Minimizing Information Leakage in the DNS

Scott Rose and Anastase Nakassis, National Institute for Standards and Technology

Abstract

The domain name system is the global lookup service for network resources. To protect DNS information, the DNS security extensions have been developed and deployed on branches of the DNS to provide authentication and integrity protection using digital signatures. However, signed DNS nodes were found to have an unfortunate side effect: an attacker can query them as reconnaissance before attacking hosts on a particular network. There are different ways a zone administrator can minimize information leakage and still take advantage of DNSSEC for integrity and source authentication. This article describes the risk and examines the protocol and operational options and looks at their advantages and drawbacks.

The domain name system (DNS) is a distributed global lookup service and is usually the first step in any network communication [1]. This also is true for many network attacks and malware operations. Most attacks also start with a network scan of some sort: either for hosts or for certain software with known vulnerabilities on discovered hosts. This scan can be made easier if the attacker has a copy of the domain database (DNS zone) they wish to attack. The DNS is increasingly being used to store data (in the form of DNS resource records [RRs]) that support other applications, which also can be of potential use to an attacker. If an attacker can somehow enumerate the contents of the victim DNS zone, it may reduce the work required for a successful exploit because the host systems on the network are already identified. This is especially valuable on an IPv6 network. Often it is easier to scan an IPv4 address block than try to enumerate a DNS zone. With IPv6, it is the opposite: The DNS zone database is much more valuable to an attacker as it is much more time consuming to scan a typical IPv6 address block.

The DNS is increasingly being used to support a variety of other Internet protocols beyond simple domain name to IP address translation. For example, there are defined RR types in the DNS to store host secure shell (SSH) key hashes and other RR types that identify mail servers for a domain. This additional information may have security or privacy concerns beyond that of IP addresses. Even the domain name itself can be tied to a WHOIS database lookup that provides information about the real-world registrant of the name and leads to a possible confidentiality violation, which is an issue of legal concern among some countries (that may have strict regulations about personal identifying information on the Internet).

DNS data has always been considered public; however, there have been some practical limits to make zone data difficult to obtain. Zone transfers (sending a query for all the contents of a zone database) can be refused, forcing an attacker to make a brute force attack in an attempt to enumerate the contents of a zone. Even with the relatively small set of characters allowed in domain names (DNS names are not case-sensitive, and all but a few special characters are invalid), this is a non-trivial task. Even with distributed attackers, it takes time due to network delays. These query streams also can trigger intrusion detection programs that could alert the victim.

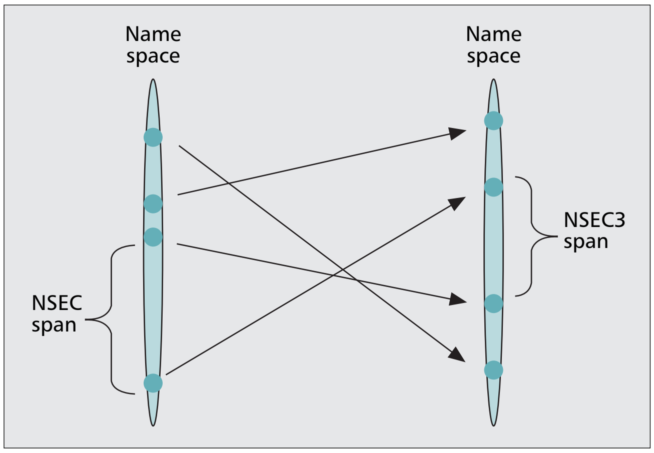
The Problem

Recent additions to the DNS protocol to add origin authentication and integrity to DNS data (referred to as the DNS security extensions [2–4] or DNSSEC for short) has had an unfortunate side effect related to zone enumeration. DNSSEC adds three new resource record types (RR types) to store public keys, hashes, and digital signatures, as well as a fourth RR type to handle authenticated denial of existence. This RR, called the next secure RR (NSEC RR) is used to provide digitally signed responses for error situations, such as a client query for a non-existent name.

The NSEC RR provides proof of non-existence by providing two valid, ordered names in the zone between which there are no valid domain names. For example, if a client queries for the IP address of the host `b.example.com` and gets back an NSEC RR with `a.example.com` and `c.example.com`, the client can deduce that the name `b.example.com` does not exist as it falls in the span bookended by `a.example.com` and `c.example.com`. As with all DNSSEC responses, a separate digital signature accompanies the response to prove the data came from an authoritative server for the zone.

