E. Rescorla
RTFM, Inc.
B. Korver
Xythos Software

INTERNET-DRAFT
<draft-rescorla-sec-cons-05.txt> (April 2002 (Expires October 2002)

Guidelines for Writing RFC Text on Security Considerations

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

1. Introduction

All RFCs are required by [RFC 2223] to contain a Security Considerations section. The purpose of this is both to encourage document authors to consider security in their designs and to inform the reader of relevant security issues. This memo is intended to provide guidance to RFC authors in service of both ends.

This document is structured in three parts. The first is a combination security tutorial and definition of common terms; the second is a series of guidelines for writing Security Considerations; the third is a series of examples.

2. The Goals of Security

Most people speak of security as if it were a single monolithic property of a protocol or system, but upon reflection that's very clearly not true. Rather, security is a series of related but somewhat independent properties. Not all of these properties are required for every application.

We can loosely divide security goals into those related to protecting communications (COMMUNICATION SECURITY, also known as COMSEC) and those relating to protecting systems (ADMINISTRATIVE SECURITY or SYSTEM SECURITY). Since communications are carried out by systems and access to systems is through communications channels, these goals obviously interlock, but they can also be independently provided.

2.1. Communication Security

Different authors partition the goals of communication security differently. The partitioning we've found most useful is to divide them into three major categories: CONFIDENTIALITY, DATA INTEGRITY and PEER ENTITY AUTHENTICATION.

2.1.1. Confidentiality

When most people think of security, they think of CONFIDENTIALITY. Confidentiality means that your data is kept secret from unintended listeners. Usually, these listeners are simply eavesdroppers. When an adversary taps your phone, that poses a risk to your confidentiality.

Obviously, if you have secrets, you're concerned that no-one else knows them and so at minimum you want confidentiality. When you see spies in the movies go into the bathroom and turn on all the water to foil bugging, the property they're looking for is confidentiality.

2.1.2. Data Integrity

The second primary goal is DATA INTEGRITY. The basic idea here is that we want to be sure that the data we receive is the one that the sender sent. In paper-based systems, some data integrity comes automatically. When you receive a letter written in pen you can be fairly certain that no words have been removed by an attacker because pen marks are difficult to remove from paper. However, an attacker could have easily added some marks to the paper and completely changed the meaning of the message. Similarly, it's easy to shorten the page to truncate the message.

On the other hand, in the electronic world, since all bits look alike, it's trivial to tamper with messages in transit. You simply remove the message from the wire, copy out the parts you like, add whatever data you want, and generate a new message of your choosing, and the recipient is no wiser. This is the moral equivalent of the attacker taking a letter you wrote, buying some new paper and recopying the message, changing it as he does it. It's just a lot easier to do electronically since all bits look alike.

2.1.3. Peer Entity authentication

The third property we're concerned with is PEER ENTITY AUTHENTICATION. What we mean by this is that we know that one of the endpoints in the communication is the one we intended. Without peer entity authentication, it's very difficult to provide either confidentiality or data integrity. For instance, if we receive a message from Alice, the property of data integrity doesn't do us much good unless we know that it was in fact sent by Alice and not the attacker. Similarly, if we want to send a confidential message to Bob, it's not of much value to us if we're actually sending a confidential message to the attacker.

Note that peer entity authentication can be provided asymmetrically. When you call someone on the phone, you can be fairly certain that you have the right person -- or at least that you got a person who's actually at the phone number you called. On the other hand, if they don't have caller ID, then the receiver of a phone call has no idea who's calling them. Calling someone on the phone is an example of recipient authentication, since you know who the recipient of the call is, but they don't know anything about the sender.

In messaging situations, you often wish to use peer entity authentication to establish the identity of the sender of a certain message. In such contexts, this property is called DATA ORIGIN AUTHENTICATION.

2.2. Non-Repudiation

A system that provides endpoint authentication allows one party to be certain of the identity of someone with whom he is communicating. When the system provides data integrity a receiver can be sure of both the sender's identity and that he is receiving the data that that sender meant to send. However, he cannot necessarily demonstrate this fact to a third party. The ability to make this demonstration is called NON-REPUDIATION.

There are many situations in which non-repudiation is desirable. Consider the situation in which two parties have signed a contract which one party wishes to unilaterally abrogate. He might simply claim that he had never signed it in the first place. Non-repudiation prevents him from doing so, thus protecting the counterparty.

Unfortunately, non-repudiation can be very difficult to achieve in practice and naive approaches are generally inadequate. Section 4.3 describes some of the difficulties, which generally stem from the fact that the interests of the two parties are not aligned--one party wishes to prove something that the other party wishes to deny.

2.3. Systems Security

In general, systems security is concerned with protecting one's machines and data. The intent is that machines should be used only by authorized users and for the purposes that the owners intend. Furthermore, they should be available for those purposes. Attackers should not be able to deprive legitimate users of resources.

2.3.1. Unauthorized Usage

Most systems are not intended to be completely accessible to the public. Rather, they are intended to be used only by certain authorized individuals. Although many Internet services are available to all Internet users, even those servers generally offer a larger subset of services to specific users. For instance, Web Servers often will serve data to any user, but restrict the ability to modify pages to specific users. Such modifications by the general public would be UNAUTHORIZED USAGE.

2.3.2. Inappropriate Usage

Being an authorized user does not mean that you have free run of the system. As we said above, some activities are restricted to authorized users, some to specific users, and some activities are generally forbidden to all but administrators. Moreover, even activities which are in general permitted might be forbidden in some cases. For instance, users may be permitted to send email but forbidden from sending files above a certain size, or files which contain viruses. These are examples of INAPPROPRIATE USAGE.

2.3.3. Denial of Service

Recall that our third goal was that the system should be available to legitimate users. A broad variety of attacks are possible which threaten such usage. Such attacks are collectively referred to as DENIAL OF SERVICE attacks. Denial of service attacks are often very easy to mount and difficult to stop. Many such attacks are designed to consume machine resources, making it difficult or impossible to serve legitimate users. Other attacks cause the target machine to crash, completely denying service to users.

3. The Internet Threat Model

A THREAT MODEL describes the capabilities that an attacker is assumed to be able to deploy against a resource. It should contain such information as the resources available to an attacker in terms of information, computing capability, and control of the system. The

purpose of a threat model is twofold. First, we wish to identify the threats we are concerned with. Second, we wish to rule some threats explicitly out of scope. Nearly every security system is vulnerable to a sufficiently dedicated and resourceful attacker.

