Global Information Assurance Certification Paper

Support for the CDI: Port 25 - SMTP

Presented as partial fulfillment for the

GCIH certification
Practical Exam v 2. 2

Steven Mancini
January 31, 2003

Table of Contents

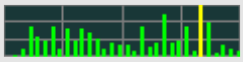
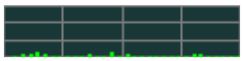
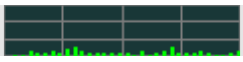
Introduction

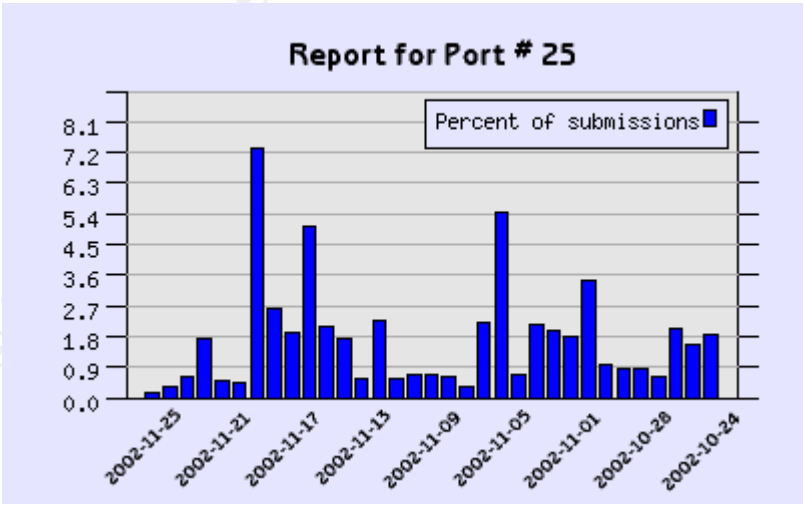
"The sendmail program can be an open door to abuse. Unless the administrator is careful, the misuse or misconfiguration of sendmail can lead to an insecure and possibly compromised system. Since sendmail is usually installed to run as an suid root program, it is a prime target for intrusion."
– Bryan Costatles, Sendmail

The Internet Assigned Numbers Authority (www. iana. org/assignments/port-numbers) has designated that port 25 (tcp/udp) is assigned to the Simple Mail Transfer Protocol (SMTP). This protocol has become the standard used for transferring electronic mail (e-mail) between systems. An important feature of SMTP is its capability to relay mail across transport service environments. This transport service environment may cover one network, several networks, or a subset of a network. To understand the potential this service holds for destroying the integrity/availability/confidentiality of systems, it is important to realize that e-mail has been around for over 2 decades. Email delivery was one of the original reasons that ARPA and DARPA gained attention and support – today it is still one of the most significant reasons people connect to the internet. It has become an essential service in both the private and public sectors. As such, this service has a great deal of appeal to the hostile attacker. This service is a given for most targets – most companies and organizations have mail enabled to both send and receive from the internet. For those seeking credit/fame for the damage they have caused, mail is a very appealing target because of the dependency that many organizations have upon it. Two traits of the routine functioning of this service also facilitate the would-be attacker; 1) it is not uncommon for the MTA to accept connections from external sources (for most MTA's this is a core requirement), and 2) it is the very function of email to distribute data through its environment. For the attacker seeking to cause the most damage, or who seeks to target a service that he/she knows a company will be hesitant to disable as part of a containment strategy, port 25 becomes a very appealing target. In addition, given the open relay between mail servers, it is possible for a clever attacker to disguise his/her enumeration scans as merely SMTP exchanges and (hopefully) go un-noticed by today's growing number of intrusion detection programs.

Incidents.org (or dshield.org) collects intrusion detection data from the Intrusion Detection Systems of volunteer's around the globe. This information is collated to form reports of the top ports attacked which are posted on the dshield website to provide indicators to security professionals – it allows them to "see which way the wind blows". The following screen shots were captured on July 10, 2002:

Top 10 Ports (reported July 10, 2002)

Service Name	Port Number	30 day history	Explanation
http	80		HTTP Web server
ftp	21		FTP servers typically run on this port
ms-sql-s	1433		Microsoft SQL Server
???	39213		
ssh	22		Secure Shell, old versions are vulnerable
???	43981		
???	6346		Gnutella is a peer-to-peer file sharing tool
smtp	25		Mail server listens on this port.
socks	1080		proxy/firewall program
webcache	8080		Frequently used for web servers



DShield - Port Report for 25 - SMTP

Not surprising, tcp port 25 is among the top 10. However, its station among the top 10 may not be attributable to a single exploit or announcement. In fact, one

must recognize that there are other nefarious, and sometimes legitimate, reasons for port 25 to be scanned.

E-Mail advertisement software may be seeking information from Mail Transfer Agents (MTA's) for possible "target audiences". With the surge of email marketing, tools have been designed to harvest email addresses from MTA's connected to the internet. This type of software connects to tcp port 25 on a mail server and uses commands such as VRFY (which verifies an email address on the server) and EXPN (expand a mailing list that is sponsored on the mail server) to harvest potential audience members or addresses to impersonate. In some instances the use of these SMTP commands may be based upon information obtained from public sources (such as internet news groups or public mailing lists). These tools also discover valid email addresses via brute force username verification processes (example: steve@myisp.com). Automated tools such as these will initiate their searches by trying to discover email servers – this is done through a simple scan on port 25.

It is also important to keep in mind that the use of this port is not designated to a platform specific application, it is possible that the reports at incidents.org are the culmination of several exploits targeting a variety of mail programs (Exchange, Lotus, Novell, etc) running on different platforms. A list of potential exploits against all email server software would be too extensive a list to cover in this document – though we will provide a sample further along in the paper to demonstrate that the threat is not limited to any platform or application.

In this paper we are going to map the surge in activity on this port to the timely discovery of the DNS resolver library buffer overflow and the vulnerability to this exploit announced by the developers of one of the most widely used mail servers for the unix platform, sendmail. Sendmail is compiled such that it relies upon this DNS resolver library to lookup addresses and map them to IP addresses to which the email is delivered. We shall proceed forward under the assumption that at least some significant component of the activity is attributable to the recent exploit announcement and the pre-emptive scanning by unfriendly perpetrators on the net eagerly awaiting (or designing) code to take advantage of this exploit. In some way, exploitation of this vulnerability draws several similarities to the ancient parable of the Trojan horse – the exploit involves the concealment of exploit code in circumvents the usual computer defense (firewalls) by including itself in a response to an otherwise trusted response.

Part 1 – Port 25

Targeted Port

The ARPANET e-mail proposals, RFC 821 (transmission protocol) and RFC 822 (message format), were introduced in 1982 and designated port (tcp/udp) 25 for

use in the transfer of e-mail. In 1984, CCITT drafted its X.400 recommendation, which was later taken over as the basis for OSI's MOTIS. Four year later (1988), CCITT modified X.400 to align it with MOTIS. MOTIS was supposed to be the representing application of OSI, a system that was to be all things to all people. X. 400 never really caught on – people were drawn to the easy implementation that RFC 821 and 822 provided. Today, most e-mail systems are based on RFC 822, whereas those based on X.400 have disappeared. In this case, the RFC won favor for its simplicity and because X.400 was too challenging to successfully implement.

The simplicity of SMTP has allowed for the development of numerous mail client (pine, elm, exmh, Eudora, Outlook, Lotus Notes, etc) and server applications (Sendmail, Qmail, Exchange, Lotus, Novell, etc) which attach to port 25 - all of which can exchange mail messages because they rely upon the taxonomy initially defined in RFC 821 and 822.

While the exchange of mail using is performed by a message transfer agent, most users normally don't deal directly with the MTA; they are usually only aware of their e-mail clients and the sending and receiving of e-mail. SMTP is the protocol that describes how two MTAs communicate with each other using a single TCP connection. SMTP uses the concept of spooling. The idea of spooling is to allow mail to be sent from a local application to the SMTP application, which stores the mail in some device or memory. Once the mail has arrived at the spool, it has been queued. A server checks to see if any messages are available and then attempts to deliver them. If the user is not available for delivery, the server may try later. Eventually, if the mail cannot be delivered, it will be discarded or perhaps returned to the sender.

Application Description

SMTP has since become the default for almost all MTA's on the net – and those MTA's listen on TCP/IP 25. While SMTP is operating system independent and used by an assortment of e-mail programs, our report focuses on one of the most popular unix MTA's – Sendmail. Sendmail is a freely available and widely distributed Mail Transfer Agent (MTA). Sendmail is packaged with most versions of UNIX, and source code is publicly available. Current Sendmail information and source code can be obtained at http://www.sendmail.org. Sendmail is based on the following protocols and formats:

- RFC821 (Simple Mail Transport Protocol): This rfc is the foundation for SMTP – it describes the model, the exchange between sender and recipient, and the command specifications for the exchange and error codes.
- RFC822 (Internet Mail Headers Format): This standard specifies a syntax for text messages that are sent among computer users, within the

framework of email. There are 3 components to a message, an envelope, a header, and the message – this rfc deals with the header and the message.