As a side effect, the client now knows two names that do exist in the zone that it might not have known before: `a.example.com` and `c.example.com`. The NSEC RRs in a zone form a chain to cover the entire name space — linking every name through NSEC RRs. Using this information, an attacker can start to "walk" the zone and send a subsequent query to find the name that exists after `c.example.com` all the way until the NSEC chain loops back to the top of the zone. For a zone with N names, it would take *N* queries to obtain a list of every name in the zone and any other information stored in the DNS. An attacker can then use this list in planning an attack against individual hosts.

One early response to this was to have "minimal spanning NSEC RRs," described in [5]. The idea is to have a server generate a set of NSEC RRs based around the error producing query name. This would increase the time in enumerating the zone using NSEC RRs to the same as a direct brute force attack against the server as the span covers only the error-generating query name; the attacker does not gain knowledge of hosts that do exist in the zone. This variant requires a serv-



■ Figure 1. *Name space and hashed name space.*

er to generate and sign a unique response for every error message, which could quickly become a denial of service (DoS) attack against the server that may do more harm to a network than a zone enumeration attack.

That does not mean there must be a trade-off between the content protection DNSSEC provides and reducing DNS information exposure. There are other methods zone administrators can utilize to minimize zone information leakage and still have the benefits of DNSSEC signed zones. One such method is using a variant of the NSEC RR known as the NSEC3 RR. The other is using operational architectures to insure that only those hosts that must be publicly accessible can be found in the global DNS, whereas private hosts are kept in a separate name space isolated from all but those clients that must query them.

NSEC3 Variant in DNSSEC

The NSEC3 resource record [6] (NSEC3) is a recent variation of the original NSEC RR in DNSSEC. It was developed to add some obfuscation to the domain names in the NSEC RR to make zone enumeration a more difficult task. Its format is identical to the NSEC RR but with hashed domain names (using a one-way hash function such as SHA-1; see Fig. 1). That way, a client can still determine that the query name does not exist (the hash of the query name falls in the span between the two hashed names provided), but not learn about any valid domain names that do exist in the zone. The NSEC3 RR is used exactly in the same way as the NSEC RR but requires both the server and the resolver to be able to perform the hash function used (by default — SHA-1, but SHA-256 and other hash functions can be defined).

There are additional fields in the NSEC3 RR to declare the number of iterations and salt value to be used with the hash calculation. The salt value is a hex string that is to be appended to the query name before hashing, and the iterations field is used to indicate how many times the hash function is to be computed for a give name. Both values can be changed periodically to compensate for increased computing power available to an attacker.

The NSEC3 RR is not a perfect solution, because existing domain names are still being leaked with every name error response. Depending on the error response, multiple NSEC3 RRs may be returned, each with two (hashed) real names that exist in the domain. This means that it is still possible to get a complete list of hashed names in the zone. Unlike DNSSEC with NSEC, an attacker cannot always choose which queries to send to obtain all of the names, but we can calculate how many queries on average. While DNSSEC with NSEC requires N queries to map the names of a zone with N names, in the naive case where hashed names are evenly dispersed throughout the (hashed) name space, DNSSEC with NSEC3 requires an attacker send $O(N(\ln N))$ queries to obtain the same list of N names and in cases where the hashed names are not evenly distributed, the required number of queries can reach $O(N^2(\ln N))$, which could still be manageable for a small number of names [7].

In practice, the number of queries and the time required to conduct this attack depends on the computing power of the attacker, as well as the network delay and the number of unique domain names in the zone. An attacker can reduce the time and number of queries required by pre-computing the hash of possible query names and rejecting those that fall in known NSEC3 spans. This only applies if the work required to construct the hash is less than the time required to conduct a DNS transaction. That is, the zone administrator chooses iteration values low enough that it is cheaper to generate a hash than to send a query for a random name to the DNS server.

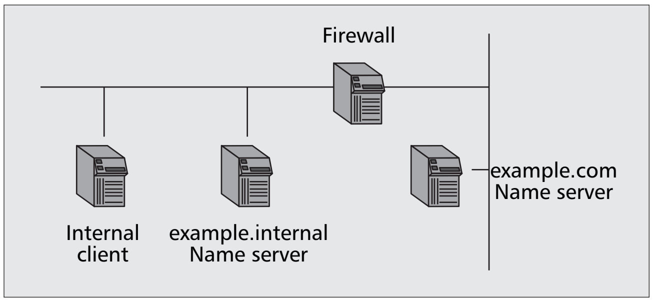
If an attacker can conduct these queries in parallel (using a botnet for example), it is still possible to obtain a list of hashed names for a particular zone in a relatively modest amount of time. The attacker can then perform a brute force attack with the list stored locally (and avoid unnecessary network traffic that may trigger a security monitoring alert).

