Lecture Notes for Web Security 2019

Part 4 — DNS Security

Martin Hell

1 Motivation for DNS

The Domain Name System (DNS) infrastructure provides several services. The main task, however, is to map domain names to IP numbers. While IP numbers can be used just as well as domain names, this mapping has several advantages. First of all, names like www.example.com are much easier to remember than a sequence of numbers, like 208.77.188.166. Another advantage is that if a webserver is moved from one computer to another, maybe even between different countries, the domain name can stay the same even if the IP address of the server is changed. Before the introduction of DNS, computers used hosts files in order to determine the IP address of a specific host. A copy of the hosts file was stored in each computer using the network. When a new host was added, or a computer changed IP address, the host file had to be updated on all computers. As the number of hosts grew this approach had to be abandoned. Instead, the DNS system was invented. DNS is a *distributed* database containing the mappings between IP addresses and domain names. As a result, a computer only needs to know the IP address of a name server and this name server can provide the computer with all other IP addresses.

1.1 The Domain Name

A domain name is written as a series of labels, separated by dots. An example is `server.example.com`. The rightmost label, `com` determines the top level domain and the other labels specify subdomains. Thus, `server` is a subdomain of the `example.com` domain and `example` is a subdomain of the `com` domain. Another terminology used is parent domain and child domain. A domain name can have at most 127 different labels (or levels) and each label can be at most 63 characters. However, the total number of characters allowed in a domain name is 253. The labels are case insensitive, so `example.com` is the same as `EXAMPLE.COM` and any mixing of uppercase and lowercase letters.

There is sometimes confusion between hostnames and domain names. Sometimes a hostname is also a domain name, but sometimes a hostname is not a domain name and vice versa. A hostname is a name of a computer, or another

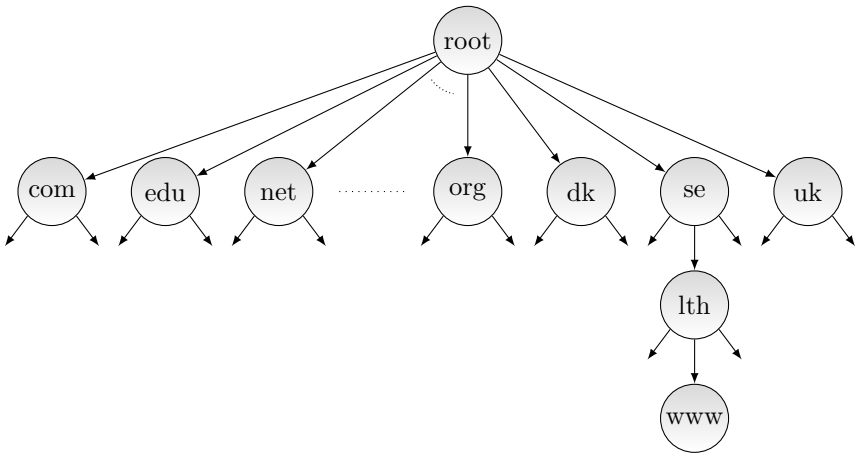

Figure 1: The DNS hierarchy.

device, that is connected to a network. For internal networks this is usually just a simple string, like `server`. A domain name does not necessarily have to correspond to a particular device, but is instead a part of a hierarchy. It can represent a device, but it can also represent a resource, e.g., a public key used in DKIM. It can also be just a separator used to administer a set of computers. As an example, `server.example.com` is a domain name, but it is also a hostname if it has an IP address. Just taking the first part, `server`, can be the hostname on some network, but it is not an Internet hostname. The second part, `example.com` is a domain name, but not necessarily a host name since it may not correspond to a device.

A *Fully Qualified Domain Name* (FQDN) is a domain name that unambiguously specifies a domain name. This means that all labels are specified, including the root label which is empty. Thus, a FQDN always ends with a dot.

2 Overview of DNS

The DNS structure is a hierarchical structure. The specification can be found in RFC 1034 [10] and RFC 1035 [11]. It can be viewed as a tree with a root node, see Figure 1. The complete tree is called the *Domain Name Space*, and each node represents a domain name.

This hierarchy can be compared to a file system, which also has a root and can be viewed as a tree. However, the domain name is written differently from the search path, since it starts with the child nodes and ends with the parent node. Each node in the tree represents a label in the domain name. The tree has only one root node, but in practice there are in total 13 root nodes spread around the world. The number of root DNS servers is more since some root

nodes correspond to many physical servers.

An *authoritative DNS server* is a server that is responsible for a particular domain. This server can in turn delegate the responsibility for subdomains to other DNS servers. These are then authoritative for this subdomain. This delegation can continue in several steps. The part of the domain name space that one server is authoritative for is called a *zone*. This hierarchy relies on the fact that a parent knows the name of the authoritative DNS server for all its children. The DNS server does not only store information about the mapping between IP addresses and hostnames, called A-records. It also stores e.g., the IP address of email servers that are responsible for accepting emails to that domain, called MX-records.

The *Internet Corporation for Assigned Numbers and Names* (ICANN) is authoritative for the root domain. ICANN delegates authority of Top Level Domains TLDs. There are two types of TLDs, namely generic top level domains (gTLD) such as .com, .edu and .org and country code top level domains (ccTLD) such as .se, .uk, .dk and .nu. A special type of gTLDs are the sponsored top level domains (sTLDs) such as .museum and .travel. These can not be registered by anyone, which in general is the case with gTLDs.

