BERKELEY INTERNET NAME DOMAIN (BIND)

Shelena Soosay Nathan, Sanjaav Selan Mohan, Adelin Rose Harudas and

Kashif Nisar,

InterNetWorks Research Lab
School of Computing, College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok, MALAYSIA
shelen_smileystarz@yahoo.com

ABSTRACT

The Domain Name System (DNS) is a hierarchical distributed naming system for computers, services, or any resource connected to the internet or a private network. It associates several of information with domain names assigned to each of the participating entities. Moreover, DNS distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating authorities name servers for each domain. In general, the DNS stores other types of information, such as the list of mail servers that accept email for a given Internet domain. It also specifies the technical functionality of this database service and as a part of the Internet Protocol Suite. The Berkeley Internet Name Domain (BIND) is the most widely used DNS software. It was originally written by four graduate students at the Computer System Research Group at the University of California, Berkeley (UCB). The original name is Berkeley Internet Name Domain. Current solution to select the proper operating system that is BIND for DNS server with configures the server in to the BIND operating system. Besides that, used to run the server and find out the DNS server IP addresses

KEYWORDS

BIND, DNS server, DNS protocols, RAD, UCB

1. INTRODUCTION

BIND (Berkeley Internet Name Domain) is an implementation of the DNS protocols and provides an openly redistributable reference implementation of the major components of the Domain Name System, including Domain Name System server, Domain Name System resolver library and tools for managing and verifying the proper operation of the DNS server. It provides a robust and stable platform on top of which organizations can build distributed computing systems with the fully compliant with published DNS standards. Moreover, BIND is open source software that implements the Domain Name System (DNS) protocols for the internet. The BIND DNS Server, “named”, is used on the vast majority of name serving machines on the Internet, providing a robust and stable architecture on top of which an organization's naming architecture can be built [1-4].

The resolver library included in the BIND distribution provides the standard APIs for translation between domain names and Internet addresses and is intended to be linked with applications requiring name service. Besides that, it is reference for implementation of those protocols, but it is also production of grade software suitable for use in high volume and high reliability applications. BIND is available for free download under the terms of the ISC License that is BSD style license [5-7].

1.1 Problems

It is a typical problem in organizations that are growing that they have to resolve two problems at once; to have a DNS server for the internal network of the company because long ago there were already too many computers to remember their IPs and even too many computers to maintain a set of host files.

To have a DNS server for the external servers, for external clients, etc. To solve this problems become a bigger problem when the growing organization can't supply more resources than one DNS server. It is a bigger problem because if you just configure your server with all your names, public and private, you'll end up polluting the Internet with private addresses, something that is very bad, and also showing the world part of the topology of your internal network. Something you don't want a possible attacker/cracker to have [8-10].

The other part of the problem is that for efficiency you may want to resolve to internal IPs when you are inside and external IPs when you are outside. Here I am taking about computers which have public and private connections. There are many different solutions to this problem and I remember solving it even with BIND4, but now I am going to use BIND9 to make a solution that is very clean. This was deployed in a Ubuntu 10.10 server but it should also work for other operating systems that run BIND9, just be sure to change your paths appropriately.

2. BIND CONFIGURATION

```
$ sudo passwd root

Password: (Enter the password for current user)

Enter new UNIX password: (Enter the password you want to set for root)

Retype new UNIX password: (Retype root password)

passwd: password updated successfully

$ su -

Password: (Enter root password here)

# apt-get update; apt-get upgrade

# apt-get install bind9

zone "linux.lan" {

type master;
file "/etc/bind/zones/linux.lan.db";

};zone

 "0.10.10.in-addr.arpa" {

type master;

file "/etc/bind/zones/rev.0.10.10.in-addr.arpa";

};

# mkdir /etc/bind/zones

linux.lan. IN SOA ns1.linux.lan. admin.linux.lan. (

2006081401

28800

3600

604800

38400 )

linux.lan. IN NS ns1.linux.lan.

IN A 10.10.0.77

mail.linux.lan. IN MX 10 mail.linux.lan.

linux.lan. IN MX 10 mail.linux.lan.


www IN A 10.10.0.77

mail IN A 10.10.0.77

ns1 IN A 10.10.0.77

# host linux.lan 127.0.0.1

linux.lan has address 10.10.0.7

linux.lan mail is handled by 10 mail.linux.lan.
```

```
# dig linux.lan

; QUESTION SECTION:

;linux.lan. IN A


;; ANSWER SECTION:

linux.lan. 38400 IN A 10.10.0.77


;; AUTHORITY SECTION:

linux.lan. 38400 IN NS ns1.linux.lan.


;; ADDITIONAL SECTION:

ns1.linux.lan. 38400 IN A 10.10.0.77
```

The BIND DNS Server, named, is used on the vast majority of name serving machines on the Internet, providing a robust and stable architecture on top of which an organization's naming architecture can be built. The resolver library included in the BIND distribution provides the standard APIs for translation between domain names and Internet addresses and is intended to be linked with applications requiring name service.

