vDRA

Decrypt TLS traffic

Feature summary and revision history

Table 1: Summary Data

Applicable Product(s) or Functional Area	vDRA
Applicable Platform(s)	Not Applicable
Default Setting	Enabled – Configuration Required
Related Changes in This Release	Not Applicable
Related Documentation	• *CPS vDRA Operations Guide*

Table 2: Revision History

Revision Details	Release
First introduced.	26.1.0

TLS traffic decryption

TLS traffic decryption is a diagnostic capability that:

- enables the decryption of captured encrypted TLS packets for troubleshooting purposes,
- supports various cipher suites including RSA and ECDSA-based algorithms, and

- allows network administrators to view deciphered Diameter traffic in external analysis tools.

This feature allows for the inspection of raw message bytes that would otherwise be inaccessible due to TLS encryption. Depending on the algorithm used during the handshake, decryption requires either a static private key or a dynamic session key log.

Understanding cipher suites for decryption

The system supports a wide range of cipher suites for TLS and mTLS. The following ciphers are preferred for decryption:

ECDSA Ciphers

- TLS_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

ECDHE-RSA Ciphers

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

Best practice for session key management

We recommend collecting the captured session key file immediately after capturing the traffic. Because the system only maintains the most recent backup file during application restarts, continuous restarts may result in the loss of the specific premaster session keys required for decryption.

Restart the application to apply changes

Ensure that you restart the application after enabling or disabling the TLS capture decryption feature. A restart is necessary for the changes to take effect and requires a re-establishment of peer connections

Configure TLS capture decryption

Enable or disable the capture of session keys used for decrypting TLS traffic.

This feature is enabled by default and is used to capture the premaster session keys required for decrypting ECDHE-RSA or ECDHE-ECDSA ciphers.

Use these steps to configure TLS decryption.

Before you begin

Ensure you have administrative access to the CLI.

Procedure

Step 1 Access the system CLI.

Example:

Step 2 To enable the feature, enter the **dra tls-capture-decryption enable true** command.

Step 3 Verify whether TLS capture is decrypted by using the **show running-config dra tls-capture-decryption** command.

Example:

```
tls-capture-decryption tls-certificate
admin@orchestrator[vpas-A1-master-0]# show running-config dra tls-capture-decryption
dra tls-capture-decryption enable true
admin@orchestrator[vpas-A1-master-0]# docker connect diameter-endpoint-s104
```

Step 4 To disable the feature, enter the **dra tls-capture-decryption enable false** command.

Step 5 Verify the configuration by using **show running-config dra tls-capture-decryption** command.

Example:

```
admin@orchestrator[vpas-A1-master-0]# dra tls-capture-decryption enable false
Setting enable as false
Success! Data written to: dra-tls-capture-decryption/enable
admin@orchestrator[vpas-A1-master-0]# show running-config dra tls-capture-decryption
```

The system updates the decryption capture setting. The changes will be active after the next application restart.

Decrypt TLS packets using ECDSA algorithms

View deciphered Diameter traffic in Wireshark when using ECDSA-based encryption.

ECDSA decryption requires a session key log file (pre-master secret) collected during the handshake.

Ensure TLS capture decryption is enabled in the CLI. Use these steps to decrypt TLS packets using ECDSA.

Procedure

Step 1 Capture the traffic in PCAP format.

Step 2 Download the captured session key file from the system path. `/etc/tls/certs/tlsSessionKeys.txt`

Step 3 Open the PCAP in Wireshark and filter for the specific transactions using IP and port.

Step 4 Navigate to `Edit > Preferences > Protocols > TLS`

Step 5 In the **(Pre)-Master-Secret log filename** field, upload the downloaded session key file. Clikc **OK**

Step 6 Right-click the application data in the packet list and choose **Decode As....**

Step 7 Select **Diameter**.

The encrypted application data is transformed into readable Diameter packets.

Decrypt TLS packets using RSA algorithms

View deciphered Diameter traffic in Wireshark when using RSA-based encryption.

Decrypting RSA requires the private key that corresponds to the public key used for encryption. Use these steps to decrypt TLS packets using RSA.

Before you begin

Ensure to obtain private key file.

Procedure

Step 1 Capture the traffic in PCAP format from he diameter container while traffic is active.

Step 2 Open the PCAP file in Wireshark.

Step 3 Navigate to `Edit > Preferences` and then expand **Protocols** and select **TLS** (or SSL in older versions).

Step 4 Open the **RSA Keys List** and click **Edit**.

Step 5 Add the IP address, port details, and the local path to the **Private Key** and then click **OK**.

Step 6 Select a TLS packet, right-click, and choose **Decode As...**

Step 7 Set the **Current** value to **Diameter** for the specified port.

The data section after the TCP segment displays the deciphered Diameter packet in a readable format.

Generate ECDSA certificates

Create the necessary certificates to use ECDSA ciphers.

ECDSA ciphers require ecparam-generated certificates for the TLS handshake.

Use these steps to generate ECDSA certificates.

Procedure

Step 1 Generate the ECDSA private key using the prime256v1 curve.

Example:

```
openssl ecparam -genkey -name prime256v1 -out ecdsa-key.pem
```

Step 2 Create the certificate using the generated key.

Example:

```
openssl req -new -x509 -key ecdsa-key.pem -out ecdsa-cert.pem -days 365
```

After completing these steps, the ecdsa-key.pem and ecdsa-cert.pem files are ready for installation.

VMware disk encryption

Feature Summary and Revision History

Table 3: Summary Data

Applicable Product(s) or Functional Area	CPS vDRA
Applicable Platform(s)	Not Applicable
Default Setting	Disabled – Configuration Required to Enable
Related Changes in This Release	Not Applicable
Related Documentation	*CPS vDRA Installation Guide for VMware*

Table 4: Revision History

Revision Details	Release
First introduced.	26.1.0

Feature Description

VMwareDdsk encryption is a security feature that:

- provides data-at-rest protection for virtual machine disks (VMDKs),
- Encrypts only the primary disk, not the secondary disks. The secondary disk stores only Prometheus data and saved ISO files, so it must remain intact during ISSM because it contains persistent data. Encrypting the secondary disk could lead to data corruption.
- leverages the AES-256-XTS algorithm for robust encryption, and
- integrates with VMware vSphere 7.0U2 and later versions to secure virtual environments.

This solution specifically targets the primary disks of Diameter Routing Agent (DRA) virtual machines to ensure compliance with security standards such as ISO/IEC 27001 and NIST SP 800-53.

A Native Key Provider (NKP) is a vSphere component that simplifies VM encryption management by removing the requirement for an external Key Management Server (KMS),:generates and manages cryptographic keys directly within the vCenter Server, and enables features like Encrypted vMotion and Encrypted Fault Tolerance (FT).

System requirements for VMware disk encryption

To enable disk encryption for DRA, the environment must meet these specifications:

Hardware and firmware:

- VMware-certified hardware equipped with AES-NI.

- BIOS with AES-NI enabled.

Software and licensing:

- vCenter Server version 7.0U2 or later.
- ESXi version 7.0 or later.
- VMware Enterprise Plus license.
- DRA Release 26.1 or later

For more information on the CLI commands, refer to the *CPS vDRA Operation Guide*.