OPSEC Working Group
Internet-Draft
Intended status: Informational
Expires: December 5, 2020

E. Wang
A. Ossipov
Cisco Systems, Inc.
R. DuToit
Broadcom

June 03, 2020

TLS Proxy Best Practice

draft-wang-opsec-tls-proxy-bp-00

Abstract

TLS proxies are widely deployed by organizations to enable security features and apply enterprise policies. This document defines a TLS proxy and discusses a wide range of security requirements to guide TLS proxy implementations.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A TLS proxy refers to a set of network functions and products that intercept TLS sessions per an organizational security policy. A TLS proxy is deployed between endpoints such as TLS clients and servers. Based on an organizational policy, it may proxy a TLS handshake by terminating it on the client side and starting a new handshake with the server. As a result, the TLS proxy is able to decrypt the traffic from each side of the TLS session for various purposes, and then optionally re-encrypt the traffic before forwarding it to the other side of the session. A TLS proxy may leverage the session

handshake information as well as decrypted traffic data to satisfy organization's policy requirements.

```
                     +----------------------------+
                     |     Application Stack      |
                     +----------------------------+

                    ...............................
+-------------+     . +----------+-----+----------+ .       +-------------+
|             |    .  |          |     |          |  .      |             |
|             |    .  |TLS Server|Proxy|TLS Client|  .      |             |
| TLS CLIENT  |-------|   Stack  |     |   Stack  |--------| TLS SERVER  |
|             |    .  |          |     |          |  .      |             |
|             |    .  +----------+-----+----------+  .      |             |
+-------------+     .         TLS PROXY             .       +-------------+
                    ...............................
                     +----------------------------+
                     |       Network Stack        |
                     +----------------------------+
```

Figure 1: TLS Proxy

Industrial and academic studies have been conducted on TLS proxies and the associated benefits and risks [SECURITY_IMPACT] [APPLIANCE_ANALYSIS] [PROXY_DETECTION]. Readers are encouraged to refer to those studies to better understand the trade-offs in the design, implementation and deployment of a TLS proxy.

This document does not attempt to establish a position on whether a TLS proxy is "good" or "bad". However, it is important to have a clear set of requirements for a TLS proxy implementation, given its sensitive nature with regards to the TLS client and server data being intercepted, and the fact that many vendors and providers have offered such functions to their customers.

This document specifies such requirements and best practices for implementing TLS proxy functions. Discussions and guidelines in this document cover proxy for TLS 1.3 [RFC8446] and earlier versions (e.g. [RFC5246]).

A detailed explanation of use cases and functions utilizing TLS proxy can be found in [TLS_IMPACT]. This document does not address any malicious use of the proxy techniques. However, readers would be able to benefit from the discussion for a better understanding and mitigation against unintentional or deliberate misuse. Last but not least, we acknowledge that some of the problems that a TLS proxy solves can be addressed through other methods; however, this document assumes that the reader has already made an implementation decision to pursue a TLS proxy network function.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

We adopt the following definitions for TLS proxy deployments in the network.

An outbound TLS proxy serves a scenario where a device that performs the TLS proxy function is in the same administrative domain as the TLS client, and the TLS server is located in an external zone such as the Internet. Usually the goal is to protect the client endpoint and the organization by controlling application behaviors and enforcing an acceptable use policy for the organizational network.

An inbound TLS proxy is deployed in front of one or a set of application services. A network device that implements the TLS proxy function is located in the same administrative domain as the application which provides a network service to clients. Usually it is not predictable or controllable which TLS client would initiate a connection.

4. Security and Privacy Requirements

A TLS proxy acts as both a TLS server (to the TLS client) and a TLS client (to the TLS server). Its design MUST abide by the general TLS server and client endpoint requirements. This section does not repeat the requirements from [RFC8446] and [RFC5246], including those on middleboxes in the "Protocol Invariants" section. It assumes full compliance with those TLS RFCs. Instead, this section highlights those specific to the proxy behavior.

4.1. Securely provision within the administrative domain

The TLS proxy MUST be provisioned as a trusted party for the TLS client. It MAY also be provisioned as a trusted party for the server. In most cases, only the client side needs to be provisioned for TLS server authentication, excluding TLS applications that use client certificate authentication. Given that client certificate authentication typically employs unique per-client certificates, it is impractical to provision such client-specific certificates on the TLS proxy and each application server at scale.

During a TLS handshake, the proxy needs to supply a certificate on behalf of the server with the same identity (e.g. subjectAltName) as the server. This is to ensure the same client-side experience while the proxy negotiates a session with the client.

