A High-Performance Capabilities-Based Network Protocol

Tilman Wolf and Kamlesh T. Vasudevan
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{wolf,kthondar}@ecs.umass.edu

***Abstract*—Capabilities-based networks present a fundamental shift in the security design of network architectures. Instead of permitting the transmission of packets from any source to any destination, routers deny forwarding by default. For a successful transmission, packets need to positively identify themselves and their permissions to the router. A major challenge for a high-performance implementation of such a network is an efficient design of the credentials that are carried in the packet and the verification procedure on the router. Based on our prior work on the design of packet credentials, we present a network protocol that implements these concepts. Our prototype implementation shows that there is some connection setup cost associated with this type of secure communication. However, once a connection is established, the throughput performance of a capabilities-based connection is similar to that of conventional TCP.**

I. Introduction

Security requirements for some network systems demand tight control over the traffic that is forwarded. Example domains for such networks are military communication, financial transactions, and telemedicine. In these networks, there is typically a clear setup of authentication and access control, which defines the users and application that are permitted to transmit traffic. The need for such control lies in the concerns about the potential impact of unauthorized and malicious traffic. In the current Internet, where no authentication and access control system is in place, malicious traffic is the cause of denial-of-service attacks and hacking attacks that target end-systems. Similar vulnerabilities are unacceptable in highly secure network environments, where denial-of-service attacks could lead to loss of life or significant financial implications. Therefore, it is important to develop secure networking protocols for such domains.

Using current Internet protocols, it is very difficult to identify traffic that is unwanted, unauthorized, or malicious. One of the main reasons is the fundamental design philosophy of the Internet, which allows any end-system to send traffic to any other end-system (“permit-by-default”). The default operation by any network router is to forward any traffic. Some systems that were added to the Internet more recently aim to identify undesirable traffic using heuristics (e.g., firewalls, intrusion detection systems). However, these devices can only identify and remove a fraction of all potentially malicious traffic and thus are not a fundamental solution to the problem.

This material is based upon work under a subcontract #069153 issued by BAE Systems National Security Solutions, Inc. and supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-08-C-2013.

An alternate approach to limiting unauthorized traffic in a network is to disallow any traffic unless it has been identified and deemed permissible. This “off-by-default” approach has been proposed in capabilities-based network, where packets carry authentication tokens (“capabilities”) and need to pass one or more checkpoints [3], [5], [18]. Our recent work extends this idea and presents a network architecture with enforcement on every hop [15]–[17], which allows for 1-hop containment of denial-of-service attacks and also solves the denial-of-capabilities problems of previous work. The capabilities data structure, which is carried in packets to identify permitted traffic, is based on a Bloom. This Bloom filter contains index bits that are computed based on cryptographic exchanges between the sending node and each router. This capabilities design exhibits the necessary characteristics for secure and high-performance networking: capabilities are cryptographically strong (i.e., very difficult to forge), but can be verified easily (e.g., during forwarding at high data rates).

In this paper, we present protocol design details and implementation results from a prototype of our capabilities-based data path architecture. Specifically, our contributions are:

- A detailed presentation of the protocols used for capabilities setup and enforcement in our prototype system.
- Performance evaluation results, which show that while connection setup can be expensive, the throughput achieved after secure connection establishment is comparable to that of regular TCP.

Note that the development of such protocols is not bound by the need for compatibility with the existing Internet. It is common to develop specialized protocols for highly secure networking domains (e.g., military networks). In addition, developments in next-generation Internet designs indicate that emerging network architectures will support multiple parallel protocol stacks (e.g., as slices on a virtualized network infrastructure [2]). Thus, the proposed solution can be deployed as a specialized network in these environments.

The remainder of this paper is organized as follows. Section II discusses related work. The functionality of our capabilities-based architecture is presented in Section III. The protocol implementation is discussed in Section IV and performance results are presented in Section V. Our work is summarized in Section VI.

