of ICANN VeriSign abandoned SiteFinder after three weeks.

- Paxfire does essentially the same, but is not an authoritative registry. Some ISPs cooperate with Paxfire and redirect *.com and other top-level domain wildcards to Paxfire. Their users' browser will show a Paxfire ad site when trying to visit a website on a non-existent domain. VeriSign later also offered a service for ISPs to redirect non-existent subdomains.

- In 2006 the .cm (Cameroon) registry issued *.cm to an ad site, redirecting some mistyped .com domains. ICANN does not have authority over national TLDs, therefore, this service is still active today.

- When an user enters a non-existent domain name into Microsoft Internet Explorer's address bar, he is by default automatically redirected to Microsoft's MSN search engine with the entered address as the search query string. This can be deactivated or changed to other search engines in Internet Explorer's configuration. MSN's biggest rival (Google) is not in the list of preconfigured search in engines although it is much more popular.

- In 2008 many ISPs in the USA return ad sites for nonexistent subdomains to existing second level domains, e.g. ww.example.com. These sites have ad slots filled by others companies. This is considered a security risk, as the ad provider would be able to read and modify the cookies of the parent domain, e.g. example.com, which would also be used by www.example.com. This can be a way to steal personal data such as e-mail address, username or password. Care must be taken by the operator of any web service with personal data in cookies to prevent this [9].

There also are two known provider-in-the-middle attacks in Germany:

- In October 2008, Hansenet starting forwarding traffic from AliceDSL clients. All non-existent domain queries starting with www. would be redirected to 64.236.172.120, which is an AOL search engine website. This service is also opt-out only, with the link to opt-out reachable from the search engine website.

- Kabel Deutschland redirects users to a search site called "Kabel Deutschland DNS Assistance" full of Google ads when they try to visit a non-existent website.

Many users are upset about these services, because they often are not opt-in. Also, they ignore the settings in the users' web browsers, which would usually redirect to a user defined search engine or display an error message when trying to visit a website with a non-existent domain name. These wildcard domains may also be a security problem because of the embedded ads, which are often served by less trustworthy instances and the cookie stealing/modifying issue mentioned above.

In January 2001, yahoo.com and microsoft.com were accidentally hijacked by a hosting company which published a *.com wildcard resource record that spread to other resolvers. As Dan Bernstein points out, technically, these entries should be considered as poison when sent by anybody else than the real authority, which is VeriSign for .com [1].

Spreading a wildcard RR about a top level domain to resolvers on other systems is not possible anymore because of better bailiwick checking that replaced BIND's credibility system. The credibility system would allow authoritative name servers that seem legit to inject information beyond the authority into the cache of a recursor.

