



Royal Holloway, University of London
Information Security Group

Internet Protocol Security Flaws

MSc in Information Security



REPORT FOR IS4 (COMPUTER SECURITY)

Course Supervisor: Chris Mitchell

December 1998

Contents

TCP/IP

SECURITY AT THE PHYSICAL AND LINK LAYERS.....	4
SECURITY AT THE IP LAYER	4
NETWORK SNIFFING	5
SNIFFING AN INTERNET MESSAGE	5
MESSAGE REPLAY	6
MESSAGE ALTERATION.....	7
MESSAGE DELAY AND DENIAL	7
AUTHENTICATION ISSUES AT THE IP LAYER.....	7
ADDRESS MASQUERADING	7
ADDRESS SPOOFING	8
TUNNELLING.....	9
UNAUTHORISED ACCESS.....	9
ROUTING ATTACKS	10
SECURITY AT THE TCP/IP LAYER.....	11
PACKET FILTERING.....	11
HIJACKING	11
DENIAL OF SERVICE INCIDENTS.....	12
DENIAL-OF-SERVICE DEFINITION AND TYPES	12
CATEGORIES OF DENIAL-OF-SERVICE ATTACK.....	12
CRYPTANALYSIS OF MICROSOFT'S POINT-TO-POINT TUNNELLING PROTOCOL (PPTP).....	16
BIT FLIPPING ATTACKS	20
PASSIVE MONITORING.....	20
SPOOFING PPP NEGOTIATIONS	21
MICROSOFT'S CHALLENGE HANDSHAKE AUTHENTICATION PROTOCOL	21
CONTROL CHANNEL	22
ATTACKING RESYNCHRONISATION.....	22
PPTP CONCLUSION	23
SSL (SECURE SOCKETS LAYER) PROTOCOL.....	24
INTRODUCTION	24
1. SSL RECORD PROTOCOL.....	24
2. SSL HANDSHAKE PROTOCOL	25
3. KEYS USED IN SSL	26
STRENGTHS OF THE SSL PROTOCOL	27
DICTIONARY ATTACK	27
BRUTE FORCE ATTACK AGAINST STRONG CIPHERS	27
REPLAY ATTACK	27
MAN-IN-THE-MIDDLE ATTACK.....	28
OMISSIONS FROM THE SSL SPECIFICATION	28
CERTIFICATION MANAGEMENT.....	28
ERROR MESSAGES	28
WEAKNESSES OF THE SSL PROTOCOL	28
RENEGOTIATION OF SESSION KEYS	28
OTHER WEAKNESSES	29

HISTORICAL FAULTS FOUND IN SSL	29
SSH (SECURE SHELL)	32
SOLUTION	32
S-HTTP (SECURE-HYPER TEXT TRANSFER PROTOCOL)	32
WEAKNESSES.....	32

This document has been prepared by

Hitesh Patel
Trevor McDowall
Mark Lodge
Mattheos Milionis
Kalliopi Sotiropoulou
Daniel Elliot
Albert James
Chettha Songthaveepol
Chia-Ling Hsieh
I Hsuan Ronnitta Chang
Semia Jaweed
Jagdish Bath
Jagjeet Sondh

Thanks also due to Chris Mitchell

TCP/IP

TCP/IP is a protocol suite conceived to allow computers of different characteristics (software and hardware), to communicate with each other. An Internet is a collection of networks that all use the same protocol suite. The Internet is based on TCP/IP.

A protocol suite is normally the combination of different protocols at various layers. TCP/IP is usually considered a 4-layer system.

TCP/IP protocol suite

APPLICATION www, e-mail, etc.
TRANSPORT TCP, UDP
NETWORK IP
HOST-to-NETWORK (Physical & Link Layers) ARP, RARP, Hardware Interface.

Security At The Physical And Link Layers

Physical and link layer security is primarily concerned with access control to, and confidentiality of, the physical transmission medium. To keep the focus on Internet security rather than network security in general, this chapter mainly concentrates on the security of the other layers.

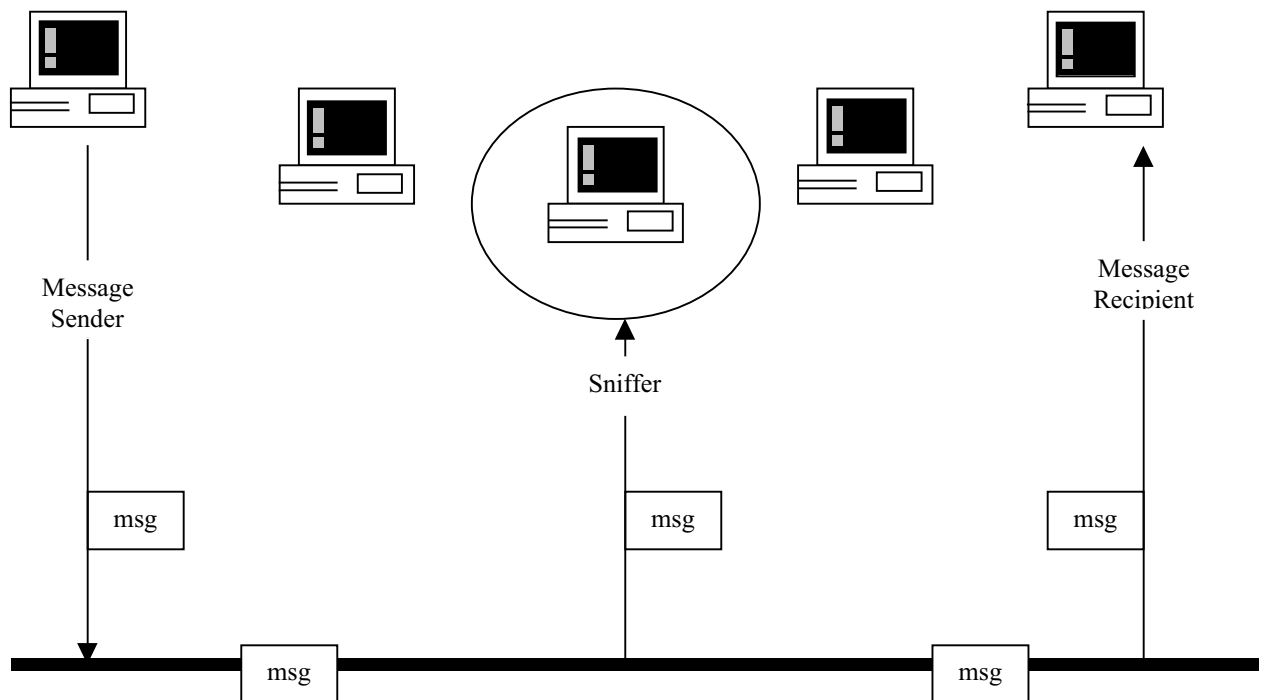
Security At The IP Layer

Security issues at the IP layer are, as you might guess, directly related to that layer's function of end-to-end datagram delivery. Whether shipping data across a tiny local network or a dozen networks spanning thousands of miles, many of the same security concerns apply. This section describes some of the IP layer's most basic weaknesses from which most of its security problems arise.

Network Sniffing

You can think of an Internet message as being like a postcard. Just as every person who handles a postcard can choose to read the message on it; any system on a network with a shared transmission medium has the potential to indiscriminately read every network datagram—even those not addressed to it. An attacker that covertly observes network traffic without disturbing it commits a *passive attack* on the network. Passive attacks are more commonly called *snooping or sniffing*. Ethernet connections, among others, are prone to this kind of attack.

Sniffing an Internet message



Many software utilities make sniffing easy. Their intended purpose is to aid in debugging application and network problems, but like any useful tool, they can be used for ill gain.

Obviously a common target for sniffers are user passwords. Once equipped to eavesdrop, it is a simple matter for an attacker to capture packets that carry responses to the recognisable login and Password prompts. On a local network with a few moderately used systems, hundreds of passwords can be collected in a single day.

Sniffing software works by placing a system's network interface into promiscuous mode. Most interfaces support this feature, but a few do not, presumably for security considerations. When in *promiscuous mode*, the interface is ready to report the contents

them. More advanced implementations analyse the various packets in an effort to diagnose problems such as faulty protocol implementations.

Operating systems like Unix and VMS require superuser or system-level privileges to access the network promiscuously. This is a desirable restriction; yet unfortunately, these systems often accept logins from remote Internet networks. An attacker who breaks into one of these systems from 1000 miles away and hacks the required privileges can snoop the distant network from the comfort of his remote location.

DOS and Macintosh computers not on a UNIX system suffer from a different problem. Inherently single-user systems, they do not accept interactive logins from the network. Command execution is not allowed from the remote machine so that remote users can not use a PC machine to sniff without inside assistance -I.e. They must somehow have physical access to the machine. Given this physical access, though, such an attacker can anonymously use it to snoop the network with impunity.