II. Related Work

The use of capabilities in networking has been proposed as an approach to deny-by-default networking in [3], [5]. An example of how denial-of-service attacks can be mitigated is discussed in [18]. Capabilities that are limited to a local area network are discussed in [8], where flows can be controlled in an enterprise network. All these designs use a single or few enforcement points within the network. Furthermore, credentials are provided by a single entity, thus exposing the network to denial-of-capabilities attacks as pointed out in [4]. In our recent work, we propose a network architecture where capabilities are verified on every hop [15]–[17]. Moreover, credentials are composed from cryptographic exchanges with each router along the path. This process ensures that denial-of-service and denial-of-capabilities attacks can be contained. The importance of 1-hop containment, which we can achieve with our design, is discussed in the context of a military network in [1].

To deploy the proposed system, the network (i.e., router systems) needs to support the functionality that we discuss in this paper. There has been work on introducing novel features in routers using programmable packet processing systems [14]. However, the proposed security solution requires support on all nodes of the network. A network-wide deployment of new protocols can be considered in the context of next-generation Internet designs [10]. In particular, the concept of network and router virtualization provides a deployment platform for domain-specific protocols [2]. There exist several example platforms on which our proposed protocols could be implemented [6], [12]. The prototype we discuss in this paper is implemented on the Emulab testbed [13].

III. Data Path Credentials Architecture

Before discussing the details of the protocol implementation in Section IV, we briefly review how capabilities are implemented in our system. Therefore, we summarize our prior work that has been published in [15]–[17]. Some of the text and figures in this section are quoted from these papers. Note that we refer to the capabilities data structure that is carried in each packet as "data path credentials."

A. Operational Principle

The general operational principle of our credential-based network architecture is illustrated in the space-time diagram in Figure 1. This figure shows the setup and initial data transfer for a connection. During the connection setup, the end-system sends a connection setup request along the path of the connection. Each router responds to the end-system with a challenge. This challenge represents the negotiation process where a router authenticates the end-system, checks local and global policies, etc. The challenges may be different for each router and thus cannot be combined into a single challenge. Once the end-system has identified itself satisfactorily to a router, the router returns its part of the credentials. The set of all router credentials is then combined into the data path credentials that are carried in each data packet. Each data packet is checked on every router. If the credentials contain the credentials of the router, the packet is forwarded. If the credentials do not match, the packet is discarded.

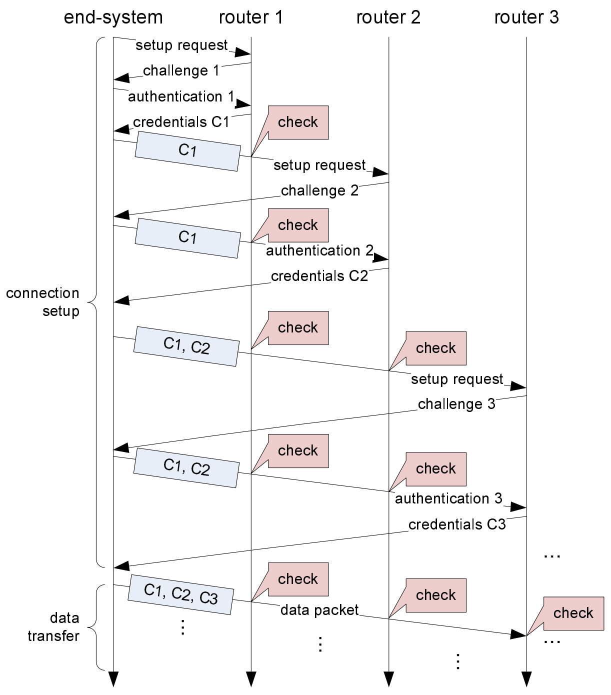

Fig. 1. Connection Setup to Establish Data Path Credentials.

With respect to achieving access control and containment of unauthorized traffic, the key aspects of this design are:

- Credentials are issued by routers and not by a central node. This design avoids the problem of denial-of-capabilities attacks, where the central node gets swamped by bogus requests. In our design, a request cannot go beyond the first hop unless that router's challenge is answered correctly. Thus, a malicious node can bring down systems within one hop of itself, but cannot spread its impact any further (i.e., 1-hop containment).
- All packets are checked on all hops. Only by verifying that packets are authorized to traverse the network everywhere, it can be ensured that malicious traffic can be contained. Note that in Figure 1, only checks in one direction of traffic are shown. In the prototype discussed in Section IV, challenges and credential messages are also checked on each hop (e.g., challenge 2 and credentials C2 on router 1).