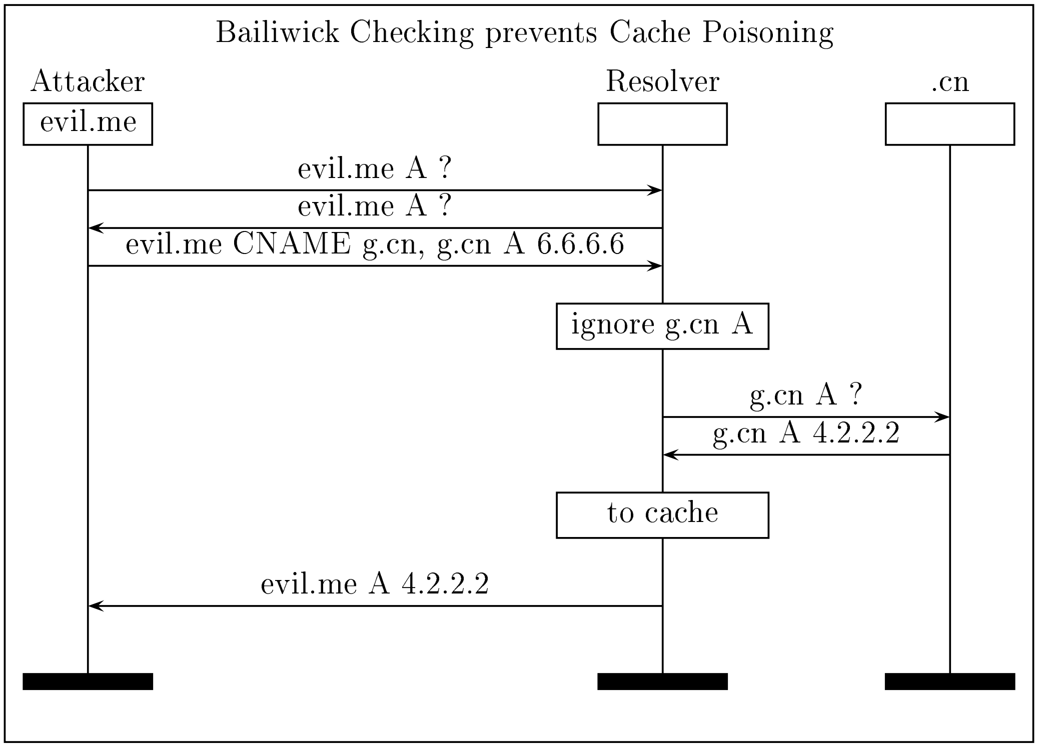
Chapter 4

Name Chaining Attacks

Name chaining attacks are a superclass of cache poisoning attacks [5]. Therefore, an attacker must be able to modify traffic or to guess source port and transaction ID of a query and beat the real resolver in a race condition. Name chaining attacks are characterized by having the resolver query more names chosen by the attacker. The intention is insertion of more forged data into the resolvers cache because it can be placed in both the authority and the answer section. So far, these attacks are hypothetical.

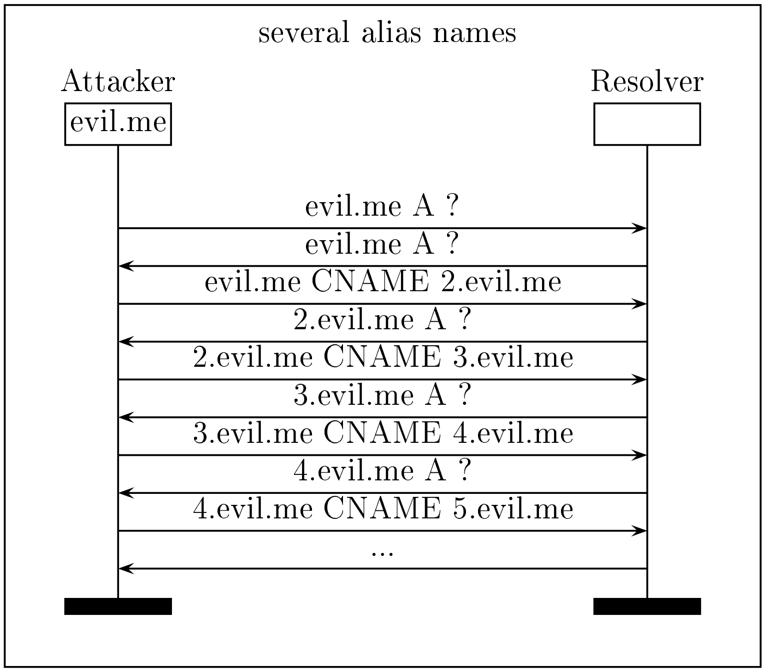
Bailiwick checking was introduced to prevent cache poisoning by checking if all data returned in a reply is within the authority of the sender. Most implementations execute bailiwick checking by testing whether the domain queried is trying to establish information about itself or about a subdomain of itself and discard all other information. This prevents cache poisoning via the additional info section and therefore, name chaining via redirects too.

In the following example an attacker causes a resolver to query "evil.me A ?", then returning "evil.me CNAME g.cn, g.cn A 6.6.6.6". The resolver would not accept the additional info section "g.cn A 6.6.6.6", because g.cn is considered out-of-bailiwick to evil.me. Therefore, evil.me has no authority over the domain g.cn or any of its subdomains.