This is where the NSEC3 iterations and salt field come into play. The salt value is a random string chosen by the zone owner. The purpose is to prevent an attacker from pre-computing a dictionary of all possible names to compare against hashed names discovered in NSEC3 RRs. If attackers cannot predict the salt value used, they cannot construct a dictionary and may be required to reconstruct the list of hashed names when the salt changes — unless they are sure they have the entire list of names. The iterations value is used to slow an attacker when conducting a brute force attack against a list of discovered hashed names. Increasing the iterations value increases the number of times the hash should be computed. The goal is to make an offline attack as time consuming as conducting a brute force dictionary attack against the server. However, choosing an iteration value that is too large (making it more expensive than a brute force attack against a server) also slows the server itself (and all clients) who must also perform hash calculations to form (and validate) responses using NSEC3 RRs.

An attacker does have several options available that can defeat high iteration values in NSEC3 RRs. First, attackers may have more computing power available than the zone owner has, either through more powerful hardware or through a network of computers (such as a botnet) that enables them to attack the list of hashes in a distributed manner. Second, the attacker need not necessarily conduct a brute force attack. The nature of DNS is that it is used to map human readable names to IP addresses or other network resources. Therefore, there is more likely a subset of all possible name combinations in use for names in the zone. For example, the majority of zones has at least one host called "www" and may have another host named "mail" or "router," and so on. Previous network studies have compiled lists of the most common names found in zones that an attacker can use to quickly discover if certain common names are among the list of hashed names obtained from the zone [8].

Split DNS

Another option to prevent zone content from attackers wishing to map the resources of a network is simply to remove those names from the global DNS. Most large organizations already separate their internal network from the global Inter-



■ Figure 2. *Split DNS using a separate internal domain.*

net and have only public servers (Web, mail, etc.) available to external clients. It makes sense to keep the name space of this internal network separate from the name space of the external facing network and prevents information leakage into the internal network, as well. There would be two separate name spaces: one that is accessible only from internal hosts and one that is accessible from the outside. The externally accessible name space would be comprised of only servers that host public services. This is often called split DNS.

Split DNS can be done in different ways [9]. Some DNS server software packages allow for two (or more) separate views of the name space with configuration language to specify which version of the domain should be used to answer queries based on the IP address of the client asking the query. This means queries coming from the internal network would obtain responses based on the internal view of the name space, and queries from external clients would obtain responses based on the external view.

More simple approaches can achieve the same result, such as having a separate internal domain or subdomain delegated from the external name space. For example, the company `example.com` may have all of its internal hosts in a separate domain `internal.example.com` or even `example.internal`. Neither of these domains should ever respond to external queries, so it does not matter if the domain suffix is not a valid top-level domain. For example, Fig. 2 shows a split DNS environment for an organization with the domain name `example.com` and with the internal domain `example.internal`. Here, the internal client has both name servers as its name servers to be able to access both internal and external zones.

However, using a split DNS does not solve every problem and may introduce a new insider attack. First, roaming clients that leave the internal network will not be able to access the internal name space unless they have a connection that allows them to use a DNS server that has the internal domain. Note that roaming clients also must be able to use the internal network to begin with, so if the connection can be established via a virtual private network (VPN) or similar, there is little extra effort required to enable internal DNS requests as well.

The new insider attack comes from an attacker obtaining an error message from the external network view and replaying it on the internal network. For example, the attacker sends a query to the external DNS server for information about a host on the internal network. It saves the error message (and NSEC/NSEC3 RRs if the zone is using DNSSEC) and replays that error message to internal hosts looking for the same internal server. The client who receives this replayed error message would believe that it is valid unless they can prove otherwise, and a denial of service results.

It is also possible to deploy DNSSEC in these split DNS scenarios [10]. Because the external DNS contains only hosts that are publicly accessible, it should not matter if the external zone is enumerated or not, so there is little need for NSEC3. DNSSEC with NSEC would be sufficient, as any sensitive hosts would be moved behind the network separation. There is little to gain by an attacker in enumerating the zone because the only hosts present are public servers that are discoverable in other ways. This reduces the danger of a successful zone enumeration attack and avoids the DoS risks associated with deployment of NSEC3. The internal name space can be signed using DNSSEC as well, but care must be taken to insure that hosts required to access both internal and external networks are able to validate both sets of signatures. That is, those hosts must have the public keys required to validate both sets of signatures (if they are different). In Fig. 2, the internal client must have the public keys of both `example.com` and `example.internal`.

More care must be taken when deploying DNSSEC on the internal zone (if desired). Again, the problem with deploying DNSSEC on internal zones is the risk of denial of service against roaming hosts. Systems that must access internal and external network resources must be able to direct their DNS queries to the appropriate server. Likewise, a roaming host must either have access to an internal DNS server (via a VPN, as discussed previously), or clear any caches it may have upon re-entering the internal network to avoid a self- inflicted DoS attack when trying to query for internal systems if the only cached information is from external servers.

