Exam Objective 7

Secure Agent Communication

Secure Agent Communication

Objective 7a: Understanding Consul security/threat model

Objective 7b: Differentiate certificate types needed for TLS encryption

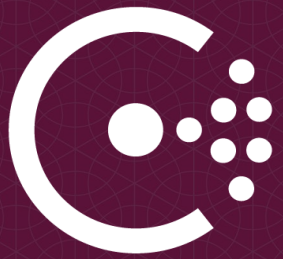
Objective 7c: Understand the different TLS encryption settings for a fully secure datacenter

1	2	3	4	5

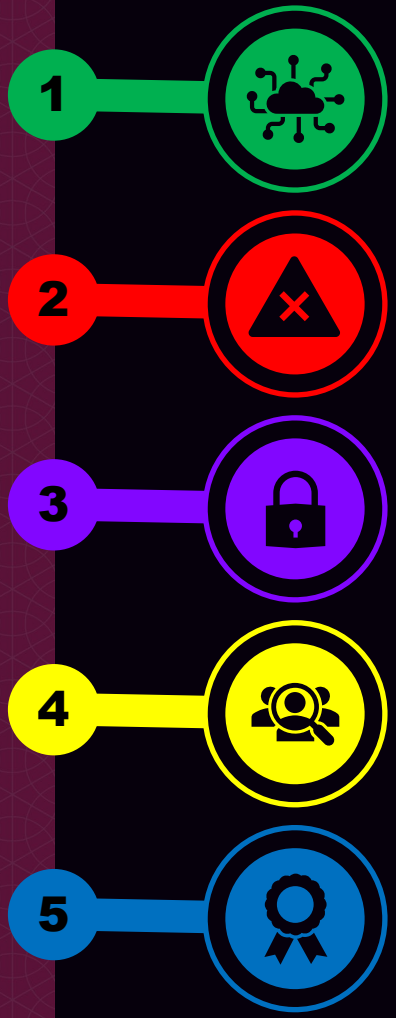
Difficulty Level

Consul is Not Secure

(by default)



Security Model



Gossip Protocol Encryption

Built-In ACL System

Consul Agent Communication

mTLS for Authenticity + Encryption

Certificate Authority

Consul Security/Threat Model

Consul secures communication using multiple methods

- Gossip protocol can encrypt communications throughout the cluster
- ACL system protects data and APIs
- Consul agent supports encrypting all communications using TLS (RPC/API)
- mTLS are used to verify authenticity and encrypt communications
- Consul can act as a CA, or natively integrate with an existing CA (Vault or other)

Consul Security/Threat Model

Gossip Protocol (Serf)

- Gossip protocol uses a symmetric key
- Essentially a 'shared secret' method for both servers and clients in a cluster
- More on Gossip Encryption in Objective 9

ACL System

- Optional system – not enabled by default
- Protects access to Consul data and HTTP APIs
- More on the ACL System in Objective 8

Consul Security/Threat Model

Consul Agent

- Supports TLS certificates to encrypt communications for RPC and API connectivity
- Allows Consul to be run over untrusted networks (public cloud, Internet, etc.)
- Enabled in the server configuration file
- Consul can verify incoming/outgoing communications and check server hostnames

mTLS to Validate Authenticity and Encrypt Communications

- Uses the CA to validate authenticity against public CA bundle
- Used for Service Mesh functionality

Consul Security/Threat Model

Certificate Authority

- Consul can act as a CA to issue certificates for the datacenter
- Certificate types include:
  - Server - consul tls cert create –server (CLI command)
  - Client – consul tls cert create –client (CLI command)
  - CLI – consul tls cert create –cli (CLI command)

mTLS to Validate Authenticity and Encrypt Communications

- Uses the CA to validate authenticity against public CA bundle
- Used for Service Mesh functionality

Certificates Required in Consul

Consul HTTP API and RPC communication require TLS certs

- Encryption

Service Mesh connectivity uses mTLS certificates

- Authenticity
- Encryption

You can manually set the HTTP port on Consul (–https-port) and provide a certificate if you want to manually configure the API to use HTTPS

Certificates Required in Consul

Consul can act as the CA

- Enabled if connect is enabled without specifying a CA provider
- Consul can automatically distribute client certificates (automated)
- Or you can do it manually

Certificates can be generated from your own CA

- You must distribute certs manually, also known as the 'operator' method

- ► Certificates must be signed by the same Certificate Authority
- ► You can update to a new provider at any time

TLS Encryption Settings

Three primary configurations when working with Consul TLS

- verify_server_hostname
- verify_incoming
- verify_outgoing

These settings are added/updated in the Consul Agent configuration file

Terminal — config.hcl

```
{
  "log_level": "INFO",
  "server": true,
  "key_file": "/etc/consul.d/cert.key",
  "cert_file": "/etc/consul.d/client.pem",
  "ca_file": "/etc/consul.d/chain.pem",
  "verify_incoming": true,
  "verify_outgoing": true,
  "verify_server_hostname": true,
  "ui": true,
  "encrypt": "xxxxxxxxxxxxx",
  …
```

