

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

F. Andreasen
N. Cam-Winget
E. Wang
Cisco Systems
July 8, 2019

TLS 1.3 Impact on Network-Based Security
draft-camwinget-tls-use-cases-05

Abstract

Network-based security solutions are used by enterprises, public sector, and cloud service providers today in order to both complement and enhance host-based security solutions. TLS 1.3 introduces several changes to TLS 1.2 with a goal to improve the overall security and privacy provided by TLS. However some of these changes have a negative impact on network-based security solutions and deployments that adopt a multi-layered approach to security. While this may be viewed as a feature, there are several real-life use case scenarios where the same functionality and security can not be offered without such network-based security solutions. In this document, we identify the TLS 1.3 changes that may impact such use cases.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

Internet-Draft

I-D

July 2019

This document is subject to [BCP 78](https://trustee.ietf.org/license-info) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction

Enterprises, public sector, and cloud service providers need to defend their information systems from attacks originating from both inside and outside their networks. Protection and detection are typically done both on end hosts and in the network. Host agents have deep visibility on the devices where they are installed, whereas the network has broader visibility. With such network and security devices in the network, it can provide, among other functions, homogenous security controls across heterogenous endpoints, covering devices for which no host monitoring is available (which is common today and is increasingly so in the Internet of Things). This helps protect against unauthorized devices installed by insiders, and provides a fallback in case the infection of a host disables its security agent. Because of these advantages, network-based security mechanisms are widely used. In fact, regulatory standards such as NERC CIP [\[NERCCIP\]](#) place strong requirements about network perimeter security and its ability to have visibility to provide security information to the security management and control systems. At the same time, the privacy of employees, customers, and other users must be respected by minimizing the collection of personal data and controlling access to what data is collected. These imperatives hold for both end host and network based security monitoring.

Network-based security solutions such as Firewalls (FW) and Intrusion Prevention Systems (IPS) rely on some level of network traffic inspection to implement perimeter-based security policies. In many use cases, only the metadata or visible aspects of the network traffic is inspected. Depending on the security functions required, these middleboxes can either be deployed as traffic monitoring devices or active in-line devices. A traffic monitoring middlebox may for example perform vulnerability detection, intrusion detection, crypto audit, compliance monitoring, etc. An active in-line

middlebox may for example prevent malware download, block known malicious URLs, enforce use of strong ciphers, stop data exfiltration, etc. A portion of such security policies require clear-text traffic inspection above Layer 4, which becomes problematic when traffic is encrypted with Transport Layer Security

(TLS) [[RFC5246](#)]. Today, network-based security solutions typically address this problem by becoming a man-in-the-middle (MITM) for the TLS session according to one of the following two scenarios:

1. Outbound Session, where the TLS session originates from a client inside the perimeter towards an entity on the outside
2. Inbound Session, where the TLS session originates from a client outside the perimeter towards an entity on the inside

For the outbound session scenario, MITM is enabled by generating a local root certificate and an accompanying (local) public/private key pair. The local root certificate is installed on the inside entities for which TLS traffic is to be inspected, and the network security device(s) store a copy of the private key. During the TLS handshake, the network security device (hereafter referred to as a middlebox) makes a policy decision on the current TLS session. The policy decision could be pass-through, decrypt, deny, etc. On a "decrypt" policy action, the middlebox becomes a TLS proxy for this TLS session. It modifies the certificate provided by the (outside) server and (re)signs it with the private key from the local root certificate. From here on, the middlebox has visibility into further exchanges between the client and server which enables it to decrypt and inspect subsequent network traffic. Alternatively, based on policy, the middlebox may allow the current session to pass through if the session starts in monitoring mode, and then decrypt the next session from the same client.

For the inbound session scenario, the TLS proxy on the middlebox is configured with a copy of the local servers' certificate(s) and corresponding private key(s). Based on the server certificate presented, the TLS proxy determines the corresponding private key, which again enables the middlebox to gain visibility into further exchanges between the client and server and hence decrypt subsequent network traffic.

