TCG Trusted Network Communications Server Discovery and Validation

Specification Version 1.0
Revision 25
16 October 2017
Published

Contact:
admin@trustedcomputinggroup.org

TCG PUBLISHED

Acknowledgements

The TCG wishes to thank all those who contributed to this specification. This document builds on considerable work done in the various work groups in the TCG.

Special thanks to the members of the TNC-WG contributing to this document:

Adrien Raffin	AMOSSYS
Padma Krishnaswamy	Battelle Memorial Institute
Eric Fleischman	Boeing
Richard Hill	Boeing
Steven Venema	Boeing
Nancy Cam-Winget	Cisco Systems
Scott Pope	Cisco Systems
Max Pritikin	Cisco Systems
Allan Thomson	Cisco Systems
Henk Birkholz	Fraunhofer SIT
Gerald Maunier	Gemalto
Nicolai Kuntze	Huawei
Yi Zhang	Huawei
Ira McDonald	High North
Dr. Josef von Helden	Hochschule Hannover
Tom Laffey	HP
Dr. Andreas Steffen	HSR University of Applied Sciences Rapperswil
James Tan	Infoblox
Steve Hanna (Editor)	Infineon Technologies
Clifford Kahn	Pulse Secure
Lisa Lorenzin (TNC-WG Co-Chair)	Pulse Secure
Atul Shah (TNC-WG Co-Chair)	Microsoft
Jon Baker	MITRE
Charles Schmidt (Editor)	MITRE
Rainer Enders	NCP Engineering
David Waltermire	NIST
Dick Wilkins	Phoenix Technologies
Carolin Latze	Swisscom
Richard Struse	United States Government
Mike Boyle	United States Government
Emily Doll	United States Government
Jessica Fitzgerald-McKay	United States Government
Jonathan Hersack	United States Government
Mary Lessels	United States Government
Chris Salter	United States Government
Andrew Cathrow	Verisign

Table of Contents

1 Introduction

1.1 Scope and Audience

The Trusted Network Communications Work Group (TNC-WG) has defined an open solution architecture that enables network operators to collect, store, and share information, and to apply access control, based on data provided by endpoints. Likewise, the TNC Architecture allows endpoints to receive instructions and other information that the endpoint uses to adjust its state or behavior. In either scenario, an endpoint is required to interact with one or more servers. This document defines standard techniques that an endpoint can use to find servers with which it needs to interact and to validate the trustworthiness of these servers.

Architects, designers, developers, and technologists who wish to implement, use, or understand Server Discovery and Validation should read this document carefully. Before reading this document any further, the reader should review and understand the TNC Architecture [13].

1.2 Motivation

Server Discovery and Validation provides a substantial improvement in security for the TNC Architecture, because it defines exactly how an endpoint can find a needed server and validate that this server is trusted. While IF-T for Tunneled EAP Methods, IF-T for TLS, and other TNC specifications include some guidance in this area, their guidance is not complete. Without this specification, an endpoint might disclose sensitive information to an untrustworthy server or accept malicious instructions.

Moreover, Server Discovery and Validation supports flexibility in configuration as to which servers a given endpoint should use. Once endpoints are provided with initial Endpoint Provisioning Information (discussed in section 3), network managers can tailor the list of servers each endpoint discovers. This set of discovered servers can be changed easily without requiring re-provisioning of the endpoint. As such, Server Discovery and Validation substantially improves security and privacy in a flexible and scalable manner.

1.3 Keywords

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [1]. This specification does not distinguish blocks of informative comments and normative requirements. Therefore, for the sake of clarity, note that lower case instances of must, should, etc. do not indicate normative requirements.

2 Background

2.1 Overview

The TNC Architecture defines a stack of network protocols that carry measurements from the endpoint and deliver instructions and measurement requests to an endpoint. Instructions come from and measurements are delivered to servers, such as a Policy Decision Point (PDP) or Compliance Evaluation Point (CEP), as defined in the TNC Architecture [13], or a NEA Server, as defined in RFC 5209 [14]. For the purposes of this document, the term **policy server** is used to refer to a PDP, CEP, or NEA Server. Likewise, if an endpoint serves in the role of a MAP Client, it needs to interact with a MAP Server, as defined in the TNC Architecture. Depending on its role and circumstances, the endpoint might need to interact with other servers, including servers that support proprietary behaviors.

In order for the endpoint to send its integrity measurements to a policy server (or policy servers) using TNC protocols, the endpoint needs to first figure out how to contact the proper policy server and decide whether it ought to trust this policy server to perform a TNC exchange. In other situations, identification and determination of trust for other types of servers might be necessary. Supporting these activities in a standardized manner is the purpose of this specification.

Discovery and validation of specific classes of servers, such as policy servers, are important steps in making the architecture easy to deploy, easy to use, and secure. With this specification, an endpoint can automatically find a policy server or other server and establish a secure and trusted TNC connection, without any need for user or administrator effort beyond providing Endpoint Provisioning Information when the endpoint is initially configured.

This document refers to many TCG specifications and their IETF equivalents. In some cases, it defines extensions to existing TCG specifications, and it often uses acronyms and terms defined in these documents. The reader ought to consult the documents listed in the References section of this document (section 9) as necessary. Further, the reader ought to be aware that the following TCG and IETF documents are interoperable:

Table 1 - Interoperability Between TCG and IETF Specifications

TCG Specification	IETF Specification
IF-TNCCS 2.0 [2]	RFC 5793 PB-TNC [15]
IF-T: Binding to TLS 2.0 [16]	RFC 6876 PT-TLS [17]
IF-T: Binding to Tunneled EAP Methods 2.0 [18]	RFC 7171 PT-EAP [19]

2.2 Supported Use Cases

The following use cases are supported by this specification:

- An endpoint connects to an IP network using Tunneled EAP methods over an 802.1X connection. The endpoint determines whether the associated policy server that is performing the EAP authentication is trusted before performing a TNC exchange with this policy server. As part of this exchange, the policy server can send information as to which servers to contact after the 802.1X exchange is complete and the endpoint is connected to the network. The endpoint then validates that the reported servers are trusted before establishing a connection to them using a process that is not dependent on its trust of the original policy server.

- An endpoint connects to an IP network. Once connected to the network, the endpoint discovers one or more servers (e.g., policy servers) and validates that it is able to trust these servers enough to interact with them. Successful interaction with these servers might alter the behavior or configuration of the endpoint, result in the logging of information about endpoint state for future reference by network systems, and/or result in greater access for the endpoint.

2.3 Non-supported Use Cases

The following use cases are not supported by this specification:

- A server experiencing heavy loads redirects an endpoint which is already interacting with it to an alternate server in the expectation that the endpoint will switch to the alternate server and balance the server load.
- A server sends address information for a backup server of the same type to an endpoint. The endpoint uses this information later, if the endpoint can no longer contact the primary server. While an endpoint might discover multiple instances of a particular type of server, and might use this list to identify a new server to contact in the event that connection to a primary server is lost, this specification is not intended to allow individual servers to indicate their own backups.

While these use cases are not supported by this specification, implementers are not prohibited from supporting them in their products. They are mentioned here because readers might incorrectly assume that this specification will include guidance related to these activities. No such guidance is included in the current version of this specification, but these use cases might be supported in future versions of this specification.

2.4 Requirements

The following are the requirements that the Server Discovery and Validation specification needs to meet in order to successfully play its role in the TNC Architecture. These are stated as general requirements, with specific requirements called out as appropriate.

1. **Supports the TNC Architecture**

   Server Discovery and Validation needs to support all the functions and use cases described in the TNC Architecture and related TNC specifications as they apply to discovering PDPs and other types of servers.

2. **Secure**

   Server Discovery and Validation needs to have a well-documented threat analysis and threat mitigation strategy. This strategy might depend on other established specifications.

3. **Easy to use and implement**

   Server Discovery and Validation needs to be easy for endpoint developers to implement, easy for network administrators to set up, and easy for users to use. It ought to utilize existing standard network infrastructure services (e.g., DNS) rather than creating new requirements for network infrastructure.

4. **Scalable and Efficient**

   Server Discovery and Validation ought to promote scalability and efficiency in all respects, since the number of endpoints and servers in a network might be quite large. One purpose of this specification is to reduce the need for manual reconfiguration of endpoints and manual choices to be made by users. Some endpoint configuration is still required, but this ought to be kept to an absolute minimum, even as other circumstances on the network change.

2.5 Non-requirements

The following “non-requirements” are provided to specifically exclude certain uses of this specification that might otherwise be assumed as implicitly required.

1. Mechanism for Configuration of Endpoint Provisioning Information

Endpoints require pre-provisioned information in order to validate a discovered server, as well as instructions on the types of servers to which they need to connect. This information is referred to in this specification as Endpoint Provisioning Information. Because Endpoint Provisioning Information needs to be trusted, but needs to be used before any server can be known to be trusted, this information needs to be pre-provisioned on an endpoint before it engages in Server Discovery and Validation. This specification does not define a mechanism for providing Endpoint Provisioning Information to an endpoint; Endpoint Provisioning Information is assumed to have been securely configured prior to the discovery and validation activities described in this specification.

2.6 Assumptions

Here are the assumptions that Server Discovery and Validation makes about components in the TNC Architecture:

1. Configuration of Endpoint Provisioning Information

As noted above, the endpoint is assumed to have Endpoint Provisioning Information that allows it to determine which servers to which it must connect and verify their trustworthiness. Requirements regarding Endpoint Provisioning Information appear in section 3.

2. Initial Connections to Servers Include Delivery of a Server Certificate to the Endpoint

Validation of a server involves comparing fields of an X.509 certificate delivered by the server to the Trust Parameters associated with the client application contacting that server. As such, when a client application initially attempts to connect to a discovered server, the connection handshake with the server needs to include delivery of the server's X.509 certificate, along with evidence that this is the server's own certificate. This requirement can be met by features of the underlying transport protocol, such as IF-T: Binding to Tunneled EAP Methods [18] and IF-T: Binding to TLS [16].

3. **Support for IF-TNCCS 2.0 or Later if Supporting TNCCS-Based Server Discovery**

This specification defines two methods for discovering servers, one of which uses IF-TNCCS messages. If this method of discovery is employed, both the endpoint and server need to support IF-TNCCS 2.0 [2] or later.

3 Endpoint Provisioning

Before a client application can engage in the Server Discovery and Validation process, the endpoint needs to be provisioned with Endpoint Provisioning Information. This Endpoint Provisioning Information consists of:

- Client Connection Policy - This instructs the endpoint as to how it ought to respond to the discovery of certain types of servers and constrains the servers to which clients can connect.
- Trust Parameters - These allow client applications to determine which servers are trusted for a given purpose.

A client application MUST have access to both types of Endpoint Provisioning Information to engage in Server Discovery and Validation. This section describes the role of, and provides requirements for, both types of Endpoint Provisioning Information. This document does not standardize the formats for either the Endpoint Compliance Policy or the Trust Parameters. Implementers are free to define the format used to store and/or display this information. In addition, both types of Endpoint Provisioning Information allow implementers to include additional information and use this information to support additional capabilities, provided that in doing so they do not violate the requirements described in this specification or prevent correct Server Discovery and Validation behavior when interacting with a component that does not support their custom extensions. As such, this section provides a minimal set of requirements for this Endpoint Provisioning Information, but implementers have a large amount of flexibility in how this information is actually supported in products.

