# A Framework for Consent and Permissions in Mediating TLS (MaRNEW Workshop)

*Oprescu, Peterson, Rooney*

Middleboxes have traditionally acted unilaterally, offering endpoints neither the means to consent to the actions being carried out nor even the means to detect the presence of such middleboxes. The increasing use of HTTP over TLS has required that middleboxes reveal themselves and participate directly in the establishment of security associations. For many middlebox deployment environments, the path of least resistance has been to simply install a root certificate in clients that allows middleboxes to man-in-the-middle all TLS traffic. However, this only works for those middlebox operators that have a close association with the owners of endpoints. Operators without such a relationship typically field HTTP proxies that, though they must reveal their presence explicitly [1], still can effectively capture or modify the content that endpoints encrypt. The Keyless SSL system [2] introduces a means for services to delegate the formation of TLS security associations to middleboxes; while this is immediately relevant to content delivery systems, it could have broader applicability. All-or-nothing disclosure of end user traffic to middleboxes is widely recognized to be problematic from a security perspective. [3]

Recent proposals including multi-context TLS (mcTLS) [4] and BlindBox [5] have attempted to provide a more nuanced positioning of middleboxes in the formation of security associations. mcTLS allows endpoints to create multiple “contexts” in a TLS session, in order to explicitly share read and/or write access to particular session content with a designating middlebox. The BlindBox proposal describes itself as “a first step towards a general protocol that will allow middleboxes and end-to-end encryption to coexist, without sacrificing the benefits of either,” and accomplishes this aim by having endpoints encrypt a separate copy of traffic for the benefit of middleboxes, which can be used to detect certain traffic elements without fully disclosing traffic content to middleboxes. These approaches typically require encrypting multiple copies of traffic, which increases protocol overhead and may lead to performance problems in mobile environments.

While the recent ramp up in encrypted HTTP traffic and the availability of the secure-only version of HTTP/2 have raised new challenges for middleboxes, trade-offs concerning consent and disclosure have been factored into network management protocols for some time. The trust models of DiffServ [6] and ECN [7] illustrate alternative strategies for endpoints and middleboxes to disclose information about the transport layer. In the DiffServ model, effectively an endpoint classifies traffic according to its own preferences, and discloses these preferences to the network in the hopes that operators will prioritize chosen traffic flows over others. This promises a better experience that simply sending undifferentiated traffic and allowing operators to discern (potentially through deep packet inspection) which packets to prioritize. ECN allows middleboxes to disclose to the endpoint information about the state of the network in the hopes that endpoints will alter the rate at which they are sending traffic. This in turn promises a better experience than simply dropping packets in a congestion scenario at the middlebox and letting the endpoint infer how it should react to such adverse conditions. More recently, the Mobile Throughput Guidance [8] proposal examines similar transport issues for mobile networks, and offers a new way for middleboxes to disclose to endpoints information that will guidance on transmission.

Underlying all of these strategies are varying assumptions about consent and permissions. Without understanding what the actors in the architecture are willing to disclose, and why the other actors should trust and act upon these disclosures, these proposals may not be aligned with the actual deployment needs of operators or users. In order to surface these tacit assumptions, a framework and classification is required that encompasses the actors, their requirements, and their incentives.

## The Framework

The major change that ubiquitous TLS introduces to the architecture is the requirement for one or both of the endpoints to explicitly consent to the presence of a middlebox. Trust and imperceptibility are fundamentally incompatible: any middleboxes which must remain undetectable by endpoints unavoidably cannot be trusted by these endpoints. This is crucial in the mobile environment especially, as so many operators deploy transparent HTTP proxies today to execute a variety of security and management policies.

In some models, responsibility for determining this trade-off may devolve to application designers. In others, end users may make implicit or explicit policy decisions, on a per-middlebox or even on a per-session basis. Once the consent is granted, it may be total consent (allowing the middlebox to arbitrarily alter traffic), or consent to only specific permissions.

### Actors within the Framework

For HTTP, the actors and their devices include: the **End User**, operator of a client endpoint; the **Transit Operator**, who deploys middleboxes (mcTLS further separates middleboxes into “readers” and “writers,” and provides a further category for potential attackers); and the **Service**, which runs a server endpoint.

## Classification of Approaches

Some qualities of the traffic depend directly on the application type. For example, for bulk media downloads, the traffic is elastic, while for real-time media, latency management is crucial.