B. Credential Design

The key aspect of the above architecture is the design of the data path credentials itself. The requirements for this data structure are: (1) compactness since it needs to be carried in each packet, (2) ease of verification since it needs to be checked on every hop, and (3) cryptographic strength to avoid forging of credentials.

To meet the above requirements, we use a data structure that is based on Bloom filters [7]. The main idea of our credentials

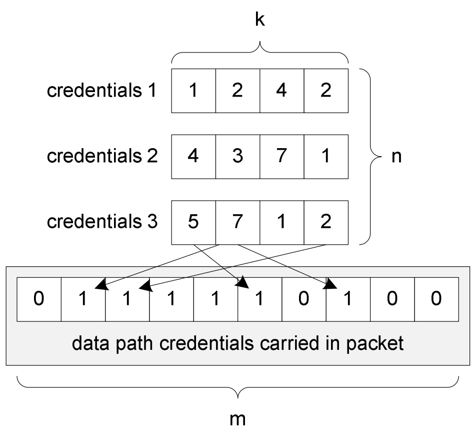

Fig. 2. Credentials Data Structure. This example shows three credentials that are aggregated to a set of 1's in the Bloom filter data structure.

design is that a Bloom filter can maintain multiple credentials in the same fixed-size data structure. Thus, the credentials of all routers along the path of a packet can be placed into this data structure.

To review, a Bloom filter is a bit array that can store m bits. Using k different hash functions $h_1(x) \dots h_k(x)$, an element x is mapped to k bit position in the array. An empty Bloom filter data structure starts with all array values set to 0. When adding element x, the bits corresponding to the hash function values for element x are set to 1. As multiple elements are added, it is possible (and intended) that set bits overlap (i.e., are combined with a logical OR function). When performing a check for membership of an element, the hash functions for the element are computed and it is checked if the according bits in the array are set. Only if all of these bits are set to 1, the element is reported to be a member of the set. The membership test is of a probabilistic nature and false positives are possible (i.e., elements that are not members of the set may be reported to be members), but false negatives are not (i.e., elements that are members of the set will never be reported as not being members). One of the properties of a Bloom filter is that at this point there is no known method to perform a reverse operation where the list of members is extracted from the Bloom filter data structure.

To use the Bloom filter data structure as credentials for packets that traverse the network, we store signatures from routers (as shown in Figure 2). When router j ($1{\le}j{\le}n$) permits transmission, it provides the source with its router credentials r_j. Router credentials are the set of indices $r_j[i]$ ($1{\le}i{\le}k$) of bits that are set in the Bloom filter array. The credentials from all routers along the path are then superimposed (i.e., logical OR operation) in the Bloom filter data structure. This creates aggregate credentials c (consisting of a single bit array of size m) that are sent with each data packet.

When receiving a packet with aggregate credentials c, router j can then check the value of all bits that were provided in router credentials r_j. If the aggregate credentials are valid, then $c[r_j[i]] = 1$ for all bits i. If the credentials do not contain the signature of a router, then it is likely that one of the bits in credentials c does not contain a 1 at one of the router credentials' bit positions. Thus, the validation of the aggregate credentials fails.

The security of the network architecture depends on the security of the credentials. That is, it should not be practically feasible to generate fake credentials for attack traffic. In the context of Bloom filter credentials, it should therefore be difficult to guess which bits are set by any given router. We can achieve this by using cryptographically strong hash functions (e.g., MD5 [11] or SHA-1 [9]) where router j uses k secret keys $s_j[i], 1 \le i \le k$. The cryptographic hash function $h_i(s_j[i], f)$ uses router j's key for bit index k to determine which bits to set in the aggregate credentials. It is important that this function also uses flow identifier f (e.g., based on a 5-tuple hash) as an input to avoid attacks where credentials from an authenticated connection are used by a different connection. If router credentials are used by a different connection, the validation step fails.

It is important to note that a Bloom filter with all bits set to 1 will always pass all tests. Therefore, there is a limit on how many bits may be set in any data path credentials. If this limit is exceeded, the packet is discarded. For more information on the credential design and its security properties, please see [15], [16].