- RFC974 (MX routing): This rfc deals with the delivery of email to a domain. Network maps have been established that sometimes one does not deliver mail to an apparent host (ie, mail may be sent from client12.domain.com within the domain.com intranet, but should be delivered to mailserver.domain.com). Instead, for a given domain, mail delivery is linked to MX records and the priority established by these records.
- RFC1123 (Internet Host Requirements): Specifically, section 5, involves Electronic Mail and the changes to 822 as necessary for developments in internet communication (including DNS). This rfc focuses on applying SMTP and 822 to the Internet. This rfc is pertinent to our current examination because it requires that the HELO command perform a domain name lookup and have valid <domain> syntax. It is this requirement that results in programs such as sendmail calling the resolver library.
- RFC1652 (SMTP 8BITMIME Extension): This rfc covers the implementation of a MIME message containing arbitrary octet-aligned material. The implementation uses the mechanism described in rfc 1651 to define an extension to the SMTP service.
- RFC1869 (SMTP Service Extensions): Issued 10 years after 821, this rfc defines the framework for expanding the SMTP service by defining how an SMTP server can inform a client as to the service extensions (esmtp) it supports.
- RFC1870 (SMTP SIZE Extension): This rfc defines an extension to the SMTP service whereby a client and server may interact to give the server an opportunity to decline a message based on the client's estimate of the message size.
- RFC1891 (SMTP Delivery Status Notifications) This rfc defines an extension to the SMTP service allowing an SMTP client to specify
  a) that delivery status notifications (DSNs) should be generated under certain conditions,
  b) whether such notifications should return the contents of the message, and
  c) additional information, to be returned with a DSN, that allows the sender to identify both the recipient(s) for which the DSN was issued, and the transaction in which the original message was sent.
- RFC1892 (Multipart/Report): This rfc defines the use of the Multipart/Report MIME content-type, a container type for electronic mail reports of any kind.
- RFC1893 (Enhanced Mail System Status Codes): This RFC addresses the evolution of email server/client programs by providing more robust set of standardized error handling codes.

- RFC1894 (Delivery Status Notifications): This memo defines a MIME content-type that may be used by a message transfer agent (MTA) or electronic mail gateway to report the result of an attempt to deliver a message to one or more recipients.
- RFC1985 (SMTP Service Extension for Remote Message Queue Starting): This memo defines an extension to the SMTP service (verb: ETRN) whereby an SMTP client and server may interact to give the server an opportunity to start the processing of its queues for messages to go to a given host. This extension is meant to be used in startup conditions as well as for mail nodes that have transient connections to their service providers.
- RFC2033 (Local Message Transmission Protocol): This rfc defines the protocol by which a mail receiver does not manage a queue in a system that is outside the scope of mail exchange between independent hosts on public networks.
- RFC2034 (SMTP Service Extension for Returning Enhanced Error Codes): This memo defines an extension to the SMTP service whereby an SMTP server augments its responses with the enhanced mail system status codes that provide more informative explanations for error conditions (defined in RFC 1893).
- RFC2476 (Message Submission) This document specifies the various headers used to describe the structure of MIME messages.
- RFC2487 (SMTP Service Extension for Secure SMTP over TLS) This RFC describes an extension to the SMTP service that allows an SMTP server and client to use transport-layer security [TLS], also known as SSL, to provide private (through use of encryption), authenticated communication over the Internet.
- RFC2554 (SMTP Service Extension for Authentication) This document defines an SMTP service extension [ESMTP] by which an SMTP client may indicate an authentication mechanism to the server, perform an authentication protocol exchange, and optionally negotiate a security layer for subsequent protocol interactions. This extension is a profile of the Simple Authentication and Security Layer [SASL] and added the verb AUTH to the SMTP protocol.
- RFC2821 (Simple Mail Transport Protocol) This RFC consolidates, updates and clarifies, but doesn't add new or change existing functionality of the following RFC's - 821, 974, 1035, 1123, 1869.
- RFC2822 (Internet Message Format) This standard supersedes RFC 822, updating it to reflect current practice and incorporating incremental changes that were specified in other RFCs.
- RFC2852 (Deliver By SMTP Service Extension) This RFC defines a mechanism which allows an SMTP client to request that the server deliver the message within a prescribed period of time. Usually there are server defined parameters for queue, delivery, and retry times.
- RFC2920 (SMTP Service Extension for Command Pipelining) This rfc defines an extension to the SMTP service whereby a server can indicate

the extent of its ability to accept multiple commands in a single Transmission Control Protocol (TCP) send operation.

Protocols

"NO! Layers! Onions have layers. Ogres have layers. Onions have layers. You get it? We both have layers." -Shrek

The complexity of this vulnerability is demonstrated by the number of protocols it would rely upon to be successfully exploited. At the application layer there is SMTP – the mail protocol that the Sendmail application relies upon to send and receive mail via the ASCII character set. Back at its inception, SMTP was designed to work on the intranet level. However, its potential was soon realized and as a result standards (RFC 1123) were created to address the communication between systems. This in turn requires the ability to look up host names and ip addresses. Sendmail does not have to provide its own routines for looking up a hosts and IP-addresses. Instead, it relies upon Doman Name Service (DNS), a session layer protocol that includes a number of library functions that do this transparently, called *gethostbyname(3)* and *gethostbyaddr(3)*. These and other related procedures, including *getanswer()* and *getnetanswer()*, are grouped in a separate DNS resolver library. It thus depends upon the DNS name server, and complimentary resolver library, to provide it with a mapping of an IP address for a given email address/domain. DNS in turn, relies upon the transport protocols, TCP and UDP. DARPA (the Defense Advance Research Projects Agency) originally developed Transmission Control Protocol/Internet Protocol (TCP/IP) to interconnect networked computers. Since then it has spread in usage throughout the internet. The TCP/IP suite includes the following relevant suites; Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and the network layer protocol, Internet Protocol (IP). Consequently, this vulnerability is like an ogre (who is like an onion) – it has layers (of complexity). Hopefully we can get to the core of the issue without shedding too many tears. ☺

Layer	Protocol
Application	SMTP
Session	DNS
Transport	TCP, UDP
Network	IP

Network Layer Protocols (IP)

The internet protocol (IP) is the routing layer datagram service used by all other protocols within the TCP/IP suite, with the exceptions of ARP and RARP. This protocol routes frames from host to host. Included in the appendices is a depiction of the header structure for IP. While the length of this datagram is traditionally 576 byte, it can be up to 65535 bytes in length. It can be fragmented, and includes a label to identify the next level protocol that is used in the data component of the datagram. For more information on this protocol I would refer the reader to RFC 791 and RFC 1853.

Transport Layer Protocols (TCP/UDP)

The User Datagram Protocol was established to provide a simple protocol with low overhead with regard to time to deliver and bandwidth. It has on occasion been referred to by the military moniker, 'fire and forget' because it does not perform and acknowledgment or retransmissions. UDP headers are extremely simplistic where compared to TCP (see appendix for diagrams of protocols for comparison) – it provides a source and destination, length, a checksum, and the data that is carries in a small packet. For some application communications, it is considered a lower overhead to send multiple UDP messages than a single TCP exchange.

Where as UDP is smaller and faster, TCP provides a reliable stream delivery and virtual connection service for applications through the use of sequenced packets that are sent, acknowledged, and retransmitted in cases where packets are not received. TCP is used by DNS when responding to a non-zone transfer. For more information on TCP, I would encourage the reader to review the following RFC's: 793, 1072, 1693, 1146, and 1323.

Session Layer Procotols (DNS)

The DNS protocol requirement is defined in section 6 of RFC 1123: "Every host MUST implement a resolver for the Domain Name System (DNS), and it MUST implement a mechanism using this DNS resolver to convert host names to IP addresses and vice-versa." During its infancy, the internet was sufficiently small that looking up an ip address could be accomplished by referencing a local copy of the table containing a list of all hosts. DNS was created to address the unwieldy swelling of this table. It consists of a distributed database that allows for the (relatively) fast translation between host names and host ip addresses. The resolver component of DNS is used to query the authorative name servers for information about a host name or address. In effect DNS is a "map" of systems on the internet. No single name server has complete information – it is distributed and replicated so that queries by the resolvers can be addressed in a timely fashion.

This is important to keep in mind when considering this vulnerability because it means that most computers will query numerous different name servers by design. It is thus possible for a DNS query to be made against a name server controlled, or spoofed, by an attacker. DNS spoofing is the process of faking a response from a DNS server in order to provide the wrong address mapping to the querying system While the identity of name servers comes from a file at system boot, subsequent queries against these name servers is cached for a period of time. Given this situation, if 'tainted' information is provided by an attacker controlled DNS server, it could be cached and reused by more than one system in your network. In the scope of this vulnerability, the caching is only important if you are running the sendmail application on more than one host such that the service would be performing a name resolution.

Given the requirement for fast name resolution, DNS was designed to not only use TCP, but also to optimize its query/responses by using UDP. While it supports TCP for sending non-zone-transfer queries (and because the authors of the RFC recognized that over time responses may at some point exceed the 512 byte limit of UDP), UDP is preferred because of the low network overhead incurred by it. This allows a resolver to send queries to multiple name servers for about the same cost of a single tcp query.

Application Layer Protocols (SMTP)

