ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH - TECHNOLOGY

DNS SECURITY : A REVIEW

Darshana Hooda*, Sushma
*Computer Centre, DCRUST,Murthal, Haryana,India
Department of CS, Banasthali Vidyapeeth,Rajsthan, India

ABSTRACT

DNS is the basic method which allows a domain name to lead customers to your website as they attempt to log onto your site. DNS has been extended to provide security services (DNSSEC) mainly through public-key cryptography. This is a review paper on the security problems affecting the Domain Name System. Corrupting the operation of DNS in this way can lead to many kinds of fraud and other malicious activity. By plugging some of the largest security holes in the Internet, DNSSEC has the potential to significantly expand the trustworthiness and thus the usefulness of the Internet as a whole.

KEYWORDS: DNS, DNSSEC, Security Vulnerability.

INTRODUCTION

Internet-connected devices are identified by IP addresses, though users typically only know web addresses—people can remember “example.edu,” for instance, more easily than “192.168.7.13.” The Domain Name System (DNS) uses a distributed network of name servers to translate text-based web addresses into IP addresses, directing Internet traffic to proper servers. The Internet doesn't work without the DNS. Unfortunately when the DNS was developed in 1983, security controls weren't built in and over the years, serious security flaws have been discovered resulting in numerous changes. Most recently, researcher Dan Kaminsky discovered a major flaw in the DNS that allowed cache-poisoning attacks, which essentially deceives a DNS server into believing it has received legitimate data when it may actually be fraudulent. One of the biggest changes to the DNS is DNSSEC, which adds security controls to the original protocol. Specifically the DNSSEC provides additional extensions to the original DNS protocol that allows for origin authentication of the DNS data, data integrity and authenticated denial of existence. In simple terms, the DNSSEC thwarts spoofing attacks by allowing websites to validate domain names and the associated IP addresses using digital signatures and public-key encryption. This mitigates the threat of bad guys hijacking your Web traffic and redirecting it to fake sites to carry out their dastardly deeds. Citizens and other users of government website services would rightfully consider this unacceptable. DNS Security Extensions (DNSSEC) adds security provisions to DNS so that computers can verify that they have been directed to proper servers. DNSSEC authenticates lookups of DNS data. An attacker who is able to send DNS responses to a vulnerable system could cause a denial of service, crashing the application that made calls to a vulnerable resolver library. It does not appear that this vulnerability can be leveraged to execute arbitrary code. There may be some risk of information disclosure if a vulnerable system returns thecontents of memory adjacent to a DNS response. Today’s complex networks must deliver the utmost security and reliability to protect against potential security threats.

BACKGROUND

Some of the important elements involved in the domain name resolution process are the following:
• Stub resolver: The originator of a DNS query. This could be a simple client machine or a web browser.
•Recursive DNS (RDNS): This is a server machine that is responsible to assist the stub resolver on resolving a domain name. These servers maintain a local cache of past resolved domain names for a certain period of time (equal to the domain name’s time to live (TTL)), and are the main target of cache poisoning attacks.
•Root and Top Level Domain (TLD) servers: These are servers that provide referrals to the TLDs and Start of Authority servers respectively.
•Start of Authority (SOA): The SOA server(s) represent the authoritative name server for an entire name zone. In the majority of the cases once a query reaches the appropriate SOA server, the SOA will be

able to provide an domain name to IP address mapping or a NXDOMAIN (or NX, for brevity) if that domain does not exist.
In Figure 1 we can see the entire resolution process for a recursive query from the stub resolver to the SOA. Assume the stub issues a query in order to resolve example.com (step 1). The query will reach the local RDNS. Assuming the domain name is not located in the cache of the RDNS server, the RDNS will initiate an iterative query process in order to retrieve the mapping between the domain name example.com and its IP address.

