**Problem**

<https://github.com/hashicorp/consul/pull/5069/files>

<https://github.com/hashicorp/consul/blob/38122cb9a4596c61422b5dfc28bee32e214827d8/website/source/docs/agent/options.html.md>

**security.html.md**

* [verify\_server\_hostname](https://github.com/hashicorp/consul/blob/38122cb9a4596c61422b5dfc28bee32e214827d8/website/source/docs/agent/options.html.md#verify_server_hostname) - If set to true, Consul verifies for all outgoing TLS connections that the TLS certificate presented by the servers matches "server.<datacenter>.<domain>" hostname. By default, this is false, and Consul does not verify the hostname of the certificate, only that it is signed by a trusted CA. This setting is critical to prevent a compromised client from being restarted as a server and having all cluster state including all ACL tokens and Connect CA root keys replicated to it. This is new in 0.5.1.
  + **Security Note:** From versions 0.5.1 to 1.4.0, due to a bug, setting this flag alone does not imply verify\_outgoing and leaves client to server and server to server RPCs unencrypted despite the documentation stating otherwise. See [CVE-2018-19653](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-19653) for more details. For those versions you **must also set verify\_outgoing = true** to ensure encrypted RPC connections.

**options.html.md**

* **Encryption enabled.** TCP and UDP encryption must be enabled and configured to prevent plaintext communication between Consul agents. At a minimum, verify\_outgoing should be enabled to verify server authenticity with each server having a unique TLS certificate. verify\_server\_hostname is also required to prevent a compromised agent restarting as a server and being given access to all secrets. verify\_incoming provides additional agent verification via mutual authentication but isn't strictly necessary to enforce the threat model since requests must also contain a valid ACL token. The subtlety is that currently verify\_incoming = false will allow servers to still accept un-encrypted connections from clients (to allow for gradual TLS rollout). That alone doesn't violate the threat model, but any misconfigured client that chooses not to use TLS will violate the model. We recommend setting this to true. If it is left as false care must be taken to ensure all consul clients use verify\_outgoing = true as noted above, but also all external API/UI access must be via HTTPS with HTTP listeners disabled.
* **Verify Server Hostname Used Alone.** From version 0.5.1 to 1.4.0 we documented that verify\_server\_hostname being true **implied verify\_outgoing** however due to a bug this was **not** the case so setting only verify\_server\_hostname results in plaintext communication between client and server. See [CVE-2018-19653](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-19653) for more details. This is fixed in 1.4.1.

**config.go**

func (c \*Config) OutgoingTLSConfig() (\*tls.Config, error) {

// If VerifyServerHostname is true, **that implies** **(vulnérabilité ici, mauvaise docum)** VerifyOutgoing

**if c.VerifyServerHostname {**

**c.VerifyOutgoing = true**

**}**

if !c.UseTLS && !c.VerifyOutgoing {

return nil, nil

}

**Fix**

**config.go**

**c.VerifyOutgoing** = true Déplacer dans le fichier **builder.go**

**builder.go**

// VerifyServerHostname implies VerifyOutgoing

**verifyServerName := b.boolVal(c.VerifyServerHostname)**

**verifyOutgoing := b.boolVal(c.VerifyOutgoing)**

if verifyServerName {

// Setting only verify\_server\_hostname is documented to imply

// verify\_outgoing. If it doesn't then we risk sending communication over TCP

// when we documented it as forcing TLS for RPCs. Enforce this here rather

// than in several different places through the code that need to reason

// about it. (See CVE-2018-19653)

**verifyOutgoing = true**

}