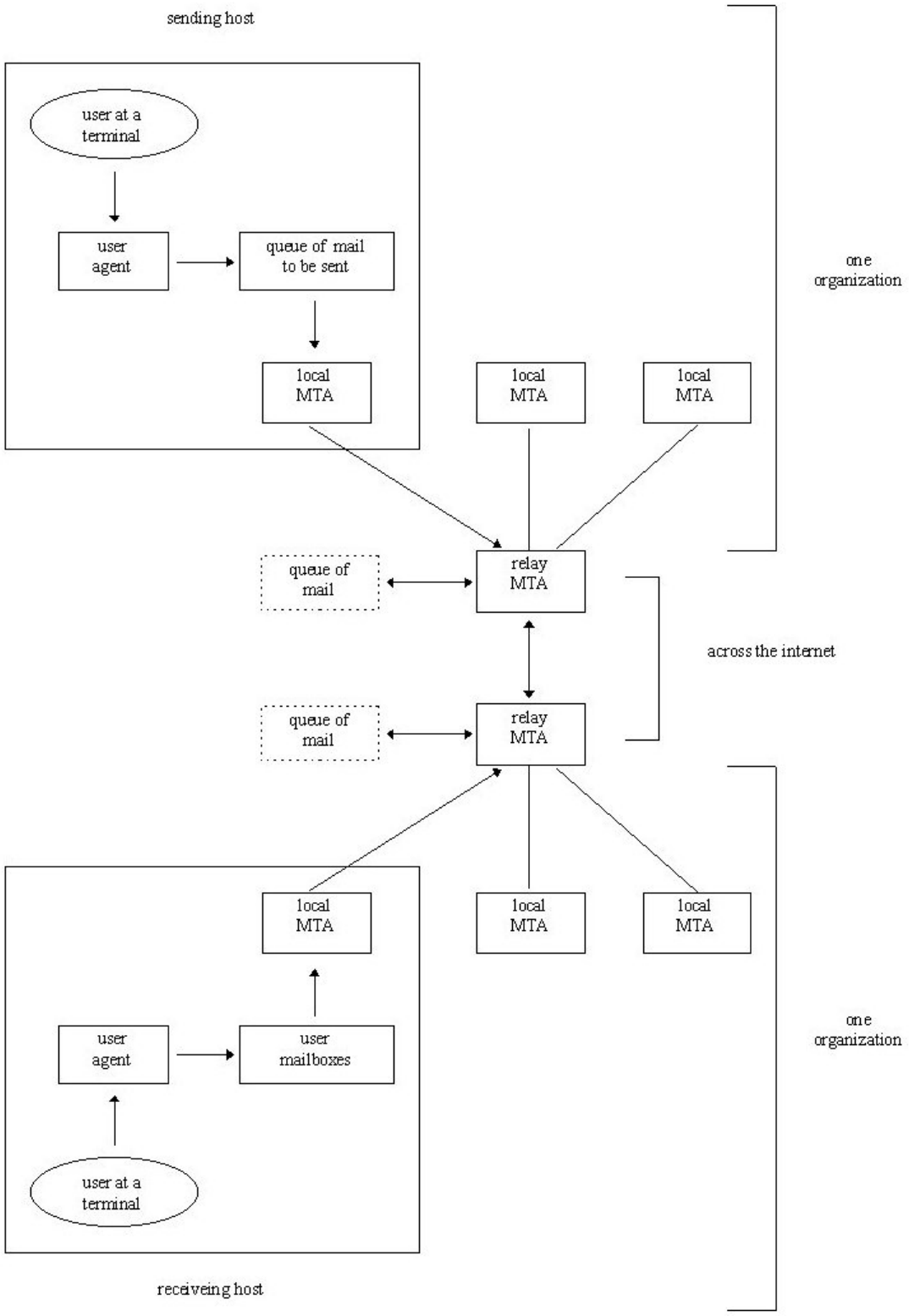
In keeping with the simplicity upon which SMTP was based, it is probably no surprise that SMTP relies upon the ASCII protocol for exchanges between systems. The use of ASCII to relay a set of very common commands (referred to as Verbs in RFC 821), allows SMTP to communicate between systems and even platforms. SMTP commands are character strings terminated by <CRLF>. The command codes themselves (Verbs) are alphabetic characters terminated by <SP> if parameters follow and <CRLF> otherwise. When the transport service provides an 8-bit byte (octet) transmission channel, each 7-bit character is transmitted right justified in an octet with the high order bit cleared to zero.

All electronic e-mail consists of 3 components: the envelope, the headers, and the body. The *envelope* consists of the identities of the sender and recipient of the e-mail. These are identified in the SMTP protocols which are defined in RFC 821. The *headers* are used by the e-mail applications to provide further definition of the message in question – these 9+ attributes are not requirements for SMTP to function, they merely augment the transaction from a human perspective. According to RFC 822, these 9 headers (Received, Message-Id, From, Date, Reply-To, X-Phone, X-Mailer, To, and Subject) follow the format of header-colon-value. The *body* is the content of the message that is exchanged using the DATA command and is sent in 1000 byte groupings per DATA command issued. The whole process starts with the user's mail application relaying the body along with the relevant header information to the MTA. The MTA then wraps the message with the envelope and sends it to another MTA via SMTP.

From the “10,000 foot view”, the delivery of electronic mail is an exchange across port 25 using the ASCII protocol. This collection of functions are therefore referred to as “the resolver". The mail program calls upon the resolver libraries to find the address to which it is to delivery mail for a given client. The resolver library either provides an ip address from its locally cached data or queries a name server for the address. The source machine establishes a TCP connection to port 25 of the destination machine. Listening to this port is an e-mail daemon that "speaks" SMTP. After establishing the TCP connection to port 25, the sending machine, operating as the client, waits for the receiving machine, operating as the server, to talk first. The server starts by sending a line of text giving its identity and telling whether or not it is prepared to receive mail. This is the 220 reply code you can see in the details of an e-mail exchange. If it is not able to connect to the target machine, the client releases the connection and tries again later. It will make repeated attempts (for up to 5 days according to the rfc) then it will give up and abandon the connection attempts and notify the sender of the failure to deliver the message. If the server is willing to accept e-mail, the client announces whom the e-mail is coming from and whom it is going to. If such a recipient exists at the destination, the server gives the client the go-ahead message. Then the client sends the message and the server acknowledges it. No checksums are generally needed because TCP provides a reliable byte stream. If there is more e-mail, that is now sent. When all the e-mail has been exchanged in both directions, the connection is released.

This is a rather simplistic view of the exchange that takes place. If all systems were merely connected point to point then the story might end here. But given the vast presence on the internet, this is not the case. As a result, most organizations with multiple users establish a mail server of some sort. The purpose of this system is to relay outbound mail. This simplifies the configurations in their environment because all systems in their environment relay to this central mail server, which in turn can route e-mail accordingly. The local clients all require a vastly simplified configuration as well as reducing the overhead on the systems resources. Second, this strategy also allows the organization to obfuscate their systems from the outside world. The sending and receiving of mail is to *user@mailhost.some-group.com* rather than to a specified system. This also makes it easier for those sending/receiving e-mails; one need only know the DNS name for the company to send a friend e-mail, you don’t need to know his localhost. This also provides the advantage that should that localhost be offline for whatever reason, mail will still be delivered to the mail server. Two things are important to keep in mind as you continue through this examination – (1) that setting up a mail relay must be done carefully or else your mail server could be used to relay mail for others, and (2) that DNS is used to route mail through the MTA relays and as such, is an ‘essential service’ for successful operations through port 25. When mail is sent, more often than not it needs to be relayed through several relay agents as it travels from sender to recipient. To do this, it has to have an address to send to.

Diagram of Email Delivery across the Internet



Vulnerabilities

The SMTP protocol has had a number of published vulnerabilities. A search of the Common Vulnerabilities and Exposures (CVE), Bugtraq, or the Computer Emergency Response Team (CERT) vulnerability lists will yield a number of results. The vulnerabilities of the smtp protocol range from Denial of Service attacks to exploits allowing the arbitrary execution of code on the remote system. The following chart contains exploits known to be targeted against services run on port 25 – they are included for completeness of this report.

This initial list for non-sendmail SMTP exploits reveals a noteworthy trend. First off you can see that Sendmail is not a black sheep of the mail program family – exploiting SMTP applications occurs on all platforms and for most popular software. Also, as you look through the list it becomes apparent that in the majority of the cases presented that the issue is in the implementation of the interfaces to the SMTP protocols. Many of these exploits involve sending additional data as part of an SMTP command to a receiving client/server application that is not protected from writing out of the memory buffer resulting in either an attack upon the availability or integrity of the application and/or system running the software.

Reference	Summary/Description
CVE-1999-0404	Buffer overflow in the Mail-Max SMTP server for Windows systems allows remote command execution.
CVE-1999-0682	Microsoft Exchange 5.5 allows a remote attacker to relay email (i.e. spam) using encapsulated SMTP addresses, even if the anti-relaying features are enabled
CVE-1999-0759	Buffer overflow in FuseMAIL POP service via long USER and PASS commands. BID 634
CAN-1999-0250	Denial of service in Qmail through long SMTP commands.
CAN-1999-0261	Netmanager Chameleon SMTPd has several buffer overflows that cause a crash. buffer overflow with 'HELO hostname' and hostname over 471 chars.
CAN-1999-0284	Denial of service to NT mail servers including Ipswitch, Mdaemon, and Exchange through a buffer overflow in the SMTP HELO command.
CAN-1999-0419	When the Microsoft SMTP service attempts to send a message to a server and receives a 4xx error code, it quickly and repeatedly attempts to redeliver the message, causing a denial of service.
CAN-1999-0512	A mail server is explicitly configured to allow SMTP mail relay, which allows abuse by spammers – this is if they allow relaying
CAN-1999-1012	SMTP component of Lotus Domino 4.6.1 on AS/400, and

	possibly other operating systems, allows a remote attacker to crash the mail server via a long string. If an attacker connects to the SMTP port (25) and sends about 200-300 bytes the server will die.
CAN-1999-1043	Microsoft Exchange Server 5.5 and 5.0 does not properly handle (1) malformed NNTP data, or (2) malformed SMTP data, which allows remote attackers to cause a denial of service (application error).
CAN-1999-1200	Vintra SMTP MailServer allows remote attackers to cause a denial of service via a malformed "EXPN *@" command.
CAN-1999-1265	SMTP server in SLmail 3.1 and earlier allows remote attackers to cause a denial of service via malformed commands whose arguments begin with a "(" (parenthesis) character, such as (1) SEND, (2) VRFY, (3) EXPN, (4) MAIL FROM, (5) RCPT TO.
CAN-1999-1511	Buffer overflows in Xtramail 1.11 allow attackers to cause a denial of service (crash) and possibly execute arbitrary commands via (1) a long PASS command in the POP3 service, (2) a long HELO command in the SMTP service, or (3) a long user name in the Control Service.
CAN-1999-1516	A buffer overflow in TenFour TFS Gateway SMTP mail server 3.2 allows an attacker to crash the mail server and possibly execute arbitrary code by offering more than 128 bytes in a MAIL FROM string.
CAN-1999-1521	Computalynx CMail 2.4 and CMail 2.3 SP2 SMTP servers are vulnerable to a buffer overflow attack in the MAIL FROM command that may allow a remote attacker to execute arbitrary code on the server.
CAN-1999-1529	A buffer overflow exists in the HELO command in Trend Micro Interscan VirusWall SMTP gateway 3.23/3.3 for NT, which may allow an attacker to execute arbitrary code.
CVE-2000-0033	InterScan VirusWall SMTP scanner does not properly scan messages with malformed attachments.
CVE-2000-0075	Super Mail Transfer Package (SMTP), later called MsgCore, has a memory leak which allows remote attackers to cause a denial of service by repeating multiple HELO, MAIL FROM, RCPT TO, and DATA commands in the same session.
CVE-2000-0428	InterScan VirusWall includes the ability to scan for virii in uuencoded files. Due to an unchecked buffer in the code, if a uuencoded file is sent that includes an embedded final filename of more than 128 characters, arbitrary remote code can be executed at the privilege level of the VirusWall software
CVE-2000-0447	Buffer overflow in WebShield SMTP 4.5.44 allows remote