In 2008 Dan Kaminsky found an attack model (discussed earlier) where the attacker could cause the victim to issue a series of queries for subdomains of the target domain. Each new query would increase the chances of randomly matching the transaction ID. However, it is unclear whether this attack model can be called name chaining, as it relies on a single forged reply packet to be accepted by the resolver.

Name chains are also used by attackers who cause the victim to sequentially query a number of names the attacker owns with a single query. This is achieved by using redirections in the answer section of each reply.



This technique can be used for examination of the resolver's behavior, e.g. transaction ID prediction as performed by Amit Klein (discussed earlier).

Translating a domain name via several alias names into a wrong IP address is not possible with a prudent name server implementation. Somewhere along the resolution, the answer would have to leave the attacker's authority for such an attack to work. This would be detected by recent resolver implementations and those entries would be ignored. There are no known applications of name chaining attacks in DNS.

Chapter 5

Improving DNS Security

There are a number of ways to prevent known attacks on DNS which may also hinder unknown attack models. Some of these methods are implementation guidelines, others change the DNS protocol. Generally, there is a trade-off between compatibility and security.

5.1 DNS over TCP

By default, DNS uses UDP for all queries and TCP for queries that exceed 512 byte. The use of TCP for all queries would make IP source forgery almost impossible, but it would also create additional load on the DNS server, because the state of the connection must be stored in the servers memory. Network load would also increase because a TCP handshake requires three packets to be sent before any payload is transmitted and four packets to close the connection. As a typical DNS query between two party is only one UDP packet in each direction, the use of DNS over TCP would create 4.5 times the network traffic. Also, denial of service attacks would be made easier because the server has to leave the ports open and wait for packets to arrive until a time-out occurs.

TCP headers contain a 32 bit segment ID which is effectively a means against forged replies from an attacker who cannot receive the actual queries. However, the use of TCP cannot prevent man-in-the-middle attacks. Also, many deployed DNS resolver implementations ignore TCP queries when they have not issued a truncated UDP packet before, which is the signal for the client to query again with TCP. Therefore, mandatory DNS over TCP is not considered a good choice in most networks.

5.2 Cryptographically Secure Pseudo Random Number Generators

The numbers used as transaction ID need to be unpredictable. This is achieved by the use of cryptographically secure pseudo random number generators (CSPRNGs). CSPRNGs are well-researched and there are various standardized functions for them since the 1980s, which do not allow an observer to (easily) recognize the internal state of the CSPRNG, thus preventing prediction of the next random numbers used. In 1994 an RFC was published on this topic [6] that called for the use of true random data in secure random number generators, i.e. randomness based metered hardware values. These techniques were later implemented in the Unix random number generator /dev/random.

Still, many developers failed to recognize the importance of secure pseudo random numbers. The most popular DNS server implementations have added CSPRNGs as late as 2007. Weak random number generators in client implementations are still common in 2008. A patch for Windows DNS stub resolvers was released only recently.

5.3 Source Port Randomization

Source port randomization uses the fact that an answer to a query will always be sent to the port it originated from by using random numbers as source port. It is a means to hamper forging of DNS query reply packets. Therefore, it cannot render an attack impossible, the attack just becomes more expensive for the attacker. Instead of guessing one correct transaction IDs among

65,535 possible values, the attacker has to guess 65,535 possible transaction IDs and the number of ports the revolver is sending packets from - up to 65,000. Since both transaction ID and source port must be guessed correctly at the same time, this requires guessing one correct pair among 65,535 times the number of ports - 2.1 billion packets for 50% chance of success. In practice, this renders all known attacks impossible on networks slower than a fast LAN.