3.1 Why Perform Server Discovery If Provisioning Is Needed?

One might ask, “Why bother with Server Discovery if one is required to provide Endpoint Provisioning Information on the endpoint? Why not just configure each client application with the trusted servers they are to use?” The answer is “flexibility”. While direct client provisioning could establish a list of trusted servers at a given point of time, any subsequent change to the list of approved servers for a client, or a change in the endpoint’s role in a network, could require re-provisioning of this trusted server list. If these changes occur while the endpoint does not have a trusted connection with the network, this could require local modification of the trusted server list on the endpoint itself before the endpoint’s clients can securely reconnect to network services.

By contrast, Server Discovery and Validation makes it easy to tailor which servers a client application ought to contact in a scalable and secure manner, even if this tailoring occurs when the endpoint is not connected to the network. The Endpoint Provisioning Information consists of general rules to guide client behavior. These rules, in combination with the Server Discovery and Validation procedures, allow a client to dynamically identify the servers with which it ought to communicate and validate that those servers are recognized as trustworthy. These changes can reflect situations such as new servers coming online, old servers being taken offline, the addition of new types of servers, reorganization of servers along different operational lines, or even temporary policy changes as to which servers a given client should use.

In short, provisioning client applications directly with a set of trusted server identities might be simpler in the short term, but it is also inflexible and, as infrastructure and endpoint circumstances change, can require repeatedly returning to the endpoint to re-provision. Server Discovery and Validation requires an initial provisioning step, but once completed, it can support changes in server and endpoint activity without further need to reprovision the endpoint. Both methods require provisioning of the endpoint, but Server Discovery and Validation allows this initial provisioning step to remain valid as network conditions and needs evolve.

3.2 Client Connection Policy

The Client Connection Policy governs the use of the list of servers that are discovered during the Discovery phase of Server Discovery and Validation. At a very simple level, the Client Connection

Policy identifies to which services, if any, the endpoint is to immediately attempt connections and to which services the endpoint is permitted to connect if a need arises. The Client Connection Policy can go beyond this, such as by identifying which client applications to use when contacting certain services.

The Client Connection Policy MUST contain a list of Server Connection Rules. The Client Connection Policy MAY contain additional implementer-defined information. The following sections discuss the requirements for Server Connection Rules. Note that this document provides requirements as to what each part of a Server Connection Rule contains, but does not specify the format of this information. As a result, there is no expectation that individual Client Connection Policy instances will be portable across different implementations.

The Server Connection Rules associate an action with a set of server criteria and, in some cases, specific client applications on the endpoint.

In Server Connection Rules, each rule consists of at least:

- an action
- zero or more criteria used to match against discovered servers

Implementers MAY support additional parts in Server Connection Rules. The following sections further describe each required part of a rule.

3.2.1 Actions

There are two Core Connection Actions defined in this specification. Conformant implementations MUST support both Core Connection Actions. The Core Connection Actions are:

- ALLOW_CONNECTION - Rules with this action indicate that certain connections are permitted, but do not require that they be attempted. This action is intended to cover situations where a client application on the endpoint seeks to initiate a connection to some discovered server. ALLOW_CONNECTION rules with this action MUST apply either to a specific server type, as might be identified using the Server Discovery process, or to a specific group of server types, or to any server type. Rules with this action MUST apply either to a specific client application on the endpoint, or to a specific group of client applications on the endpoint, or to any client application on the endpoint.
- CONNECT - Rules with this action indicate not only that a connection is permitted, but that the described connection MUST be attempted on a “best effort” basis as soon as it is feasible to do so. Failure to successfully establish a connection, for any reason, is not considered a policy violation. CONNECT rules MUST identify a specific server type, as might be identified using the Server Discovery process, and MUST identify a specific application on the endpoint that is to act as the client in the connection with the server. In other words, unlike an ALLOW_CONNECTION rule, which can apply to multiple server types and/or multiple server client applications, CONNECT rules need to be specific in both the application and the target server type.

A connection between a client application on the endpoint and a server of a given type that is not explicitly permitted by a rule MUST be denied. As such, there is no action specifying an explicit denial of connections.

Table 2 summarizes the meanings of each type of rule action, including the absence of a given rule.

Table 2 - Rule Meanings

	Connection Allowed	MUST Attempt Connection
CONNECT	Yes	Yes
ALLOW_CONNECTION	Yes	No
No rule	No	No

For a connection to be considered attempted, the following steps MUST occur until either the connection is established or one of the steps fails. Note that the steps identified in the first two bullets can occur in any order; the step in the third bullet depends on the successful completion of the two preceding steps and thus needs to occur last.

- A server is identified whose type matches that rule. In addition, if that rule includes other criteria, the server properties need to match those criteria as well.
- If the rule names a specific client application, that client must exist and be capable of contacting the identified server. Implementers MAY include support for additional constraints in rules to identify specific client applications to use.
- The client contacts the identified server and successfully completes the Server Validation process. Note that the validated server might not be the same as the initially-contacted server. For example, the initial connection request might be redirected to a different server to support load balancing or for other reasons. Only the server to which the initial connection is sent needs to match the type and criteria given in the rule; the validated server does not need to match the type or criteria given in the rule.

Once those steps are complete, the client is considered to have established a connection with the validated server (which is not necessarily the initially-contacted server) and can engage in an appropriate exchange for that server type. The client application is responsible for determining what, if anything, happens in the case that the connection attempt fails.

Implementers MAY define additional actions. Implementers MUST NOT redefine the Core Connection Actions.

A client application that uses Server Discovery results MUST use a Client Connection Policy to control all server connections it attempts. Note that it may be the case that some client applications on an endpoint do not use Server Discovery results. For example, some client applications might only support explicit configuration of servers to use. Clients that do not use Server Discovery information are not required to be constrained by any Client Connection Policy, since they might not include any hooks to support consultation of the policy. Of specific note, connection to a policy server over IF-T for Tunneled EAP is not subject to the Client Connection Policy, because the specific policy server to which the endpoint connects is selected by network architecture rather than through the endpoint's use of Server Discovery results. (That said, the client MUST still apply Server Validation for this policy server, as described in section 5.) Any servers discovered via Server Discovery results from that initial policy server would be subject to the Client Connection Policy.

CONNECT rules mandate action by specific client applications on the endpoint. ALLOW_CONNECT rules are only employed in the case where some client application on the endpoint independently attempts to contact a server. All of the CONNECT rules MUST be processed upon completion of the Server Discovery process as soon as it is feasible to do so, which is often as soon as an endpoint joins a network. Certain situations could lead to the CONNECT rules being run again, e.g. if an endpoint joins a new network or a VPN that makes connectivity to new types of servers feasible. Some examples of situations under which the CONNECT rules should be (re)run could include:

- An endpoint completes an 802.1X exchange with a PDP using IF-T for Tunneled EAP Methods and is assigned an IP address to use in the network. (Note that this initial connection to the PDP is not subject to the Client Connection Policy, since the PDP is selected by the network architecture rather than through the endpoint's use of Server Discovery information.) During the 802.1X exchange, it receives Server Discovery information using an IF-TNCCS message, as described in section 4.2. Once the endpoint has received this IP address, it has "joined the network" and is required to immediately run its CONNECT Server Connection Rules.
- An endpoint connects to a network without a preceding 802.1X exchange and performs Server Discovery over DNS, as described in section 4.3. At the completion of the Server Discovery process, it is required to immediately run its CONNECT Server Connection Rules.
- An endpoint has joined a network and run its CONNECT Server Connection Rules. It subsequently establishes a VPN over this network. When the endpoint establishes the VPN, it is considered to have "joined a network" and is required to immediately rerun its CONNECT Server Connection Rules again to connect to servers that were not accessible without the VPN connection.

Note that these are intended as examples. Different implementers and enterprises might use any of these conditions or define their own conditions for running CONNECT rules. In some situations, CONNECT Server Connection Rules could be run multiple times; in the case that the endpoint already has an active connection to a server that complies with a given CONNECT rule, that rule SHOULD automatically be treated as satisfied without requiring the endpoint to establish another connection.

Both CONNECT and ALLOW_CONNECT rules permit matching client applications to contact services that match the rule at a later time. For example, user action could cause a client to need to establish a connection with a given type of server to complete the user request. As long as there is a matching CONNECT or ALLOW_CONNECT rule, the client is allowed to attempt this connection.

3.2.2 Criteria

All rules using Core Connection Actions MAY include additional criteria. These criteria are compared against properties of servers that match the indicated server type to further refine which discovered servers are applicable to the rule. For example, a criterion could be a Local Group ID value, as conveyed by a TNCCS-Server-Referral message, or an IP address range. In such cases, only discovered server instances that match that Local Group ID, or whose IP addresses match that range, would be eligible to fulfill that rule. Implementers MUST support the inclusion of criteria in rules and MUST support at least the use of Local Group ID, IP address ranges (both IPv4 [3] and IPv6 [4]), and wild-carded DNS [5] names as criteria. Implementers MAY support other criteria. Moreover, implementers SHOULD support the use of logical operators (such as AND and OR) to allow combinations of criteria in a rule. Implementers are otherwise free to design the syntax and capabilities of these criteria.

3.2.3 A Worked Example

This section illustrates how the parts of a Server Connection Rule work together to control client connections to discovered servers. Note that the rule syntax used in this example is informative in nature, as this document does not dictate the syntax of Client Connection Policies.

Assume the following two Server Connection Rules in the Client Connection Policy:

```
{CONNECT
    C:/Application-path/PDP-client.exe      ; Identifies a specific client application to use
    PDP                                     ; A server type
```

```
    Local Group ID = 1 OR Local Group ID = 2  ; Criteria
}
{ALLOW_CONNECTION
    MAP                                       ; A server type
}                                             ; No criteria are provided
```

The first rule is a CONNECT rule. The rule uses the term “PDP” to indicate a server type of a TNC Policy Decision Point. The CONNECT rule also includes criteria. The effective meaning of this rule is:

> “Upon completion of Server Discovery, the PDP-client.exe application identified in the rule is required to attempt to connect to and validate a Policy Decision Point. Only try to connect to a Policy Decision Point with a Local Group ID of 1 or 2.”

Note that this rule does not constrain any IF-T for Tunneled EAP connection to a PDP since the identity of the server to which the EAP connection made is dictated by the network architecture rather than the endpoint.

The second rule is an ALLOW_CONNECTION rule. Unlike the CONNECT rule, this rule does not include any requirements regarding the client application to which the rule applies. The rule also contains no criteria sets. The effective meaning of this rule is:

> “If any client seeks to establish a connection to an MAP Server, then any validated MAP Server is acceptable for this.”

Given this rule set, after the endpoint has joined the network and processed the CONNECT rule, if some client application other than the one named in the CONNECT rule sought to establish a new connection to a Policy Decision Point, this attempt would automatically be denied, as there is no rule supporting that connection.

