SEAMS: A Signaling Layer for End-host-Assisted Middlebox Services

René Hummen, Jan Henrik Ziegeldorf, Tobias Heer, Hanno Wirtz, Klaus Wehrle
Chair of Communication and Distributed Systems
RWTH Aachen University
Germany
{hummen, ziegeldorf, heer, wirtz, wehrle}@cs.rwth-aachen.de

***Abstract*—On-path network elements, such as NATs and firewalls, are an accepted commonality in today's networks. They are essential when extending network functionality and providing additional security. However, these so called middleboxes are not explicitly considered in the original TCP/IP-based network architecture. As a result, the protocols of the TCP/IP suite provide middleboxes with the same information about data flows as packet-forwarding routers. Yet, middleboxes typically perform complex functions within the network that require additional knowledge. Inferring this knowledge from observing the sparse information available in network packets requires these devices to base their decisions on ambiguous or forgeable data. In this paper, we first discuss problems arising from insufficient information and identify the resulting informational requirements of middleboxes. We then propose SEAMS, a signaling layer that provides middleboxes with descriptive and verifiable data flow contexts in addition to the IP address and port information that many middleboxes use today. Specifically, SEAMS enables middleboxes to request and use detailed information about the host, application, and user that is accessible at the communicating end hosts. This information can then be used to provide more secure and richer middlebox functions in home and enterprise network scenarios. Our evaluation shows that SEAMS is a feasible addition to TCP/IP-based networks and that it scales well in the presence of multiple on-path middleboxes.**

I. Introduction

Middleboxes are special purpose network elements that *"perform functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host"* [1]. Middleboxes implement increasingly complex functions within the network that cannot be implemented in end hosts for technical or security reasons. Firewalling, network admission control, tunneling, and address translation are examples of such functions. These middlebox functions frequently require information about the origin, the destination, or the purpose of a data flow. However, the protocols of the TCP/IP suite offer middleboxes only the bare information required for forwarding packets towards the destination host and application. This information is neither cryptographically secured, nor does it answer important high-level questions, such as: i) *"Which user is accessing a network resource?"*, ii) *"Which application is causing the current data flow?"*, and iii) *"Which host is using the network?"*.

In many cases, inferring this *context* of a data flow by observing end-point addresses and port numbers is either impossible or inaccurate. For example, viruses and malware often use well-known port numbers and communication patterns of benign applications to avoid detection. Thus, middleboxes, such as firewalls, have to resort to costly mechanisms (e.g., deep packet inspection) to identify communication of malicious software. Approaches such as 802.1X [2] and VPN tunneling [3] aim at providing a part of the missing context to the network by tying network admission to user authentication. However, actions of different users on a multi-user system still remain difficult to distinguish within the network.

The gap between the *required* and the *available* information for the intended middlebox functions severely limits the potential of today's middleboxes. In contrast, the end hosts possess detailed information about the application and the user as the origin of network traffic. However, the TCP/IP network stack does not provide end hosts with the means to reveal this information to on-path middleboxes. Even if this information was revealed to middleboxes, verifying its authenticity inside the network remains challenging.

In this paper, we first analyze the availability of information about the origin of data flows as required by middleboxes such as firewalls and QoS systems. We then propose SEAMS, a signaling layer that provides middleboxes with expressive information about traffic origin. In contrast to related work [4], [5], our approach focuses on the question how end hosts can make descriptive end-point information available to middleboxes in a *verifiable* way. Furthermore, our signaling layer enables the dynamic negotiation of the required end-point information between end hosts and middleboxes for reasons of accountability and efficiency. Notably, in SEAMS, hosts do not need to signal end-point information to each middlebox on the communication path individually. Instead, middleboxes request and obtain end-point information within a single end-to-end handshake that is performed between the two communicating end hosts. These properties of SEAMS enable multiple middleboxes to efficiently acquire and use fine-granular information about the communicating end-points.

II. Middleboxes: Issues and Requirements

Today, many middleboxes, on the one hand, serve security-related purposes as firewalls, tunnel end-points and proxies. On the other hand, they extend the capabilities of a network as Network Address Translators (NATs) or Quality of Service (QoS) systems. Due to their important role in today's networks, one would expect middleboxes to work on detailed and precise information when performing their functions. Firewalls, for example, need to unambiguously identify end hosts, applications, and services as the origin of a data flow in order to decide which flows to permit and which to block. Likewise, QoS middleboxes require information about applications and users, their preferences and requirements to treat their traffic accordingly. However, the protocols of the TCP/IP suite provide middleboxes with little more than IP addresses, port numbers, and protocol IDs as easily accessible information. As a result, middleboxes must either approximate the *required* from the *available information* (e.g., guess applications and QoS requirements based on port numbers) or must violate layer boundaries by inspecting packet contents [6]. Neither of both options presents a clean, reliable, and efficient way to acquire detailed information about i) hosts, ii) applications and their requirements, as well as iii) users and their privileges.