TLS Encryption Settings

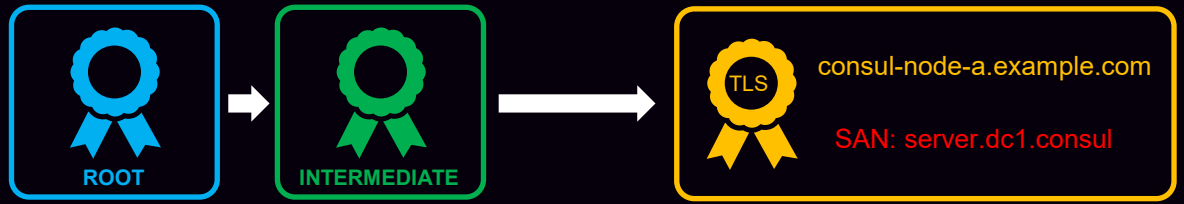
TLS Encryption Settings

verify_server_hostname

- All outgoing connections perform hostname verification
- Ensures that servers have a certificate valid for server.<datacenter>.<domain>
- Ensures a client cannot modify the Consul Agent config and restart as a server
- Without this setting, Consul only verifies that the cert is signed by a trusted CA

Consul CA server certificates will include this hostname by default

If you are using your own CA to create certificates for Consul, you MUST include server.<datacenter>.<domain> as a SAN (subject alternative-name)



TLS Encryption Settings

Terminal — config.hcl

```
{
  "log_level": "INFO",
  "server": false,
  "key_file": "/etc/consul.d/cert.key",
  "cert_file": "/etc/consul.d/client.pem",
  "ca_file": "/etc/consul.d/chain.pem",
  "verify_incoming": true,
  "verify_outgoing": true,
  "verify_server_hostname": true,
  "encrypt": "xxxxxxxxxxxxx",
…
```

Terminal — config.hcl

```
{
  "log_level": "INFO",
  "server": true,
  "key_file": "/etc/consul.d/cert.key",
  "cert_file": "/etc/consul.d/client.pem",
  "ca_file": "/etc/consul.d/chain.pem",
  "verify_incoming": true,
  "verify_outgoing": true,
  "verify_server_hostname": true,
  "encrypt": "xxxxxxxxxxxxx",
…
```

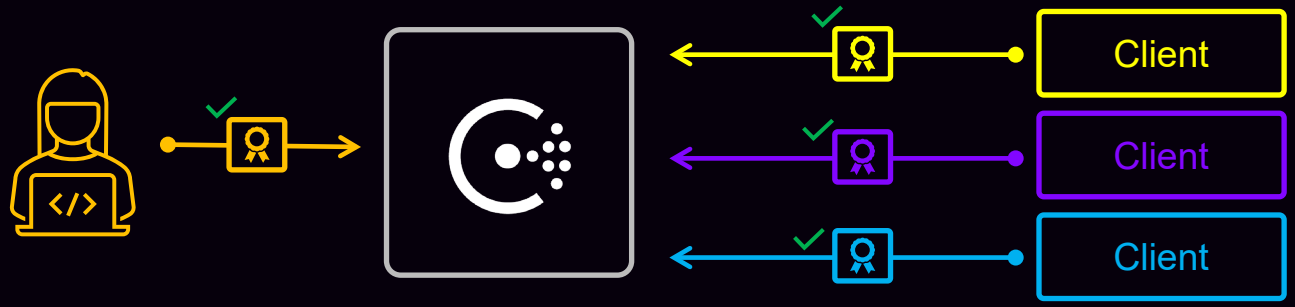
If somebody changes the configuration to a server

TLS Encryption Settings

verify_incoming

- Requires that all incoming connections use TLS
- The TLS cert must be signed by a CA includes in the ca_file or ca_path
- By default, this setting is false (must be enabled)

This setting is valid for both RPC and HTTPS API connectivity

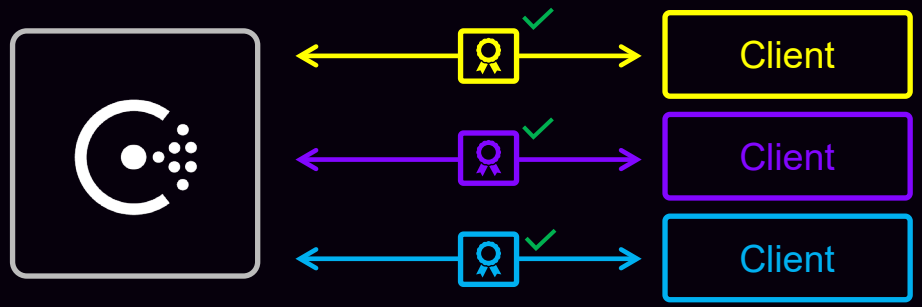


TLS Encryption Settings

verify_outgoing

- Requires that all outgoing connections use TLS
- The TLS cert must be signed by a CA includes in the ca_file or ca_path
- By default, this setting is false (must be enabled)

This applies to both clients and servers as each will make outgoing connections



Secure Agent Communication

Objective 7a: Understanding Consul security/threat model

Objective 7b: Differentiate certificate types needed for TLS encryption

Objective 7c: Understand the different TLS encryption settings for a fully secure datacenter

1	2	3	4	5

Difficulty Level

END OF SECTION