3.3 Trust Parameters

Any TNC-enabled client application that complies with this specification MUST be configurable with a set of Trust Parameters that are used to determine whether a server is trusted. This information MAY also include to what degree or purpose a server is trusted, but if and how such information is expressed is up to implementers. The Trust Parameters MUST include at least the following items:

- A set of Trusted Server Names

  Trusted Server Names are DNS host names [5] or IP addresses that are matched against the subject name or subject alternative name in the server’s certificate as part of the Server Validation process. Wildcards in Trusted Server Names MAY be used to define the set of Trusted Server Names for DNS host names (e.g. *.example.com to mean any DNS host name that ends with .example.com). Wildcards MUST NOT be used in IP addresses in the Trusted Server Names. Note that server certificate subject name and subject alternative name fields can contain wildcards. Diligent attention is required when writing wildcarded Trusted Server Names to avoid inadvertent matches; for example, the Trusted Server Name entry "foo*.mycompany.com" will match a certificate with the name "*bar.mycompany.com".[1]

[1] RFC 6125 recommends against most forms of wildcarded DN names in server certificates. However, in practice, many environments do employ server certificates with wildcarded DNS names. It is to support such existing environments that this specification supports matching of wildcarded Trusted Server Names against wildcarded server certificate names. However, there is not a similarly compelling reason to support wildcards in IP addresses, as this is almost never seen in practice. Thus, while both forms of wildcard-to-wildcard comparisons pose the same risk of

- A set of Trusted Root Certificates

  Trusted Root Certificates are X.509 certificates that identify the Certification Authorities (CAs) that are trusted to issue certificates for trusted servers.

  This set of trusted X.509 root certificates SHOULD NOT be the same set that is trusted for issuing certificates for web sites, because the list of trusted root certificates for web sites is generally quite long. There is no need to trust hundreds of CAs to issue certificates for servers. Having the administrator configure a separate list of Trusted Root Certificates for servers - or at least select one or two trusted root certificates from the long list of web root certificates - reduces the risk that an attacker can obtain a certificate that chains back to one of the Trusted Root Certificates.

A full description of how this information is used to validate servers is provided in section 5. Briefly, the procedure is as follows: When a client connects to a server, the server provides and proves ownership of an X.509 server certificate. The server is considered validated if and only if the following two facts are true:

1) The server's X.509 certificate is valid and rooted in one of the Trusted Root Certificates.
2) There is a match between one of the Trusted Server Names and the name of the server as provided by the subject name or subject alternate name fields of the server's X.509 certificate.

There are a few caveats and details in this process of which implementers need to be aware, so it is important to carefully review the full validation procedure as described in section 5.

Note that this specification does not provide any guidance with regard to whether there is a link between a certain group of Trusted Server Names and a certain group of Trusted Root Certificates. Implementers MAY support expressing such links (e.g., "for one group of Trusted Server Names only use the following Trusted Root Certificates, but for another group of Trusted Server Names, a different group of Trusted Root Certificates can be employed"). Alternately, implementers MAY choose to make all listed Trusted Root Certificates valid for use with all Trusted Server Names without providing any means to express such links.

The exact format of the Trust Parameters is not defined in this specification and can vary from one implementer to another. Moreover, an implementer MAY include additional information in its expression of Trust Parameters. For example, certain Trusted Server Names MAY be associated with additional information that can then be used as criteria in the Server Connection Rules.

An implementer's Trust Parameters MAY associate different sets of servers with different trust levels. For example, a computer issued to an auditor who often works at customer sites with confidential data might have the auditing firm's PDPs configured as fully trusted and therefore able to send remediation instructions, while the customers' PDPs might be configured as partially trusted, so that endpoint health information can be released to them but remediation instructions from them will be ignored. For another example, a student might configure their personal equipment to release a limited set of endpoint health information to their school. Supporting different levels of trust in this manner is an optional feature for TNC-enabled endpoint software. In addition, different client applications MAY define their own set of Trust Parameters, or share Trust Parameters with other applications. In all of these cases, the manner in which the Trust Parameters are organized and expressed is not specified in this document and is left to implementers to define.

3.4 Provisioning

The exact manner of configuring endpoints with Endpoint Provisioning Information is not defined in this specification and can vary from one piece of TNC-enabled endpoint software to another.

incorrect interpretation, there is less need to accept that risk when dealing with IP addresses, and thus the use of wildcarded IP addresses is prohibited.

However, the process for the configuration of Endpoint Provisioning Information MUST enforce that these parameters are configured securely (e.g. by an authenticated and authorized administrator via a secure protocol). Improper configuration of these parameters can lead to the endpoint software trusting a server that is not trustworthy, leading to possible compromise of the endpoint.

In addition, this specification does not dictate how tightly bound a given set of Endpoint Provisioning Information is to specific clients on an endpoint. A given client application MAY have its own set of Endpoint Provisioning Information that only it uses. Alternately, multiple client applications MAY share the use of a single set of Endpoint Provisioning Information.

4 Server Discovery

This section describes the techniques for discovery of servers. Sections 4.2 and 4.3 define two kinds of Server Discovery: discovery via the IF-TNCCS interface and discovery via DNS SRV. Section 4.1 provides guidance on when to use each of these discovery techniques. Note that endpoints that implement this specification MAY also support other discovery techniques, such as static configuration, in addition to the Server Discovery techniques described in this section.

4.1 When to Use Each Server Discovery Technique

This specification defines two techniques for Server Discovery: the TNCCS-Server-Referral message and DNS SRV. A client application that conforms to this specification and that engages in IF-TNCCS 2.0 or later (henceforth, IF-TNCCS 2.0+) exchanges - i.e., it includes a TNC Client (TNCC) - MUST implement both of these Server Discovery techniques. A client application that conforms to this specification but which does not participate in IF-TNCCS 2.0+ exchanges MUST support the use of DNS SRV to discover servers, but is not required to support TNCCS-Server-Referral messages. A TNC-enabled server that engages in the IF-TNCCS 2.0 exchange and that conforms to this specification MUST support TNCCS-Server-Referral messages. TNC-enabled servers do not need to take any explicit steps to support or be supported by DNS SRV, because only endpoints and DNS servers are involved in this discovery method. Each method of discovery is useful in different situations.

Server Discovery via the TNCCS-Server-Referral message is designed to be used when an endpoint has established a connection to a trusted policy server but needs to transition to a different policy server, transition to a different communications mechanism, or establish connections to other types of TNC-enabled servers. For example, when an endpoint connects to a policy server using an 802.1X exchange, it might perform a TNC handshake via IF-T for Tunneled EAP Methods over 802.1X and then need to transition to IF-T for TLS for further communications with a policy server. The transition from 802.1X to TLS can be effected easily by having the policy server send a TNCCS-Server-Referral message during the 802.1X exchange.

Server Discovery via DNS SRV records is designed to be used when an endpoint has an IP network connection and needs to locate an instance of a given type of server. This method of Server Discovery can be used to locate any type of TNC-enabled serve.

While it is possible to get by with only DNS SRV records, the TNCCS-Server-Referral message can be sent during an initial 802.1X exchange (if this is done) without contacting a separate server and does not require the system administrator to create a DNS SRV record. Furthermore, using a TNCCS-Server-Referral message allows an initial policy server to direct the endpoint to a particular policy server that has all the context necessary to continue monitoring the endpoint without needing to reacquire data. A policy server found via DNS SRV records might not have this context. This said, since TNCCS-Server-Referral messages only support endpoints that engage in an IF-TNCCS exchange prior to Server Discovery, a network might need to support both methods to cover all cases efficiently.

4.2 Discovery via IF-TNCCS

When an endpoint makes an initial connection to a TNC-managed network using IF-T for Tunneled EAP Methods, its request is automatically routed to a policy server by the network infrastructure. However, once this initial TNC handshake completes, the endpoint no longer has any connection to this policy server. In order to facilitate seamless reconnection with a policy server after the termination of the Tunneled EAP connection, the TNC Server (TNCS) on the policy server MAY send a TNCCS-Server-Referral Message to the TNCC on the endpoint via the Tunneled EAP connection. This message includes information necessary for the endpoint to connect to a policy server or other typo of server.

In order to successfully use the TNCCS-Server-Referral message, the TNCC receiving the message needs to convey this referral information to the component of the endpoint responsible

for establishing subsequent connections to that server. As there is no standards-based TNC interface to support such an exchange, the TNCC needs to be implemented to either use a proprietary interface, or be integrated with the appropriate network component directly.

4.2.1 TNCCS-Server-Referral

The IF-TNCCS 2.0+ message type of TNCCS-Server-Referral is indicated with a Vendor ID value of 21911 (0x005597) and a Message Type value of 6 (0x00000006). This message is sent by the TNCS to inform the TNCC of the presence of relevant TNC-enabled servers. The endpoint's own connection policy MAY provide additional guidance as to which identified server it ought to contact. Contact with any discovered server is subject to the endpoint's Client Connection Policy (as described in section 3.2) and any discovered server to which the endpoint connected would need to undergo Server Validation (as described in section 5).

The TNCS MAY include one or more messages of this type in any batch of any type. Together these messages can identify any number of servers and/or server types. Other TNCCS messages MAY also be included in the same batch.

Any TNCC (or more precisely, any TNC-enabled endpoint software with a TNCC) that claims to implement this specification MUST implement support for the TNCCS-Server-Referral message. The TNC-enabled endpoint software SHOULD act on this message as soon as possible. However, the user MAY configure their TNCC to ignore this message. A TNCC that receives one or more TNCCS-Server-Referral messages MAY ignore them but SHOULD process them and act on them if possible. TNCCS-Server-Referral messages do not convey prioritization of reported servers, so if several TNCCS-Server-Referral messages are received in a single batch identifying multiple servers of the same type and/or group, each of those servers SHOULD be considered equivalent. (However, the endpoint's pre-configured Endpoint Provisioning Information, as described in section 3, MAY indicate a prioritization of TNC-enabled servers.)

If a later batch contains one or more TNCCS-Server-Referral messages, the TNCC MUST completely replace the previous list of servers received in an earlier batch with the new list. In other words, the TNCC MUST NOT engage in a piecemeal replacement of an older list of known servers, even if the new list lacks any information about servers of a particular type that were included in a preceding TNCCS-Server-Referral message.

The NOSKIP flag in the IF-TNCCS Message Header MUST be cleared for this message type. This ensures that older TNCCs that do not implement this message will simply ignore it without ignoring other messages in the same TNCCS batch.

Some TNCSs MAY automatically send a TNCCS-Server-Referral message during any TNC handshake over IF-T for Tunneled EAP Methods to enable a smooth transition to IF-T for TLS. However, the act of sending a TNCCS-Server-Referral message MAY be subject to administrative policy and other considerations. For example, sending a TNCCS-Server-Referral message is generally not appropriate if the TNCS has already decided to reject the endpoint's network access request or if connections to other servers are not necessary. Because of these variations, servers that support IF-TNCCS exchanges and are compliant with this specification MUST have the ability to send a TNCCS-Server-Referral message, but are given wide latitude in deciding when and whether to actually send a TNCCS-Server-Referral message.

Servers identified in a TNCCS-Server-Referral message MUST NOT be trusted automatically. The TNCCS-Server-Referral message does not establish a cryptographic basis for strong authentication and authorization of a server. Therefore, contact of a discovered server MUST be constrained by the Client Collection Policy (as described in section 3.2), and a Server Validation process (as described in section 5) MUST be completed when the endpoint connects to any of the identified servers prior to the acceptance or delivery of sensitive information.

