DNS security: poisoning, attacks and mitigation

The Domain Name Service underpins our use of the Internet, but it has been proven to be flawed and open to attack. **Richard Agar** and **Kenneth Paterson** explain the problem and outline ways in which it could be fixed.

Royal Holloway University of London

SearchSecurity.co.UK

1. INTRODUCTION TO THE DOMAIN NAME SYSTEM (DNS)

COMPUTERS COMMUNICATE over the Internet using Internet Protocol (IP) addresses, such as *134.219.204.35*; they find this easy. As humans we do not; we find it easier to remember names, such as *rhul.ac.uk*. The DNS provides the mapping between the names that we use to identify applications, websites, e-mail recipients, etc and the IP addresses that are used by the components in IP networks. An analogy that is often used for the DNS is that it is the phonebook for the Internet.

The DNS is a distributed, scalable database of millions of servers spread across the globe that enables the Internet to operate as it does today. Websites, e-mail, instant messaging, distributed applications, e-commerce and many more applications depend on the DNS to be able to communicate.

In our daily lives many of us have become dependent on these applications, and are therefore dependent on the DNS. It is not an overstatement to say that if the DNS were to fail, then the Internet would cease to operate as it does today.

To illustrate how the DNS works, let's look at an example - accessing a website.

1.1. ACCESSING A WEBSITE

Bob wants to view Alice's website, which has the domain name www.alice.example.com. To access the website Bob will type *www.alice.example.com* into his browser; the browser will use the DNS to obtain the IP address for Alice's website. An approximation of how this works is shown below:

- The browser calls the resolving function of the Operating System, which sends a DNS query for www.alice.example.com to a DNS server.
- The DNS server may know the answer already; if it does it will respond to the query, if it does not it will start sending queries to other DNS servers until it gets a response.
- The DNS server will know about the DNS root servers (servers that hold the records for all of the top level domains such as com, net, uk etc); it selects one of them and sends a query for www.alice.example.com. The root server may know the answer already; if it does it will respond

to the query, if it does not it will suggest that the DNS sever asks the com servers and will supply their addresses.

- The DNS server will ask one of the com servers for *www.alice.example.com*. The com server may know the answer already; if it does it will respond to the query, if it does not it will suggest that the DNS sever asks the example.com servers and supply their addresses.
- The DNS server will ask one of the example.com servers for *www.alice.example.com*. The example.com server is authoritative (it holds the definitive records); it will therefore respond to the query with a valid answer.
- The DNS server can now respond to the original query from the browser's Operating System.
- The browser can set up a session to the *www.alice.example.com* website.

This process involves sending and responding to a number of queries. If the DNS always operated in this way it would be inefficient; to improve efficiency DNS servers use caching (keeping the records in memory). DNS responses also include a Time-To-Live (TTL) field, which tells the querying server how long to cache records before discarding them. This process distributes information in the DNS, improving efficiency and availability.

2. DNS POISONING

If an attacker can poison the DNS (introduce invalid information) then the user (relying party) may unknowingly connect to the attacker's service, rather than the correct one. The user may then be exposed to confidentiality, integrity and availability issues.

If an attacker can poison the DNS (introduce invalid information) then the user (relying party) may unknowingly connect to the attacker's service, rather than the correct one.

If an attacker wishes to poison the DNS he must somehow trick the relying party's DNS server into accepting a fake response. DNS servers perform the following checks to validate a response:

1. The IP address and UDP ports of the response must match those of the query.
2. The transaction ID (a 16 bit number, giving

65,536 possible values) of the response must match that of the query.

3. The query section and any additional information supplied in the response must match the query.

There are two ways that an attacker can produce a response with the correct values. He can either read them from the query, or he can guess them.

The first response that passes these checks is deemed valid and accepted; any subsequent responses are ignored. If the attacker can produce a response with the correct values then the response is guaranteed to be accepted.

There are two ways that an attacker can produce a response with the correct values. He can either read them from the query, or he can guess them.