C. Router Operation

A router needs to implement two aspects of this network architecture: the control path for connection setup and the data path for packet forwarding. The conceptual view of such a router is shown in Figure 3. The control path is shown on top, where credentials are created. The credential computation is triggered by either an explicit request by a router or by packets that need to be checked, but do not have an entry in the credentials cache. (Note that in practice even credential requests arrive via the network interface shown in the data path.) In the data path, packets are classified to obtain a flow identifier, which serves as an index in the credentials cache. This cache contains the indices that must be set in the Bloom filter carried in the packet. If the validation succeeds, the packet is forwarded. Otherwise it is dropped. If a packet encounters a credential cache miss, the recomputation of credentials takes lowest priority over all other tasks (to avoid attacks that cause system overload from bogus credentials computations).

IV. Protocol Implementation

We have designed and prototyped a protocol that implements the functionality of data path credentials as discussed above. This Data Path Credentials Protocol (DPCP) is located between the network layer and the transport layer of the Internet protocol stack. In principle, it is possible to integrated data path credentials with a new network layer protocol. However, we do not consider the redesign of the Internet Protocol in this work. Instead, we leverage the functionality of the existing IP protocol and add DPCP on top of it.

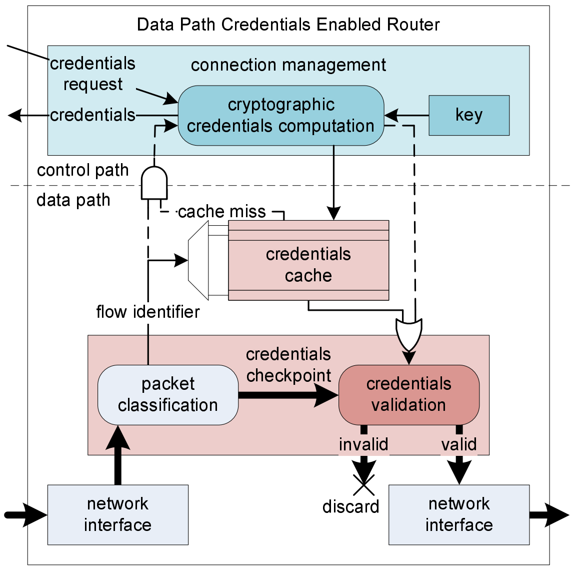

Fig. 3. Router with Data Path Credentials (from [16]).

A. Credentials Header

The header for DPCP is shown in Figure 4. The overall header is 28 bytes in size and is based on a Bloom filter configuration of $m = 128$ and $k = 4$. The fields in the header are used as follows:

- Port numbers: The first 32 bits are port numbers identical to how they are used in TCP and UDP. These fields are mere copies of the values that are carried in the layer 4 header of the packet (which is assumed to be either TCP or UDP to allow for packet classification). It can be debated if it is a "cleaner" design to copy these values or to let DPCP read the layer 4 protocol header. If the compactness of the header is important, these fields could be omitted in an optimized implementation.
- Next protocol: This 8-bit field indicates the layer 4 protocol header in the packet. This field is identical to the next protocol field in the IP header. (For our implementation, the IP header indicates 253 for DPCP, which is reserved for experimental protocols.)
- Flags: The flags that are used in our protocol implementation to indicate packet types used during connection setup. The types used are the following:
  - Setup flag (S): Indicates a packet containing a setup request.
  - Challenge flag (C): Indicates a packet containing a challenge to an end-system.
  - Response flag (R): Indicates a packet containing a response to a challenge.
  - Credentials flag (I): Indicates a packet containing Bloom filter indices.
- Setup field: This field can be used in different ways for establishing connections. The use is identified by the flags.
  - Challenge nonce (32 bits): This nonce is used by the router to challenge the end-system.
  - Response (32 bits): The end-system sends this encrypted nonce to prove its identity.
  - Bloom filter indices (28 bits): These four indices (each with a size of $\log m = \log 128 = 7$ bits) indicate the bits that are set by a router as its credential in the Bloom filter.
- Data path credentials: This 128-bit field is the Bloom filter that carries all router credentials.

Note that the Bloom filter indices are separate from the Bloom filter itself. The reason for this design is that during connection setup, the Bloom filter is used to permit communication to and from the router from which credentials need to be obtained. The credential itself is carried in the Bloom filter indices. Since these indices are not used after connection setup, an optimized implementation may use two different header formats for setup and data transfer.