A. Inadequate host context

Due to the lack of more meaningful host identifiers within the network, middleboxes use IP addresses to refer to hosts. Yet, as discussed by many locator/identifier-split approaches (e.g., HIP [7] and LISP [8]), IP addresses primarily refer to routable network locations rather than to hosts. As locators, IP addresses may be assigned dynamically or change depending on the location of the host in the network. In addition, middleboxes such as NATs or proxies modify IP addresses on the communication path. Thus, middleboxes cannot use these addresses as stable and global references to hosts.

Furthermore, IP addresses carry little semantic meaning. The ownership of an IP address typically conveys no additional information about a host. However, for middleboxes, the role of the host, its type (e.g., notebook or server), its owner, its administrator, or its privileges (e.g., authorized to use a specific network application) are important additional information. Inferring these properties from bare IP addresses makes the configuration of middleboxes complex and error-prone.

Finally, IP addresses have no built-in defense mechanisms against spoofing as they are not cryptographically verifiable. Hence, the security of IP-address-based rules at middleboxes largely depends on the underlying network topology. Host identification based on IP addresses should, therefore, only be used within small-scale networks (e.g., at home) or within tightly controlled environments (e.g., in data centers) where IP address spoofing or hijacking are improbable.

As IP addresses only provide a crude way to identify hosts, a useful *host context* should have the following properties:

Stability: It should be stable over time and independent from the location of the host and middleboxes within the network.

Expressiveness: It should simplify the process of providing meaningful host-specific information to middleboxes.

Verifiability: It should be hard to forge and enable middleboxes to authenticate the context information.

A host context with these properties would enable middleboxes to simply, securely, and unambiguously identify hosts as sources or destinations of data flows. However, since hosts are controlled by users and applications, middleboxes may need to further differentiate the cause and purpose of such flows.

B. Local scope of the application context

A networked host consists of the device hardware, the operating system, and running applications. The operating system has detailed information about which local application triggered which data flow in the network. However, once the data flow leaves the end host, this detailed information is reduced to IP addresses, port numbers, and a protocol ID. While certain well-defined ports hint at application types (e.g., port 21 for FTP), many port numbers do not convey meaningful information about the actual application and the purpose of a data flow. Many network operators meet this fact by using firewalls to restrict network traffic to WWW flows due to the lack of better classification mechanisms. However, more and more applications circumvent such restrictive firewall policies by using HTTP over TCP with port 80 in order to disguise as web-traffic [9], [10].

To determine the application context of a data flow despite the lack of meaningful packet header information, middleboxes may inspect the payload data of forwarded packets. However, such deep packet inspection requires middleboxes to understand the higher-layer protocols. While decoding the standard protocols of the TCP/IP suite may still be feasible, trying to parse the many transport and application layer protocols becomes impracticable and error-prone [6]. Hence, an *application context* should have the following properties:

Expressiveness: It should enable middleboxes to unambiguously identify applications based on information such as their application name.

Verifiability: It should allow to verify the authenticity of the application information.

Application contexts with these properties may serve as a basis for better traffic classification and can lead to improved and more efficient QoS treatment. Moreover, access control lists (ACLs) that restrict or permit the use of specific applications become easier to specify and more efficient to enforce. Hence, such application contexts can support administrators in controlling the network in a more fine-granular way.

C. Missing user context

Unlike host and application, the user who triggered a data flow (e.g., by opening a web page) does not have a representation in the TCP/IP protocol suite. Hence, middleboxes must assume that the user and the host (represented by the IP address) are the same. However, this assumption often is not valid, as most modern operating systems are multi-user systems that allow different users to operate a host. Parallel use of a system is more common than one might expect as remote login capabilities exist for most operating systems. Furthermore, processes initiated by one user may continue to run in the background after she logged out and another user logged in. Thus, data flows from a single host may have different users as their origin. This leaves middleboxes with no means to unambiguously map flows to the responsible user.

The need for a user identity within the network has led to widespread standard solutions such as 802.1X [2] authentication and VPN tunnels [3], which authenticate users before granting access to the network. However, the established user context is not sufficient for two reasons. Firstly, the design of these mechanisms may not allow for a distinction between different users on the same host. For example, 802.1X opens a (virtual) port at a switch and, thus, admits traffic on a per-host basis. Secondly, the established user identity is not passed on to middleboxes further along the communication path. Even if the user identity could be communicated to subsequent middleboxes, it might not be valid in different administrative domains that use different representations (e.g., user names) or different credentials (e.g., passwords) for the same user. Hence, a *user context* should have the following properties:

Granularity: It should enable middleboxes to determine the user at a per-flow instead of a per-host granularity.

Scope: It should allow a user to be authenticated in different administrative domains.

Visibility: It should be visible to middleboxes along the communication path.

A user context with these properties enables middleboxes to base decisions on information about the user rather than deriving possible user contexts based on application or host information. This is especially valuable in scenarios where the user who triggered a data flow is of special interest within the network. Access control, accounting, and logging are examples for middlebox functions that may benefit from meaningful user identities instead of technical identifiers (i.e., IP addresses or ports of a switch) that merely hint at the identity of a user.