Conclusions

Recent security extensions to the DNS (DNSSEC) have been developed to provide integrity and source authentication for DNS zone data. However, this attempt to prevent one class of attacks (client redirection) has inadvertently made a new type of attack on the DNS possible — zone enumeration. This attack was always theoretically possible but was too expensive to perform compared to the gain. Whereas enumeration may be a direct risk to only a few zones, it is often the start of a directly targeted attack. There are multiple ways to minimize this risk; each with advantages and drawbacks.

The most direct solution is to have the DNS server generate a NSEC RR for each negative response that shrinks the span so that only the non-existent query name is covered. This would force an attacker to expend the same number of queries as a direct brute force attack. It also would force a server to generate NSEC RRs (and sign them) during run time, which could quickly become a DoS attack if the attacker could send a large volume of queries.

The NSEC3 variant of DNSSEC was developed to minimize zone information leakage and zone enumeration attacks. NSEC3 RRs use hashed domain names instead of cleartext domain names when forming negative answers in the DNS. However, it does not make it impossible to discover the names in a particular zone database. It only requires more work by the attacker to learn about actual host names by first obtaining the list of hashed names and launching an attack against that list.

The operational means to minimize zone enumeration is to keep separate zone files for internal and external hosts. Most organizations do this to some extent already by keeping systems they wish to remain private behind a firewall.

This can be done as a separate view (to keep the same domain suffix) or as a separate zone or subzone. Because outside attackers have no knowledge of the existence of the internal zone and cannot access the internal network, it becomes impossible to enumerate from outside the firewall. External servers are meant to be public, and it should not matter if the external network is enumerated. However, this set up requires much more planning and effort on the zone administrator's part and makes DNSSEC deployment more difficult to maintain if both internal and external zones are to be signed. Having DNSSEC also makes an insider attack possible if an attacker can replay external zone error messages inside the private network. This would result in a DoS attack against the internal network client.

The type of defense to select depends on the organization. Because most organizations already have a split network, extra care to insure that only public servers appear in the external DNS makes sense. For zone administrators that have privacy concerns above zone enumeration, NSEC3 deployment may be the only choice. The DNS was designed to be public, and there is no way (short of using encrypted, secure network transmission) to totally prevent a zone enumeration attack. The only possible goal is to minimize its effectiveness.

Acknowledgments

This work was funded by the Department of Homeland Security Science and Technology Directorate.

References

[1] C. Liu and P. Albitz, *DNS and BIND*, 5th ed., O'Reilly Media, May 2006.
[2] R. Arends *et al.*, "DNS Security Introduction and Requirements," RFC 4033, Mar. 2005.
[3] R. Arends *et al.*, "Resource Records for the DNS Security Extensions," RFC 4034, Mar. 2005.
[4] R. Arends *et al.*, "Protocol Modifications for the DNS Security Extensions," RFC 4035, Mar. 2005.
[5] S. Weiler and J. Ihren, "Minimally Covering NSEC Records and DNSSEC Online Signing," RFC 4470, Apr.2006.
[6] B. Laurie *et al.*, "DNSSEC Hashed Authenticated Denial of Existence," draft-ietf-dnsext-nsec3-10.txt, Jan. 2007, work in progress.
[7] More detailed proof available at http://www-x.antd.nist.gov/dnssec/NSEC3-query-proof.html
[8] ISC Internet Domain Survey; http://www.isc.org/ops/ds/reports/2007-02/
[9] R. Chandramouli and S. Rose, "Secure Domain Name System (DNS) Deployment Guide" NIST special pub. 800-81, May 2006.
[10] S. Krishnaswamy, "Split-View DNSSEC Operational Practices," draft-krishnaswamy-dnsop-dnssec-split-view-03.txt, Aug. 2006, work in progress.

Biographies

SCOTT ROSE (scottr@nist.gov) received a B.A. in computer science from the College of Wooster and an M.S. in computer science from the University of Maryland, Baltimore. He works as a computer scientist at the National Institute for Standards and Technology (NIST). He was on the editor team that produced the latest version of the DNS security extensions (DNSSEC).

ANATASE (TASSOS) NAKASSIS (tassos@nist.gov) studied mathematics at the University of Athens and the University of Paris (VI), obtaining a doctoral degree in algebra, and at the University of South Florida, obtaining a doctoral degree in probability. He is employed by NIST and is currently working on problems that straddle mathematics, information theory, and algorithm development with an emphasis on the classical aspects of quantum cryptography.