2.1. DNS POISONING BY READING THE CORRECT VALUES FROM THE QUERY

If an attacker has access to the network that queries are sent over (e.g. an employee on a corporate network, an employee of an ISP network, or an attacker on a public wireless network, etc) and is able to access DNS queries, then he can read the information directly from the query and create a valid response every time.

2.2. DNS POISONING BY GUESSING THE CORRECT VALUES

For the attacker's response to be accepted it must pass the three checks mentioned above. Guessing each of these can be addressed by the attacker as follows:

1. Prior to summer 2008 almost all DNS servers used a source and destination UDP port of 512. The source IP address is easily obtained by an attacker, for example by sending a targeted e-mail or by social engineering. The destination IP address is public information. This makes it easy for an attacker to know the ports and IP addresses involved.

2. The attacker sends many fake responses with randomly chosen transaction IDs. For example, if the attacker can send 100 fake responses to the DNS server before the valid response arrives then he improves the chances of guessing correctly to 1 in 656.

3. For a known query, the query section (and any additional information) supplied in the query

response can be predicted – meaning the attacker wouldn't need to guess this.

Prior to announcements in summer 2008 by security researcher Dan Kaminsky, it was widely believed that the TTL provided protection against poisoning by limiting the number of queries sent to once per TTL (as caching DNS servers would only send one query per TTL period).

The likelihood of an attack succeeding can be calculated by the formula in Hubert and van Mook's paper 'Measures to prevent DNS spoofing'. The formula has been used to calculate the which an attack would require to have a 50% change of success:

- Probability required (50%).
- Number of distinct transaction IDs available (65536).
- Number of source ports used by DNS server (1).
- Number of authoritative nameservers for a domain (2). DNS requires a minimum of 2 nameservers for a domain; the attacker does not know which of these will respond, so has to guess this.
- Number of fake responses sent per second by the attacker (7000).
- Window of opportunity, in seconds, bounded by the round trip delay between the requesting server and the authoritative servers (0.1s).
- Number of identical outstanding DNS queries (1), a server may have multiple outstanding queries for the same record.
- TTL of record being poisoned (1 day).

Hubert and van Mook's formula show that for a 50% chance of success (assuming the TTL provided protection form poisoning) an attack would require an average of 129.4 days.

Dan Kaminsky realised that the attack was not limited to one attempt per TTL; he showed how, even if the attack is not successful, the attacker can start again immediately.

THE KAMINSKY VULNERABILITY

Dan Kaminsky realised that the attack was not limited to one attempt per TTL; he showed how, even if the attack is not successful, the attacker can start again immediately. Kaminsky realised

that if he could trigger queries for a record that did not exist (say 0001.alice.example.com) then it could not possibly be in the cache of the DNS server, and so it would always trigger a query. The attacker could try and spoof a response to that query. If the attacker is lucky he guesses the transaction ID correctly; if not he can try for another record that does not exist (say *0002.alice.example.com*). Sooner or later the attacker will get a match and his response will be added to the cache.

Sooner or later the attacker will get a match and his response will be added to the cache.

Adding *0002.alice.example.com* to the cache is of no benefit to the attacker. What is important is that the response includes additional information telling the server to send future queries for the example.com domain to servers controlled by the attacker. This is also added to the cache for the duration of the TTL; this can be a long time, up to 136 years. The attacker can now choose how to respond to queries for all hosts on the example.com domain.

What Kaminsky discovered made the attack much easier; Hubert and van Mook's formula (modified to remove the TTL) shows that for a 50% chance of success an attack would require an average of only 130 seconds.

2.3. DNS POISONING EXAMPLE — SPOOFING AN SSL WEBSITE

We have seen how it is possible to poison a DNS server, making it possible to send all future queries for a domain to servers that the attacker controls. The attacker can then respond to DNS queries with the addresses of servers of his choosing rather than the correct ones. Here we will look how this can be used to obtain a valid SSL certificate that will be trusted by browsers and then used with a spoofed copy of an SSL website.

Part of the process of obtaining a certificate for a website is the authentication of the requestor by the Certification Authority (CA). If this authentication is not properly completed then it is possible for an attacker to obtain a certificate for a domain that he does not own.