An authoritative name server is either a *master* or a *slave*. A master DNS server has local access to the zone file, i.e., the information that it is authoritative for. A slave receives the data from a master server through a *zone transfer* over a network. Both are authoritative for a zone, but the configuration is done only at the primary master. Several possible configurations exists and the best configuration is situation dependent. It is of course possible to have several slaves for one zone for increased redundancy. It is also possible to have several masters, provided that the zone files are kept synchronized in some way. In that case, one is the *primary master*. This is defined to be the server given in the SOA resource record, see Section 4. One server can serve as master for one zone and slave for another zone. For security, the master DNS server can be hidden. By allowing only one slave to know of its existence, this slave can be updated through zone transfers and other slaves can be updated using zone transfers from that slave, regarding it as a master. Then, only slaves are known to the public and security critical information on the master server is more protected. One situation where this can be attractive is when DNSSEC is used.

The terminology primary and secondary DNS server was replaced by master and slave in BIND 8.x, but can still be seen in the literature. Things get more confusing due to the term primary master, which should not be confused with primary as historically used. The term primary master was introduced in RFC 2136 [14].

A resource record is a piece of information stored on the authoritative DNS server. More details on this will be given in Section 4.

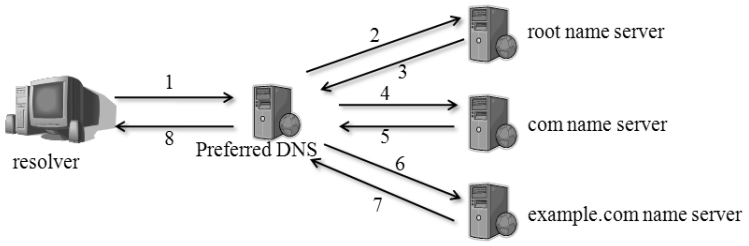

Figure 2: A typical DNS query. The resolver sends a recursive query and the preferred DNS sends iterative queries.

3 Finding the IP Address of a Host

The clients that query the name servers for the resource records are called resolvers. DNS queries uses the UDP protocol on port 53 and a query is one of the following two types.

- Recursive - The resolver asks the DNS server to fully answer the query, or return an error message. If the DNS server does not have the answer in its own database, the DNS server is responsible for finding the answer by e.g., making iterative queries.
- Iterative - The resolver only asks the DNS server for the best answer it can get. Typically, the answer is a reference to a more appropriate DNS server.

Usually, a resolver is implemented in the operating system, sending recursive queries to a preferred DNS server. This preferred DNS server then sends iterative queries to other DNS servers. The IP address of a specific hostname is achieved by systematically traversing the hierarchical structure of the DNS. As an example, assume that the webpage `www.example.com` is typed into a web browser. Then the following happens, which is illustrated in Figure 2.

1. The resolver sends a recursive query for `www.example.com` to the preferred DNS server. This is usually the corporate DNS server or the ISPs DNS server.
2. Since the preferred DNS server is not authoritative for the zone to which `www.example.com` belongs, it sends an iterative query asking for the domain `www.example.com` to a root name server.
3. The root name server does probably not know the answer, but it knows the name servers authoritative for the `com` zone, so it will refer to those name servers.
4. The DNS server will choose one of the referred name servers and again send an iterative query for `www.example.com` to this server.

5. The `com` name server does probably not know the answer and will refer to the name servers authoritative for the `example.com` zone.

6. The DNS server will choose one of the referred name servers and again send an iterative query for `www.example.com` to the name server authoritative for `example.com`.

7. This name server knows the IP address of `www.example.com` and will return this address to the preferred DNS server.

8. The preferred DNS server returns the IP address of `www.example.com` to the resolver.

The above example shows how the tree can be traversed in order to find the desired information. However, if this procedure was always done, it would render heavy traffic to and from the root servers. Instead, answers are cached by the querying DNS server. Using the cache, the DNS server starts the iterative queries at the name server of the longest match in the cache. If the hostname `www.example.com` has been queried before and the IP of the host `www2.example.com` is later queried by the resolver, then the DNS server will immediately ask the DNS server authoritative for `example.com` since the address of this DNS server is in the cache. If the resolver queries for `www.server.com`, the DNS server will immediately go to the `com` server and continue from there. The resolver can also have its own cache and, for efficiency, the applications using the resolver might also implement their own DNS cache. Some implementations support negative caching as well, i.e., if a domain does not exist the server can remember this so that the next time the same query is made the server does not have to repeat its failed attempt.

An answer includes a time-to-live (TTL) value. This determines for how long time an answer should stay in the cache. Normal time is about 24 hours, but it can be set to several days if you know that your IP will not change for that amount of time. If large websites set the TTL too short, it will result in an increase of network traffic to root and top level domain servers.

If the response is larger than 512 bytes, it is truncated. In that case a new query is made but instead of UDP, a TCP connection is used. In order to support new functionality for DNS, such as DNSSEC, an extension mechanism for DNS (EDNS) was proposed in RFC 2671 [12]. Using EDNS much larger responses are supported also over UDP.

4 Zone Files and Resource Records

The zone file contains resource records for a zone and are located in the authoritative name servers. The format of this file is standardized and not implementation dependent. The format is defined in RFC 1035. A resource record is a piece of information describing a property of the zone. There are many different types of records. Some of them are listed below.

- **SOA** - This is the start of authority record. It defines global parameters for the zone, such as a serial number, the minimum TTL and the email address to the person responsible for the zone. There can only be one SOA record for a zone.
- **A** - This is a 32 bit IP address (IPv4) and is used to map hostnames to IP addresses. It is also used in DNSBL to determine if an IP address is listed as sending spam.
- **NS** - This record defines a name server for a domain. An NS record for a domain exists in the domain zone file itself, but it must also appear in the parent zone. The `com` name server must store NS records for `example.com`, i.e., records that points to the name servers authoritative for the `example.com` domain. For redundancy and robustness, a public domain must have at least two name servers.
- **CNAME** - The canonical name for an alias. Several names can be used to point to the same IP. If users should be able to enter both `example.com` and `www.example.com` in the browser and end up at the same place, one can be an alias for the other.
- **MX** - This record points to the server that is responsible for receiving emails to the domain.
- **PTR** - This is a pointer record that is used to find the hostname for an IP address.