When sending a TNCCS-Server-Referral message, the IF-TNCCS Message Length field (a.k.a., PB-TNC Message Length) of the IF-TNCCS 2.0+ message (as described in section 4.2 of the IF-TNCCS 2.0 specification [2]) MUST contain the length of the entire TNCCS-Server-Referral message, including the fixed-length fields at the start of the IF-TNCCS 2.0 message (i.e., Flags,

Vendor ID, Message Type, and Message Length fields) and the fields listed below. Since some of the fields listed below are variable length, the value in the IF-TNCCS Message Length field varies also. However, it MUST always be at least 24 to cover the fixed-length fields of the IF-TNCCS 2.0 message and TNCCS-Server-Referral message contained therein. Any TNCC that receives a TNCCS-Server-Referral message with a Message Length field that contains an invalid value (e.g., less than 24) MUST respond with a fatal Invalid Parameter error code in a CLOSE batch. Particular Server Identifier Types require even longer Message Lengths to accommodate the contents of the Server Identifier field for that Server Identifier Type.

The following diagram illustrates the format and contents of the IF-TNCCS Message Value field for a TNCCS-Server-Referral message type. The text after this diagram describes the fields shown here.

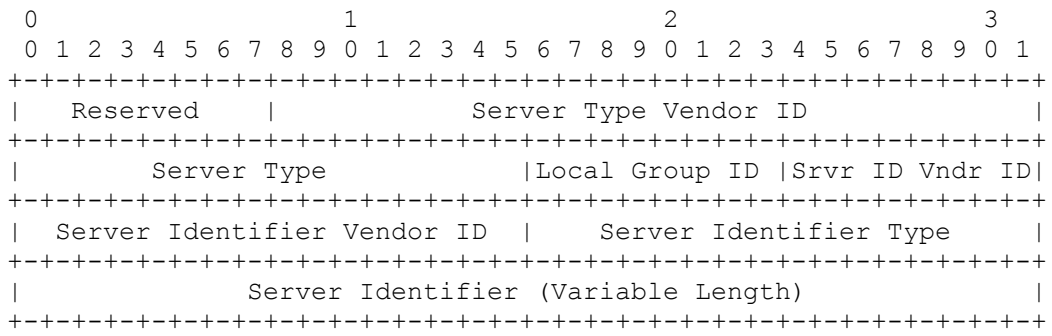

Figure 1 - TNCCS-Server-Referral Messages

Reserved (8 bits)

The Reserved field MUST be set to 0 on transmission and ignored on reception.

Server Type Vendor ID (24 bits)

The Server Type Vendor ID field identifies an implementer by using the SMI Private Enterprise Number (PEN). Any organization can receive its own unique PEN from IANA, the Internet Assigned Numbers Authority. Values unassigned by IANA SHOULD NOT be used. This specification defines some server types (in section 4.2.2) associated with the TCG's PEN of 21911 (0x005597).

Server Type (16 bits)

The Server Type field and the Server Type Vendor ID together specify the type of server identified in this message.

The TCG and any other organization with a PEN can define 2^16 unique Server Types, as long as the organization's PEN is placed in the Server Type Vendor ID field of the message. Since the Server Type is qualified by the Server Type Vendor ID, there is no risk of conflicts as long as each organization uses its own PEN for the Server Type Vendor ID and manages its own set of 2^16 server type values.

Values unassigned by the implementer whose PEN is contained in the Server Type Vendor ID field of this message SHOULD NOT be used.

This specification discusses Server Types in section 4.2.2.

Local Group ID (8 bits)

The Local Group ID field indicates a given server belongs to a locally defined group of servers. Local administrators can use this as a criterion in Client Connection Policies. Section 4.2.3 describes the use of Local Group IDs in more detail.

Server Identifier Vendor ID (24 bits)

The Server Identifier Vendor ID field identifies an implementer by using the SMI PEN. Any organization can receive its own unique PEN from IANA, the Internet Assigned Numbers Authority. Values unassigned by IANA SHOULD NOT be used. This specification defines some server identifiers (in section 4.2.4) associated with the TCG's PEN of 21911 (0x005597).

Server Identifier Type (16 bits)

The Server Identifier Type field and the Server Identifier Vendor ID together identify the type of Server Identifier contained in the Server Identifier field.

This specification discusses Server Identifier Types in section 4.2.4. Please consult that section for definitions of these identifier types. Future TCG specifications might assign further Server Identifier Type values.

Server Identifier (variable length)

The Server Identifier field contains an identifier, which can be used to establish a connection to a server. The format and semantics of the Server Identifier field depend on the Server Identifier Vendor ID and Server Identifier Type included in the same TNCCS-Server-Referral message. Section 4.2.4 describes the meaning and structure of this field when used with the Server Identifier Types defined in this specification.

The length of this field can be determined by subtracting 24 (the length of the fixed-length fields at the start of the IF-TNCCS 2.0 message and TNCCS-Server-Referral message) from the message length contained in the IF-TNCCS 2.0 Message Length field.

4.2.2 Server Types

Server types are indicated by a combination of the Server Type Vendor ID and Server Type fields. This specification defines four Server Types for use with TCG's Server Type Vendor ID of 21911 (0x005597): PDP Server, CEP Server, NEA Server, and MAP Server. These are referred to as the "TCG Core Server Types" in this specification. Table 3 shows the Server Type values for all of these types of servers. Future versions of this specification might define additional TCG Server Type values. Implementers can also define new Server Type values associated with their own SMI PEN as recorded in the Server Type Vendor ID field.

Table 3 - TCG Core Server Type Values

Type of Server	Server Type Field Value
PDP	0x0000
CEP	0x0001
NEA Server	0x0002
MAP Server	0x0003

All endpoints compliant with this specification MUST support all of the TCG Core Server Types. Specifically, Client Connection Policies for all conformant endpoint applications MUST be able to specify any of the TCG Core Server types, as identified in a TNCCS-Server-Referral message, within their Server Connection Rules. Note that those Client Connection Policies on the endpoint determine if an endpoint is allowed to or required to attempt a connection to such servers.

Although Server Type values beyond the TCG Core Server Types listed in Table 3 are permitted, including implementer-specific Server Types, TNCCs and TNCSs MUST NOT require other parties to support Server Types beyond the TCG Core Server Types listed in Table 3. TNCCs and TNCSs MUST be able to interoperate with other parties despite any differences in the set of Server Types supported (although they can rely on compliant parties supporting the TCG Core Server Types).

If a TNCC receives a TNCCS-Server-Referral message with a Server Type that it does not recognize or cannot process, the TNCC SHOULD simply ignore that TNCCS-Server-Referral message and MUST NOT treat it as an error condition.

4.2.3 Local Group IDs

The Local Group ID field allows network managers to define their own groupings of servers by characteristics other than the type of the server. These groupings can be used in the Client Connection Policy, as described in section 3.2. There are many ways in which an organization could use Local Group IDs to organize servers and control endpoint connections thereto. For example, an organization might group servers by sub-units of their organization, creating groups for Marketing, Sales, Engineering, etc. Another organization might choose to organize servers based on access levels, with highly-privileged activities facilitated by one set of servers while less-restricted access is facilitated by other servers. Many other methods of grouping servers are also possible.

The Local Group ID field allows information about such groupings to be exposed in a TNCCS-Server-Referral message. The value of 0 (0x00) is reserved to indicate that an identified server does not have an associated Local Group ID. The effective meaning of all other Local Group ID values is reflected in the Client Connection Policy where such values can be used as criteria in Server Connection Rules. Since Client Connection Policies are not intended to be portable between organizations, Local Group ID values are only meaningful within a local enclave; one organization's use of Local Group ID values might not align with another organization's use of those values.

4.2.4 Server Identifier Types

This specification defines three Server Identifier Types for use with TCG's Server Identifier Vendor ID of 21911 (0x005597): FQDN Identifier, IPv4 Identifier, and IPv6 Identifier. Future revisions of this specification might define additional Identifier Types.

Table 4 - TCG Server Identifier Type Values

Server Identifier	Server Identifier Type Field Value	Identifier Defined In
FQDN Identifier	0x0000	Section 4.2.4.2
IPv4 Identifier	0x0001	Section 4.2.4.3
IPv6 Identifier	0x0002	Section 4.2.4.4
Reserved for future use	0x0003 - 0xFFFF	

TNCCs that claim compliance with this specification MUST correctly process all three of the Server Identifier Types listed in Table 4. However, a particular TNCC might not be able to use a particular Server Identifier Type at a particular moment. For example, if the endpoint does not have an IPv6 protocol stack installed and enabled, it will not be able to use an IPv6 Server Identifier. For this reason, a TNCS ought to send multiple equivalent Server Identifier Types for each identified server when possible.

Implementers can define their own Server Identifier Types by providing a Server Identifier Type value combined with their own Server Identifier Vendor ID. The implementer is responsible for defining the structure and meaning of their Server Identifier. TNCCs and TNCSs MAY support, but MUST NOT require, Server Identifier Types beyond the three defined in this specification (i.e., FQDN Identifier, IPv4 Identifier, and IPv6 Identifier). TNCCs and TNCSs MUST be able to interoperate with other parties despite any differences in the set of Server Identifier Types supported.

If a TNCC receives a TNCCS-Server-Referral message with a Server Identifier Type that it does not recognize or cannot process, the TNCC SHOULD simply ignore this TNCCS-Server-Referral message and MUST NOT treat it as an error condition.

4.2.4.1 Protocol Fields in Server Identifiers

All three types of the Server Identifiers defined in this specification include fields to identify the network protocol to use when contacting the identified server. Server Identifiers other than the three types defined in this specification, including implementer-defined Server Identifiers, might or might not include information about the network protocol to use. The network protocol is identified by a pair of fields: the Protocol Vendor ID field and the Protocol field. The Protocol Vendor ID identifies an implementer by using the SMI PEN. Each implementer can define up to 256 different protocols, indicated by values in the Protocol field when the Protocol Vendor ID contains that implementer's SMI PEN.

The TCG, whose SMI PEN is 21911 (0x005597), defines two protocols in this specification. These protocols are indicated in Table 5.

Table 5 - TCG-Defined Protocol Field Values

Protocol	Protocol Field Value	Servers Using This Protocol
PT-TLS	0x0000	TNC PDP, TNC CEP, NEA Server
IF-MAP	0x0001	TNC MAP Server

When contacting a server identified in a Server Identifier, the Protocol Vendor ID and Protocol fields indicate the protocol to use when making that connection. Future versions of this specification might define additional TCG Protocol field values.

All TNCCs and TNCSs that conform to this specification MUST recognize the TCG-defined protocols that appear in Table 5. TNCCs and TNCSs MUST NOT require recognition of protocols beyond the TCG-defined protocols in Table 5. TNCCs and TNCSs MUST be able to interoperate with other parties despite any differences in the set of protocols appearing in Server Identifiers.

If a TNCC encounters a Protocol Vendor ID and Protocol value combination that it does not understand, it SHOULD ignore that TNCCS-Server-Referral message and MUST NOT treat it as an error condition.