HOW THE ATTACKER OBTAINS A TRUSTED CERTIFICATE

To obtain a certificate trusted by web browsers

the attacker must:

- Create a spoofed copy of the target website.
- Poison the CA's DNS server for the target domain. The attacker can do this either using the Kaminsky vulnerability, or by having access to the network that the CA's DNS traffic will use (the CA's network or the CA's ISP's network). Whatever method is used the attacker will receive DNS queries for the target domain.

If the CA's DNS server had been poisoned, the email would have been delivered to the attacker who could then have completed the authentication process to the CA's satisfaction.

- Generate and submit a request for a free certificate for the target domain to the CA. During the preparation of the project report that this paper is based on, a request for a free certificate was generated and submitted to a well-known CA. The only authentication required of the requestor by the CA was the ability to respond to an e-mail at the domain for which the certificate was being requested. If the CA's DNS server had been poisoned, the email would have been delivered to the attacker who could then have completed the authentication process to the CA's satisfaction. The certificate was automatically issued by e-mail, with DNS poisoning enabling delivery to the attacker.
- The attacker then installs the certificate on the spoofed website.

DIRECTING TRAFFIC TO THE SPOOFED WEBSITE

To direct traffic to the spoofed website the attacker must poison the DNS server of the users he wishes to attack. He can do this either using the Kaminsky vulnerability or by having access to the network that the DNS traffic will use (a corporate network, ISP network, or a public wireless network etc). Any users of the poisoned DNS server will be directed to the spoofed website rather than the valid website.

THE RESULT

The result is that the user sees a spoofed website that is authenticated by the SSL protocol. Even users who have been educated to check for the

presence of a padlock would be fooled and are likely to believe that the website is genuine. A user checking the details of the certificate may notice that the certificate was issued as a free certificate valid only for a limited time and was 'domain control authenticated': how many users are likely to understand and check this? In this instance the certificate has assisted the attacker to trick the user into trusting a spoofed website; this may be used in a variety of attacks to obtain valuable and confidential information.

The authentication used by at least one CA when issuing free certificates is inadequate and is vulnerable to a DNS poisoning attack allowing an attacker to falsely obtain a public key certificate that is trusted by browsers by default.

3. CURRENT MITIGATION

3.1. SOURCE PORT RANDOMISATION

Before the Kaminsky vulnerability was publically disclosed, a summit was held to investigate options to mitigate the vulnerability. A solution was needed quickly and needed to be easy to implement and backwards compatible with deployed DNS software (preventing solutions that required changes to the DNS protocol). A decision was made to implement source port randomisation for DNS queries. Source port randomisation makes it more difficult for the attacker to spoof DNS responses by randomising the source UDP port used to send queries from the DNS server. With source port randomisation the attacker must correctly guess both the transaction ID and the source port of the query.

Patched Windows 2000, 2003 and 2008 servers allocate a 2,500-port range for the DNS source port. The number 2,500 is the default setting and was chosen to prevent the allocation conflicting with ports used by other services; it is however configurable for up-to 10,000 ports. Windows server 2008 R2 is configurable for up to 65,535 ports.

Using Windows with a 2,500 port range makes the attack 2,500 times more difficult. Hubert and van Mook's formula (modified to remove the TTL and using 2,500 source ports) shows that for a 50% chance of success an attack would require an average of 3.8 days (assuming source ports are chosen randomly).

After patching, BIND (a popular DNS server) can use the 1,024-65,535 range. Using BIND with the maximum 64,511 port range makes the attack 64,511 times more difficult. Hubert and van Mook's formula (modified to remove the TTL and using

64,511 source ports) shows that for a 50% chance of success an attack would require an average of 96.8 days (assuming source ports are chosen randomly).

Whilst adding randomisation to the query provides protection from the Kaminsky vulnerability, it does not remove the vulnerability; it just makes it more difficult to exploit. A fully patched BIND server has been successfully attacked by researcher Evgeniy Polyakov. Polyakov ran the attack on two servers directly connected to the target DNS server by Gigabit Ethernet in just under 10 hours. During the attack each server was able to send up to 50,000 fake responses before the valid response was received. That's a significant amount of traffic to go unnoticed; it's likely that the DNS servers would suffer from performance issues during the attack and this may lead to the attack being discovered. Polyako points out that a Trojan on a compromised host may be able to successfully poison a patched DNS server overnight when it's less likely to be noticed.

