Effects of Mobility and Multihoming on Transport-Protocol Security

Tuomas Aura
Microsoft Research, Cambridge, UK
tuomaura@microsoft.com

Pekka Nikander and Gonzalo Camarillo
Ericsson Research, Jorvas, Finland
Pekka.Nikander@ericsson.com
Gonzalo.Camarillo@ericsson.com

Abstract

The Stream Control Transmission Protocol (SCTP) is a reliable message-based transport protocol developed by the IETF that could replace TCP in some applications. SCTP allows endpoints to have multiple IP addresses for the purposes of fault tolerance. There is on-going work to extend the SCTP multihoming functions to support dynamic addressing and endpoint mobility. This paper explains how the multihoming and mobility features can be exploited for denial-of-service attacks, connection hijacking, and packet flooding. We propose implementation guidelines for SCTP and changes to the mobility extensions that prevent most of the attacks. The same lessons apply to multihomed TCP variants and other transport-layer protocols that incorporate some flavor of dynamic addressing.

1. Introduction

In this paper, we discuss the effects of mobility and multihoming on the threat models and security mechanisms used in designing transport-layer protocols. We focus on the Stream Control Transmission Protocol (SCTP) because it incorporates multihoming support and there are proposals for extending the protocol towards transport-layer mobility. Besides SCTP, the lessons from our security analysis apply to all transport-layer protocols that support multihoming or mobility with end-to-end signaling.

Security is not the first goal we tend to associate with the transport layer. Nevertheless, transport protocols incorporate many features that have been designed with security in mind. The purpose of these features is to prevent spoofing of data and hijacking of connections and to mitigate denial-of-service threats. The best-known example is the TCP sequence numbers, which are initialized to an unpredictable random value in order to make packet spoofing more difficult. Another example is the congestion control and acknowledgement mechanisms that limit the number of packets sent into the network. Although these features were not originally introduced to prevent malicious behavior, their significance to security is now widely acknowledged. New transport protocols, such as SCTP, make security an explicit design objective.

When a mobile Internet host changes its location and its point of access to the internet changes, its IP address typically changes. The aim of mobility protocols is to solve the following two problems: to enable continuous communication over address changes, and to provide a reachability mechanism whenever the mobile is connected to the Internet. Mobility solutions exist for all major protocol layers. Link-layer mobility protocols avoid IP address changes. Network-layer protocols (e.g., Mobile IP) hide them from the layers above. Transport-layer mobility protocols – the topic of this paper – maintain a continuous connection between two endpoints over address changes. Higher, session and application-layer solutions re-establish transport-layer connections after an address change. All these solutions have their advantages and disadvantages. Transport-oriented approaches to mobility and end-to-end security are in some ways natural because this is the first layer in the stack where we can differentiate communication endpoints from addresses. Moreover, the transport-layer controls data flows and, thus, is instrumental in preventing some packet-flooding attacks.

A multihomed Internet host has multiple IP addresses. While the goal of mobility protocols is to enable communication for moving hosts, the aim of multihoming is typically to increase reliability in a static setting. When one address fails, communication is switched to another one. However, despite their different goals, mobility and multihoming can be seen as two flavors of the same phenomenon: dynamic multi-addressing. That is, a multihomed or mobile endpoint has a set of IP addresses that changes dynamically. In this paper, we are interested in the whole spectrum of such behavior.

Multihoming and mobility affect the security of transport protocols in several ways. First, existing security mechanisms are often based on implicit assumptions of a

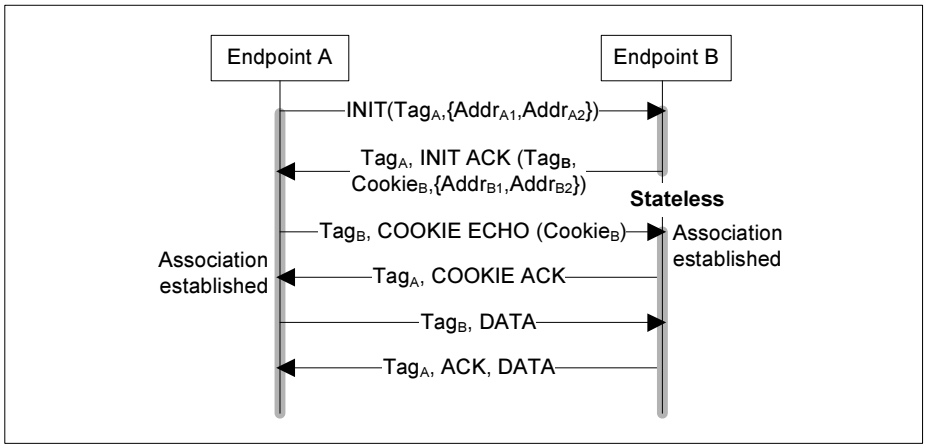

Figure 1: SCTP handshake

static network topology and unchanging addresses. When the assumptions are invalidated, the existing security mechanisms may become ineffective. Second, it is possible to misuse mobility signaling. Potential attacks include denial of service by preventing legitimate communication, connection hijacking, spoofing and intercepting data, and redirecting packet flows to the target of a flooding attack. We found such security issues in SCTP and in several other transport-layer protocols that support multihoming or mobility. Fortunately, it turns out that the vulnerabilities can be remedied with relatively small changes to the transport-protocol specifications and implementations.

In addition to reviewing protocol specifications for vulnerabilities, we looked at three open-source SCTP implementations to check that the attack scenarios described in this paper are realistic. It should be noted that the paper describes the specifications and implementations as they were at the time of writing. The SCTP implementation guidelines and major implementations have since been updated to reflect many of our discoveries and we believe that the remaining issues can be solved satisfactorily.

We only briefly mention strong end-to-end protection of user data in this paper. SCTP and TCP are both vulnerable to man-in-the-middle attacks where the attacker is on the path between the endpoints. The security mechanisms commonly used in transport protocols are relatively weak, similar to the random initial sequence numbers in TCP. In general-purpose transport-layer protocols, such weak mechanisms are preferable to expensive cryptographic operations and reliance on a security infrastructure. The focus of this paper is on the changes that are needed to the weak security mechanisms when the transport protocols support multihoming and mobility.

The rest of the paper is organized as follows. We first describe SCTP in Section 2. Section 3 discusses security-critical assumptions that are invalidated or weakened by the introduction of general-purpose multihoming and mobility. Section 4 covers the details of some interesting attacks that arise from the invalid assumptions. Section 5 suggests low-cost modifications to the SCTP protocol and Section 6 mentions some implementations. In Section 7, we briefly examine similar security issues in other transport-layer protocols that support dynamic addressing. Section 8 surveys related work and Section 9 concludes the paper.

2. SCTP protocol

The Stream Control Transmission Protocol (SCTP) [23] is a standard transport-layer protocol for the IPv4 and IPv6 Internet. SCTP was originally intended for the transport of PSTN telephony signaling messages over IP but it is now specified as a general-purpose alternative to TCP and UDP. The general applicability implies that any security mechanisms in the protocol will have to work correctly in a much wider range of settings than just telephony signaling.

The designers of SCTP have carefully considered past lessons on transport-protocol security. For example, the security function of TCP sequence numbers has been factored into a separate mechanism and the respondent remains stateless during the handshake in order to prevent state-exhaustion attacks. However, the design of Internet mobility protocols has brought attention to some new types of denial-of-service attacks that were not known or fully appreciated during the SCTP design process. It is important to prevent these attacks because, in some cases, not only the SCTP endpoints but also third parties may be exposed to denial-of-service.