Sniffing detection usually involves utility software checking for interfaces working in promiscuous mode, or alternatively, physical checking of all the connections. Also sniffer logs can become large and a lot of file space can be used which can also point to the sniffer.

Now we know how these attacks can be detected, *but can we prevent or neutralise these attacks?* Yes, there are three ways of doing this:

- ◆ **Network Segmentation.** A network segmentation consists of a set of machines *sharing* low-level devices and wiring and *seeing* the same set of data on their network. Repeaters and passive hubs do not limit the flow of data arriving to any of its interfaces. However the use of Switches, active hubs and bridges, do limit the flow of data. Hence allowing the prevention of sniffing on untrustworthy machines.
- ◆ **Encryption.** However this solution is not universal as there is a trade off between how much information needs to be encrypted and the speed or cost of encryption. PGP and SSL are designed to guarantee secrecy, integrity and authenticity of related data.
- ◆ **S/Key and other one-time password techniques to prevent password sniffing.** Here the server presents a challenge to the connecting user (client), who, using the challenge information and the real password, either calculates or selects a new string and sends it back to the server. The string is then entered into the servers comparing algorithm and if a match is obtained the connection is allowed to continue. An alternative to one-time password systems is Kerberos. Kerberos allows workstations to authenticate themselves to services running on servers without ever sending a password in clear text over the network.

Message Replay

To achieve a message replay attack, an attacker first sniffs and records a conversation between two systems, say alpha and beta. The attack might be launched from either of these systems, or (as is often the case) from a third system, say gamma. At a later point in time, much like a tape recorder, gamma plays back some or all of the messages sent by

alpha to beta. With luck, gamma unlawfully achieves the same results that alpha earlier did.

Message Alteration

Currently no widely implemented mechanism guarantees message integrity at the IP layer. An attacker who modifies the contents of a datagram also can recalculate and update its header checksum, and the message recipient will be unable to detect it. The only real way to prevent message alteration is to use cryptographic techniques that ensure data integrity.

Message Delay And Denial

It is possible for an attacker to adversely affect the IP mechanism by employing message delay and denial tactics. The effects of delay- causing datagrams to be held or otherwise made undeliverable for an unwarranted period of time- can range from mildly annoying to destructive. Denial- causing datagrams to be discarded before final delivery- effectively blocks the communication path. An attacker can instigate these attacks in several ways:

- ◆ By gaining unauthorised control of a router or routing host, then modifying executable code or routing and screening rules used by the code. This is best prevented by applying proper authentication and access control mechanisms to the routing systems.
- ◆ By overwhelming a routing device, or one of the communicating end systems, with an inordinate amount of network traffic. This is easily detected, but obviously difficult to prevent.

Authentication Issues At The IP Layer

Authentication at the IP layer is concerned with the identity of computer systems, not computer users. The identity of a computer on the Internet is often thought of as a host name, for example, mypc.mydept.myco.com. The existence and use of host name, however, are mostly a human convenience. The true elements of identity on the Internet are IP network addresses, such as 124.16.32.211, which are mapped to host names through the Internet's *Domain Name Systems* (DNS).

Problematically, no ubiquitous authentication scheme for securing the IP layer exists. IP addresses are software configurable, and the mere possession (or fraudulent use) of one enables communication with other systems. Within a few functional bounds, one computer can claim to be another by pilfering its network address, and other systems will usually believe it. Two such techniques are address masquerading and spoofing:

Address Masquerading

Because network addresses are configured in software, it is usually as easy to select one address for a machine that will operate on a local network as it is another. If Tom and Jerry share an office, nothing but good conscience prevents Tom from unplugging Jerry's machine from the network and configuring his with Jerry's address. This might buy Tom access to some services intended for Jerry. Address masquerading occurs when an

Address Spoofing

Address spoofing was first theorised by Robert Morris and further analysed by Steven Bellovin. Unlike address masquerading, a form of attack restricted to the confines of a local network, address spoofing is possible in some circumstances across an arbitrarily large Internet. It is a sophisticated attack on the three-way TCP handshake that establishes a reliable transport connection.

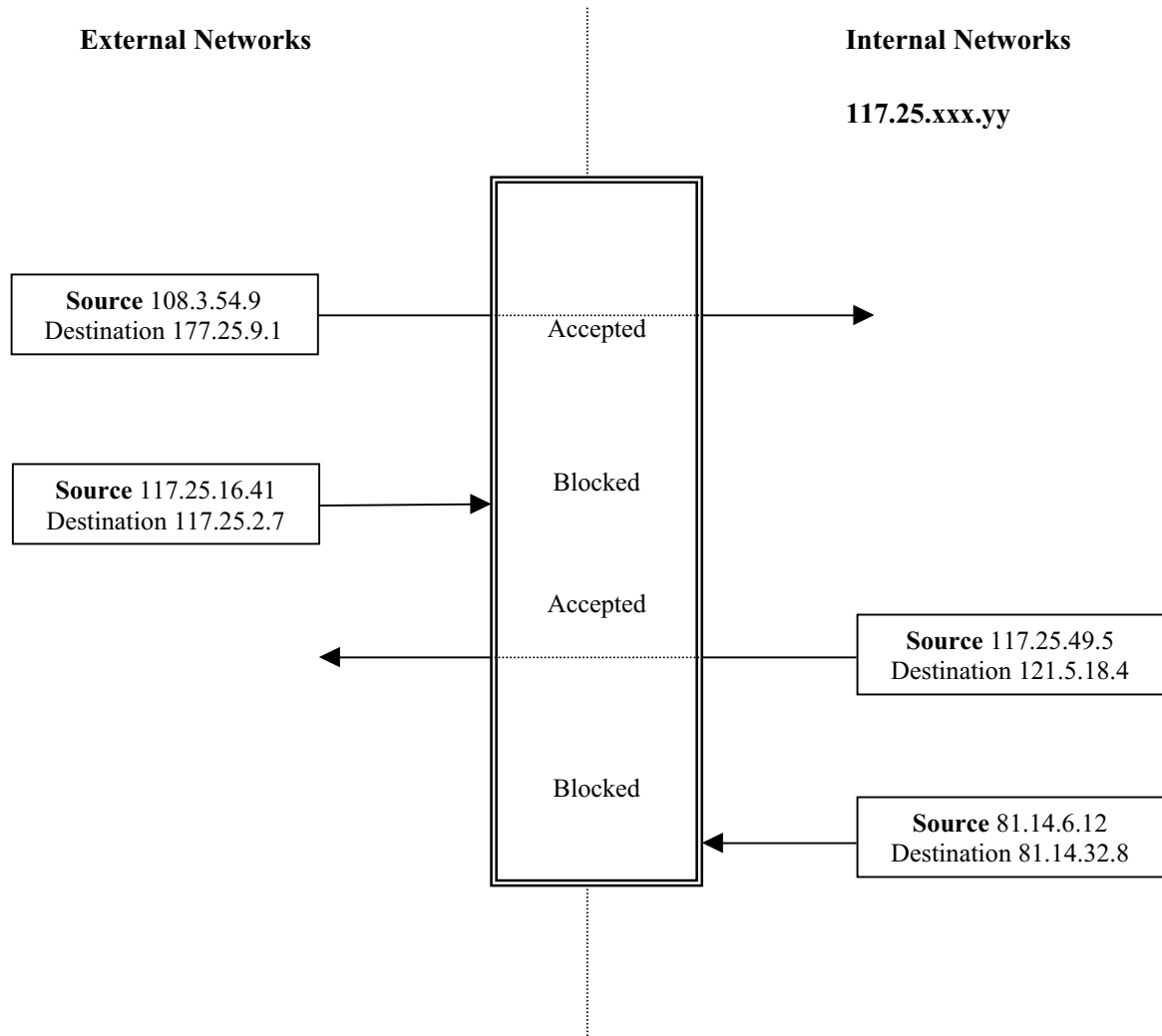
There are several practical defences against address spoofing. The first, as with address masquerading, is to avoid reliance on address-based authentication and trust mechanisms like those used by rsh. This is especially important when networks external to your own have access to your internal network.

The second approach involves the use of a screening router, a device that can intelligently filter network packets based on configurable rules. This does not prevent all possibility of spoofing, but it can prevent the following incidents:

- ◆ Inbound attacks that originate from external networks.
- ◆ Outbound attacks that originate inside of your own network.

The first form of attack is prevented by configuring the router to discard incoming datagrams with a source address belonging to the internal network. This should simply never occur. Similarly, the second is prevented by discarding outgoing datagrams with a source address from an external network. (See diagram below)

A screen router preventing address spoofing



Tunnelling

Today, it is common to tunnel one protocol inside another. This can give an attacker an opportunity to create a Trojan Horse attack by secreting their code inside yours. For example, it is possible to place an IP packet as a message part in a Domain Name Service or TELNET message. To prevent this, avoid giving an attacker an entry point: Encrypt between nodes. Tunnels can originate from a person in your organisation or in another.