To date, there are a number of use case scenarios that rely on the above capabilities when used with TLS 1.2 [[RFC5246](#)] or earlier. TLS 1.3 [[RFC8446](#)] introduces several changes which prevent a number of these use case scenarios from being satisfied with the types of TLS proxy based capabilities that exist today.

It has been noted, that currently deployed TLS proxies on middleboxes may reduce the security of the TLS connection itself due to a combination of poor implementation and configuration, and they may compromise privacy when decrypting a TLS session. As such, it has been argued that preventing TLS proxies from working should be viewed as a feature of TLS 1.3 and that the proper way of solving these

issues is to solely rely on endpoint (client and server) based solutions instead. We believe this is an overly constrained view of the problem that ignores a number of important real-life use case scenarios that improve the overall security posture. For instance, it goes against a layered defense approach. We also note that current endpoint-based TLS proxies suffer from many of the same security issues as the network-based TLS proxies do [[HTTPSintercept](#)].

The purpose of this document is to provide a representative set of _network based security_ use case scenarios that are impacted by TLS 1.3. For each use case scenario, we highlight the specific aspect(s) of TLS 1.3 that make the use case problematic with a TLS proxy based solution.

It should be noted that this document addresses only _security_ use cases with a focus on identifying the problematic ones. The document does not offer specific solutions to these as the goal is to describe how current network security solutions rely on network traffic inspection to address customer requirements and use cases.

[1.1](#). Requirements notation

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [BCP 14](#), [RFC 2119](#) [[RFC2119](#)].

[2](#). TLS 1.3 Change Impact Overview

Aiming to improve its overall security and privacy, TLS 1.3 introduces several changes to TLS 1.2, but some of the changes present a negative impact on network based security. In this section, we describe those TLS 1.3 changes and briefly outline some scenario impacts. We divide the changes into two groups; those that impact inbound sessions and those that impact outbound sessions.

[2.1.](#) Inbound Session Change Impacts

[2.1.1.](#) Removal of Static RSA and Diffie-Hellman Cipher Suites

TLS 1.2 supports static RSA and Diffie-Hellman(DH) cipher suites, which enables the server's private key to be shared with server-side middleboxes. TLS 1.3 has removed support for these cipher suites in favor of supporting only ephemeral mode Diffie-Hellman in order to provide perfect forward secrecy (PFS). As a result of this, it is no longer possible for a server to share a key with the middlebox a priori, which in turn implies that the middlebox cannot gain access to the TLS session data.

Example scenarios that are impacted by this include network monitoring, troubleshooting, compliance, etc.

For further details (and a suggested solution), please refer to [\[I-D.green-tls-static-dh-in-tls13\]](#).

[2.2.](#) Outbound Session Change Impacts

[2.2.1.](#) Encrypted Server Certificate

In TLS, the ClientHello message is sent to the server's transport address (IP and port). The ClientHello message may include the Server Name Indication (SNI) to specify the hostname the client wishes to contact. This is useful when multiple "virtual servers" are hosted on a given transport address (IP and port). It also provides passive observers and security devices information about the domain the client is attempting to reach. Note that while SNI is optional in TLS 1.2, it is mandatory in TLS 1.3.

The server replies with a ServerHello message, which contains the selected connection parameters, followed by a Certificate message, which contains the server's certificate and hence its identity.

Note that even `_if_` the SNI is provided by the client, there is no guarantee that the actual server responding is the one indicated in the SNI from the client. SNI alone, without comparison of the server certificate, does not provide reliable information about the server that the client attempts to reach. Where a client has been compromised by malware and connects to a command and control server, but presents an innocuous SNI to bypass protective filters, it is undetectable under TLS 1.3.

In TLS 1.2, the ClientHello, ServerHello and Certificate messages are all sent in clear-text, however in TLS 1.3, the Certificate message is encrypted thereby hiding the server identity from any intermediary.