This section describes SCTP with focus on features that are relevant to the following discussion on its security. For a complete description of SCTP, we refer the reader either to the protocol specification [23] or to the book by Stewart and Xie [22].

2.1. Protocol basics

An SCTP *association* is a relationship between two SCTP endpoints. An *endpoint* is a set of transport addresses and a *transport address* consists of a network-layer address and a port number. In SCTP, all transport addresses of an endpoint must share the same port number. Thus, in practice, an SCTP endpoint is identified with a non-empty set of IP addresses and a single port number. For the purposes of this paper, a pair of transport addresses is called a *path*. Each transport address can belong to only one endpoint at a time. This means that no special endpoint identifiers are needed. The receiver of an SCTP packet identifies the source and destination endpoints and the association to which the packet belongs based on the source and destination IP addresses and port numbers.

An SCTP packet comprises a common header and zero or more *chunks*. The chunks may carry either SCTP signaling information or user data (DATA chunk). Multiple chunks, such as data and acknowledgements, may be bundled into one packet. SCTP provides ordered and reliable multi-stream transport but this is largely irrelevant to our discussion and we skip the details.

Figure 1 shows the messages sent during a typical SCTP association setup, which is a 4-way handshake. The messages in the figure are identified by the names of the signaling chunks that they contain. Although not shown in the figure, user data can be bundled into the third and fourth messages of the handshake, thus saving one round trip.

2.2. Verification tags

In the two first messages of the handshake, the endpoints exchange random 32-bit nonces. The header of all but the first packet from Endpoint A to B must include B's nonce. Correspondingly, B must include A's nonce in the header of all packet that it sends to A. The SCTP specification calls the nonces *verification tags* (denoted by Tag_A and Tag_B in Figure 1.)

The verification tags serve the same security purpose as the randomly initialized sequence numbers in TCP. That is, they provide a level of security against packet spoofing. In SCTP, the security function and the sequence numbers have been factored into separate mechanisms. Obviously, the verification tags are not a very strong security mechanism. Any node that sees packets belonging to the association learns the verification tag values and can consequently spoof packets for that association.

2.3. Multi-homing and failover

During the SCTP handshake, each of the two endpoints may send to the other a list of IP addresses (in the INIT and INIT ACK chunks). Each endpoint selects one of the peer's addresses as the *primary destination address*, and one of it owns addresses as the best source address for routing packets to the destination. If the choice is not mandated by the upper-layer protocol, the algorithm for choosing the destination address is implementation dependent. The typical choice is either the source address of the first received packet or the first address in the peer's list.

Each endpoint sends all packets from the chosen source address to the primary destination address. The other addresses are used only if the primary path fails, i.e., if the primary destination address becomes unreachable. The policy for selecting the new address pair in failover is implementation dependent.

Each endpoint monitors the reachability of the secondary addresses of its peer so that it always knows which addresses are available for the failover. The monitoring is done by sending a heartbeat request (a HEARTBEAT chunk) to an idle destination address, which the peer acknowledges. The default frequency for the heartbeats is every 30 seconds. The implementation may start sending heartbeat requests immediately after the association has been established but it is not required to do so.

The heartbeat request contains a field for sender-specific information, which the peer will copy verbatim to the heartbeat acknowledgement. The exact contents of this field are not specified but the SCTP specification suggests including the destination address and a timestamp. The idea is that the requester can process the acknowledgements without storing the details of each individual heartbeat request it has sent.

2.4. State cookie

An important feature of the SCTP handshake is that the respondent (Endpoint B) remains stateless between sending the second and receiving the third message. The respondent encodes the protocol state, including the contents of the INIT, into a *state cookie,* which it sends to the initiator (Endpoint A) in the INIT ACK. The initiator returns the cookie to the respondent in the COOKIE ECHO. This prevents state-exhaustion attacks similar to the TCP SYN flooding [18].

The SCTP specification gives some guidelines on what data should go into the state cookie but the exact format of the cookie depends on the respondent implementation. The respondent must protect the integrity of the cookie with a message authentication code (MAC). No key distribution is needed because the respondent both creates and verifies the MAC with a local secret key that it never reveals to anyone else. The cookie also contains a timestamp to limit replays.

If an endpoint receives an INIT chunk from a peer with which it already has an association state, it needs to distinguish between several possible causes. The verification tags have a special role in accomplishing that. The two endpoints may have initiated the association simultaneously, thus causing a handshake collision, or the peer may have been restarted and is trying to re-establish the association. Moreover, messages can be duplicated, reordered, delayed, or lost. In order to determine which of the various complex scenarios occurred, the endpoint that responds to the unexpected INIT includes the verification tags of the existing association in the state cookie, where they are called *tie tags*. This helps it to decide how to proceed after receiving the third message.

2.5. Acknowledgements

SCTP acknowledgements combine both selective and cumulative information but we only need to consider the latter. The acknowledgements contain a cumulative sequence number for the received data in which no gaps remain. Like TCP, an SCTP endpoint maintains congestion windows that limit the amount of unacknowledged data that may be in flight at a time. There is a separate window for each transport address of the peer endpoint. The window size is calculated with a TCP-like algorithm that includes slowstart and congestion-avoidance phases, and is limited by the receiver's advertised buffer space.

An interesting piece of information for our discussion is that the receiver can, with one cumulative acknowledgement, acknowledge all the outstanding packets in the congestion window. Thus, a receiver that is willing to break the protocol rules can maintain a data flow by sending only one cumulative acknowledgement for each congestion-window-size block of data. On the other hand, SCTP protects against the ACK-splitting and optimistic-ACKing attacks outlined by Savage et al. [17]. The receiver needs to acknowledge individual packets in order to accelerate the sending rate.

2.6. ABORT and ICMP error messages

The SCTP specification defines the ABORT chunk for closing an association in an error situation. For example, an endpoint sends an ABORT when it receives an out-of-the-blue packet, i.e., one that does not match any existing association. This causes the receiver of the ABORT to delete its association state.

It is clearly important to prevent the spoofing of ABORT messages but the end endpoint that receives an out-of-the-blue packet has no matching association state and does not know what the peer's verification tag is supposed to be. SCTP solves this dilemma by using the verification tag from the out-of-the-blue packet instead. The tag proves that the sender of the ABORT has at least seen a packet that belongs to the association.

The abort mechanism is, of course, used only by nodes that support the SCTP protocol. When a non-SCTP node receives an out-of-the-blue SCTP packet, it either sends an Internet Control Message Protocol (ICMP) error message or it silently discards the packet. The possible ICMP messages are "Destination unreachable" and "Unknown next header type" (the latter in IPv6). Spoofing of ICMP error messages is prevented in way similar to the ABORT. The sender of an ICMP message copies some bytes from the beginning of the violating packet, including the verification tag, to the error message. The SCTP specification does not say how an endpoint should react to a received ICMP error message.

2.7. Dynamic address reconfiguration

Standard SCTP supports multihoming with a static set of addresses. We will now look at a proposed SCTP extension, *dynamic address reconfiguration* [21], which enables dynamic multi-addressing. It is worth remembering that this extension is work in progress and the correct forum for discussing its details is the relevant IETF working group. The proposal was originally intended for infrequent address changes, such as network renumbering, but there have been many ideas on using it as a mobility mechanism [7][9][26]. In this paper, we will focus on the general lessons that can be learned about the security of transport-layer mobility.