	attackers to execute arbitrary commands via a long configuration parameter to the WebShield remote management service. Network Associates WebShield SMTP is susceptible to a buffer overflow attack if 208 or more bytes of data accompanying a configuration parameter is transmitted to the remote management service listening at port 9999. It is possible to force the program to execute arbitrary code at the privelege level of the service's account (default SYSTEM).
CVE-2000-0448	By default, Network Associates WebShield SMTP runs the management agent on port 9999. A remote user may gain access to this agent and modify the configuration of WebShield SMTP simply by connecting to this particular port. Issuing the command "GET_CONFIG<CR>" will return the current configuration. The management agent grants access based on a list of authorized hostnames, but will grant access to any IP adress which cannot be resolved to a hostname (WINS, DNS, netbios) even if 'MailCfg' is set to only allow configuration from localhost.
CVE-2000-0582	Check Point FireWall-1 4.0 and 4.1 allows remote attackers to cause a denial of service by sending a stream of invalid commands (such as binary zeros) to the SMTP Security Server proxy. Sending a stream of binary zeros (or other invalid SMTP commands) to the SMTP port on the firewall raises the target system's load to 100% while the load on the attacker's machine remains relatively low.
CVE-2000-0738	WebShield SMTP 4.5 allows remote attackers to cause a denial of service by sending e-mail with a From: address that has a . (period) at the end, which causes WebShield to continuously send itself copies of the e-mail. this vulnerability can be exploited by sending an email with a dot character trailing the domain name such as 'user@companyxyz.com
CVE-2000-0932	MAILsweeper for SMTP 3.x does not properly handle corrupt CDA documents in a ZIP file and hangs, which allows remote attackers to cause a denial of service.
CVE-2000-0990	cmd5checkpw 0.21 and earlier allows remote attackers to cause a denial of service via an "SMTP AUTH" command with an unknown username.
CVE-2000-1022	Due to improper input validation and error trapping, supplying cmd5checkpw with a non-existent username will cause it to segfault. In turn, the qmail-smtpd-auth Qmail patch incorrectly interprets this failure as a successful authentication. As a result, an attacker providing invalid input to cmd5checkpw can create a falsely-authenticated session, leaving the victim host open to receiving and

	forwarding mail from unauthenticated systems.
CVE-2000-1047	Buffer overflow in SMTP service of Lotus Domino 5.0.4 and earlier allows remote attackers to cause a denial of service and possibly execute arbitrary commands via a long ENVID keyword in the "MAIL FROM" command. The problem exists in the ENVID variable, as specified in RFC 1891. The SMTP server does not conduct adequate bounds checking on the ENVID keyword of the "MAIL FROM:" field. This makes it possible for a malicious user to custom craft an ENVID that could result in remote execution of code as the UID the SMTP server is operating as.
CAN-2000-0158	Buffer overflow in MMDF server allows remote attackers to gain privileges via a long MAIL FROM command to the SMTP daemon. By sending long, well crafted buffers to the smtpd mail daemon, as part of a "MAIL FROM:" command, it may be possible for an attacker to gain access to the machine under the user running the smtpd program.
CAN-2000-0657	Buffer overflow in AnalogX proxy server 4.04 and earlier allows remote attackers to cause a denial of service via a long HELO command in the SMTP protocol.
CAN-2000-1129	McAfee WebShield SMTP 4.5 allows remote attackers to cause a denial of service via a malformed recipient field In the event that WebShield SMTP receives an outgoing email containing six "%20" followed by any character within the recipient field, the application will crash, resulting in an access violation error upon processing of the email
CAN-2000-1203	Lotus Domino SMTP server 4.63 through 5.08 allows remote attackers to cause a denial of service (CPU consumption) by forging an email message with the sender as bounce@[127.0.0.1] (localhost), which causes Domino to enter a mail loop. If this occurs, the server will attempt to bounce the message and will go into a bounce loop and consume all of the system's CPU, requiring that the server be restarted and the message be manually removed from the queue.
CVE-2001-0039	IMail server SMTP service is subject to a denial of service. By specifying a base 64encoded SMTP AUTH password containing 80 to 136 bytes, the IMail server will stop responding and refuse any new connections.
CVE-2001-0280	Buffer overflow in MERCUR SMTP server 3.30 allows remote attackers to execute arbitrary commands via a long EXPN command.
CVE-2001-0494	Buffer overflow in IPSwitch IMail SMTP server 6.06 and

	possibly prior versions allows remote attackers to execute arbitrary code via a long From: header.
CVE-2001-0504	Vulnerability in authentication process for SMTP service in Microsoft Windows 2000 allows remote attackers to use incorrect credentials to gain privileges and conduct activites such as mail relaying.
CVE-2001-0690	Format string vulnerability in exim (3.22-10 in Red Hat, 3.12 in Debian and 3.16 in Conectiva) in batched SMTP mode allows a remote attacker to execute arbitrary code via format strings in SMTP mail headers. The vulnerability has to do with handling of the hostname string in an email address argumenting the 'From:' field.
CVE-2002-0055	SMTP service in Microsoft Windows 2000, Windows XP Professional, and Exchange 2000 to cause a denial of service via a command with a malformed data transfer (BDAT) request.
CAN-2002-0054	SMTP service in (1) Microsoft Windows 2000 and (2) Internet Mail Connector (IMC) in Exchange Server 5.5 does not properly handle responses to NTLM authentication, which allows remote attackers to perform mail relaying via the server. By design, the Windows 2000 SMTP service and the Exchange Server 5.5 IMC, upon receiving notification from the NTLM authentication layer that a user has been authenticated, should perform additional checks before granting the user access to the service. The vulnerability results because the affected services don't perform this additional checking correctly. In some cases, this could result in the SMTP service granting access to a user solely on the basis of their ability to successfully authenticate to the server.
CAN-2002-0416	Buffer overflow in SH39 MailServer 1.21 and earlier allows remote attackers to cause a denial of service, and possibly execute arbitrary code, via a long command to the SMTP port.
CAN-2002-0432	A vulnerability has been reported in the SMTP support included in some versions of Citadel/UX. When intially connecting to the SMTP server, including an oversized parameter with the HELO command will cause a buffer overflow condition. Stack memory will be corrupted, leading to a denial of service attack. It may be possible to exploit this vulnerability to execute arbitrary code. This has not been confirmed

CVE	Description
CVE-1999-0057	The Vacation program (used to notify people who send you mail that you may not be reading it right away) allows command execution by remote users through a sendmail command.
CVE-1999-0095	When the debug command (-d) in Sendmail is enabled, it may allow attackers to execute commands as root.
CVE-1999-0096	The Sendmail decode alias can be used to overwrite sensitive files, which allows an attacker to gain control of a system.
CVE-1999-0129	Sendmail allows local users to write to a file and gain group permissions via a . forward or :include: file.
CVE-1999-0130	Local users can start Sendmail in daemon mode and gain root privileges.
CVE-1999-0131	Buffer overflow and denial of service in Sendmail 8.7.5 and earlier through GECOS field gives root access to local users.
CVE-1999-0145	Sendmail WIZ command enabled, allowing root access.
CVE-1999-0204	Sendmail 8.6.9 allows remote attackers to execute root commands using ident
CVE-1999-0203	In Sendmail, attackers can gain root privileges via SMTP by specifying an improper address in either the MAIL or RCPT verbs that would cause the mail to bounce to a program.
CVE-1999-0204	Sendmail 8. 6. 9 allows remote attackers to execute root commands using ident
CVE-1999-0206	A MIME buffer overflow in Sendmail 8.8.0 and 8.8.1 gives root access.
CVE-1999-0393	Remote attackers can cause a denial of service in Sendmail 8.8.x and 8.9.2 by sending messages with a large number of headers.
CVE-1999-0404	Buffer overflow in the Mail-Max SMTP server for Windows systems allows remote command execution.
CVE-1999-0478	Denial of service in HP-UX sendmail 8.8.6 related to accepting connections
CVE-1999-0976	Sendmail allows local users to reinitialize the aliases database via the newaliases command, then cause a denial of service by interrupting Sendmail.
CVE-1999-1109	Sendmail before 8.10.0 allows remote attackers to cause a denial of service by sending a series of ETRN commands then disconnecting from the server, while Sendmail continues to process the commands after the connection has been terminated.
CVE-1999-1309	Sendmail before 8.6.7 allows local users to gain root access via a large value in the debug (-d) command line option.

CVE	Description
CVE-2000-0042	Buffer overflow in CSM mail server allows remote attackers to cause a denial of service or execute commands via a long HELO command.
CVE-2000-0319	mail. local in Sendmail 8.10.x does not properly identify the . \n string which identifies the end of message text, which allows a remote attacker to cause a denial of service or corrupt mailboxes via a message line that is 2047 characters long and ends in \n.
CVE-2000-0348	A vulnerability in the Sendmail configuration file sendmail. cf as installed in SCO UnixWare 7.1.0 and earlier allows an attacker to gain root privileges.
CVE-2000-0506	The "capabilities" feature in Linux before 2.2.16 allows local users to cause a denial of service or gain privileges by setting the capabilities to prevent a setuid program from dropping privileges, aka the "Linux kernel setuid/setcap vulnerability. "
CVE-2001-0653	Sendmail 8.10.0 through 8.11.5, and 8.12.0 beta, allows local users to modify process memory and possibly gain privileges via a large value in the 'category' part of debugger (-d) command line arguments, which is interpreted as a negative number.
CVE-2001-1075	poprelayd script before 2.0 in Cobalt RaQ3 servers allows remote attackers to bypass authentication for relaying by causing a "POP login by user" string that includes the attacker's IP address to be injected into the maillog log file.
CVE-2002-0906	Buffer overflow in Sendmail before 8.12.5, when configured to use a custom DNS map to query TXT records, allows remote attackers to cause a denial of service and possibly execute arbitrary code via a malicious DNS server.