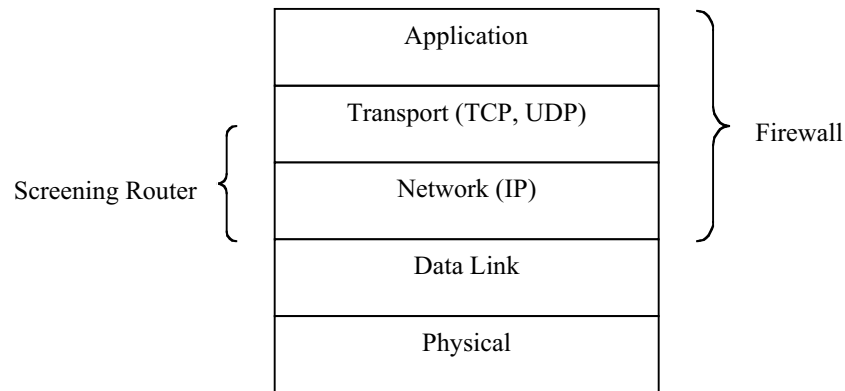
Unauthorised Access

For better or worse, depending entirely on your perspective, remote access into many Internet networks is completely unrestricted. Millions of users in one hemisphere can connect to tens of thousands of computers in the other hemisphere as easily as they can to computers in the next room.

Several ways are available to sensibly restrict Internet access into (and even out of) your network. These include deploying a screening router or a firewall, one or more systems

that bulwark internal trusted networks from external untrusted ones. From the perspective of the IP layer, both methods can perform the same packet filtering function; although as you can see in the figure below, the overall scope of these approaches differs slightly. Packet filters parse the headers of incoming and outgoing messages and apply a series of locally defined rules to determine if individual packets should be blocked or forwarded. It is worth noting that most router software has built-in filtering capabilities; consult your router's documentation for specific details.

The scope of screening routers and firewalls



Routing Attacks

First, IP supports a source routing option. Normally, a given route between two points is dynamically determined based on factors such as link availability, distance and speed between router hops, and so forth. However, a sending system can override a dynamic route by specifying a source route—that is, an explicit path through which the datagram should be sent to its recipient. (Actually two flavours of source routing exist: strict, which demands that each address in the return path be used and no other; and loose, which permits use of additional addresses between the ones specified.)

Although valid for some purposes, such as testing the routing capability of a newly established channel, source routing can be harmfully leveraged by someone who demands that one or more specific routing devices be used. For example, if an attacker attempts to connect to a distant host, he might be foiled by an intervening packet filter that blocks his path. If he detects an alternate route to the host that erroneously lacks the blocking rule, however, including it in a source route will give him the access he wants. It is generally considered good security practice to disable the source routing feature in routing software. If it is needed temporarily, enable it only for the short time it is required.

Second, most routing protocols do not use secure authentication mechanisms. It is therefore possible for an attacker to construct bogus route update messages that hosts and other routers will believe and obey. After hosts begin routing packets to an address chosen by the attacker, every datagram that traverses the route is endangered. Imminent risks include message alteration, delay, and denial. Naturally, what limited confidentiality there may have been prior to attack also is lost.

Third, an attacker can use ICMP's redirect feature to fool a host into believing that the attacker's system owns the best route to a foreign host or network. ICMP redirects are obeyed by end systems, not routers; so to succeed, the attacker's system must be on the same local network as the victimised host. Further, redirects apply to only one host, with the intent for it to establish exactly one new route. The negative effects of these attacks are similar to those resulting from attacks on the routing protocols, only on a smaller scale. Unfortunately, this makes them less detectable, unless redirects are visibly logged where a network or system administrator will notice them.

Security At The TCP/IP Layer

Because TCP segments and UDP datagrams traverse the network within IP datagrams, they are subject to some of the same security risks and problems just presented. At the least, TCP and UDP messages can be replayed or altered, and there is no guarantee of confidentiality. These problems must really be seen, however, as network-layer vulnerabilities, because they result directly from exposure to the network. Further, a scheme that prevents these exploitations at the network layer also (for the most part) does so at the transport layer. Because most processing of the TCP and UDP protocols is handled within the confines of local machines, this section concentrates there instead.

Packet Filtering

Packet filters can selectively (or globally) block access to networks and hosts located on either side of the filtering device. Packet filtering rules can apply to either the IP or TCP layers, or more usefully to both layers, per the local policy and configuration.

Hijacking

Hijacking occurs when an intruder uses ill-gotten privileges to tap into a system's software that accesses or controls the behaviour of the local TCP.

A successful hijack enables an attacker to borrow or steal an open connection (say, a telnet session) to a remote host for his own purposes. In the likely event that the genuine user has already authenticated to the remote host, any keystrokes sent by the attacker are received and processed as if typed by the user. If the hijack victim is root, for example, the attacker might issue a command to change the superuser password, or add a privileged account to the password database. A truly skillful hijack accomplishes these without divulging a single clue to the user; lesser implementations echo the attacker's keystrokes and responses from the remote host to the user, who is hopefully observant.

Because hijacking results from compromised system security, the best defense is to prevent attackers from gaining root access in the first place.

Denial of Service Incidents

The Internet Worm incident during the first week of November 1988, was the incident that resulted in the establishment of the CERT®/CC¹. It was also the first widespread denial-of-service attack on the Internet. It is of special interest to us here because it made use of TCP in order to propagate throughout the Internet. Service was denied in two ways. First, infected hosts were rendered useless because multiple copies of the worm program absorbed their processing capability. Until all copies of the worm were removed, these hosts were not available for their intended use. Second, although most hosts on the Internet were not infected by the worm, the fear of infection effectively "shut down" the Internet for several days as many sites disconnected from the network as a defensive measure.

Since the Internet Worm, there has not been another large-scale denial-of-service incident on the Internet. On the other hand, operating systems for host computers on the Internet provide few protections from denial-of-service attacks. It would, therefore, seem possible that denial-of-service incidents could become widespread on the Internet.

For more information on the Worm try

<http://www.ee.ryerson.ca:8080/~elf/hack/iworm.html>

Denial-of-service Definition and Types

The baseline security that every user needs from a computer system is availability. Hardware and software must be kept working efficiently or else they become useless. If computer hardware, software, and data are not kept available, productivity can be degraded, even if nothing has been damaged. Denial-of-service can be conceived to include both intentional and unintentional assaults on a system's availability. The most comprehensive perspective would be that regardless of the cause, if a service is supposed to be available and it is not, then service has been denied.

An attack, however, is an intentional act. A denial-of-service attack, therefore, is considered to take place only when access to a computer or network resource is intentionally blocked or degraded as a result of malicious action taken by another user. These attacks do not necessarily damage data directly, or permanently (although they could), but they intentionally compromise the availability of the resources.

Categories of Denial-of-Service Attack

- ◆ **Destruction** - If an attacker obtains access to user, host, or network files, the attacker could delete or corrupt some or all of these files. The effect could be to deny the use of these files. At the user level, an attacker could delete some or all of the account's files, rendering the account unusable. At the host level, critical system files could be deleted. On Unix systems, this could be files such as the /etc/passwd file, or files containing the system's programs. All files on the host's hard disk could

¹ The CERT®/CC, located at CMU's Software Engineering Institute (SEI), has been on the "front lines" in defence of the Internet since November 1988. The purpose of the CERT®/CC is to provide the Internet community a single organisation that can co-ordinate responses to security incidents on the Internet. CERT®/CC accomplishes this during a security incident by establishing and maintaining

also be removed, or the disk itself could be reformatted. This would make the host computer inaccessible or unusable to all users. At the network level, network files could be destroyed. The network or some of its services could then be degraded or unavailable.

- ◆ Computer viruses (self-replicating, autonomous computer code fragments), or worms (self-replicating complete programs) often contain destructive payloads which corrupt or destroy some or all of a system's files. When a virus or worm operates in this manner, it would be causing denial-of-service.
- ◆ **Process Degradation** - Instead of destroying files, denial-of-service could be accomplished through overloading processes on a host computer to such a point that the users' ability to use the resource is degraded either by reduced performance, or by the resource becoming unavailable. This can take place in two ways. First, an attacker could connect to a host across the Internet and then spawn multiple processes on the host to the point where the host could no longer support any new processes, either for an individual user, or for all the users on the target host computer. The targeted user, or users, would then not be able to run processes of their own. Programs that accomplish this are sometimes referred to as fork bombs. A second method would be to slow the host computer by spawning many processes that consume large amounts of central processing unit (CPU) time, causing a CPU overload.
- ◆ **Storage Degradation** - A similar, although distinguishable, method of attack is aimed at consuming disk storage capacity on the target host or network of hosts. Since a disk has finite capacity, if an attacker fills up a user's disk quota, or fills up the space available for all users, then the user's account or the entire host, will not be available for use until the disk full condition is changed. An attacker can create either too many files for the system, or a few files that are too large. The same is true for a network, where the files may be distributed across multiple computers.
- ◆ An example of such an attack is "mail bombardment," or "mail spam." The attacker accomplishes this attack by either flooding a user, or group of users, with numerous, perhaps thousands, of electronic mail (e-mail) messages, by flooding the user with very large messages, or by flooding the user having messages with large attachments. Any of these would quickly fill a user's Mailbox, which would then deny the user access to e-mail, and perhaps all system services. Depending on how the system is configured, this could cause the system to run out of storage space and then stop processing for all users on the host or network. The attacker could also easily forge the "From:" block in these messages, which would disguise their origin.
- ◆ **Shutdowns** - The last two categories of denial-of-service attacks shown in Figure 1 are process shutdown and system shutdown attacks. In these types of attacks, the attacker aims at halting a process, or all processing, on a host or network. If the attacker has privileged access, this could be accomplished by issuing the appropriate commands to kill a process or shutdown the system completely. The kill command in Unix is an example of a command that could be used to terminate a process.