III. Design of SEAMS

We now propose our Signaling layer for End-host-Assisted Middlebox Services: SEAMS. The goal of SEAMS is to provide middleboxes with host, user, and application contexts that fulfill the properties discussed in Section II. To this end, we introduce an *end-to-end handshake* that SEAMS-enabled end hosts perform between each other prior to the establishment of a new application connection. Middleboxes can actively participate in this handshake by *requesting* missing context information for the new connection from the communicating end hosts. End hosts locally look up the requested *end-point contexts* and signal these within the handshake. The signaled contexts are cryptographically bound to the end points, allowing middleboxes to *verify* and use the requested information for the provisioning of their respective function. Fig. 1 depicts an example network scenario, highlighting the context information SEAMS-aware on-path middleboxes could request from end hosts in order to improve their functions.

Fig. 1. Example network scenario with two administrative domains X and Y. Middleboxes are located on-path between the communicating end hosts. Firewall (FW) rules are based on the host context H_X and the user context U_X in domain X as well as the application A, whereas the NAT only requires a stable host identifier H_X. The QoS system prioritizes data connections based on the user context U_Y in domain Y and the application A.

We now give a high-level overview of the SEAMS signaling architecture and the signaling handshake. We then discuss the proposed host, user, and application contexts and show how SEAMS integrates in the existing network infrastructure.

A. High-level architectural description

The SEAMS architecture consists of two components: i) an *inspection layer* that monitors and controls all network connections at an end host or a middlebox, and ii) a *signaling layer* that allows middleboxes to request the context of a network connection and end hosts to answer these requests. As shown in Fig. 2, the inspection layer is located below the IP layer. Hence, all IP-based application traffic passes through our inspection layer. The signaling protocol runs on top of the transport layer. Thus, the signaling protocol appears like a regular application layer protocol to middleboxes that do not support SEAMS. Note that consumer-grade operating systems provide APIs that our inspection and signaling layers can use for packet inspection and transmission. Hence, SEAMS does not require modifications to the operating system of end hosts and can be installed and executed as a privileged user-space application. We now describe how the SEAMS layers interact on the different devices in our architecture.

On an end host, the inspection layer monitors the network stack for new outbound application connections. When detecting a new connection attempt (e.g., a TCP SYN), it delays the connection, i.e., the *data channel*, and notifies the signaling layer about the connection attempt. The signaling layer triggers a SEAMS handshake with the destination host of the data channel. Furthermore, it gathers the host, user, and application contexts (e.g., the user and application name) associated with the data channel at the operating system. For the sake of

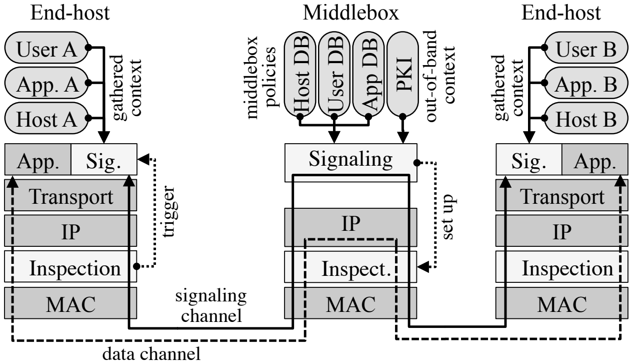


Fig. 2. SEAMS architecture. The inspection layer of SEAMS at the end host detects new application connections and triggers the SEAMS signaling layer. The signaling layer gathers connection information and signals them to the peer host. The inspection layer at an on-path middlebox passes SEAMS signaling messages to the signaling layer. The signaling layer then sets up the data channel at the inspection layer according to the exchanged information.

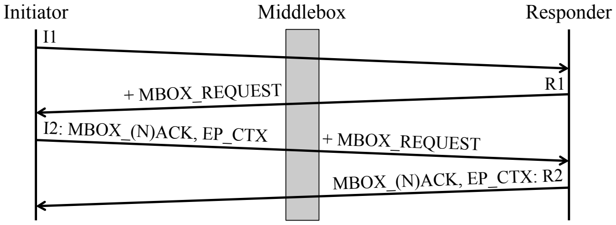


Fig. 3. SEAMS protocol handshake. Middleboxes add MBOX_REQUEST parameters to the HIP handshake in order to request context information. End hosts acknowledge or reject these requests and provide requested end-point contexts encapsulated in EP_CTX parameters.

clarity, we defer the detailed discussion about the signaling protocol and the context information to the following sections.