The Internet environment has a fairly well understood threat model. In general, we assume that the end-systems engaging in a protocol exchange have not themselves been compromised. Protecting against an attack when one of the end-systems has been compromised is extraordinarily difficult. It is, however, possible to design protocols which minimize the extent of the damage done under these circumstances.

By contrast, we assume that the attacker has nearly complete control of the communications channel over which the end-systems communicate. This means that the attacker can read any PDU (Protocol Data Unit) on the network and undetectably remove, change, or inject forged packets onto the wire. This includes being able to generate packets that appear to be from a trusted machine. Thus, even if the end-system with which you wish to communicate is itself secure, the Internet environment provides no assurance that packets which claim to be from that system in fact are.

It's important to realize that the meaning of a PDU is different at different levels. At the IP level, a PDU means an IP packet. At the TCP level, it means a TCP segment. At the application layer, it means some kind of application PDU. For instance, at the level of email, it might either mean an RFC-822 message or a single SMTP command. At the HTTP level, it might mean a request or response.

3.1. Limited Threat Models

As we've said, a resourceful and dedicated attacker can control the entire communications channel. However, a large number of attacks can be mounted by an attacker with fewer resources. A number of currently known attacks can be mounted by an attacker with limited control of the network. For instance, password sniffing attacks can be mounted by an attacker who can only read arbitrary packets. This is generally referred to as a PASSIVE ATTACK [INTAUTH]

By contrast, Morris's sequence number guessing attack [SEQNUM] can be mounted by an attacker who can write but not read arbitrary packets. Any attack which requires the attacker to write to the network is known as an ACTIVE ATTACK.

Thus, a useful way of organizing attacks is to divide them based on the capabilities required to mount the attack. The rest of this section describes these categories and provides some examples of each category.

3.2. Passive Attacks

In a passive attack, the attacker reads packets off the network but does not write them. The simplest way to mount such an attack is to simply be on the same LAN as the victim. On most common LAN configurations, including Ethernet, 802.3, and FDDI, any machine on the wire can read all traffic destined for any other machine on the same LAN. Note that switching hubs make this sort of sniffing substantially more difficult, since traffic destined for a machine only goes to the network segment which that machine is on.

Similarly, an attacker who has control of a host in the communications path between two victim machines is able to mount a passive attack on their communications. It is also possible to compromise the routing infrastructure to specifically arrange that traffic passes through a compromised machine. This might involve an active attack on the routing infrastructure to facilitate a passive attack on a victim machine.

Wireless communications channels deserve special consideration, especially with the recent and growing popularity of wireless-based LANs, such as those using 802.11. Since the data is simply broadcast on well-known radio frequencies, an attacker simply needs to be able to receive those transmissions. Such channels are especially vulnerable to passive attacks. Although many such channels include cryptographic protection, it is often of such poor quality as to be nearly useless. [WEP]

In general, the goal of a passive attack is to obtain information which the sender and receiver would rather remain private. Examples of such information include credentials useful in the electronic world such as passwords or credentials useful in the outside world, such as confidential business information.

3.2.1. Confidentiality Violations

The classic example of passive attack is sniffing some inherently private data off of the wire. For instance, despite the wide availability of SSL, many credit card transactions still traverse the Internet in the clear. An attacker could sniff such a message and recover the credit card number, which can then be used to make fraudulent transactions. Moreover, confidential business information is routinely transmitted over the network in the clear in email.

3.2.2. Password Sniffing

Another example of a passive attack is PASSWORD SNIFFING. Password sniffing is directed towards obtaining unauthorized use of resources.

Many protocols, including [TELNET], [POP], and [NNTP] use a shared password to authenticate the client to the server. Frequently, this password is transmitted from the client to the server in the clear over the communications channel. An attacker who can read this traffic can therefore capture the password and REPLAY it. That is to say that he can initiate a connection to the server and pose as the client and login using the captured password.

Note that although the login phase of the attack is active, the actual password capture phase is passive. Moreover, unless the server checks the originating address of connections, the login phase does not require any special control of the network.

3.2.3. Offline Cryptographic Attacks

Many cryptographic protocols are subject to OFFLINE ATTACKS. In such a protocol, the attacker recovers data which has been processed using the victim's secret key and then mounts a cryptanalytic attack on that key. Passwords make a particularly vulnerable target because they are typically low entropy. A number of popular password-based challenge response protocols are vulnerable to DICTIONARY ATTACK. The attacker captures a challenge-response pair and then proceeds to try entries from a list of common words (such as a dictionary file) until he finds a password that produces the right response.

A similar such attack can be mounted on a local network when NIS is used. The Unix password is crypted using a one-way function, but tools exist to break such crypted passwords [KLEIN]. When NIS is used, the crypted password is transmitted over the local network and an attacker can thus sniff the password and attack it.

Historically, it has also been possible to exploit small operating system security holes to recover the password file using an active attack. These holes can then be bootstrapped into an actual account by using the aforementioned offline password recovery techniques. Thus we combine a low-level active attack with an offline passive attack.

3.3. Active Attacks

When an attack involves writing data to the network, we refer to this as an ACTIVE ATTACK. When IP is used without IPsec, there is no authentication for the sender address. As a consequence, it's straightforward for an attacker to create a packet with a source address of his choosing. We'll refer to this as a SPOOFING ATTACK.

Under certain circumstances, such a packet may be screened out by the network. For instance, many packet filtering firewalls screen out all

packets with source addresses on the INTERNAL network that arrive on the EXTERNAL interface. Note, however, that this provides no protection against an attacker who is inside the firewall. In general, designers should assume that attackers can forge packets.

However, the ability to forge packets does not go hand in hand with the ability to receive arbitrary packets. In fact, there are active attacks that involve being able to send forged packets but not receive the responses. We'll refer to these as BLIND ATTACKS.

Note that not all active attacks require forging addresses. For instance, the TCP SYN denial of service attack [TCPSYN] can be mounted successfully without disguising the sender's address. However, it is common practice to disguise one's address in order to conceal one's identity if an attack is discovered.

Each protocol is susceptible to specific active attacks, but experience shows that a number of common patterns of attack can be adapted to any given protocol. The next sections describe a number of these patterns and give specific examples of them as applied to known protocols.

3.3.1. Replay Attacks

In a REPLAY ATTACK, the attacker records a sequence of messages off of the wire and plays them back to the party which originally received them. Note that the attacker does not need to be able to understand the messages. He merely needs to capture and retransmit them.

For example, consider the case where an S/MIME message is being used to request some service, such as a credit card purchase or a stock trade. An attacker might wish to have the service executed twice, if only to inconvenience the victim. He could capture the message and replay it, even though he can't read it, causing the transaction to be executed twice.