CAN	Description
CAN-1999-0098	Buffer overflow in SMTP HELO command in Sendmail allows a remote attacker to hide activities.
CAN-1999-0163	In older versions of Sendmail, an attacker could use a pipe character to execute root commands.
CAN-1999-0205	Denial of service in Sendmail 8.6.11 and 8.6.12.
CAN-1999-0418	Denial of service in SMTP applications such as Sendmail, when a remote attacker (spammer) uses many "RCPT TO" commands in the same connection.
CAN-1999-1468	rdist in various UNIX systems uses popen to execute sendmail, which allows local users to gain root privileges by modifying the IFS (Internal Field Separator) variable.
CAN-2001-1349	Sendmail before 8.11.4, and 8.12.0 before 8.12.0.Beta10, allows local users to cause a denial of service and possibly corrupt the heap and gain privileges via race conditions in

	signal handlers.
CAN-2001-0713	Sendmail before 8.12.1 does not properly drop privileges when the -C option is used to load custom configuration files, which allows local users to gain privileges via malformed arguments in the configuration file whose names contain characters with the high bit set, such as (1) macro names that are one character long, (2) a variable setting which is processed by the setoption function, or (3) a Modifiers setting which is processed by the getmodifiers function.
CAN-2001-0714	Sendmail before 8.12.1, without the RestrictQueueRun option enabled, allows local users to cause a denial of service (data loss) by (1) setting a high initial message hop count option (-h), which causes Sendmail to drop queue entries, (2) via the -qR option, or (3) via the -qS option.
CAN-2001-0715	Sendmail before 8.12.1, without the RestrictQueueRun option enabled, allows local users to obtain potentially sensitive information about the mail queue by setting debugging flags to enable debug mode.
CAN-2001-0789	Format string vulnerability in avpkeeper in Kaspersky KAV 3.5.135.2 for Sendmail allows remote attacker to cause a denial of service or possibly execute arbitrary code via a malformed mail message.
CAN-2001-1349	Sendmail before 8.11.4, and 8.12.0 before 8.12.0.Beta10, allows local users to cause a denial of service and possibly corrupt the heap and gain privileges via race conditions in signal handlers
CAN-2002-0651	Buffer Overflow in Multiple DNS Resolver Libraries

Looking at the above alerts one thing becomes clear – the same simplicity that has made SMTP the defacto protocol for e-mail, has also left numerous opportunities for an attacker to gain escalated privileges on a variety of email applications remotely because port 25 needs to be open and listening to receive/relay e-mail. For most e-mail tools, this service needs to run with escalated privileges so that it can deliver (write) information into a variety of user folders and files.

Part 2: Specific Exploit

Exploit Details

Name:
Buffer Overflow in DNS resolver libraries

CVE:
CERT Advisory CA-2002-19 Buffer Overflows in Multiple DNS Resolver Libraries
CAN-2002-0651
BID 5100: Multiple Vendor libc DNS Resolver Buffer Overflow
CERT/CC Vulnerability Note VU#803539

Operating Systems/Applications:
Cray UNICOS 9.2.4
Cray UNICOS 9.2
Cray UNICOS 9.0.2.5
Cray UNICOS 9.0
Cray UNICOS 8.3
Cray UNICOS 8.0
FreeBSD FreeBSD 4.6-RELEASE
FreeBSD FreeBSD 4.6
FreeBSD FreeBSD 4.5-STABLE
FreeBSD FreeBSD 4.5-RELEASE
FreeBSD FreeBSD 4.5
FreeBSD FreeBSD 4.4-STABLE
FreeBSD FreeBSD 4.4-RELENG
FreeBSD FreeBSD 4.4
FreeBSD FreeBSD 4.3-STABLE
FreeBSD FreeBSD 4.3-RELENG
FreeBSD FreeBSD 4.3-RELEASE
FreeBSD FreeBSD 4.3
ISC BIND 9.2.1
+ Caldera OpenUnix 8.0
ISC BIND 9.2
+ Conectiva Linux 8.0
+ MandrakeSoft Linux Mandrake 8.2
+ MandrakeSoft Linux Mandrake 8.1 ia64
+ MandrakeSoft Linux Mandrake 8.1
+ RedHat Linux 7.3 i386
+ RedHat Linux 7.3
ISC BIND 9.1.3
+ RedHat Linux 7.2 ia64
+ RedHat Linux 7.2 i686
+ RedHat Linux 7.2 i586

+ RedHat Linux 7.2 i386
+ RedHat Linux 7.2
+ S.u.S.E. Linux 8.0i386
+ S.u.S.E. Linux 8.0
+ S.u.S.E. Linux 7.3sparc
+ S.u.S.E. Linux 7.3ppc
+ S.u.S.E. Linux 7.3i386
+ S.u.S.E. Linux 7.3
ISC BIND 9.1.2
+ Conectiva Linux 7.0
+ S.u.S.E. Linux 7.2i386
+ S.u.S.E. Linux 7.2
ISC BIND 9.1.1 + MandrakeSoft Linux Mandrake 8.0 ppc
+ MandrakeSoft Linux Mandrake 8.0
ISC BIND 9.1
+ Caldera OpenUnix 8.0
+ HP Secure OS software for Linux 1.0
+ RedHat Linux 7.1 ia64
+ RedHat Linux 7.1 i386
+ RedHat Linux 7.1 alpha
+ RedHat Linux 7.1
+ S.u.S.E. Linux 7.1x86
+ S.u.S.E. Linux 7.1sparc
+ S.u.S.E. Linux 7.1ppc
+ S.u.S.E. Linux 7.1alpha
+ S.u.S.E. Linux 7.1
ISC BIND 9.0
+ S.u.S.E. Linux 7.0sparc
+ S.u.S.E. Linux 7.0ppc
+ S.u.S.E. Linux 7.0i386
+ S.u.S.E. Linux 7.0alpha
+ S.u.S.E. Linux 7.0
ISC BIND 8.2.5
ISC BIND 8.2.4
ISC BIND 8.2.3
ISC BIND 8.2.2
ISC BIND 8.2.1
ISC BIND 8.2
ISC BIND 8.1.2
ISC BIND 8.1.1
ISC BIND 8.1
ISC BIND 4.9.8
ISC BIND 4.9.7
ISC BIND 4.9.6
ISC BIND 4.9.5
ISC BIND 4.9.4

ISC BIND 4.9.3
ISC BIND 4.9
NetBSD NetBSD 1.5.2
NetBSD NetBSD 1.5.1
NetBSD NetBSD 1.5 x86
NetBSD NetBSD 1.5 sh3
NetBSD NetBSD 1.5
NetBSD NetBSD 1.4.3
NetBSD NetBSD 1.4.2 x86
NetBSD NetBSD 1.4.2 SPARC
NetBSD NetBSD 1.4.2 arm32
NetBSD NetBSD 1.4.2 Alpha
NetBSD NetBSD 1.4.2
NetBSD NetBSD 1.4.1 x86
NetBSD NetBSD 1.4.1 SPARC
NetBSD NetBSD 1.4.1 sh3
NetBSD NetBSD 1.4.1 arm32
NetBSD NetBSD 1.4.1 Alpha
NetBSD NetBSD 1.4.1
NetBSD NetBSD 1.4 x86
NetBSD NetBSD 1.4 SPARC
NetBSD NetBSD 1.4 arm32
NetBSD NetBSD 1.4 Alpha
NetBSD NetBSD 1.4
OpenBSD OpenBSD 3.1
OpenBSD OpenBSD 3.0
OpenBSD OpenBSD 2.9
OpenBSD OpenBSD 2.8
OpenBSD OpenBSD 2.7

Protocols/Services:
Libbind resolver libraries which are used in sendmail to resolve SMTP MX records

Description:

"The Domain Name System (DNS) provides name, address, and other information about Internet Protocol (IP) networks and devices. By issuing queries to and interpreting responses from DNS servers, IP-enabled network operating systems can access DNS information. When an IP network application needs to access or process DNS information, it calls functions in the stub resolver library, which may be part of the underlying network operating system. On BSD-based systems, DNS stub resolver functions are implemented in the system library libc. In ISC BIND, they are implemented in libbind, and on GNU/Linux-based systems, they are implemented in glibc. " (VU#803539)

Sendmail uses the BIND resolver API, and is compiled with the BIND resolver library (libbind) so that it can resolve domains in the email addresses to which it sends email. Sendmail relies on two library functions that do this transparently, called *gethostbyname()* and *gethostbyaddr()*. These and other related procedures, including *getanswer()* and *getnetanswer()*, are grouped in a separate DNS resolver library. It is the reply to these functions that allows the attacker to write and execute code on the mail server as the userid running the application that is querying DNS through the resolver library. As a result, the sendmail application, which is often run from an account with escalated privileges (root) could be leveraged from its queries for host ip/address pairs from the vulnerable resolver library. A buffer overflow returned to the sendmail process via the response could either crash the sendmail application or potentially execute commands crafted by the attacker.

Variants:

While not a true variant, a subsequent exploit was released that was quickly confused with this one. This vulnerability involves a buffer overflow in which sendmail is configured to use a custom DNS map to query TXT records that allows remote attackers to cause a denial of service and possibly execute arbitrary code via a malicious DNS server. Whereas the exploit in question is dependent upon libbind, this exploit requires the use of a customized dns map definition to query unsafe DNS TXT records. For more information, see CAN-2002-0906

Another denial of service attack was reported that is also similar to this exploit. In CAN-2002-1146 an exploit has been reported with the stub resolver library [res_search() and res_query()] in BIND. A stub resolver relies on the services of a recursive name server on the connected network or a "nearby" network. This scheme allows the host to pass on the burden of the resolver function to a name server on another host because it allows all of the workstations to share the cache of the recursive name server and hence reduce the number of domain requests exported by the local network. BIND 4, BIND 8.2.x stub resolver libraries, glibc 2.2.5 and earlier, libc, and libresolv libraries use the maximum buffer size instead of the actual size when processing a DNS response [getanswer()], which causes the stub resolvers to read past the actual boundary, allowing remote attackers to cause a denial of service.

