Assessing the Security of a Clean-Slate Internet Architecture

Gowtham Boddapati[‡] John Day[‡] Ibrahim Matta[†] Lou Chitkushev[‡]

[‡]Metropolitan College [†]College of Arts & Science
Computer Science, Boston University
{gowtham, day, matta, ltc}@bu.edu

December 2010
An earlier version appears in Technical Report BUCS-TR-2009-021

Abstract—**The TCP/IP architecture was originally designed without taking security measures into consideration. Over the years, it has been subjected to many attacks, which has led to many patches to counter them. Our investigations into the fundamental principles of networking have shown that carefully following an abstract model of Inter-Process Communication (IPC) addresses many problems [1]. Guided by this IPC principle, we designed a clean-slate Recursive InterNetwork Architecture (RINA) [2]. In this paper, we show how, without the aid of cryptographic techniques, the bare-bones architecture of RINA can resist most of the security attacks faced by TCP/IP and of course is only more secure if cryptographic techniques are employed. Furthermore, the RINA model indicates specifically where those security measures reside. We also show how hard it is for an intruder to compromise RINA. Then, we show how RINA inherently supports security policies in a more manageable, on-demand basis, in contrast to the monolithic one-size-fits-all approach of TCP/IP.**

I. Introduction

The TCP/IP architecture has shown signs of weakness as the Internet has grown and evolved. These problems are partly due to changing requirements—including mobility, quality-of-service, and security—but partly because of the architecture's rigid one-size-fits-all structure. In this paper, we focus on the security properties that are inherent in the Internet architecture.

As is often lamented, the TCP/IP architecture was originally designed without taking security considerations into account. Over the years, many vulnerabilities have been discovered and led to many patches to counter them. Given its rigid structure, security mechanisms have mostly been inserted into TCP/IP as "shim" sublayers lacking a comprehensive approach to security.

Most recently, there have been attempts to design *clean-slate* internet architectures. Our own investigations into the fundamental principles of communication led to a rather simple, elegant model based on a generalization of Inter-Process Communication (IPC). However, this model, referred to as RINA (Recursive InterNetwork Architecture) [2], was developed from IPC considerations alone, without explicitly considering security. Hence, it seemed wise to investigate its security properties at the outset.

Space does not allow us to consider all aspects of the security of RINA in this paper. (We hope to cover other aspects in subsequent papers.) Here, after a very brief overview of the pertinent aspects of RINA, we consider three types of vulnerabilities that have been found in TCP/IP: port-scanning attacks, connection-opening attacks and data-transfer attacks. What we find is that unlike the TCP/IP architecture, without the aid of cryptographic techniques, the bare-bones architecture of RINA is more secure and resistant to these attacks, even if we assume that a RINA network has been fundamentally compromised. Though further analysis is to be conducted, this might suggest that good design is as important to good security as explicit consideration of security.

We also show how RINA's model organizes cryptographic techniques in a way that clearly indicates the proper placement of security mechanisms, rather than the piecemeal approach of TCP/IP.

The rest of the paper is organized as follows. Section II reviews elements of TCP/IP and RINA that are most relevant to the security aspects discussed in this paper, specifically access control, addressing, and connection management. Section III compares the resiliency of TCP/IP and RINA to transport attacks, namely port-scanning, connection-opening and data-transfer. Section IV compares the two architectures in terms of their organizational support for diverse security policies. Section V concludes the paper.

II. Background: TCP/IP vs. RINA

Figure 1 illustrates the TCP/IP architecture. In [2], we identified the shortcomings of this architecture and attributed them to: (1) exposing addresses to applications, (2) artificially isolating functions of the same scope[1], and (3) artificially limiting the number of layers (levels).

Figure 2 illustrates our RINA architecture [2], which leverages the inter-process communication (IPC) concept.[2] In an operating system, to allow two processes to communicate, IPC requires certain functions such as locating processes,

[1] Transport and routing/relaying are split into two layers: Data Link and Physical layers over the same domain/link, and Transport and Network layers internet-wide.

[2] We use IPC in its long lost original sense of passing data messages between processes, rather than the broader current sense used today that encompasses this as well as all synchronization techniques.