The proposal defines a new SCTP chunk type ASCONF. An endpoint uses the ASCONF chunk to notify its peer about changes to its address set. The chunk contains one or more instructions for adding and deleting addresses and for setting the primary address. The recipient executes these instructions in the order in which they appear in the chunk. It is easy to see how such instructions can be used to implement a location update: add the new address, set it as the primary address, and delete the old address. (See the right-hand side of Figure 3 for an example.)

3. Assumptions invalidated by mobility

In this section, we consider several implicit assumptions that were made in the design of the SCTP protocol. Most of these assumptions are entirely reasonable if the protocol is used for its original purpose, i.e., for telephony signaling. We show that they are not necessarily true when SCTP is used as a general-purpose transport protocol for multihomed or mobile endpoints.

3.1. Static vs. dynamic network topology

As we already mentioned, if an attacker sees SCTP packets belonging to an association and learns the verification tags, it can spoof packets for the same association. To be able to spoof packets in both directions, the attacker needs to see one packet in each direction (unless it sees the INIT ACK). This weakness is usually considered acceptable because the set of nodes on a path between two transport addresses and, consequently, the number of potential attackers that can see the tags is relatively small.

Multihomed endpoints, however, have several possible paths between them, which may all be tested when recovering from a path failure. The SCTP specification encourages trying all different source-destination address pairs. If this is implemented, the number of potential paths grows quadratically with the number of addresses per endpoint. Thus, the number of network nodes that can sniff the verification tags may increase rapidly with the number of secondary addresses and with the frequency of communications or endpoint failures.

The situation becomes even worse if the endpoints are mobile and use dynamic address reconfiguration. Anyone along any past path between the endpoints may have seen the verification tags. Moreover, mobile endpoints are often located in wireless networks, which due to their broadcast nature expose the packets to sniffing. Thus, the assumption that the tags will only be seen by a small fixed set of nodes no longer holds. On the contrary, the longer the association lifetime and the more often the mobile moves, the larger group of nodes that have an opportunity to sniff the tags.

The general lesson on transport-layer security is that plaintext secrets (verification tags, secret sequence numbers, nonces, etc.) that can be used in more than one message are more vulnerable with multihomed or mobile endpoints than in a static setting. They may still be acceptable as security mechanisms if the rate of change is slow and connection lifetimes limited.

3.2. Faith of old addresses

Another assumption made in the standard SCTP protocol is that the transport addresses belong to the association endpoints until the end of the association lifetime. The danger of this assumption is that if the endpoint loses control of an address and the address is subsequently allocated to another node, the peer will continue to send packets (such as heartbeat requests) to the lost address. The new owner of the address will receive the packets and learn the verification tags in them. If the new owner of the address is malicious, it may use this information for spoofing attacks.

In the currently typical SCTP applications, such as telephony signaling, the risk of attacks caused by this vulnerability is small. Multihomed SCTP endpoints usually do their best to ensure that they retain all their addresses throughout the association lifetime, for example, by using only statically configured IP addresses. On the other hand, when the dynamic-address-reconfiguration mechanism is used for mobility, it becomes normal for an endpoint to leave an address. While the mobile endpoint will send delete-address instructions to its peers, some packets will usually be in flight when the address change occurs and may end up at a new owner of the address.

The general lesson for mobility-protocol design is that we should consider what happens to the old addresses of the mobile node, and the possibility that an attacker gains control of one.

3.3. Who ate the cookie

There is one peculiar feature in the SCTP protocol that makes it particularly dangerous for an endpoint to lose an IP address to an attacker. One might think that the attacker cannot do anything because it does not know the verification tags for the association, but that is not true. Recall that the state cookie in the second message of the handshake (INIT ACK) contains the verification tags of any existing association between the same endpoints. The attacker can use the state-cookie mechanism as an oracle to discover both verification tags of the association whose address it controls.

The attacker starts by sending an INIT to the peer endpoint using the same IP address and port number as in the existing association. (It does not need to list any secondary addresses.) The peer automatically responds with an INIT-ACK that contains a state cookie. Inside the state cookie, the attacker finds both verification tags. The location of the tags in the cookie depends on the peer-endpoint implementation but it is not hard to guess.

The worst part is that the attacker usually does not need to know the correct port number, the peer's address, or

even that the association exists. This is because the arrival of any packet belonging to the association (such as a heartbeat request) alerts the new address owner to the opportunity for misuse. A single packet also provides all the necessary information for the above attack.

The conclusion we draw from this section is that it is dangerous to trust secrets to a peer that has just exhibited symptoms of a failure, even though it may be tempting to do so in the hope of a quick recovery.

3.4. Correctness of the address sets

In a telephone signaling system, it is reasonable to assume that the endpoints know their own addresses and even each other's address sets. They can also be trusted to give each other correct information about their secondary addresses in the handshake messages. The heartbeat mechanisms is used to monitor the peer addresses but its purpose is to discover which addresses are currently reachable, not to cast any doubt on whether the addresses were correct to begin with.

In a general-purpose multihoming protocol, on the other hand, we need to prepare for the possibility that the endpoints sometimes make mistakes about their address sets and even purposely misrepresent them. Moreover, we cannot expect there to be any application-level verification or recovery mechanism that would mitigate the consequences of such false information. Thus, when SCTP is used as a general-purpose transport protocol, we need to worry about attacks where the endpoints lie about their addresses.

This concern about the correctness of the peer's addresses is not specific to SCTP. In any multihoming or mobility protocol, we need to consider the possibility of an endpoint making false claims about its addresses.

3.5. Address ownership

Two SCTP endpoints are not supposed to share the same transport address, i.e., the same IP address and port number. The original SCTP specification does not define how an endpoint should react if two of its peers have a common transport address. However, there is an implementation guideline [20] for rejecting INIT packets that do not match any existing association but contain an address that is already used in one. This prevents situations where two endpoints share a transport address.

To be precise, the guideline says that if an endpoint receives an INIT that otherwise matches an existing association but has new addresses added to it, the endpoint receiving the INIT will respond with an ABORT, instead of the usual INIT ACK. The reason for issuing this guideline was that early SCTP implementations were vulnerable to an address-injection attack: when they received an INIT chunk that matched an existing association but contained an additional secondary address, the implementations processed the INIT as an indication of a peer restart and added the new address to the re-established association.

Following the guideline effectively means that the first peer to start using a transport address as either a primary or as a secondary address gets the priority to it. Any later claims by others to the same address are treated as errors.

An address conflict in a telephony signaling protocol is probably an operator error and it is best to report the error and let the operator resolve it. The situation changes when we consider a general-purpose transport protocol. There may not be any operator that could help. Thus, it is essential to either avoid the conflicts or to resolve them automatically in a fair way.

3.6. Value of spoofing a single packet

SCTP prevents packet spoofing with the 32-bit verification tags. Nodes that want to spoof packets for an association must either be on the path between the association endpoints or guess the verification tag value. It is obvious that the 32 secret bits are not sufficient to prevent an off-path attacker from occasionally guessing the right value and succeeding to spoof a single packet. If the attacker knows the right port numbers and sends 2^{32} packets to the attack target, one of them will have the correct verification tag and will be accepted.