Dan Kaminsky's Chaining Attack can be executed on a Gigabit Ethernet LAN in the course of 10 hours even if the victim resolver is using full port randomization. This is not considered critical since an attacker within the LAN could perform other more threatening attacks, e.g. man-in-the-middle via ARP poisoning or DHCP poisoning. He could then intercept and modify all DNS traffic, act as a DNS resolver himself or act as a gateway to alter incoming authoritative replies. Also, it is considered difficult to send such amounts of data even on a Gigabit Ethernet LAN without alerting even basic intrusion detection software or suspicious users.

Port randomization was uncommon in resolver implementations with djbdns being the most notable one. However, on July 8th, 2008 several developers released patches for their implementations simultaneously to introduce port randomization. This joint effort was driven and coordinated by Dan Kaminsky to mitigate his newly found attack model (discussed earlier).

Even when an attacker could find a new birthday paradox-type attack scenario on a source-port randomizing resolver, he would have to send too many packets for a home network connection to have a realistic chance to succeed in a race condition, i.e. without suppressing the real reply.

The same that has been said about the randomness of the transaction ID applies for the source port, too. Good random numbers are required to achieve the full increase in security.

5.4 The 0x20 Bit in Domain Names

0x20 as described by Vixie and Dagon in an Internet draft [18] in March 2008 is a technique to make queries more identifiable, just like source port randomization and cryptographically strong random query IDs. 0x20 exploits the fact that DNS resolvers ignore case by protocol standard but almost all known implementations retain case because they simply copy strings. A query about WWW.EXAMPLE.COM will be treated the same as a query about www.example.com, but both will have different reply packets with the upper- and lowercase letters still intact just like in the original query. This also applies for mixed case, such as wWw.eXAMmpLe.COm. The effect is a kind of covert channel that results in one bit of additional entropy for the query per character in the domain name. 0x20 is named like this because it is the bit mask of the difference in upper and lower case letters in ASCII encoding (A=0x41, a=0x61).

An attacker who cannot guess the combination of upper- and lowercase letters in a domain name in a query cannot do better that to send forged answers with all possible combinations of upper- and lowercase letters, which costs 2^{n-1} (with n the number of letters in name) attempts on average.

Because retaining but ignoring case is standard in most implementations in use, this approach works with nearly all implementations already deployed. DNS server implementations need to randomize case to take advantage of additional security in queries sent by themselves. This is not considered as a hard task for software developers.

The biggest disadvantage is the dependence on the number of letters in a domain. a1111111.com would only receive four additional bits of entropy, www.example.com 13 bits, meaning $2^{13-1} = 4096$ the average effort for an attack. The shortest commonly used domain name is probably g.cn for Google China, which would only gain three bits, meaning four times the average effort for an attack.

The 0x20 Internet draft also suggests methods for dealing with servers not retaining case to be added to the DNS protocol specification.

5.5 IPv6 and IPsec

IPsec (Internet Protocol Security) is a suite of protocols for securing IP communications. It provides authentication and/or encryption for each packet transmitted. This would increase security for DNS and prevent attackers from forging packets even when they are able to guess transaction ID and source port. IPsec also facilitates cryptographic key establishment.

IPSEC was introduced in 1995 with RFCs 1825 and 1829, redefined in 1998 with RFC 2401 and 2412 and redefined again in December 2005 with RFC 4301 and 4309, with each redefinition being incompatible with its predecessor. With the last update, IPSEC was relabeled IPsec.

IPsec is a mandatory component for IPv6, but can be used with IPv4, too. Because of the slow deployment of IPv6, today IPsec is more often used in IPv4.

IPv6 is a replacement for IPv4. It was first defined in 1996 with RFC 2460 and designed to add some features IPv4 was missing. As IPv4 was not originally meant to be used on a network with hostile parties, it lacks security. Another shortcoming is the limited number of available addresses. Without NAT, only $2^{32} = 4$ billion hosts are possible in IPv4, and many large address blocks of $2^{24} = 16.8$ million addresses have been delegated to universities and cooperations before the rapid growth of the number of Internet users since the late 1990s. This has become a burden for emerging markets, especially Asian Internet service providers. IPv6 provides 2^{128} addresses and IPsec, but isn't deployed widely and creates connectivity issues for networks not aware of IPv6. Although most current network devices and applications are IPv6-capable, it is not deployed yet on a wide basis, because of the cost of replacing old devices and installations.