On-path middleboxes overhear the signaling handshake between the two end hosts and match the context information contained in the signaling messages against their local context policies. A firewall, for example, may check its ACL for a matching host identifier. If the available context information does not satisfy the configured policies, the middlebox signaling layer queries the missing contexts from the end host by appending a *middlebox context request* to the overheard handshake message. At the destination, the signaling layer processes the context requests and looks up the corresponding *end-point contexts*. Moreover, the signaling layer adds the requested contexts to the response message and sends it towards the initiator of the handshake. Back at the middlebox, the signaling layer processes the replied end-point contexts and uses them as a basis for the provided middlebox function. In case of a firewall, this means that it allows or denies the application connection according to the matching rule. To this end, the signaling layer instructs the inspection layer to forward or drop packets belonging to the corresponding data channel. As the inspection and the signaling layer reside on the same device, we require the paths of signaling messages and data packets to be the same for the network segments that use SEAMS (path-coupled signaling). After a successful handshake, the signaling layer at the initiating end host notifies the inspection layer to resumes the application connection.

B. SEAMS protocol overview

The SEAMS handshake is based on the Host Identity Protocol (HIP) [7], because it provides SEAMS with three key elements of our design. First, HIP introduces a new cryptographic namespace in the network stack that offers a stable and verifiable host identity. Second, it is an end-to-end signaling protocol with a signed message exchange affording verification by middleboxes. Third, on-path middleboxes can inspect HIP signaling messages and obtain the transferred signaling information in the message content. In previous work [11], we introduced a HIP protocol extension that protects middleboxes from replay attacks and allows them to securely verify HIP signaling messages. We now use these mechanisms in our SEAMS architecture as a building block to securely negotiate and provide the required context information between end hosts and on-path middleboxes.

In SEAMS, a new application connection triggers a HIP handshake between the two end hosts, the *initiator* and the *responder*. As shown in Fig. 3, we extend the last three messages of the four-way HIP handshake. Each on-path middlebox with missing initiator-side context information appends an MBOX_REQUEST parameter, i.e. a *middlebox context request*, to the first responder message R1. Specifically, it adds the parameter to the unsigned part of this message. The MBOX_REQUEST parameter contains a *request ID* in order to enable the middlebox to map the end host response to its own request. Hence, the middlebox must choose a unique request ID with respect to the possibly already included MBOX_REQUEST parameters in the message. Furthermore, the MBOX_REQUEST parameter consists of *request values* that indicate specific requests for end-point information needed by the middlebox function.

We distinguish between two *request priorities*, which are encoded in the last bit of a request value: critical and uncritical requests. Critical requests denote the minimal end-point contexts required for the provisioning of the middlebox function. Hence, an end host must satisfy these requests in order to successfully complete the handshake. Contrarily, uncritical requests query the end host for context information that may be beneficial in providing the respective middlebox function. For example, a QoS system may provide minimal service guarantees according to the needs of the signaled application and reserve additional link capacity for the connection if it also learns the identity of a high-privileged user.

When the initiator receives the R1 message, it first processes the message as an unmodified HIP message. It then checks for the existence of MBOX_REQUEST parameters. If the message contains parameters of this type, the initiator gathers the context requests of the on-path middleboxes (e.g., for the host and user identities). It may consult a local policy database in order to determine whether to signal the requested contexts or not. Depending on this policy decision and the priorities of context requests, the initiator adds MBOX_ACK or MBOX_NACK parameters to the second initiator message I2. Both parameter types contain the request ID of the corresponding MBOX_REQUEST parameter for identification at the middleboxes. MBOX_NACK parameters may optionally include the reason for the negative acknowledgment. This information can, e.g., be used by a network administrator to analyze and resolve failing handshakes. Finally, the initiator looks up the allowed contexts and adds each *end-point context* (EP_CTX) in a separate parameter to the I2. MBOX_(N)ACK and EP_CTX parameters are included in the signed part of the message. Hence, no on-path or off-path attacker can modify the signaled information. On receipt of the I2 message, an on-path middlebox inspects the message for an MBOX_(N)ACK parameter, that contains the echoed request ID, and the requested EP_CTX parameters. In case of an MBOX_ACK, it uses the acquired context information for the provisioning of the respective middlebox function. Our extensions to the HIP protocol handshake are symmetric for the context information negotiation between middleboxes and the responder.

C. *Verifiable end-point information*

A key requirement of SEAMS is the ability for middleboxes to verify the context information signaled by the end hosts. To provide this verifiability, we use public key cryptography in the design of SEAMS. Most importantly, a host or a user is represented by the public key of a public/private key pair, i.e., a *cryptographic identity*, as proposed by HIP [7] and UNA [12]. Each end-host system is assigned a cryptographic identity by its administrator, whereas users are responsible for their own key pairs (similar to SSH keys). Middleboxes can request the host and user public keys from end hosts in order to authenticate these communication end points. To allow middleboxes to verify the requested identities, the SEAMS signaling layer of the end host adds the requested identities to the response message and uses the identities to sign the message. The signature additionally allows middleboxes to attribute the content of the handshake messages to the signaled identities. Moreover, the public key signatures afford non-repudiation, allowing middleboxes to prove that certain communication has taken place, e.g., for accounting purposes.

In contrast to hosts and users, applications are digital entities that are executed on account of a user and processed by the operating system of a host. Specifically, the operating system controls their execution and can, for example, inspect the allocated memory of running applications and the files accessed by them. Due to these abilities, applications cannot ensure that a secret, such as a private key, stays disclosed from the operating system at all times. As a result, the application context is as trustworthy to middleboxes as the context of the host that the application is running on or the user who started the application. In the design of SEAMS, we account for this fact by modeling the context information of an application as statements that the end host makes about the application. We discuss such statements in the following section.