4.2.4.2 FQDN Identifier

When the Server Identifier Vendor ID field of a TNCCS-Server-Referral message contains the TCG SMI PEN of 21911 (0x005597) and the Server Identifier Type field contains the value 0 (0x0000), the Server Identifier field contains an FQDN (Fully Qualified Domain Name) Identifier.

Figure 2 illustrates the format and contents of the Server Identifier field for this Server Identifier Type. The text after this diagram describes the fields shown here.

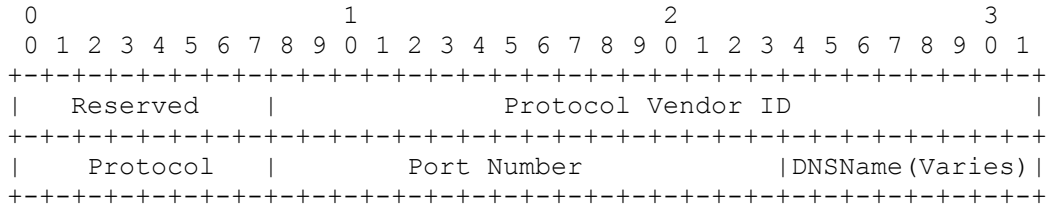

Figure 2 - FQDN Server Identifier

Reserved (8 bits)

The Reserved field MUST be set to 0 (0x00) on transmission and ignored on reception.

Protocol Vendor ID (24 bits)

The Protocol Vendor ID field identifies an implementer by using the SMI PEN. Any organization can receive its own unique PEN from IANA, the Internet Assigned Numbers Authority. Values unassigned by IANA SHOULD NOT be used. This specification defines some protocols (in section 4.2.4.1) associated with the TCG's PEN of 21911 (0x005597).

Protocol (8 bits)

The Protocol Vendor ID and Protocol fields together identify the protocol that the TNC-enabled endpoint software uses if it attempts to connect to the server whose DNS name is given. The Protocol field, along with the Protocol Vendor ID field, is discussed in section 4.2.4.1.

Port Number (16 bits)

The Port Number field indicates the port number that is used to contact the server whose DNS name is given below. The TNC-enabled endpoint software MUST use this port number as the destination port when attempting to contact the server.

DNS Name (variable length)

The DNS Name field contains the Fully Qualified Domain Name (FQDN [6]) of a server. If the TNC-enabled endpoint software uses this Server Identifier and a connection to the identified server is permitted by the Client Connection Policy, the software MUST use the specified FQDN, protocol, and port number if/when it connects to this server. In this case, the following steps are employed:

1. Wait for IP connectivity on the network interface where the Server Identifier was received.

2. Look up the DNS name given in this Server Identifier, looking for A or AAAA records. During this query, CNAME records will naturally be followed by the DNS server.

3. Attempt to connect to the resulting IP addresses using the protocol and port number specified in the Server Identifier. This connection attempt SHOULD use the network interface on which the Server Identifier was received, if possible. Note that the server that responds (which might not be the same server that was originally contacted) still needs to be verified as trustworthy, as described in section 5.

The length of the DNS Name field can be determined by subtracting the length of the fixed-length fields at the start of the TNCCS-Server-Referral message (the 24 bytes of fixed length fields in the IF-TNCCS 2.0 message and TNCCS-Server-Referral message plus the Server Identifier fields of Reserved, Protocol Vendor ID, Protocol, and Port Number) from the message length contained in the IF-TNCCS 2.0 Message Length field. The length of these fixed-length fields is 31 octets. Further, a zero length DNS name has no utility here. Therefore, any TNCC that receives a TNCCS-Server-Referral message with an FQDN Identifier with a Message Length field whose value is less than 32 MUST respond with a fatal Invalid Parameter error code in a CLOSE batch.

Note that section 5 of this document specifies how the TNC-enabled endpoint software verifies that the server eventually found through this referral mechanism is trustworthy. Implementers of TNC-enabled endpoint and server software ought to carefully review that section.

4.2.4.3 IPv4 Identifier

When the Server Identifier Vendor ID field of a TNCCS-Server-Referral message contains the TCG SMI PEN of 21911 (0x005597) and the Server Identifier Type field contains the value 1 (0x0001), the Server Identifier field contains an IPv4 Identifier.

Figure 3 illustrates the format and contents of the Server Identifier field for this Server Identifier Type. The text after this diagram describes the fields shown here.

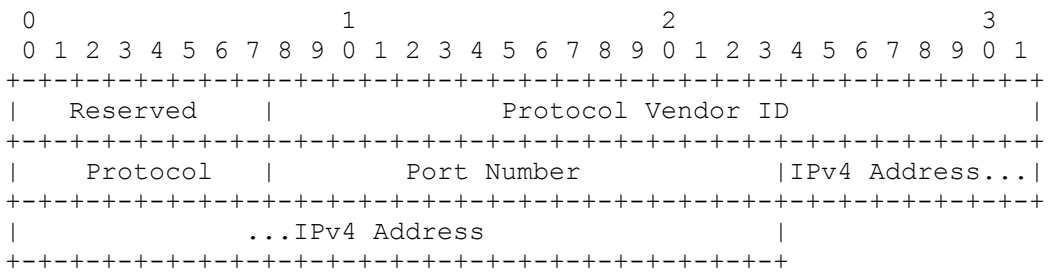

Figure 3 - IPv4 Server Identifier

Reserved (8 bits)

The Reserved field MUST be set to 0 (0x00) on transmission and ignored on reception.

Protocol Vendor ID (24 bits)

The Protocol Vendor ID field identifies an implementer by using the SMI PEN. Any organization can receive its own unique PEN from IANA, the Internet Assigned Numbers Authority. Values unassigned by IANA SHOULD NOT be used. This specification defines some protocols (in section 4.2.4.1) associated with the TCG's PEN of 21911 (0x005597).

Protocol (8 bits)

The Protocol field identifies the protocol that the TNC-enabled endpoint software will use to connect to the server whose IPv4 address is given below. The Protocol field, along with the Protocol Vendor ID field, is discussed in section 4.2.4.1.

Port Number (16 bits)

The Port Number field indicates the port number that is used to contact the server whose IPv4 address is given below. The TNC-enabled endpoint software MUST use this port number as the destination port when attempting to contact the server.

IPv4 Address (32 bits)

The IPv4 Address field contains the IPv4 address [3] of a server. If the TNC-enabled endpoint software uses this Server Identifier and a connection to the identified server is allowed by the Client Connection Policy, the software MUST use the specified IPv4 address, protocol, and port number if/when it connects to this server after IP connectivity has been achieved. The endpoint SHOULD use the network interface on which the message was received for any such connection attempt.

When an IPv4 Address Server Identifier is used, the IF-TNCCS 2.0 Message Length field for the TNCCS-Server-Referral message MUST always be 35, because the message is composed entirely of fixed-length fields whose total length is 35 octets. Therefore, any TNCC that receives a TNCCS-Server-Referral message with an IPv4 Identifier with a Message Length field whose value is not 35 MUST respond with a fatal Invalid Parameter error code in a CLOSE batch.

This identifier ought to only be used if the IPv4 address of the server in question is statically assigned. Because the list of servers identified via TNCCS-Server-Referral has no expiration, it is possible that dynamically-assigned IPv4 addresses might change after the endpoint receives the Server Identifier without the endpoint being aware of the change. This could lead to the endpoint attempting to connect to a server using the wrong IPv4 address. While the Server Validation process (described in section 5) would prevent the endpoint from trusting an entity that was not the intended server, other issues remain possible. For this reason, IPv4 identifiers are not recommended to identify a server whose address is dynamically assigned.

Note that section 5 of this document specifies how the TNC-enabled endpoint software verifies that the server eventually found through this referral mechanism is trustworthy. Implementers of TNC-enabled endpoint and server software ought to carefully review that section.

4.2.4.4 IPv6 Identifier

When the Server Identifier Vendor ID field of a TNCCS-Server-Referral message contains the TCG SMI PEN of 21911 (0x005597) and the Server Identifier Type field contains the value 2 (0x0002), the Server Identifier field contains an IPv6 Identifier.

Figure 4 illustrates the format and contents of the Server Identifier field for this Server Identifier Type. The text after this diagram describes the fields shown here.

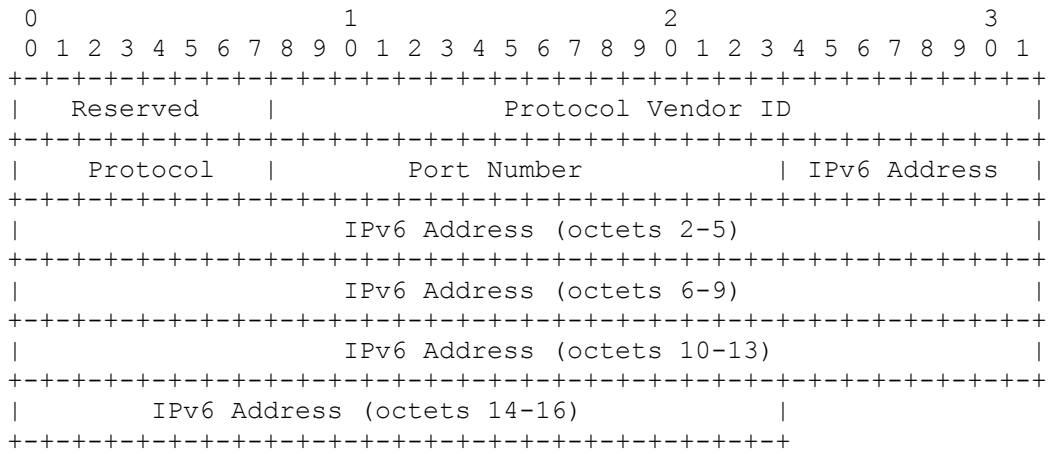

Figure 4 - IPv6 Server Identifier

Reserved (8 bits)

The Reserved field MUST be set to 0 on transmission and ignored on reception.

Protocol Vendor ID (24 bits)

The Protocol Vendor ID field identifies an implementer by using the SMI PEN. Any organization can receive its own unique PEN from IANA, the Internet Assigned Numbers Authority. Values unassigned by IANA SHOULD NOT be used. This specification defines some protocols (in section 4.2.4.1) associated with the TCG's PEN of 21911 (0x005597).

Protocol (8 bits)

The Protocol field identifies the protocol that the TNC-enabled endpoint software uses if it attempts to connect to the server whose IPv6 address is identified. The Protocol field, along with the Protocol Vendor ID field, is discussed in section 4.2.4.1.

Port Number (16 bits)

The Port Number field indicates the port number that is used to contact the server whose IPv6 address is given below. The TNC-enabled endpoint software MUST use this port number as the destination port when attempting to contact the server.

IPv6 Address (128 bits)

The IPv6 Address field contains the IPv6 address [4] of a server. The IPv6 address MUST be fully defined; IPv6 abbreviation is not permitted. If the TNC-enabled endpoint software uses this Server Identifier and a connection to the identified server is allowed by the Client Connection Policy, the software MUST use this address if/when it connects using the protocol and port number specified in the fields above once IP connectivity has been achieved. The endpoint SHOULD use the network interface on which the message was received for any such connection attempt.