2.1 Differences in BIND8 and BIND9

Apart from being multi-threaded, and a complete code rewrite - which should provide better stability and security in the long term, there are other differences If there is a syntax error in named.conf, BIND9 will log errors and not reload the named server. BIND8 will log errors and the daemon will die! Extensive support of TSIGs (shared keys) for access control, for example, "update-policy" can be used for fine grained access control of dynamic updates. The tool for starting/stopping/reloading etc., rndc is different from the v8 ndc - different communications, authentication and features.

- Syntax in zone files is more rigorously checked (e.g. a TTL line must exist)
- In named.conf
- V8 options 'check-names' and 'statistics-interval' are not yet implemented in V9.

- The default for the option ‘auth-nxdomain’ is now ‘no’, if you don’t set this manually, BIND 9 logs a corresponding message on startup.
- The root server list, often called named.root or root.hints in BIND8 is not necessary in BIND 9, as it is included within the server.

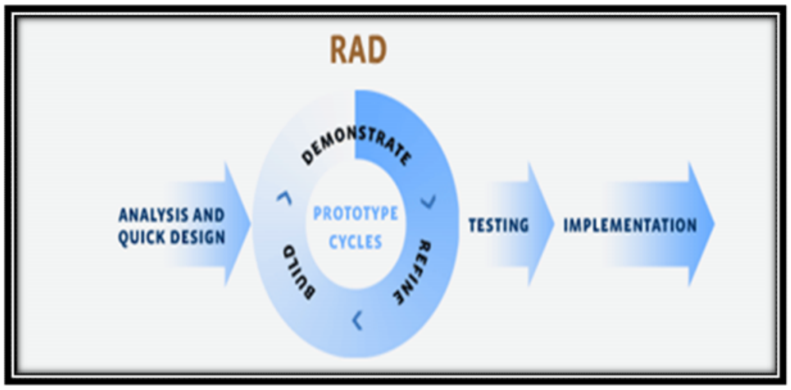

Figure 1. RAD is a software development methodology

3. METHODOLOGY

RAD is a software development methodology that uses minimal planning in favour of rapid prototyping. The planning of the software developed using RAD is interleaved with writing the software itself. The lack of extensive pre planning generally allows software to be written much faster, and makes it easier to change requirements. Structured techniques and prototyping are especially used to define user’s requirements and to design the final system. The development process starts with the development of preliminary data models and business process models using structured techniques. As below Figure 1 shows RAD methodology.

In the next stage, requirements are verified using prototyping, eventually to refine the data and process models. These stages are repeated iteratively, further development results in a combined business requirements and technical design statement to be used for constructing new systems. RAD approaches may entail compromises in functionality and performance in exchange for enabling faster development and facilitating applications maintenance. Furthermore, RAD promotes strong collaborative atmosphere and dynamic gathering of requirements. RAD contains four phases that is requirements planning phase, user design phase, construction phase and cutover phase.

3.1. How DNS works

Suppose that local client wishes to learn the IP address of `www.sans.org`. The client contacts a local name server which has been configured on the local client by the administrator (statically or via DHCP, etc.). The local DNS server actually does all of the work required to resolve the IP address and then will hand the result back to the client. As below Figure 2 shows DNS works.

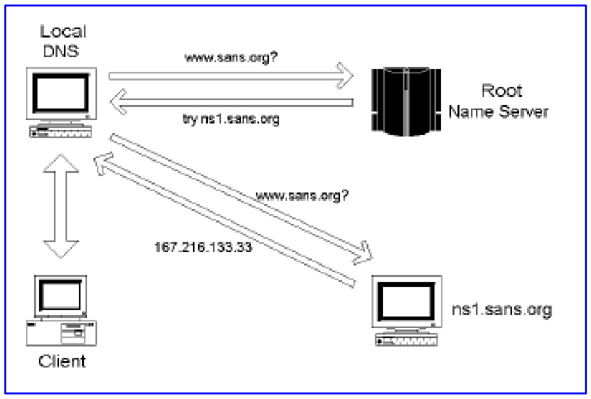

Figure 2. Process of DNS works

The local name server first attempts to contact one of the several *root name servers* that have been deployed on the Internet. Root name servers maintain a mapping between domains (`sans.org`) and name servers (`ns1.sans.org`)– when your local name server asks for the IP address of `www.sans.org`, it receives a *referral* from the root name servers which essentially says "unable to answer your question, but here is the name/address of somebody who can". In order to be able to contact a root name server, your local name server must be statically configured with the names and IP addresses of the available root name servers.