D. *Descriptive end-point information*

Besides presenting middleboxes with verifiable identities for hosts and users, end hosts in SEAMS can also provide middleboxes with descriptive end-point information about a data flow. Regarding descriptive contexts, we differentiate between i) properties stated by a *third party* and ii) properties stated by an *end host*. Third party context enables entities such as the network administrator to make statements about a host (e.g., type="file-server"), a user (e.g., privileges="lan-only"), and an application (e.g., name="firefox"). Third party contexts are encoded in certificates, which are signed by the third party. Middleboxes can acquire host and user certificates from a public key infrastructure, that may be located in the local network. Alternatively, middleboxes can request certificates as additional end-point context from end hosts during the SEAMS handshake. End hosts signal the requested certificates along the communication path subsequent to the signaling handshake. If the public key of the third party is set as trusted at a middlebox, the middlebox can verify the certificate signature and validate the host or user identity against it. Afterwards the middlebox adds the conveyed context to the context information gathered during the handshake.

Descriptive end host contexts are properties that are directly retrievable from the operating system. Examples for such properties include the name and the version of the operating system, the average bandwidth used by a user, and the number of open connections for an application. If these contexts are requested by an on-path middlebox, the signaling layer of SEAMS uses standard APIs provided by the operating system to look up such information. It then adds the contexts as EP_CTX parameters to the signed part of the handshake messages. As a result, descriptive contexts of an end host are cryptographically bound to the host or user identity. This binding enables on-path middleboxes to verify the signaled contexts by validating the signature contained in the signaling messages. A requesting middlebox may then use this verified information along with the cryptographic host and user identities for the provisioning of its function.

E. *Integration of SEAMS*

We now discuss how SEAMS integrates in today's end host systems and network infrastructures. First, we describe a typical usage scenario based on a SEAMS-aware firewall. Here, we assume that the communicating end hosts are already SEAMS-enabled. We then explain how SEAMS can be enabled on present end hosts and show how to equip for today's middleboxes that are legacy devices with respect to SEAMS.

Using SEAMS: To enable the use of cryptographic host identities in the network, the administrator needs to generate and store the cryptographic host identity when configuring the host system. Likewise, users generate their own identities and use them on the end host, where they are currently logged in. Hence, our proposed identities are generated and used in analogy to the server and user key-pairs employed in SSH.

In a basic usage scenario of SEAMS, the identity of a host can be registered with the network administrator similarly to the procedure of providing the MAC address for static DHCP configurations. User identities can be registered by performing a SEAMS handshake with a registration service in the network, supplying the username and password in order to bind the two identities. The network administrator then uses the registered identities in the rules of a SEAMS-enabled firewall to specify allowed host and user identities, while dropping all non-matching traffic. In contrast to IP and port-based firewalling, such a setup enables the firewall to distinguish users on a multi-user system. Furthermore, identity-based rules allow the firewall to permit data connections for specific hosts or users independent from their current point of network attachment (e.g., within company premises or at home).

In a more advanced scenario, the network administrator may additionally equip hosts and users with certificates that bind privileges to them. Furthermore, the rules of the SEAMS-enabled firewall are not restricted to identities, but rather define a verifiable identity in combination with descriptive host, user, and application contexts as properties of allowed data flows. Hence, a rule might state detailed contexts such as *(host-ID, application-name="skype")* or *(certificate-issuer, user-privilege="lan-only", application-type="jabber")*. As a result of the first rule, the Skype application can only access the network when executed on the host identified by the public key *host-ID*. The second rule, on the other hand, restricts Jabber communication for users with the *"lan-only"* property to the local network. Such expressive rules enable more fine-granular network access configurations than possible today, while making the configuration of middleboxes less error-prone than IP and port-based policies. This makes SEAMS a valuable approach for home and enterprise networks.

Deployment of SEAMS: Our design affords an installation of the SEAMS inspection and signaling layers on end-hosts as privileged user-space applications. Since SEAMS makes use of certificates and standard operating system APIs for context information lookups and packet inspection, no modification of the operating system is required. Furthermore, the use of HIP enables the integration of SEAMS in the network stack of a host that is transparent to other applications. As a result, no additional configuration or modification of applications are required. However, SEAMS may not be as easily installed on present middleboxes. Hence, these middleboxes are legacy devices that do not benefit from the additional end-point information and may prevent the correct forwarding of SEAMS messages. To enable the traversal of SEAMS signaling messages through on-path legacy middleboxes, we employ UDP-encapsulation for these messages. Thus, legacy middleboxes can continue to process packets based on IP addresses and ports and are not required to support the SEAMS signaling exchange. However, port-based policies may be needed to allow forwarding of SEAMS signaling messages.