The SCTP designers have reasonably decided that such a brute-force attack is not a major threat. First, it requires huge effort on the attacker's part to spoof a packet. Second, the benefit is limited to spoofing a single packet. In order to spoof a second packet, the attacker needs to repeat the effort. Probably the best option for the attacker is to send ABORT chunks and try to terminate the association for denial of service. Experience from the TCP protocol, where Reset packets can be similarly spoofed, shows that this kind of attacks are not a major threat in practice.

Dynamic address reconfiguration changes the situation considerably. If the attacker manages to spoof an ASCONF chunk, it can add its own address to the peer endpoint and set it as the primary address. While the attack still requires in the order of 2^{32} packets (approximately 200 gigabytes in IPv4) to succeed, the reward is the complete control over the association. Thus, the brute-force attack has become much more attractive. With the current network technology, the attack is feasible only if the attacker has a particularly high-bandwidth connection to the target but this may change as the Internet bandwidth grows. Fortunately, the authors of [21] have been aware of the dangers and are working on a stronger ASCONF authentication mechanism.

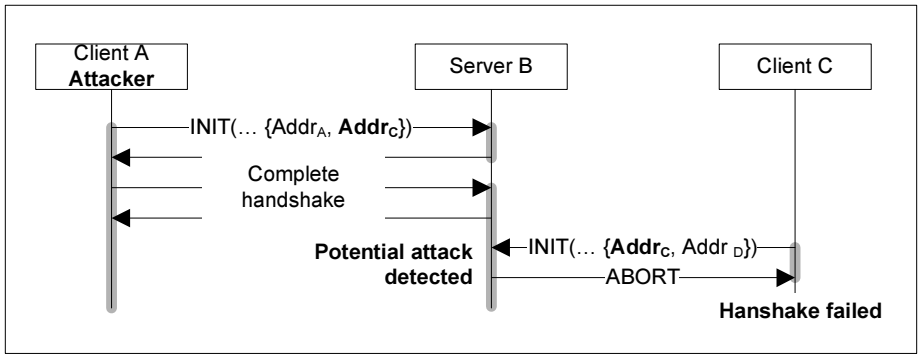

Figure 2: Address squatting

In any protocol with dynamic addressing, it is important to consider the possibility of spoofed signaling messages as well as spoofed data. The cost-benefit trade-offs may be quite different for different types of messages and the addition of new signals may render existing protection insufficient.

3.7. Importance of error messages

The SCTP specification defines clear rules for sending an ABORT in response to out-of-the-blue packets and for processing the ABORT. Non-SCTP nodes should respond to an out-of-the-blue SCTP packet by sending an ICMP error message, either "Destination unreachable – protocol unreachable" in IPv4 or "Parameter problem - unrecognized next header type" in IPv6. Like ABORT, these error messages indicate that the recipient did not want to receive the packet. Firewalls and routers may also respond with ICMP "Destination unreachable", which sometimes signals unwillingness to receive SCTP packets. It is surprising that the SCTP specification does not mention the issue of processing ICMP error messages.

Implementations at the time of writing this paper were not much better. Two of the implementations that we reviewed (see Section 6) ignore all the critical ICMP messages and one processes only some of them. This is probably an attitude inherited from TCP implementations, which often treat ICMP messages in the same way.

The difference between TCP and SCTP is, however, that multihoming makes the error messages much more important. It can, for example, happen that an endpoint loses control of a secondary address and that a new owner of the address wants to stop packets being sent to it. With the dynamic address reconfiguration and mobile endpoints, it is even more likely that packets end up being sent to a wrong address. If the sender of the offending data takes no notice of the error messages, the recipient has no way of defending itself. The same, of course, applies to any transport protocol with multihoming and mobility support.

4. Redirection attacks

This section describes complete attack scenarios against SCTP, which all depend on one or more of the protocol weaknesses discovered in Section 3. The attacker can prevent communication between honest endpoints by squatting their addresses, hijack an association without being on the route between the endpoints, trick a server to flood a target address with data, and forward associations in an unexpected way. Obviously, the more full-featured the mobility protocol is, the more flexibility the attacker has. Nevertheless, we will see that even static multihoming may make a transport protocol vulnerable to some attacks.

4.1. Address squatting

The way SCTP resolves address conflicts, i.e., situations where two endpoints share an address, can be exploited in a denial-of-service attack. That is, an attacker can "squat" addresses. Consider the scenario in Figure 2. Two honest endpoints B and C try to communicate while the attacker A wants to prevent them. If the attacker knows the transport address of C, it can create an association with B using the same port number and include one of C's addresses as a secondary address in its own address list. When C tries to create an association with B, there will be an address conflict between the two associations (A-B and C-B). B will reject the INIT from C and respond with an ABORT. Thus, C will fail to establish an association with B.

In order for the above attack to succeed, the attacker needs to know or guess which port number C will use when connecting to B. This attack works particularly well against applications like telephony signaling where the

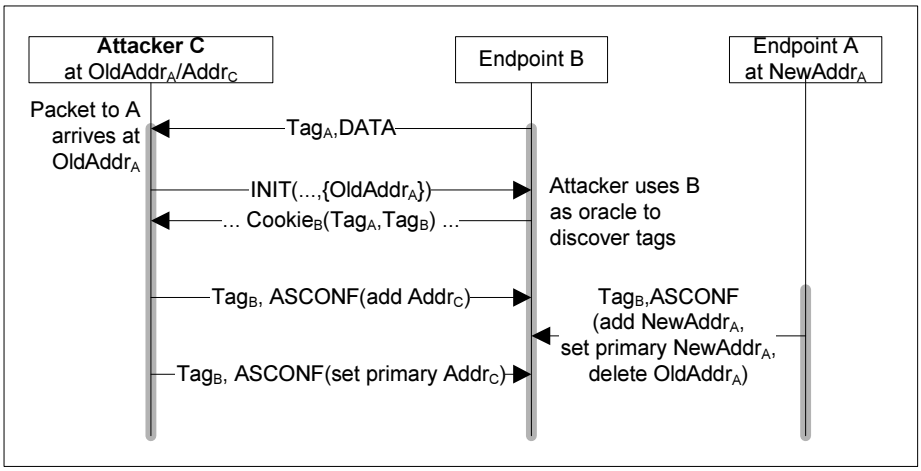

Figure 3: Association hijacking

port numbers for both endpoints of an association are often fixed and well known. In other applications, the attack is slightly more difficult to execute because the attacker will either have to squat all 2^{16} port numbers or it has to guess which numbers are more likely. The guessing is not as difficult as it may appear because current implementations typically allocate port numbers sequentially. The effectiveness of the attack also depends on the timing of heartbeat requests, which varies between implementations. If B sends a heartbeat request to the squatted address, it will receive an ABORT from A, which will terminate the squatter's association.

The address-squatting attack is perhaps the most serious threat that we discovered against the standard SCTP protocol when used for its original application. It could lead to serious denial-of-service issues when SCTP is used for transporting telephony signaling over the public Internet.

4.2. Association hijacking

An attacker that is permanently on the route between the two endpoints can mount a man-in-the-middle attack and hijack the entire association. Like the SCTP designers, we accept that unless a strong end-to-end security mechanism is used, we cannot solve this problem. Instead, we are interested in stopping attackers that only temporarily or accidentally see packets belonging to the association.

In Sections 3.1-3.3, we explained several ways in which mobility and a carelessly designed state-cookie mechanism make it more likely that an attacker learns the verification tag values. In this section, we will consider the consequences of such a compromise if the endpoints support dynamic address reconfiguration.