This information is maintained by the InterNIC and downloaded by the administrator into a static file on the local name server. Having received the names and IP addresses of the name servers for `sans.org` from the root name server, your local name server then contacts one of these machines and asks for the IP address of `www.sans.org`. The name server for `sans.org` returns the IP address to your local name server and the local name server hands the information back to your client.

3.2. Security Issues

The primary risk with running DNS is that you give away too much information that can be used by people who wish to attack your systems and networks. BIND has historically had buffer overflow problems in various releases. Some have led to root compromise attacks; others have simply been denial-of-service type attacks. The best defences against these attacks is to stay up to date on the version of BIND you are running, though the *Running a Name Server* section suggests how to configure BIND to run in a `chroot`()ed environment, which can help protect you in the event of an exploitable buffer overflow. As below Figure 3 shows cache poisoning.

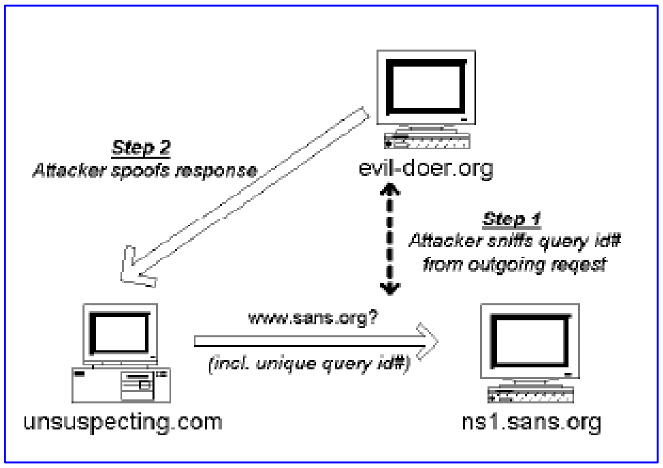

Figure 3. Cache Poisoning – Spoofing

Cache poisoning occurs when a name server has been tricked into believing erroneous information from some external source. Sometimes this occurs by accident, but most often it is used by attackers who wish to embarrass an organization or exploit trust relationships based on hostname/address information. As below Figure 4 shows unauthorized zone transfer

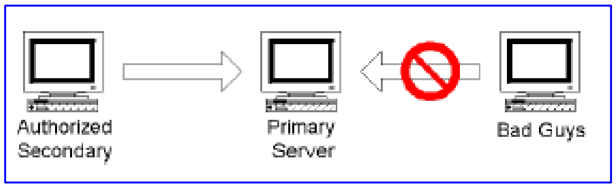

Figure 4. Block Unauthorized Zone Transfer

Generally, each organization runs one master DNS server and one or more slave servers for redundancy. Periodically, the slaves must contact the master and download any updates to the local DNS database– this is referred to as a *zone transfer*. By default, name servers running BIND allow any remote system to perform a zone transfer– whether that system is a legitimate name server for that domain or not. Zone transfers can even be requested from the slave name servers for the domains. Attackers often attempt zone transfers in order to gather information about client local network. If they succeed then they have instantly gotten all of the information about your internal hosts and networks with very little effort. Of course, a split horizon DNS configuration can limit the amount of information an attacker will receive, but it is still a good idea to prevent unauthorized hosts from downloading from zone databases. Figure 5. Show the DNS Firewall Architecture.

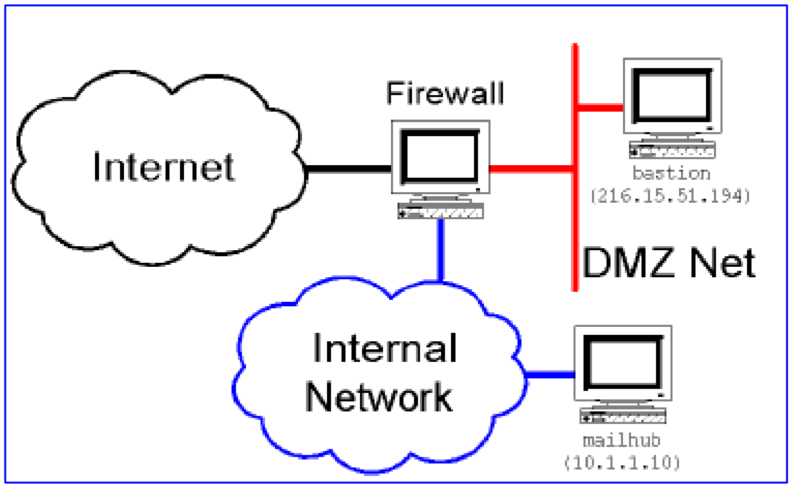

Figure 5. DNS Firewall Architecture

Typical DNS firewall configuration in use at many organizations today. The organization has a multi-legged firewall which connected the external Internet to both a semi-private de-militarized zone (DMZ) network and a private internal corporate network. The DMZ network is where an organization would put its Web and FTP servers and any other machines that the outside world needed to reach– and for purposes of this example a machine called bastion which will be the "external" DNS server for our split-horizon example. On the internal network there is a machine mailhub which acts as the primary server for the "internal" DNS.

