Security Target

Tanium Platform (Tanium Server 6.2 and Tanium Client 6.0)

Document Version 1.0

September 17, 2015

Prepared For:

Tanium, Inc.

1625 shattuck Avenue

Berkley, CA 94709

www.tanium.com

Prepared By:

Primasec Limited

Le Domaine de Loustalviel

11420 Pech Luna, France

www.primasec.com

Abstract

This document provides the basis for an evaluation of a specific Target of Evaluation (TOE): Tanium Platform 6.2. This Security Target (ST) defines a set of assumptions about the aspects of the environment, a list of threats that the product intends to counter, a set of security objectives, a set of security requirements, and a specification for the IT security functions provided by the TOE that meet the set of requirements.

Table of Contents

List of Tables

List of Figures

1 Introduction

This section identifies the Security Target (ST), Target of Evaluation (TOE), Security Target organization, document conventions, and terminology. It also includes an overview of the evaluated product.

1.1 ST reference

ST Title	Security Target: Tanium Platform (Tanium Server 6.2 and Tanium Client 6.0)
ST Revision	1.0
ST Publication Date	September 17, 2015
Authors	Primasec Limited and Tanium Incorporated

1.2 TOE reference

TOE Reference	Tanium Platform (Tanium Server 6.2 and Tanium Client 6.0)
TOE Type	Security management software

1.3 Document organization

This Security Target follows the following format:

SECTION	TITLE	DESCRIPTION
1	Introduction	Provides an overview of the TOE and defines the hardware and software that make up the TOE as well as the physical and logical boundaries of the TOE
2	Conformance Claims	Lists evaluation conformance to Common Criteria versions, Protection Profiles, or Packages where applicable
3	Security Problem Definition	Specifies the threats, assumptions and organisational security policies that affect the TOE
4	Security Objectives	Defines the security objectives for the TOE/operational environment and provides a rationale to demonstrate that the security objectives address the threats

SECTION	TITLE	DESCRIPTION
5	Extended Components Definition	Describes extended components of the evaluation
6	Security Requirements	Contains the functional and assurance requirements for the TOE
7	TOE Summary Specification	Identifies the IT security functions provided by the TOE and also identifies the assurance measures targeted to meet the assurance requirements.

Table 1 – ST Organization and Section Descriptions

1.4 Document conventions

The notation, formatting, and conventions used in this security target are consistent with those used in Version 3.1 of the Common Criteria. Selected presentation choices are discussed here to aid the security target reader. The Common Criteria allows several operations to be performed on functional requirements: the allowable operations defined in Part 2 of the Common Criteria are *refinement, selection, assignment* and *iteration*.

- The assignment operation is used to assign a specific value to an unspecified parameter, such as the length of a password. An assignment operation is indicated by *italicized* text, contained within square brackets.
- The refinement operation is used to add detail to a requirement, and thus further restricts a requirement. Refinement of security requirements is denoted by **bold text**. Any text removed is indicated with a strikethrough format (Example: ~~TSF~~).
- The selection operation is picking one or more items from a list in order to narrow the scope of a component element. Selections are denoted by <u>underlined</u> text, contained within square brackets.
- Iterated functional and assurance requirements are given unique identifiers by appending to the base requirement identifier from the Common Criteria an iteration number inside parenthesis, for example, FIA_UAU.1 (1) and FIA_UAU.1 (2) refer to separate instances of the FIA_UAU.1 security functional requirement component.

Outside the SFRs, italicized text is used for both official document titles and text meant to be emphasized more than plain text.

1.5 Document terminology

The following tables describe the terms and acronyms used in this document:

TERM	DEFINITION
API	Application Programming Interface

TERM	DEFINITION
CC	Common Criteria
CM	Configuration Management
EAL	Evaluation Assurance Level
GB	Giga-Byte
GUI	Graphical User Interface
I&A	Identification and Authentication
IT	Information Technology
MB	Mega-Byte
OS	Operating System
OSP	Organisational Security Policy
PC	Personal Computer
PP	Protection Profile
RAM	Random Access Memory
SOAP	Simple Object Access Protocol
SFR	Security Functional Requirement
ST	Security Target
TOE	Target of Evaluation
TSC	TOE Scope of Control
TSF	TOE Security Function
TSFI	TOE Security Function Interface

Table 2 –Acronyms Used in Security Target

1.6 TOE overview

Tanium Platform is a software product comprised of Tanium Server and Tanium Client. It is designed to provide an in-depth knowledge of the state of an enterprise environment. It allows the retrieval and distribution of information on large enterprise networks, monitoring and management of patching levels, security policies and software compliance.