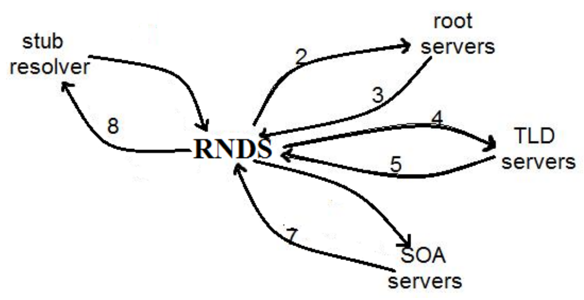

Fig.1: A common DNS query resolution process[4]

In the next step (step 2) the RDNS will contact the root servers asking for example.com. The root servers have no knowledge of the IP address of the queried domain. The only thing they can provide is a referral (step 3) to the .com TLD servers. The RDNS will then ask the .com TLD servers (step 4) for the IP of example.com, and the TLD will response with a referral to the SOA forthe queried domain (step 5). By contacting the SOA (steps 6 and 7) the RDNS will finally get the A record containing the IP address for example.com. At this point the RDNS will forward the answer to the stub resolver (step 8) [4].

DNS- INTRODUCTION

The Internet and TCP/IP, IP addresses are used to route packets from source to destination. A single IP address, for example 203.192.135.234, is not difficult to remember. But trying to learn or track thousands of these addresses, including which server/node is associated with each address, is a daunting task. So instead, the client uses domain names to refer to systems with which client wants to communicate [5]. Domain Name System (or Service or Server) an internet service that translates domain names into IP addresses. Because domain names are alphabetic, they're easier to remember. The Internet however, is really based on IP addresses [1]. Because maintaining a central list of domain name/IP address correspondences would be impractical, the lists of domain names and IP addresses are distributed throughout the Internet in a hierarchy of authority [2].The Domain Name System is a standard technology for managing the names of Web sites and other Internet domains. DNS technology allows you to type names into your Web browser like compnetworking.about.com and your computer to automatically find that address on the Internet. A key element of the DNS is a worldwide collection of DNS servers [3]. The DNS system is, in fact, its own network. If one DNS server doesn't know how to translate a particular domain name, it asks another one, and so on, until the correct IP address is returned [1].

DNS- WORKING

The process of retrieving data from DNS is called name resolution or simply resolution. There are two modes of resolution in DNS: iterative and recursive. In the iterative mode,
when a name server receives a query for which it does not know the answer, the server will refer the querier to other servers that are more likely to know the answer. Each server is initialized with the addresses of some authoritative servers of the root zone. Moreover, the root servers know the authoritative servers of the second-level domains (e.g., edu domain). Second-level servers know the authoritative servers of third-level domains, and so on. Thus by following the tree structure, the querier can get closer to the answer after each referral. When a root server receives an iterative query for the domain name, it refers the querier to the edu servers. The querier will locate the authoritative servers and obtain the IP address. In the recursive mode, a server either answers the query or finds out the answer by contacting other servers itself and then returns the answer to the queries.
Internet is an IP network. Every host is affected an IP address that must be known to any other host willing to communicate. It would be possible to create the mappings between IP addresses and names locally to each computer. DNS provides a way to know the IP address of any host on the Internet [13].
Reference 1 provides more detailed information of the basics of DNS [13].

DOMAIN NAME SYSTEM SECURITY EXTENSION(DNSSEC)

The Domain Name System Security Extensions (DNSSEC) adds data origin authentication and data integrity to the Domain Name System. The security extensions consist of a set of new resource record types and modifications to the existing DNS protocol. The DNS security extensions provide origin authentication and integrity protection for DNS data, as well as a means of public key distribution. These extensions do not provide confidentiality [1].

Authentication Chain: An alternating sequence of DNS public key (DNSKEY) RRsets and Delegation Signer (DS) RRsets forms a chain of signed data, with each link in the chain vouching for the next. A DNSKEY RR is used to verify the signature covering a DS RR and allows the DS RR to be authenticated. The DS RR contains a hash of another DNSKEY RR and this new DNSKEY RR is authenticated by matching the hash in the DS RR. This new DNSKEY RR in turn authenticates another DNSKEY RRset and, in turn, some DNSKEY RR in this set may be used to authenticate another DS RR, and so forth until the chain finally ends with a DNSKEY RR whose corresponding private key signs the desired DNS data.