Results		
Corruption of Information		
Disclosure of Information		
Theft of Service		
Denial-of-service		
Destruction:		Users
1 - all disk files		Hosts
2 - individual files		Networks
Process Degradation:		
3 - multiple processes		
4 - CPU overload		
5 - network application		
6 - network service		
Storage Degradation:		
7 - Disk		
8 - I-nodes		
Process Shutdown:		
9 - commands		
10 - software bug		
System Shutdown:		
11 - commands		
12 - software bug		

Figure 1 Denial-of-Service Attack Methods

Denial-of-service attacks over the Internet can be directed against three types of targets: a user, a host computer, or a network. An attacker must begin a denial-of-service attack by using tools to exploit vulnerabilities and then either obtain unauthorised access to an appropriate process or group of processes, or to use a process in an unauthorised way. The attacker then completes the attack by using some method to destroy files, degrade processes, degrade storage capability, or cause a shutdown of a process or of the system.

Unlike other attacks reported to the CERT/CC, denial-of-service incidents grew at a rate around 50% per year greater than the rate of growth of Internet hosts. This indicates that denial-of-service was becoming a greater problem for the Internet during this period, although the total number of denial-of-service incidents was small.

The largest single method used for denial-of-service attacks as recorded in CERT/CC records was the use of mail spam to degrade storage capacity (49 incidents, 32% of instances). Another large category was process degradation (40% of the instances).

The average number of sites involved in denial-of-service incidents was found to be relatively low compared to root and account level break-ins. In addition, a large number of the attackers were apparently identified, compared to the average for all incidents.

Point-to-Point Tunnelling Protocol

The Point-to-Point Tunnelling Protocol (PPTP) can be regarded as an extension of Point-to-Point Protocol (PPP). PPTP adds a new level of enhanced security and multi-protocol communications over the Internet.

PPTP specifically enables implementation of secure, multi-protocol Virtual Private Networks (VPN's) through public data networks such as the Internet. Through PPTP, it is possible for remote users to access their corporate networks and applications by dialling into the local Internet Service Provider's (ISP's) Point of Presence (POP), instead of dialling directly into the company network. PPTP connects directly to the target server by creating a virtual network for each remote client; one that the server administrator can monitor and manage like any other Remote Access port.

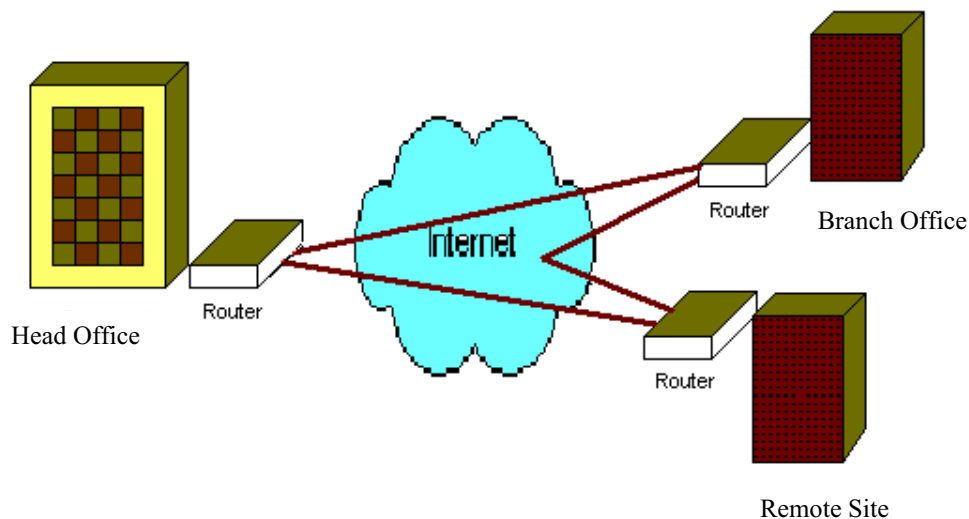


Diagram Illustrating A Virtual Private Network through the use of PPTP

PPTP solves many problems for the network administrator who needs to accommodate remote users, and wish to avoid building and maintaining a relatively costly Wide Area Network through private lines.

The Microsoft PPTP is embedded in Microsoft's Windows operating system and is used with Microsoft's Routing and Remote Access Service. It sits at the data-link layer (which maps approximately to layer two of the OSI model). It encapsulates PPP with IP packets and uses simple packet filters and the Microsoft Domain networking controls to provide access control. The 'tunnel' allows the target server to perform all of the security checks and validations, and enables administrators and clients to encrypt data, making it much safer to send information over non-secure networks.

Bruce Schneier², president of Counterpane Systems; a consulting firm specialising in cryptography and computer security, found security flaws in Microsoft's PPTP that allows attacks to sniff passwords across the network, break the encryption scheme and read confidential data, and also mount denial of service attacks against PPTP servers.

² Bruce Schneier, President, Counterpane Systems, 101 East Minnehaha Parkway, Minneapolis, MN 55419. Electronic mail: schneier@counterpane.com

Bruce Schneier came across the flaws while he was working on a project for a client, and was further assisted by expert hacker Peter Mudge, in the cryptanalysis work on Microsoft's implementation of PPTP.

They did not find any flaws in PPTP itself, but only in Microsoft's implementation of it. Such flaws causing attacks to be carried out are described in greater detail below.

Cryptanalysis of Microsoft's Point-To-Point Tunnelling Protocol (PPTP)

To clarify the discussion included later within this report, let us first define the two methods used within Windows NT and Windows 95 to *securely* store passwords. Windows NT and Windows 95 store a database of user passwords in an encrypted format. More precisely, Windows NT uses two one-way hash functions to protect passwords. The first of these functions, the *Lan-Manager* hash function, was developed by Microsoft to protect passwords within IBM's OS2 operating system, but was later implemented into Windows 3.1, Windows for Workgroups and Windows 95. Windows NT uses a slightly different, and more secure, method aptly named the *Windows NT* hash function. To enable backward compatibility, required as many Windows NT networks incorporate machines using the Windows 3.1, Windows for Workgroups and Windows 95 operating systems, both hash mechanisms are used to secure stored passwords. This enables machines using Windows 3.1, Windows for Workgroups, Windows 95 and Windows NT to operate successfully on a *virtual network*.

The *Lan Manager* hash value is calculated in the following way:

1. A user's password is converted into a 14 character string. This is achieved by either taking the first 14 characters of a password with a length greater than 14 characters, or by *padding* passwords with a length of less than 14 characters. A padded character is represented by a *null* value.
2. Convert all lowercase letters to uppercase. Non-alphabetic characters, such as numbers and symbols, are left unchanged.
3. Split the 14 byte character string into two 7-byte halves.
4. Take both 7-byte character strings separately and use each as a DES³ key. With these keys, encrypt a fixed *constant* value. This produces two 8-byte encrypted strings.
5. Finally, concatenate the two 8-byte encrypted strings to create one 16-byte string.

Because of the nature of the mechanism used to create the *Lan Manager* hash value, attacks on the stored encrypted passwords are possible. These attacks can be made, without any significant effort.

1. Dictionary attacks are possible as most users choose common words as passwords.

³ DES: Data Encryption Standard

2. Before the *Lan Manager* hash function is used to encrypt a password, all alphabetic characters are converted to uppercase. Because of this fact, text dictionaries used within a dictionary attack should be converted to uppercase prior to execution. As all passwords are not case-sensitive, no permutations of case are required within the dictionary, making the searching process trivial.
3. Exhaustive search attacks become feasible when the password mechanism is not case-sensitive.
4. No salt is used within the *Lan Manager* hash process. Because of this fact, any two users with the same password will have an identical password hash value. It is therefore possible to compile a comprehensive text dictionary that contains plain-text words with their equivalent *Lan Manager* hash value. This removes the need for dictionary-attacking software to encrypt individual dictionary strings before making comparisons against entries within the password files.
5. The two 7-byte password halves are hashed independently, no feedback encryption is incorporated into the algorithm. Therefore, this allows an attacker to approximate the length of a user's password. A password with a length of 7 characters or less will produce a constant value within bytes 8 to 14 (the second encrypted string). This is because the encryption of the fixed constant with seven null key values will always produce a determinable hash value. Therefore, it will always be possible to identify individuals who use passwords with a length of less than 7 characters. Independent hashing of each 7-byte entry also allows the strings to be attacked independently, thus reducing the time required by a machine to obtain a particular user's password.