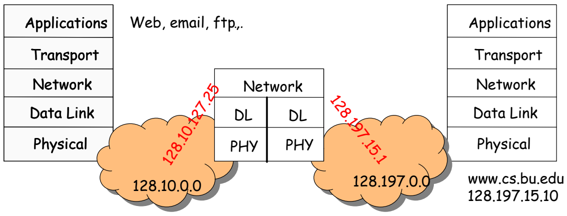


Fig. 1. TCP/IP Architecture

determining permission, passing information, scheduling, and managing memory. Similarly, two applications on different end-hosts should communicate by utilizing the services of a distributed IPC facility (DIF) that provide the same functions plus those required by the lack of a common memory. A DIF is an organizing structure—what we generally refer to as a layer. What functions constitute this layer, however, is fundamentally different. A DIF is a collection of IPC processes (nodes). Each IPC process executes routing, transport and management functions. IPC processes communicate and share state information. How a DIF is managed, including addressing, is hidden from the applications. To understand why layers must be organized this way, see [1].

The goal of a DIF is to provide a distributed service that allows application processes to communicate. One use of a DIF might be as a private network or overlay. Two novel aspects of a DIF is that it *repeats* and is *relative*. Each repetition addresses a different range of operation and/or scope. As shown in Figure 2, two IPC processes $P1$ and $P2$ in an N-level DIF communicate by utilizing the services of an (N-1)-level DIF. Thus, while the specific function of IPC processes is to do IPC, they are also application processes requesting IPC from a lower layer. Our IPC-based architecture can be found in [2]. In this section, we only highlight key aspects of this architecture that have a fundamental impact on security.

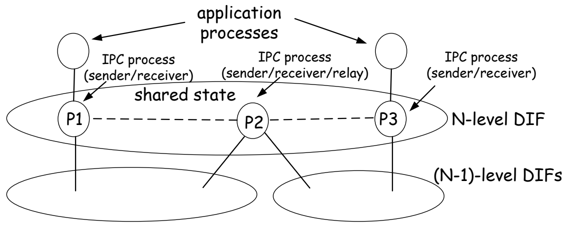


Fig. 2. RINA Architecture

A. Access Control

Unlike TCP/IP, RINA requires explicit enrollment for an IPC process within a system to either join an existing DIF, or create a new DIF.

a) Adding a New Member to an (N)-DIF: Suppose that DIF I consists of a number of IPC processes on a set of systems. Suppose that an IPC process, *j*, wants to join DIF I. j knows the application (service) name of an IPC process, *i*, in I, not its address — *j has no way of knowing the address of any process in DIF I*. i and j are connected by an underlying (N-1)-DIF[3]. Using the underlying (N-1)-DIF, j requests that the (N-1)-DIF establish an IPC channel (connection) with i using the application name of *i*. In RINA, application processes incorporate a common protocol for establishing application connections that includes a plug-in module for authentication.

The (N-1)-DIF determines whether i exists and whether j has access to *i*. After the connection has been established, i authenticates j and determines whether it can be a member of DIF I. This authentication can be as strong or as weak as required by the DIF. If the result is positive, *i* assigns an (N)-address to *j*. Note that the address is taken from the name space for DIF I, *i.e.*, DIFs have their own name (address) space. j uses the (N)-address to identify itself to other members of DIF I. Other initialization parameters associated with DIF I are exchanged with *j*, possibly including a shared secret key. The IPC process, *j*, is now a member of DIF I.

b) Creating a New DIF: Creating a new DIF is a simple matter. A management or similar application with the appropriate permissions causes an IPC process to be created and initialized, including pointing it to one or more (N-1)-DIFs. As part of its initialization, the IPC process is given the means to recognize allowable members of the DIF (*e.g.*, a list of application process names, a digital signature, and so on). It might be directed to initiate enrollment with them or to simply wait for them to find this initial IPC process. When this has been achieved, adding more members to the DIF proceeds as described earlier.