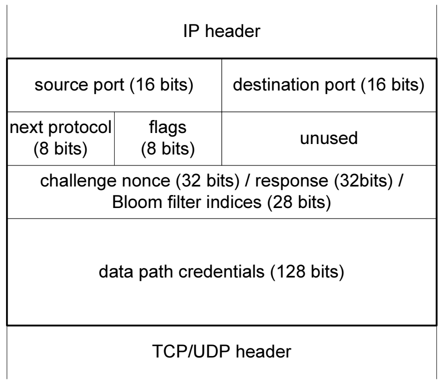
IP header
source port (16 bits)
destination port (16 bits)
next protocol (8 bits)
flags (8 bits)
unused
challenge nonce (32 bits) / response (32bits) / Bloom filter indices (28 bits)
data path credentials (128 bits)
TCP/UDP header

Fig. 4. Data Path Credentials Protocol Header.

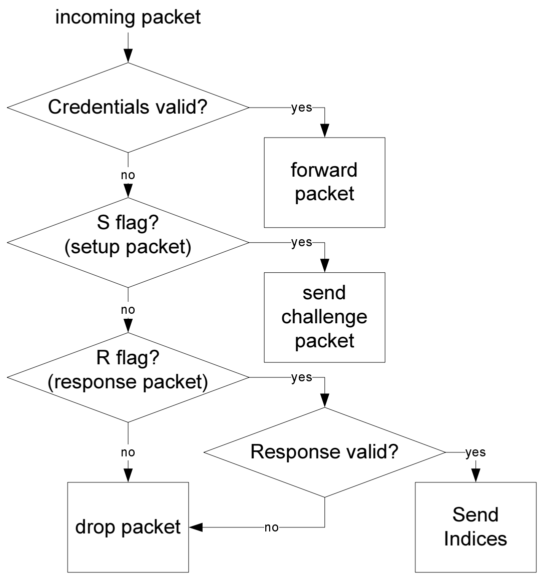

Fig. 5. Decision Diagram for DPCP Processing on Router.

B. Router Processing

When DPCP packets arrive on a router, it needs to be distinguished if they belong to a connection setup request to this router or if they should be forwarded. Note that a router needs to forward connection requests that are directed at other routers along the path. To make this distinction, the router simply checks if the data path credentials carried in the packet are valid (see Figure 5). If they are, then there is no need to further consider the packet locally. (Note that this is also true in case the credentials of previous routers in the path accidentally cause a false positive in the Bloom filter.) In case of a setup packet, the appropriate challenge is sent to the end-system. In case of a response, it is verified and the Bloom filter indices are returned to the sender. In all other cases, the packet is dropped.

C. Bidirectional Verification

One important practical consideration is that most communication in the Internet is bidirectional. Even in the setup phase, the challenge and credential packets need to reach the sender. To avoid that routers have to set up (and store) credentials for this return path, we set up the system to use the same data path credentials for both directions of traffic. (Note that we assume symmetric routes.) Thus, the challenge and credential packets simply carry the same data path credentials that were carried in the original packet.

To implement this type of bidirectional verification, it is necessary to modify the flow identification process. In our system, the classification result of a 5-tuple of a connection in one direction should match that of a connection in the opposite direction. We achieve this by sorting the IP addresses and port numbers before providing them to the classifier. Thus, when the IP addresses and port numbers are swapped on the return path, the sorting ensures that the classification result is still the same.