The *Windows NT* hash value is calculated in the following way:

1. A user's password, with a length 14 characters (again, using padding to increase the length of short passwords or trimming the lengths of long passwords) is converted into Unicode. Here, the hash mechanism takes into account case-sensitive passwords. There is no explicit conversion of lowercase password characters into uppercase.
2. The password is then hashed using the MD4⁴ algorithm. This produces a single string with a length of 16 bytes.

The *Windows NT* hash mechanism is a definite improvement on the *Lan Manager* method. This is because password case-sensitivity is taken into account by the MD4 together, as one complete 14-byte string, as opposed to splitting the string into two 7-byte halves. This removes the possibility of attacking each password half independently. Hashing the string in its entirety also removes the immediate ease of detecting passwords exhibiting a length of 7 or less characters. However, within the *Windows NT* hash mechanism, there is still no provision for a salt value. Thus, two people using the same password will have the same encrypted entry within the password file. Again, this provides an attacker with the option of compiling a comprehensive text dictionary that contains plain-text words with their equivalent *Windows NT* hash value.

⁴ MD4: Message Digest Algorithm Number 4

A more serious implication of the *Windows NT* hash mechanism is discovered when one considers Microsoft's policy regarding backward compatibility. As highlighted above, many Windows NT networks also contain machines using the Windows 3.1, Windows for Workgroups or Windows 95 operating system. Because of this fact, it is not possible to use only the more secure *Windows NT* hash mechanism alone. To allow machines using a variety of versions of Windows to participate in network activity, both the *Windows NT* and the *Lan Manager* hash mechanisms must be used for all network communications⁵. This poses a very serious risk to network security as it is possible to attack the weaker *Lan Manager* hashed password excluded within a data communication packet. When successful, this extracted uppercase password can be used to attack the encrypted case-sensitive password created by the *Windows NT* hash function⁶. When complete, this system would allow an attacker to obtain any communicated password with little effort (if brute force mechanisms were employed), even if the password contained characters of both upper and lowercase.

As we have established how various Microsoft operating systems protect user's passwords, it is now possible to examine the security mechanisms used within the PPTP technique. Microsoft Point-To-Point Encryption Protocol (MPPE) provides a method of encrypting data that is transmitted across a network. MPPE assumes the existence of a key shared by both ends of a connection. The encryption of data, communicated within packets, is performed by the RC4 cipher. When a communication path and the use of MPPE has been established between parties, encrypted communication can begin. It is interesting to note that only packets whose protocol numbers are in the range 0x0021 to 0x00fa are encrypted. All other packets are passed in the clear. This occurs even if the network administrator has specifically instructed the server to operate only in encryption mode⁷.

Due to export licensing laws, when using MPPE outside the United States of America, only 40-bit encryption is possible. The key used within MPPE communication is established by each party⁸ as follows:

1. Each party takes the *Lan Manager* hash value obtained from the user's password. This value is encrypted with the SHA (Secure Hash Algorithm) to produce a deterministic 64-bit key.

⁵ It is interesting to note that networks containing only machines using the Windows NT operating system *still* contain both the *Windows NT* and *Lan Manager* hash value within their communication packets by *default*. This use of the default value poses a serious security risk for an unsuspecting network administrator.

⁶ Once the uppercase password has been extracted from the *Lan Manager* hash value, it is a trivial task to ascertain case-sensitivity. The extracted uppercase string can be converted into each permutation of case. This is a relatively small number of permutations, even for a password with a length of 14 characters. When all case permutations of the uppercase password have been produced, they can be independently tested against the *Windows NT* hash value.

⁷ RFC1700 details the type of packets that are and are not encrypted. No authentication is provided for *any* packets during communication.

⁸ Typically, the parties comprise of an individual user's host machine and the network server. Both parties must have a copy of the user's encrypted password if they are able to independently calculate the session key. Because of this, communication is possible between a user's host computer and the network server.

2. The high-order 24-bits of the 64-bit key are set to 0xD1269E.
3. The resulting 40-bit key is used to initialise the RC4 encryption algorithm in the usual manner and then to encrypt data packets. As both the server and the user's host computer each have access to the user's password, independent calculation of the same session key is possible.

When 256 packets have been communicated between parties, MPPE generates a *new* session key. This *new* is calculated in the following way:

1. The previous 40-bit session key and the user's password are hashed together by the SHA algorithm. As before, this results in the production of a deterministic 64-bit key.
2. The high-order 24-bits of the 64-bit key are again set to 0xD1269E.

When this method is used during communication, there are several fruitful attacks that can be made on captured packets. It is possible to establish the user's password, and trivial to break the encryption system, as the strength of the encryption is based solely on the strength of the user's password. When used in the United States of America, the *Windows NT* hash function is used within the encryption process, but as mentioned above, this is not incorporated within exported software packages⁹.

In reality, most passwords actually provide a great deal less than 40-bits of encryption security. The use of the *Lan Manager* hash function also generates vulnerabilities because of the maximum password size, a limited alphabet and the use of only uppercase characters. The use of a 40-bit session key, combined with the fact that no salt value is used when encrypting user passwords, makes the encrypted communicated packets susceptible to attack. It is possible for an attacker to compute a dictionary containing ciphertext packet headers and their plaintext equivalents.

The same 40-bit RC4 key is generated every time the same user initialises the PPTP protocol. Because of this fact, once a user's password has been compromised, and previous session keys calculated by an attacker, it is possible to decipher the encrypted packets without firstly establishing the keys used. A set of keys, alternating after 256 packets have been communicated, can be recorded and used each time the attacker wishes to intercept a particular user's communication. This interpretation would be possible until the user changed his password.

⁹ However, it is important to note that the use of a 128-bit key and the *Windows NT* hash function (possible only in the United States of America) does not significantly improve cryptographic security. As both the *Lan Manager* and *Windows NT* hash functions are included within communication (for backwards compatibility), it is still possible to identify the session keys, and the user's password by firstly attacking the weak *Lan Manager* hash value. A successful attack on this value produces an uppercase password. From this, the *Windows NT* hash value can then be attacked. Once the user's password has been compromised, it is then possible to deduce all session keys used within communication.

As RC4 is an output-feedback mode stream cipher, it is trivial to break the encryption from the ciphertext obtained from two separate sessions. When using stream ciphers, the same key should *never* be used in both communication directions.

In conclusion, Microsoft's PPTP and MPPE protocols use weak encryption methods based upon the security of individual user's passwords. However, even if users choose strong and secure passwords, the nature of the implementation means that it is trivial to attack password files. If one considers an entry within a UNIX password file, containing 8 characters (upper or lower case letters, numbers or symbols), the potential to perform a dictionary or a brute force attack exists, but is unlikely to succeed. However, the password storage mechanism used within the Windows environment results in inadequately encrypted network traffic. The use of the *Lan Manager* hash function is a major source of concern. This is because of the conversion of all alphabetic characters to uppercase letters. This process severely reduces the *variety* of passwords available to a user and therefore makes the system more open to attack. The encryption of passwords using a cipher operating in *Electronic Codebook Mode* (ECB) also increases vulnerability.

In the words of Bruce Schneier, an independent security consultant who thoroughly tested the security of Microsoft's PPTP and MPPE protocols:

"It's kindergarten cryptography. These are dumb mistakes"

Bit Flipping Attacks

This type of attack can be accomplished without knowledge of the encryption key or the client's password. The attack is made possible on Microsoft PPTP servers due to the fact that there is no authentication mechanism provided in the Microsoft Point-to-Point Encryption (MPPE) protocol. RC4 is used in the encryption process, which is an output-feedback mode stream cipher but does not provide any authentication of the actual ciphertext stream itself. As a result an attacker can undetectably flip bits in the ciphertext. This attack can be very appealing if the underlying protocol is sensitive to particular bit flipping which as a result can enable or disable features and could even reset parameters.

Passive Monitoring

Passive monitoring is basically when an unauthorised user is observing traffic on a network from a particular point. An enormous amount of information can be gathered by just watching PPTP sessions traverse across the Internet. This sort of information is invaluable for traffic analysis where the information may be analysed to gain knowledge of inside network workings and should be protected. But the server publicly announces information such as the maximum number of channels that it has available. This information can be used to assess the approximate size of the PPTP server, and to monitor its load.

By simply setting up a standard sniffer and eavesdropping on public communications, information such as the Client Machine IP address and information leading to the retrieval of the users password hash can be easily achieved from Microsoft PPTP servers. If communications are encrypted and the user assumes some level of confidentiality, then such information should not be so easily obtainable.

