Global Information Assurance Certification Paper

In Yet Another BIND
Ron Baklarz
February 20, 2001

Introduction

Without a doubt, the Domain Name System (DNS) and its popular implementation -- the Berkeley Internet Name Domain (BIND) software are absolutely essential to the proper functioning of the Internet. Through the years, a number of BIND vulnerabilities have surfaced that have been successfully exploited by the *hacker* community. This paper examines the BIND problems as alerted by the CERT Coordination Center (CERT/CC ®) and some interesting perspectives associated with the details of the software and the associated exploitations.

What Are DNS and BIND?

In the early days of the Internet, the population of hosts on the Internet was sufficiently small that all the information was contained in a single file – *HOSTS.TXT.* As the general Internet protocol evolved into today's TCP/IP and the number of hosts increased, a more technically eloquent solution was needed. The result was DNS as architected by Paul Mockapetris of USC's Information Sciences Institute and codified in RFC's 882 and 883.

Essentially, the Domain Name System (DNS) is a distributed database containing a hierarchical scheme (much like the tree directory structure one sees in a UNIX system). DNS operates in a client/server mode to ensure a robust and performance-enhanced architecture supported through replication and caching. Just like the UNIX file directories, DNS has domains and subdomains that behave in much the same way. We are all familiar with .com, .org, .edu, .mil, etc. and these are called Top Level Domains. As you inversely "climb" the directory tree (out on a limb so to speak), you get closer to a specific, unique host destination on the tree.

BIND is the program that implements DNS as identified and codified in the appropriate and latest RFC. Paul Mockapetris wrote the first implementation of DNS and called it *JEEVES.* A subsequent implementation was written by Kevin Dunlap as BIND and was included as part of Berkeley's 4.3BSD UNIX operating system. Currently, Paul Vixie maintains BIND under the Internet Software Consortium (ISC).

DNS and BIND work to allow the back-and-forth translation between domain names (e.g., www.mysite.com) and their numeric equivalent or IP address (e.g., 123.456.789.123).

How Do They Work?

As discussed above, DNS employs a client/server architecture. The server side of the equation uses programs called *name servers.* Name servers contain information about some segments of the entire domain (e.g., .com, .edu, etc.). The client-side of the process uses program(s) called *resolvers*. You can think of *resolvers* as library routines that create queries to send across the Internet to a name server.

Resolvers simply query name servers, interpret the response, and then return the results to the requesting program(s). Name servers have the job of finding answers to the queries that may require them to query other name servers. This simplistic resolver is referred to as a *stub resolver* in the DNS specifications. It should be noted that more sophisticated resolvers exist in other implementations of DNS, but BIND's stub resolver is the most common found on the Internet.

Resolvers issue queries in two forms: *recursive* and *iterative.* The recursive query requires the name server to make additional queries (to other name servers) if it does not have the answer. In kind, name servers can send recursive or iterative queries to other name servers. (Later versions of BIND allow administrators the ability to have their name servers refuse recursive queries for obvious reasons.) Iterative queries only require the name server to give the best answer it already knows back to the resolver without additional queries to other name servers.

This information will give enough of a background to go ahead with an explanation of the vulnerabilities associated with BIND. More detail as to how DNS works is beyond the scope of this document.

Why all the fuss? (Part 1)

The "Holy Grail" of hacking is the root compromise. If hackers are able to obtain "root" on a system, they are able to have complete control over the victimized system as well as potential access to other systems that have trusted relationships with the compromised system.

Root compromises are usually possible via *buffer overflow conditions*. Buffer overflows occur when a user or system (program) process attempts to put more data into a "buffer" (fixed memory array or area) than has been allocated. Buffer overflow conditions are usually the result of poor programming practice where the programmer has not written sufficient error-handling code that would check for the overflow condition, handle it, and return an error message.

If a system/program is vulnerable to buffer overflow conditions, the system can either succumb to a Denial of Service (DOS) condition and crash, or evoke an even more dangerous condition by executing arbitrary code. If the latter

condition exists, the attacker may be able to overflow the buffer and have the system execute /bin/sh (i.e., shell) with all the attendant system privileges that the compromised program/process was running under – usually *root.*

Why all the fuss? (Part 2)

Since 1997, CERT® has published 12 documents describing various exploits and vulnerabilities with BIND. One exploit has been tracked by CERT® was identified by CA-1999-14 on November 10, 1999. This advisory identified the BIND T_NXT record processing vulnerability as one of multiple vulnerabilities in BIND. Specifically, Vulnerability Note VU#16532 states that the vulnerability "*allows remote attackers to execute code with the privileges of the process running named. This vulnerability was widely exploited from November 1999 to December 2000"*. The BIND T_NXT vulnerability is a buffer overflow exploit.