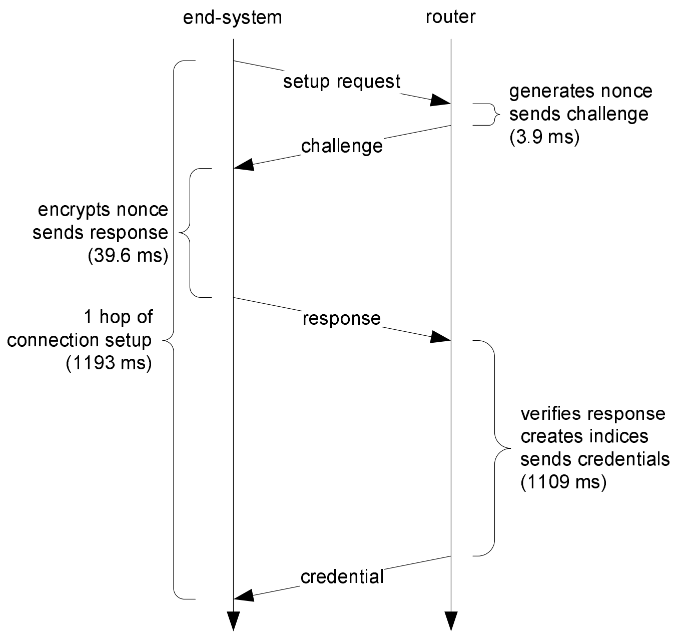

Fig. 6. Connection Setup Time for Single Hop.

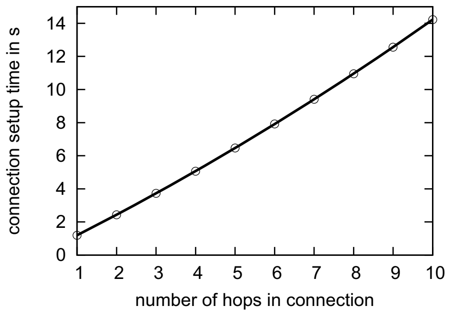

Fig. 7. Connection Setup Time for Different Number of Hops.

V. Performance Results

Using the DPCP prototype implementation on Emulab, we have collected several performance results. The most important result, of course, is the correct operation of the overall protocol. In our implementation, packets are forwarded when they carry the correct data path credentials and they are dropped when they do not.

A. Connection Setup

The connection setup is one of the more complex aspects of our protocol and thus requires considerably more time than conventional TCP setup. The measurement results for a single hop of the connection setup process are shown in Figure 6. We assume the propagation delay between each pair of hops is 10 ms. The cryptographic operations involved in this process are the main reasons for the amount of time required for setting up data path credentials. The challenge and response step involves a public-key/private-key cryptographic operation, which requires around 40 ms. The more expensive step, however, is determining the Bloom filter indices. This process involves the computation of an SHA-1 cryptographic hash, which takes more and 1 s.

Figure 7 shows the growth in setup time for connections with more hops. The increase is slightly steeper than linear due to the increase in propagation time to reach nodes that are farther away. Our implementation did not optimize for performance during the connection setup. Clearly, these cryptographic operations can be performed faster with better software implementation or with cryptographic co-processors. The Emulab platform does not provide cryptographic co-processors, so we were not able to explore the potential speedup from using such devices.

B. Forwarding Delay and Throughput Performance

As mentioned above, one of the key goals of our work is to design credentials that can be enforced on every hop because they can be easily validated. The check of bits in a Bloom filter against the indices stored in the credentials cache can be implemented efficiently. The results of an evaluation of the packet processing delay on a node is shown Figure 8. Data is

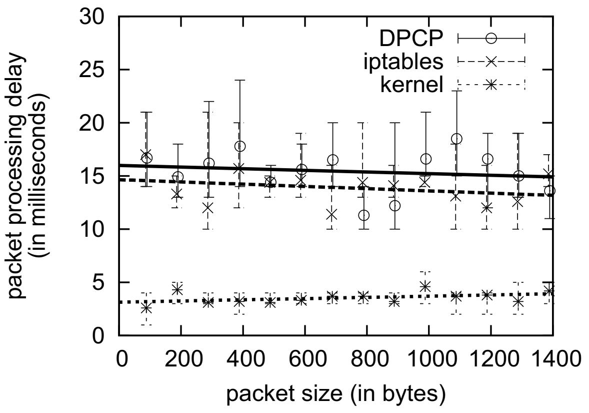

Fig. 8. Forwarding Delay. Error bars indicate minimum and maximum values measured.