When an IPv6 Address Server Identifier is used, the IF-TNCCS 2.0 Message Length field for the TNCCS-Server-Referral message MUST always be 47 because the message is composed entirely of fixed-length fields whose total length is 47 octets. Therefore, any TNCC that receives

a TNCCS-Server-Referral message with an IPv6 Identifier with a Message Length field whose value is not 47 MUST respond with a fatal Invalid Parameter error code in a CLOSE batch.

This identifier ought to only be used if the IPv6 address of the server in question is statically assigned. Because the list of servers identified via TNCCS-Server-Referral has no expiration, it is possible that dynamically-assigned IPv6 addresses might change after the endpoint receives the Server Identifier, without the endpoint being aware of the change. This could lead to the endpoint attempting to connect to a server using the wrong IPv6 address. While the Server Validation process (described in section 5) would prevent the endpoint from trusting an entity that was not the intended server, other issues remain possible. For this reason, IPv6 identifiers are not recommended to identify a server whose address is dynamically assigned.

Note that section 5 of this document specifies how the TNC-enabled endpoint software verifies that the server eventually found through this referral mechanism is trustworthy. Implementers of TNC-enabled endpoint and server software ought to carefully review that section.

4.3 Discovery via DNS SRV Records

DNS SRV [7] defines a method for discovering the proper server for a particular service. TNC-enabled endpoint software that implements this specification MUST be able to use DNS SRV as described in this section. Such software is not required to always use DNS SRV to find its servers. For example, it might be pre-configured with a specific list of servers or, if the software uses IF-TNCCS to communicate with its server, it might receive a TNCCS-Server-Referral message from a trusted server. However, TNC-enabled endpoint software that implements this specification MUST have the capability to use DNS SRV to find its servers.

To use DNS SRV to find a TNC-enabled server, the TNC-enabled endpoint software MUST use the procedure defined in the “Usage rules” section of DNS SRV [7]. The values of the service name and protocol name fields in the DNS request depend on the type of server attempting to be discovered. Table 6 identifies the values for these fields for commonly-used TNC servers. Note that values of the protocol name field in DNS request differ from the Protocol field values used in TNCCS-Server-Referral message Server Information fields. The DNS SRV procedure also requires a target domain. If the TNC-enabled endpoint software has been configured with a domain name that is associated with or governed by the target server, it SHOULD use this domain as the target domain. If the endpoint software has not been configured with such a domain name, it SHOULD use the domain name provided to it by the DHCP server in DHCP option 15 [8].

Table 6 - DNS Request Service Name and Protocol Name for Common TNC Servers

Server Type	DNS Request: Service Name	DNS Request: Protocol Name
PDP Server	_tnc-pdp	_tcp
CEP Server	_tnc-cep	_tcp
NEA Server	_nea	_tcp
MAP Server	_tnc-map	_tcp

Note the use of a leading underscore in both the service name and protocol name values in conformance with conventions established in RFC 2782. [7]

For example, an attempt to find a PDP would result in a lookup with a query string of “_tnc-pdp._tcp.*domain*”, where *domain* is the domain associated with the target server.

The response to this request will be some number of SRV Resource Records (RR). Each record is of the form:

```
_Service._Proto.Name TTL Class SRV Priority Weight Port Target
```

`_Service` and `_Proto` correspond to the service name and protocol name, respectively, in the DNS request. All fields in this RR are interpreted as described in RFC 2782, but a special caveat needs to be noted for the `Priority` field. The RFC states that “A client MUST attempt to contact the target host with the lowest-numbered priority it can reach...” This requirement remains in force, but with the understanding that an “attempt” MUST end in failure if the Client Connection Policy does not permit connection to the identified server. The client application can then attempt the next lowest priority server, and so on until an attempt completes successfully (including validation of the server), or all discovered servers are exhausted. In other words, while servers with the lowest-numbered priority are to receive first consideration, server priority does not override the Client Connection Policy.

As with servers discovered using TNCCS-Server-Referral message, servers discovered via DNS SRV lookup MUST NOT be automatically trusted. Prior to sharing sensitive data with a discovered server, the server MUST be validated using the procedure described in section 5.

5 Server Validation

Any TNC-enabled endpoint software that complies with this specification MUST be able to use the techniques described in this section to validate a server's identity and authorization when establishing a connection to the server. Any TNC-enabled server that complies with this specification MUST be able to provide the information necessary for this endpoint software to make this determination, specifically by providing and proving ownership of a server certificate during the endpoint software's connection attempt. The endpoint MUST successfully complete this validation process before sending any sensitive data to a compliant server. Likewise, the endpoint MUST NOT trust data received from a server compliant with this specification until the Server Validation process has been completed.

Any TNC-enabled server that lacks the ability to support Server Validation cannot claim conformance with this specification, even if it supports Server Discovery. Such a server MAY still be discoverable via the Server Discovery methods discussed in section 4 of this specification. TNC-enabled endpoint software MUST NOT treat any such servers as trusted, and MUST avoid sending sensitive information to or acting on instructions from such servers without taking additional precautions. Note that useful interactions with an untrusted server are possible - DNS SRV Server Discovery is an example of such an interaction, as there is no need to trust the DNS server's information, since Server Validation will later ensure the trustworthiness of servers discovered in this way.

Note that the result of Server Validation MAY be more sophisticated than a simple binary trusted/untrusted answer. One server might be trusted for some purposes (e.g., health checking) and not for others (e.g., remediation). Likewise, some servers might be more trusted for certain purposes than others. However, this document does not specify if or how an endpoint might make such distinctions, leaving it to implementers to determine if or how such information is expressed in the Client Connection Policy or used during Server Validation. Implementers MAY include mechanisms to support a determination that a server is trusted for certain tasks but not for others. Note that such a determination needs to occur during Server Validation, as none of the Server Discovery mechanisms include information designed specifically to support such determinations. Alternatively, the endpoint software MAY simply determine whether the server is trusted or not, with no additional granularity.

Note also that Server Discovery, as described in section 4, does not always need to precede Server Validation. For example, when an endpoint makes an 802.1X connection to a policy server, the server to which it connects is determined by the network rather than the endpoint. The endpoint still needs to validate this server using the techniques described in this section. Other situations could arise where the endpoint seeks to connect to a server without needing to discover that server first. Regardless of whether Server Discovery happens first, the endpoint needs to validate any server before sensitive data is sent or instructions are received.

5.1.1 Server Validation Procedure

When a client application that conforms to Server Discovery and Validation connects to a server that also conforms to this specification (whether discovered by the methods described in section 4 or identified through some other method), the server includes an X.509 certificate (henceforth the "server certificate") during the establishment of connection and proves that it possesses the private key corresponding to that certificate. This is part of the normal TLS and EAP tunnel establishment procedures used by multiple TNC-enabled servers. While this action proves to the client application that the server in question is the owner of the presented certificate (i.e., to authenticate the server identity), this is not sufficient to validate the server as trusted by the endpoint software (i.e., to determine whether the server is authorized to interact with the endpoint software). The following steps are designed to validate a TNC-enabled server as trusted (and possibly to determine the degree to which it is trusted, if the system is configured to support partial trust).

1. The TNC-enabled endpoint software MUST use the Basic Path Validation algorithm described in section 6.1 of RFC 5280 [9] to verify that the server certificate is valid, using the set of Trusted Root Certificates in its pre-configured Trust Parameters. As part of this

process, the endpoint software SHOULD verify that the server certificate and any certificates on which it depends have not been revoked. Toward this end, in the case where the connection supports TLS tunnels, the TNC-enabled server SHOULD support sending Online Certificate Status Protocol (OCSP) responses in the TLS Certificate Status Request extension [10] within the TLS tunnel, and the endpoint software SHOULD support receiving such responses and using them to verify the revocation status of the server's certificate(s). In the case where network protocols other than TLS are employed, the endpoint software and/or TNC-enabled server SHOULD support some other mechanism to check for certificate revocation, such as consulting Certificate Revocation Lists (CRLs). Checking revocation monitoring helps prevent attacks where a known bad certificate is used to impersonate a trusted server.

2. The TNC-enabled endpoint software MUST examine the names in the subject name and subject alternative name fields of the server certificate to confirm that at least one of them matches an entry in the Trusted Server Names pre-configured in the endpoint software's Trust Parameters.

   Note that the comparison is only between the endpoint software's Trusted Server Names and the names in the server certificate. In particular, the Server Identity information used by the endpoint software to contact the server is not used during validation. This allows a server to redirect the endpoint software's request to a different server without automatically invalidating the request. Such redirection might occur to support load balancing or other network requirements. In other words, the endpoint can successfully validate the server with which it is communicating, even if this is not the server that it initially attempted to contact.

   When comparing an IPv4 or IPv6 address from the server certificate, the TNC-enabled endpoint software MUST check whether there is an exact match between some entry in the list of Trusted Server Names and some subjectAltName of type iPAddress in the server certificate (as defined in section 3.1.3.2 of RFC 6125 [11]). If so, the server is trusted. If not, the server is not trusted.

   When comparing a DNS name from the server certificate, the TNC-enabled endpoint software MUST check this name against the Trusted Server Names. Note that both the DNS-format Trusted Server Name and the examined fields of the server certificate can contain wild cards. As such, this match MUST be capable of comparing two wildcarded strings. Operators need to take care that wildcarded Trusted Server Names behave as expected in the presence of the enterprise's server certificates. The endpoint software MUST use the rules defined in section 6 of RFC 6125 to verify that at least one of the server names contained in the server certificate (the "presented identifiers") matches a DNS name from the Trusted Server Names. If so, the server is trusted. If there is no match, the server is not trusted.

The following restrictions MUST apply when following the RFC 6125 section 6 rules:

- Any SRV-IDs and URI-IDs in the server certificate MUST be ignored.
- The CN-IDs in certificates SHOULD NOT be used.
- Wildcards MAY appear in the DNS Names (but not IPv4 or IPv6 addresses) used in the Trusted Server Names pre-populated on the endpoint software, which differs from the usual RFC 6125 verification.

Servers are considered valid and trusted when there is a match between the names in the TNC-enabled endpoint software's Trusted Server Names and one or more names from the server certificate. If no presented identifiers pass this RFC 6125 verification and no IP addresses match, the endpoint software SHOULD follow the instructions in section 6.6.4 of RFC 6125 for handling a validation failure.

6 Examples

The examples in this section correspond to the use cases defined in section 2.2, illustrating how they can be implemented with the standards in this document. None of the text in this section is normative.

6.1 Server Discovery and Validation with TNCCS-Server-Referral

In this example, a corporation uses 802.1X to secure its wireless network. An endpoint seeking to join the network contacts a policy server using IF-T for Tunneled EAP. The endpoint performs validation of this policy server using its Trust Parameters and the X.509 certificate presented when the Tunneled EAP connection is established. After the policy server makes the determination to allow the endpoint onto the network, the policy server sends a TNCCS-Server-Referral message - as part of the 802.1X exchange - to direct the endpoint to establish an IF-T for TLS connection to a policy server after the endpoint connects to the network and is assigned an IP address. When the endpoint connects to the network, the Client Connection Policy is processed, which contains a CONNECT rule associated with the PDP server type. Based on this, the endpoint selects a PDP that matches criteria associated with this rule and initiates a connection. The endpoint uses Server Validation to verify that the contacted server is trusted before the endpoint sends it any sensitive information.