Protocol Description

In an early section we began our discussion of the manner in which sendmail communicated via the SMTP protocol. This preliminary examination served as an introduction and was application independent – all mail transfer agents speak

SMTP. The simplicity of this protocol results in a need to rely upon other more complex programs for some of its data – in particular, the addresses to which the mail will be delivered.

A mail transaction involves several data objects which are communicated as arguments to different commands which are referred to as verbs in the RFC. Once these arguments are transmitted, they are held pending the confirmation communicated by the end of mail data indication, which finalizes the transaction. The model for this is that distinct buffers are provided to hold the types of data objects, that is, there is a reverse-path buffer, a forward-path buffer, and a mail data buffer. Specific commands cause information to be appended to a specific buffer, or cause one or more buffers to be cleared.

To demonstrate the simplicity of smtp, one need only examine the minimal set of commands, referred to as verbs, needed to send email across a network:

Verb	Description
HELO	Command used by the SMTP-sender to identify itself to an SMTP-reciever that has responded with a 220 reply code to an initial inquiry on port 25. The argument field contains the host name of the sender-SMTP. The receiver-SMTP identifies itself to the sender-SMTP in the connection greeting reply, and in the response to this command.
MAIL	This command is used to initiate a mail transaction in which the mail data is delivered to one or more recipient “mailboxes”. The argument field contains a reverse-path that consists of an optional list of hosts and the sender mailbox. When the list of hosts is present, it is a "reverse" source route and indicates that the mail was relayed through each host on the list (the first host in the list was the most recent relay). This list is used as a source route to return non-delivery notices to the sender. As each relay host adds itself to the beginning of the list, it must use its name as known in the IPCE to which it is relaying the mail rather than the IPCE from which the mail came so as to assure a viable return path.
RCPT	Identifies the intended recipient of the message. There is 1 RCPT command per intended recipient. The forward-path consists of an optional list of hosts and a required destination mailbox. When the list of hosts is present, it is a source route and indicates that the mail must be relayed to the next host on the list.
DATA	Command used to transmit the mail messages. The receiver treats the lines following the command as mail data from the sender. This command causes the mail data from this command to be appended to the mail data buffer. The mail data may contain any of the 128 ASCII character codes.
QUIT	Terminates the connection between the systems
RSET	Aborts the current transaction and causes both systems to reset-

	this deletes any information about the sender, recipient, and message.
VRFY	This command asks the server receiving mail to verify the recipients address without sending e-mail.
NOOP	Command that causes the receiving MTA to send a reply code (220).
EXPN	Used to expand a mailing list established on the target MTA.
TURN	This command reverses the roles of the MTA's involved in an exchange without having to re-establish a connection.

The following is a sample of a simple SMTP exchange between 2 servers, host1 and host2. It provides an example of the exchange that is made between the systems. Note the mail program has already executed its routines for querying DNS for the IP addresses of the MAIL FROM and RCPT TO designations (in red). It is for these commands that the DNS resolver library is called so that SMTP knows where to route it's connection request (HELO):

```
220 postoffice.host1.com ESMTP Sendmail 8.12.5/8.12.5; Mon, 11 Oct 2002
04:22:25 GMT
HELO host2.com
250 postoffice.host1.com Hello mailserver.host2.com [10.1.0.1], pleased to
meet you
MAIL FROM: marco@postoffice.host1.com
250 marco@postoffice.host1.com... Sender ok
RCPT TO: polo@host2.com
250 polo@host2.com... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Message-ID: <20021011184815.33912@postoffice.host1.com>
From: Marco <marco@host1.com>
To: Polo <polo@host2.com>
Subject: Just saying hi
Date: 11 Oct 2002
I just wanted to say Hi.
.
250 EAA88467 Message accepted for delivery
QUIT
221 postoffice.host1.com closing connection
```

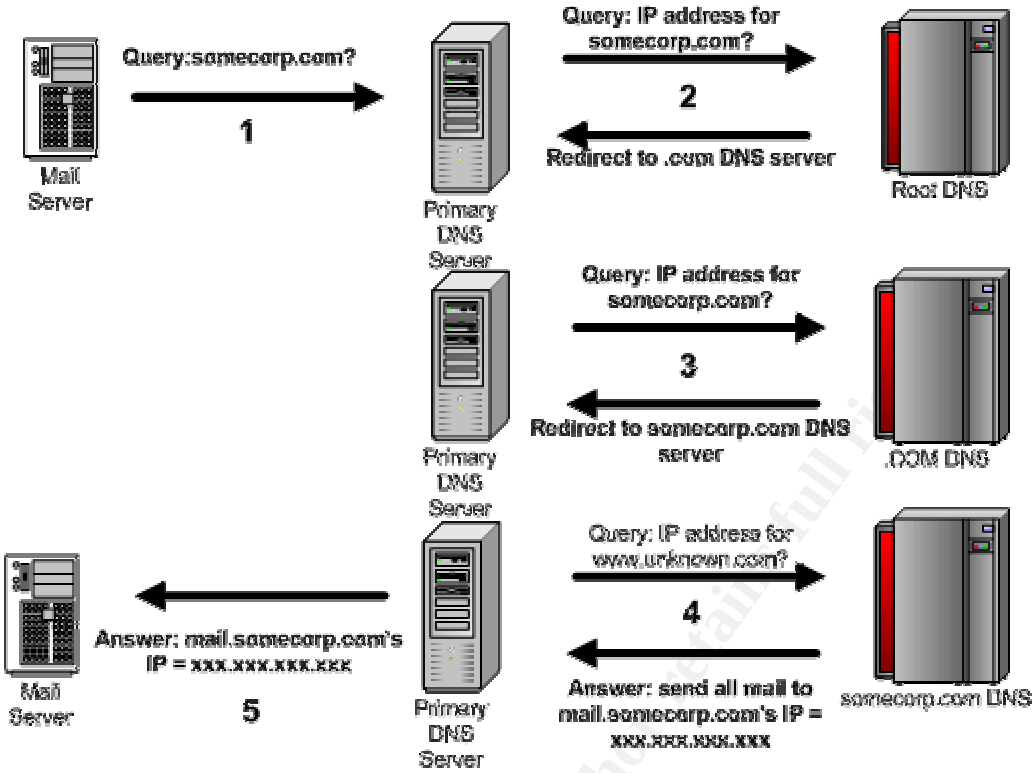
Mail is addressed in the format, username@some.domain.com, and that SMTP on its own does not possess the ability to resolve this address. This is where SMTP and sendmail, become dependent upon the dns resolver. When referring

to “the resolver", we do not mean any specific application, but rather refer to the *resolver library*, a collection of functions [*gethostbyname()*, *gethostbyaddr()*, *getanswer()* and *getnetanswer()*] that can be found in the standard C library (netdb.h). The central routines are gethostbyname() and gethostbyaddr() which look up all IP addresses belonging to a host, and vice versa via querying DNS. This library is included when sendmail is compiled. As a result, the vulnerability within the resolver library is incorporated into the sendmail program and if exploited will run with the privileges of the sendmail program (typicallly root).

The problem is that sendmail is conforming to the SMTP standard (RFC 974) which requires that the application provides the record for any host prior to sending mail to it. For sendmail to send a message via SMTP, it needs to know the IP address of the machine which it will open a connection to on port 25. It requires the IP address be returned by the name server in any of 3 possible formats:

- An **MX** (Mail Exchanger) record list one or more machines that will receive mail from the site (multiple machines will receive it in a pre-defined order of preference)
- An **A** (address) record provides the IP address directly.
- A **CNAME** (alias) record that will refer sendmail to the real name which sendmail will use to try to retrieve an A or MX record

Referring to the diagram below, Step 1 below occurs when a local mail server queries the local DNS server for an address/ip pair. Once sendmail has the name of the destination (ex: user@somecorp.com), it calls *gethostbyname()* to get the needed network address(es). Presuming that the DNS server does not have that information cached, it proceeds through an iterative search seeking the address. From our previous brief discussion on the DNS protocol, we know that the IP/address pairs comprise an enormous amount of data that cannot efficiently be stored on a single server – thus the servers will either provide the needed address, or “point the server in the right direction” as it traverses through the hierarchical tree structure of DNS where the server can issue another query. In step 2 it begins its query at the root DNS server – which will reply with several com DNS servers who can help find the address. In step 3, the local DNS server repeats its query. Again, if the information is not found, the local DNS server is directed toward another server who can provide information on somecorp.com (Step 3). Finally in Step 4, somecorp.com will provide the local DNS server with the address where mail is to be directed. In step 5, the local DNS server will pass this pack to sendmail on the mail server. It then opens a network connection and attempts to deliver the mail.



DNS Query

How the exploit works

The key to exploiting this vulnerability lies utilizing the potential for a buffer overflow attack encapsulated in the format of the response a DNS server can provide to an address query. A buffer overflow occurs when a program or process tries to store more data in a buffer than it was designed to hold. Since buffers are created to contain a finite amount of data, the extra information - which has to go somewhere - can overflow into unprotected adjacent buffers, corrupting or overwriting the valid data held in them.

DNS messages have specific byte alignment requirements, resulting in padding in messages. In a few instances in the resolver code, this padding is not taken into account when computing available buffer space. The resolver is compiled as part of the sendmail program – so that when the sendmail binary needs to look up an ip address to delivery mail to, it calls upon the subroutines included in the resolver library. These extra bytes are then passed (unchecked) back to the sendmail program where they are used to exploit the server attacking availability and potentially the integrity of the system.