Figure 3 shows a worst-case scenario that combines several of the vulnerabilities discussed in Section 3. An attacker C is the new owner of an IP address from which a mobile endpoint A has just moved away. The attacker also has a second IP address $Addr_C$. The attack is triggered when the attacker receives a packet that belongs to the A-B association. The attacker uses B and the state cookie mechanism as an oracle to discover the verification tags of the A-B association. The attacker then sends an ASCONF chunk to B in order to add its other address $Addr_C$ to the association. The ASCONF from the attacker arrives at B before the one from the mobile endpoint A does so that B still accepts packets from A's old address. Later, the attacker sets this address as the primary address. As a result, the attacker has managed to hijack the association from the mobile endpoint A.

Admittedly, this attack depends on a number of assumptions and on unfortunate timing of the events. Experience from security protocol design shows, however, that attacks that initially appear theoretical may, in the right circumstances, become practical.

4.3. Bombing attack

The idea in the bombing attack is that the attacker redirects a data flow to a target node in order to flood it with packets. In order for the attack to make sense, the attacker must get someone else to send the packets, and the number of packets arriving at the target must be larger than the number of packets sent by the attacker. Even a relatively small amplification factor can have serious consequences if it is used in connection with a distributed denial-of-service attack. In this section, we explain how SCTP multihoming and mobility can be exploited in such

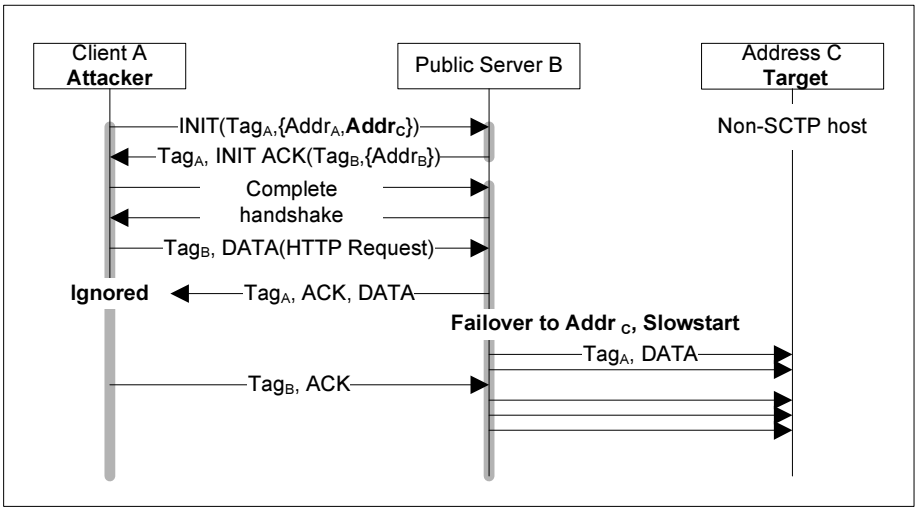

Figure 4: Bombing attack exploiting static multihoming

attacks if SCTP is used as a general-purpose transport protocol.

As shown in Figure 4, an attacker A starts by creating an association with a public server B, such as a web server that supports SCTP. In its own list of addresses in the INIT chunk, the attacker includes both its own IP address and the IP address of the target C. The attacker then starts downloading a data stream, such as a large data file, from the server. Immediately thereafter, the attacker stops sending any packets, including acknowledgements. This causes the server to switch to using the target address as the primary address. As a result, the data flow from the server will be redirected to the target. Note that the failover needs to happen before the server sends the first heartbeat request to the target address.

The next step is to spoof acknowledgements from the target C to the server B. The attacker can do this because it knows the verification tag values. In order to accelerate the rate at which the server sends data to the target, the attacker should spoof an acknowledgement for each data packet. After the initial acceleration, the attacker only needs to send one acknowledgement per congestion window in order to sustain the data flow. The attacker will have to guess or measure the maximum sustainable rate at which the server can send data so that it knows when to stop accelerating the data flow and does not get ahead in the acknowledgements.

As described above, the bombing attack works with standard SCTP multihoming. The same attack can be executed more effectively with the dynamic address reconfiguration functions. The attacker does not need to wait for the server to switch to the target address after a timeout. Instead, it can immediately set the target address as the primary address.

There is one major limitation to the above attack. As explained in Section 2.6, a target that supports the SCTP protocol will respond with an ABORT and the server will stop sending data. This means that the target has to be a non-SCTP node or a non-existent address, in which case the real target is the router towards the network of that address. The non-SCTP nodes and routers may respond with ICMP error messages but the current SCTP implementations typically ignore them.

The bombing attack is attractive for the attacker because it can direct a large data flow from the server to the target while only sending and receiving a few messages itself. The attacker does not even need to spoof the source IP address of the acknowledgements as it is legal for them to come from a secondary address. The attack is more damaging if several data streams are redirected to the same host or router.

4.4. Association forwarding

This section describes an unexpected feature of the SCTP multihoming: association forwarding. Invoking this feature does not constitute an attack in itself but we conjuncture that, if not well understood by application designers, it could be exploited as a building block in application-level attacks.

Like in any other transport protocol, a node can act as a proxy between two SCTP endpoints. The proxy acts as a server for a client and as a client for a server and forwards upper-layer data between the two. Assume, as in Figure 5, that endpoint A initiates a connection with endpoint B. B can (with or without A's knowing) open an association to another server C and act as a proxy between A and C.

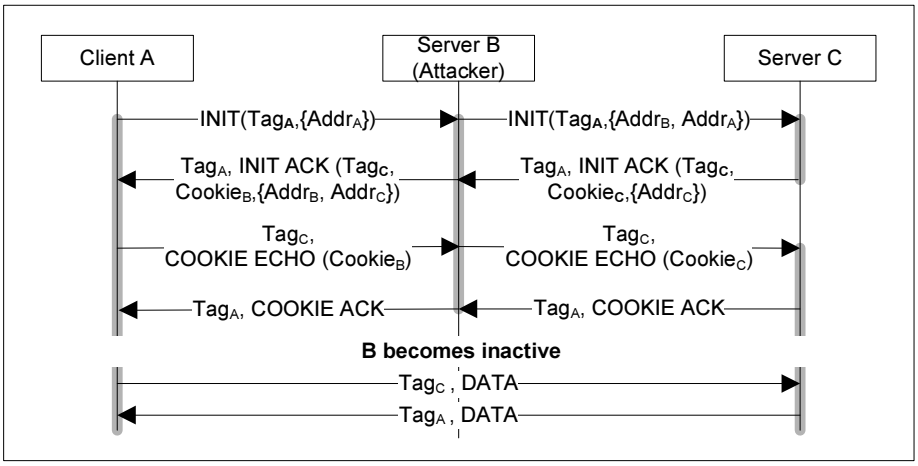

Figure 5: Association forwarding

Whether this behavior is desirable depends on the application requirements.

In SCTP, the proxy can go one step further: it can remove itself from the middle and let A communicate directly with C. B uses two tricks to achieve this. The first trick is to use the same port numbers and verification tags for both associations (A-B and B-C). B waits for connections at the same well-known port number as C. When B receives an INIT from A, B copies A's port number and tag to its own INIT to C. After receiving the INIT ACK from C, B copies C's tag to its own INIT ACK to A. As a result, both associations use the same pair of port numbers and the same pair of verification tags. The second trick is to switch over to the direct path A-C. In its INIT chunk to C, B includes A's transport address as a secondary address. In the INIT ACK to A, B includes C's transport address as a secondary address. After the handshake, B stops responding to A's and C's packets. This causes A and C to switch to communicating directly. B will not receive any further packets, except occasional heartbeat requests, which it can ignore. There is also a variation of this scenario where the server connects two simultaneously connecting clients to each other and removes itself from the middle. The dynamic address reconfiguration functions would, of course, make the switching to direct communication faster.