Other records that will be covered later are RRSIG, DNSKEY, NSEC, but there are many more. The TXT record is e.g., used to return keys in DKIM and the IPv6 version of A records are called AAAA records, reflecting the fact that IPv6 addresses are 128 bits instead of 32 bits.

When delegating authority for a zone, an NS record pointing to a name server for that zone is given. If the domain name for that name server is in the zone itself, a corresponding A record for this domain name must also be provided. These are called *glue records*.

A resource record consists of several fields separated by white space. The fields are

```
domain-name  TTL  class  type  data
```

The domain-name is the domain that the resource record applies to. If it does not end with a dot, the zone origin is appended to the name. The TTL is the time that the record can be cached. The class for internet is IN. The domain-name, TTL and class can be left empty. In that case they will default to the previous explicitly stated values. The type of the resource record is any of A, NS, MX, TXT, etc, and the data depends on the type. For an A record, the data is an IP address, for an NS record the data is a domain name, for an MX record the data is a priority number and a domain name for a mail server, for

```
$ORIGIN example.com.     ; designates the start of this zone file in the name space
$TTL 1d                  ; default TTL
example.com.  IN  SOA  ns.example.com. username.example.com. (
              2007120710 ; serial number of this zone file
              1d         ; slave refresh (1 day)
              2h         ; slave retry time in case of a problem (2 hours)
              4w         ; slave expiration time (4 weeks)
              1h         ; minimum caching time in case of failed lookups (1 hour)
              )

example.com.  NS    ns                    ; a nameserver for example.com
              NS    ns.server.se.         ; a backup nameserver for example.com
example.com.  MX    10 mail.example.com.  ; primary mail server for example.com
              MX    20 mail2.example.com. ; backup mail server for example.com
example.com.  A     10.0.0.1              ; IPv4 address for example.com
              A     10.0.0.2              ; Another IPv4 address for load balancing
ns            A     10.0.0.3              ; IPv4 address for ns.example.com
mail          A     10.0.0.4              ; IPv4 address for mail.example.com,
mail2         A     10.0.0.5              ; IPv4 address for mail2.example.com
www           CNAME example.com.          ; www.example.com is an alias for example.com

; Delegate authority of the sub.example.com domain
sub           NS    ns1.sub.example.com.   ; Name server authoritative for the domain
              NS    ns2.sub.example.com.   ; backup name server
ns1.sub       A     10.10.0.20             ; glue record
ns2.sub       A     10.10.0.20             ; glue record
```

Figure 3: Example of a zone file.

a TXT record the data is any text string. A semicolon marks the start of a comment. Parentheses can be used to group data over several lines.

An example of a zone file is given in Figure 3.

5 Reverse DNS Lookup

In some cases it is useful to find the corresponding FQDN for a given IP address. Often, this is used for troubleshooting, but it is also used in the "Received" header of emails. Looking at the tree in Figure 1 it looks difficult to accomplish this since it would correspond to searching the whole tree. Instead, this possibility is realized by defining a domain, `in-addr.arpa`, dedicated to this. (IPv6 uses the domain `ipv6.arpa`.) The subtree of this domain corresponds to the IP addresses, i.e., `in-addr.arpa` has 256 children and each of these has 256 children etc. If the domain name for IP address 1.2.3.4 is queried, the query is made for `4.3.2.1.in-addr.arpa` setting the type to PTR. Note that the bytes in the IP address are reversed. The reason for this is that the right bytes in an IP address are more specific than the left, the opposite of the case in a domain name. The ISP that is in charge of a range of IP addresses is responsible for the PTR record of that address, or for delegating this responsibility to another DNS server. Historically IP addresses were assigned in classes. Class A, B and C networks consists of 2^{24}, 2^{16} and 2^8 IP addresses respectively, with the leftmost

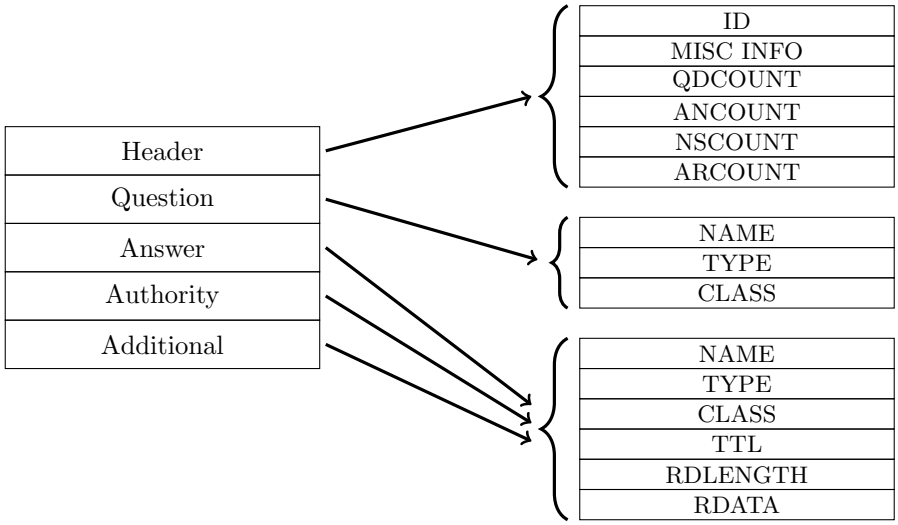

Figure 4: DNS Messages

byte(s) being fixed. This allows for easy reverse DNS lookup. Today, addresses can be assigned in other ways, using a subnet mask to identify the number of bits that are fixed in the assigned range. RFC 2317 [6] specifies how reverse DNS lookup should be used for these addresses.