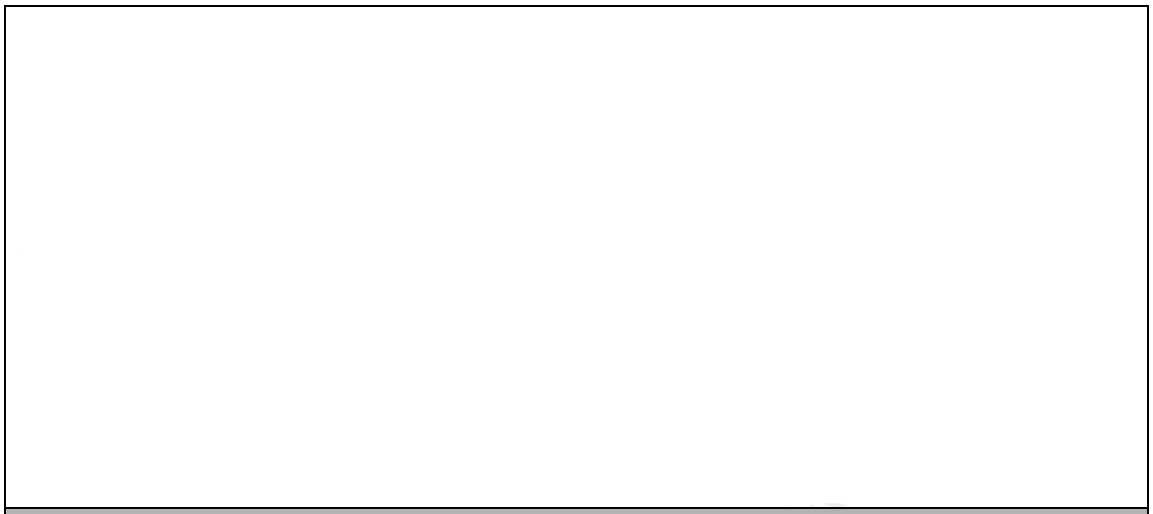
In the resolver library, two variables, gethnamaddr.c [getanswer()] and getnetnamadr.c [getnetanswer()], manage packet buffer parsing - a pointer to the byte we are looking at, and the remaining length on the buffer. As a result of the

remaining length pointer not being updated consistently, it is possible for an attacker to write a few bytes (for each record) outside the buffer in a malicious DNS response. When sendmail queries for an address to provide to SMTP it does so using these subroutines. By crafting the response records correctly, we can then write outside the memory on the mail server with an escalated privilege since sendmai is traditionally run as root – the buffer overflow is thus written as root into memory and (potentially) executed with root privileges. Unlike the more common buffer overflows in network daemons, any outgoing DNS query made to a hostile server could expose the vulnerability. The threat posed by this exploit is magnified when one realizes that in many environments, clients are permitted to make DNS queries directly against the name servers outside their DMZ. Since this is an outbound query, this exploit could affect a system that is behind a firewall. Most firewalls allow for:

- outbound UDP/53
- inbound UDP/53
- outbound TCP/53
- inbound TCP/53

The diagram below provides a top level process flow for implementing the exploitation scenario (external attacker) of this vulnerable and hopefully stresses the danger this exploit poses since it could be executed through a firewall. The workstation attempts to send mail outside the organization through whatever email reader the user employs and this mail is relayed to the organization's Mail Server. Presuming there is no cached information, the mail server, calling upon the resolver library functions it is compiled with, than attempts to make a query outside the organization's firewall to an external name server. The attacker reroutes the query from the intended destination DNS server to a server that they control (Evil DNS Server) through a tactic such as DNS spoofing. DNS spoofing involves faking a response from the client/target's DNS server in order to provide the wrong address/IP mapping to the targeted system. This is done by sending a spoofed response directly to the target system (in this case, the local DNS server which is queried by the mail server). The result is that the local DNS server initiates (or continues, depending on when the spoofing is executed) a lookup against an attacker controlled system rather than the trusted system it presumes it is communicating with. The query is then processed and sent back to the Mail Server through a firewall that would presumably be configured with a ruleset that accepts inbound responses to DNS queries. The response contains the extra bytes that are used to attack the availability and/or integrity of the mail server. These extra bytes could in theory do something as simple as crash the server, to the execution of code on the mail server, or worse, allow the attacker to install binaries on the system (such as netcat) to allow them access even through the firewall.

Diagram



Exploited Mail Server

How to use the exploit

At this time, no known exploit exists for this vulnerability. As such, we are left with the exercise of proposing a scenario how this vulnerability would be used. Some steps involve groundwork to set up services and employ deceptive tactics to direct queries to a controlled system which can respond with a malicious response:

1. The first requirement for the successful exploitation of this vulnerability is the establishment of a machine to respond to DNS queries with the response and the exploit code invoking the desired results.

2. The attacker is counting on the local sendmail process executing a DNS query against a DNS server they wish to spoof/intercept. At this point the sendmail program needs to query for an address it does not already know or have cached.

3. The attacker must force (or wait for) the vulnerable process to make DNS queries against the attacker-controlled DNS server. This would most easily be accomplished through DNS spoofing. For an external attacker, this step could prove the most daunting since they need to be able to monitor traffic so that they may inject their malicious responses.

4. When sendmail references the vulnerable resolver library, this library queries the malicious DNS server.

5. The attacker crafts their desired effect (denial of service, opening a remote shell, etc) which is returned to the sendmail server as part of the record request.

6. When the DNS response is processed [getanswer() or getnetanswer()], the response will include the expected answer plus the additional bytes that will allow for the buffer overflow condition will be exploited. The attacker's code may execute as the vulnerable process, which in this case means it runs as root since sendmail usually is setuid 0, providing local access to the attacker.

Signature of the attack

At this point, there is no know exploit which we can distill a signature from. Should an exploit become available, we would need to focus our attention on the packets returned from the (potentially) malicious DNS server to uncover the malicious code within the server's response. Given we have no signature for an attacker, we should at least begin by understanding what "normal" traffic would look like. Let's assume that mail.yousite.com is your mail server, your local DNS server is dns.yoursite.com, and destination is some address that your sendmail server wishes to send email to. A reply would resemble the following:

13:45:13.947300 dns.yoursite.com.53 > mail.yoursite.com.3163: 1 q: destination. 3/4/6 destination. CNAME destination, destination. CNAME destination. destination. A destination.xx.yy.zz (283)

Recall that smtp requires an A record – this is what we have at the end of the above captured network traffic. Abnormal traffic would include formats dissimilar from the above. Abnormal may also be defined as different answers – recall that DNS will send multiple queries to different servers in hopes of improving the response ability of UDP.

While we have no signatures for the response, there may be clues about the execution of such an attack located in syslog. Presuming that the response may take more than 1 attempt to execute cleanly, if you notice a sudden rise in the following syslog message types (there are several with a variety of messages appended to the error), it may be worth further scrutiny:

Malformed response from [xxx.xxx.xxx.xxx].53 <MESSAGE>

If the attacker fails to correctly spoof a DNS server, you may suddenly see entries in your syslog file indicating that your name server received a response from a remote name server but that it hadn't queried that name server, and therefore didn't expect (and dropped) the response:

Response from unexpected source ([XXX.XXX.XXX.XXX].53)

How to protect against it

> "*The best defense for sendmail attacks is to disable sendmail if you are not using it to receive mail over a network. If you must run sendmail, ensure that you are using the latest version with all relevant security patches.* " – Scambray, McClure and Kurtz

The Quick Fix

1. If you are running your MTA on a GNU/linux system, you can change the default options used in the name service. You can reduce your risk by making sure the "networks" line in /etc/nsswitch.conf does not refer to DNS – it should instead refer to 'files'. This will mitigate the risk as you no longer will be routing your inquiries to the potentially malicious DNS servers.
2. Don't allow mail forwarding – removing this ability will reduce the ability of an attacker to force your mail server to make queries which could increase the likelihood their server will be queried.

Use of a local caching DNS server is not an effective workaround!

When this advisory was initially published, it was thought that a caching DNS server that reconstructs DNS responses would prevent malicious code from reaching systems with vulnerable resolver libraries. This workaround is not sufficient. It does not prevent some DNS responses that contain malicious code from reaching clients, whether or not the responses are reconstructed by a local caching DNS server. DNS responses containing code that is capable of exploiting the vulnerabilities described in VU#803539 and VU#542971 can be cached and reconstructed before being transmitted to clients. Since the server may cache the responses, the malicious code could persist until the server's cache is purged or the entries expire.

Don't Run Sendmail as Root:

To reduce the risk associated with integrity based attacks, you can change the userid which sendmail runs as. In the case of a buffer overflow, the code that is written into memory is run with the same user privileges as the user who performs the write. For sendmail, that is often root. To reduce the immediate risk, you can engineer sendmail such that it does not run as root. Keep in mind *this does not reduce any risk from the vulnerability itself*, only the potential

consequences as a result of an integrity-based attack. If sendmail is run as a non-root user with limited privileges, the overflow code will also only be able to do what this non-root account can do. The authors of sendmail provide a very good write up on how to accomplish this. It can be found at: http://www.sendmail.org/secure-install.html

Elevate the difficulty for successful DNS Spoofing:

DNSSEC (DNS Security) is a technique for securing the Domain Name System. It is a set of extensions to DNS, which provide end-to-end authenticity and integrity and was designed to protect the Internet from certain attacks. By accepting signed records only from trusted sources which are validated locally, you can decrease your chances of accepting records from a malicious source.

I don't know that a snort rule could catch this sort of attack. However, one of the IDS products that track state and monitor protocol behavior should be able notice this sort of pattern, especially one that can track outbound DNS requests on a per-host basis and then match it to incoming replies and check the MAC addresses of the request vs. response. A rule that matched an outbound DNS request's destination MAC address with the source MAC address coming back would provide a moderate confidence rule but could be fooled. A rule that looked for multiple responses to a request and looked for different IP addresses being provided as the answers in the different responses would also provide moderate or better confidence but again could potentially be fooled (though maybe not as easily).