B. Addresses and their Binding

The TCP/IP architecture has a global addressing space, which allows any system to freely connect to any other system. On the contrary, in RINA, the addresses are *internal* to a DIF. For two application processes to communicate, they have to have access to a DIF in common. If there is no common DIF, then one must be created either by joining an existing DIF or creating a new one. This provides the opportunity to restrict access based on the security policy of the DIF.

In the TCP/IP architecture, TCP overloads the port-id to be both a local handle, which identifies the application process, and connection-endpoint-id, which identifies the data-transfer connection. Figure 3 illustrates TCP's management of data-transfer connections. And by overloading the port-id again by giving it application semantics as a *well-known* destination port forces the receiver to rely on the sender's id information for its identity/consistency checking, rather than ids it generated, which makes it easier for attackers to guess/spoof the source port and thwart any consistency checking by the receiver.

[3] Ultimately the lowest level DIF is the physical medium.

Unlike TCP/IP, RINA does *not* conflate port allocation (which must be hard-state / explicitly signaled) with transport state synchronization (which is timer-based / soft-state). In RINA, applications do not listen to a well-known port. Rather an application process requests service using the destination application-name. The local communication IPC process returns a port-id with only local significance to the user to use as an opaque handle. The request is translated into a set of policies for an EFCP (Error and Flow Control Protocol) flow. One end of the flow is instantiated by creating an EFCP-instance, identified by a different local identifier, referred to as a connection-endpoint-id (CEP-id). The local communication process then issues a create-request to find the destination application and if the request is successful/accepted, allocates the flow. Figure 4 illustrates RINA's management of data-transfer connections.

When the communication IPC process at the destination gets the create-request, it determines if it can accept the request. The degree of access control is a matter of policy — it could be quite elaborate, or null like the current Internet. If the request is accepted, the destination communication process instantiates an EFCP-instance with its own local CEP-id, and the result is returned to the requesting application. The source and destination CEP-ids are concatenated for use as a connection or flow id. If the create-request returns with a negative response, it is determined whether the cause is fatal or not. If not fatal, the source communication process may modify the request and try again. If the create-request returns with a positive response, the CEP-id is bound to the port-id. Note that each end uses only ids that it has generated to distinguish the flow.

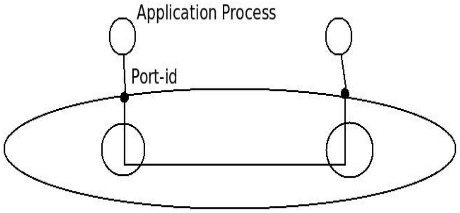


Fig. 3. TCP Connection

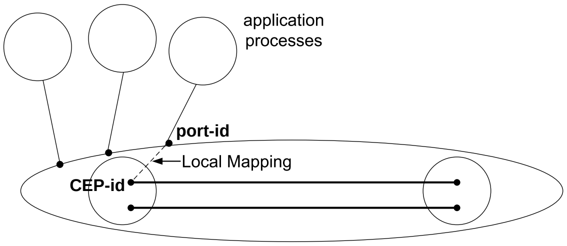


Fig. 4. RINA connections

C. Data Transfer

By separating port allocation (and access control) from transport state synchronization, data transfer in RINA can be cleanly done in a soft-state fashion and thus can support reliable or unreliable, short or long transfers. If there is a lull in the data transfer that is long enough to cause transport timers to expire, the connection state is simply deleted but ports are not deallocated. Ports are managed in a hard-state style. After a lull, once data transfer resumes, the connection state is immediately created.

RINA uses a soft-state data transfer protocol, built around Watson's Delta-t protocol [3]. This is in contrast to the hybrid hard-state/soft-state approach of TCP. In Delta-t, unless refreshed by data/ACK packet arrivals, a flow state is deleted after $2 \times MPL$ (Maximum Packet Lifetime) at the receiver, and $3 \times MPL$ at the sender. Figure 5 depicts a generic RINA sender/receiver. TCP, on the other hand, requires explicit control messages to synchronize the sender and receiver for the purpose of providing data reliability (i.e., no data loss or duplication). This makes TCP more vulnerable to attacks that fabricate such control messages, or cause them to be dropped [4]. It is worth noting that unlike TCP/IP where connection synchronization is overloaded with security mechanisms such as SYN cookies, RINA decouples authentication as part of enrollment when IPC processes first join a DIF.