6 Queries and Responses

In order to fully understand the attacks on the DNS system, knowledge about the information included in the queries and the corresponding responses is needed. A DNS message consists of five sections, see Figure 4.

The first section is the header. This is included in all DNS messages. The next section is the actual question followed by three sections containing the resource records returned in a response. Each of these three sections contains a possibly empty list of resource records. The answer section contain records answering the question, the authority section contain records that point toward an authoritative name server and the additional section contain other records which are related to the query but does not immediately answer the question.

The header section starts with a 16-bit transaction ID. The ID authenticates a response as the ID used in the response must be the same ID that was used in the corresponding query. The fields denoted "misc info" in Figure 4 contain information about the message, e.g., if the message is a query or a response, if the message has been truncated since it is to large, and if recursion is desired/available. There is also a field indicating if the response is authoritative i.e., if the responding server is authoritative for the domain name in the question section. The QDCOUNT field specifies the number of entries in the

question section. The ANCOUNT, NSCOUNT and ARCOUNT fields denote the number of resource records returned in the answer, authority and additional sections respectively.

The question section contain a list of entries, which represent the information that is being asked for. Each entry consists of a domain name, which type of resource record is being asked for (e.g., A, NS or MX), and the class of the query (e.g., IN for Internet).

The answer, authority and additional section all share the same format, i.e., they consist of a list of resource records. The information in the resource records are the same as the information given in the zone file, see section 4. The RDLENGTH field is not present in the zone file and specifies the number of bytes in the RDATA field. This is needed as the RDATA field is of variable length.

7 Attacks on DNS

The domain name system is very important to the Internet. If the root servers would go down, Internet would practically stop working. Historically, there have been two documented attempts to take down the root servers.

The first attempt was on October 21, 2002. A distributed denial of service (DDOS) attack was launched targeting all 13 root servers. This attack lasted for about one hour and some of the servers became unreachable. However, the DNS system showed robustness against these types of attacks and the attack did not seem to impact the end-users in any way [13].

The second attack was launched on February 6, 2007. It lasted for about 24 hours and six root servers were affected. However, also in this case the system proved to be robust, and in particular the anycast technology used to distribute one server over several locations proved effective [7].

The attacks that are covered in this section are DNS amplification attacks and DNS cache poisoning attacks, but it can be noted that DNS implementations have been shown to suffer from weaknesses too, allowing e.g., buffer overflow attacks on the server.

7.1 DNS Amplification

DNS uses the connectionless UDP protocol as the default transmission protocol for queries and responses. It is very difficult to spoof an IP address if the TCP protocol is used. The reason is the three-way handshake, in which the client needs to successfully guess the sequence number to be acknowledged in the third packet. If sequence numbers are random enough (have enough entropy), then the client will not be able to guess it. The only way to return the correct sequence number is to see which number is sent by the server, and that can only be accomplished if you give the server the correct IP address. With UDP, there is no handshake. Messages, in this case DNS queries, are sent to the DNS server. These messages, like all IP packets, have a source IP address and this

address is used by the server when the response is sent. Thus, spoofing the IP address in UDP is very easy.

Spoofing the source address in a DNS query is the idea behind the DNS amplification attack. The amplification in the attack stems from the fact that the query is included in the response so a response is usually much larger than a query. A small query can be around 60 bytes while a response can be up to 512 bytes. The amplification in this case is 8.5 times. This can be used to amplify DOS attacks. To mount a DOS attack on a victim, instead of sending as much packets as possible directly to that host, packets are sent to DNS servers instead. By claiming that the source IP of the DNS queries is the IP address of the victim, the DNS will send the response to the victim. This response packet can be about 8.5 times larger than would be possible if the packets were sent to the victim directly.

The extension mechanisms (EDNS) allow response packets to be much larger than 512 bytes, still using UDP. It is possible for the client to tell the server that EDNS should be used. In this case the amplification can be around 60 times instead of 8.5. Combining this with a botnet in which every zombie sends queries to a DNS server, this attack can bring down most victims.

7.2 DNS Cache Poisoning

A DNS cache poisoning attack aims to add or change DNS records on servers so that the wrong answer will be sent to clients. If the attacker can control the IP addresses sent as response to queries, then traffic can be directed to an attacker's computer. There are many different ways to mount this attack and some of the most powerful are based on implementation flaws in DNS servers.

One simple variant is to send false responses to queries. Assume that an attacker is controlling the authoritative DNS for the domain `attacker.com`. Other DNS servers querying IP addresses for hosts in this domain will get the A records for these hosts. Additionally, the DNS can send other A records in the response, e.g., an A record stating that the domain `www.bank.com` has some particular IP. When the DNS server is later queried for WWW.BANK.COM, this record will be in the cache and the IP address chosen by the attacker is returned to the resolver. The protection against this attack is to only accept records that belong to the domain that was actually queried. If `www.attacker.com` was queried, records for other domains, such as `www.bank.com`, should not be accepted.

Since UDP and not TCP is used for DNS queries and responses, the IP address in the data packet is easy to spoof. If TCP is used, the 32-bit sequence numbers prevent this type of spoofing, provided that the sequence numbers are chosen randomly at the start of the session. A similar protection is used for DNS packets, but instead of using sequence numbers, each query has a 16-bit transaction ID. This ID is used to authenticate the DNS response. A response will be accepted if the following holds:

1. The question section is the same in the response as in the query.

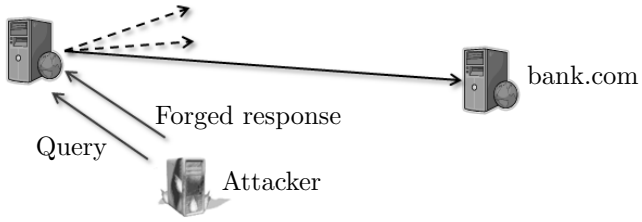

Figure 5: A DNS cache poisoning attack.

