Title: # GCIH Practical Assignment

DNS Security

Author: Thomas A. Greco

Version Number: 2.1a: Option 2 – Support for the Cyber Defense Initiative

Date: March 13, 2003

Abstract

The Domain Name Service (DNS) is probably one of the most critical yet under-appreciated services on the Internet. Almost every Internet user today, consumer or business, is highly dependent upon its operation. Until recent years, little attention was paid to the security of this important resource. Vulnerabilities in DNS span its architecture, protocol, and implementation. This paper provides an overview of DNS, outlines the major vulnerabilities, and details a particular exploit where DNS itself can be used to launch a Denial of Service attack.

Contents

Targeted Port: TCP/UDP 53

Introduction

According to data obtained from the Consensus Intrusion Database on February 12, 2003 port 53 was ranked 4th in the top 10 most-attacked ports on the Internet (Table 1 and Figure 1).

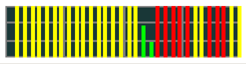
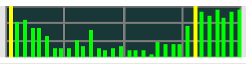
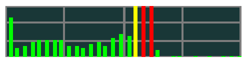
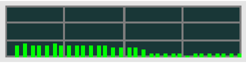
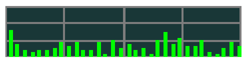
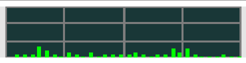
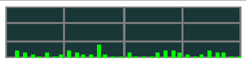
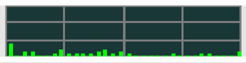
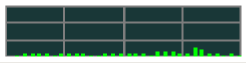
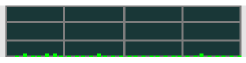
Service Name	Port Number	30 day history	Explanation
netbios-ns	137		
http	80		HTTP Web server
ms-sql-m	1434		
domain	53		Domain name system. Attack against old versions of BIND
ms-sql-s	1433		Microsoft SQL Server
microsoft-ds	445		
netbios-ssn	139		Windows File Sharing Probe
ftp	21		FTP servers typically run on this port
???	4662		eDonkey P2P software
smtp	25		Mail server listens on this port.

Table 1 CID Top 10 Ports – February 12, 2003

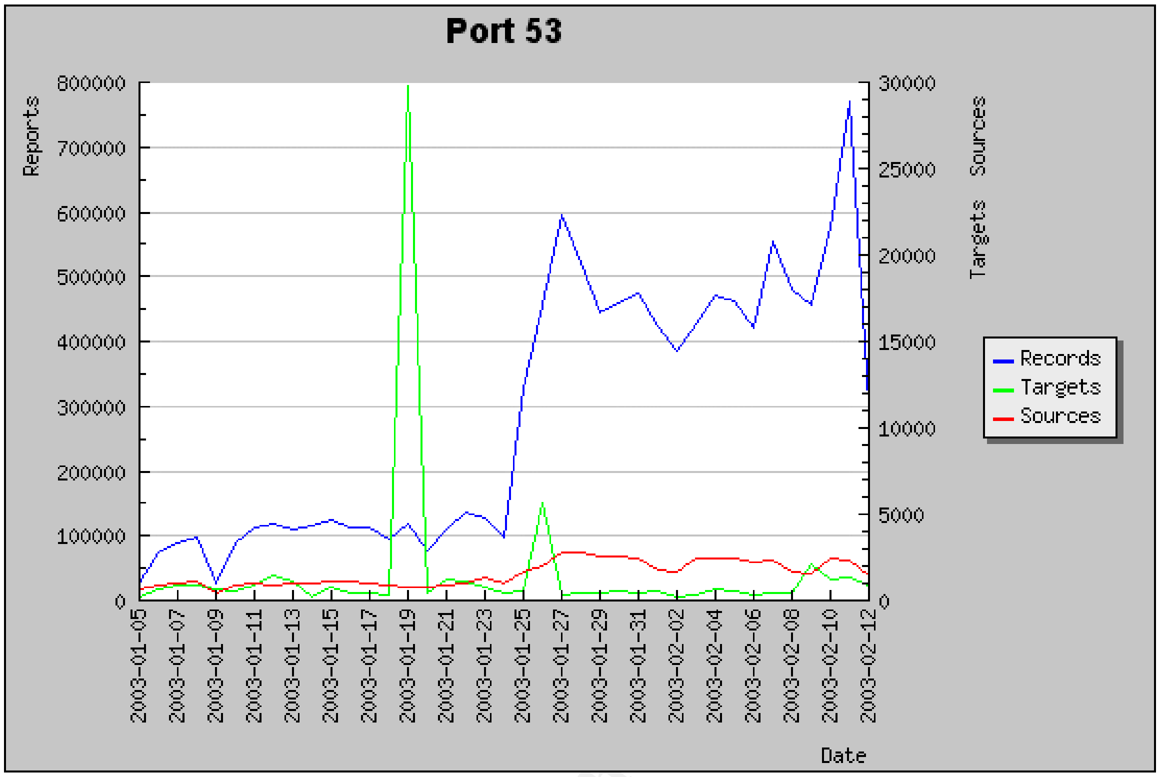


Figure 1 CID graph of port 53 activity around February 12, 2003.

Ports TCP/UDP 53 are commonly associated with the Domain Name Service (DNS). DNS is used to resolve Internet Protocol (IP) addresses from hostnames and back again. A numeric IP address is required to access a resource on an IP-based network such as the Internet. Each device on a network is assigned an IP address having the form x.x.x.x where each instance of "x" represents an 8-bit number (decimal ranging from 0 to 255). Details of the IP protocol are beyond the scope of this assignment. Several excellent tutorials on this topic exist. A few are listed in the References section. For the moment, assume that the IP addresses are properly constructed. Users must know the mapping between the devices and the IP addresses to be productive on the network. For example, they must know that 1.2.3.4 is the mail server, or that 1.2.3.5 hosts the company's financial applications. It is precisely the problem of this mapping that eventually led to the development of DNS.

Before the existence of DNS, it was necessary to either remember all of the numeric addresses, or to assign names to the devices, and maintain hostname to IP address mappings in files on every device. As the Internet grew, remembering all of the numbers was impossible. The concept of naming the devices was much more appealing. In general, humans are much better at remembering names than they are at remembering numbers. However, maintenance of the mapping files was daunting even for a moderate sized network.

The need for a naming system was recognized quite early in the Internet's history. In 1971, Peggy Karp conceived the concept of standardized host designators or Internet names. She followed this a year later with the creation of the HOSTS.TXT file which contained hostname to IP address mappings in a standardized text format. System administrators were responsible for submitting their entries for inclusion in HOSTS.TXT to the Stanford Research Institute. Stanford maintained the authoritative copy of the file and made it available globally via FTP.

By the early 1980's, maintenance of HOSTS.TXT was becoming unwieldy. In response, a series of RFC's beginning with the work of Dr. David Mills collectively identified the basic building blocks of modern day DNS. They defined concepts such as domain names, sub-domains, zones, authority, and delegation. In addition, as the network morphed from ARPAnet to NSFnet to, eventually, the Internet, components of the formal DNS infrastructure such as the governing bodies, top-level domains, and root name servers began to solidify. In 1992, in what, in retrospect, could be considered to a landmark event, Network Solutions, Inc. was awarded the contract from the National Science Foundation for management of the entire domain system placing control of this vital resource into the hands of the private sector once and for all.

Today, DNS is THE index to the Internet. Its care and feeding is distributed on a global basis. Most take for granted that DNS will be there to do its job. Few are even aware of its existence at all. When one types www.ebay.com into their web browser, it is expected that ebaY's home page will magically appear. There is no thought that the web browser had to query a DNS server to find out that www.ebay.com resolves to 66.135.192.83 (among 4 addresses at the time of this writing), or that their local DNS server probably had to make further queries on their behalf to figure this out.