In addition, Tanium Platform allows information to be retrieved by asking questions in plain English.

It is built around a peer-to-peer architecture designed to expedite those processes.

1.7 TOE Description

1.7.1 Basic components

The following diagram shows the basic components of the Tanium Platform, and their communication interactions:

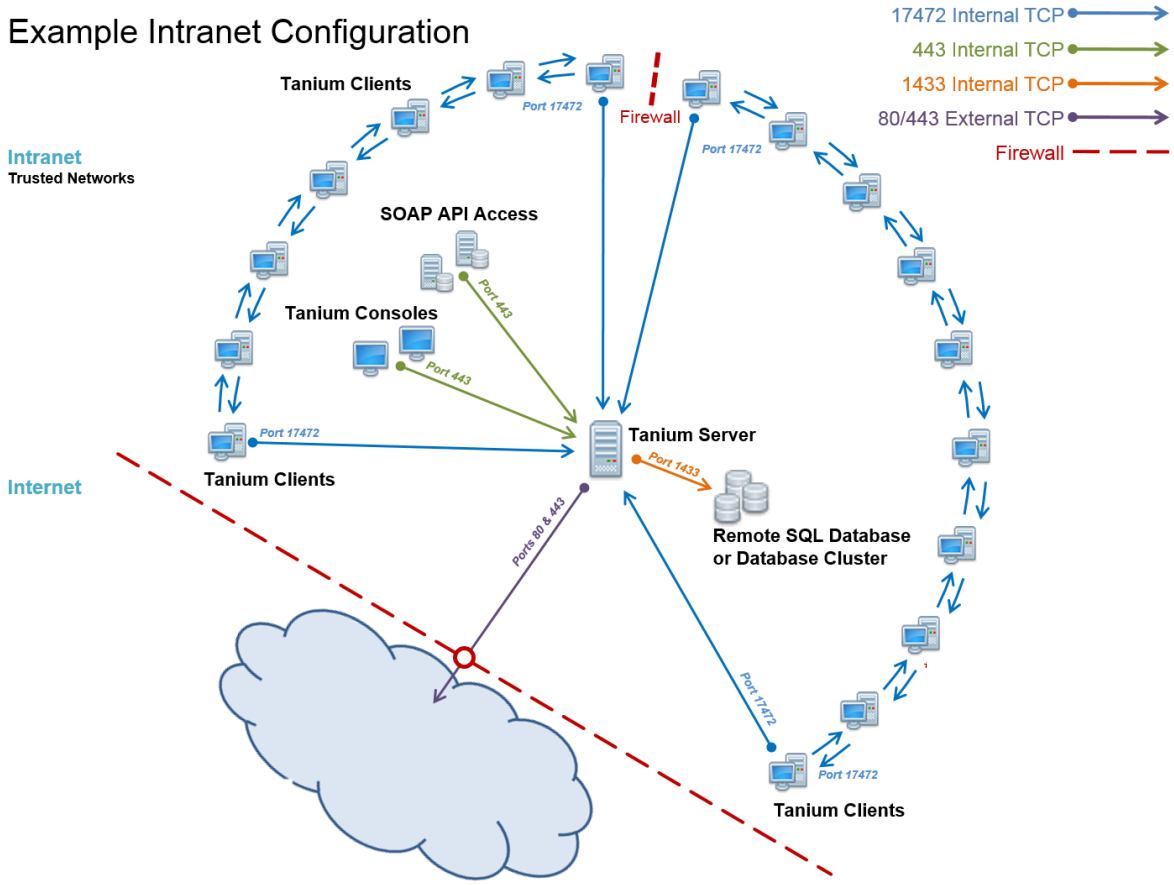


Figure 1 – Example configuration

As shown above, Tanium is comprised of a peer-to-peer Client architecture that is controlled by a single Tanium Server component. Peer-to-peer communication is used to query or modify endpoints within the network. The system is managed through the Tanium Console interface in any Flash-enabled browser. All of the data visible on the console is also available through a SOAP interface, allowing integration with other systems.

The Tanium Server component is typically installed on a Windows Server. A SQL Server database is also required. For the TOE evaluated configuration this is installed on a separate server. The Tanium Server package includes an Apache HTTP Server with a PHP database interface to the SQL Server, which together host the Console web interface used by administrators to manage Tanium Platform. In addition, the Tanium Server hosts a Tanium Server service, which handles all communication with the Clients.