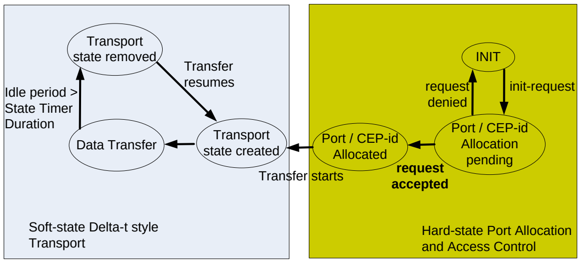


Fig. 5. RINA Sender / Receiver Protocol State

III. Transport Attacks on TCP/IP vs. RINA

A. Port-Scanning Attacks

Port scanning is often viewed as a first step for an attack, wherein the attacker explores "open" ports to which processes on a system are listening. In RINA, a service is accessed by its application-name—the requesting applications never see addresses nor CEP-ids. In fact they are not privy to any data-transfer identifiers. This is in contrast to TCP/IP in which a destination application process is assumed to listen to a well-known port. RINA also supports local access control domains that restrict which applications are visible to the DIF that the requestor belongs to. As described earlier, source and destination port-ids then get assigned locally on-demand. Ports are also dynamically mapped to separate data-transfer (connection) endpoints, and contrary to TCP/IP, ports are not part of the flow/connection id. This makes traditional port-scanning attacks not possible in RINA.

In RINA, however, the attacker might try to scan application names. But this is more difficult because application names are strings of variable length, a far larger name space. Furthermore, the malicious user has to be a member of the same DIF to be able to address other members in the DIF. Joining a DIF requires that the new IPC process be authenticated, providing further barriers to compromise RINA.

B. Connection-Opening Attacks

In this type of attack [5], the intruder attempts to establish a connection with the server, impersonating a trusted user A.

In TCP/IP, this attack exploits the explicit three-way handshake of TCP in which the client and server exchange (synchronize) their Initial Sequence Numbers (ISN) prior to data transfer. A malicious handshake sequence with server S, intruder X, and spoofed client A, may look like:

$$\begin{array}{lll} \text{X} & \longrightarrow & \text{S : SYN(ISNx), SRC= A} \\ \text{S} & \longrightarrow & \text{A : SYN(ISNs), ACK(ISNx)} \\ \text{X} & \longrightarrow & \text{S : ACK(ISNs), SRC = A} \\ \text{X} & \longrightarrow & \text{S : ACK(ISNs), SRC = A, malicious-data} \end{array}$$

In this attack, we assume that the attacker X already knows the destination port and IP address, as well as the source IP address. The destination port and IP address are easy to obtain, as they are generally published, as well-known ports. The source IP address is also generally easy to obtain, as this is simply the client that is being spoofed. As this is a connection establishment phase, the intruder can use any one of the ports as source port-id. This attack also assumes that the acknowledgment (ACK) sent by the server and destined to the spoofed system A, is lost or delayed, either because A itself was down or slow (possibly through a separate attack) or the ACK is intercepted and dropped by the intruder X.

The difficult part of launching this attack is determining the ISN of the server. This could be more easily obtained if the intruder is in the middle and observes the (unencrypted) traffic between A and S. Otherwise, the intruder has to guess ISNs, which given 32-bit sequence numbers and random selection of ISNs, involves 2^{32} possibilities.

In TCP/IP, the data packet that follows the three-way handshake can contain any arbitrary, perhaps malicious, data. This can lead to attacks such as connection-opening attacks, unless TLS (Transport Layer Security) is used. On the other hand, RINA *requires* TLS functionalities to be applied recursively. Specifically, in RINA, the communicating application processes *inherently* use a common application protocol for establishing and releasing application connections. By using this protocol the receiver expects the authenticated packets to follow the connection establishment phase, which greatly reduces the risk of connection-opening attacks. A message sequence illustrating RINA's transport connection establishment, followed by application authentication (challenge / response), in the presence of an attacker X spoofing client A, looks like:

$$\begin{array}{lll} \text{X} & \longrightarrow & \text{S : create-request(service-name, A, S,} \\ & & \qquad\qquad\text{source CEP-id, QoS, } \cdots) \\ \text{S} & \longrightarrow & \text{A : create-response(OK, destination CEP-id, } \cdots) \\ \text{X} & \longrightarrow & \text{S : ACK(destination CEP-id), ISNc, } \cdots \\ \text{S} & \longrightarrow & \text{A : challenge(}\cdots) \\ \text{X} & \longrightarrow & \text{S : response(}\cdots) \\ \text{X} & \longrightarrow & \text{S : data} \end{array}$$

In this RINA attack scenario, we assume that the intruder X has somehow thwarted the DIF enrollment authentication described earlier, and is a member of the DIF as are A and S, but we note that these are the hurdles that a TCP intruder does not need to overcome. If that is the case, X is able to know the addresses of A and S, *i.e.*, X is launching an insider attack. As this is a connection establishment phase, the intruder can use any source CEP-id. And since in RINA, there is no need for synchronizing sequence numbers [3][4], the sender can also use any initial sequence number. Assuming X does not observe the reply with the destination CEP-id, it has to guess this CEP-id. Assuming standard field lengths, we take the length of CEP-id to be the same as that of a port-id (i.e., 16 bits), thus guessing CEP-id involves 2^{16} possibilities. This makes this type of attack equivalent to port-scanning attacks, in which an intruder may be attempting an unallocated destination CEP-id. Such attacks raise more suspicion (and hence, are easier to detect) than TCP attacks that guess ISN.

C. Data-Transfer Attacks

Data-transfer attacks, known as *blind in-window attacks* [6], are those where the attacker does not have access to the data packets of the victim connection but still attempts to inject packets that seem legitimate. Forming a legitimate packet requires guessing various fields in the packet's header.

In TCP/IP, the goal of this type of attack might be to abort an ongoing connection by injecting a TCP "reset" [6], [7]. The damage depends on the application running above the TCP connection. One such application is BGP, where a connection abort would result in entries of the routing table being flushed. In this attack we assume that the attacker knows the destination port and IP address, as well as the source IP address. The destination port and IP address are easy to obtain, as they are published. The source IP address is also generally easy to obtain, as this is simply the spoofed client. The intruder has to guess the source port as well as the sequence number that has to lie within the window of the receiver.

To guess the source port-id, given 16-bit port numbers, we have at most 2^{16} possibilities. Furthermore, for each possible source port-id, given 32-bit sequence numbers and say 64KB window size[5], we have $\frac{2^{32}}{2^{19}} = 2^{13}$ possibilities for selecting a sequence number that lies within the current

[4] Recall that RINA uses a Delta-t [3] style data transfer protocol, whereby new and old data connections are distinguished by connection ids that are assigned for at least $2 \times MPL$ to ensure data packets and duplicates for a particular connection have died out before reusing the same connection id. Thus, there is no need to synchronize sequence numbers for that purpose.

[5] 64KB is the default TCP maximum window size, without window scaling options. Note that 64KB = $2^{6+10+3} = 2^{19}$.

receiver's window. Thus, there is a total of $2^{16+13} = 2^{29}$ possibilities. Note that for larger window sizes[6], typical of higher bandwidth-delay-product networks, the attack will be easier to launch.

In the case of RINA, the intruder can launch an attack during two different phases of a connection: (1) after the resource-allocation request is complete and before the data transfer phase starts, or (2) during the data transfer phase. Again here we assume that the intruder is in the same DIF, so the attacker knows the addresses of the source and destination IPC processes.

In the first case, the attacker has to guess the source CEP-id and the destination CEP-id. The attacker also has to guess other agreed-upon parameters of the connection, such as the QoS-id, though as a member of the DIF, he/she knows the legal range of QoS-ids. Since the data transfer phase has not started, the attacker can use any ISN. Given 16-bit CEP-ids and 8-bit QoS-id, the attacker has $2^{16+16+8}$ = 2^{40} possibilities for guessing the CEP-ids and QoS-id for the victim connection.