3.3.2. Message Insertion

In a MESSAGE INSERTION attack, the attacker forges a message with some chosen set of properties and injects it into the network. Often this message will have a forged source address in order to disguise the identity of the attacker.

For example, a denial-of-service attack can be mounted by inserting a series of spurious TCP SYN packets directed towards the target host. The target host responds with its own SYN and allocates kernel data structures for the new connection. The attacker never completes the

3-way handshake, so the allocated connection endpoints just sit there taking up kernel memory. Typical TCP stack implementations only allow some limited number of connections in this "half-open" state and when this limit is reached, no more connections can be initiated, even from legitimate hosts. Note that this attack is a blind attack, since the attacker does not need to process the victim's SYNs.

3.3.3. Message Deletion

In a MESSAGE DELETION attack, the attacker removes a message from the wire. Morris's sequence number guessing attack [SEQNUM] often requires a message deletion attack to be performed successfully. In this blind attack, the host whose address is being forged will receive a spurious TCP SYN packet from the host being attacked. Receipt of this SYN packet generates a RST, which would tear the illegitimate connection down. In order to prevent this host from sending a RST so that the attack can be carried out successfully, Morris describes flooding this host to create queue overflows such that the SYN packet is lost and thus never responded to.

3.3.4. Message Modification

In a MESSAGE MODIFICATION attack, the attacker removes a message from the wire, modifies it, and reinjects it into the network. This sort of attack is particularly useful if the attacker wants to send some of the data in the message but also wants to change some of it.

Consider the case where the attacker wants to attack an order for goods placed over the Internet. He doesn't have the victim's credit card number so he waits for the victim to place the order and then replaces the delivery address (and possibly the goods description) with his own. Note that this particular attack is known as a CUT-AND-PASTE attack since the attacker cuts the credit card number out of the original message and pastes it into the new message.

Another interesting example of a cut-and-paste attack is provided by [IPSPPROB]. If IPsec ESP is used without any MAC then it is possible for the attacker to read traffic encrypted for a victim on the same machine. The attacker attaches an IP header corresponding to a port he controls onto the encrypted IP packet. When the packet is received by the host it will automatically be decrypted and forwarded to the attacker's port. Similar techniques can be used to mount a session hijacking attack. Both of these attacks can be avoided by always using message authentication when you use encryption. Note that this attack only works if (1) no MAC check is being used, since this attack generates damaged packets (2) a host-to-host SA is being used, since a user-to-user SA will result in an inconsistency between the port associated with the SA and the target port. If the receiving

machine is single-user than this attack is infeasible.

3.3.5. Man-In-The-Middle

A MAN-IN-THE-MIDDLE attack combines the above techniques in a special form: The attacker subverts the communication stream in order to pose as the sender to receiver and the receiver to the sender:

```
  What Alice and Bob think:
  Alice  <-----------------------------------------------> Bob

  What's happening:
  Alice  <----------------> Attacker <----------------> Bob
```

This differs fundamentally from the above forms of attack because it attacks the identity of the communicating parties, rather than the data stream itself. Consequently, many techniques which provide integrity of the communications stream are insufficient to protect against man-in-the-middle attacks.

Man-in-the-middle attacks are possible whenever a protocol lacks PEER ENTITY AUTHENTICATION. For instance, if an attacker can hijack the client TCP connection during the TCP handshake (perhaps by responding to the client's SYN before the server does), then the attacker can open another connection to the server and begin a man-in-the-middle attack. It is also trivial to mount man-in-the-middle attacks on local networks via ARP spoofing--the attacker forges an ARP with the victim's IP address and his own MAC address. Tools to mount this sort of attack are readily available.

Note that it is only necessary to authenticate one side of the transaction in order to prevent man-in-the-middle attacks. In such a situation the the peers can establish an association in which only one peer is authenticated. In such a system, an attacker can initiate an association posing as the unauthenticated peer but cannot transmit or access data being sent on a legitimate connection. This is an acceptable situation in contexts such as Web e-commerce where only the server needs to be authenticated (or the client is independently authenticated via some non-cryptographic mechanism such as a credit card number).

4. Common Issues

Although each system's security requirements are unique, certain common requirements appear in a number of protocols. Often, when naive protocol designers are faced with these requirements, they choose an obvious but insecure solution even though better solutions are available. This section describes a number of issues seen in many

protocols and the common pieces of security technology that may be useful in addressing them.

4.1. User Authentication

Essentially every system which wants to control access to its resources needs some way to authenticate users. A nearly uncountable number of such mechanisms have been designed for this purpose. The next several sections describe some of these techniques.

4.1.1. Username/Password

The most common access control mechanism is simple USERNAME/PASSWORD The user provides a username and a reusable password to the host which he wishes to use. This system is vulnerable to a simple passive attack where the attacker sniffs the password off the wire and then initiates a new session, presenting the password. This threat can be mitigated by hosting the protocol over an encrypted connection such as TLS or IPSEC. Unprotected (plaintext) username/password systems are not acceptable in IETF standards.

4.1.2. Challenge Response and One Time Passwords

Systems which desire greater security than USERNAME/PASSWORD often employ either a ONE TIME PASSWORD [OTP] scheme or a CHALLENGE-RESPONSE. In a one time password scheme, the user is provided with a list of passwords, which must be used in sequence, one time each. (Often these passwords are generated from some secret key so the user can simply compute the next password in the sequence.) SecureID and DES Gold are variants of this scheme. In a challenge-response scheme, the host and the user share some secret (which often is represented as a password). In order to authenticate the user, the host presents the user with a (randomly generated) challenge. The user computes some function based on the challenge and the secret and provides that to the host, which verifies it. Often this computation is performed in a handheld device, such as a DES Gold card.

Both types of scheme provide protection against replay attack, but often still vulnerable to an OFFLINE KEYSEARCH ATTACK (a form of passive attack): As previously mentioned, often the one-time password or response is computed from a shared secret. If the attacker knows the function being used, he can simply try all possible shared secrets until he finds one that produces the right output. This is made easier if the shared secret is a password, in which case he can mount a DICTIONARY ATTACK--meaning that he tries a list of common words (or strings) rather than just random strings.

These systems are also often vulnerable to an active attack. Unless communication security is provided for the entire session, the attacker can simply wait until authentication has been performed and hijack the connection.