It is not clear how serious this kind of association forwarding is. On one hand, it is unexpected and potentially unwanted behavior. The attacker could, for example, continue to insert occasional application-level messages into the association in order to trick A and C into communicating unnecessarily. On the other hand, association forwarding does not necessarily violate any well-defined security policy. It could even be used for optimizing proxy implementations and for implementing connection-establishment functionality. Thus, it is probably best to consider association forwarding a protocol feature that application designers need to know about.

5. Solutions

This section suggests a variety of remedies to the security issues identified earlier in the paper. We hope that by this time at least some of the solutions have become apparent to the reader. Each suggested solution fixes a major vulnerability and, thus, prevents a large class of attacks. Therefore, we suggest implementing as many of them as one can without breaking compatibility with the current protocol specification. The designers of protocol extensions and future transport protocols should, of course, take all these lessons into account.

5.1. Protecting old addresses

In IPv6, it is possible to prevent an attacker from gaining the ownership of the old address of an honest endpoint. The Secure Neighbor Discovery (SEND) protocol [1] uses cryptographically generated addresses (CGA) [2], i.e., IPv6 addresses where some address bits encode a secure hash of the address owner's public signature key. This makes it possible for anyone to verify the binding between the public key and the address without any security infrastructure. All address resolution messages on the local link are signed, which prevents the attacker from using some else's address.

Since the supply of IPv6 addresses is essentially unlimited, it is not necessary to reuse them. An access router that supports SEND will refuse to route packets to a node that tries to use someone else's old address. For the details, we refer the reader to [1].

In IPv4, there is no such elegant solution available. The best the access network can do is to avoid recycling addresses sooner than is absolutely necessary.

5.2. Preventing address squatting

There are several solutions for the address squatting attacks. The first possibility is to assign port numbers at the client end randomly rather than sequentially as in the current implementations. Random port numbers make it impossible for the attacker to guess which port number it should squat, unless it repeats the attack for all 2^{16} of them. Randomizing the port numbers requires only a minor modification to the implementations.

Another potential solution would be to relax the definitions of an endpoint and association so that multiple endpoints can use the same address. An endpoint could allow two of its peers to share a transport address and identify the association of a received packet based on not only the source and destination transport addresses but also the verification tag. The changes required to implementations would be rather complex but, like random port numbers, the changes could be deployed locally at each endpoint without breaking compatibility with existing implementations.

Peer address verification, which will be explained in Section 5.5, can also prevent address squatting.

5.3. Encrypting cookies

As is obvious from our earlier discussion, the state cookie should not be transferred in plaintext. In order to prevent attackers from using the cookie mechanism as an oracle to discover the verification tag values of an existing association, the cookie should be encrypted. This can be done conveniently with the same local key that is used for computing the MAC on the cookie.

Encryption is, however, a relatively expensive cryptographic operation and it may not be suitable for all nodes. One alternative is to compute one-way hashes of the tie tags and to send them in the cookie instead of the plaintext values. Later, when the tie tag values are compared with other tags, the comparison can just as easily be done with hash values.

An even simpler solution, designed by Stewart and Tüxen in reaction to an early version of this paper, is to replace the tie-tags with nonces that are sent in the cookie and stored in the existing TCB by the server.

5.4. Secure acknowledgements

SCTP acknowledgements, like all SCTP messages, can be spoofed if the attacker knows the recipient's verification tag. The argument is that the verification tags are a reasonable trade-off between security and cost. However, it is possible to improve the robustness of the acknowledgements without resorting to expensive cryptography. The idea is to ensure that whoever sends an acknowledgement has really seen the data.

In the secure acknowledgement scheme proposed by Savage at al. [17], every packet contains a nonce (i.e., an unpredictable random or pseudorandom number). The acknowledgement contains the sum (or exclusive or) of the nonces from all the received data packets. This idea is particularly easy to implement for cumulative acknowledgements but it is fairly efficient for selective ones as well. In order to spoof such acknowledgements, one needs to see every data packet. This would practically stop the bombing attacks because the attacker could not spoof the acknowledgements. Moreover, by requiring every SCTP packet to contain a cumulative acknowledgement (as in TCP), we can extend some spoofing protection from the acknowledgements to other messages. Unfortunately, replacing the acknowledgement scheme would require a major revision of the SCTP protocol and is perhaps an unrealistic plan.

5.5. Peer address verification

The bombing attack is possible because an endpoint simply believes the list of addresses it receives from the peer. This insight inspires another potential solution: an endpoint should verify that the peer addresses are active and that the nodes in these addresses want to receive data for the particular association. It should initially flag all the peer addresses as unverified and only clear the flags after it has polled each address and received a positive response.

There are two ways to verify a peer address. First, the address that the peer uses during the handshake is automatically verified as soon as the peer returns the first packet with the correct verification tag. Second, the heartbeat protocol can be used to poll addresses. For this purpose, the heartbeats need to be implemented securely so that it is not possible to spoof acknowledgements. This can be achieved by including an unpredictable random or pseudorandom nonce in the sender-specific information of the heartbeat request.

As long as a peer address remains unverified, the endpoint should not send data to the address or assume that it really belongs to the peer endpoint. Instead, it should allow an unverified address to appear in multiple endpoints. Once the address has been verified for one endpoint, it is safe to enter the slowstart phase for that address and to assume that any other endpoints claiming to have the same address are either mistaken or lying.

If there are many peer addresses, verifying them may take a while because the SCTP specification allows

sending only one heartbeat at a time. If at any time there are no active verified addresses available and data is waiting in the queue to be sent, the address verification can be optimized by bundling the heartbeat request, including the nonce, with the first data to the new transport address. The acknowledgements for the heartbeat and for the data can also be bundled in one packet. The result is very much like the secure acknowledgement scheme discussed in the previous section, except that it is used only for the first packet to the destination.

The address verification is relatively straightforward to implement because the main component, the heartbeat mechanism for polling addresses, already exists. The hardest part is probably that the socket API will need to be modified to indicate the verification status of each address. At the time of writing, one implementation already included a nonce in the heartbeat messages. We also understand that there already was a plan to add an address verification mechanism, similar to the one described here, to the SCTP implementation guidelines.

5.6. ICMP processing

As an alternative to the address verification, the bombing attacks can be prevented by processing error messages correctly. For this to work, it is important that hosts, routers and firewalls send the ICMP error messages, which they currently do not always do. The failure to send the error messages makes them ideal targets for all kinds of packet flooding attacks. Non-SCTP hosts and firewalls might also consider implementing a minimal version of the protocol that sends ABORT chunks in response to all received SCTP packets.

The obvious way to process a received ICMP error message is to treat it as an ABORT and to delete the association state. However, error messages sent by routers and firewalls often do not provide enough information to justify terminating the association and the only possible reaction is to merely increment an error counter. Also, it can be argued that dropping the entire association is too drastic a measure when an endpoint receives an ICMP error message, or even an ABORT, that was triggered by what the recipient thought was an out-of-the-blue packet. A more reasonable reaction might be to delete the address from which the error message was received.