Authentication Key: A public key that a security-aware resolver has verified and can therefore use to authenticate data. A security-aware resolver can obtain authentication keys in
three ways. (i) The resolver is generally configured to know about at least one public key; this configured data is usually either the public key itself or a hash of the public key as found in the DS RR (see "trust anchor"). (ii) The resolver may use an authenticated public key to verify a DS RR and the DNSKEY RR to which the DS RR refers. (iii) The resolver may be able to determine that a new public key has been signed by the private key corresponding to another public key that the resolver has verified. Note that the resolver must always be guided by local policy when deciding whether to authenticate a new public key, even if the local policy is simply to authenticate any new public key for which the resolver is able verify the signature.

Authoritative RRset: Within the context of a particular zone, an RRset is "authoritative" if and only if the owner name of the RRset lies within the subset of the name space that is at or below the zone apex and at or above the cuts that separate the zone from its children, if any. All RRsets at the zone apex are authoritative, except for certain RRsets at this domain name that, if present, belong to this zone's parent.

Non-Validating Security-Aware Stub Resolver: A security-aware stub resolver that trusts one or more security-aware recursive name servers to perform most of the tasks discussed in this document set on its behalf. In particular, a non-validating security-aware stub resolver is an entity that sends DNS queries, receives DNS responses, and is capable of establishing an appropriately secured channel to a security-aware recursive name server that will provide these services on behalf of the security-aware stub resolver.

Security-aware stub resolver, validating security-aware stub resolver Non-Validating Stub Resolver: A less tedious term for a non-validating security-aware stub resolver. Security-Aware Name Server: Entity acting in the role of a name server that understands the DNS security extensions defined in this document set. In particular, a security-aware name server is an entity that receives DNS queries, sends DNS responses, supports the EDNS0 message size extension and the DO bit and supports the RR types and message header bits defined in this document set.

Security-Aware Recursive Name Server: An entity that acts in both the security-aware name server and security-aware resolver roles.A more cumbersome but equivalent phrase would be "a security-aware name server that offers recursive service".

Security-Aware Resolver: Entity acting in the role of a resolver that understands the DNS security extensions defined in this document set. In particular, a security-aware resolver is an entity that sends DNS queries, receives DNS responses, supports the EDNS0 message size extension and the DO bit and is capable of using the RR types and message header bits defined in this document set to provide DNSSEC services.

Security-Aware Stub Resolver: entity acting in the role of a stub resolver that has enough of an understanding the DNS security extensions defined in this document set to provide additional services not available from a security-oblivious stub resolver. Security-aware stub resolvers may be either "validating" or "non-validating", depending on whether the stub resolver attempts to verify DNSSEC signatures on its own or trusts a friendly security-aware name server to do so [7].

SECURITY VULNERABILITY TO DNS

It is known the fact that DNS is weak in several places. Using the Domain Name System we face the major problem of DNS amplification, DNS cache poisoning and DNS spoofing and some other problem Misdirected Destination: Trusting Faked Information , Name Based Authentication/Authorization . DNS protocol attacks are based on flaws in the DNS protocol Implementation. In order to be able to assess the potential threats and the possible counter-measures it is first and foremost necessary to understand the normal data flows in a DNS system. Diagram

(1) The primary source of Zone data is normally the Zone Files (and don't forget the named.conf file which contains lots of interesting data as well). This data should be secured and securely backed up. This

threat is classified as Local and is typically handled by good system administration.

(2) If you run slave servers you will do zone transfers. Note: You do NOT have to run with slave servers, you can run with multiple masters and eliminate the transfer threat entirely. This is classified as a Server-Server (Transaction) threat.

(3) The BIND default is to deny Dynamic Zone Updates. If you have enabled this service or require to it poses a serious threat to the integrity of your Zone files and should be protected. This is classified as a Server-Server (Transaction) threat.