4.1.3. Certificates

A simple approach is to have all users have CERTIFICATES [PKIX] which they then use to authenticate in some protocol-specific way, as in [TLS] or [S/MIME]. A certificate is a signed credential binding an entity's identity to its public key. The signer of a certificate is a CERTIFICATE AUTHORITY (CA), whose certificate may itself be signed by some superior CA. In order for this system to work, trust in one or more CAs must be established in an out-of-band fashion. Such CAs are referred to as TRUSTED ROOTS or ROOT CAS. The primary obstacle to this approach in client-server type systems is that it requires clients to have certificates, which can be a deployment problem.

4.1.4. Some Uncommon Systems

There are ways to do a better job than the schemes mentioned above, but they typically don't add much security unless communications security (at least message integrity) will be employed to secure the connection, because otherwise the attacker can merely hijack the connection after authentication has been performed. A number of protocols ([EKE], [SPEKE], [SRP]) allow one to securely bootstrap a user's password into a shared key which can be used as input to a cryptographic protocol. One major obstacle to the deployment of these protocols has been that their Intellectual Property status is extremely unclear. Similarly, the user can authenticate using public key certificates. (e.g. S-HTTP client authentication). Typically these methods are used as part of a more complete security protocol.

4.1.5. Host Authentication

Host authentication presents a special problem. Quite commonly, the addresses of services are presented using a DNS hostname, for instance as a URL [URL]. When requesting such a service, one has to ensure that the entity that one is talking to not only has a certificate but that that certificate corresponds to the expected identity of the server. The important thing to have is a secure binding between the certificate and the expected hostname.

For instance, it is usually not acceptable for the certificate to contain an identity in the form of an IP address if the request was for a given hostname. This does not provide end-to-end security because the hostname-IP mapping is not secure unless secure name resolution [DNSSEC] is being used. This is a particular problem when the

hostname is presented at the application layer but the authentication is performed at some lower layer.

4.2. Generic Security Frameworks

Providing security functionality in a protocol can be difficult. In addition to the problem of choosing authentication and key establishment mechanisms, one needs to integrate it into a protocol. One response to this problem (embodied in IPsec and TLS) is to create a lower-level security protocol and then insist that new protocols be run over that protocol.

Another approach that has recently become popular is to design generic application layer security frameworks. The idea is that you design a protocol that allows you to negotiate various security mechanisms in a pluggable fashion. Application protocol designers then arrange to carry the security protocol PDUs in their application protocol. Examples of such frameworks include GSS-API [GSS] and SASL [SASL].

The generic framework approach has a number of problems. First, it is highly susceptible to DOWNGRADE ATTACKS. In a downgrade attack, an active attacker tampers with the negotiation in order to force the parties to negotiate weaker protection than they otherwise would. It's possible to include an integrity check after the negotiation and key establishment have both completed, but the strength of this integrity check is necessarily limited to the weakest common algorithm. This problem exists with any negotiation approach, but generic frameworks exacerbate it by encouraging the application protocol author to just specify the framework rather than think hard about the appropriate underlying mechanisms, particularly since the mechanisms can very widely in the degree of security offered.

Another problem is that it's not always obvious how the various security features in the framework interact with the application layer protocol. For instance, SASL can be used merely as an authentication framework--in which case the SASL exchange occurs but the rest of the connection is unprotected, but can also negotiate TLS as a mechanism. Knowing under what circumstances TLS is optional and which it is required requires thinking about the threat model.

In general, authentication frameworks are most useful in situations where users have a wide variety of credentials that must all be accomodated by some service. When the security requirements of a system can be clearly identified and only a few forms of authentication are used, choosing a single security mechanism leads to greater simplicity and predictability. In situations where a framework is to be used, designers SHOULD carefully examine the framework's options and

specify only the mechanisms that are appropriate for their particular threat model. If a framework is necessary, designers SHOULD choose one of the established ones instead of designing their own.

4.3. Non-repudiation

The naive approach to non-repudiation is simply to use public-key digital signatures over the content. The party who wishes to be bound (the SIGNING PARTY) digitally signs the message in question. The counterparty (the RELYING PARTY) can later point to the digital signature as proof that the signing party at one point agreed to the disputed message. Unfortunately, this approach is insufficient.

The easiest way for the signing party to repudiate the message is by claiming that his private key has been compromised and that some attacker (though not necessarily the relying party) signed the disputed message. In order to defend against this attack the relying party needs to demonstrate that the signing party's key had not been compromised at the time of the signature. This requires substantial infrastructure, including archival storage of certificate revocation information and timestamp servers to establish the time that the message was signed.

Additionally, the relying party might attempt to trick the signing party into signing one message while thinking he's signing another. This problem is particularly severe when the relying party controls the infrastructure that the signing party uses for signing, such as in kiosk situations. In many such situations the signing party's key is kept on a smartcard but the message to be signed is displayed by the relying party.

All of these complications make non-repudiation a difficult service to deploy in practice.

4.4. Authorization vs. Authentication

AUTHORIZATION is the process by which one determines whether an authenticated party has permission to access a particular resource or service. Although tightly bound, it is important to realize that authentication and authorization are two separate mechanisms. Perhaps because of this tight coupling, authentication is sometimes mistakenly thought to imply authorization. Authentication simply identifies a party, authorization defines whether they can perform a certain action.

Authorization necessarily relies on authentication, but authentication alone does not imply authorization. Rather, before granting permission to perform an action, the authorization mechanism must be

consulted to determine whether that action is permitted.

4.4.1. Access Control Lists

One common form of authorization mechanism is an access control list (ACL) that lists users that are permitted access to a resource. Since assigning individual authorization permissions to each resource is tedious, often resources are hierarchically arranged such that the parent resource's ACL is inherited by child resources. This allows administrators to set top level policies and override them when necessary.

4.4.2. Certificate Based Systems

While the distinction between authentication and authorization is intuitive when using simple authentication mechanisms such as username and password (i.e., everyone understands the difference between the administrator account and a user account), with more complex authentication mechanisms the distinction is sometimes lost.

With certificates, for instance, presenting a valid signature does not imply authorization. The signature must be backed by a certificate chain that contains a trusted root, and that root must be trusted in the given context. For instance, users who possess certificates issued by the Acme MIS CA may have different web access privileges than users who possess certificates issued by the Acme Accounting CA, even though both of these CAs are "trusted" by the Acme web server.

Mechanisms for enforcing these more complicated properties have not yet been completely explored. One approach is simply to attach policies to ACLs describing what sorts of certificates are trusted. Another approach is to carry that information with the certificate, either as a certificate extension/attribute [PKIX, SPKI] or as a separate "Attribute Certificate".