### 1. Preconfigured Client Consent: Client-side Root Certificate

**Consent:** Client side consents (by virtue of agreeing to root cert install) to grant the middlebox read and write access to traffic. Presence of the middlebox in the security association is transparent to the server.

**Permissions**: Full disclosure of all traffic to the middlebox, which can arbitrarily modify traffic.

### 2. Dynamic Client Consent: Explicit Proxy

**Consent**: Client side learns the presence of the middlebox after attempting to contact the service, and receives a certificate for the middlebox. At this point, the client can proceed with the connection or not, and by doing so, explicitly grants consent to the proxy.

**Permissions**: Full disclosure of all traffic to the middlebox, which can arbitrarily modify traffic.

### 3. Preconfigured Service Consent: Keyless SSL

**Consent**: Service side consents (by virtue of implementing the Keyless SSL service) to give the middlebox read and write access to traffic. Presence of the middlebox in the security association is transparent to the client.

**Permissions**: Potentially full disclosure of all traffic to the middlebox, which can arbitrarily modify traffic.

### 4. Partial Read-only Consent: BlindBox

**Consent**: The middlebox relies on the endpoints themselves to mutually assess the correctness of the setup and to make sure that the middlebox is receiving its own encrypted copy of the traffic. In this architecture, the endpoints give partial read-only permission to the middlebox (MB) by trusting the Rule Generator (RG) who is a separate entity.

**Permissions**: The middlebox is permitted only to inspect the traffic for particular keywords that match the attack rules established by the RG. Potentially, the architecture permits a more complex setup where upon detection of an attack, it is possible to disclose some of the user content to the MB.

### 5. Partial Read/Write Consent: mcTLS

**Consent**: Both client and service side implement mcTLS. A sender of traffic on either side assigns components of the traffic to particular contexts; assignment to a context grants explicit permissions to the middlebox.

**Permissions**: Fine-grained permissions are possible. mcTLS stipulates that "protocol designers need to standardize how their applications will use mcTLS." The example in 4.1 explains that if contexts are set on a per-section basis, for HTTP, "four contexts will be sufficient: request headers, request body, response headers, and response body," though the authors concede that "you could imagine extreme cases in which each HTTP header has its own access control settings."

## Conclusion

Mobile operators in the HTTP/2 environment will continue to want to deploy proxies, and in order to become a part of the security association between endpoints, operators will need to win the consent of one or the other. As mobile handsets belong to the customers of mobile operators, it is natural to seek an approach where the operator uses its business relationship with customers to win consent. In the face of widespread pervasive surveillance, however, a large number of users might decide to not opt-in.

If deployment of Keyless SSL becomes widespread, mobile operators could approach websites is the position of a Keyless SSL server, which would interface with the Key Server of the websites. It is however unlikely that deployment of Keyless SSL would ever become truly ubiquitous, and thus mobile operators must either forgo proxies for website without a Key Server, or not permit traffic to reach those websites. Because of the all-or-nothing permissions awarded by Keyless SSL, some web sites would be unlikely to consent to this arrangement with mobile operators as well.

It is conceivable that an approach like mcTLS could be combined with a dynamic explicit proxy notification, whereby clients learn of the presence of middleboxes during TLS negotiation and can then generate a "context" that grants middleboxes appropriate access. The major missing piece in this case would be the negotiation protocol that enables the middlebox to a) identify itself, further b) enumerate the components of traffic that middleboxes need to read or write, and finally c) explain why it performs these functions. Users could then set policies that grant or withhold access to components of traffic as they deem appropriate.

[1] Explicit Proxy. https://tools.ietf.org/html/draft-mcgrew-tls-proxy-server-01

[2] Keyless SSL. https://blog.cloudflare.com/keyless-ssl-the-nitty-gritty-technical-details/

[3] SEMI workshop. https://www.iab.org/activities/workshops/semi/

[4] mcTLS. http://conferences.sigcomm.org/sigcomm/2015/pdf/papers/p199.pdf

[5] BlindBox. https://eprint.iacr.org/2015/264.pdf

[6] DiffServ. https://tools.ietf.org/html/rfc2474

[7] ECN. https://tools.ietf.org/html/rfc3168

[8] MTG. https://tools.ietf.org/html/draft-flinck-mobile-throughput-guidance-02