There is however no easy way for Microsoft PPTP to encrypt this information, since the leaks come from outside the channel that MPPE (Microsoft Point-to-Point Encryption) controls. In some cases, these packets are the configuration and setup for the stream cipher inside of MPPE, and must be transmitted before encryption can begin. The only solution is to encrypt the control channel, or severely reduce the information being sent over it.

Spoofting PPP Negotiations

The Point-to-Point (PPP) negotiation packets occur before and after the encryption can be applied. Since the method for resynchronisation of keys is done via PPP CCP packets, these communications can never be encrypted in the same sleeve. Along with this is the fact that there is no real authentication of the packets. This configuration stage is thus entirely open to attack.

Spoofting the configuration packet containing the DNS server could be used to force all name resolution to happen through a compromised name server.

Similarly, spoofting the configuration packet containing the internal IP address could be used to circumvent stateful packet filtering firewalls by forcing the client to connect to external machines from inside the private network.

Microsoft's Challenge Handshake Authentication Protocol

PP has in it several methods for handling authentication. One of these is the Challenge Handshake Authentication Protocol (CHAP). Microsoft's PPP CHAP implementation (MS CHAP) is almost identical to the authentication on its Windows-based networks.

MS-CHAP works as follows;

Client requests a login challenge.

Server sends back an eight-byte random challenge.

The client calculates the LAN manager hash, and adds five nulls to create a 21 byte string, and partitions the string into three seven byte keys. Each key is used to encrypt the challenge, resulting in a 24 byte encrypted value. This is returned to the Server as a response. The client does the same with the Windows NT hash.

Server looks up the hash in its database, encrypts the challenge with the hash, and compares it with the encrypted hashes it received. If they match, the authentication completes.

The server could make the comparison on the Windows NT hash or the LAN manager hash; the results would be the same. Which hash the server uses depends on a particular flag in the packet. If the flag bit is set, the server tests against the Windows NT hash; if the flag bit is not set, the server tests against the LAN manager hash.

On the surface, the challenge/response protocol is standard; the use of a random login challenge makes precomputed dictionary attacks impossible against MS-CHAP as they

are against the file of stored password hashes. Still, because both the LAN Manager and Windows NT hashes are transmitted even in a Windows NT only environment, it is possible to attack the weaker LAN Manager hash in every case. And because the client's reply is divided into thirds, and each third is encrypted independently, it is possible to attack the MS-CHAP protocol itself.

The last eight bytes of the LAN Manager hash is a constant if the password is seven characters or less. This is true despite the random challenge. Therefore, the last eight bytes of the Client's reply will be the challenge encrypted with that constant. It is easy to test whether a given password is seven characters or less. After an attacker finds the LAN Manager hash, he can use that information to recover the Windows NT hash.

Additionally, in this protocol only the Client is authenticated. So basically, an attacker who hijacks a connection can trivially masquerade as the server. If encryption is enabled, the attacker will not be able to send or receive messages (unless he breaks the encryption), but by reusing an old challenge value he can obtain two sessions of ciphertext encrypted with the same key.

Control Channel

Microsoft's actual implementation of the Control Channel is seriously flawed. It was quickly found through the analysis that it is trivial to make a Windows NT machine running a PPTP server crash with kernel panics of varying types, sometimes referred to as the dreaded Blue Screen of Death (BSOD). It in fact became very difficult for the experts to test the control channel without crashing the PPTP server. Most of the attacks that were actually attempted, (such as sending invalid values in the PPTP Control Packet headers) crashed the server before the attacks could complete.

Packets can be sent to the PPTP server from outside a firewall, without any authentication. This, of course, assumes there is no firewall configuration that only allows PPTP datagrams to the PPTP server from particular IP addresses or networks. However, if the users have the ability to access the PPTP server from anywhere in the world, then an attacker can send these queries in from anywhere in the world too.

Attacking Resynchronisation

If a packet is dropped in transit or arrives with an unexpected coherency count in the MPPE header, a resynchronization of keys takes place. The end that received the inconsistent packet sends a message to the sender requesting resynchronization. When the sender end receives this request he re-initializes the RC4 tables and sets the "flushed" bit in the MPPE header. When a system sees the flushed bit set in a packet, it re-initializes its RC4 tables and sets the coherency count to match the one it just received.

This creates the problem whereby an attacker can either spoof resynchronization requests or forge MPPE packets with incorrect coherency counts. If this is done continuously just prior to the 256th packet exchange, where the session key would normally be updated, an attacker can succeed in forcing the communications channel to never re-key.

This can be used to recover encrypted plaintext. All an attacker needs to do is to force resynchronization. A simple XOR of the original stream and the resynchronized stream results in a XOR of the two plaintexts.

PPTP Conclusion

To conclude, we would like to stress that the weakness is not with the PPTP specification, but rather with Microsoft's implementation of NT and Windows 95 support.

Bruce Schneier states that "a lot of people are creating their virtual private networks using NT which makes the flaw that much more serious".

In a market study by Infonetics Research, PPTP was found to be the most popular VPN protocol currently in use. This is undoubtedly due to the 'freeware' distribution of their PPTP client and server functionality, but speaks, as well, to its general applicability and usefulness.

Microsoft promises to fix the problem as soon as possible.

However, Schneier feels that despite the stress on getting fixes out as soon as possible, many times such patches just make more problems for system administrators. He states that, "Last time they released a fix, it broke so many other parts of Windows NT, that Microsoft had to pull it off the Web site three weeks later.

The product manager for Windows NT security claims that part of the problem is already fixed, but also states the following, in a weak attempt to put the flaws in perspective, "The amount of security an organization enforces depends on its needs. The CIA spends billions of dollars on security - but our customers do not need that level of security."

This statement can be interpreted as meaning that Microsoft is not taking this implementation problem as seriously as it should be taken.

Some consultants have gone so far as to recommend avoiding PPTP entirely, and opt for the emerging Ipsec¹⁰ instead.

¹⁰ Internet Protocol Security - a security enhancement for IP networks.

SSL (Secure Sockets Layer) Protocol

Introduction

SSL, Secure Sockets Layer, is a protocol designed and implemented by Netscape Communications. Netscape claims it is designed to work, as the name implies, at the socket layer, to protect any higher level protocol built on sockets, such as telnet, ftp, or HTTP. As such, it is ignorant of the details of higher level protocols, and what is being transported. A free reference version of SSL, SSLRef, is available from Netscape. Many of the functions provided by SSL are part of the IPv6.

SSL provides for encryption of a session, authentication of a server, and optionally a client, and message authentication. The SSL Handshake Protocol and the application protocol both operate on top of the SSL Record Protocol, a simple means of encapsulating authentication information. SSL-Record Layer works on TCP or some other reliable transport mechanism. Session establishment takes from 5 to 8 messages, depending on options used. SSL relies on the existence of a key certification mechanism for the authentication of a server. SSL does not provide for renegotiation of keys within a session. (This is not a problem in HTTP, but might be with other protocols.) A multitude of ciphers and secure hashes are supported, including some explicitly weakened to comply with export restrictions.

1. SSL Record Protocol

All SSL protocol messages move in records of up to 32,767 bytes. Each message has a header of either 2 or 3 bytes. The headers include a security escape function, a flag to indicate the existence of padding, and the length of the message (and possibly the padding.) A two byte header has no padding, a three byte header includes some padding.

The meaning of the header bits is as follows:

```

 1 2 3 4 5 6 7 8
+-----+-----+-----+
|# S Length      | Length      | Padding length |
+-----+-----+-----+
```

is the number of bytes in the header

0 indicates a 3 byte header, max length 32,767 bytes.

1 indicates a 2 byte header, max length 16,383 bytes

S indicates the presence of a security escape, although none are currently implemented. (Several suggestions for security escapes are in the weaknesses section.)

There is no version information within the SSL record header, although it is available in the handshake.

Within a record, there are three components: MAC-DATA, Actual-data, and Padding-data. MAC is the Message Authentication Code, Actual-data is the actual data being sent, and Padding-data is padding.

The MAC-DATA is a hash of a key, the data, padding, and a sequence number. The hash is chosen based on the cipher-choice. The key used is the (sender)-write-key, which is the same key as the (receiver)-read-key. When cipher-choice is not defined, there is no mac-data or padding-data.

Padding is used to ensure that the data is a multiple of the block size when a block cipher is used. Padding data is always discarded after the MAC has been calculated.

Sequence numbers are unsigned 32 bit integers incremented with each message sent. Sequence numbers wrap to zero after 0xFFFFFFFF. Each dialog direction has its own sequence number.

Failure to authenticate, decrypt, or otherwise get correct answers in a cryptographic operation results in I/O errors, and a close of connection.