Without DNS, the Internet would become almost completely useless to a majority of the networked community. Given the level of commercialization and the reliance on the Internet by businesses and consumers alike, DNS has undoubtedly become very important to the stability of the economy. Arguably, it is as important as the Internet itself.

Architecture

The design goals of DNS center around a database that is stored and managed in a distributed fashion for the purpose of communicating hostname to IP address mappings on a global basis. The components of DNS can be divided into two major categories: infrastructure and protocol. The elements of the DNS infrastructure include the root name servers, top-level domain servers, The Internet Corporation for Assigned Names and Numbers (ICANN) and multiple domain registrars. Beyond these there exists a multitude of resolvers and lower-level name servers. At the heart of all of these elements is the DNS protocol itself defining the name space and the rules by which it is stored and communicated.

Protocol

Domain Name Space and Domains

DNS defines a distributed database of domain names and IP addresses. The structure of this database is commonly termed the *Domain Name Space*. A sample of the Domain Name Space structure is depicted in Figure 1. It is a nodal structure beginning with a single parent node, the root node, labeled by a period, ".". The root node has several children each of which may have children of their own. Thus, with the exception of the root node, any node in the structure may simultaneously exist as both a parent and child.

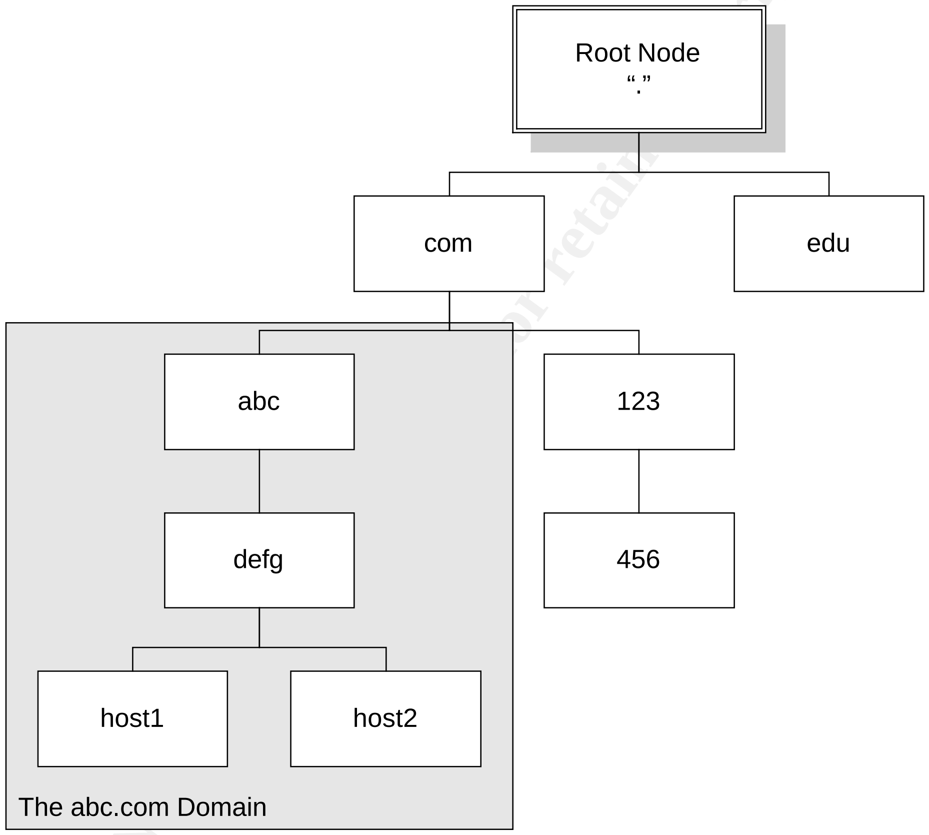


Figure 2 Domain Name Space structure.

Domain names are constructed by starting at any node and following the structure up to the root node. Simply concatenate the node labels using a period as a separator. Referring to Figure 1, some examples of domain names would be "defg.abc.com." or "123.com.". A domain name constructed by starting from a node possessing no children is known as an absolute domain name or *Fully Qualified Domain Name* (FQDN). Again referring to Figure 1, an example FQDN would be "host1.defg.abc.com.".

A group of nodes descendent from a single node comprise a *domain*. For example, in Figure 1, "abc.com" represents a domain. It contains a *sub-domain*, "defg.abc.com", that possesses two FQDN's, "host1.defg.abc.com" and "host2.defg.abc.com".

Delegation, Authority and Zones

The actual content making up the domain name space did not spring spontaneously from the vacuum of deep space. Various organizations are responsible for defining the nodes at every level. Responsibility is assigned on a domain basis. Specific organizations, such as ICANN, are responsible for the very top-level domains. ICANN and the concept of top-level domains are discussed further in the Infrastructure section. Below the top level, responsibility is delegated to the organizations having direct interest in a particular domain. This assignment of responsibility from a higher-level domain to a lower one is called *delegation*.

A university would have responsibility for their domain (e.g. university.edu) delegated to them from the keeper of the edu domain. The university and their respective name servers thus have *authority* for the university.edu domain. In this scenario, university.edu is also referred to as a *zone*. A zone, in other words, is a domain for which authority has been delegated.

Class Types and Resource Records

The existence of nodes in a domain name space is not in itself a very useful construct. Recall, however, that among other things DNS is a database. Underlying the nodal structure of the name space are records holding the relevant content of the associated domains. As is true for just about any database, DNS has specific rules regarding the structure and content of the records in its database.

In general, individual records are termed *Resource Records* (RR). Each resource record contains the following information:

- Domain name
- Class
- RR type
- Time-To-Live (TTL)
- Resource data length
- Resource data (dependent upon Class and RR type)

The class types are Internet, Hesiod, and CHAOS. Hesiod and CHAOS are rarely used anymore. The discussion from here forward will therefore be limited to records of the class IN. The TTL parameter specifies the time interval for which the record can be cached. Caching is discussed further in the Infrastructure section below. The resource data length simply specifies the length of the resource data section of the RR. The resource data section itself contains the specific information based upon both the class and RR type.

Although there are many more, Table 2 contains some commonly referenced RR types of the IN class and their definitions.

RR TYPE	DEFINITION	PURPOSE
A	Host address	Identifies the IP address of a host.
CNAME	Alias to a canonical name.	Identifies an alias to the real name (canonical name) of a host.
HINFO	Host information	Specifies additional, non-functional, information pertaining to a host.
MX	Mail exchange	Identifies a host acting as the mail server for the domain.
PTR	Reverse Record	Associates an IP address with a canonical name.
SOA	Start of Authority	Specifies the start of a zone of authority.

Table 2 Common RR types of the IN class.

For details of all RR types and their formats, refer to RFC 1035.

<u>Message Formats and Transports</u>

DNS functions based upon query and response messages. Regardless of the message type, all DNS messages share a common high-level structure. Figure 2 shows the DNS message format adapted from RFC 1035.