5.6 DNSSEC

DNSSEC (also called Secure DNS) is an extension to DNS that facilitates origin authentication of DNS data, data integrity and authenticated denial of existence. This would make DNS secure to the point where an attacker cannot forge packets without breaking strong cryptography or using a fallback to legacy DNS. DNSSEC cryptographically binds query replies to their query IDs, so not even man-in-the-middle attacks in scenarios where the attacker controls the channel would be possible anymore. The attacker cannot succeed because he cannot create valid digital signatures without knowledge of the private key of the real server.

Digital signatures in DNS reply packets are created for each resource record. Signatures are verified against the public key of the sender, which can be exchanged via DNSSEC queries, which are signed by the parent domain. Therefore, each new client is required to already have a list of trusted keys. There also is some controversy about who is to hold the private key for the root zone, .com and other important top-level domains because of the power and responsibility that comes with this task.

The recommended key sizes for DNSSEC are 1024, 1300 and 2048 bit for low-value, medium-value and high-value domains respectively when the key is rolled over once a year. Value is depending on the view of the zone owner. Typically, higher leafs in the DNS tree are considered more valuable [16]. Greater key lengths increase computational load on the parties involved, smaller key sizes than 1024 are considered insecure. Key rollover on a regular basis is suggested to prevent attackers from breaking a key by using long-running calculations.

DNSSEC was first standardized in RFC2535 in March 1999 and revised by RFC4033 in March 2005 because of scaling issues. Although today most resolver implementations support DNSSEC it is still not deployed widely on public networks, because there are some technical and political concerns regarding its large-scale use, many of them related to key management.

DNSSEC also introduces NSEC, a resource record (RR) for authenticated denial of existence. It facilitates a resolver to signal a client that a domain within his authority does not exist without need to calculate a signature on-line. This avoids costly signature operations and helps avoid having the private key on the same system. Since this reply needs a signature, the new resource record NSEC was introduced. This resource record provides the closest two existing names in the zone in canonical order, indicating that no other names exists between them. However, this creates a new security issue called Zone Walking or Zone Enumeration. The way NSEC works allows an attacker to completely list the zone and monitor it for updates. The existence of a public, complete listing of domains is often not desirable and in some scenarios objectonable, e.g. when using DNS records to store personal data. This data privacy issue is one of the main reasons why DNSSEC was not yet widely deployed.

RFC4470 and RFC4471 released in April 2006 describe a method to dynamically create NSEC records and their signatures, thus preventing zone walking. However, this method requires private keys to be stored on all name servers on-line which limits its use to some special cases. Also, there were no implementations.

The newly introduced NSEC3 resource record as described in RFC5155 in March 2008 is another method to prevent zone walking. It uses hashed host names instead of plaintext and therefore does not disclose any additional zone data to an attacker. It preserves backwards compatibility as all implementations not aware of NSEC3 will not try to verify hashed host names but simply consider the answer to be insecure. Some implementations already use NSEC3 and all major implementations including BIND 9 are currently adding support for NSEC3. Therefore, the zone walking issue seems to be solved.

Signed top level domains are:

- .bg since October 2007
- .br has deployed DNSSEC and the .jus.br (Judiciary) domain has mandatory DNSSEC use
- .cz deploys DNSSEC in September 2008
- .pr has deployed DNSSEC in August 2006
- .se was signed in 2005

There are about 1000 signed second level domains in other TLDs, many of them within .ru.

A survey conducted by the ccNSO in 2007 among 61 ccTLD registries about DNSSEC deployment [4] lists various reasons why most of them have not deployed DNSSEC yet. General lack of resources and waiting for DNSSEC to mature are the most popular reasons. Low project priority and the fact that the root zone is not signed yet were the less popular reasons. Three ccTLD registries already have a DNSSEC test running and are ready for rollout, but are waiting for the root zone to be signed and for the zone walking issue to be solved.