(4) The possibility of Remote Cache Poisoning due to IP spoofing, data interception and other hacks is a judgment call if you are running a simple web site. If the site is high profile, open to competitive threat or is a high revenue earner you have probably implemented solutions already. This is classified as a Server-Client threat.

(5) We understand that certain groups are already looking at the implications for secure Resolvers but as of early 2004 this was not standardized. This is classified as a Server-Client threat.

DNS AMPLIFICATION

Several attackers massively exploited recursive name servers to amplify DDoS(Distributed Denial of Service) attacks against several networks utilizing IP spoofing. The DNS uses a tree-like system of delegations. Recursion is the process of following the chain of delegations, starting at the Root zone, and ending up at the domain name requested by a user. A recursive name server may need to contact multiple authoritative name servers to resolve given name on behalf of the requestor. A recursive name server should only accept queries from a local, or authorized clients. Unfortunately, many recursive name servers accept DNS queries from any source. Furthermore, many DNS implementations enable recursion by default, even when the name server is intended to only serve authoritative data. We say that a name server is an "open resolver DDoS attacks using recursive name servers can create an amplification effect similar to the now-aged Smurf attack (A SMURF attack (named after the program used to perform the attack) is a method by which an attacker can send a moderate amount of traffic and cause a virtual explosion of traffic at the intended target).

The Smurf attack works by sending an ICMP Echo request (type 8, a ping) to broadcast addresses on affected networks. These receiving hosts in turn relay the request and a reply to the spoofed location are initiated. In the Smurf effect, on a multi-access broadcast network, one can expect every single ping to result in attack amplification by triggering replies from all the active computers on the amplification subnet. The amplification effect in a recursive DNS attack is based on the fact that small queries can generate larger UDP packets in response. In the initial DNS specification, UDP packets were limited to 512 bytes.

New RFC specifications, - in support of IPv6, DNSSEC, NAPTR and other extensions to the DNS system, - require name servers to return much larger responses to queries. This increased UDP payload capability is now being used to launch attacks with higher UDP response amplifications. The amplification of a standard Smurf attack relies on sending a packet to a broadcast address which then causes multiple systems to respond to a victim, DNS amplification occurs due to the response packet being significantly larger than that of the query [8].

The addendum to this paper contains a detailed description of three of these attacks. DNS Amplification Attacks by Randal Vaughn and Gadi Evron, March 17, 2006[8].

DNS CACHE POISONING

DNS cache poisoning is a serious threat to today's Internet. DNS cache poisoning results in a DNS resolver storing (i.e., caching) invalid or malicious mappings between symbolic names and IP addresses. Because the process of resolving is a name depends on authoritative servers located elsewhere on the Internet. An attacker may poison the cache by compromising an authoritative DNS server or by forging a response to a recursive DNS query sent by a resolver to an authoritative server [9]. When you type a URL into your browser, a DNS resolver checks the Internet for the proper name/number translation and location. DNS will accept the first response or answer without question and send you to that site. It will also cache that information for a period of time until it expires, so upon the next request for that name/number, the site is immediately delivered. DNS won't need to query the Internet again and uses that address until that entry expires. Since users assume they are getting the correct information, it can get ugly when a malicious system responds to the DNS query first with modified, false information, as it does with DNS cache poisoning.

The DNS servers first send the user to the bad link but also cache that fake address until it expires. Not only does that single computer get sent to the wrong place, but if the malicious server is answering for a service provider, then thousands of users can get sent to a rogue system. This can last for hours to days, depending on how long the server stores the information, and all the other DNS servers that propagate the information can also be affected. The