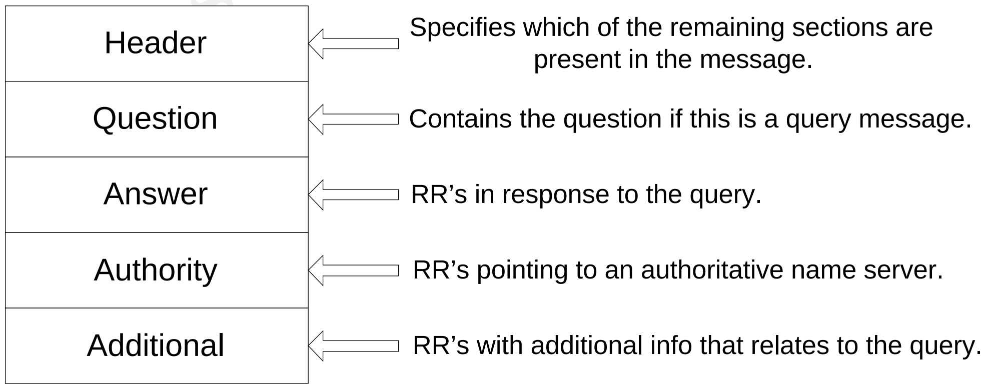

Header
Specifies which of the remaining sections are present in the message.
Question
Contains the question if this is a query message.
Answer
RR's in response to the query.
Authority
RR's pointing to an authoritative name server.
Additional
RR's with additional info that relates to the query.

Figure 3 DNS message format.

Not all sections are present in every message. The Header section must always exist. The Header will specify the content of the message indicating which of the other sections are actually present. A query, for example, will contain only the Header and Question sections. Further details may be obtained from RFC 1035.

The transport mechanisms for DNS messages are the IP protocols UDP and TCP utilizing port 53 in both cases for standard implementations. Queries are typically handled via UDP because of its fire-and-forget nature. There is low overhead with UDP, and response times will be minimized if everything is working correctly. However, a retransmission mechanism is essential since UDP-based queries and their responses may be lost. The DNS RFC's do not specify how the retransmission should function. This decision is left to the developers of the particular implementation. The BIND resolver, for example, will send a query 3 times at 0, 12, and 24 seconds.

TCP is recommended for DNS functions requiring a more reliable transport. Copying all RR's for an entire zone from one name server to another, known as a zone transfer, is an example of a transaction best served by using TCP. It is important to know in this case that the either the entire zone has been transferred or that the communication as been severed at some point along the way. TCP is also used when responses are truncated by the 512 byte limit of the DNS UDP datagram.

Infrastructure

Name Servers, Resolvers and Resolution

Generally speaking, the DNS infrastructure consists largely of a hierarchy of servers that follows the hierarchy of the domain name space itself. These *name servers* store domain (zone) information and respond to queries. Name servers store RR's of the zones for which they are authoritative by either reading them from *zone files* located on a filesystem, or by transferring them from other name servers that are authoritative for the same zone.

Queries originate from entities known as *resolvers.* Resolvers are available to applications (e.g. telnet, ftp, web browser) on a host system through libraries or other facilities. Resolvers are used to formulate the queries, send them to name servers, and then interpret the responses.

Queries can be either iterative or recursive. If a resolver issues an iterative query, the name server will respond with a referral to another name server if it doesn't know the answer. In this case, the resolver itself will have to initiate further queries. If the resolver issues a recursive query, the name server will assume responsibility for making further queries on behalf of the resolver. For most implementations, the entire resolution process is both recursive and iterative as show in Figure 3. The resolver will make a recursive query to its local name server. The name server will, in turn, execute a series of iterative queries in search of the answer and return the answer to the resolver.

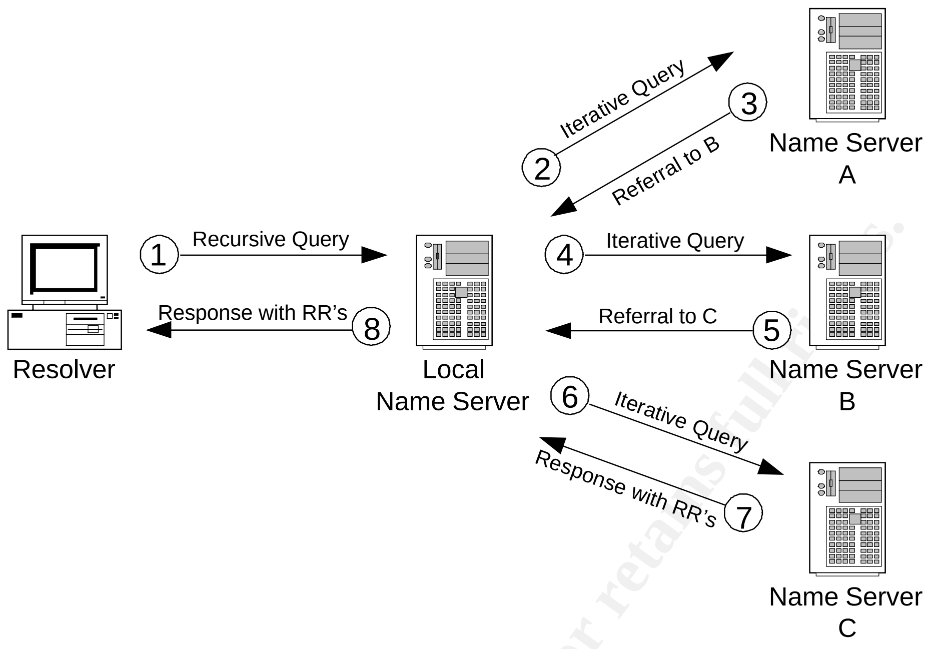


Figure 4 Resolution process demonstrating iterative and recursive queries.

Name servers can also store information about zones for which they are not authoritative through caching. As the name server executes the iterative queries in Figure 3, it will store the responses it receives in cache. The name server will retain these RR's in its cache for the time period specified in the TTL parameter that was included in each response it receives from the other name servers. Subsequent responses to queries from the resolver for the same RR's will be read from cache. Caching is a performance-enhancing design element intended to increase response times and reduce bandwidth utilization as well as the load on the name servers.

Root Servers and Top-Level Domains

Just as the hierarchy of the domain name space is arranged in a "top-down" fashion, so is the hierarchy of name servers. At the top of the domain name space is the root domain. At the top of the domain name server hierarchy are the root servers. Presently there are 13 root servers named x.ROOT-SERVERS.NET where x ranges from A to M. The number of root servers is limited to 13 because more would cause an NS response for the root servers to be truncated (recall the 512 byte limitation of DNS UDP datagrams).

The root servers contain information about the name servers that are authoritative for the *top-level domains* (TLD). TLD's are nodes in the domain name space that are direct descendants of the root node. They are split into two categories. There are the 14 generic TLD's (gTLD) such as .com, .org, .net, .mil, and the over 240 country code TLD's (ccTLD) as in .us, .de, .ch, .jp. The TLD name servers are the registries containing RR's of the name servers authoritative for the second-level domains (e.g. icann.org, odu.edu).

Referring to Figure 3, in a basic resolution scenario, Name Server A would represent a root server, Name Server B would represent a TLD name server, and Name Server C would represent the name server authoritative for the domain associated with the query. Note that there are alternate resolution scenarios. Consult the references for more details.

ICANN and Registrars

Root servers, domain name space, top-level domains… but who's in charge of it all? The Internet Corporation for Assigned Names and Numbers (ICANN) is a non-profit organization responsible for management of DNS at the highest level. Among other things, ICANN establishes policies for DNS management, and manages all aspects of the root zone. In addition, ICANN delegates responsibility for operating the TLD name servers to various organizations often referred to as registry operators. For example, Verisign, Inc. is responsible for operating and maintaining the .com TLD name servers. A complete list can be found at http://www.icann.org/tlds.