Example scenarios that are impacted by this involve selective network security policies on the server, such as whitelists or blacklists based on security intelligence, regulatory requirements, categories (e.g. financial services), etc. Under TLS 1.3, these scenarios now require the middlebox to perform decryption and inspection of every connection to have the same information to make policy decisions. Further, the middlebox is not able to make the policy decisions without actively engaging in the TLS 1.3 session from the beginning of the handshake, and it cannot step out of the connection once it has been determined to be benign, without dropping the whole connection. In TLS 1.2, middleboxes could be more selective in

choosing what connections to engage with, and make decisions based on the certificate without actively decrypting the connection to access the certificate(s).

While conformant clients can generate the SNI and check that the server certificate contains a name matching the SNI, there are non-conformant clients that do not and some enterprises also require a level of validation. Thus, from a network infrastructure perspective, policies to validate SNI against the Server Certificate can not be validated in TLS 1.3 as the Server certificate is now obscured to the middlebox. This is an example where the network infrastructure is using one measure to protect the enterprise from non-conformant (e.g. evasive) clients and a conformant server. As a general practice, security functions conduct cross checks and consistency checks wherever possible to mitigate imperfect or

malicious implementations; even if they are deemed redundant with fully conformant implementations.

[2.2.2.](#) Resumption and Pre-Shared Key

In TLS 1.2 and below, session resumption is provided by "session IDs" and "session tickets" [[RFC5077](#)]. If the server does not want to honor a ticket, then it can simply initiate a full TLS handshake with the client as usual.

In TLS 1.3, the above mechanism is replaced by Pre-Shared Keys (PSK), which can be negotiated as part of an initial handshake and then used in a subsequent handshake to perform resumption using the PSK. TLS 1.3 states that the client SHOULD include a "key_share" extension to enable the server to decline resumption and fall back to a full handshake, however it is not an absolute requirement.

Example scenarios that are impacted by this are middleboxes that were not part of the initial handshake, and hence do not know the PSK. If the client does not include the "key_share" extension, the middlebox cannot force a fallback to the full handshake. If the middlebox policy requires it to inspect the session, it will have to fail the connection instead.

Note that in practice though, it is unlikely that clients using session resumption will not allow for fallback to a full handshake since the server may treat a ticket as valid for a shorter period of time than what is stated in the ticket_lifetime [[RFC8446](#)]. As long as the client advertises a supported DH group, the server (or middlebox) can always send a HelloRetryRequest to force the client to send a key_share and hence a full handshake.

Clients that truly only support PSK mode of operation (provisioned out of band) will of course not negotiate a new key, however that is not a change in TLS 1.3.

[2.2.3.](#) Version Negotiation and Downgrade Protection

In TLS, the ClientHello message includes a list of supported protocol versions. The server will select the highest supported version and

indicate its choice in the ServerHello message.

TLS 1.3 changes the way in which version negotiation is performed. The ClientHello message will indicate TLS version 1.3 in the new "supported_versions" extension, however for backwards compatibility with TLS 1.2, the ClientHello message will indicate TLS version 1.2 in the "legacy_version" field. A TLS 1.3 server will recognize that TLS 1.3 is being negotiated, whereas a TLS 1.2 server will simply see a TLS 1.2 ClientHello and proceed with TLS 1.2 negotiation.

In TLS 1.3, the random value in the ServerHello message includes a special value in the last eight bytes when the server negotiates either TLS 1.2 or TLS 1.1 and below. The special value(s) enable a TLS 1.3 client to detect an active attacker launching a downgrade attack when the client did indeed reach a TLS 1.3 server, provided ephemeral ciphers are being used.

From a network security point of view, the primary impact is that TLS 1.3 requires the TLS proxy to be an active man-in-the-middle from the start of the handshake. It is also required that a TLS 1.2 and below middlebox implementation must handle unsupported extensions gracefully, which is a requirement for a conformant middlebox.