4.5. Providing Traffic Security

Securely designed protocols should provide some mechanism for securing (meaning integrity protecting, authenticating, and possibly encrypting) all sensitive traffic. One approach is to secure the protocol itself, as in [DNSSEC], [S/MIME] or [S-HTTP]. Although this provides security which is most fitted to the protocol, it also requires considerable effort to get right.

Many protocols can be adequately secured using one of the available channel security systems. We'll discuss the two most common, IPsec [AH, ESP] and [TLS].

4.5.1. IPsec

The IPsec protocols (specifically, AH and ESP) can provide transmission security for all traffic between two hosts. The IPsec protocols support varying granularities of user identification, including for example "IP Subnet", "IP Address", "Fully Qualified Domain Name", and individual user ("Mailbox name"). These varying levels of identification are employed as inputs to access control facilities that are an intrinsic part of IPsec. However, a given IPsec implementation might not support all identity types. In particular, security gateways may not provide user-to-user authentication or have mechanisms to provide that authentication information to applications.

When AH or ESP is used, the application programmer might not need to do anything (if AH or ESP has been enabled system-wide) or might need to make specific software changes (e.g. adding specific setsockopt() calls) -- depending on the AH or ESP implementation being used. Unfortunately, APIs for controlling IPsec implementations are not yet standardized.

The primary obstacle to using IPsec to secure other protocols is deployment. The major use of IPsec at present is for VPN applications, especially for remote network access. Without extremely tight coordination between security administrators and application developers, VPN usage is not well suited to providing security services for individual applications since it is difficult for such applications to determine what security services have in fact been provided.

IPsec deployment in host-to-host environments has been slow. Unlike application security systems such as TLS, adding IPsec to a non-IPsec system generally involves changing the operating system, either by tampering with the kernel or installing new drivers. This is a substantially greater undertaking than simply installing a new application. However, recent versions of a number of commodity operating systems include IPsec stacks, so deployment is becoming easier.

In environments where IPsec is sure to be available, it represents a viable option for protecting application communications traffic. If the traffic to be protected is UDP, IPsec and application-specific object security are the only options. However, designers MUST not assume that IPsec will be available. A security policy for a generic application layer protocol SHOULD not simply state that IPsec must be used, unless there is some reason to believe that IPsec will be available in the intended deployment environment. In environments where IPsec may not be available and the traffic is solely TCP, TLS is the method of choice, since the application developer can easily ensure its presence by including a TLS implementation in his package.

4.5.2. SSL/TLS

The currently most common approach is to use SSL or its successor TLS. They provide channel security for a TCP connection at the application level. That is, they run over TCP. SSL implementations typically provide a Berkeley Sockets-like interface for easy programming. The primary issue when designing a protocol solution around TLS is to differentiate between connections protected using TLS and those which are not.

The two primary approaches used are to have a separate well-known port for TLS connections (e.g. the HTTP over TLS port is 443) [HTTPTLS] or to have a mechanism for negotiating upward from the base protocol to TLS as in [UPGRADE] or [STARTTLS]. When an upward negotiation strategy is used, care must be taken to ensure that an attacker can not force a clear connection when both parties wish to use TLS.

Note that TLS depends upon a reliable protocl such as TCP or SCTP. This produces two notable difficulties. First, it cannot be used to secure datagram protocols that use UDP. Second, TLS is susceptible to IP layer attacks that IPsec is not. Typically, these attacks take some form of denial of service or connection assassination. For instance, an attacker might forge a TCP RST to shut down SSL connections. TLS has mechanisms to detect truncation attacks but these merely allow the victim to know he is being attacked and do not provide connection survivability in the face of such attacks. By contrast, if IPsec were being used, such a forged RST could be rejected without affecting the TCP conection.

4.5.3. Remote Login

In some special cases it may be worth providing channel-level security directly in the application rather than using IPSEC or SSL/TLS. One such case is remote terminal security. Characters are typically delivered from client to server one character at a time. Since SSL/TLS and AH/ESP authenticate and encrypt every packet, this can mean a data expansion of 20-fold. The telnet encryption option [ENCOPT] prevents this expansion by foregoing message integrity.

When using remote terminal service, it's often desirable to securely perform other sorts of communications services. In addition to providing remote login, SSH [SSH] also provides secure port forwarding for arbitrary TCP ports, thus allowing users run arbitrary TCP-based applications over the SSH channel. Note that SSH Port Forwarding can be security issue if it is used improperly to circumvent firewall and improperly expose insecure internal applications to the outside world.

4.6. Denial of Service Attacks and Countermeasures

Denial of service attacks are all too frequently viewed as an fact of life. One problem is that an attacker can often choose from one of many denial of service attacks to inflict upon a victim, and because most of these attacks cannot be thwarted, common wisdom frequently assumes that there is no point protecting against one kind of denial of service attack when there are many other denial of service attacks that are possible but that cannot be prevented.

However, not all denial of service attacks are equal and more importantly, it is possible to design protocols such that denial of service attacks are made more difficult if not impractical. Recent SYN flood attacks [TCPSYN] demonstrate both of these properties: SYN flood attacks are so easy, anonymous, and effective that they are more attractive to attackers than other attacks; and because the design of TCP enables this attack.

Because complete DoS protection is so difficult, security against DoS must be dealt with pragmatically. In particular, some attacks which would be desirable to defend against cannot be defended against economically. The goal should be to manage risk by defending against attacks with sufficiently high ratios of severity to cost of defense. Both severity of attack and cost of defense change as technology changes and therefore so does the set of attacks which should be defended against.

Authors of internet standards MUST describe which denial of service attacks their protocol is susceptable to. This description MUST include the reasons it was either unreasonable or out of scope to attempt to avoid these denial of service attacks.

4.6.1. Blind Denial of Service

BLIND denial of service attacks are particularly pernicious. With a blind attack the attacker has a significant advantage. If the attacker must be able to receive traffic from the victim then he must either subvert the routing fabric or must use his own IP address. Either provides an opportunity for victim to track the attacker and/or filter out his traffic. With a blind attack the attacker can use forged IP addresses, making it extremely difficult for the victim to filter out his packets. The TCP SYN flood attack is an example of a blind attack. Designers should make every attempt possible to prevent blind denial of service attacks.

4.6.2. Distributed Denial of Service

Even more dangerous are DISTRIBUTED denial of service attacks (DDoS) [DDOS] In a DDoS the attacker arranges for a number of machines to attack the target machine simultaneously. Usually this is accomplished by infecting a large number of machines with a program that allows remote initiation of attacks. The machines actually performing the attack are called ZOMBIEs and are likely owned by unsuspecting third parties in an entirely different location from the true attacker. DDoS attacks can be very hard to counter because the zombies often appear to be making legitimate protocol requests and simply crowd out the real users. DDoS attacks can be difficult to thwart, but protocol designers are expected to be cognizant of these forms of attack while designing protocols.