In the second case, in addition to the CEP-ids and QoS-id, the attacker has to guess the sequence number which falls within the window of the receiver. This guessing involves 2^{40+13} = 2^{53} possibilities, assuming 64KB window size. This type of attack is made even harder because of RINA's use of a Data-Run-Flag (DRF) during its Delta-t's style data transfer [3]. If the DRF bit is set, this implies that the sender has no data left to be acknowledged or it is starting a new data run. Thus, the DRF bit periodically synchronizes the sender and receiver, and so setting it incorrectly in the attack packet would raise suspicion.

For example, if the DRF bit is not set and the receiver's connection state had timed out (because it has not been refreshed by new data from the sender), the attack packet is simply dropped by the receiver. Let's then assume that the attacker always sets the DRF bit, along with an arbitrary sequence number, in its attack packet. This attack packet is accepted only if the receiver had no state for this connection. Otherwise, the receiver can verify whether the setting of the DRF bit makes sense, which is the case only if the receiver has indeed acknowledged all prior data packets.

Finally, this type of attack is not possible or harder to launch in RINA for two reasons: (1) RINA uses a soft-state approach in managing connections, thus it does not use explicit connection "reset" messages, which precludes "reset" attacks,[7] and (2) RINA supports the dynamic assignment of CEP-ids during the lifetime of a connection, binding them to the same port-ids that are only locally-visible. This would make it very hard for an attacker to guess the source and destination CEP-ids.

1) Blind TCP Data Injection through Fragmented IP Traffic: Zalewski [8] described a possible attack that can be performed on TCP/IP that does not require the attacker to guess or know the aforementioned TCP connection parameters and could therefore be successfully exploited in some scenarios with less effort than that required to exploit the more traditional data-injection attacks.

The attack is performed when one system is transferring information to a remote peer by means of TCP, and the resulting IP packet gets fragmented. In this case, the first IP fragment will usually contain the entire TCP header, including port numbers, sequence number, and other information that may be relatively difficult for a third party (the attacker) to guess otherwise. The other fragments carry the remaining sections of the TCP payload, which would be put back together (reassembled) at the receiver. Instead of attempting to guess TCP header's information such as port and sequence numbers, the attacker may spoof any of the IP fragments subsequent to the first fragment, inserting malicious data into the TCP payload that causes the reassembly to fail. Zalewski [8] discusses the feasibility of such attack.

This security problem arises in the TCP/IP architecture because fragmentation/reassembly is done by both TCP and IP—TCP can produce segments that are larger than IP's MTU (Maximum Transfer Unit) size. In RINA, because the transport and routing functions are integrated into the same DIF layer [9], fragmentation/reassembly occurs only once to segment/fragment Service Data Units (SDUs).

D. Summary

Table I summarizes our comparison of RINA against TCP/IP under transport-level attacks. We assume 32-bit sequence numbers, 16-bit port-ids/CEP-ids, 64KB window size, and 3-bit QoS-id. To be able to make a direct comparison, we had to assume that a RINA network had been compromised and a rogue member had been allowed to join—a hurdle that is not present in TCP/IP networks.

IV. Security Policies in TCP/IP vs. RINA

RINA decouples the various security functions of authentication and confidentiality/integrity. The former is done by the applications of the DIF where applications of the DIF authenticate each other. The latter is done at the bottom of the DIF where the IPC processes encrypt their traffic if they do not trust the lower DIFs. These security functions are applied recursively, so IPC processes themselves would authenticate each other when communicating through lower-level DIFs. Policies of the DIF determine the levels of authentication and encryption. Figure 6 illustrates this functional organization.

In contrast, TCP/IP implements security functions piecemeal, for example, using TLS under the application layer and IPSec below the network layer. The TCP/IP organizing structure is rigid and can only accommodate security functions as "shim" sublayers, rather RINA accommodates them as an integral part of (recursive) inter-process communication.

Figure 7 illustrates a "middlebox" solution to enable the support of "private" domains in TCP/IP. Such a middlebox is known as Network Address Translator (NAT) since it aggregates private addresses of systems inside the private

[6] Larger window sizes are possible using window scaling options.