5.7. ASCONF authentication

Although not yet published, the authors of the dynamic address reconfiguration specification [21] are planning to use unauthenticated key exchange to establish a session key at the start of an association. This key will then be used to authenticate ASCONF chunks. The idea in this kind of leap-of-faith authentication is that, if the attacker is not present at the time of the key exchange, it cannot later spoof messages. Although not strong authentication in the traditional sense, the solution provides a higher level of security than plaintext verification tags and prevents ASCONF spoofing.

5.8. Strong end-to-end security

For strong end-to-end security in SCTP, the choice is currently between running either IPSec under the transport layer [4] or TLS on top of it [12]. The advantage of IPSec in comparison to TLS is that it protects also the signaling chunks and not only user data. Thus, it can prevent most of the attacks described in this paper. The verification of peer addresses is still necessary, though, because IPSec cannot prevent the bombing attack. A disadvantage of relying on IPSec to prevent the other attacks is that it limits the applicability of the protocol to situations where the endpoints can authenticate each other.

Both end-to-end security mechanisms need small modifications to work well with a transport protocol that supports multihoming. In IPSec, in order to avoid creating a separate security association between all pairs of addresses, it has been necessary to slightly modify the key exchange and the security association lookup. It is currently not clear how to cope with the dynamic address reconfiguration. TLS has also been adapted to work on top of SCTP.

6. SCTP Implementations

In order to check whether the vulnerabilities found in the SCTP specification exist in implementations, we reviewed the following three open-source implementations in October 2003:

- SCTP user-level library implementation by Andreas Jungmaier et al., version sctplib-1.0.0
- KAME snapshot of 2003-10-20
- Linux kernel implementation, on kernel version 2.5.67

Based on the review, we believe that all these implementations contained all of the vulnerabilities discussed in this paper with the following exceptions: KAME processed correctly the ICMP Host Unreachable and Network Unreachable messages but not ICMP messages from a reachable host that does not support SCTP. KAME also sent a nonce in the first heartbeat to each destination address.

At the time of preparation of the final version of this paper, most of the vulnerabilities in the KAME implementation have been fixed and we are working with the implementers on the remaining minor issues.

7. Other protocols

There are quite a few proposals for extending TCP to support multihoming and mobility. All the proposed TCP multihoming and mobility extensions take measures to prevent connection hijacking and data spoofing. These measures vary from unpredictable sequence numbers and nonces to strong cryptographic authentication. On the other hand, all the end-to-end TCP multihoming and mobility extensions that we have looked at are vulnerable to the bombing attack.

Multi-homed TCP by Huitema [11] unifies multihoming, mobility and network renumbering support. The receiver of a packet identifies the connection based on a context identifier that is sent in a TCP option, rather than based on the source and destination addresses. Outgoing packets are sent to one or more addresses from which data has recently been received for the same connection. Huitema already identifies the threat of connection hijacking. The bombing attack is also possible because, by spoofing source addresses, an endpoint can trick its peer into sending all future packets to an arbitrary address.

TCP-R by Funato et al. [10] implements mobility (but no multihoming) with TCP options. After a mobile host obtains a new address, it sends a redirect message to its correspondent. In the primary mode, the security is based on the TCP sequence numbers that must be in a certain range after the move. There is also a strongly authenticated mode. Nevertheless, there is nothing to stop an attacker from lying about its own location and spoofing acknowledgements in the bombing attack.

TCP Migrate by Snoeren and Balakrishnan [19] is similar to TCP-R. Elaborate cryptography is used to prevent connection hijacking but bombing attacks are again possible.

TCP Multi-Home Options by Matsumoto et al. [13] uses TCP options to implement full SCTP-like dynamic multi-addressing. Because of the similarity with SCTP, it suffers from all the vulnerabilities described in this paper and would benefit from the solutions we have proposed.

MAST by Crocker [8] is a proposal to isolate dynamic multi-addressing into a separate layer that can be inserted under any transport protocol. Again, similarity with SCTP means that essentially all our attacks and defenses are relevant to MAST.

It is our feeling that any protocol that supports multihoming or mobility with end-to-end signaling is potentially vulnerable to the bombing attack and should prevent it by verifying the peer addresses before using them. Mobility protocols based on directories and other infrastructure may also suffer from similar problems:

Dynamic DNS [24][25] can be used as a location register for slowly-moving mobile nodes. While bombing attacks that exploit Dynamic DNS have not been documented, it is, at least in theory, possible to redirect data to a target by setting a DNS entry to point to the target IP address.

8. Related work

The protocols discussed in this paper are end-to-end protocols in the sense of Saltzer et al. [16]. That is, both signaling and data are sent between end nodes and the only function of the network is to route packets between them. In the proposed security solutions, we have followed the same principle. The alternative would be to build the security and mobility functions into the network architecture, for example, in the style of cellular phone networks or overlay networks. While this probably could solve most of the security issues discussed in this paper, such a solution would not be consistent with the Internet design philosophy.

Mobile IPv6 is a network-layer mobility protocol that hides the address changes from the transport protocols. The bombing attack where the attacker gets a server to send a flow of packets to the target by lying about its own location was first recognized in Mobile IPv6 [3]. Similar to the bombing attack described in this paper, an attacker can start downloading a TCP stream and then trick the server into redirecting it to a target address. The attacker can also spoof TCP acknowledgements because it knows the initial sequence numbers.

Mobile IPv6 solves the flooding problem with a mechanism called *return routability*. The server sends a nonce to the mobile's new address to test whether the mobile really is there. This inspired our heartbeat-based solution for SCTP. The disadvantage of Mobile IPv6 is that since it is not a transport protocol, the nonce cannot be bundled with application data. Thus, our solution for SCTP can save one roundtrip in comparison to TCP over Mobile IPv6.

Using addresses as identifiers for multihomed and mobile endpoints creates in inconsistency that is at least partly responsible for some of the attacks described in this paper. Chiappa [6] proposed the decoupling of endpoint names and network addresses. This would probably be a good starting point if we wanted to design a new transport protocol.

The Host Identity Payload (HIP) protocol unifies mobility, multihoming and security into a new protocol layer between the network and transport layers [14]. One of the fundamental ideas of HIP is to define a separate namespace for endpoint identifiers and to map them securely onto network addresses. HIP now incorporates the return routability test for verifying peer addresses. Because of the emphasis on endpoints, one could argue that HIP is a transport-oriented protocol.

Protocol Weakness	Attack Scenario: Data spoofing with sniffed tag	Brute-force ASCONF spoofing	Address squatting (Fig.2)	Association hijacking (Fig.3)	Bombing attack (Fig.4)	Association forwarding (Fig.5)	Solution	Implement immediately
Spoofing a single packet possible		X					Leap-of-faith authentication for ASCONF	
Plaintext cookie				X			Encrypt cookies or replace tie-tags with nonces	X
Dynamic topology exposes tags	X				X		Secure acknowledgements	
Attacker is the new address owner				X			Secure neighbor discovery	
Unverified peer addresses			X		X		Peer address verification with secure heartbeats	X
ICMP error messages ignored					X		Implement ICMP error processing	X
Endpoints cannot share addresses			X				Randomize initiator port number or redefine endpoint	X
Proxying possible						X	Understand and use with care	

Table 1: Main weaknesses, attacks and solutions

The Real-Time Transport Protocol (RTP) is used for transmitting audio and video streams over the Internet. Rosenberg [15] explains how SIP and other session-establishment protocols can be used to direct RTP streams to a flooding target. If we abstract away the cumbersome details of the solution proposed in [15], the idea is similar to our peer-address verification. The proposal also suffers from the same connection-forwarding problem as SCTP.