2. The transaction ID matches the ID in the question.
3. The response comes from the same IP as the query was sent to.
4. The response comes to the same port as the query was sent from.

Consider the following attack, visualized in Figure 5. An attacker sends a recursive query for `www.bank.com` to a DNS server. The DNS server starts sending iterative queries to a root DNS server, the com DNS server and finally to the bank.com DNS server. Before the reply is sent from the last server, the attacker sends a forged response from this server, answering with an IP address of a computer she controls. This is also called DNS spoofing or DNS forgery. Looking at the requirements needed for the answer to be accepted, only the transaction ID and the port are not immediately known to the attacker. Assuming that the port number can easily be guessed by e.g., looking at the source code of the DNS server and see how it is chosen, only the 16-bit transaction ID is unknown. Thus, the reply is authorized using only a 16-bit number. Comparing this to the 32-bit sequence number used in TCP, we can immediately conclude that the spoofing protection of DNS replies is not as good as plain TCP connections.

Similar to the case with TCP sequence numbers, if the transaction ID is not randomly chosen, this spoofing attack will be significantly easier. Early versions of the BIND DNS server, the most commonly used DNS server, assigned transaction IDs sequentially, making it very easy to predict transaction IDs and successfully perform a DNS cache poisoning attack. Even if transaction IDs are not sequential it is possible that the algorithm used to generate them does not generate them randomly. This will increase the probability of correct guesses. If the transaction ID is completely unpredictable there is still a 2^{-16} probability that a guess will be correct.

If the DNS server that is being attacked will send out multiple queries for the same IP address it is possible to use the birthday paradox to improve the success probability of the attack. By sending 300 queries and 300 replies with different transaction ID to the DNS server, the probability that there is a collision in the transaction IDs is very high, since $\sqrt{2^{16}} = 256$. For this attack to be successful, the DNS server must treat all queries independently and send out 300 queries for the same IP address. Moreover, no real answer is allowed to arrive from the spoofed DNS before all queries are sent to the spoofed DNS. This might be difficult to achieve, but the attacker can buy some time by mounting a DDOS

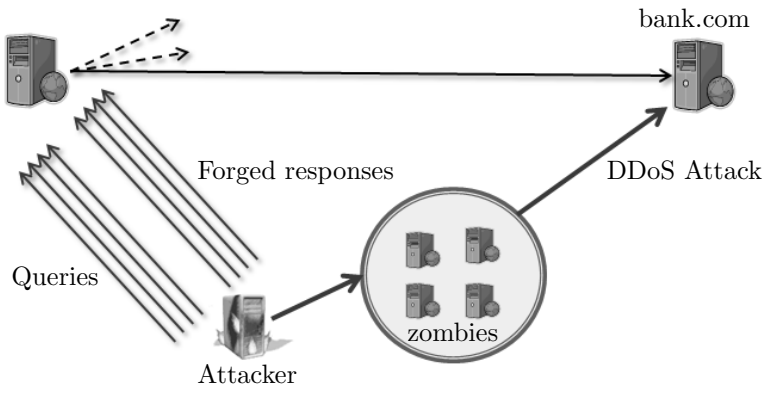

Figure 6: A DNS cache poisoning attack using multiple queries and responses.

attack on the spoofed server, using a large number of computers. This attack is visualized in Figure 6. One way to prevent this kind of attack is to not allow several outstanding requests for the same domain. If the birthday, or collision, attack is not possible, sending out about 2^{16} queries and forged replies will still result in a successful attack. (Actually, as many replies as possible should be sent before the real one arrives). Unfortunately for the attacker, if the first attack attempt fails, then the correct IP address will be in the DNS server's cache, so it is not possible to try another attack attempt until the cached value has expired. This time will depend on the TTL of the answer. In 2008, an attack was published, see [4], that avoids this problem. Instead of sending requests and replies for `www.bank.com`, new random subdomains of `www.bank.com` was tested for each attack attempt, e.g., first `a.www.bank.com` was tested, then `b.www.bank.com` and so on. Since these will not be found in the cache, the DNS server authoritative for `www.bank.com` was queried every time. The forged replies sent the A record for the queried domain but it also included an A record for `www.bank.com`. Since they both belong to the same domain, both records are accepted. This attack was shown to be successful in a matter of seconds. The solution to this problem was to randomize also the source port used in the query. (Some implementations already did this and were not vulnerable.)

If additionally the port number used in the query is random, another 16 bits of uncertainty are added to the authorization requirements. This will make the attack much more difficult. If the birthday attack is possible, instead of 300 queries and replies, about 100000 will be needed. Still, even if this attack is prevented by not allowing multiple outstanding requests, it is still possible to perform a successful attack by testing in the order of 2^{32} queries and replies. This will take a long time, but the vulnerability still exists. Of course, if the query sent to the spoofed DNS server can be sniffed, the attack will work with only one query and reply.

7.2.1 Consequences of DNS Cache Poisoning Attacks

DNS cache poisoning allows pharming attacks. A pharming attack is an attack that redirects users from one webpage to another, without the victim being aware of it. An attacker can design a webpage that looks identical to another, e.g., `www.example.com`. By mounting a DNS cache poisoning attack users can be directed to the fake webpage, thinking that it is the real one. Some consequences of this can be:

- *Identity theft* - If `www.example.com` is a site with login functionality, users can be tricked to submit username and password to the attacker.
- *Distribution of false information* - If `www.example.com` is a news site or a company webpage, false information can be distributed through the fake site.
- *Distribute malware* - If `www.example.com` contain downloadable programs, users can download programs that are in fact some type of malware.