[7] In a soft-state approach, the connection's state at the receiver is automatically reset after 2 × MPL if not refreshed by the sender [3], thus there is no need for an explicit "reset" message.

TABLE I

Comparison of TCP/IP and RINA under Transport Attacks. To be able to make a direct comparison, we had to assume that a RINA network had been compromised and a rogue member had been allowed to join—a hurdle that is not present in TCP/IP networks.

Vulnerability	TCP/IP	RINA
Port-scanning	possible due to well-known ports	not possible with unknown CEP-ids
Connection-opening	2^{32} possibilities to guess ISN	2^{16} possibilities to guess destination CEP-id
Data-transfer (right after conn. open)	2^{29} possibilities to guess source port-id and valid SN	2^{40} possibilities to guess source and destination CEP-ids and agreed-upon QoS-id
Data-transfer (after transfer started)	2^{29} possibilities to guess source port-id and valid SN	2^{53} possibilities to guess source and destination CEP-ids, agreed-upon QoS-id, and valid SN

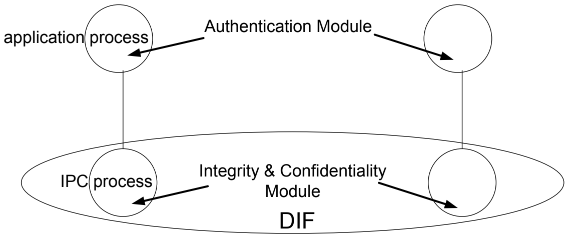


Fig. 6. Security policies applied recursively

domain (such as system "B" in the figure) into the NAT public address. Communication across the private domain and the public (Internet) domain, say between systems "B" and "A", is done through the NAT, which translates between its public NAT address and port number, which identifies "B" externally, and B's actual private address and port number. Furthermore, the NAT acts as a firewall, preventing attacks on private addresses and ports. However, it is clear that this kind of hand-crafted arrangement makes it hard to coordinate communication across domains when we want to.

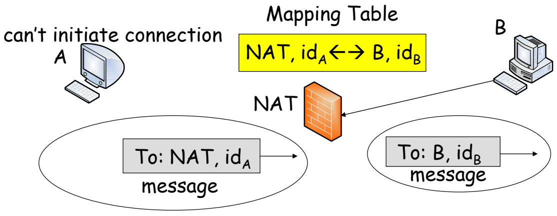
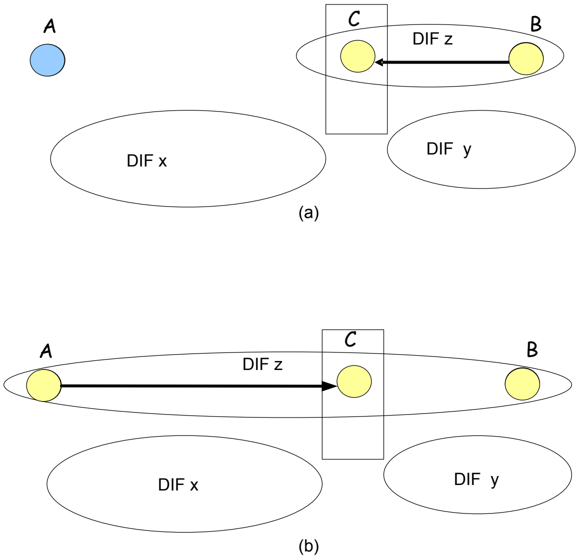


Fig. 7. Security through NATs in TCP/IP

Figure 8 illustrates the procedure in RINA, where communication is established between application processes to join the same DIF. First, process "B" joins DIF *z*, which initially only contains process "C" (Figure 8(a)). As mentioned earlier, this explicit enrollment procedure happens using a common underlying DIF (DIF *y*, in this example), and involves authenticating that B is a valid member of DIF *z*, initializing it with current DIF information, and assigning B an internal address for use in coordinating communication within DIF *z*. Then, similarly, process "A" joins DIF *z* (Figure 8(b)). Thus, in RINA, there are no "middleboxes" per se, but rather processes join and leave DIFs as determined by management (security) policies. Furthermore, such enrollment procedures can be repeated horizontally to create concurrent DIFs, or vertically to create stacked DIFs.