As anyone who has ever registered a domain would know, there are a multitude of *registrars* working in conjunction with the TLD registry operators to provide domain registration services. ICANN also has the responsibility of accrediting registrars. The accreditation process can also be found on the ICANN web site.

Vulnerabilities

Consider a cursory business impact analysis using the following scenario:

Business Objective = Establish an e-business selling widgets over the Internet

- Task 1 = On-line research for widget marketing and sales strategy
  - Asset 1.1 = Network infrastructure
  - Asset 1.2 = Resolvers, widgets.com name servers
- Task 2 = Widget engineering and manufacturing
  - Asset 2.1 = Network infrastructure
  - Asset 2.2 = Engineering and manufacturing systems
  - Asset 2.3 = CAD data
- Task 3 = Sell widgets
  - Asset 3.1 = Network infrastructure

Asset 3.2 = Web servers, application servers, database servers

Asset 3.3 = Applications, data

Asset 3.4 = widgets.com external name servers, zone files (i.e. DNS content pertaining to widgets.com zone)

Mapping objectives to tasks to assets demonstrates the impact that a particular asset or set of assets has on the objectives. In this case, it is clear that the organization's DNS infrastructure is fairly critical since it facilitates two tasks associated with the primary objective (Assets 1.2 and 3.4).

Using confidentiality, integrity, and availability as the review elements and a simple High, Medium, Low rating scale the impact can be evaluated qualitatively. Qualitative analysis is simple and the results are useful for prioritizing the application of risk analysis and the subsequent implementation of controls.

Regarding confidentiality, ask the question:

What would be the impact on the business objective if DNS content pertaining to widgets.com was disclosed to an unauthorized individual with or without malicious intent?

Regarding integrity, ask the question:

What would be the impact on the business objective if the integrity of the DNS content were compromised?

Regarding availability, ask the question:

What would be the impact on the business objective if the DNS content were unavailable for one minute… one hour… one day… one week… or more?

Based upon the scenario above, these questions are easily answered. On a name server intended for queries by the general public, RR's are considered just that, public. There is no restriction on who can query the external name servers. Therefore, the concept of an "unauthorized individual" is difficult to define. An individual without malicious intent might be a customer who, probably unbeknownst to them, queries the widgets.com public name servers when accessing the web site. For an e-business, this is certainly not an unauthorized disclosure. An individual with malicious intent might be an attacker using the public name servers for reconnaissance in the early stages of attack planning. Still, the use by the attacker may not necessarily constitute unauthorized disclosure. Using standard query tools and techniques, the attacker can execute perfectly acceptable queries. Executing a zone transfer by circumventing access controls is an entirely different story. Yet the zone transfer itself would not disclose non-public information (unless RR's for internal hosts were loaded onto the external name servers). It merely provides an immediate aggregation of information that would be more difficult to collect otherwise. Thus, the overall impact due to a breach of confidentiality would be extremely low.

In contrast, a breach of integrity or a lack of availability would have a High impact. In both cases, business resources would be unreachable by customers, suppliers, or other partners. Any disruption in DNS, even for a short period of time, can be damaging to an e-business. A breach of integrity might even lead to fraud or other hostile activity if an attacker were able to modify RR's and direct legitimate customers to his own site.

Focusing then on integrity and availability issues, a risk analysis can be performed. First compile the list of threats.

Possible threats to the integrity of widgets.com DNS content are:

- Intentional or accidental modification of RR's stored in zone files.
- Intentional or accidental modification of RR's in memory or cache.
- Forgery of responses to legitimate queries.

Possible threats to the availability of widgets.com DNS content are:

- Root servers inaccessible.
- .com TLD name servers inaccessible.
- widgets.com external name servers inaccessible.

The associated priority of these threats is evaluated based upon a combination of their likelihood of occurrence and their impact if they were to actually occur. Assume that all carry a high impact. It then remains to determine their likelihood by judging how susceptible widgets.com would be to these threats in the absence of compensating controls. Only then can the identification and prioritization of controls be accomplished.

Regarding integrity, accidental modification of anything is probably a function of procedures and awareness. Widgets.com system administrators are much too professional for this to be a likely threat (or so we hope).

The remaining threats would all be realized either through vulnerabilities in the systems hosting DNS (widgets.com name servers, root servers, etc.), the applications that implement DNS (e.g. BIND), or the DNS protocol itself. Table 3 displays the correlations between these categories of vulnerabilities and the threats listed above along with some examples.

Category	Examples of Vulnerabilities	Associated Threats
DNS Systems	Remotely exploitable vulnerabilities resulting in system access, execution of commands, or denial of service Locally exploitable vulnerabilities resulting in elevation of privileges, execution of commands, or denial of service	Modification of zone files External name servers inaccessible Root servers inaccessible

		.com TLD name servers inaccessible
DNS Implementation	Remotely exploitable buffer overflows in BIND	Modification of zone files Modification of RR's in memory or cache External name servers inaccessible Root servers inaccessible .com TLD name servers inaccessible
DNS Protocol	Spoofing or hijacking Cache poisoning DDoS attack against name servers	Root servers inaccessible .com TLD name servers inaccessible Injection of false responses to legitimate queries

Table 3 Vulnerabilities by category.

<u>Systems</u>

The range of potential vulnerabilities in the DNS systems in general is large. With the high rate of discovery of new vulnerabilities in systems and applications, there is certainly plausibility that a name server could be compromised. It's probably safe to say that the root and TLD servers are less likely to be penetrated compared to corporate name servers such as widgets.com's external name servers. The root and TLD servers draw much attention for obvious reasons and there has been a significant effort in the past couple of years to harden these servers as well as strengthen the security procedures surrounding their maintenance. Corporate name servers do not have the benefit of being subject to stringent standards as a rule. While some corporations do an excellent job with security standards a large number are still lacking. The high rate of vulnerability discovery coupled with lapses in corporate security places the probability of this family of vulnerabilities in the medium to high range for the widgets.com external name servers. The low to medium range is more appropriate for the root and TLD servers.

<u>Implementation</u>

Vulnerabilities in DNS implementation are essentially a subset of the vulnerabilities in the DNS systems themselves since the implementation represents and application or service on the system. DNS implementations such as BIND are susceptible to the same flaws as any other system (e.g. buffer overflows, command injection, misconfiguration). Their exploitation can result in access to the system, a compromise of system integrity, or denial of service. Not to pick on BIND, but seeing how it is currently the most widely used implementation of DNS it lends itself to being a decent example. Reviewing the BIND security page at http://www.isc.org/products/BIND/bind-security.html reveals a variety of application-level vulnerabilities. The severity of these vulnerabilities ranges from mild annoyance to possible remote root compromise. Other DNS implementations will have their vulnerabilities as well. However, assuming that widgets.com is using BIND, the probability here could be classified as medium to high.

Protocol

Regarding the DNS protocol, the top-down architecture it imposes places significant criticality on upper-level servers. The root servers are certainly critical, but perhaps not as critical as the TLD's. Recall that the root servers provide the locations of the TLD servers. In addition, responses from the root servers will be cached for two days due to their TTL. Thus, it is highly probably that a local name server will have the locations of the TLD servers (at least those for whom the most queries are made) cached. Greater importance is then shifted to the integrity and availability of the TLD servers since they provide direction to the multitude of second-level domain servers. Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks have captured much publicity in recent months. Probability remains high that these types of attacks will continue to be perpetrated against DNS. However, the root and TLD servers are definitely going to be more resilient than the widgets.com external name servers.