4.1.1. Outbound provisioning

For the outbound TLS proxy scenario, the administrator has no control of the TLS server side since it is typically outside of their administrative domain. The administrator may provision an enterprise CA certificate on each client, either into the shared trust store of the operating system or individual applications, such as a web browser; describing a specific provisioning process falls outside of the scope of this document.

The proxy uses the CA certificate to issue a proxy certificate for each of the unique TLS servers that are accessed through it. Once the proxy generates a certificate on behalf of a particular server, these are typically cached locally and re-used for connections to the same server for some period of time.

It has to be noted that this scenario may not work if the application uses client certificate authentication or if the client obtains and stores the actual server's public key and the corresponding certificate through a network path that does not include the TLS proxy in question (a.k.a. certificate pinning). In certain cases, the administrator may be able to configure the client to accept an alternative CA or the trust list from the operating system. However, if such options is not available, the proxy SHOULD be able to either block the connection or bypass decryption based on a configured policy.

The fact that TLS sessions that bypass decryption on the TLS proxy could still use post-handshake authentication (TLS 1.3) or renegotiation (TLS 1.2 and below) does not negate the responsibility of the TLS proxy to enforce secure policy using the visible elements of the TLS sessions. However, those measures are limited thus new methods are required to be developed for such scenarios. The increased adoption of application level encryption also calls for alternative technologies to be developed, which is outside the scope of this document.

4.1.2. Inbound provisioning

Provisioning for inbound TLS proxy is much more straightforward. The administrator may import certificates and key pairs exported from a server into the proxy. Alternatively, the proxy may need to periodically retrieve server key pairs and corresponding certificates

from a shared secure repository (e.g. a Hardware Security Module (HSM)). Frequent server key rotation helps to minimize security exposure if a participating TLS proxy device is compromised. The proxy may also enroll for a certificate on behalf of the server. Once the proxy has been provisioned, the TLS client will be able to complete TLS handshakes with the server through the TLS proxy.

4.2. Maintain security posture and limit modifications

Because a TLS proxy effectively builds two TLS sessions, one with a TLS client and one with a server respectively, the two sessions MUST maintain the same or higher level security posture (such as using stronger ciphers) than the direct session between the client and the server. The TLS server and client stacks in the TLS proxy MUST be conformant to [RFC8446] and [RFC5246].

The proxy MAY remove cipher suites from a client-initiated Client Hello message, add new cipher suites, and re-order them in the proxy-initiated Client Hello message. The purpose may be to increase the security posture of the combined sessions by removing weak cipher suites, or to optimize decryption and encryption performance (e.g. due to hardware acceleration capabilities), or from other business policy criteria. These changes MUST NOT negatively impact the security posture of the session.

TLS proxy stacks MAY provide user configurable options, such as an option to accept self-signed server certificates, an option to let the handshake continue with expired but otherwise valid server certificates, etc. However, the administrator MUST be provided with full information about any associated risks, such as from accepting a self-signed server certificate. The administrator MUST have complete control over the use of such options.

4.3. Be secure by default

The TLS proxy configuration and policy should be secure by default. The configuration of a TLS proxy requires special knowledge, and few users have the skillset to do it securely and correctly. TLS proxy vendors SHOULD educate users and TLS proxy developers SHOULD anticipate areas of configuration that could potentially create confusion. TLS proxy developers SHOULD simplify decisions that a user must make as long as sufficient configuration flexibility is maintained. A product that implements a TLS proxy MUST also provide default configuration settings that guarantee a high level of security. Configuration choices that deviate from the secure defaults SHOULD be presented alongside clear explanations of the impact of each option.

4.4. Use independent key material

The server and client sides of a TLS stack in a TLS proxy MUST negotiate key material independently. The proxy MUST NOT reuse and propagate key material or nonces received from the TLS client or the server. The TLS proxy MUST NOT inadvertently reuse the same random values for different TLS sessions; for example, the random generator MUST be fork-safe.

4.5. Protect against known vulnerabilities

Malicious entities rapidly exploit vulnerabilities in applications and protocols. TLS proxy vendors, and the developers in particular, SHOULD actively track vulnerabilities and respond with fixes in a timely manner. Issues in TLS stack implementations, or the TLS protocol itself, usually have far-reaching implications because other applications depend on the confidentiality, authentication, and data integrity properties of TLS. TLS proxy developers should go beyond just reading technical news articles, and should also follow discussions about current and future TLS related standards, track changes in open source TLS stacks as well as major open source TLS endpoint applications (e.g., Chrome, Firefox), and be familiar with application level protocols, especially in the context of how endpoint applications integrate with the TLS protocol.