4.6.3. Avoiding Denial of Service

There are two common approaches to making denial of service attacks more difficult:

4.6.3.1. Make your attacker do more work than you do

If an attacker consumes more of his resources than yours when launching an attack, attackers with fewer resources than you will be unable to launch effective attacks. One common technique is to require the attacker perform a time-intensive operation, such as a cryptographic operation. Note that an attacker can still mount a denial of service attack if he can muster substantially sufficient CPU power. For instance, this technique would not stop the distributed attacks described in [TCPSYN].

4.6.3.2. Make your attacker prove they can receive data from you

A blind attack can be subverted by forcing the attack prove that they can can receive data from the victim. A common technique is to require that the attacker reply using information that was gained earlier in the message exchange. If this countermeasure is used, the attacker must either use his own address (making him easy to track) or to forge an address which will be routed back along a path that traverses the host from which the attack is being launched.

Hosts on small subnets are thus useless to the attacker (at least in the context of a spoofing attack) because the attack can be traced back to a subnet (which should be sufficient for locating the attacker) so that anti-attack measures can be put into place (for instance, a boundary router can be configured to drop all traffic from that subnet). A common technique is to require that the attacker reply using information that was gained earlier in the message

exchange.

4.6.4. Example: TCP SYN Floods

TCP/IP is vulnerable to SYN flood attacks (which are described in section 3.3.2) because of the design of the 3-way handshake. First, an attacker can force a victim to consume significant resources (in this case, memory) by sending a single packet. Second, because the attacker can perform this action without ever having received data from the victim, the attack can be performed anonymously (and therefore using a large number of forged source addresses).

4.6.5. Example: Photuris

[PHOTURIS] specifies an anti-clogging mechanism that prevents attacks on Photuris that resemble the SYN flood attack. Photuris employs a time-variant secret to generate a "cookie" which is returned to the attacker. This cookie must be returned in subsequent messages for the exchange to progress. The interesting feature is that this cookie can be re-generated by the victim later in the exchange, and thus no state need be retained by the victim until after the attacker has proven that he can receive packets from the victim.

4.7. Object vs. Channel Security

It's useful to make the conceptual distinction between object security and channel security. Object security refers to security measures which apply to entire data objects. Channel security measures provide a secure channel over which objects may be carried transparently but the channel has no special knowledge about object boundaries.

Consider the case of an email message. When it's carried over an IPSEC or TLS secured connection, the message is protected during transmission. However, it is unprotected in the receiver's mailbox, and in intermediate spool files along the way. Moreover, since mail servers generally run as a daemon, not a user, authentication of messages generally merely means authentication of the daemon not the user. Finally, since mail transport is hop-by-hop, even if the user authenticates to the first hop relay the authentication can't be safely verified by the receiver.

By contrast, when an email message is protected with S/MIME or OpenPGP, the entire message is encrypted and integrity protected until it is examined and decrypted by the recipient. It also provides strong authentication of the actual sender, as opposed to the machine the message came from. This is object security. Moreover, the receiver can prove the signed message's authenticity to a third

party.

Note that the difference between object and channel security is a matter of perspective. Object security at one layer of the protocol stack often looks like channel security at the next layer up. So, from the perspective of the IP layer, each packet looks like an individually secured object. But from the perspective of a web client, IPSEC just provides a secure channel.

The distinction isn't always clear-cut. For example, S-HTTP provides object level security for a single HTTP transaction, but a web page typically consists of multiple HTTP transactions (the base page and numerous inline images.) Thus, from the perspective of the total web page, this looks rather more like channel security. Object security for a web page would consist of security for the transitive closure of the page and all its embedded content as a single unit.

5. Writing Security Considerations Sections

While it is not a requirement that any given protocol or system be immune to all forms of attack, it is still necessary for authors to consider them. Part of the purpose of the Security Considerations section is to explain what attacks are out of scope and what countermeasures can be applied to defend against them.

There should be a clear description of the kinds of threats on the described protocol or technology. This should be approached as an effort to perform "due diligence" in describing all known or foreseeable risks and threats to potential implementers and users.

Authors MUST describe

1. which attacks are out of scope (and why!)
2. which attacks are in-scope
2.1 and the protocol is susceptable to
2.2 and the protocol protects against

At least the following forms of attack MUST be considered: eavesdropping, replay, message insertion, deletion, modification, and man-in-the-middle. Potential denial of service attacks MUST be identified as well. If the protocol incorporates cryptographic protection mechanisms, it should be clearly indicated which portions of the data are protected and what the protections are (i.e. integrity only, confidentiality, and/or endpoint authentication, etc.). Some indication should also be given to what sorts of attacks the cryptographic protection is susceptible. Data which should be held secret (keying material, random seeds, etc.) should be clearly labeled.

If the technology involves authentication, particularly user-host authentication, the security of the authentication method MUST be clearly specified. That is, authors MUST document the assumptions that the security of this authentication method is predicated upon. For instance, in the case of the UNIX username/password login method, a statement to the effect of:

> Authentication in the system is secure only to the extent that it is difficult to guess or obtain a ASCII password that is a maximum of 8 characters long. These passwords can be obtained by sniffing telnet sessions or by running the 'crack' program using the contents of the /etc/passwd file. Attempts to protect against on-line password guessing by (1) disconnecting after several unsuccessful login attempts and (2) waiting between successive password prompts is effective only to the extent that attackers are impatient.
>
> Because the /etc/passwd file maps usernames to user ids, groups, etc. it must be world readable. In order to permit this usage but make running crack more difficult, the file is often split into /etc/passwd and a 'shadow' password file. The shadow file is not world readable and contains the encrypted password. The regular /etc/passwd file contains a dummy password in its place.

It is insufficient to simply state that one's protocol should be run over some lower layer security protocol. If a system relies upon lower layer security services for security, the protections those services are expected to provide MUST be clearly specified. In addition, the resultant properties of the combined system need to be specified.

Note: In general, the IESG will not approve standards track protocols which do not provide for strong authentication, either internal to the protocol or through tight binding to a lower layer security protocol.

The threat environment addressed by the Security Considerations section MUST at a minimum include deployment across the global Internet across multiple administrative boundaries without assuming that firewalls are in place, even if only to provide justification for why such consideration is out of scope for the protocool. It is not acceptable to only discuss threats applicable to LANs and ignore the broader threat environment. All IETF standards-track protocols are considered likely to have deployment in the global Internet. In some cases, there might be an Applicability Statement discouraging use of a technology or protocol in a particular environment. Nonetheless, the security issues of broader deployment should be discussed in the document.