There is no DNSSEC signature checking among operating system stub resolvers yet. A Firefox plugin for DNSSEC signature checking called Drill Extension is available which would at least provide protection against DNS spoofing in websites on the end user connection. However, as cache poisoning is a much more powerful attack than a direct attack on the end user, the deployment of DNSSEC still helps increasing DNS security by preventing cache poisoning between DNSSEC-aware recursors.

5.7 DNSCurve

DNSCurve is new a protocol that was only recently announced by Dan Bernstein [3]. Designed as an incompatible alternative to DNSSEC providing link-level public key protection to DNS packets, it uses 255 bit elliptic curve Diffie-Hellmann as cryptographic primitive instead of RSA for quicker creation and verification of signatures while providing more security than 1024 bit RSA. DNSCurve uses the name server's hostname to store its public key. As it is manually added on the name server one level above, it is already known to the verifier.

Currently DNSCurve development focuses on an updated version of dnscache from djbdns to integrate DNSCurve and on a forwarding service to be implemented which adds DNSCurve security features. The forwarder is run on the same machine as the resolver but with a different IP. It would allow using existing name server implementations without changing their zones or adding key management.

uz5xgm1kx1zj8xsh51zp315k0rw7dcsgyxqh2sl7g8tjg25ltcvhyw.nytimes.com is an example of a host name containing its public key. The first subdomain starts with uz5, is exactly 54 characters long and does not to contain the characters a, e, i and o to distiguish it from ordinary domain names.

DNSCurve is also meant to add confidentiality to DNS by encrypting requests and replies. It also said to add some protection against denial-of-service attacks. DNSCurve is supposed to be faster, more secure and easier to deploy and administer than DNSSEC because it only uses existing records in databases and zonefiles, e.g. for public key storage. It is difficult to verify these claims because there are no implementations yet.

5.8 DNS Intrusion Detection

Intrusion detection is a means to change behavior of a server when an attack is underway. This is achieved by monitoring incoming packets, detecting unusual traffic and changing the servers' behavior in a pre-specified way or in a way chosen by the administrator in an unforeseen attack or when automatic means fail. Many of the means to mitigate an attack are executed at the cost of performance, so care must be taken.

Methods to detect an attack include:

- Monitoring the number of packets received with wrong transaction ID, wrong port number or unauthorized content in the additional info section
- Monitoring the number of unusually large packets
- Waiting for a second reply to arrive for a single query can detect forged replies if the authentic replies are not suppressed by the attacker, e.g. by a simultaneous denial-of-service attack on the authoritative server.
- Monitor the number of NXDOMAIN replies and alert if it is exceedingly high.
- Reresolving a name after a short delay and checking if the results are identical is not feasible because there a legitimate reasons for one domain to have different IPs, e.g. load balancing.

Methods to apply during an attack:

- Limit data rate for certain senders. This slows attacks down and reduces general server load, but can also slow down regular users if the sender is another authoritative server.

- Switching outgoing queries to TCP can make an attack harder because the attacker has to find a way to spoof the correct 32 bit TCP sequence number. However, this creates additional traffic. Also, many servers including root servers do not support TCP packets unless they exceed 512 byte, which is uncommon in most scenarios.

- Limiting data rate for the whole server slows attacks down and reduces general server load, but will also slow down regular traffic. Because the attacker probably easily fills all network queues, regular users cannot reach the server at all. Thus, general rate limiting in attack mode facilitates a denial-of-service attack and should not be used.

The most popular IDS systems for DNS include snort (GNU public license) and Microsoft Internet Security and Acceleration Server (proprietary license).