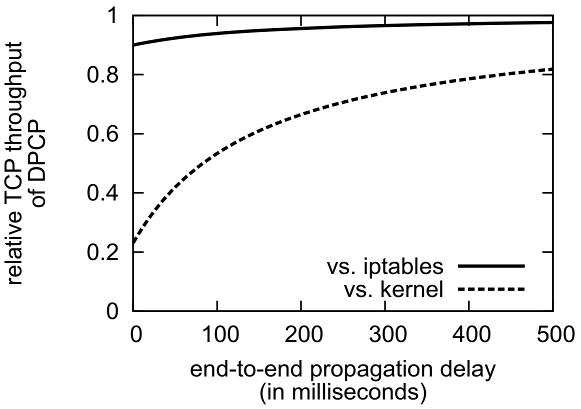

Fig. 9. Throughput Performance on 10-Hop Path.

shown for three different implementations: packet forwarding in the kernel, packet forwarding using *iptables*, and DPCP. Since our implementation operates in user space and receives packets through the *iptables* system, the difference between DPCP and *iptables* shows most accurately the overhead for enforcing credentials (roughly 1.55 ms per packet). If credential checks were implemented in the system kernel, the packet forwarding performance of the kernel would decrease similarly.

To show how little impact the credential check has on TCP traffic, we show a comparison of TCP throughput in Figure 9. These values are derived using the analytical TCP throughput model assuming a 10-hop path. As expected, in a comparison with a kernel implementation, DPCP achieves a lower performance. However, DPCP always achieves at least 90% of the throughput of *iptables* (and more for higher end-to-end propagation delays). This result indicates that our system can implement hop-by-hop credential checks with very limited impact on the performance of the network.

VI. Summary

Some of the main security problems in current networks can be tackled by capabilities-based networks that enforce traffic control on every hop. Our prior work has described to general ideas for such a system. In this paper, we present the implementation of a prototype system that demonstrates this protocol. Our evaluation results show that the cryptographic operations at the startup of the protocol are expensive and need to be optimized for large-scale deployment. However, once connections are established, the throughput performance of our protocol is nearly equal to that of conventional TCP. Thus, our protocol can provide secure networking with little overhead in the data path.

References

[1] S. Alexander, B. DeCleene, J. Rogers, and P. Sholander. Requirements and architectures for intrinsically assurable mobile ad hoc networks. In *Proc. of the 2008 IEEE Conference on Military Communications (MILCOM)*, San Diego, CA, Nov. 2008.

[2] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet impasse through virtualization. *Computer*, 38(4):34–41, Apr. 2005.

[3] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet denial-of-service with capabilities. *SIGCOMM Computer Communication Review*, 34(1):39–44, Jan. 2004.

[4] K. Argyraki and D. Cheriton. Network capabilities: The good, the bad and the ugly. In *Proc. of Fourth Workshop on Hot Topics in Networks (HotNets-IV)*, College Park, MD, Nov. 2005.

[5] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by default! In *Proc. of Fourth Workshop on Hot Topics in Networks (HotNets-IV)*, College Park, MD, Nov. 2005.

[6] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In VINI veritas: realistic and controlled network experimentation. In *SIGCOMM '06: Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 3–14, Pisa, Italy, Aug. 2006.

[7] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[8] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: taking control of the enterprise. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 1–12, Kyoto, Japan, Aug. 2007.

[9] D. E. Eastlake and P. E. Jones. US secure hash algorithm 1 (SHA1). RFC 3174, Network Working Group, Sept. 2001.

[10] A. Feldmann. Internet clean-slate design: what and why? *SIGCOMM Computer Communication Review*, 37(3):59–64, July 2007.

[11] R. L. Rivest. The MD5 message digest algorithm. RFC 1321, Network Working Group, Apr. 1992.

[12] J. S. Turner. A proposed architecture for the GENI backbone platform. In *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communication Systems (ANCS)*, pages 1–10, San Jose, CA, Dec. 2006.

[13] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.

[14] T. Wolf. Challenges and applications for network-processor-based programmable routers. In *Proc. of IEEE Sarnoff Symposium*, Princeton, NJ, Mar. 2006.

[15] T. Wolf. A credential-based data path architecture for assurable global networking. In *Proc. of the 2007 IEEE Conference on Military Communications (MILCOM)*, Orlando, FL, Oct. 2007.

[16] T. Wolf. Design of a network architecture with inherent data path security. In *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communication Systems (ANCS)*, pages 39–40, Orlando, FL, Dec. 2007.

[17] T. Wolf. Data path credentials for high-performance capabilities-based networks. In *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communication Systems (ANCS)*, San Jose, CA, Nov. 2008.

[18] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. *SIGCOMM Computer Communication Review*, 35(4):241–252, 2005.