There should be a clear description of the residual risk to the user or operator of that protocol after threat mitigation has been deployed. Such risks might arise from compromise in a related protocol (e.g. IPsec is useless if key management has been compromised), from incorrect implementation, compromise of the security technology used for risk reduction (e.g. a cipher with a 40-bit key), or there might be risks that are not addressed by the protocol specification (e.g. denial of service attacks on an underlying link protocol).

There should also be some discussion of potential security risks arising from potential misapplications of the protocol or technology described in the RFC. This might be coupled with an Applicability Statement for that RFC.

6. Examples

This section consists of some example security considerations sections, intended to give the reader a flavor of what's intended by this document.

The first example is a 'retrospective' example, applying the criteria of this document to a historical document, RFC-821. The second example is a good security considerations section clipped from a current protocol.

6.1. SMTP

When RFC-821 was written, Security Considerations sections were not required in RFCs, and none is contained in that document. Had that document been written today, the Security Considerations section might look something like this:

6.1.1. SMTP Security Considerations

SMTP as-is provides no security precautions of any kind. All the attacks we are about to describe must be provided by a different protocol layer.

A passive attack is sufficient to recover message text. No endpoint authentication is provided by the protocol. Sender spoofing is trivial, and therefore forging email messages is trivial. Some implementations do add header lines with hostnames derived through reverse name resolution (which is only secure to the extent that it is difficult to spoof DNS -- not very), although these header lines are normally not displayed to users. Receiver spoofing is also fairly straight-forward, either using TCP connection hijacking or DNS spoofing. Moreover, since email messages often pass through SMTP gateways, all intermediate gateways must be trusted, a condition nearly

impossible on the global Internet.

Several approaches are available for alleviating these threats. In order of increasingly high level in the protocol stack, we have:

SMTP over IPSEC
SMTP/TLS
S/MIME and PGP/MIME

6.1.1.1. SMTP over IPSEC

An SMTP connection run over IPSEC can provide confidentiality for the message between the sender and the first hop SMTP gateway, or between any pair of connected SMTP gateways. That is to say, it provides channel security for the SMTP connections. In a situation where the message goes directly from the client to the receiver's gateway, this may provide substantial security (though the receiver must still trust the gateway). Protection is provided against replay attacks, since the data itself is protected and the packets cannot be replayed.

Endpoint identification is a problem, however, unless the receiver's address can be directly cryptographically authenticated. No sender identification is available, since the sender's machine is authenticated, not the sender himself. Furthermore, the identity of the sender simply appears in the From header of the message, so it is easily spoofable by the sender. Finally, unless the security policy is set extremely strictly, there is also an active downgrade to cleartext attack.

6.1.1.2. SMTP/TLS

SMTP can be combined with TLS as described in [STARTTLS]. This provides similar protection to that provided when using IPSEC. Since TLS certificates typically contain the server's host name, recipient authentication may be slightly more obvious, but is still susceptible to DNS spoofing attacks. Notably, common implementations of TLS contain a US exportable (and hence low security) mode. Applications desiring high security should ensure that this mode is disabled. Protection is provided against replay attacks, since the data itself is protected and the packets cannot be replayed. [note: The Security Considerations section of the SMTP over TLS draft is quite good and bears reading as an example of how to do things.]

6.1.1.3. S/MIME and PGP/MIME

S/MIME and PGP/MIME are both message oriented security protocols. They provide object security for individual messages. With various settings, sender and recipient authentication and confidentiality may be provided. More importantly, the identification is not of the sending and receiving machines, but rather of the sender and recipient themselves. (Or, at least, of cryptographic keys corresponding to the sender and recipient.) Consequently, end-to-end security may be obtained. Note, however, that no protection is provided against replay attacks.

6.1.1.4. Denial of Service

None of these security measures provides any real protection against denial of service. SMTP connections can easily be used to tie up system resources in a number of ways, including excessive port consumption, excessive disk usage (email is typically delivered to disk files), and excessive memory consumption (sendmail, for instance, is fairly large, and typically forks a new process to deal with each message.)

6.1.1.5. Inappropriate Usage

In particular, there is no protection provided against unsolicited mass email (aka SPAM).

SMTP also includes several commands which may be used by attackers to explore the machine on which the SMTP server runs. The VRFY command permits an attacker to convert user-names to mailbox name and often real name. This is often useful in mounting a password guessing attack, as many users use their name as their password. EXPN permits an attacker to expand an email list to the names of the subscribers. This may be used in order to generate a list of legitimate users in order to attack their accounts, as well as to build mailing lists for future SPAM. Administrators may choose to disable these commands.

6.2. VRRP

The second example is from VRRP, the Virtual Router Redundance Protocol ([VRRP]). We reproduce here the Security Considerations section from that document (with new section numbers). Our comments are indented and prefaced with 'NOTE:'.

6.2.1. Security Considerations

VRRP is designed for a range of internetworking environments that may employ different security policies. The protocol includes several

authentication methods ranging from no authentication, simple clear text passwords, and strong authentication using IP Authentication with MD5 HMAC. The details on each approach including possible attacks and recommended environments follows.

Independent of any authentication type VRRP includes a mechanism (setting TTL=255, checking on receipt) that protects against VRRP packets being injected from another remote network. This limits most vulnerabilities to local attacks.

> NOTE: The security measures discussed in the following sections only provide various kinds of authentication. No confidentiality is provided at all. This should be explicitly described as outside the scope.

6.2.1.1. No Authentication

The use of this authentication type means that VRRP protocol exchanges are not authenticated. This type of authentication SHOULD only be used in environments were there is minimal security risk and little chance for configuration errors (e.g., two VRRP routers on a LAN).

6.2.1.2. Simple Text Password

The use of this authentication type means that VRRP protocol exchanges are authenticated by a simple clear text password.

This type of authentication is useful to protect against accidental misconfiguration of routers on a LAN. It protects against routers inadvertently backing up another router. A new router must first be configured with the correct password before it can run VRRP with another router. This type of authentication does not protect against hostile attacks where the password can be learned by a node snooping VRRP packets on the LAN. The Simple Text Authentication combined with the TTL check makes it difficult for a VRRP packet to be sent from another LAN to disrupt VRRP operation.

This type of authentication is RECOMMENDED when there is minimal risk of nodes on a LAN actively disrupting VRRP operation. If this type of authentication is used the user should be aware that this clear text password is sent frequently, and therefore should not be the same as any security significant password.