Another aspect of the DNS protocol, which makes it particularly susceptible to certain types of attacks, is its use of UDP datagrams. UDP is connectionless and therefore does not rely on mechanisms such as sequence numbers to maintain the state of the connection. While sequence numbers are not meant to be a strong form of security, if they are sufficiently random they do impose a hurdle that must be cleared for spoofing or hijacking a connection. DNS queries using UDP have no such protection clearing the way for spoofing and hijacking attacks.

Even though UDP cannot prevent spoofing or hijacking, DNS does present one obstacle to such an attack. DNS employs a mechanism known as a query ID. It is essentially a 16-bit number used for matching responses to their original queries. However, the query ID has weaknesses that an attacker can leverage fairly easily. Although the query ID itself makes them possible, a particularly weak DNS implementation will ease the execution of these types of attacks. Attacks using the query ID take advantage of the fact that the query ID is really the only mechanism used to authenticate responses. Query ID attacks follow a common pattern. Recursive queries are generated, either by legitimate or forged requests. The attacker then injects the data of his choosing by formulating a spoofed DNS response and sending it to the victim name server. With sufficient knowledge of the query ID generation algorithm, or by using brute force techniques involving the generation of hundreds of responses, the attacker can trick the victim name server into accepting the bogus responses. Depending upon the particular DNS implementation, this can be achieved with a surprisingly high degree of success.

An attacker can also inject bogus data into a victim name server by configuring his own name server and loading it with falsified RR's. In this scenario, the attacker sets up a name server for the attacker.com zone. In his zone files he creates false RR's including a mapping of www.victim.com to the IP address of www.attacker.com. The attacker then sends recursive queries for RR's in the attacker.com zone to the victim.com name server. When the victim.com name server queries the attacker.com name server, the bogus RR's are returned as additional data to the response and stored in the victim.com name server's cache. Now victim.com's name server will resolve www.victim.com to the IP address of www.attacker.com thereby sending all victim.com traffic to the attacker's site.

The concept of delegation could be considered a vulnerability in the DNS protocol. Certainly, DNS would not be DNS without delegation. However, along with delegating the responsibility of managing a lower-level domain comes the delegation of responsibility for managing the security of its infrastructure. Below the TLD's, there are no set standards for operators of name servers. In other words, due to delegation the security of DNS below the TLD's rests in the hands of the masses for which no security standard could realistically be imposed. Given the number of second-level domains (thousands upon thousands), this set of servers represents a target-rich environment for exploiting DNS.

Overall, the probability of attacks on the DNS protocol relating to availability is relatively high. Query ID-based attacks directed toward integrity require slightly more expertise and higher effort to execute, and therefore carry a medium probability.

Using the priority model in Table 4, the threats can be prioritized as follows:

Threat	Priority
Widgets.com external name servers inaccessible.	A
Intentional or accidental modification of RR's stored in zone files.	A
Forgery of responses to legitimate queries.	B
Intentional or accidental modification of RR's in memory or cache.	B
.com TLD name servers inaccessible.	B
Root servers inaccessible.	C

	High Vulnerability	Medium Vulnerability	Low Vulnerability

High Impact	A	B	C
Medium Impact	B	B	C
Low Impact	C	C	D

Table 4 Threat priority model

For brevity, consider controls for the top 3 threats only. First and foremost, the risks present due to the A-level threats can be reduced significantly through configuration and change management along with tight security configuration standards. After all, the best response to an incident is to have prevented it from occurring in the first place. Configuration management ensures that security configurations are applied according to the appropriate standards and are kept recent through discovery efforts aimed at the particular systems in question. Change management will preserve both integrity and availability when discovery efforts necessitate configuration changes or the application of patches. These activities reduce the risks associated with vulnerabilities in both the DNS systems in general and those in BIND.

One of the accepted practices for defending against spoofing and cache poisoning attacks is to restrict recursive queries to trusted resolvers for RR's outside of the widgets.com zone. Queries from external name servers to the widgets.com local name servers would never have a reason to query widgets.com external name servers for any information other than that contained in the widgets.com zone anyway. Any legitimate requests for RR's outside of the widgets.com zone would be coming from internal clients attempting to connect to external resources on the Internet.

Taking into account DoS/DDoS threat scenarios during incident response planning, centered on coordination with the widgets.com upstream ISP's, reduces the risk of the external name servers becoming inaccessible.

Risks due to weaknesses in the DNS protocol itself cannot necessarily be reduced by widgets.com. However, DNS security protocols such as DNSSEC attempt to address the vulnerabilities surrounding spoofing, forgery, and hijacking. DNSSEC in particular, under development for many years now, focuses on data integrity and data origin authentication. Transaction signatures (TSIG) is another DNS security specification. TSIG is aimed at securing zone transfers as well as data origin authentication. TSIG proposes the use of shared secret keys to sign and authenticate communication between trusted name servers.

And as always, a variety of system and intrusion monitoring systems should be deployed to provide the early detection required for identifying and containing an incident, and hopefully preventing any significant disruption in business operations.

Specific Exploit: DoS Using DNS

Overview

The previous section was devoted to analyzing the threats facing DNS. Ironically, the exploit discussed in this section uses the DNS infrastructure to realize one of these very same threats, DoS. As such, it could be used against DNS itself or other portions of the Internet infrastructure.

It was documented in AUSCERT Alert AL-1999.004 entitled, “Denial of Service (DoS) attacks using the Domain Name System” as well as CIAC Advisory J-063 which just encapsulates AL-1999.004. A corresponding exploit was posted to BUGTRAQ as a S0ftPr0ject advisory (SPJ-002-000 at www.s0ftpj.org). This attack employs recursive DNS queries sent to multiple name servers with spoofed source IP addresses in order to illicit DNS responses that represent the DoS attack. The spoofed source IP is that of the desired target. Thus, the resulting mass of DNS responses is directed at the target. The goal is to saturate the target’s network resulting in denial of access to network resources. The success is based upon the observation that, with DNS, you usually get more out than you put in, and that many name servers are in a vulnerable configuration that allows recursive queries from unknown sources. Also, it is not dependent upon any particular operating system or application. Any semi-sophisticated attacker with moderate available bandwidth could generate quite a storm.

Variants

There is one variant to this exploit termed *Name Server Traffic Amplification* This variant is documented in Teso Security Advisory TESO-ADVISORY-003 found at http://teso.scene.at/advisories.php.

In order to execute this attack, the attacker must first identify chains of forwarding name servers. Like the DoS using DNS attack, it too relies upon name servers that accept recursive queries from unknown sources and, in addition, is assisted by those configured to execute multiple retry attempts. Spoofed queries are sent to the first name servers in the identified chain and then amplified by forwarding name servers.

In contrast to the DoS using DNS attack, the tricky part here is identifying a chain of servers large enough to provide adequate amplification. The authors include a tool for assisting in this purpose. Also, the attacker in this case needs only to send queries to the first name servers in the chain. For DoS using DNS, the attacker must send queries to every name server.

Exploit Details

Protocol

A complete overview of the DNS protocol can be found in the Targeted Port section above. A bit more detail regarding queries is given here in order to better understand the DoS using DNS attack.

The high-level structure of a DNS message was shown in Figure 3. The Question segment allows for the specification of the *query type.* The values for the query type basically follow the definitions of the RR types (examples of RR types are given in Table 2). The query type tells the name server what RR types to return in the response. There are a couple of special query types that have no RR type analog. One is the AFXR query type that signifies a request for a zone transfer. The other is the ANY query type. The ANY query type initiates a request for all RR's regardless of their RR type. Logically, in a query message of type ANY, the response often holds the greatest amount of data as compared to other query types. The name server will load up the Answers, Authority, and Additional info segments of the message with as many RR's as it can find that are associated with the domain in question.