imminent dangers posed by a rogue site include delivering malware, committing fraud, and stealing personal or sensitive information [11]. The nature of DNS cache poisoning attacks and present a precise, formal model of the bailiwick rule and the record overwriting mechanism of modern DNS resolvers, including BIND v9.4.1, unbound v1.3.4, and MaraDNS v1.3.07.09.
Step 1: The resolver checks the resolver cache in the workstation's memory to see if it contains an entry for Farpoint.companyA.com.
Step 2: Having found no entry in the resolver cache, the resolver sends a resolution request to the internal DNS server.
Step 3: When the DNS server receives the request, it first checks to see if it's authoritative. In this case, it isn't authoritative for companyA.com. The next action it takes is to check its local cache to see if an entry for Farpoint.companyA.com exists. It doesn't. So in Step 4 the internal DNS server begins the process of iteratively querying external DNS servers until it either resolves the domain name or it reaches a point at which it's clear that the domain name entry doesn't exist.
Step 4: A request is sent to one of the Internet root servers. The root server returns the address of a server authoritative for the .COM Internet space.
Step 5: A request is sent to the authoritative server for .COM. The address of a DNS server authoritative for the companyA.com domain is returned.
Step 6: A request is sent to the authoritative server for companyA.com. This is identical to the standard process for an iterative query – with one exception. A cracker has decided to poison the internal DNS server's cache. In order to intercept a query and return malicious information, the cracker must know the transaction ID. Once the transaction ID is known, the attacker's DNS server can respond as the authoritative server for companyA.com.
Although this would be a simple matter with older DNS software (e.g. BIND 4 and earlier), newer DNS systems have built-in safeguards. In our example, the transaction ID used to identify each query instance is randomized. But figuring out the transaction ID is not impossible. All that's required is time. To slow the response of the real authoritative server, our cracker uses a botnet (Botnets are groups of computers connected to the Internet that have been taken over by a hacker. The hacker controls all the computers and they behave like a "robot network") to initiate a Denial of Service (DoS) attack. While the authoritative server struggles to deal with the attack, the attacker's DNS server has time to determine the transaction ID.
Once the ID is determined, a query response is sent to the internal DNS server. But the IP address for Farpoint.companyA.com in the response is actually the IP address of the attacker's site. The response is placed into the server's cache.
Step 7: The rogue IP address for Farpoint is returned to the client resolver.
Step 8: An entry is made in the resolver cache, and a session is initiated with the attacker's site. At this point, both the workstation's cache and the internal DNS server's cache are poisoned. Any workstation on the internal network requesting resolution of Farpoint. companyA.com will receive the rogue address listed in the internal DNS server's cache. This continues until the entry is deleted [10].

DNS SPOOFING

DNS spoofing is another one of the man-in-the-middle attacks that can force victims to navigate onto a fake website purporting as a real one. DNS spoofing is based on the presentation of false or fake DNS information to the victim in a response to their DNS request and as a result forcing them to visit a site which is not the real one. As an example, suppose the user requests the IP address of mail.yahoo.com which is supposed to be XX.XX.XX.XX. But the attacker would respond to the DNS query before the actual response arrives with a spoofed address of YY.YY.YY.YY. The user's system will make a connection request to YY.YY.YY.YY thinking that mail.yahoo.com is located at that IP address. So effectively the user is routed to a completely different site from the one which he or she was originally destined to navigate. Normal DNS communication occurs when the system request from the IP of a particular website and the DNS server responds back with the actual IP address of the website. The system then connects to the website through the IP address it received as a response. With DNS spoofing, the attacker intercepts the DNS request and sends out a response which doesn't contain the actual IP actual but a spoofed IP address. This means that the rather than connecting to the real website, the victim connects to a malicious website which can cause harm.[14]
Detailed Description of Spoofing from RFC 5452 and Bernhard muller, SEC Consult Vulnerability Lab,Vienna(a pdf file) [12].