Root Cause Resolution:

To remove this vulnerability, the vendors will need to correct the remaining length pointer to remove the ability to write outside the buffer allocated. This can be carried out by making sure a max length is not exceeded in the response. Most of the vendors have issued this repair as of this writing.

Sendmail uses the BIND resolver API, and is commonly linked with the BIND resolver library (libbind). As a result, it is necessary to either upgrade BIND (recommended) or patch your current version. ISC has reported that versions 9.2+ are not vulnerable. You then need to rebuild sendmail which relies upon the resolver libraries and restart your sendmail processes. The upgrades can be acquired at the isc.org site (provided in the references section of this paper).

Source code/ Pseudo code

There is currently no released exploit code for this vulnerability. Pseudo-code for this exploit would include:

```
struct netent
{
        # This is a data structure for information from network database

  char         *n_name;       # Points to the official name of the network
  char         **n_aliases;    # Points to the first element in a list of pointers to
                               #  alternate names (aliases) for the network
  int          n_addrtype;    # type of the network number returned
  unsigned long   n_net;      # the network number (in host order).
 };


QueryResponse(){
        netent EvilResponse;        #DNS query expects a netent struct returned

        # Evil Actions is your choice – DoS, Remote Shell Command, etc

        ValidResponse = GetValidResponse();

        # Add the evil code in the buffer

        EvilResponse = append(ValidResponse, EvilActions)

  return (Evil_Reponse);
}
```

References/Additional Information

http://www.kb.cert.org/vuls/id/542971

http://www.cert.org/advisories/CA-2002-19.html

http://online.securityfocus.com/bid/5100

http://www.cert.org/advisories/CA-2002-19.html

http://www.ietf.org/rfc/rfc0821.txt

http://www.isc.org/products/BIND/bind-security.html

http://online.securityfocus.com/infocus/1221

http://www.ietf.org/internet-drafts/draft-ietf-dnsext-dns-threats-02.txt

Costales, Bryan, Eric Allman. Sendmail 2nd Edition. O'Reilly & Associates, Inc, 1997

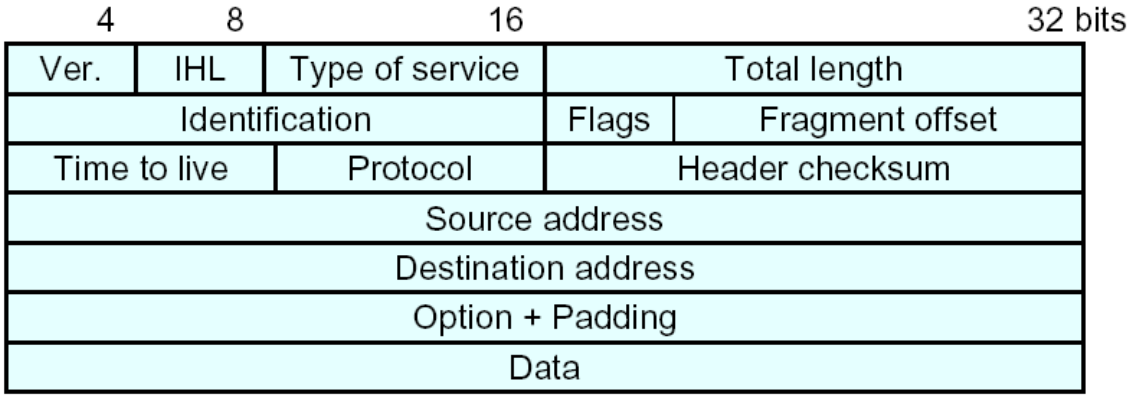
Garfinkle, Simson, Gene Spafford. Practical UNIX and Internet Security, 2nd Edition. O'Reilly & Associates, Inc, 1996. 811

Scambray, Joel, Stuart McClure, George Kurtz. Hacking Exposed, Network Security Secrets & Solutions 2nd Edition. The McGraw Hill Companies, 2001. 324,325.

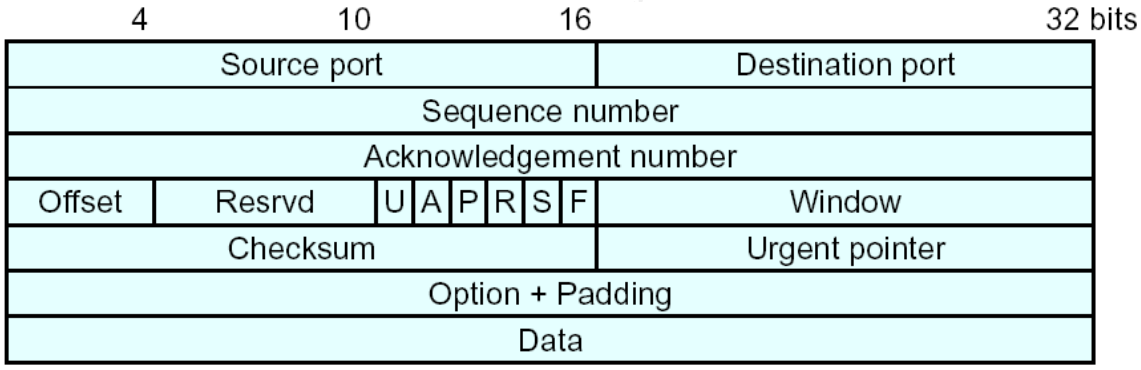
Postel, J., ed., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, USC/Information Sciences Institute, NTIS AD Number A111091, September 1981.

Appendix A: Protocol diagrams

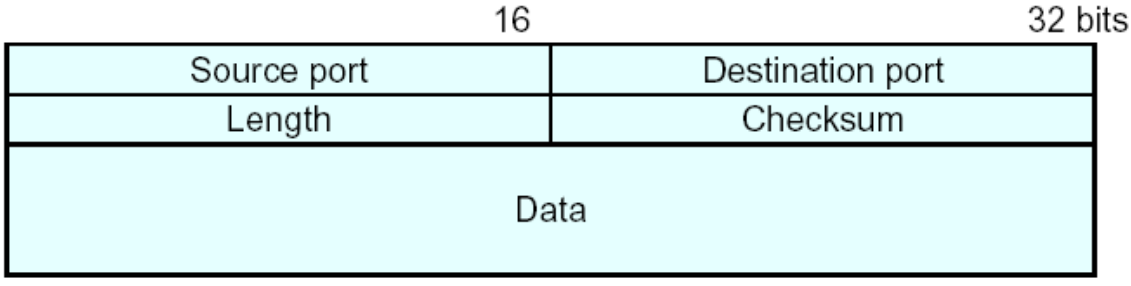
IP header structure



TCP Header Structure



UDP Header Structure



Appendix B: Unix Man Page for netdb.h

Unix Man Page

NAME

gethostbyname, gethostbyaddr, sethostent, endhostent, herror, hstrerror - get network host entry

SYNOPSIS

```
#include <netdb. h>
extern int h_errno;

struct hostent *gethostbyname(const char *name);

#include <sys/socket. h>        /* for AF_INET */
struct hostent *gethostbyaddr(const char *addr, int len, int type);

void sethostent(int stayopen);

void endhostent(void);

void herror(const char *s);

const char * hstrerror(int err);
```

DESCRIPTION

The gethostbyname() function returns a structure of type hostent for the given host name. Here name is either a host name, or an IPv4 address in standard dot notation, or an IPv6 address in colon (and possibly dot) notation. (See RFC 1884 for the description of IPv6 addresses.) If name is an IPv4 or IPv6 address, no lookup is performed and gethostbyname() simply copies name into the h_name field and its struct in_addr equivalent into the h_addr_list[0] field of the returned hostent structure. If name doesn't end in a dot and the environment variable HOSTALIASES is set, the alias file pointed to by HOSTALIASES will first be searched for name. (See hostname(7) for the file format.) The current domain and its parents are searched unless name ends in a dot.

The gethostbyaddr() function returns a structure of type hostent for the given host address addr of length len and address type type. The only valid address type is currently AF_INET.

The sethostent() function specifies, if stayopen is true (1), that a connected TCP socket should be used for the name server queries and that the connection should remain open during successive queries. Otherwise, name server queries will use UDP datagrams.
The endhostent() function ends the use of a TCP connection for name server queries.

The (obsolete) herror() function prints the error message associated with the current value of h_errno on stderr.

The (obsolete) hstrerror() function takes an error number (typically h_errno) and returns the corresponding message string.

The domain name queries carried out by gethostbyname() and gethostbyaddr() use a combination of any or all of the name server named(8), a broken out line from /etc/hosts, and the Network Information Service (NIS or YP), depending upon the contents of the order line in /etc/host. conf. (See resolv+(8)). The default action is to query named(8), followed by /etc/hosts.

The hostent structure is defined in <netdb. h> as follows:

```
struct hostent {
        char    *h_name;        /* official name of host */
        char    **h_aliases;    /* alias list */
        int     h_addrtype;     /* host address type */
        int     h_length;       /* length of address */
        char    **h_addr_list;  /* list of addresses */
}
#define h_addr  h_addr_list[0]  /* for backward compatibility */
```

The members of the hostent structure are:

h_name The official name of the host.

h_aliases
A zero-terminated array of alternative names for the host.

h_addrtype
The type of address; always AF_INET at present.

h_length
The length of the address in bytes.

h_addr_list
A zero-terminated array of network addresses for the host in network byte order.

h_addr The first address in h_addr_list for backward compatibility.

RETURN VALUE
The gethostbyname() and gethostbyaddr() functions return the hostent structure or a NULL pointer if an error occurs. On error, the h_errno variable holds an error number.

ERRORS
The variable h_errno can have the following values:

HOST_NOT_FOUND
The specified host is unknown.

NO_ADDRESS or NO_DATA
The requested name is valid but does not have an IP address.

NO_RECOVERY
A non-recoverable name server error occurred.

TRY_AGAIN
A temporary error occurred on an authoritative name server. Try again later.

FILES
/etc/host. conf
resolver configuration file

/etc/hosts
host database file

SEE ALSO
resolver(3), hosts(5), hostname(7), resolv+(8), named(8)