The Attack

In concept, the DoS using DNS attack is akin to a DDoS attack. It uses the technique of traffic amplification by exploiting multiple, vulnerable intermediary systems in an attempt to saturate a target network. In DDoS-speak, these intermediary hosts are called zombies. They are the nameless drones whose only crime is having a security vulnerability that the attacker exploits and uses as an entry point through which the DDoS zombie code is installed. The attacker then installs a controller on another compromised host and sets it to run at his leisure. The traffic amplification potential is enormous. With DDoS tools, it is possible to achieve amplification in both the size and number of packets sent from the zombies compared to those received by the controller.

Comparing DoS using DNS to DDoS, the multiple, vulnerable intermediary systems (i.e. zombies) are the many name servers. Unlike DDoS, these servers are not compromised and no code is loaded onto them. Rather, the state of their configuration is exploited to generate the attack traffic. The flaw in their configuration is that they allow recursive queries from untrusted or unknown systems. How does this facilitate the attack? Recall the discussion about recursive queries. If the resolver submits a recursive query, the name server will fulfill the query on their behalf and return the answer. Thus, without recursion, it would not be possible to trick the name server into answering the query and sending that packet to the target.

By virtue of its query-response nature, the attack only works if amplification of the packet size can be achieved. The number of packets is not increased. The attacker has to send out one request for every response sent to the target. Therefore, the increase in the size of the packet is imperative to achieving DoS against the target without doing the same to the attacker's network. This attack is neither elegant nor stealthy, but it could be effective under the right circumstances.

DoS using DNS is distributed in the sense that the attacker seeks as many vulnerable name servers as possible. The actual number and how "distributed" they are is dependent upon the motivation of the attacker. The attacker must also choose how to use them when developing his exploit code. He might choose to cycle through them all in a round-robin fashion. Alternatively, he might pick them randomly from a table and perhaps send a few queries to each before moving to the next, or he might decide to send queries to all of them simultaneously. Depending upon the number and location of the zombie name servers, the characteristics of the resulting packet flood will vary. Experimentation would be necessary in order to achieve optimal parameters.

To see the exploit in action, refer to Figure 5 below. The attacker would first have to complete some homework. Namely, the pool of zombie DNS servers needs to be identified. This could be accomplished in a variety of ways. Recall that identifying name servers that will accept recursive queries from the Internet at large is the goal. The attacker could use any number of methods. The ugliest would be brute force searching. It is conceivable, armed with a list of domains, to use the dig command and local resolvers to find name servers, and then send recursive queries to these servers using the host command to see if they respond. Dig and host are tools distributed by ISC (authors of BIND). For each domain, the dig command would return an entry similar to the following:

```
bash-2.05$ dig google.com NS

; <<>> DiG 9.2.1 <<>> google.com NS
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18112
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.                   IN      NS

;; ANSWER SECTION:
google.com.          344874  IN      NS      ns1.google.com.
google.com.          344874  IN      NS      ns2.google.com.
google.com.          344874  IN      NS      ns3.google.com.
google.com.          344874  IN      NS      ns4.google.com.
```

The “Name Server:” lines would be used in a host command. The host command generates a recursive query by default. It would be sufficient to do a query of type A (again the default). The actual RR requested is irrelevant. On the command line, the attacker would see either

```
bash-2.05$ host www.ebay.com ns.domain1.com
Using domain server:
Name: ns.domain1.com
Address: X.X.X.X#53
Aliases:

www.ebay.com is an alias for pages.ebay.com.
pages.ebay.com has address 66.135.192.87
pages.ebay.com has address 66.135.192.88
pages.ebay.com has address 66.135.192.11
pages.ebay.com has address 66.135.192.83
```

or

```
bash-2.05$ host www.ebay.com ns.domain2.com
Using domain server:
Name: ns.domain2.com
Address: X.X.X.X#53
Aliases:
```

In these examples, the name server ns.domain1.com allows recursive queries from untrusted sources and ns.domain2.com does not. Alternatively, the attacker could have used host rather than dig (host –t NS ebay.com) to gather NS records. However, dig can reveal other goodies (e.g. authority records, additional RR’s) that wouldn’t necessarily be as evident when using host.

Another slightly more elegant method of collecting name servers requires that the attacker have access to a caching name server that supports a large user community. Dumping the cache on such a name server would reveal NS records for many domains. It would then remain to test these for recursion in the same manner as shown above. Either of these processes could be automated. However, in the brute force method, the list of domain names used as input would be hard to gather. The whois databases used to allow wildcard searches that could dump lists of domains (e.g. *.com). For security and performance reasons, that capability has been disabled.