Another option is to mount a *man-in-the-middle attack*. By directing users that want to visit `www.example.com` to the attackers own computer, the attacker can then forward the data to `www.example.com`. This will allow the attacker to delete, add and modify packets at her own will before forwarding them. Yet another option is to redirect users to a page with lots of ads.

7.3 DNS Rebinding Attacks

The same-origin policy says that an HTTPHtmlRequest can not make requests for documents with a different origin than the document itself, i.e., a document with different protocol, domain or port. This check is implemented in the user-agent. However, actual requests are made to IP addresses and the DNS is in charge of the mapping between IP addresses and domains. An authoritative DNS can claim that any IP belongs to the domain by just putting the IP in an A record in the zone file. If one domain has several A records, the user-agent will accept that all IP addresses belong to the same origin. These properties are exploited in the DNS rebinding attack.

Flash Player and Java applets are common targets for this vulnerability, but JavaScript in the browser can also be used and vulnerabilities in both Silverlight and Adobe Acrobat can be used to perform the attack. The idea and method of the attack is very similar in all cases. The description below focuses on the JavaScript variant. For a more in-depth description of various aspects of DNS rebinding attacks, see [8].

The main idea is that the attacker provides two A records for a queried domain name. One record is for the attackers own web page, and the other is for another domain (origin). In the original attack [5], Java applets were targeted. Applets are only allowed to make connections to the origin that the applet was loaded from, but by providing two A records in a DNS response, one to download the applet from and one to connect to, it was possible to connect

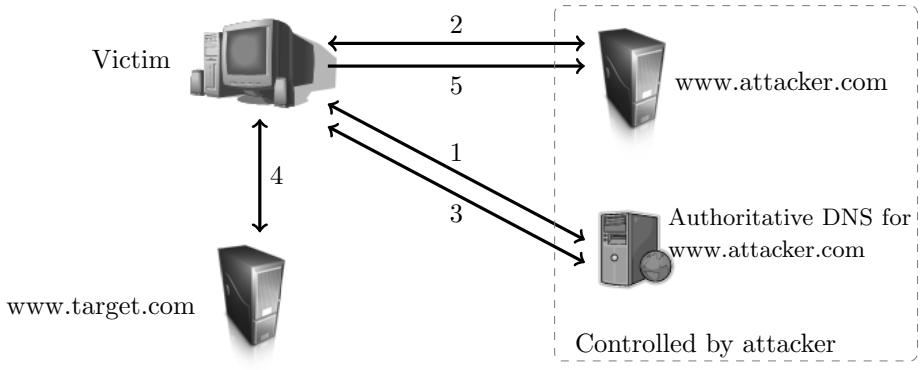

Figure 7: DNS rebinding attack.

to any computer. This problem was easily fixed by only allowing the applet to connect to the same IP as it was loaded from.

Current attacks instead use the XMLHttpRequest object to make requests to documents or resources with different origin. Using a very short value for the TTL in the DNS record the steps in the attacks can be summarized as follows, see Figure 7.

1. A user wishes to visit www.attacker.com and queries the name server authoritative for the domain. The attacker controls this name server and responds with an A record pointing to the IP of www.attacker.com. The TTL of the record is set to zero or just a few seconds.

2. The user downloads the web page which contains a JavaScript with an XMLHttpRequest calling www.attacker.com. This is as expected and complies with the same-origin policy.

3. As the record for www.attacker.com has expired, another query has to be made. This time the attacker responds with another A record, pointing to a target server.

4. The XMLHttpRequest is made to this new IP, and the user-agent consider the two IPs to belong to the same origin. Instead, this IP might actually belong to www.target.com. The response is received from www.target.com and can be read by the JavaScript.

5. Information can be sent back to the attacker using e.g., an HTML form and POST.

The attack above tricks the user-agent to think that a machine has moved to another IP address, while in fact, the DNS is just answering with a fake IP address immediately after it has answered with the true IP address. To protect against this attack, *DNS pinning* is implemented in most browsers. The

browser keeps the mapping between a host name and IP address in an internal pin database, regardless of the TTL. The exact time can be up to 30 minutes depending on the browser. Within this time, no additional DNS queries for that host name are made.

The challenge in current DNS rebinding attacks is to circumvent the protection given by DNS pinning. Thus, the name *Anti-DNS pinning* is sometimes also used in relation to these attacks and sometimes the attacks are called *Anti-DNS Pinning Rebinding Attacks*. It has been shown that, in some circumstances, the browser can be forced to drop the info in the pin database before the pinning time has expired. In [9] it was shown that if the attacker's web server was disconnected after the victim has retrieved the document with the JavaScript, then the connection back to www.attacker.com will fail. This causes the browser to drop the DNS pinning and make a new DNS query. Meanwhile, the attacker changes the A record in his DNS to point to the target. Instead of disconnecting the web server, a dynamic firewall rule can be used.

Another problem is plug-ins for browsers, e.g., Flash Player and Java. These, and other, plug-ins use separate pin databases so it can be possible to pin the browser to one IP address while the plug-in is pinned to another, opening up possibilities for attacks.

7.3.1 Consequences of DNS Rebinding Attacks

Making the user-agent connect to a different server than allowed by the same-origin policy might at first not be seen as a serious problem. After all, the attacker can just connect to that server himself if he want access to it. However, the important difference is that the connection made from the victim's IP and not the attacker's. This allows for two important types of attacks, namely *firewall circumvention* and *IP hijacking*.