2. SSL Handshake Protocol

A handshake occurs when a machine attempts to use a SSL connection. There are three basic types of handshaking. The first is when no session exists "recently". Recently is not explicitly defined, but is suggested to be under 100 seconds (SSL, C.8). The second is when a set of session identifiers still exists and the third is when client authentication is desired.

For the initial connection, when a client wishes to establish a secure connection, it sends a CLIENT-HELLO message, including a challenge, along with information on the cryptographic systems it is willing or able to support. The server responds with a SERVER-HELLO message, which is connection id, its key certificate, and information about the cryptosystems it supports. The client is responsible for choosing a cryptosystem it shares with the server.

The client then verifies the server's public key, and responds with a CLIENT-MASTER-KEY message, which is a randomly generated master key, encrypted or partially encrypted with the servers public key. The client then sends a CLIENT-FINISHED message. This includes the connection-id, encrypted with the client-write-key. (All these keys are explained separately, in the next section.) The server then sends a SERVER-VERIFY, verifying its identity by responding with the challenge, encrypted with the server write key. The server got its server-write-key sent to it by the client, encrypted with the server's public key. The server thus must have the appropriate private key to decrypt the CLIENT-MASTER-KEY message, thus obtaining the master-key, from which it can produce the server-write-key.

When a session identifier is found by the client, a slightly different initialization takes place. It is initially distinguishable by the CLIENT-HELLO message including a session-id. The SERVER-HELLO response includes a set "session_id_hit" bit if the session-id is found. The client then sends an encrypted connection-id for this session in the CLIENT-FINISH message. The server responds with a SERVER-VERIFY and a SERVER-FINISH message, which are close to identical to those used in the "No session identifier case." The only difference is that the SERVER-FINISH message contains a session-id instead of new-session-id.

If client authentication is in use, then the server must at some point, send a REQUEST-CERTIFICATE message, which contains a challenge (called challenge') and the means of authentication desired. The client responds with a CLIENT-CERTIFICATE message, which includes the client certificate's type, the certificate itself, and a bunch of response data. The server then sends a SERVER-FINISH message.

To summarize those messages, they are:

Assuming no session-identifier

client-hello	C -> S:	challenge, cipher_specs
server-hello	S -> C:	connection-id, server_certificate, cipher_specs
client-master-key	C -> S:	{master_key} server_public_key
client-finish	C -> S:	{connection-id} client_write_key
server-verify	S -> C:	{challenge} server_write_key
server-finish	S -> C:	{new_session_id} server_write_key

Assuming a session-identifier was found by both client & server

client-hello	C -> S:	challenge, session_id, cipher_specs
server-hello	S -> C:	connection-id, session_id_hit
client-finish	C -> S:	{connection-id} client_write_key
server-verify	S -> C:	{challenge} server_write_key
server-finish	S -> C:	{session_id} server_write_key

Assuming a session-identifier was used and client authentication is used

client-hello	C -> S:	challenge, session_id, cipher_specs
server-hello	S -> C:	connection-id, session_id_hit
client-finish	C -> S:	{connection-id} client_write_key
server-verify	S -> C:	{challenge} server_write_key
request-certificate	S -> C:	{auth_type, challenge'} server_write_key
client-certificate	C -> S:	{cert_type, client_cert, response_data} client_write_key
server-finish	S -> C:	{session_id} server_write_key

In this last exchange, the response_data is a function of the auth_type. (These tables are taken verbatim from the SSL Specification, Section numbers as indicated.)

There are some errors defined for the handshake. They are NO-CIPHER-ERROR, NO-CERTIFICATE-ERROR, BAD-CERTIFICATE-ERROR, and UNSUPPORTED_CERTIFICATE_TYPE. NO-CIPHER occurs when the client and server can't agree on a cipher. (It is not clear why SSL doesn't require support for some simple cipher, such as RC4-40.) The other errors are recoverable if the client is the one with the bad certificate.

3. Keys used in SSL

There are a number of keys used over the course of a conversation. There is the server's public key, a master key, a client-read-key and a client-write-key. (The standard uses the

term server-write-key as another name for client-read-key, and server-read-key as another name for client-write-key.)

Client-write-key and client-read-key are derived via a secure hash from the master key, an ordinal character, the challenge, and connection-id. Of this input, only the master key is sent encrypted (with the server's public key.) The master key is reused across sessions, while the read- & write- keys are generated anew for each session.

Two or more sets of key material are generated, based on how many bits of key are needed. Hash tends to be MD5, but is not required to be so by the spec. It is MD5 for all cipher types currently defined.

key-material-N = hash (master key, "N", challenge, connection-id) Keys are taken from the output of these hashes. Because of the design of secure hashes, changing 'N' will result in the output of these hashes being very different.

Strengths of The SSL Protocol

Dictionary Attack

This attack works in situations where some plaintext is known to make up part of the original message. In the case of a protocol, the very fact that a rigid structure of data exists is often enough information. Similarly if the application data being transferred has a few often used commands, in the case of HTTP the "get" command for instance. The attack works by taking the known plaintext and precomputing the encrypted form of the plaintext using every possible key. Given an encrypted message, a search is performed looking for occurrences of the precomputed ciphertexts in the message. When one is found it is then known that the key which was used to give that ciphertext is the key which was used to encrypt the whole message.

SSL is protected from this attack by having very large key spaces for all its ciphers. Even the export ciphers support 128 bit keys (albeit 88 bit transmitted in the clear) and this makes the dictionary prohibitively large. For the export ciphers a dictionary could be produced of size 2^{40} entries, but this would have to be done for each session and this in effect makes it identical to a brute force attack (discussed later on Weaknesses of The SSL Protocol).

Brute Force Attack Against Strong Ciphers

See the section on Key Length for a description of the brute force attack and times required for successful attacks on different key lengths. A brute force attack against the ciphers with 128 bits or more is completely impractical in the foreseeable future. The only non-export cipher which would be susceptible to this attack is the standard DES 56 bit cipher - it is recommended that this cipher not be used if at all possible.

Replay Attack

A replay attack is one where a third party records an exchange of messages between a client and server and attempts to rerun the client messages at the server at a later date. SSL foils this possibility by introducing a nonce number, this is in the form of the connection-id which the server randomly generates and sends to the client. Since the nonce differs for each connection, no two connections are likely to have the same nonce and thus the old set of client messages do not satisfy the server. SSL nonces are 128 bits

Man-In-The-Middle Attack

The Man-In-The-Middle attack occurs when an adversary is able to intercept client messages read them and pass them on to the server and vice versa. This attack is prevented by SSL forcing the server to use its private key to decrypt the master key if it is to continue to handshake with the client. To be able to do this requires that it has a valid certificate which the client can verify. An adversary would have to fake a certificate and break the certification authorities key in order to compromise this chain; a task which is more difficult than doing a brute force attack on the cipher itself.

Omissions from The SSL Specification

Certification Management

The protocol states that certification systems are out of the scope of the document. However any security related protocol has to deal fully with all issues of negotiation and authentication. There are several certification standards already available, notably the PEM hierarchical system and the PGP web-of-trust system. It is recommended that a hybrid system be used, to cater for all communications requirements.

Error Messages

Two error messages have been omitted from the specification document. The first, an unsupported-authentication-type-error message, is a mistake which would prevent the protocol using different methods of authentication of a client.

The second is not essential but would let an implementation cater for all eventualities. An unexpected-message-error would allow an implementation to close the connection cleanly if an implementation sent an out-of-order message.

Weaknesses of The SSL Protocol

Brute Force Attack Against Weak Ciphers

The most obvious weakness of the protocol is the susceptibility of the ciphers which use small keys to brute force attack, in particular RC4-40¹¹, RC2-40 and to some extent DES-56. As has been discussed, this was a weakness which was forced upon Netscape by US export law. However it remains a serious flaw and has been exploited several times already. To the best of my knowledge successful brute force attacks against RC4-40 have been carried out at least five times since the protocol's release in February 1995.

Renegotiation of Session Keys

In the present situation, once a connection is established, the same session key (master key) is used throughout. If SSL were layered underneath a long running connection, such

¹¹ The use of RC4 is troublesome, albeit understandable. RC4 is a newly published cipher, and although it was designed by the very competent Ron Rivest, it has not been subjected to the same kind of intense professional scrutiny that DES and IDEA have undergone. The decision to use RC4-40 is driven by the fact that it gets automatic export approval from the State Department.

as a Telnet application, then this failure to alter session keys would become a serious security hole. One of the best methods of increasing the security of a system such as this is to force renegotiation of the session key at regular intervals, thus multiplying the difficulty and cost of a brute force attack by the number times the session key is changed. This could actually be used to increase the security of the export ciphers. If an adversary were faced with the task of having to decrypt 100,000 records all encrypted using a different, albeit weak, key rather than the same key, then the task becomes 100,000 times more difficult.