4.6. Detect and handle protocol version downgrade markers

An in-path attacker may downgrade a TLS session to a protocol version lower than what is supported by both endpoints in order to exploit some known vulnerability in the lower version. The recommended downgrade protection mechanism for TLS 1.2 is the TLS_FALLBACK_SCSV, as described in [RFC7507]. The TLS 1.3 downgrade detection mechanism in [RFC8446] utilizes markers in the Server Hello random field. A TLS proxy MUST implement both mechanisms, correctly process downgrade markers sent by [RFC8446] compliant TLS servers, and be able to generate such markers toward a compliant TLS client.

The TLS 1.3 capable proxy MUST ignore the TLS 1.3 downgrade marker from the TLS 1.3 server if it has generated a TLS 1.2 ClientHello. Similarly, if the TLS 1.3 capable proxy prefers to generate a TLS 1.2 ServerHello, it MUST include the TLS 1.3 downgrade marker in response to a TLS 1.2 ClientHello, and it MUST NOT include the marker in response to a TLS 1.3 ClientHello.

4.7. Implement special measures to handle session resumption

If a TLS client attempts to resume a TLS session with an identifier (ID/ticket/PSK) that is not known to the TLS proxy then the TLS proxy MUST NOT indiscriminately exempt those sessions from decryption. At a minimum, the TLS proxy SHOULD enforce a configured policy for such sessions.

Should a TLS client use another network path to the TLS server, or should the TLS proxy be removed from its on-path network position then TLS session resumption attempts using TLS key material previously negotiated between the TLS client and TLS proxy SHOULD NOT result in TLS session resumption failures on the TLS server. A naive example of this is a TLS proxy that propagates the session ID generated by the TLS server to the TLS client, resulting in the same session ID referring to different master keys on the two sides of the TLS proxy and creating the potential for a decrypt_error alert if the TLS proxy is not involved in the TLS client's resumption attempt.

Applications that send TLS 1.3 early data rely on specific knowledge about the security properties of said payload. A TLS proxy most likely does not have access to the application level security properties of the payload, which implies that the TLS proxy MUST NOT inadvertently upgrade the security properties of early data received from the TLS client by forwarding it to the TLS server as post-handshake payload. The TLS proxy upper network stack SHOULD take into account the security properties of the decrypted early data as part of payload processing.

4.8. Adapt to protocol changes

The TLS proxy MUST minimize protocol ossification between the TLS client and server, specifically following the TLS protocol extensibility guidance from the Protocol Invariants section of [RFC8446]. The TLS proxy SHOULD be readily updateble, so that when ossification problems are discovered, they can be fixed.

The TLS proxy SHOULD be able to identify the start of a new TLS handshake. The static approach is through a pre-configured list of destination ports such as 443, 8443, etc. Dynamic approaches include parsing the application layer protocols to identify the STARTTLS message, and using signatures to identify TLS handshake messages. Those functions MAY be provided by the underlying platform, not part of the TLS proxy implementation. However, the TLS proxy MUST conduct proper TLS protocol checks to avoid false identification of TLS handshakes, while taking special care not to contribute to protocol ossification. Implementations SHOULD perform a consistency check for

TLS sessions started within a tunnel or following a previous transaction, such as an HTTP CONNECT request.

4.9. Respect regulations and privacy

Due to the privileged network location of the TLS proxy it potentially has access to a user's data such as PII (Personally identifiable information) and PHI (Protected Health Information). The TLS proxy MUST act responsibly to protect the privacy of the user and SHOULD allow masking of personal data by all the security tools that touch the decrypted payload. A product implementing TLS proxy SHOULD also allow to limit interception of certain categories of data, which could include evaluation of risk level and geolocation.

4.10. Respect sensitivity of decrypted payload

The TLS proxy SHOULD, similarly to respecting privacy and regulations, respect the sensitivity of the decrypted payload. Unless specifically required by an upper-level function per the organizational security policy, the decrypted payload MUST NOT be stored on the product that implements TLS proxy beyond the lifetime of the TLS session. The decrypted data SHOULD be securely destroyed upon completing any necessary inspection functions of that particular application message. The TLS proxy SHOULD also provide an option that guarantees that changes to decrypted content are not propagated to the endpoints, which would limit the impact of misbehaving upper-level functions while also simplifying compliance with regulations.