Source port randomisation provides no protection against an attacker with access to the network that queries are sent over. The attacker can read the source port and transaction ID directly from the query, allowing him to create a valid response every time.

3.2. TRANSPORT SECURITY FOR DNS TRAFFIC

The use of transport security such as Transaction Signatures (TSIG) or IPSec can be used to protect against modification of DNS traffic. TSIG uses a message authentication code (HMAC-MD5) with a shared key to provide data integrity and data origin authentication. IPSec can be used to provide protection between the client and its DNS server (available for Windows 7 clients with Windows 2008 R2 DNS severs).

3.3. MULTIPLE QUERIES

DNS Severs could be configured to send queries for the same record to multiple different servers. Sending multiple queries means that the attacker must poison multiple responses simultaneously, increasing the difficulty of an attack.

3.4. SOURCE CASE RANDOMISATION

DNS is case insensitive; however it does require that the case is preserved between the query and the response. This makes it possible to introduce additional randomisation into a query by randomising the case of the letters in a domain name. Each letter in a domain name doubles the difficulty of a poisoning attack.

3.5. CACHE LOCKING IN WINDOWS 2008 R2

Windows 2008 R2 can be configured to prevent or restrict entries in the cache being overwritten during their TTL. With the default configuration records in the cache can't be overwritten before expiry of the TTL.

4. FUTURE MITIGATION

The DNS Security Extensions (DNSSEC) and DNSCurve provide cryptographic responses to protect against poisoning attacks, as discussed below.

4.1. DNSSEC

DNSSEC provides assurance for DNS records to protect against poisoning. It achieves this using public key cryptography to generate digital signatures on DNS records; the validity of these signatures can be traced back by a DNSSEC-aware DNS server to a trust anchor. The aim is that DNSSEC will have a single anchor at the DNS root that all DNS servers will trust and use to validate DNS responses. DNSSEC provides the following services:

- Data Origin Authentication
- Data Integrity
- Authenticated Denial of Existence

The DNSSEC trust model is based on verifiable proof that the digital certificate corresponding to a DNS record was signed by the holder of a key. Any holder of a trust anchor public key gains data origin authentication and data integrity assurance for that record. Clients and servers that are not DNSSEC-aware must trust their DNS servers to validate the digital certificates for records and to serve them without modification; mechanisms such as IPSec or TSIG can be used to protect the link for this traffic.

Currently the DNS root is not signed so the single trust anchor model can't be followed today; if the relying party wanted to verify any *example.com* responses today he would need a trusted copy of the example.com public key. This adds an overhead to obtain and import these keys. DNSSEC Lookaside Validation (DLV) has been proposed to provide a mechanism for a server to automatically obtain a copy of the public key.

DNSSEC also aims to provide authenticated denial of existence – i.e. verifiable proof that a domain name does not exist. It does this using Next Secure (NSEC) and NSEC3 records. If a request is made for a nonexistent host, the NSEC response includes the names for the valid preceding record and the valid following record (names are arranged in a defined manner). This listing of

hosts allows the requestor to identify if a host exists. However, it also allows an attacker to work their way through all records and to identify all hosts in a domain, a valuable tool in reconnaissance. NSEC3 has been proposed to provide authenticated denial of existence whilst providing measures to address the reconnaissance issue of NSEC. It achieves this by replacing the names in the records with cryptographic hashes. NSEC3 does provide an improvement on NSEC; however,

NSEC3 has been proposed to provide authenticated denial of existence whilst providing measures to address the reconnaissance issue of NSEC.

it does not protect against offline attacks against the hashed values of the records. Whilst this is more difficult and time consuming than NSEC, it does not prevent the possibility of (at least) partial enumeration. On 19th January 2010 CERT announced a vulnerability in the BIND NSEC/NSEC3 validation code that may allow an attacker to poison a DNSSEC DNS server and cause it to return fake responses indicating that a domain name does not exist when in reality it does.