SEAMS requires support at both communicating end hosts. However, when an end host in a SEAMS-enabled network aims to communicate with an end host located outside the local network, it may find that the peer host does not support SEAMS. To foster isolated or incremental deployments, the network may set up SEAMS proxies at the network borders that take over the role of such SEAMS-unaware end hosts in the signaling handshake, while forwarding packets on the data channel to the destination host. Consequently, these proxies allow SEAMS-enabled end hosts to provide SEAMS-aware middleboxes in the local network with additional context information despite the lack of support of the peer host. Likewise, SEAMS-aware middleboxes may provide SEAMS-unaware end hosts, which do not signal additional context information, with the same restrictive IP and port-based functions as legacy middleboxes. Hence, SEAMS is also applicable in scenarios that do not provide full support for our signaling approach.

IV. Evaluation

Our SEAMS prototype is based on the HIP for Linux (HIPL) [13] implementation and extends it with functionality for requesting and signaling our proposed context information. The prototype consists of the end-host components and a firewall implementation for middleboxes. The firewall grants or blocks traffic based on the content and validity of the signaled information. The inspection layer uses *iptables* for monitoring in- and outbound connections. When a new connection is detected on an end host, our implementation uses *netstat* to look up the application and user corresponding to the connection. We use this information to further look up user identities and end-point contexts (e.g., from certificates) and selectively provide this information to on-path middleboxes.

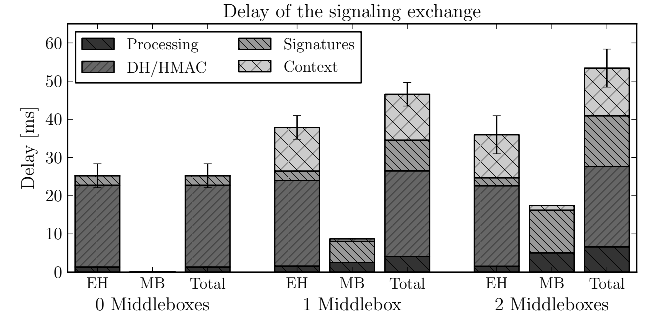
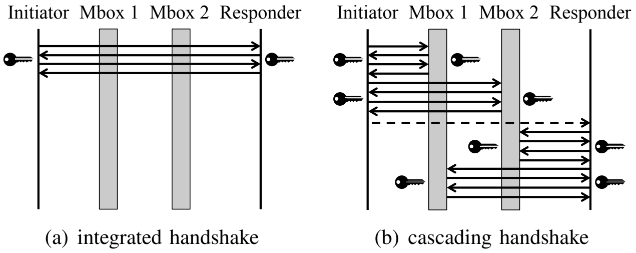


Fig. 4. SEAMS signaling delay for middleboxes requesting the end host identities and application contexts. EH represents the combined overhead on both end hosts, whereas MB depicts the overhead of on-path middleboxes.

Our test setup consists of two end hosts and two firewalls that are connected by a Gigabit switch. The end hosts are Intel Core i7-870 desktop computers and the firewalls are 500 MHz AMD Geode ALIX boards with 100 MBit/s Ethernet. All measurements are the means of 100 runs and show a maximum standard deviation of 5.0 ms. The four-way SEAMS handshake includes a network delay of two roundtrip times. As this delay is situation-dependent and network-specific, we do not consider it in our measurements.

Performance evaluation: Notably, SEAMS does not add a delay to the data channel once the signaling handshake is complete. However, the SEAMS handshake creates an additional delay when end hosts establish a new application connection. This delay depends on the number of on-path SEAMS-aware middleboxes as well as the type of context information that they request. Fig. 4 shows this delay for zero to two middleboxes that were configured to query both end hosts for the host identities (1024-bit RSA) and application names. To answer these middlebox requests, both end hosts combined introduce a total overhead of 37.88 ms. This overhead is composed of i) the host signature generation and processing on both end hosts (2.47 ms), ii) the lookup of the application contexts (11.41 ms), iii) the end-to-end security measures of HIP, namely the DH key-exchange and HMACs (22.41 ms), and iv) general HIP packet processing (1.59 ms). As a primary advantage of the end-to-end signaling in SEAMS, the end hosts only have to gather context information and perform cryptographic operations once per connection for all on-path middleboxes. Hence, the end-host overhead stays constant with an increasing number of on-path middleboxes (see "EH" bars in Fig. 4). Furthermore, the delay caused by SEAMS decreases to the overhead of the underlying HIP implementation (left bar), if the end hosts do not receive any context requests or if they are only queried for the host identity. Apart from context signaling, the use of HIP additionally enables end hosts to roam between networks and communicate securely.



(a) integrated handshake (b) cascading handshake

Fig. 5. High-level overhead comparison of the integrated and the cascading handshake. Keys represent cryptographic operations. The dashed line denotes an additional message to trigger a new connection on the responder side.

In such cases, the overhead of otherwise required signaling mechanisms (e.g., Mobile IP and IKE) can be omitted.