NOTE: This section should be clearer. The basic point is that no authentication and Simple Text are only useful for a very limited threat model, namely that none of the nodes on the local LAN are hostile. The TTL check prevents hostile nodes off-LAN from posing as valid nodes, but nothing stops hostile nodes on-LAN from impersonating authorized nodes. This is not a particularly realistic threat model in many situations. In particular, it's extremely brittle: the compromise of any node the LAN allows reconfiguration of the VRRP nodes.

6.2.1.3. IP Authentication Header

The use of this authentication type means the VRRP protocol exchanges are authenticated using the mechanisms defined by the IP Authentication Header [AH] using [HMAC]. This provides strong protection against configuration errors, replay attacks, and packet corruption/modification.

This type of authentication is RECOMMENDED when there is limited control over the administration of nodes on a LAN. While this type of authentication does protect the operation of VRRP, there are other types of attacks that may be employed on shared media links (e.g., generation of bogus ARP replies) which are independent from VRRP and are not protected.

NOTE: It's a mistake to have AH be a RECOMMENDED in this context. Since AH is the only mechanism that protects VRRP against attack from other nodes on the same LAN, it should be a MUST for cases where there are untrusted nodes on the same network. In any case, AH should be a MUST implement. Additionally, there should be a required algorithm (HMAC-SHA1)

NOTE: Specifically, although securing VRRP prevents unauthorized machines from taking part in the election protocol, it does not protect hosts on the network from being deceived. For example, a gratutitous ARP reply from what purports to be the virtual router's IP address can redirect traffic to an unauthorized machine. Similarly, individual connections can be diverted by means of forged ICMP Redirect messages.

Acknowledgments

This document is heavily based on a note written by Ran Atkinson in 1997. That note was written after the IAB Security Workshop held in early 1997, based on input from everyone at that workshop. Some of the specific text above was taken from Ran's original document, and

some of that text was taken from an email message written by Fred Baker. The other primary source for this document is specific comments received from Steve Bellovin. Early review of this document was done by Lisa Dusseault and Mark Schertler

References

[AH] Kent, S., and Atkinson, R., "IP Authentication Header", RFC 2402, November 1998.

[DDOS] "Denial-Of-Service Tools" CERT Advisory CA-1999-17, 28 December 1999, CERT http://www.cert.org/advisories/CA-1999-17.html

[DNSSEC] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[EKE] Bellovin, S., Merritt, M., "Encrypted Key Exchange: Password-based protocols secure against dictionary attacks", Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1992.

[ENCOPT] Tso, T., "Telnet Data Encryption Option", RFC 2946, September, 2000.

[ESP] Kent, S., and Atkinson, R., "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[GSS] Linn, J., "Generic Security Services Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[HTTPTLS] Rescorla, E., "HTTP over TLS", RFC 2818, May 2000.

[HMAC] Krawczyk, H., Bellare, M., Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[INTAUTH] Haller, N., Atkinson, R., "On Internet Authentication", RFC 1704, October 1994.

[IPSPPROB] Bellovin, S. M., "Problem Areas for the IP Security Protocols", Proceedings of the Sixth Usenix UNIX Security Symposium, July 1996.

[KLEIN] Klein, D.V., "Foiling the Cracker: A Survey of and Improvements to Password Security", 1990.

[NNTP] Kantor, B, and Lapsley, P., "Network News Transfer Protocol", RFC 977, February 1986.

[OTP] Haller, N., Metz, C., Nesser, P., "A One-Time Password System", Straw, M., RFC 2289, February 1998.

[PHOTURIS] Karn, P., and Simpson, W., "Photuris: Session-Key Management Protocol", RFC 2522, March 1999.

[PKIX] Housley, R., Ford, W., Polk, W., Solo, D., Internet X.509 "Public Key Infrastructure Certificate and CRL Profile", RFC 2459, January 1999.

[POP] Myers, J., and Rose, M., "Post Office Protocol - Version 3", RFC 1939, May 1996.

[RFC-2223] Postel J., and Reynolds J., "Instructions to RFC Authors", RFC 2223, October 1997.

[SASL] Myers, J., "Simple Authenticatin and Security Layer (SASL)", RFC 2222, October 1997.

[SEQNUM] Morris, R.T., "A Weakness in the 4.2 BSD UNIX TCP/IP Software", AT&T Bell Laboratories, CSTR 117, 1985.

[SPKI] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T., "SPKI Certificate Theory", RFC 2693, September 1999.

[SPEKE] Jablon, D., "Strong Password-Only Authenticated Key Exchange", Computer Communication Review, ACM SIGCOMM, vol. 26, no. 5, pp. 5-26, October 1996.

[SRP] Wu T., "The Secure Remote Password Protocol", ISOC NDSS Symposium, 1998.

[SSH] Ylonen, T., "SSH - Secure Login Connections Over the Internet", 6th USENIX Security Symposium, p. 37-42, July 1996.

[STARTTLS] Hoffman, P., "SMTP Service Extension for Secure SMTP over TLS", RFC 2487, January 1998.

[S-HTTP] Rescorla, E., and Schiffman, A., "The Secure HyperText Transfer Protocol", RFC 2660, August 1999.

[S/MIME] Ramsdell, B., Ed., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[TELNET] Postel, J., and Reynolds, J., "Telnet Protocol Specification", RFC 854, May 1983.

[TLS] Dierks, T., and Allen, C., "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[TCPSYN] "TCP SYN Flooding and IP Spoofing Attacks", CERT Advisory CA-1996-21, 19 September 1996, CERT. http://www.cert.org/advisories/CA-1996-21.html

[UPGRADE] Khare, R., Lawrence, S., "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

[URL] Berners-Lee, T., Masinter, M., McCahill, M., "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[VRRP] Knight, S., Weaver, D., Whipple, D., Hinden, R., Mitzel, D., Hunt, P., Higginson, P., Shand, M., Lindemn, A., "Virtual Router Redundancy Protocol", RFC 2338, April 1998.

[WEP] Borisov, N., Goldberg, I., Wagner, D., "Intercepting Mobile Communications: The Insecurity of 802.11", http://www.isaac.cs.berkeley.edu/isaac/wep-draft.pdf

Security Considerations

This entire document is about security considerations.

Author's Address

Eric Rescorla <ekr@rtfm.com>
RTFM, Inc.
2439 Alvin Drive
Mountain View, CA 94043
Phone: (650)-320-8549

Brian Korver <bkorver@xythos.com>
Xythos Software
77 Maiden Lane, Suite 200
San Francisco, CA, USA
Phone: (415)-248-3800

Table of Contents