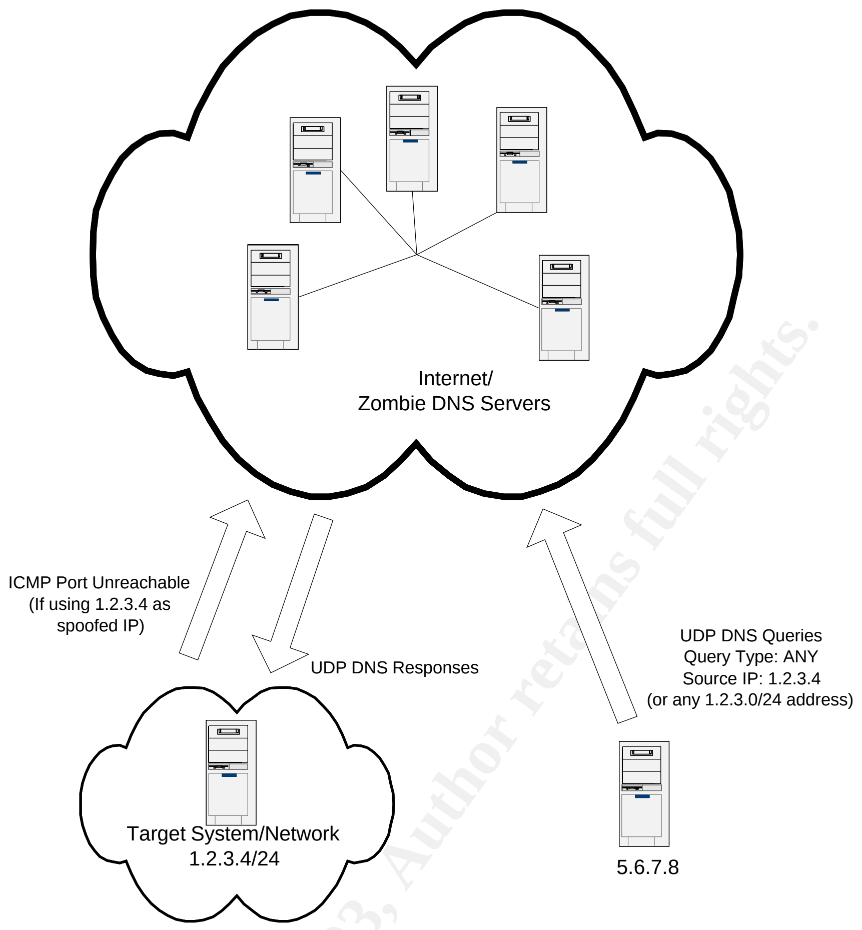


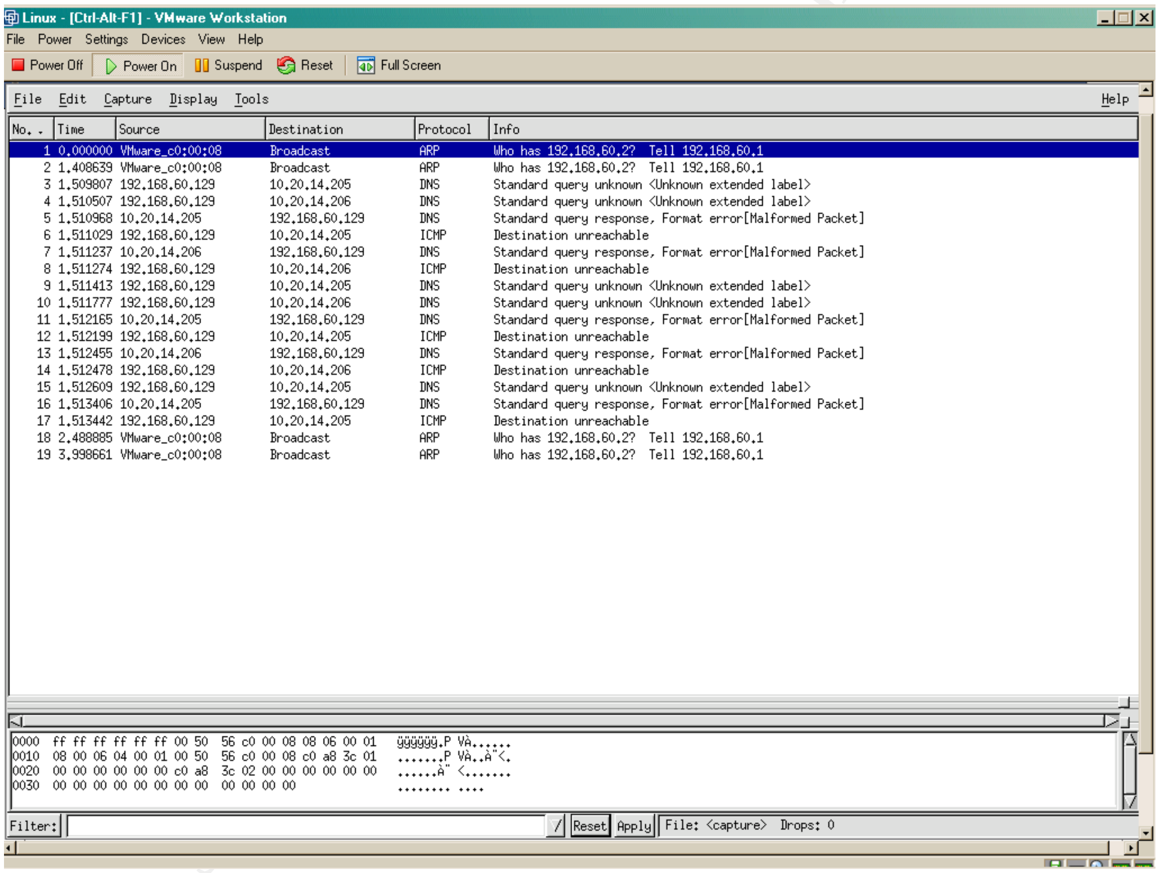
Figure 5 DoS Using DNS exploit example.

Now, armed with a list of zombie name servers, the attacker can automate the process of generating the spoofed queries. The query type ANY will undoubtedly be used since it has the potential to generate the largest response. With an ANY query of roughly 20-30 bytes, responses of >300 bytes have been observed. Of course, the maximum would be 512 bytes. The input/output budget is therefore favorable (~10x) for a DoS attack.

As discussed earlier, the design of the exploit requires a strategy for how to send the queries (e.g. round-robin, random, all at once). Beyond this information, it is a simple matter to write the exploit code, input the list of name servers, and fire away at your favorite target. The exploit would generate the ANY query for a given domain name. The value of the domain name needs only to be a valid domain. For the purposes of coding the exploit, it could also be read from a list. The query would then be sent to the zombie name servers following the chosen strategy. The source IP address in the queries would be on the network of the target. The zombie name servers would then send their large responses to the target network creating the flood.

An exploit program does exist (refer to the Exploit Source Code section below). And it functions in the same manner just described. The user is required to input lists of name servers and domains. These are currently hard-coded lists so they must be entered before compiling. Once compiled, the user executes the code with the target host and the desired number of packets as input parameters. Certainly, an attacker could execute the steps that are automated by the exploit program. Generating UDP packets with spoofed source IP addresses is possible using netcat, for example. However, he would have to be an awfully quick typist to generate the number of requests in the short amount of time necessary to achieve a packet flood!

At a minimum, the signature of this attack will be the increased DNS traffic on the network. Naturally, a tell-tale sign of a DoS attack is degradation in performance. Degradation in performance normally leads to network engineers running around performing network captures. If the network is experiencing a DoS using DNS attack, the network capture will look very similar to the following:



Linux - [Ctrl-Alt-F1] - VMware Workstation
File Power Settings Devices View Help
Power Off Power On Suspend Reset Full Screen
File Edit Capture Display Tools Help
Filter: Reset Apply

```
No. . Time     Source            Destination     Protocol Info
    1 0.000000 VMware_c0:00:08   Broadcast       ARP      Who has 192.168.60.2?  Tell 192.168.60.1
    2 1.408639 VMware_c0:00:08   Broadcast       ARP      Who has 192.168.60.2?  Tell 192.168.60.1
    3 1.509807 192.168.60.129    10.20.14.205    DNS      Standard query unknown <Unknown extended label>
    4 1.510507 192.168.60.129    10.20.14.206    DNS      Standard query unknown <Unknown extended label>
    5 1.510968 10.20.14.205      192.168.60.129  DNS      Standard query response, Format error[Malformed Packet]
    6 1.511029 192.168.60.129    10.20.14.205    ICMP     Destination unreachable
    7 1.511237 10.20.14.206      192.168.60.129  DNS      Standard query response, Format error[Malformed Packet]
    8 1.511274 192.168.60.129    10.20.14.206    ICMP     Destination unreachable
    9 1.511413 192.168.60.129    10.20.14.205    DNS      Standard query unknown <Unknown extended label>
   10 1.511777 192.168.60.129    10.20.14.206    DNS      Standard query unknown <Unknown extended label>
   11 1.512165 10.20.14.205      192.168.60.129  DNS      Standard query response, Format error[Malformed Packet]
   12 1.512199 192.168.60.129    10.20.14.205    ICMP     Destination unreachable
   13 1.512455 10.20.14.206      192.168.60.129  DNS      Standard query response, Format error[Malformed Packet]
   14 1.512478 192.168.60.129    10.20.14.206    ICMP     Destination unreachable
   15 1.512609 192.168.60.129    10.20.14.205    DNS      Standard query unknown <Unknown extended label>
   16 1.513406 10.20.14.205      192.168.60.129  DNS      Standard query response, Format error[Malformed Packet]
   17 1.513442 192.168.60.129    10.20.14.205    ICMP     Destination unreachable
   18 2.488885 VMware_c0:00:08   Broadcast       ARP      Who has 192.168.60.2?  Tell 192.168.60.1
   19 3.998661 VMware_c0:00:08   Broadcast       ARP      Who has 192.168.60.2?  Tell 192.168.60.1
```

```
0000  ff ff ff ff ff ff 00 50  56 c0 00 08 08 06 00 01   ÿÿÿÿÿÿ.P VÀ......
0010  08 00 06 04 00 01 00 50  56 c0 00 08 c0 a8 3c 01   .......P VÀ..À¨<.
0020  00 00 00 00 00 00 c0 a8  3c 02 00 00 00 00 00 00   ......À¨ <.......
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