5.9 Conclusion: Entropy vs. Cryptography

Entropy is the random data needed for the reply packet to authenticate against the resolver. The more entropy is used, the more effort the attacker needs to succeed in forging a packet. At first, only the transaction ID was used, which over the years gradually increased entropy from sequential numbers to weak random number generators (late 1990s) to cryptographically strong random number generators (2007). The transaction ID provides 16 bit of entropy by DNS protocol standard.

In 2008, source port randomization was added on most implementations, adding up to 16 bits of entropy. Source port randomization is limited by firewalls and general network policies, often delivering just 14 bits.

The deployment of 0x20 would add 3 to 15 bits of entropy depending on the lenth of the domain name using it and could be deployed within months, should the draft be approved. Using TCP instead of UDP for all queries would add another 32 bit of entropy, but has serious network and memory load side effects.

In a man-in-the-middle-attack scenerio where the attacker controls the channel (i.e. LAN or weak routing security), all of the above mentioned steps can be circumvented by the attacker without significant effort. Cryptographic protocols are required to prevent these attacks.

DNSSEC cryptographically authenticates query replys by the use of digital signatures. This adds entropy with the key length because the attacker would have to forge signatures. In general at least 1024 bit is used as key length, which is generally considered to be equally secure as a 80 bit symmetric key. Thus, an attacker needs another 2^{79} attempts on average to forge a signature, which is ample in a race condition. A smarter attacker could try to guess the private key, which would take equally long but can be done offline. However, DNSSEC is difficult to deploy in terms of software configuration, security policies and key management.

DNSCurve is designed to achieve even more goals than DNSSEC, to provide more cryptograhic security (corresponding to a 127 bit symetric key) and to be easier to deploy than DNSSEC. Further studies are needed to show if these claims are true.

Chapter 6

Summary

DNS is a quite simple protocol with a huge impact on everyday users' experience of the Internet. 13 root servers are indirectly used by 1.4 billion people to reach 540 million hosts on the Internet. But despite the generally smooth performance for end users of DNS, its security history is typical for an old protocol that evolved from the times when all parties on the network were considered trustworthy to a protocol fit for a global network. The evolution of the grade of randomness of the transaction ID in many popular DNS implementations is a didactic play for security researchers. Obviously the importance of the transaction ID was vastly underestimated and it seems that some implementations were made by developers not aware of security basics, such as buffer overflow prevention techniques, the importance of nonces and cryptographically secure pseudo random number generators, even in very recent times.

As the underdog role of the security-focused DNS implementation djbdns shows, DNS is a field for power struggles, which has hampered security improvements. The lack of consensus about IANA's operation and the signature holder of the root zone is another example for this problem.

There were many severe network security bugs discovered this year: An attacker could choose the password length in SNMP, enabling him to guess as little as one character per password to modify the routers that keep the Internet together, thereby enabling man-in-the-middle-attacks on a whole new level. The popular Linux distribution Debian contained a weakness in OpenSSL, generating only 65000 different private keys, which was only discovered after 18 months of operation - "16 bit RSA/DH" cannot be considered secure. Even SSH's forward secrecy was broken. And finally, Kaminsky's DNSRake attack enables home users to poison any DNS cache in less than 10 seconds.

With network security in general gaining attention because of the high number of security issues this year and the zone walking issue finally fixed, the deployment of DNSSEC on a wide basis might finally take off. However, DNSCurve provides an interesting alternative to DNSSEC which circumvents many of DNSSEC's deployment challenges. Yet still, it can't avoid the root zone signing controversy and further study needs to be made on DNSCurve, with DNSSEC being researched since almost ten years. And finally, IPv6 is waiting to be deployed, which would solve most DNS security issues for good anyways.

No matter which of these alternatives will make it, when DNS finally gets fixed by cryptography after almost 25 years of insecure operation, the focus for both attackers and security researchers will probably shift from design flaws to implementation errors and cryptographic issues.

Bibliography

[1] Dan Bernstein. Notes on *.com wildcards. http://cr.yp.to/djbdns/com-wildcard.html, November 2002.