[2.2.4.](#) SNI Encryption in TLS Through Tunneling

As noted above, with server certificates encrypted, the Server Name Indication (SNI) in the ClientHello message is the only information available in cleartext to indicate the client's targeted server, and the actual server reached may differ.

[I-D.ietf-tls-sni-encryption] proposes to hide the SNI in the ClientHello from middleboxes.

Example scenarios that are impacted by this involve selective network security, such as whitelists or blacklists based on security intelligence, regulatory requirements, categories (e.g. financial services), etc. An added challenge is that some of these scenarios require the middlebox to perform inspection, whereas other scenarios require the middlebox to not perform inspection. Without the SNI,

however, the middlebox may not have the information required to

determine the actual scenario before it is too late.

3. Inbound Session Use Cases

In this section we explain how a set of real-life inbound use case scenarios are affected by some of the TLS 1.3 changes.

3.1. Use Case I1 – Data Center Protection

Services deployed in the data center may be offered for access by external and untrusted hosts. Network security functions such as IPS and Web Application Firewall (WAF) are deployed to monitor and control the transactions to these services. While an Application level load balancer is not a security function strictly speaking, it is also an important function that resides in front of these services

These network security functions are usually deployed in two modes: monitoring and inline. In either case, they need to access the L7 and application data such as HTTP transactions which could be protected by TLS encryption. They may monitor the TLS handshakes for additional visibility and control.

The TLS handshake monitoring function will be impacted by TLS 1.3.

For additional considerations on this scenario, see also [\[I-D.green-tls-static-dh-in-tls13\]](#).

3.2. Use Case I2 – Application Operation over NAT

The Network Address Translation (NAT) function translates L3 and L4 addresses and ports as the packet traverses the network device. Sophisticated NAT devices may also implement application inspection engines to correct L3/L4 data embedded in the control messages (e.g., FTP control message, SIP signaling messages) so that they are consistent with the outer L3/L4 headers.

Without the correction, the secondary data (FTP) or media (SIP) connections will likely reach a wrong destination.

The embedded address and port correction operation requires access to the L7 payload which could be protected by encryption.

3.3. Use Case I3 – Compliance

Many regulations exist today that include cyber security requirements requiring close inspection of the information traversing through the network. For example, organizations that require PCI-DSS [\[PCI-DSS\]](#)

compliance must provide the ability to regularly monitor the network to prevent, detect and minimize impact of a data compromise. [PCI-DSS] Requirement #2 (and [Appendix A2](#) as it concerns TLS) describes the need to be able to detect protocol and protocol usage correctness. Further, [PCI-DSS] Requirement #10 detailing monitoring capabilities also describe the need to provide network-based audit to ensure that the protocols and configurations are properly used.

Deployments today still use factory or default credentials and settings that must be observed, and to meet regulatory compliance, must be audited, logged and reported. As the server (certificate) credential is now encrypted in TLS 1.3, the ability to verify the appropriate (or compliant) use of these credentials are lost, unless the middlebox always becomes an active MITM.

[3.4.](#) Use Case I4 - Crypto Security Audit

Organizations may have policies around acceptable ciphers and certificates on their servers. Examples include no use of self-signed certificates, black or white-list Certificate Authority, valid certificate expiration time, etc. In TLS 1.2, the Certificate message was sent in clear-text, however in TLS 1.3 the message is encrypted thereby preventing both a network-based audit and policy enforcement around acceptable server certificates.

While the audits and policy enforcements could in theory be done on the servers themselves, the premise of the use case is that not all servers are configured correctly and hence such an approach is unlikely to work in practice. A common example where this occurs includes lab servers.

[4.](#) Outbound Session Use Cases

In this section we explain a set of real-life outbound session use case scenarios. These scenarios remain functional with TLS 1.3 though the TLS proxy's performance is impacted by participating in the DHE key exchange from the beginning of the handshake. Similarly, while with TLS 1.2 the handshake packets could be passively inspected, with TLS 1.3 the TLS proxy may have to perform full decryption to inspect the certificates or to affect other policies impacting its performance.