```
File: <capture>  Drops: 0
```

At the time of this writing, the exploit code obtained from s0ftpr0ject was not functioning correctly. Although this screen shot is indicative of the traffic for this attack, the queries were malformed. The result here is that the input traffic is actually larger than the output since the query packets are larger than the Format Error replies. This is not a good thing for a DoS attack!

In any case, analysis of the network capture shows the queries being directed at two name servers, 10.20.14.205 and 10.20.14.206. The actual source IP of the attacker was 10.20.45.7. However, the queries appear to originate from 192.168.60.129, the target. Subsequently, there are many responses and corresponding ICMP unreachable messages. The responses are directed at the spoofed source IP addresses, and that system is sending out the ICMP unreachables because it is not listening for those replies. Note that the attacker and the target were on the same network here. Normally, a trace on the target network would not reveal any query messages. They are left in here for illustration purposes.

If the network normally carries a large amount of DNS traffic, the differentiators here would be the massive imbalance between DNS queries and responses as well as, possibly, ICMP port unreachable messages correlated with the response packets. Since the attacker is using spoofed source addresses, any host on the target network that actually receives a response packet will generate an ICMP port unreachable message and send it back to the originating name server. The hosts on the target network did not initiate any communication with the name servers and therefore will not be listening for the return packets. This is standard UDP behavior.

The number of ICMP port unreachables will depend on the attacker's algorithm for generating the spoofed source IP addresses. For example, if the attacker is using a single source address and that address is in use by a host system, there will be one ICMP message for every DNS response. On the other hand, if the attacker is randomizing the source IP's, the presence of the ICMP messages will be hit-or-miss depending upon the probability of generating an IP that is actually assigned to a host. Further, if the attacker uses a single source that is not assigned to a host, no ICMP unreachable messages will be present at all.

This is an interesting point. Note that the attacker does not have to use an IP address that is actually assigned to one of the victim's hosts. The attacker only cares that the flood of responses is routed properly to the victim network. Using actively assigned IP addresses does have the advantage that the ICMP port unreachable messages add to the traffic flood. However, the ICMP messages are small and are definitely a clue to identifying the attack. The attacker will have to weigh these options during the planning stage.

Defense

The defense against this attack is very basic. A name server can't be used as an amplifier if it doesn't respond to recursive queries. Most name servers are configured to support recursion. It is a useful thing. In these cases, the name server is probably in the TCP/IP configuration of many resolvers. A good example would be the corporate name server that is distributed to DHCP clients on a corporate LAN. Recursion is usually necessary for a couple of reasons. First, the resolvers on the LAN will only have access to this single name server through the firewall. This is common security policy these days. Second, allowing an upstream name server to handle the multiple queries required to resolve a domain keeps all of that traffic off of the LAN. The trick, then, is to allow recursive queries only from those systems and networks that the name server considers to be trusted.

Most DNS implementations today have this capability. BIND is no exception. In BIND, access control lists can be defined in the configuration file and used in statements that will limit recursive queries. The commands are

```
acl acl-name { x.x.x.x/xx; x.x.x.x/xx; };
```

where acl-name is a unique user-assigned name for the list and x.x.x.x/xx are the networks that you wish to use in the access control statements. Recursion can then be limited to the networks in the acl list by including the following in the config file:

```
allow-recursion { acl-name; };
```

In terms of what the vendor should do to help with this vulnerability, ALL DNS implementation vendors should include facilities for limiting recursive queries. In addition, they should make the configuration of this option a common part of the installation procedure. This will raise the awareness that the problem exists. The user can always opt out of configuring it if they so choose.

Exploit Source Code

The source code of an exploit for this attack can be found at:

http://www.s0ftpj.org/docs/spj-002-000.txt

The code requires that the user fill two character arrays before compiling:

dns_def = the list of zombie name servers

domains = the list of domains to use in the queries.

It can be compiled on most Unix platforms (I used gcc on Linux: gcc -o doomdns doomdns.c). It is executed by issuing the command

```
doomdns target [n]
```

where target is the hostname or IP address of the target and n is an optional parameter specifying the number of packets to send. By default the program sends 100 packets.

Most of the work is carried out by the functions main(), doomzone() and forge(). The main() routine the calling arguments and spits out usage syntax if necessary. If the input is correct, it runs the main loop which iterates calling the doomzone() function, and then printing out a "." on the screen. It does this either 100 or n times depending upon the calling arguments.

Doomzone() first checks to see if the end of the name server list has been reached. If so, it resets the counter to start back at the beginning of the list. So it cycles through the defined name servers in a round-robin fashion. It then resolves the name server hostname if necessary (through a call to another routine, nameResolve()). Finally it generates the source and destination portnumbers and calls the forge function with the name server ip, the source port, and the destination port as arguments.

Forge() generates the DNS query message, constructs the UDP datagram and sends it to the name server. It then increments the counter that causes the next name server and domain in their respective lists to be used for the next iteration.

Running the program produces the following screen output:

```
/usr/local/doomdns# ./doomdns 10.20.44.141 10

D00M DNS
DNS Flooder by FuSyS
Inithints by |scacco|

……….

/usr/local/doomdns#
```

Additional Information

The CIAC advisory describing the vulnerability is at:

http://www.ciac.org/ciac/bulletins/j-063.shtml

As mentioned above, exploit code can be found here:

http://www.s0ftpj.org/docs/spj-002-000.txt

CERT brought up the topic almost a year after the original post. Their discussion is at:

http://www.cert.org/incident_notes/IN-2000-04.html

Vern Paxson included this topic as a subset of the content in his more general paper on reflectors. Read the paper here:

http://www.icir.org/vern/papers/reflectors.CCR.01/

References

Rader, Ros W. “Alphabet Soup: The History of DNS.” June 2001. URL: http://www.whmag.com/content/0601/dns (21 Feb. 2003)

ICANN Security and Stability Advisory Committee. “ICANN DNS Security Update #1.” 4 January 2002. URL: http://www.icann.org/committees/security/dns-security-update-1.htm (21 Feb. 2003)

Albitz, Paul & Liu, Cricket. DNS and BIND. Sebastopal: O’Reilly, 1998.

Men & Mice. “On-line DNS Glossary.” URL: http://www.menandmice.com/online_docs_and_faq/glossary/index.htm (21 Feb. 2003)

Mockapetris, P. “RFC 1034.” November 1987. URL: http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1034.html (27 Feb. 2003)

Mockapetris, P. “RFC 1035.” November 1987. URL: http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1034.html (27 Feb. 2003)

Unkown. “DNS Abuse.” URL: http://www.ussrback.com/docs/papers/protocols/mi004en.htm (28 Feb. 2003)

Farrow, Rik. “DNS Root Servers: Protecting the Internet.” January 6, 2003. URL: http://www.networkmagazine.com/article/printableArticle?doc_id=NMG20021223S0008 (28 Feb. 2003)

Stewart, Joe. “DNS Cache Poisoning – The Next Generation.” January 27, 2003. URL: http://www.securityfocus.com/guest/17905 (28 Feb. 2003)

CIAC. “J-063: “Domain Name System (DNS) Denial of Service Attacks (DoS).” September 1, 1999. URL: http://www.ciac.org/ciac/bulletins/j-063.shtml (1 Mar. 2003)

Scacco. “Possible Denial of Service using DNS.” July 19, 1999. URL: http://www.s0ftpj.org/docs/spj-002-000.txt (1 Mar. 2003)

TESO. “Many name servers are vulnerable to traffic amplification and NS route discovery.” February 15, 2000. URL: http://www.securiteam.com/exploits/5YP0E1F0KU.html (1 Mar. 2003)