For the purposes of this example, assume that two PDPs are deployed at each campus to ensure high availability while maintaining security. These PDPs are configured to support both IF-T for Tunneled EAP Methods and IF-T for TLS. Each wireless access point (AP) or wireless LAN controller (WLC) acting as a Policy Enforcement Point (PEP) is configured to connect to one of these PDPs with the other as a backup.

The corporate-owned endpoints are configured with a list of corporate networks (SSIDs) and with the corporate Endpoint Provisioning Information. The endpoint's Client Connection Policy contains a CONNECT rule associated with the PDP server type. This configuration takes place as part of the corporate endpoint configuration process.

Here is a walk-through of a corporate endpoint successfully connecting to the corporate wireless network:

1. The endpoint's Network Access Requester (NAR) associates with a particular corporate wireless network and PEP.
2. The PEP begins EAP authentication by sending EAP-Request/EAP-Identity to the NAR.
3. The NAR responds with EAP-Response/EAP-Identity, which the PEP forwards to the policy server over RADIUS. (Note that the Endpoint Connection Policy does not apply to this step because the network architecture, rather than the endpoint, determined the policy server to which the endpoint connects.)
4. The NAR and the policy server negotiate the use of a Tunneled EAP Method. As part of this process, the policy server sends a server certificate to the NAR.
5. During the tunnel establishment, the endpoint's TNC Client (TNCC) performs Server Validation as described in section 5. This process verifies that the policy server is trusted by the TNCC.
6. During the TNC exchange over the Tunneled EAP Method, the TNC Server (TNCS) on the policy server sends a TNCCS-Server-Referral message to the TNCC with identity information for the same policy server, which is to be used after an IP address is assigned to the endpoint.
7. At the end of the Tunneled EAP exchange, the policy server instructs the PEP to admit the endpoint to the network.
8. The endpoint gets an IP address from the DHCP server. At this point, for the purpose of this specification, the endpoint is considered to have joined the network.

9. The endpoint's Client Connection Policy includes a CONNECT rule for a PDP matching a server identified in the TNCCS-Server-Referral message. The TNCC establishes an IF-T for TLS connection to the identified policy server. As part of the establishment of this connection, the TNCC performs Server Validation.
10. After the TNC handshake has completed, the policy server will generally monitor the endpoint's health and respond to deviations from accepted policy, possibly by modifying the access provisioned to the endpoint.

6.2 Server Discovery and Validation with DNS SRV

In this example, a corporation does NOT use 802.1X to secure its wired Ethernet network. Instead, they rely on physical security, interior enforcement (e.g. firewalls, etc.), and other measures to protect their critical assets. This corporation still wishes to deploy TNC CEP Servers to monitor endpoint security and requires its endpoints to find and connect to those CEPs. Once connected to a CEP, the endpoints can report on changes in their state, allowing network management components to take appropriate actions.

In this example, the endpoint uses DNS SRV to find a local CEP to connect to and uses Server Validation to verify that this CEP is trusted. After the endpoint connects to the CEP and validates the CEP as trusted, it passes authentication, health checks, and other authorization to the CEP.

The corporate-owned endpoints are configured with the corporate Endpoint Provisioning Information and further configured to use Server Discovery via DNS SRV to attempt to connect to a corporate CEP under the corporate domain name "example.com" when possible. This configuration takes place as part of the corporate endpoint configuration process.

Here is a walk-through of a corporate endpoint successfully connecting to the corporate wired network, finding a CEP, performing a TNC handshake, and reporting endpoint state to the CEP:

1. The endpoint connects to the wired network. No authentication or health check is required.
2. The endpoint gets an IP address from the DHCP server. At this point, for the purpose of this specification, the endpoint is considered to have joined the network.
3. The TNC-enabled endpoint software performs a DNS lookup for SRV records with a service name of _tnc-cep, a protocol of _tcp, and a target domain of example.com and finds several SRV records.
4. The endpoint's Client Connection Policy includes a CONNECT rule that matches one of the servers discovered via the SRV records. This CONNECT rule tells the endpoint to attempt to connect to a matching server as soon as possible.
5. The endpoint contacts the indicated server. As part of the establishment of this connection, the TNC-enabled endpoint software performs Server Validation. If the validation succeeds, the process moves to step 6.
6. Once Server Validation has succeeded, the TNC-enabled endpoint software finishes establishing an IF-T for TLS connection to the CEP and performing a TNC handshake. The CEP might send remediation instructions to the endpoint over the TNC handshake, if necessary. The IF-T for TLS session might be maintained indefinitely to permit monitoring of the endpoint's security by conducting multiple TNC handshakes over the single IF-T for TLS session.
7. After the TNC handshake has completed, the CEP will generally publish information about the endpoint to a Configuration Management Database (CMDB), if one has been established. This information can then be used by other network management components to monitor the endpoint's health and respond to deviations from accepted policy.

7 Security Considerations

Server Discovery and Validation provides a substantial improvement in security for the TNC Architecture, because it defines exactly how an endpoint can find a server and validate that this server is trusted. While IF-T for Tunneled EAP Methods and IF-T for TLS include some guidance in this area, their guidance is not complete. Without this specification, an endpoint might disclose information about vulnerabilities to an untrustworthy server or accept malicious instructions posing as security configuration fixes.

Of course, no single specification can remove all risk from a computing system. Therefore, sections 7.1 through 7.3 analyze the risks that remain and recommend countermeasures to address these risks.

7.1 Trust Model for Discovery and Validation of Servers

The first step in analyzing the security of Server Discovery and Validation is to describe an applicable trust model, listing what each architectural element is trusted to do. The items listed here are assumptions, but provisions are made in the Threat Model and Countermeasures sections for elements that fail to perform as they were trusted to do.

7.1.1 Network

The network used to carry messages described in this specification is only trusted to:

- Perform best effort delivery of network traffic

The network is not trusted to:

- Provide confidentiality or integrity protection for messages sent over it
- Provide timely or reliable service (although an unreliable network can impede service)

7.1.2 Policy Servers

Authorized policy servers are trusted to:

- Properly perform their policy decision-making function
- In the case of PDPs, send appropriate instructions to PEPs
- Preserve the confidentiality of sensitive data obtained from endpoints
- Preserve the confidentiality of policy server credentials
- Present and prove ownership of their server credentials when an endpoint connects to them
- Send TNCCS-Server-Referral messages.

Unauthorized policy servers are not trusted at all.

7.1.3 Other TNC Servers

TNC servers that are not policy servers can provide or broker a wide range of services. As such, it is difficult to precisely characterize their roles in a trust model. In general, however, authorized TNC servers that are not policy servers can be trusted to:

- Send appropriate instructions to the endpoint
- Preserve the confidentiality of sensitive data obtained from endpoints
- Preserve the confidentiality of their server credentials
- Present and prove ownership of their server credentials when an endpoint connects to them

- Send TNCCS-Server-Referral messages if the server can participate in IF-TNCCS exchanges

Depending on their role, authorized TNC servers might be trusted to perform other activities that can impact endpoint security, state, or access, or the use of sensitive resources in the network.

7.1.4 Endpoints

Endpoints are trusted to:

- Properly perform the Server Discovery and Server Validation algorithms defined in this document
- Support secure provisioning of Endpoint Provisioning Information
- Store and protect Endpoint Provisioning Information from unauthorized modification.

An endpoint that does not perform these algorithms correctly can do considerable damage to its own security because it opens itself to attack by malicious servers. This in turn can cause the endpoint to negatively impact the security of the network overall.

7.1.5 DNS Servers

When DNS SRV is used for Server Discovery, DNS Servers do not need to be trusted since any servers discovered using this method are validated using information that is independent of what the DNS Servers provide.

7.1.6 Certification Authorities

Certification Authorities (CAs) trusted by endpoints to issue certificates for trusted servers are trusted to:

- Only issue certificates to trusted servers
- Revoke previously issued certificates as needed
- Safeguard the CA's credentials and ensure that they are not stolen or used improperly

7.2 Threat Model for Discovery and Validation of Servers

To properly ensure the security of Server Discovery and Server Validation, the threats to this trust model need to be analyzed.

7.2.1 Network Attacks

A variety of attacks can be mounted using the network. For the purposes of this subsection the phrase "network traffic" refers to messages and/or parts of messages. Any of these attacks can be mounted by network elements, by parties who control network elements, and (in many cases) by parties who control network-attached devices.

- Network traffic can be passively monitored, gleaning information from any unencrypted traffic
- Even if all traffic is encrypted, valuable information can be gained by traffic analysis (volume, timing, source and destination addresses, etc.)
- Network traffic can be modified in transit
- Previously transmitted network traffic can be replayed
- New network traffic can be added
- Network traffic can be blocked, perhaps selectively

- A “Man In The Middle” (MITM) attack can be mounted where an attacker interposes itself between two communicating parties and poses as the other end to either party or impersonates the other end to either or both parties
- Undesired network traffic can be sent in an effort to overload an architectural component, thus mounting a denial of service attack

7.2.2 Policy Server Attacks

An untrusted policy server cannot mount any significant attacks because all properly implemented endpoints and PEPs will refuse to engage in any meaningful dialog with the policy server.

On the other hand, a trusted policy server can mount several attacks:

- Request information about endpoint configuration and user/endpoint identity that it is not authorized to have
- Request user or endpoint credentials that it is not authorized to have
- Reconfigure endpoints without authorization, if remediation is enabled
- Improperly permit or deny access
- Disclose sensitive information about an endpoint
- Misrepresent endpoint information to other parties

While it is possible that a trusted policy server could send inaccurate TNCCS-Server-Referral messages, the impact of this is likely to be minimal. Even if a policy server were to identify a malicious TNC server during discovery, Server Validation should fail to validate that server and the endpoint would not trust it. At worst, the policy server could fail to provide the endpoint with any valid servers during discovery, potentially leading to denial of services to that endpoint.

7.2.3 Other TNC Server Attacks

An untrusted TNC server cannot mount any significant attacks because all properly implemented endpoints and other TNC components will refuse to engage in any sensitive dialog with an untrusted TNC server.

On the other hand, a trusted TNC server can mount attacks. The details of such attacks depend significantly on the type of TNC server in question. In general, attacks from a trusted TNC server include many of the attacks that a trusted policy server could perform. In addition, a trusted-but-malicious TNC server could mount other attacks specific to its role in the network.

As with a policy server, a TNC server that supports TNCCS could send inaccurate discovery information via a TNCCS-Server-Referral message. However, Server Validation should prevent connection to any untrusted servers. At worst, this attack should only cause a loss of services to the client through failure to identify any valid servers.

7.2.4 Endpoint Attacks

The damage that a compromised endpoint can inflict is limited, since endpoints are generally not trusted. Here are some possible effects of endpoint compromise:

- Obtain any credentials provided by the user and perhaps export those credentials to other parties or use those credentials for purposes not intended by the user
- Impersonate a user or another endpoint, if the endpoint has the necessary credentials or if no credentials are needed for impersonation (e.g. with MAC authentication)
- Provide false information about its health to a policy server
- Mount a dictionary attack or similar brute force attack against servers to attempt to guess user or endpoint credentials

- Mount a denial of service attack against servers or against other components
- Attempt to exploit vulnerabilities in servers or other components
- Fail to properly execute the algorithms described in this document, resulting in endpoint connection to an untrusted server

These potential attacks are not substantially different from those that can be mounted by any compromised endpoint, but here they are analyzed in terms of discovery and validation of servers and the endpoint's role therein.