4.11. Enforce certificate validation standards

A TLS proxy MUST maintain a local certificate truststore populated with well-known public CA certificates. This trust store MUST be regularly updated by the product vendor in an automated manner, similarly to how commonly used web browsers manage it. It MUST allow an administrator to populate this store with locally trusted certificates or remove an existing certificate from the store. The proxy MUST validate a presented server's X.509 certificate against this trust store according to [RFC5280], including but not limited to validating the certificate path and checking the stated validity range.

The TLS proxy SHOULD also validate the certificate against a locally configured or specified Certificate Revocation List (CRL) repository, and MAY also use online status checking as specified in [RFC6960]. The TLS proxy SHOULD favor in-band OCSP over the out-of-band [RFC6960] mechanism, as specified in [RFC6066] and [RFC8446]. The

TLS proxy SHOULD provide an option to configure a policy for handling certificate validation failures and specifying exceptions.

Client applications, especially browsers, also validate a server name against names listed in the presented X.509 certificate. TLS proxies SHOULD follow the guidelines in [RFC6125] and MAY provide an option to enforce checks that would prevent look-alike (e.g. Cyrillic) international domain names from spoofing legitimate domain names.

4.12. Provide a secure operating environment

As a single device in the network that can decrypt encrypted transmissions, the TLS proxy would be a high value target for a bad actor. Therefore, a device implementing the TLS proxy function MUST provide a secure environment. It MUST comply with any organizational product security baseline requirements to achieve a high level of product integrity and trust. While an exact list of such product requirements may vary from implementer to implementer and lies outside of the scope of this document, some commonly acceptable protection elements include:

- Hardware and software authenticity attestation (Secure Boot)
- Digitally signed software images and configuration files
- Utilize a hardware security module, such as Trusted Platform Module (TPM), when appropriate and available
- No predefined authentication credential (password, certificates, keys)
- Secure storage of keys, passwords, and other sensitive data
- Use well-established and validated cryptographic libraries
- Use an effective random number generator [RFC4086]

5. Usability and Functional Requirements

5.1. Provide the ability to enforce policy without intercept

A TLS proxy MAY facilitate an ability to enforce security policies without the need for intercepting the handshake. Examples are deny or blacklist policies based on an observed Server Name Indication (SNI), cipher suites and certificates exchanged during the handshake (when in the clear).

The TLS proxy MUST use caution when applying security policies other than deny or blacklist based on the TLS handshake information. The TLS proxy SHOULD NOT use the handshake information for permit, whitelist, or security exemptions. This is because the information

from the handshake may not accurately reflect actual intentions or content carried in the encrypted session.

5.2. Use caution when enforcing policy with SNI

TLS proxy implementations tend to utilize the hostname information in the SNI for policy decisions. However, SNI by itself is not a trustworthy policy information element.

There are three categories of TLS clients when it comes to SNI handling:

1. Those who properly enforce hostname checks. Those TLS clients will populate the SNI field with the right name and check it. The SNI information from those TLS clients are trustworthy.

2. Those who do not properly enforce hostname checks. Those TLS clients will probably populate the SNI field but won't check it. But note that it is not that they are not checking SNI against the hostname but rather that they are not checking the certificate against the hostname.

3. Those who are intentionally trying to subvert the checks. The TLS client may conduct this behavior on purpose to circumvent the organization's access policy. For instance, it may deliberately set the SNI with a fake hostname with "good enough" reputation. Because [RFC6066] does not mandate a TLS server to abort the handshake when the server does not recognize the server name, the handshake may continue with the conformant server sending a certificate that contains the real hostname with a low reputation.

In an organization's network, it is unknown to the TLS proxy which category a TLS client falls into. Therefore, even if not intercepting the TLS session, the TLS proxy SHOULD passively observe the handshake until its successful completion before acting on the data from the handshake.

By doing so, the TLS proxy helps protect those TLS clients in category 2, and is able to detect the evasive clients in category 3.

It should be noted that in a variation of category 3 when both the TLS client and server are non-conformant, the TLS proxy will not be able to detect the malicious behavior passively even if it observes the handshake to its completion. In this scenario, the non-conformant server colludes with the TLS client by responding with a certificate containing a matching hostname presented in the SNI.

The above scenarios apply even if the SNI is encrypted by the TLS client ([ESNI]).

5.3. Selectively decrypt based on policy

A TLS proxy SHOULD be able to make policy decisions on whether to decrypt a particular session or not. Such a policy is referred to as "decryption policy". Decryption policy allows the TLS proxy to selectively decrypt certain sessions while exempting others, for efficiency, privacy and compliance considerations.