3. Conclusions

BIND 4 and BIND 8 have both had a substantial number of serious security vulnerabilities over the years, and as such their use is now strongly discouraged. While BIND 9 was a complete rewrite, ostensibly to mitigate these on-going security issues, it has also experienced a large number of serious security vulnerabilities Bind listens on port 53 UDP and TCP. TCP is normally only used during zone transfers so it would appear that you could filter it if you have no slaves. However If the response to a query is greater than 1024 bytes, the server sends a partial response, and client and server will try to redo the transaction with TCP. Responses that big do not happen often, but they happen. And people do quite often block 53/tcp without their world coming to an end. But this is where one usually inserts the story about the Great DNS Meltdown when more root servers were added. This made queries for the root list greater than 1024 and the whole DNS system started to break down from people violating the DNS spec (RFC1035) and blocking TCP.

Acknowledgements

The authors would like to thank Universiti Utara Malaysia for research work.

REFERENCES

[1] A. Gulbrandsen, P. Vixie, A DNS RR for specifying the location of services (DNS SRV), October 1996

[2] BCP 20, H. Eidnes et. al. Classless IN-ADDR.ARPA delegation, March 1998. This is about CIDR, or classless subnet reverse lookups.

[3] C . Farrell, M. Schulze, S. Pleitner, D. Baldoni, DNS Encoding of Geographical Location, 11/01/1994.

[4] D. Barr, Common DNS Operational and Configuration Errors, 02/28/1996.

[5] Gizem, Aksahya & Ayese, Ozcan (2009) Coomunications & Networks, Network Books, ABC Publishe

[6] Lee, S.hyun. & Kim Mi Na, (2008) “This is my paper”, ABC Transactions on ECE, Vol. 10, No. 5, pp120-122

[7] M. Andrews, Negative Caching of DNS Queries, March 1998. About negative caching and the $TTL zone file directive.

[8] P. Vixie, Extension Mechanisms for DNS (EDNS0) August 1999

[9] R. Ullmann, P. Mockapetris, L. Mamakos, C. Everhart, New DNS RR Definitions, 10/08/1990.

[10] Y. Rekhter, R. Moskowitz, D. Karrenberg, G. de Groot, E. Lear, Address Allocation for Private Internets, 02/29/19

Authors

Shelena Soosay Nathan was student of Northern University of Malaysia, Sintok, Kedah Darul Aman, Malaysia. She was an undergraduate from School of Computing, Northern University of Malaysia, Sintok, Kedah Darul Aman,Malaysia in Bachelor of Information Technology majoring Networking (Hons).

Sanjaav Selan Mohan was student of Northern University of Malaysia, Sintok, Kedah Darul Aman, Malaysia. He was an undergraduate from School of Computing, Northern University of Malaysia, Sintok, Kedah Darul Aman, Malaysia in Bachelor of Information Technology majoring Networking (Hons).

Adelin Rose Harudas was student of Northern University of Malaysia, Sintok, Kedah Darul Aman, Malaysia. She was an undergraduate from School of Computing, Northern University of Malaysia, Sintok, Kedah Darul Aman,Malaysia in Bachelor of Information Technology majoring Networking (Hons).

Dr. Kashif Nisar received his first degree Bachelor in Computer Science from Karachi, Pakistan and his MS degree in Information Technology from the Universiti Utara Malaysia (UUM). He received his PhD degree in Computer Networks (specializing in WLANs, VoIP, IPv6 and he proposed a novel Voice Priority Queue (VPQ) scheduling system model) from the Universiti Teknologi PETRONAS (UTP) in the Malaysia. Dr. Kashif's research interests are in multi-disciplinary areas, including Real-time Multimedia Technologies, Network Performance, Network Traffic Engineering, Wireless and Mobile Networking, Local Area Networking, Telecommunications/Network Management, Wireless Sensor Networks (WSNs), Mobile Ad Hoc Networks, Audio/Video Transport Extensions, Call Control UUI Service for SIP, Audio/Video Transport Payloads, VoIP, IPv6, and evaluation of wireless networks. He is currently working as a Senior Lecturer at School of Computing, Collage of Arts and Science, Universiti Utara Malaysia, Malaysia. He is also serving as a Senior Member of InterNetWorks Research Group.