CERT® Advisory CA-2001-02 contains a chart of reported VU#16532 incidents. The chart depicts about a one-month lag between the identification of the exploit in November 1999, and its appearance "in the wild" in December 1999. Its peak exploitation rate occurred at about 2 months (January 2000) subsequent to the issuance of the advisory.

CERT® Advisory CA-2001-02 identifies four BIND vulnerabilities that can be summarized as: (a) two - buffer overflows; (b) one - "input validation error" (close cousin to buffer overflow that can result in the execution of arbitrary code; and (c) one - "disclosure" vulnerability that permits access to the program stack and thus disclosing program and/or environmental variables.

Let us examine the buffer overflow vulnerabilities identified in this, the latest advisory of BIND-related vulnerabilities.

VU#196945 ISC BIND 8 contains buffer overflow in transaction signature (TSIG) handling code.

1. BIND 8 receives a query from TCP (transport protocol) and the TCP stream is read into the malloc()'d buffer. BIND reuses this space when formulating a reply to the query.

2. As BIND processes the request, it appends data to the DNS response in the malloc()'d buffer. The length of the DNS message response as well as the number of bytes that can be written are stored in two variables.

3. When a transaction signature is included in the query, BIND skips normal processing and attempts to verify the signature.

    a. If the signature is valid, a TSIG response is appended to a memory location BIND "thinks" is at the end of the message (based on the two

variables mentioned in Item 2 above). Since BIND has not processed the message normally, the TSIG response is actually far away from where it should be within the bounds of the allocated space. While this exploit is categorized as a buffer overflow, it cannot be exploited in the same manner as a stack overflow since it uses the "bss" or "heap" region of the process memory. Since this region of memory is not executable, attackers must use other buffer overflow exploitation techniques in order to compromise the system.

b. If the signature is invalid, the same processing occurs as with a valid signature with the exception that TSIG response can be written partially over the executing function's stack frame. Since the TSIG response is made up of fixed values (including zero-value bytes) and the least significant byte of the saved base pointer in the stack frame is overwritten, it could reference memory under control of an attacker. If this occurs, the attacker will have control over the stack frame of the calling function.

VU#573182 ISC BIND 4 contains buffer overflow in nslookupComplain().

1. BIND 4 name server receives a query for a host name and first checks it's own zone and cache for the information. If the query cannot be satisfied locally, BIND obtains the name servers that are responsible for the host's domain.

2. Once BIND has the NS records, it call nslookup() to get the IP addresses of the name server. A validity check is performed for each name server's IP address by nslookup(). If the IP address is invalid, the function nslookupComplain is called to log the error in syslog. There are two manifestation of this vulnerability:

   a. An attacker may be able to overwrite arbitrary locations in memory with almost arbitrary values. This occurs by passing syslog a string attacker-supplied input (name server host name) as the format string argument. This is acted upon by the printf() function used by syslog() and may not be exploitable due to restrictions on characters used in domain names.

   b. When generating the error message (as the result of the invalid IP address identified in Item 2 above) nslookupComplain() uses sprintf() to construct the null termination string. The termination string is 999 bytes in length and is a local variable. If the name server host name exceeds the length of the allocated buffer size, it will be copied over nslookupComplain()'s stack variables. This exploit may not be valid since in order to construct a fully qualified domain name, the attacker may have to use several malicious DNS servers and it may not be possible to build a valid return address from these bytes.

Summary

DNS and its implementation via BIND are absolutely essential for the proper functioning of the Internet and the millions of Internet-connected computers. With the current BIND vulnerabilities, hackers have an opportunity to wreak havoc over the Internet due to the widespread use (and need) of BIND. If the CERT/cc ® historical perspective remain true, we should be seeing a dramatic rise in attacks against DNS servers in the coming weeks.

Hopefully, this paper has provided the audience with a better appreciation for the technical complexities associated with exploits of this nature. The bottom line is that as a minimum, administrators would do well to: (1) keep up on the latest vulnerabilities by accessing resources such as SANS and (2) apply the latest patches from vendors.

References:

Albitz, Paul and Liu, Cricket. DNS and BIND Second Edition, O'Reilly & Associates, USA 1997

Scambray, McClure, and Kurtz, Hacking Exposed: Network Security Secrets & Solutions Second Edition, Osborne/Mc Graw-Hill, USA 2001

Poulson, Kevin M. "BIND holes mean big trouble" January 29, 2001 URL: http://www.securityfocus.com/templates/article.html?id=144

CERT ® Advisory CA-2001-02 Multiple Vulnerabilities in BIND Original release date: January 29, 2001 Last revised: February 15, 2001 URL: http://www.cert.org/advisories/CA-2001-02.html

SAFER (Security Alert for Enterprise Resources) Volume 4 Issue 2 February 2001 The Relay Group URL: http://www.safermag.com/pdf/Safer33.pdf