[4.1.](#) Use Case 01 - Acceptable Use Policy (AUP)

Enterprises deploy security devices to enforce Acceptable Use Policy (AUP) according to the IT and workplace policies. The security

devices, such as firewall/next-gen firewall, web proxy and an

Internet-Draft

I-D

July 2019

application on the endpoints, act as middleboxes to scan traffic in the enterprise network for policy enforcement.

Sample AUP policies are:

- o "Employees are not allowed to access 'gaming' websites from enterprise network"
- o "Temporary workers are not allowed to use enterprise network to upload video clips to Internet, but are allowed to watch video clips"

Such enforcements are accomplished by controlling the DNS transactions and HTTP transactions. A coarse control can currently be achieved by controlling the DNS response (though this may become infeasible if it is also protected by TLS), however, in many cases, granular control is required at HTTP URL or Method levels, to distinguish a specific web page on a hosting site, or to differentiate between uploading and downloading operations.

The security device requires access to plain text HTTP header for granular AUP control.

[4.2.](#) Use Case 02 – Malware and Threat Protection

Enterprises adopt a multi-technology approach when it comes to malware and threat protection for the network assets. This includes solutions deployed on the endpoint, network and cloud.

While endpoint application based solution may be effective, to an extent, at detecting and preventing some types of attack, defense in depth is widely considered to be best security practice because it provides additional protection against compromise of endpoints. For example, network-based solutions can detect malware and threats based on network visibility and provide discovery to a compromised endpoint, even though the logs of such a compromised endpoint appear normal. That is, network based solutions provide such additional detection, prevention and mitigation of attacks with the benefit of rapid and centralized updates.

The network based solutions utilise network traffic for a range of purposes, including but not limited to: preventing malware landing on the endpoint through signatures, detecting abnormal data exfiltration, allowing 0-day analysis and mitigation of successful attacks."

The security functions require access to clear text HTTP or other application level streams on a needed basis.

Andreasen, et al.

Expires January 9, 2020

[Page 10]

Internet-Draft

I-D

July 2019

[4.3.](#) Use Case 03 - IoT Endpoints

As the Internet of Everything continues to evolve, more and more endpoints become connected to the Internet. From a security point of view, some of the challenges presented by these are:

- o Constrained devices with limited resources (CPU, memory, battery life, etc.)
- o Lack of ability to install and update endpoint protection software.
- o Lack of software updates as new vulnerabilities are discovered.

In short, the security posture of such devices is expected to be weak, especially as they get older, and the only way to improve this posture is to supplement them with a network-based solution. IoT deployments are further challenged in that they host a variety of these devices, each with different update cycles and often, are very slow to update their software or firmware to ensure availability and safe of the environments they operate. This in turn requires network based solutions to afford a consistent security baseline. This solution can range from selective passive monitoring to a full and active MiTM.

[4.4.](#) Use Case 04 - Unpatched Endpoints

New vulnerabilities appear constantly and in spite of many advances in recent years in terms of automated software updates, especially in reaction to security vulnerabilities, the reality is that a very large number of endpoints continue to run versions of software with known vulnerabilities.

In theory, these endpoints should of course be patched, but in practice, it is often not done which leaves the endpoint open to the vulnerability in question. A network-based security solution can look for attempted exploits of such vulnerabilities and stop them before they reach the unpatched endpoint.

[4.5.](#) Use Case 05 – Rapid Containment of New Vulnerability and Campaigns

When a new vulnerability is discovered or an attack campaign is launched, it is important to patch the vulnerability or contain the campaign as quickly as possible. Patches however are not usually available immediately for every device on the network, and even when they are, most endpoints are in practice not patched immediately, which leaves them open to the attack.

Andreasen, et al.