If the target server and the victim is behind a firewall, separating e.g., an internal network from the Internet, then the attacker has typically no access to the server. If the requests are instead made from the victim, the attacker can circumvent the firewall using the DNS rebinding attack. The JavaScript can be used to crawl the entire intranet by following links and then send the information back to the attacker. Moreover, computers on an intranet might not be updated and patched as often as Internet-facing computers. This can allow the attacker to exploit known vulnerabilities on computers on the internal network. Also the victim machine itself can be subject to attacks.

In IP hijacking the attacker uses the victim's IP to send packets to servers. Connections to servers can also be set up, allowing the attacker to fully communicate with a server using the victim's IP address. If an attacker hosts a website with advertisements, this attack can be used to let all users click on an ad. Also, the victim's computer can be seen as a proxy and if the attacker performs illegal actions on other computers, the victim's IP address will be seen in logs. Sending spam email and bypass IP based authentication are other possible consequences of this attack.

7.3.2 Protection Against DNS Rebinding Attacks

The protection provided by DNS pinning works in the basic scenario, but can sometimes be bypassed. One very effective defence against this attack is to have web servers check the host header. In HTTP 1.1, this header is required and can be used to separate virtual hosts residing on the same IP. In the first connection made to www.attacker.com, that host name is used in the host header. When the second connection is made, the user-agent thinks that it is still connecting to www.attacker.com but using a new IP address. Thus, the same host header is used. However, the actual host is www.target.com. This discrepancy can be used by the web server at www.target.com and requests can be ignored if it detects that the wrong host is used.

The fact that the host header is wrong also has the consequence that the attack does not work if www.target.com is a virtual host on a web server.

8 DNSSEC

DNS cache poisoning is a problem that has been patched several times, but the underlying problem remains, namely that there is no way to guarantee that the answer comes from the claimed sender. The solution to this problem requires cryptographic algorithms. DNS Security Extensions (DNSSEC) is an attempt to solve the problem. It adds source authentication protection to the resource records by using digital signatures. DNSSEC is specified in RFC 4033 [1], RFC 4034 [3] and RFC 4035 [2].

The digital signature will add integrity and origin authentication to the response sent by the authoritative DNS server, since only the holder of the private key can sign data. By checking the digital signature it is possible to verify that the data received is identical to the data that is stored in the authoritative DNS server. Note that the data is not encrypted, only signed. DNSSEC adds new resource records to DNS, namely the *DNS public key* (DNSKEY), the *Resource Record Signature* (RRSIG), the *Next Secure* (NSEC) and the *Delegation Signer* (DS) records. These resource records are specified in detail in RFC 4034.

Every zone has its own private/public key pair. Each resource record set (RRset) is digitally signed using the private key. An RRset is a set of all resource records that have the same name, same class and same type. The public key, together with the algorithm that is used to sign the records, is stored in the DNSKEY record and the signature for a RRset is stored in the RRSIG record. Every response to a DNS query will include a corresponding RRSIG record and the DNSKEY.

8.1 Authenticated Denial of Existence

DNSSEC does not only support authentication of existing records, it also supports authenticated negative responses, i.e., responses saying that a certain record does not exist. This will defeat denial of service attacks when the attacker injects a packet stating that a domain name or other resource records

do not exist. This is accomplished using the NSEC record type. All domain names in a zone are sorted according to a canonical ordering. For DNS names this is alphabetical ordering according to first the most significant label, then the next most significant and so on. Then for each domain a record is signed that includes this domain and the next domain, together with all record types that exist for the domain. Using this record it can be verified that there is not a domain name that is in between the two names in the NSEC record, and it will also verify which records that do exist for the domain. As an example, assume that a zone includes the domains `alfa.example.com` and `gamma.example.com`. Then the NSEC record for alfa.example.com, will contain `alfa.example.com`, `gamma.example.com` and all record types that exist for the domain `alfa.example.com`. If DNS server sends a query for `beta.example.com`, the NSEC record for `alfa.example.com` is returned together with a signature and the public key. This allows the querying DNS server to verify that `beta.example.com` does not exist.

8.2 Verifying the Public Key

Having a resource record, the signature of the record (RRSIG) and the public key (DNSKEY), the signature can be verified. However, nothing is achieved if we cannot verify that the public key actually belongs to the queried DNS server. Anyone can create a private/public key pair, sign a resource record with the private key and send the record together with a valid signature and the corresponding public key to a DNS server. Integrity and origin authentication of the data can only be verified if we know that the public key belongs to the zone. Typically this is achieved using certificates and the approach taken in DNSSEC is similar. The public key for a zone is signed by the parent zone. More specifically, the DS record type contains the hash of the public key and is stored in the parent zone together with a corresponding RRsig record. This will achieve the same functionality as that of certificates. As an example, to verify the public key of `example.com`, the DNS server authoritative for the `com` domain is asked for the DS record and the RRSIG corresponding to this key. Also the public key of the `com` zone is queried. Using this public key the signature in the RRsig record can be used to verify that the hash value stored in the DS record is the hash value of the public key of `example.com`. By comparing this hash value with the hash of the public key received from `example.com`, the authenticity of this key is verified. Thus, as long as we trust the public key of the com zone, we trust the public key of example.com. Similar to digital root certificates, when a trusted public key is found, the verification is done. This trusted public key is called a *trust anchor*. Ideally, this would be the public key of the root domain, but it can in general be a public key in any subdomain.

8.3 Drawbacks

Many people consider DNSSEC to be a very important step in order to make Internet more secure. It will no longer be possible to send forged DNS replies. It

would make it possible to store important data in DNS servers. SPF records and DKIM public keys are stored in DNS records and the trust in the authenticity of these records can be increased. However, DNSSEC also has some drawbacks.