(a)

(b)

Fig. 8. (a) Process "A" is about to join DIF *z*, (b) Process "A" after joining DIF *z*.

V. Conclusion

In this paper, we compare a clean-slate internet architecture, RINA, that is based on fundamental IPC principles, to TCP/IP in terms of architectural support for security. We specifically compare the resiliency of RINA to security vulnerabilities found in the TCP/IP architecture. In some cases, to make a fair comparison, we had to assume that a RINA network had been compromised and a rogue member had been allowed to join. (A hurdle that is not present in TCP/IP networks.) Even so, we found RINA to be more secure and resistant to these attacks.

We focused on access control, addresses and their binding, and data transfer. We contrast the open access of TCP/IP to the controlled access of RINA, which requires an *explicit* enrollment phase to join a network of IPC processes (DIF). Unlike TCP/IP, in RINA, node addresses (of IPC processes) are internally assigned by a DIF, and are not exposed to application processes. Furthermore, data connections are *dynamically* assigned connection endpoint ids (CEP-id), which are bound to dynamically assigned ports. This late (dynamic) binding of addresses / ids provides levels of indirection that make RINA inherently more secure than TCP/IP, which exposes static addresses and port numbers to applications.

We compare the resiliency of RINA and TCP/IP to transport-level attacks. We show how the static assignment of addresses and ports, as well as the hard-state approach of TCP/IP to synchronizing connection states for reliable data transfer, makes TCP/IP quite vulnerable to port-scanning, connection-opening, and data-transfer attacks. On the other hand, the dynamic assignment of addresses and ports, the decoupling of port numbers from CEP-ids, and the soft-state approach to data transfer, makes RINA quite resilient to such attacks. We believe that this is an interesting result, given that no more consideration of security was present in the development of RINA than in the development of the TCP/IP architecture. One might be led to conclude that strong design is as important to good security as explicit consideration of security. In other words, TCP/IP does not suffer as much from a lack of foresight as a weak design.

Finally, we argue that the recursive nature of RINA organizes the security policies in a clean way, decoupling authentication from integrity and confidentiality.

VI. Acknowledgment

This work was supported in part by NSF grants CNS-0963974, CCF-0820138 and CSR-0720604.

References

[1] J. Day, *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall, 2008.

[2] J. Day, I. Matta, and K. Mattar, ""Networking is IPC": A Guiding Principle to a Better Internet," in *Proceedings of ReArch'08 - Re-Architecting the Internet*. Madrid, SPAIN: Co-located with ACM CoNEXT 2008, December 2008.

[3] R. Watson, "Timer-Based Mechanisms in Reliable Transport Protocol Connection Management," *Computer Networks*, vol. 5, pp. 47–56, 1981.

[4] G. Gursun, I. Matta, and K. Mattar, "Revisiting A Soft-State Approach to Managing Reliable Transport Connections," in *Proceedings of the 8th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT)*, Lancester, PA, November 2010.

[5] S. M. Bellovin, "Security Problems in the TCP/IP Protocol Suite," *Computer Communication Review*, vol. 19, no. 2, pp. 32–48, 1989.

[6] P. Watson, "Slipping in the Window: TCP Reset attacks," Presentation at 2004 CanSecWest, 2004, http://cansecwest.com/csw04archive.html.

[7] F. Gont, "Security Assessment of the Transmission Control Protocol," CPNI Technical Note, Feburary 9 2009.

[8] M. Zalewski, "A New TCP/IP Blind Data Injection Technique?" Post to the bugtraq mailing-list, 2003.

[9] K. Mattar, I. Matta, J. Day, V. Ishakian, and G. Gursun, "Declarative Transport: A Customizable Transport Service for the Future Internet," in *Proceedings of the* 5^{th} *International Workshop on Networking Meets Databases (NetDB 2009), co-located with SOSP 2009*, Big Sky, MT, October 2009. [Online]. Available: http://www.cs.bu.edu/fac/matta/Papers/netdb09.pdf