Each middlebox introduces an additional delay of approximately 8.7 ms on the signaling channel. This delay results from i) the signature verification of both host identities (5.58 ms), ii) end-point context processing (0.60 ms), and iii) general packet processing (2.51 ms). Hence, the total delay increases from 46.57 ms with one requesting middlebox to 53.42 ms for two requesting middleboxes. Notably, the per-middlebox overhead in SEAMS only occurs for the four-way end-to-end signaling exchange between the end hosts. As our signaling approach does not require direct signaling between an end host and a middlebox, the middlebox-related signaling delay is independent of the round-trip times encountered in the network. Hence, our prototype allows for several on-path middleboxes at a modestly perceivable per-middlebox overhead.

Certain on-path middleboxes may require knowledge about the user as the origin of a data flow. In this case, the verification of the users' identities for both end points requires 3.99 ms. A middlebox may further require third party host or user context. The corresponding certificate verification requires approximately 2.80 ms for a certificate that is directly issued by the third party (i.e., without intermediate certificates).

SEAMS allows for different levels of bit security by supporting a range of public key algorithms and key lengths. For 2048 bit RSA keys, the cryptographic costs increase from 2.47 to 11.22 ms at end hosts and from 5.58 to 15.86 ms at middleboxes. ECDSA (NIST P-256) offers shorter identities and signatures at a similar bit security, but it increases the delay at end hosts to 10.56 ms and to 88.03 ms for middleboxes.

In the design of SEAMS, we opted for an *integrated handshake* that consists of a total amount of four messages independent of the number of on-path middleboxes (see Fig. 5(a)). As a benefit, the signaling delay discussed above only occurs once during the connection establishment. However, the integrated handshake is affected by message space limitations when signaling extensive context information.

In an alternative approach to SEAMS, end hosts would exchange context information *individually* with on-path middleboxes in a *cascading handshake* (see Fig. 5(b)). Context information could be transferred on the (secured) data channel between the end host and the middlebox. However, while this approach does not exhibit the same message space limitations as SEAMS, the signaling overhead increases linearly with the number of on path middleboxes. Hence, the design of SEAMS trades off signaling space for scalability. To reduce the effect of the per-message space limitation, we could further extend the integrated SEAMS handshake with subsequent handshake messages similar to the signaling of certificates.

V. Security Considerations

In the following, we identify and briefly discuss attacks that an adversary Mallory can mount against SEAMS:

Impersonating a host or a user: Mallory could aim at misusing resources in a network with SEAMS-aware middleboxes. To this end, she needs cryptographic identities that authenticate her to on-path middleboxes. Generating her own identities does not suffice for in-network authentication because middlebox policies would only permit traffic from legitimate (i.e., registered) identities. Thus, Mallory may try to impersonate *another* legitimate host or user towards middleboxes. To succeed, she needs access to the respective private keys. If she were successful, Mallory could use the stolen identities on a host of her choice (e.g., a host with a compromised SEAMS implementation) to signal wrong flow contexts.

Hosts and users can effectively protect their identities against theft by using well-known measures such as encryption of their private keys or secure storage (e.g., in a smart card or a trusted platform module). Concealing the private keys from Mallory frustrates the aforementioned impersonation attacks.

Tampering with the end host: Mallory could try to tamper with the SEAMS layers at a compromised host in order to hide unauthorized traffic by signaling wrong context information. However, this would require Mallory to alter the behavior of kernel- or user-space components that are executed in privileged system contexts. Furthermore, the signaled host identity would allow a network administrator to identify the compromised host when a malicious action is detected.

Misusing an established data channel: Mallory could aim at using the data channels, that other hosts have established, for her own purposes. However, a data channel is bound to a SEAMS signaling channel by means of IP addresses, port numbers, and a protocol ID. Hence, her data packets would have to imitate these properties of the victim data channel. As a result, Mallory cannot communicate freely because her traffic is limited to the established source and destination addresses used in the legitimate data channel. This considerably reduces the usefulness of exploiting an established channel.

Full end-host context disclosure: MBOX_REQUEST parameters are transmitted in the unsigned part of SEAMS messages and do not contain end host verifiable information about the middleboxes. This enables Mallory to request all obtainable context information from an end-host. As Mallory can overhear EP_CTX parameters for legitimate context requests, SEAMS-aware end hosts should set up signaling policies to ensure that they do not provide more information to Mallory than they would signal to benign middleboxes.

DoS attacks against middleboxes: SEAMS requires middleboxes to perform cryptographic operations on signaling messages and to allocate additional state compared to today's middleboxes. Thus, Mallory may try to force middleboxes into excessive cryptographic operations or state allocations by opening multiple connections or by replaying legitimate signaling messages. We counter these attacks by using middlebox puzzles and nonces that we proposed for HIP [11].

VI. Related Work

Several other architectures aim for overcoming the shortcomings of today's networks with respect to middleboxes. NUTSS [4] and Pedigree [5] enhance middleboxes by providing them with additional end-point information.