[2] Dan Bernstein. The libresolv Security Disaster. http://cr.yp.to/djbdns/res-disaster.html, November 2002.

[3] Dan Bernstein. DNSCurve: Usable security for DNS. DNSCurve for DNS software authors. http://www.dnscurve.org/impl.html, 2008.

[4] ccNSO. DNSSEC Survey Results. http://www.ccnso.icann.org/surveys/dnssec-survey-report-2007.pdf, 2007.

[5] D. Atkins and R. Austein of ISC. Threat Analysis of the Domain Name System (RFC 3833). www.ietf.org/rfc/rfc3833.txt, August 2004.

[6] D. Eastlake of DEC, S. Crocker of Cybercash, and J. Schiller of MIT. Randomness Recommendations for Security. RFC 1750, http://www.ietf.org/rfc/rfc1750.txt, December 1994.

[7] Dan Kaminsky of IOActive. Black Ops 2007: Design Reviewing The Web. http://www.doxpara.com/slides/DMK_BO2K7_Web.ppt, August 2007.

[8] Dan Kaminsky of IOActive. It is The End Of The Cache As We Know It (Black Ops 2008). http://www.doxpara.com/DMK_BO2K8.ppt, August 2008.

[9] Kelly Jackson Higgins. 'Provider-in-the-Middle Attacks' Put Major Websites, Users at Risk. http://www.darkreading.com/document.asp?doc_id=151497, April 2008.

[10] J. Postel of ISI. Internet Name Server. IEN 116 ftp://ftp.rfc-editor.org/in-notes/ien/ien116.txt, August 1979.

[11] Collin Jackson, Adam Barth, Andrew Bortz, Weidong Shao, and Dan Boneh of Stanford University. Protecting Browsers from DNS Rebinding Attacks. http://crypto.stanford.edu/dns/dns-rebinding.pdf, 2007.

[12] Gregg Keizer. Microsoft DNS bug long known, familiar to researchers. Computerworld, http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9047638, November 2007.

[13] Amit Klein. BIND 8 DNS Cache Poisoning. http://www.trusteer.com/bind8dns, July-August 2007.

[14] Amit Klein. BIND 9 DNS Cache Poisoning. http://www.trusteer.com/bind9dns, March-June 2007.

[15] Amit Klein. Microsoft Windows DNS Stub Resolver Cache Poisoning. http://www.trusteer.com/files/Microsoft_Windows_resolver_DNS_cache_poisoning.pdf, March-May 2007.

[16] O. Kolkman and R. Gieben of NLnet Labs. DNSSEC Operational Practices. RFC 4641 http://tools.ietf.org/html/rfc4641, September 2006.

[17] Microsoft. Microsoft Security Advisory (935964): Vulnerability in RPC on Windows DNS Server Could Allow Remote Code Execution. http://www.microsoft.com/technet/security/advisory/935964.mspx, April-May 2007.

[18] P. Vixie of ISC and D. Dagon of GaTech. Use of Bit 0x20 in DNS Labels to Improve Transaction Identity. http://tools.ietf.org/id/draft-vixie-dnsext-dns0x20-00.txt, March 2008.

[19] University Of California, San Diego. SD Supercomputer Center Researchers Find Unnecessary Traffic Saturating A Key Internet 'Root' Server. http://www.sciencedaily.com/releases/2003/01/030124074245.htm, January 2003.

[20] Duane Wessels and Haven Hash. 2007 Day In The Life DNS Root Server Analysis. WIDE+CAIDE Workshop No. 8 http://www.caida.org/workshops/wide/0707/slides/hash.pdf, July 2007.

[21] Michal Zalewski. Strange Attractors and TCP/IP Sequence Number Analysis. http://lcamtuf.coredump.cx/oldtcp/tcpseq.html, April 2001.

Eidesstattliche Erklärung:

Hiermit erkläre ich, dass ich die am heutigen Tag eingereichte Master Thesis selbständig verfasst und ausschließlich die angegebenen Quellen verwendet habe.

Bochum, den