Expires January 9, 2020

[Page 11]

Internet-Draft

I-D

July 2019

A network-based security solution can look for attempted exploits of such new vulnerabilities or recognize an attack being launched based on security intelligence related to the campaign and stop them before they reach the vulnerable endpoint.

[4.6.](#) Use Case 06 – End-of-Life Endpoint

Older endpoints (and in some cases even new ones) will not receive any software updates. As new vulnerabilities inevitably are discovered, these endpoints will be permanently vulnerable to exploits without security solutions that are not endpoint-based.

A network-based security solution can help prevent such exploits with the MITM functions.

[4.7.](#) Use Case 07 – Compliance

This use case is similar to the inbound compliance use case described in [Section 3.3](#), except its from the client point of view.

[4.8.](#) Use Case 08 – Crypto Security Audit

This is a variation of the use case in [Section 3.4](#).

Organizations may have policies around acceptable ciphers and

certificates for client sessions, possibly based on the destination. Examples include no use of self-signed certificates, black or white-list Certificate Authority, etc. In TLS 1.2, the Certificate message was sent in clear-text, however in TLS 1.3 the message is encrypted thereby preventing either a network-based audit or policy enforcement around acceptable server certificates.

It is not possible to implement a full security solution by relying on the client alone in this case. For example, in the many cases where the device is not under configuration control of the organisation (i.e. "Bring Your Own Device" devices, which are present in many modern organisations), as audits and policy enforcements can't be done on such clients or on clients that are not properly configured.

[5.](#) IANA considerations

This document does not include IANA considerations.

[6.](#) Security Considerations

This document describes existing functionality and use case scenarios and as such does not introduce any new security considerations.

[7.](#) Acknowledgements

The authors thank Eric Rescorla, the National Cyber Security Center and Dan Wing who provided several comments on technical accuracy and middlebox security implications.

[8.](#) Change Log

[8.1.](#) Version -01

Updates based on comments from Eric Rescorla.

[8.2.](#) Version -03

Updates based on EKR's comments

9. Version -04

Updates based on Kirsty's comments

10. References

10.1. 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>>.
- [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>>.
- [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>>.

10.2. Informative References

- [HTTPSintercept] "The Security Impact of HTTPS Interception", n.d., <<https://jhalderm.com/pub/papers/interception-ndss17.pdf>>.

Andreasen, et al. Expires January 9, 2020 [Page 13]

Internet-Draft I-D July 2019

- [I-D.green-tls-static-dh-in-tls13] Green, M., Droms, R., Housley, R., Turner, P., and S. Fenter, "Data Center use of Static Diffie-Hellman in TLS 1.3", [draft-green-tls-static-dh-in-tls13-01](#) (work in progress), July 2017.
- [I-D.ietf-tls-sni-encryption] Huitema, C. and E. Rescorla, "Issues and Requirements for SNI Encryption in TLS", [draft-ietf-tls-sni-encryption-04](#) (work in progress), November 2018.

- [NERCCIP] "North American Electric Reliability Corporation, (CIP) Critical Infrastructure Protection", n.d., <<http://www.nerc.com/pa/stand/Pages/ReliabilityStandardsUnitedStates.aspx?jurisdiction=United%20States>>.
- [PCI-DSS] "Payment Card Industry (PCI): Data Security Standard", n.d., <https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2.pdf>.
- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", [RFC 5077](#), DOI 10.17487/RFC5077, January 2008, <<https://www.rfc-editor.org/info/rfc5077>>.

Authors' Addresses

Flemming Andreasen
Cisco Systems
111 Wood Avenue South
Iselin, NJ 08830
USA

Email: fandreas@cisco.com

Nancy Cam-Winget
Cisco Systems
3550 Cisco Way
San Jose, CA 95134
USA

Email: ncamwing@cisco.com

Eric Wang
Cisco Systems
3550 Cisco Way
San Jose, CA 95134

USA

Email: ejwang@cisco.com