On June 3 2009 the US National Telecommunications and Information Administration (part of the US Department of Commerce) and the US National Institute of Standards and Technology announced that they were working with the Internet Corporation for Assigned Names and Numbers (ICANN) and VeriSign to an approach to signing the DNS root. On the 6th October 2009 ICANN and VeriSign (who operate some of the DNS root servers and administer the com and net domains) announced full deployment of DNSSEC at the root zone by July 1st 2010. VeriSign have also announced that the net zone will be signed by the end of 2010 and the com zone in early 2011.

It is 13 years since the first DNSSEC RFC was published. However it is still suffering from issues including:

- Denial of service (DoS). DNSSEC produces much larger records; these can be used in amplification attacks. DNSSEC servers may also be the target of DoS attacks where resources are consumed attempting to validate fake signatures.
- Leaks information. NSEC and NSEC3 leak information about hosts in a DNS domain through support for authenticated denial of existence.

- No protection for user hosts that do not perform validation of the digital signature. A malicious DNSSEC server may provide a user host with a response marked as validated, but which includes a fake response.
- Complexity. DNSSEC is much more complicated than standard DNS introducing additional risk such as configuration errors or bugs.
- Signature expiration. Signatures are valid for a limited time, if not regenerated then records will not validate.
- Migration. DNSSEC deployment will take time. For this interim period where some records are signed and some are not, yet unsigned records will need to be trusted.
- Revocation. DNSSEC does not provide revocation and is therefore vulnerable to replay attacks where records have been changed before the end of the validity of the digital signature.

DNSSEC has recently received much attention and is gaining traction, probably due to the Kaminsky vulnerability. The increased attention is likely to help DNSSEC to address these issues over time.

4.2. DNSCURVE

DNSCurve provides assurance for DNS records at the transport layer; it achieves this using elliptic curve cryptography to provide link level encryption (confidentiality). DNSCurve provides the following services:

- Confidentiality
- Integrity
- Availability

The integrity of DNS messages is validated by a resolver to identify if a message has been modified. DNSCurve claims to enhance availability by dropping DNS responses that fail the integrity check, meaning the responses it accepts are the ones sent by the server (protecting against attackers denying service by poisoning the DNS with incorrect records).

DNSCurve does not require any changes to the DNS protocol; the above services are provided between two DNSCurve-aware hosts whilst maintaining interoperability with DNS. DNSCurve provides hop-by-hop data integrity; it requires that the resolver trusts the servers involved in resolution.

DNSCurve does not provide the same level of assurance as DNSSEC; its trust model only provides assurance that data is not modified in transit. Protection is not provided against any modification of data whilst stored, or over non DNSCurve links. DNSCurve resolvers must trust their DNS servers

to serve records without modification.

DNSCurve does not have the same goals as DNSSEC; it does not provide data origin authentication or end-to-end integrity. It provides a solution to cache poisoning attacks and provides confidentiality. DNSCurve is beneficial in that it provides a solution that is less complicated than DNSSEC, but it provides less assurance.

5. SUMMARY

We all rely on the DNS, most of us daily; it plays a crucial role for users of the Internet and private networks. The validity of DNS responses is critical for the users of these networks to enable them to connect to the correct services. The Kaminsky vulnerability showed that it was easy to poison the DNS. Whilst source port randomisation has provided a short term solution to that problem, it does nothing to protect against DNS poisoning by an inline attacker. A solution (such as DNSSEC or DNSCurve) is required to provide greater assurances for DNS traffic for protection against DNS poisoning attacks. In the interim we need to ensure that our DNS servers are patched and properly configured and protected by best practice guidelines. ■

ABOUT THE AUTHORS

Richard Agar started in the IT industry in 1999 working at Nortel Networks, where he developed an interest in information security. He has recently completed the Masters degree in Information Security at Royal Holloway, and is now working for intrusion prevention specialists TippingPoint, part of HP, as a Senior Systems Engineer.

Professor Kenneth Paterson has research interests in theoretical and applied cryptography; network and mobile security; and coding theory and mathematics of communications.