A decryption policy decision MAY be made based on the server certificate or other trustworthy parameters. To verify possession of private keys that are associated with a particular server certificate, the proxy SHOULD complete an out-of-band TLS handshake with the same TLS server IP address and TCP port as targeted by the TLS client.

A decryption policy decision MUST NOT be made solely based on an SNI extension for reasons discussed in the previous section.

The TLS proxy SHOULD provide a configurable option for how to handle sessions that it is not able to decrypt.

The TLS proxy MUST be able to disengage from a proxy session and allow the TLS client and server to negotiate a new session directly. A use case is to avoid decryption of certain types of traffic such as banking, as required by industry regulations.

There are several points where the TLS proxy can make a "not-to-decrypt" or "disengagement" decision. The decision MAY be made based on identifying the server by its IP address or the certificate or other techniques. In any of the cases, the user on the client side SHOULD NOT experience any disruption (e.g. a "connection lost" error).

A TLS proxy MAY decide to decrypt a session that is supposed to be denied (such as a request for a blocked category) for the purpose of showing the client a User Information Page to inform the client that this session has been denied. The TLS proxy MUST terminate the TLS session immediately after sending the notification, so that no additional data is transferred.

5.4. Limit performance impact

A TLS proxy SHOULD limit the impact of its functionality on network throughput, latency and jitter. Algorithm selection should be driven

by security as well as performance, such as using X25519 as a named curve in the calculation of a TLS 1.3 Client Hello key_share value.

TLS proxies SHOULD cache locally issued server certificates for a period of time in order to service similar sessions without the additional performance impact from re-issuing the certificate. A specific timeout for certificate cache entries MUST be configurable by the TLS proxy administrator.

6. Security & Privacy Considerations

This document is about security related requirements for TLS proxy implementations. A TLS proxy enables organizations to apply security functions and enforce organizational policy on the traffic. In that sense, the TLS proxy is part of an integrated security solution. The TLS proxy could become a single point of compromise in the organization's network. Implementations must satisfy the requirements described in this document to properly protect the TLS proxy while facilitating other security functions.

This document also lays out privacy requirements for TLS proxy implementations. Although a TLS proxy does not introduce new or modify existing TLS protocols that would change the privacy posture of the protocols, it is prone to introducing the risk of privacy invasion given its access to decrypted data. Implementations MUST follow the privacy practices described in this document to ensure that the TLS proxy does not weaken the privacy posture of the overall system.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC7507] Moeller, B. and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks", RFC 7507, DOI 10.17487/RFC7507, April 2015, <https://www.rfc-editor.org/info/rfc7507>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

8.2. Informative References

[APPLIANCE_ANALYSIS]
Waked, L., Mannan, M., and A. Youssef, "To Intercept or Not to Intercept: Analyzing TLS Interception in Network Appliances", 2018,

<https://users.encs.concordia.ca/~mmannan/publications/enterprise-interception-asiaccs2018.pdf>.

[ESNI] Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "TLS Encrypted Client Hello", draft-ietf-tls-esni-07 (work in progress), June 2020.

[PROXY_DETECTION] O'Neill, M., Ruoti, S., Seamons, K., and D. Zappala, "TLS Proxies: Friend or Foe?", 2016, <https://isrl.byu.edu/pubs/imc2016.pdf>.

[SECURITY_IMPACT] Durumeric, Z., Ma, Z., Springall, D., Barnes, R., Sullivan, N., Bursztein, E., Bailey, M., Halderman, J., and V. Paxson, "The Security Impact of HTTPS Interception", 2017, <https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_04A-4_Durumeric_paper_0.pdf>.

[TLS_IMPACT] Cam-Winget, N., Wang, E., Danyliw, R., and R. DuToit, "Impact of TLS 1.3 to Operational Network Security Practices", draft-camwinget-opsec-ns-impact-00 (work in progress), May 2020.

Acknowledgments

The authors thank Bill Hudson and Tobias Mayer for their contributions. The authors thank Flemming Andreasen, Richard Barnes, Nancy Cam-Winget, Alissa Cooper, Jay Iyer, James Kasper, David McGrew, Eric Rescorla and Eric Vyncke for their comments.

Authors' Addresses

Eric Wang
Cisco Systems, Inc.

Email: ejwang@cisco.com

Andrew Ossipov
Cisco Systems, Inc.

Email: aossipov@cisco.com

Roelof DuToit
Broadcom

Email: roelof.dutoit@broadcom.com