Other weaknesses

SSL, being a low level protocol, does little to protect you once your host is compromised. Also, once a key in a certificate is compromised, it can remain compromised, as there is no mechanism in place for consulting the root of a CA to confirm the key you are using has not been revoked. The keys however do include expiry dates. Climbing to the root CA is not a commonplace step, but a mechanism should be available to do so, for high value transactions. Out of band key repudiation is also desirable. Today, trusted key certifiers are compiled into the Netscape binary.

There are also a number of areas in the design of SSL as it stands that could become exploitable problems. None suggest an immediate means of attack, but they are things which could be modified for possible added safety.

Handshaking protocol: The challenge data, sent in the CLIENT-HELLO message, could in some types of handshakes, be sent later, encrypted. The data used in generating the session keys could include more data not sent in the clear. The means of generating the master key should be better specified, probably with reference to RFC-1750.

Record protocol: Bad MAC-data should not terminate a connection, it should cause a repeat-request message. There are few attacks that will get anything from having the same data resent, and closing the connection on a bad message opens avenues for denial of service attacks. Sequence numbers should be randomly initialized. There are quite a few non-obvious attacks on sequence numbers (in IP, and NFS); it can't hurt to start with a non-predictable number.

There should be a way for one or both sides to demand renegotiation of keys. Perhaps this could be implemented as a security escape. This is not needed for HTTP connection security, since the connections are very short lived, but if SSL is used, as the authors suggest, for telnet or FTP, the sessions could last substantially longer.

Historical Faults Found in SSL

Although SSL is a well defined protocol there have been some well known weaknesses discovered and these are well documented in the literature.

The first was discovered by two PhD students in the University of California, Berkeley ([Ian Goldberg](#) and [David Wagner](#)). It had to do with a weakness in the generation of the 48 bytes random number that is used for the generation of the client-master-key. The whole security of the protocol is based on the randomness of these 48 bytes. These random bytes are known, only to the client (because it produced them) and the server (because they were encrypted under its private key and then they were sent to it).

The security of SSL, like that of any other cryptographic protocol, depends crucially on the unpredictability of this secret key. If an attacker can predict the key's value or even narrow down the number of keys that must be tried, the protocol can be broken with much less effort than if truly random keys had been used. Therefore, it is vital that the secret keys be generated from an unpredictable random-number source. In one early implementation of the SSL in the Netscape browser that was not the case. The method Netscape used to seed its random number generator (RNG) depended only on the time of the day, the process ID, and the parent process ID. Thus, an adversary who could predict these three values could predict the seed and thus could compute the key.

When a connection is first established, a challenge value is calculated and sent unencrypted from the Netscape client to the secure server. This allows an attacker to learn that value, which will be useful later.

Netscape's UNIX browsers were more difficult to attack than its browsers for other platforms, since the seeding process used in the UNIX browsers utilized more-random quantities than did the process used in the browsers for other platforms. An attack on the UNIX browsers should be something like this.

An attacker who has an account on the UNIX machine running the Netscape browser could easily discover the pid and ppid values used in RNG using the ps command (a utility that lists the process IDs of all processes on the system).

All that remained was to guess the time of day. Most popular Ethernet sniffing tools (including tcpdump) record the precise time they see each packet. Using the output from such a program, the attacker could guess the time of day on the system running the Netscape browser to within a second. It was probably possible to improve this guess significantly. This recovers the seconds variable used in the seeding process. (There may be clock skew between the attacked machine and the machine running the packet sniffer, but this is easy to detect and compensate for.)

Of the variables used to generate the seed (seconds, microseconds, pid, ppid), so far are known the values of seconds, pid, and ppid; only the value of the microseconds variable remains unknown. However, there are only one million possible values for it, resulting in only one million possible choices for the seed. The attacker could use the defined algorithm to generate the challenge and secret_key variables for each possible seed. Comparing the computed challenge values to the one he had intercepted he could reveal the correct value of the secret key. Testing all one million possibilities takes about 25 seconds on an HP 712/80.

Even if the attacker did not have an account on the attacked UNIX machine, which means the pid and ppid quantities were no longer known, he could predict these quantities, or even use some other tricks to recover them.

The second well known weakness had also to do with a design feature of the PKCS #1 (RSA encryption standard) sometimes used in SSL. So the problem here is an implementation error, not a fault of the protocol itself. The PKSCS #1 encoding format for RSA encryption is intended primarily to provide confidentiality, typically for distribution of symmetric encryption keys. In particular it is not plaintext aware which means that it is feasible to construct a valid ciphertext without knowing the actual corresponding plaintext Integrity is assured by other means than PKSC#1 encryption. However PKCS #1 gives no guidance as to what these "other means" might be.

Some recipient servers, after performing the PKCS#1 decryption operation, output an error message if they find out that the result does not have the expected form. An

opponent can then determine from this behaviour some information about the encryption of the arbitrary ciphertext. S/he can determine an entire byte (and probably more) if the server doesn't output a message indicating that the encryption has the wrong form. That is because the opponent knows that the plaintext starts with bytes 00 02 (this fact has to do with the encoding method that is used). Exploiting this, the opponent can find a set of valid ciphertext and with this determine the key.

In this way the recipient server enables the opponent to compute the plaintext of a selected ciphertext with an attack using chosen ciphertext. With the present PKCS #1 encoding method, roughly one in 2^{16} to 2^{18} randomly chosen ciphertext will be valid. Against an interactive key establishment protocol such as SSL, this attack is particularly effective since the recipient is willing to process many messages, and may reveal the success or failure of each operation. Moreover, because SSL may not require client authentication, the opponent can easily remain anonymous.

There are several countermeasures for this attack such as good key management (if the servers' key pair is changed frequently then old ciphertext is not endangered). A server can reduce the probability of invalid ciphertext being mistaken for valid ciphertext if it checks the format of the message after decryption (Eg. Has it a valid SSL version number? etc). Also the same error message should be sent for every possible error (whether caused by transmission errors or decryption specific errors) in the same amount of time since time delay info could be just as helpful to the attacker as an error message. Other counter measures exist and they are easy to implement because they don't change the protocol.

Other Protocols

SSH (Secure Shell)

SSH stands for Secure Shell and is a secure login program. It has changed remote management of network hosts over the Internet and can be used to:-

- Log in to another computer over a network,
- Execute commands on a remote machine,
- Move files from one machine to another.

It is a very powerful application that uses strong cryptography for protecting all transmitted confidential data, including passwords, binary files, and administrative commands and is easy to use.

Although it is much more secure to use than telnet, rlogin and rsh, there is a security flaw in this protocol that can be exploited by an attacker with access to the encrypted SSH stream. He may insert encrypted blocks in the stream that will decrypt to arbitrary commands to be executed on the SSH server. This is not an easy attack to perform. It requires known plaintext and extensive knowledge of TCP/IP networks and the SSH protocol. The attacker has to perform an active network attack first in order to exploit the bug.

SSH servers for Unix up to version 1.3.4 are affected. SSH clients for Unix up to version 1.3.4, for Windows up to version 1.1 and for Macintosh up to version 1.0.1 are affected but the implications are less severe as the clients are not able to run any commands. The attacker will not get access to the keys being used in encryption or the information transmitted between SSH client and server.

Solution

SSH2 is a new security protocol that will replace the former SSH protocol. SSH2 is based on SSH, although almost totally rewritten and so does not contain the above protocol flaw. SSH2 is intended as a complete replacement for ftp, telnet, rlogin, rsh, rcp, and rdist.

S-HTTP (Secure-Hyper Text Transfer Protocol)

S-HTTP was designed to secure HTTP connections. It provides many mechanisms including confidentiality, authentication and integrity. The system was designed so that it was not tied to any particular cryptographic format, system or key infrastructure.

The general nature of S-HTTP makes it difficult to assess exactly what the threats are, but are similar to those against SSL.

Weaknesses

There is a problem ensuring that keys are transferred properly. For example an improper transfer would be a scheme that sends Key B as $E_a(B)$. That is to say, key B which replaces key A can not be sent using key A to encrypt it. If an attacker has broken key A, then he will have key B, and the chance of how is a waste of time (with respect to that

attacker.) The problem is that S-HTTP is very flexible and may offer programmers enough rope to hang themselves. A programmer who is not familiar with security and cryptographic issues could think that using S-HTTP would protect him and totally fail to provide any cryptographic protections for his information. The likelihood of this happening may be open to question, but the problem is worth considering.

References

SSL References

Overview of SSL, Adam Shostack, May 1995

<http://www.homeport.org/~adam/ssl.html>

Randomness and Netscape Browser, Jan 1996, ([Ian Goldberg](#) and [David Wagner](#))

http://www.ddj.com/ddj/1996/1996_01/wagner.htm

PKCS #1 weakness, Daniel Bleichenbacher, Burt Kaliski, Jessica Staddon

<http://www.rsa.com/rsa/developers/#RESOURCES>

SSL, Internet Commerce, Electronic Payment [Hagenauer Andreas, e9325125, E881](#)

<http://stud1.tuwien.ac.at/~e9325125/seminar/ssl.html>