We already mentioned the work of Savage et al. [17] on TCP acknowledgements. The attacks we are trying to prevent are slightly different; SCTP was already designed with the TCP ACK-splitting attack in mind. Nevertheless, the same secure acknowledgement scheme works well to prevent the spoofing of SCTP acknowledgements. With secure acknowledgements, we would not even need to send a heartbeat request to verify the peer address. The first secure acknowledgement would have the same effect as a secure heartbeat. SCTP, in fact, almost does the right thing because, unlike TCP mobility solutions, it maintains a different congestion window for each peer address.

9. Conclusion

Adding multihoming and mobility support to Internet transport protocols changes the environment in which transport-layer security mechanisms operate. This may cause non-cryptographic security mechanisms, such as TCP sequence numbers and SCTP verification tags, to break. The attacker may be able to spoof data and signaling messages and hijack connections. Dynamic multi-addressing also gives raise to new types of attacks such as address squatting, redirection of data from a server to the target of a bombing attack, and connection forwarding. In this paper, we describe a number of such attacks against SCTP and suggest low-cost changes to the protocol specification and implementations. Several SCTP implementations were found to be vulnerable to all or most of the attacks described in this paper. Table 1 summarizes the main protocol weaknesses, attacks and solutions and how they relate to each other. (Some attacks depend on multiple vulnerabilities, which is indicated by multiple crosses in the column. We have also marked the changes that should be made immediately to the implementations.) The lessons from our security analysis apply to other transport protocols and to practically any multihoming or mobility solution that uses end-to-end signaling.

Acknowledgements

We thank Randall Stewart, Michael Tüxen and Alf Zugenmaier for their extensive comments that were invaluable in preparing this paper.

References

[1] Jari Arkko, James Kempf, Bill Sommerfeld, Brian Zill, and Pekka Nikander. Secure neighbor discovery (SEND). Internet-Draft, IETF Securing Neighbor Discovery Working Group, February 2004. Work in progress.

[2] Tuomas Aura. Cryptographically generated addresses (CGA). In Proc. 6th Information Security Conference (ISC '03), LNCS 2851, pages 29-43, Bristol, UK, October 2003. Springer.

[3] Tuomas Aura, Michael Roe, and Jari Arkko. Security of Internet location management. In Proc. 18th Annual Computer Security Applications Conference, Las Vegas, NV USA, December 2002. IEEE Computer Society.

[4] Steven Bellovin, John Ioannidis, Angelos D. Keromytis, and Randall R. Stewart. On the use of stream control transmission protocol (SCTP) with IPSec. RFC 3554, IETF, July 2003.

[5] Scott Bradner, Allison Mankin, and Jeffrey I. Schiller. A framework for purpose-built keys (PBK). Internet-Draft, IETF, June 2003.

[6] J. Noel Chiappa. Endpoints and endpoint names: A proposed enhancement to the internet architecture. Expired Internet Draft, 1999. (Archived at http://www.watersprings.org/.)

[7] Phillip T. Conrad, Gerard J. Heinz, Armando L. Caro Jr., Paul D. Amer, and John Fiore. SCTP in battlefield networks. In Proc. MILCOM '01, Washington, DC USA, October 2001.

[8] Dave Crocker. Multiple address service for transport (MAST), an extended proposal. Internet-Draft, IETF, September 2003. Work in progress.

[9] Thomas Dreibholz, Andreas Jungmaier, and Michael Tüxen. A new scheme for IP-based Internet mobility. In Proc. 28th Annual IEEE Intl. Conference on Local Computer Networks (LCN '03), pages 99-108, Königswinter, Germany, October 2003. IEEE Computer Society.

[10] Daichi Funato, Kinuko Yasuda, and Hideyuki Tokuda. TCP-R: TCP mobility support for continuous operation. In International Conference on Network Protocols (ICNP '97), pages 229-236, Atlanta, GA USA, October 1997. IEEE Press.

[11] Christian Huitema. Multi-homed TCP. Expired Internet Draft, May 1995. (Archived at http://www.watersprings.org/.)

[12] Andreas Jungmaier, Eric Rescorla, and Michael Tuexen. Transport layer security over stream control transmission protocol. RFC 3436, IETF, December 2002.

[13] Arifumi Matsumoto, Masahiro Kozuka, Kenji Fujikawa, and Yasuo Okabe. TCP multi-home options. Internet-Draft, IETF, October 2003. Work in progress.

[14] Pekka Nikander, Jukka Ylitalo, and Jorma Wall. Integrating security, mobility, and multi-homing in a HIP way. In Proc. Network and Distributed Systems Security Symposium (NDSS '03), pages 87-99, San Diego, CA USA, February 2003.

[15] Jonathan Rosenberg. The real time transport protocol (RTP) denial of service (DoS) attack and its prevention. Internet-Draft, IETF SIP Working Group, June 2003. Work in progress.

[16] Jerome H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. ACM Transactions on Computer System, 2(4):277-288, November 1984.

[17] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. TCP congestion control with a misbehaving receiver. Computer Communication Review, 29(5):71-78, October 1999.

[18] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spaffold, Aurobindo Sundaram, and Diego Zamboni. Analysis of a denial of service attack on TCP. In Proc. 1997 IEEE Symposium on Security and Privacy, pages 208-223, Oakland, CA USA, May 1997. IEEE Computer Society Press.

[19] Alex C. Snoeren and Hari Balakrishnan. An end-to-end approach to host mobility. In Proc. 6th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), pages 155-166, Boston, MA USA, August 2000.

[20] Randall R. Stewart, Lyndon Ong, Ivan Arias-Rodriguez, Kacheong Poon, Phillip T. Conrad, Armando L. Caro Jr., and Michael Tuexen. Stream control transmission protocol (SCTP) implementer's guide. Internet-Draft draft-ietf-tsvwg-sctpimpguide-09, IETF Transport Area Working Group, September 2003. Work in progress.

[21] Randall R. Stewart, Michael Ramalho, Qiaobing Xie, Michael Tuexen, Ian Rytina, Maria-Carmen Belinchon, and Phillip Conrad. Stream control transmission protocol (SCTP) dynamic address reconfiguration. Internet-Draft draft-ietf-tsvwg-addip-sctp-08, IETF Transport Area Working Group, September 2003. Work in progress.

[22] Randall R. Stewart and Qiaobing Xie. Stream Control Transmission Protocol (SCTP), A Reference Guide. Addison-Wesley, 2001.

[23] Randall R. Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Juergen Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang, and Vern Paxson. Stream control transmission protocol. RFC 2960, IETF, October 2000.

[24] Paul Vixie, Susan Thomson, Yakov Rekhter, and Jim Bound. Dynamic updates in the domain name system (DNS UPDATE). RFC 2136, IETF Network Working Group, April 1997.

[25] Brian Wellington. Secure domain name system (DNS) dynamic update. RFC 3007, IETF, November 2000.

[26] Wei Xing, Holger Karl, Adam Wolisz, and Harald Müller. M-SCTP: design and prototypical implementation of an end-to-end mobility concept. In Proc. 5th Intl. Workshop The Internet Challenge: Technology and Applications, Berlin, Germany, October 2002.