NAME BASED AUTHENTICATION/AUTHORIZATION

Some applications, unfortunately spreader all over the Internet, make use of an extremely insecure mechanism: name based authentication/authorization. It is the case, for example, of the UNIX "r-commands" such as rlogin, rsh or rcp that use the concept of "remote equivalence" to allow the remote access to a computer. In these networks, system administrators or, even worse, users can declare the remote equivalence of two accounts on two different machines (e.g., by means of the files /etc/hosts.equiv or .rhosts). This equivalence associates two users of two different hosts simply on the basis of their names. The access to a remote computer is then granted if the remote user is declared equivalent to a local user, and if the requesting hostname matches the one contained in the equivalence definition. No other authentication mechanisms are used, so we can talk of name based (weak) authentication. As an example, user joe can login as the user doe to the computer host.mydomain.com from the computer otherhost.mydomain.com if the file /etc/hosts.equiv contains the equivalence between the local user doe and the user joe@otherhost.mydomain.com. Remote commands have been designed at the dawn of the Internet for the use in trusted local network, where all the users were known to the system administrator, and the network was not connected to the big Internet. Unfortunately, remote commands survived to the Internet growth and they are still present and used in many networks. If name based authentication/authorization is used, it is possible to access to a remote machine simply spoofing the name of a host. Also, if the local network is protected by a firewall, all the hosts that use name based authentication/ authorization are at risk if an attacker can get control of a single machine of the firewall-protected network. The attacker can monitor network traffic learning the equivalences used in that network, and spoof the IP address of an equivalent host (e.g., performing a denial of service attack on that machine, or simply waiting for the machine to shut-down). Now, the attacker's host is completely equivalent to the spoofed host for all the computers using remote equivalence.

MISDIRECTED DESTINATION: TRUSTING FAKED INFORMATION

Suppose the following scenario: a user wants to connect to host A by means of a telnet client. The telnet client asks through a resolver the local name server to resolve the name A into an IP address, it receives a faked answer, and then initiates a TCP connection to the telnet server on the machine A (so it thinks). The user sends his login and password to the fake address. Now, the connection drops and the user retry the whole procedure this time to the correct IP address of the host A. He might ignore what just happened but the malicious attacker that spoofed the name of the host A is now in control of his login and password. This happened because the present routers have no capacity to disallow packets with fake source addresses. So, if the attacker can route packets to someone, then he is capable of forging those packets to look as if they come from a trustworthy host. Therefore, in our case the attacker predicts the time when a query will be sent and he starts to flood the resolver with his fake answers. With a firewall for the user's network the resolver would not be reachable from the outside world, but his local name server would. So, if the local name server can be corrupted in thesame manner as described above then the attacker can redirect such application with vital information towards hosts controlled by him and capture this information. Following these assumptions, we observe that in this case we have the possibility of a Denial of Service (DoS) attack. In case of such an attack, if the name server can be spoofed and the attacker's machine can impersonate the true name server then it can maliciously provide that certain names in the Domain does not exist. Later on, we present a way in which such an attack is annihilated in DNSSEC.

REFERENCES

1. http://www.webopedia.com/TERM/D/DNS. html.
2. http://searchnetworking.techtarget.com/definition/domain-name-system.
3. http://compnetworking.about.com/od/dns_domainnamesystem/f/ dns_servers.htm.
4. Roberto Perdisci, Manos Antonakakis, and Wenke Lee, "Solving the DNS Cache Poisoning Problem Without Changing the Protocol", May 16, 2008.
5. Tom Olzak, "DNS Cache Poisoning: Definition and Prevention", March 2006.
6. http://compsec101.antibozo.net/papers/dnssec/dnssec.html.
7. R.Arends,R.Austein and M.Larson ," DNS Security Introduction and Requirements",RFC 4033,March 2005.
8. Randal Vaughn and Gadi Evron, "DNS Amplification Attacks",March 17, 2006.
9. Matsuzaki Yoshinobu, "DNS amplification attacks",April 25,2006.
10. Tom Olzak, "DNS Cache Poisoning:Definition and Prevention",March 2006.
11. Peter Silva (Technical Marketing Manager), "DNSSEC: The Antidote to DNS Cache Poisoning and Other DNS Attacks",With contributions from(Nathan Meyer, Product Manager Michael Falkenrath, Senior Field Systems Engineer).

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

12. Bernhard muller,SEC Consult Vulnerability Lab,Vienna,improved DNS Spoofing using Node Re-Delegation,July 14,2008.
13. Stevens, Glenn. “The Domain Name Service”. June 21, 1995. URL: http://eeunix.ee.usm.maine.edu/guides/dns/dns.html
14. Http://www.securitysupervisor.com/security-q-a/network-security/195-what-is-dns-spoofing.