NUTSS identifies communicating end-points by *(user, domain, service)* 3-tuples and makes this information available in the network. The key differentiation between SEAMS and NUTSS are the characteristics of the signaled end-point information. While cryptographic host and user identities in SEAMS afford verification of end-point contexts at middleboxes, all end-point information in NUTSS is non-cryptographic in nature. The authors of NUTSS mention that standard authentication protocols may be used on the signaling path, hinting at an authentication overhead that is similar to the cascading approach discussed above. Furthermore, NUTSS does not enable middleboxes to distinguish between hosts and users and does not provide them with the descriptive context information offered by SEAMS. Thus, NUTSS does not allow for similarly fine-grained middlebox policies. Finally, NUTSS performs name-based routing on the signaling channel and address-based routing on the data channel. As a result, the paths of both channels may diverge. Hence, NUTSS requires additional signaling on the data path to supply middleboxes with authorization information.

Pedigree is a system that provides middleboxes with host and application identifiers as well as a history of application level interactions across multiple end hosts (i.e., data provenance). As identified by the authors, Pedigree does not employ sufficient cryptographic protection for the signaled information to mitigate replay, forgery, and impersonation attacks. As SEAMS provides such protection, it could serve as a secure signaling channel for Pedigree's provenance information.

AIP [14], DOA [15], and UNA [12] are architectures that explicitly name hosts or users and expose this information on the communication path. However, in contrast to SEAMS, they only offer limited verifiable end-point information and no descriptive contexts to on-path middleboxes.

UPnP [16], NSIS [17], and Midcom [18] are frameworks that enable end hosts to set up state for a data channel at a middlebox. However, these approaches are limited to the authentication of a single end point (i.e., either user or host). Furthermore, their cascading signaling and authentication approach results in higher computational overhead and additional network delay compared to SEAMS.

VII. Conclusion and Future Work

In this paper, we showed that middlebox functions can be improved effectively by making expressive and secure end-point information available in the network. To this end, we introduced a signaling layer, SEAMS, that enables middleboxes to request missing end-point information within a signaling handshake between the end hosts. In response to these requests, end hosts signal cryptographic host and user identities as well as cryptographically bound host, user, and application contexts towards the network. This signaled information enables middlebox policies (e.g., ACLs and QoS rules) that are more meaningful, simpler, and more fine-granular than today's rules that are based on packet headers and packet content. Moreover, the SEAMS signaling handshake only takes place during the initial establishment of a new connection and scales well with multiple on-path middleboxes.

While we discussed desirable properties of end-host contexts for middleboxes, we did not focus on end-host-related aspects in this paper. As one such aspect, the use of our signaling layer may lead to privacy issues when revealing sensitive context information. Although our proposed context negotiation mechanism affords minimal context disclosure for end hosts with benign middleboxes, it only represents a first step towards a privacy-aware solution. We consider the design of mechanisms that enable the selective disclosure of privacy-sensitive context information to authenticated middleboxes and context blinding for unauthorized parties future work.

References

[1] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and Issues," RFC 3234, IETF, 2002.

[2] Institute of Electrical and Electronics Engineers, "802.1X-2004 - Port Based Network Access Control," 2004.

[3] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis, "A Framework for IP Based Virtual Private Networks," RFC 2764, IETF, 2000.

[4] S. Guha and P. Francis, "An end-middle-end approach to connection establishment," in *Proc. of ACM SIGCOMM*, 2007.

[5] A. Ramachandran, K. Bhandankar, M. B. Tariq, and N. Feamster, "Packets with provenance," Technical Report GT-CS-08-02, 2008.

[6] A. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," *Passive and Active Network Measurement*, 2005.

[7] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host Identity Protocol," RFC 5201, IETF, 2008.

[8] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID Separation Protocol (LISP)," draft-ietf-lisp-22, IETF, 2012.

[9] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is P2P dying or just hiding?" in *Proc. of IEEE GLOBECOM*, 2004.

[10] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing Skype Traffic: When Randomness Plays with You," in *Proc. of ACM SIGCOMM*, 2007.

[11] T. Heer, R. Hummen, M. Komu, S. Götz, and K. Wehrle, "End-host Authentication and Authorization for Middleboxes based on a Cryptographic Namespace," in *Proc. of IEEE ICC*, 2009.

[12] Y. Wang, C. Peng, J. Bi, and H. Hu, "UNA: a new internet architecture for user-level multi-homing and mobility," in *Proc. of the 6th International Conference on Future Internet Technologies (CFI)*, 2011.

[13] "HIP for Linux," [Online] Available at: http://launchpad.net/hipl.

[14] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable Internet Protocol (AIP)," in *Proc. of ACM SIGCOMM*, 2008.

[15] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker, "Middleboxes no longer considered harmful," in *Proc. of USENIX OSDI*, 2004.

[16] UPnP Forum, "UPnP Device Architecture 1.1," 2008.

[17] R. Hancock, G. Karagiannis, J. Loughney, and S. V. den Bosch, "Next Steps in Signaling (NSIS): Framework," RFC 4080, IETF, 2005.

[18] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan, "Middlebox communication architecture and framework," RFC 3303, IETF, 2002.