7.2.5 Certification Authority Attacks

A compromised Certification Authority (CA) can cause significant damage if that CA is trusted by endpoints to issue certificates for trusted servers. Here are some possible effects of CA compromise:

- Certify a compromised or otherwise untrustworthy server, leading to the attacks described in sections 7.2.2 or 7.2.3
- Fail to revoke the certificate for a compromised or otherwise untrustworthy server
- Fail to safeguard the CA's private key or the credentials needed to access the CA, potentially leading to either of the attacks listed earlier in this section

Note that compromise of a Registration Authority (RA) or similar party trusted by the CA can lead to similar attacks. For example, a bad RA might recommend that the CA issue a certificate for an untrustworthy server. If the CA follows this recommendation, the attacks described in sections 7.2.2 of 7.2.3 can be mounted. This risk is inherent in any PKI system and is not increased by this specification. In fact, the recommendations and requirements in section 5 substantially reduce these risks since mere possession of a valid certificate is not sufficient for the server to be validated.

7.3 Countermeasures

This section describes recommended techniques for preventing, detecting, or mitigating the attacks described in section 7.2. Some of these techniques are required by this specification or by the TNC Architecture. Other techniques are simply recommended.

On a general note, all entities that use X.509 certificates SHOULD use the Online Certificate Status Protocol (OCSP). OCSP allows these entities to efficiently retrieve information about revocation of certificates.

7.3.1 Securing the Network

To address network attacks, the IF-T for Tunneled EAP Methods and IF-T for TLS specifications both include required encryption, authentication, integrity protection, and replay protection. When combined with the validation requirements in section 5 of this specification, these requirements reduce the threat from unauthorized network entities or from the network itself, leaving primarily traffic analysis and denial of service as potential attacks. Countermeasures for traffic analysis (e.g. masking) are often difficult but can be employed. Countermeasures for denial of service (e.g. detecting and blocking particular sources) SHOULD be used when appropriate to detect and block denial of service attacks. This is a routine practice in network security.

7.3.2 Securing Policy Servers

Properly implementing Server Validation as described in section 5 of this specification largely eliminates the threat from untrusted policy servers.

Because of the serious consequences of policy server compromise, policy servers SHOULD be especially well hardened against attack and their services minimized to reduce their attack surface. They SHOULD support serving as a TNC client to allow other network components to monitor their integrity and SHOULD utilize a Trusted Platform Module (TPM) for identity and/or integrity

measurements of the policy server. They need be well managed to minimize vulnerabilities in the underlying platform and in systems upon which the policy server depends. Network security measures such as firewalls or intrusion detection systems can be used to monitor and limit traffic to and from a policy server. Policy server administrators' use of password-based authentication is not recommended; instead use non-reusable credentials and multi-factor authentication (where available). Employ physical security measures to prevent physical attacks on policy servers.

To ease detection of policy server compromise, monitor policy server behavior to detect unusual behavior (such as a policy server reboot, unusual traffic patterns, or other odd behavior). Endpoints SHOULD log and/or notify users and/or administrators when peculiar policy server behavior is detected. To aid forensic investigation, policy servers SHOULD maintain permanent read-only audit logs of security-relevant information (especially administrative actions). If policy server compromise is detected, revoke the policy server's certificate and perform careful analysis of the source and impact of this compromise. Replace any reusable credentials that might have been compromised.

Endpoints can reduce the threat of policy server compromise by minimizing the number of trusted policy servers they are willing to validate. TNC-enabled endpoint software SHOULD use a small number of Trusted Server Names and Trusted Root Certificates for policy servers. Ensure that the CAs corresponding to the Trusted Root Certificates are managed in accordance with the best practices cited in section 7.3.6.

7.3.3 Securing Other TNC Servers

Most procedures to secure policy servers can also be applied to secure other types of TNC servers. However, depending on the role and criticality of these other TNC servers, it might not be necessary to secure those TNC servers to the same degree as policy servers. Due to their central nature in managing network access, compromise of a policy server can have an especially devastating impact. Compromise of other types of TNC servers might have a lesser impact on the network, endpoints, and sensitive resources. As such, they might not warrant the same time and expense in securing them. Network architects are advised to look at the role of and dependencies on each of their TNC servers and apply a risk-appropriate response to reducing their chances of compromise and reducing their ability to negatively impact endpoints, resources, and the network in general if they become compromised. In particular, network architects are advised to consult the security considerations portion of the appropriate TNC server specification for guidance on the potential impact of that server being compromised and ways to avoid and mitigate compromise. For example, section 6 of the TNC IF-MAP Binding for SOAP version 2.2 describes the security considerations for MAP Servers. [20]

7.3.4 Securing Endpoints

Many well-known techniques can be used to prevent the compromise of endpoints: employing anti-malware software, patching software, using TPM and TNC for health checks, etc. Certainly, these techniques ought to be employed. However, some amount of endpoint compromise is inevitable; in fact, some endpoints might be operated by malicious users. Therefore, all components need to protect themselves against compromised endpoints.

The security of the Endpoint Provisioning Information is critical to the correct functioning of the Server Validation process for a given endpoint. For this reason, endpoints MUST be capable of being securely provisioned with this information. The specific procedure by which Endpoint Provisioning Information is installed on an endpoint is left to the discretion of implementers, but implementers need to ensure that it can be done securely (e.g. by an authenticated and authorized administrator via a secure protocol). Similarly, once provisioned, the Endpoint Provisioning Information MUST be protected against tampering. Again, it is left to implementers to determine the optimal way to protect Endpoint Provisioning Information integrity in their products.

To minimize the effects of compromised endpoints, harden servers and other devices on the network to avoid exploitation of vulnerabilities on those devices. Use intrusion detection and anomaly detection systems to detect endpoint compromise as quickly as possible, and deploy Network Access Control so that compromised endpoints can be blocked from the network or

shunted to a honeynet until the source of the compromise can be tracked down and remediated. Passwords or similar reusable credentials are not recommended as user credentials, as a compromised endpoint might be able to exfiltrate such credentials to external malicious parties. Instead, use non-reusable credentials and multi-factor authentication (where available) to authenticate users. Utilize health reports from endpoints in conjunction with other sources of information about endpoint state and activities, since a compromised endpoint might lie about its health. TPM-based endpoint integrity attestation can improve the trust level of endpoint health reports. Policy servers SHOULD detect dictionary attacks and similar brute force attacks, employing countermeasures such as locking an account after a few failed logins.

7.3.5 Securing DNS Servers

DNSSEC [12] SHOULD be used to provide greater confidence in DNS results, but this is rarely supported by endpoints. Using DNSSEC significantly reduces the opportunity for and impact of compromise to the DNS Server.

The impact of a compromised DNS Server is limited, because the endpoint will use the Server Validation algorithms in section 5 to validate any server found through DNS. Still, compromised DNS Servers can impede the effectiveness of a TNC system by misdirecting endpoints. Therefore, DNS Servers SHOULD be secured as much as possible, using techniques similar to those described for securing policy servers in section 7.3.2.

7.3.6 Securing CAs

A compromised or poorly managed CA that is trusted for purposes of Server Validation can cause significant damage, as described in section 7.2.5. Therefore, the set of trusted CAs ought to be kept to a minimum, and be managed securely. The CA/Browser Forum [21] has published several good guides for secure CA management that can be useful for reference. To reduce the risk of CA compromise, the CAs used for issuing server certificates and the CAs configured as Trusted Root Certificates in the Trust Parameters ought to only be used to issue certificates for TNC-enabled servers and not for issuing other types of certificates. This reduces the chance that someone can get a server certificate through some accident, misunderstanding, or false pretense. It also diminishes the need to have a large number of Registration Authorities or to support rapid, online certificate request processing. The latter practices can be practical and necessary for issuing web site certificates, but they are not advisable for issuing server — and especially policy server — certificates, especially if the endpoints will trust the policy server enough to send endpoint configuration information to them or even to accept remediation instructions from them.

8 Privacy Considerations

Server Discovery and Validation does not directly utilize personally identifiable information (PII). However, it enables securely finding and validating a server so that an endpoint can engage in a TNC handshake. In this later handshake, PII might be transmitted.

Allowing the endpoint to be configured with Trust Parameters that select which servers are trusted improves user control over the set of parties with whom their PII is shared.

9 References

9.1 Normative References

[1] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, March 1997, IETF.

[2] Trusted Computing Group, TNC IF-TNCCS: TLV Binding, Revision 2.0, January 2010.

[3] Postel, J., Internet Protocol, RFC 791, September 1981, IETF.

[4] Hinden, R., S. Deering, IP Version 6 Addressing Architecture, RFC 4291, February 2006, IETF.

[5] Mockapetris, P., Domain Names – Concepts and Facilities, RFC 1034, November 1987

[6] Malkin, G., Internet Users' Glossary, RFC 1983, August 1996, IETF.

[7] Gulbrandsen, A., P. Vixie, L. Esibov, A DNS RR for specifying the location of services (DNS SRV), RFC 2782, February 2000, IETF.

[8] Alexander, S., R. Droms, DHCP Options and BOOTP Vendor Extensions, RFC 2132, March 1997, IETF.

[9] Cooper, D., S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280, May 2008, IETF.

[10] Eastlake 3rd, D., Transport Layer Security (TLS) Extensions: Extension Definitions, RFC 6066, January 2011, IETF.

[11] Saint-Andre, P., 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, March 2011, IETF.

[12] Arends, R., R. Austein, M. Larson, D. Massey, S. Rose, *DNS Security Introduction and Requirements*, RFC 4033, March 2005, IETF.

9.2 Informative References

[13] Trusted Computing Group, *TNC Architecture for Interoperability*, Revision 2.0, October 2017.

[14] Sangster, P., H. Khosravi, M. Mani, K. Narayan, J. Tardo, *Network Endpoint Assessment (NEA): Overview and Requirements*, RFC 5209, June 2008, IETF.

[15] Sahita, R., S. Hanna, R. Hurst, K. Narayan, PB-TNC: A Posture Broker (PB) Protocol Compatible with Trusted Network Connect (TNC), RFC 5793, March 2010, IETF.

[16] Trusted Computing Group, TNC IF-T: Binding to TLS, Revision 2.0, February 2013.

[17] Sangster, P., N. Cam-Winget, J. Salowey, A Posture Transport Protocol over TLS (PT-TLS), RFC 6876, February 2013, IETF.

[18] Trusted Computing Group, TNC IF-T: Binding to Tunneled EAP Methods, Revision 2.0, May 2014.

[19] Cam-Winget, N., P. Sangster, PT-EAP: Posture Transport (PT) Protocol For Extensible Authentication Protocol (EAP) Tunnel Methods, RFC 7171, May 2014, IETF.

[20] Trusted Computing Group, TNC IF-MAP Binding for SOAP 2.2, Revision 9, March 2014

[21] CA/Browser Forum, https://cabforum.org/