- Response messages will be significantly larger since RRSIG records must also be sent.
- Everytime something is changed in a RRset, the server needs to recompute the signatures.
- The zone file will grow due to all RRSIG and NSEC records.
- DNS lookups will be slower since a signature needs to be verified.

While the problem with DNS cache poisoning can be combated using the signed responses in DNSSEC, the problem with DNS amplification attack is increased. Since the size of the response messages are increased, the DNS servers are even more suitable for this kind of denial of service attacks.

Exercises

Exercise 401 *Would it be possible to have a centralized database for DNS instead of a distributed? What would be the consequences?*

Exercise 402 *When signing up with an ISP we also get a DNS to use. The resolver in the operating system then sends recursive queries to this DNS, which in turn make iterative queries to authoritative DNS servers. Another approach would be to let the resolver in the OS make all the iterative queries. Then we would not need this extra DNS server provided by our ISP. What would the consequences of such an approach be?*

Exercise 403 *The* `.se` *top-level domain has delegated the administration of the* `lu.se` *domain to LU. Thus, all DNS records belonging to the* `lu.se` *domain are adminstered by an LU name server. If we ask a* `.se` *name server for the IP address of* `www.lu.se`*, it will answer with a response telling us that the IP is not known but we can instead ask the name server* `lundns.lu.se`*. (This is an NS record stored in the* `.se` *name server telling us who is authoritative for* `lu.se`*.) In order to ask* `lundns.lu.se` *we need to find the IP of* `lundns.lu.se`*, but that we cannot do since we do not have the IP to the name server authoritative for* `lu.se` *(which is* `lundns.lu.se`*). How is this catch-22 problem solved?*

Exercise 404 *An advantage of DNS is that a server can change IP address but still keep the same host name. The A record for that name in the DNS just has to be changed so that it gives the new IP. One problem might be that the A record has been cached by resolvers and/or applications. In that case the new IP will not be retrieved, but the old IP will be taken from the cache. How would you deal with this problem?*

Exercise 405 *To protect against a DNS cache poisoning attack both the transaction ID and the port number should be random. The port number cannot be completely random since many ports are predefined to be used for certain protocols, but let us for simplicity assume that all ports can be used by the DNS resolver. The port and transaction ID are chosen from a uniform distribution.*

a) Describe a DNS cache poisoning attack that takes advantage of the birthday paradox.

b) How many queries and responses must be sent according to the birthday paradox to make the attack succeed with high probability?

Exercise 406 *In February 2008, a Danish court ordered Tele2, a Danish Internet Service Provider, to block all access to the popular website "The Pirate Bay". The website was blocked by using a DNS redirect. This means that, when the client tries to look up the IP to the website, the DNS server owned by Tele2 (the users' preferred DNS) responds with another name.*

a) Give a few different ways to easily bypass this block.

b) It is easy to conclude that this kind of web site blocking is rather useless. The implementation of the blocking directed users to another website with information that the pirate bay was blocked. Compare this to the DNS cache poisoning attack. What similarities and differences can you see?

As a side note, it was claimed that the number of visits to the pirate bay from Denmark increased by 12% during the first few days after the block. The number of visits from Tele2 was unchanged.

Exercise 407 *In DNSSEC, the resource record RRSIG contains a signature of one or a few resource records. Explain how this signature is verified.*

Exercise 408 *What is the purpose of the NSEC records in DNSSEC?*

Exercise 409 *State some advantages and drawbacks of DNSSEC.*

References

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005. Updated by RFC 6014, Available at: *http://www.ietf.org/rfc/rfc4033.txt*.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard), March 2005. Updated by RFCs 4470, 6014, Available at: *http://www.ietf.org/rfc/rfc4035.txt*.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard), March 2005. Updated by RFCs 4470, 6014, Available at: *http://www.ietf.org/rfc/rfc4034.txt*.

[4] CERT. Cert vulnerability note vu#800113: Multiple dns implementations vulnerable to cache poisoning. Available at: *http://www.kb.cert.org/vuls/id/800113*, 2008.

[5] D. Dean, E. W. Felten, and D. S. Wallach. Java security: From hotjava to netscape and beyond. In *IEEE Symposium on Security and Privacy*, 1996.

[6] H. Eidnes, G. de Groot, and P. Vixie. Classless IN-ADDR.ARPA delegation. RFC 2317 (Best Current Practice), March 1998. Available at: *http://www.ietf.org/rfc/rfc2317.txt*.

[7] ICANN. Factsheet - root server attack on 6 february 2007. Available at: *http://www.icann.org/en/announcements/factsheet-dns-attack-08mar07.pdf*, 2007.

[8] C. Jackson, A. Barth, A. Bortz, W. Shao, and D. Boneh. Protecting browsers from dns rebinding attacks. *ACM Trans. Web*, 3:2:1–2:26, January 2009.

[9] M. Johns. (somewhat) breaking the same-origin policy by undermining dns-pinning. Available at: *http://seclists.org/bugtraq/2006/Aug/0290.html*, 2006.

[10] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, Available at: *http://www.ietf.org/rfc/rfc1034.txt*.

[11] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, Available at: *http://www.ietf.org/rfc/rfc1035.txt*.

[12] P. Vixie. Extension Mechanisms for DNS (EDNS0). RFC 2671 (Proposed Standard), August 1999. Available at: *http://www.ietf.org/rfc/rfc2671.txt*.

[13] P. Vixie, G. Sneeringer, and M. Schleifer. Events of 21-oct-2002. Available at: *http://c.root-servers.org/october21.txt*, 2002.

[14] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136 (Proposed Standard), April 1997. Updated by RFCs 3007, 4035, 4033, 4034, Available at: *http://www.ietf.org/rfc/rfc2136.txt*.