The Tanium Client component is installed on each asset managed by Tanium. During installation, the Client is given the address or DNS name of the Server, as well as a public key that allows it to authenticate that traffic it sees on the peer-to-peer network originated at the Server. Other than that common information, which is the same on every Client in the environment, no per-Client administrative information needs to be provided during installation or thereafter.

In the evaluated configuration the Client is deployed via the Tanium Console.

The Tanium Console UI is a Flash application provided through an HTTPS interface, and can be accessed through all major web browsers that support Adobe Flash/Flex. Tanium Console users use their Active Directory or Windows Server credentials, coupled with Management Rights granted within Tanium, to log in to the Tanium Console.

1.7.2 Peer-to peer network

Tanium Platform is built around a peer-to-peer architecture that allows the system to scale to hundreds of thousands of Clients with a single Server, and provides nearly real-time data, with latency measured in seconds, regardless of the scale of the network. Clients are able to build this network with no manual interaction from system or network administrators in the environment.

To establish this peer-to-peer network after they have been installed, Clients maintain contact with the Server constantly sending out small packets. Based on a very small amount of data that the Server provides regarding peers in their vicinity, Clients automatically start determining which peer Clients around them are the best choices for receiving and routing data. Clients are then able to keep the network intact through aggressive routing around Clients that are removed or are unable to communicate effectively, fast addition of new Clients that come online, and the ability to utilize the Server to “reflect” around network-level blockages such as firewall blocks in core backbone routing.

The result of the process can be seen as a “ring” of Clients, with each Client having a single Client that is feeding information to it, and a single Client to which it is feeding information. A Tanium peer-to-peer ring can contain hundreds of thousands of Clients. The Clients also maintain contact with their peer Clients by constantly sending out small packets to their next and previous peers, and working through their list of close peers if the next/previous peer fails to respond.

In a Tanium Platform deployment, the peer-to-peer ring can deliver any new piece of information (Questions, Actions, Sensors, Settings, and more, as described below) to every Client in the environment in a minute or less, regardless of the scale of the network. This is possible because of optimization in the Client communications architecture, which allows a single message to be serially transmitted through over 100+ Clients per second in real networks with average LAN latency. Furthermore, because the messages being transmitted around the peer-to-peer ring are quite small, the ring can transmit 100+ messages per second to every node in the network without any appreciable load on the assets themselves, or the network infrastructure.

1.7.3 Component Interactions

This section identifies the different communication pathways in the system, and the stimuli that cause each type to occur. There are five major types of communication within the Tanium Platform:

- Registration
- Questions
- Actions
- Sensors
- Settings

1.7.3.1 Registration

The purpose of the registration process is to allow the Server to keep a record of which Clients exist in the environment, and where they are in relation to the other Clients that are currently online. It also serves as a way to "seed" the peer-to-peer network with new commands, such as the Questions, Actions, Settings, and Sensors that will be described in the following sections.

1.7.3.2 Questions

Administrators who wish to collect information about Clients in the environment can do so using the Question message. Upon creation, a Question is initially recorded into the SQL database on the Server, where it is noticed by the Tanium Server service. When the Tanium Server service determines that a new Question has been asked, it queues that Question for Clients that subsequently register.

Questions like "Computer Names and IP Addresses of computers with the Running Application is Firefox.exe and the Region is Europe" can be encoded to allow the Clients to determine what pieces of data are being requested, and what Clients should be supplying them. The question is passed from client to client, with each client supplying an answer and passing to a peer. This process continues until either the Answer section is "full" (by default, 1000 unique answers have been provided), a certain number of Clients have touched the message (by default 100), or a Client who has already seen the Question and has already answered it is reached. In any of those cases, the Question and its Answers are reported to the Server, where the Answers are submitted into the database for viewing on the Console.

1.7.3.3 Actions

In addition to being able to collect data from the entire environment using Questions, Tanium also provides the ability to take action, or make modifications. The Tanium solution provides the ability to deploy Packages that consist of a command line call and may include a set of files needed by the configured command line call. For example, if an administrator needed to uninstall an application, patch a third-party application, or make changes to a Windows registry key, they can create a

Package, upload any needed files by the Action, and deploy it to any or all machines in the environment.

Each Action and Package is signed by the Server using a FIPS 140-2 certified 512-bit ECC cryptography algorithm, and validated by the Client prior to any Action execution. When a Client receives an Action or Package from the Server, it is validated to ensure authenticity using the public key that was installed with the Client upon initial installation, which matches the private key resident on the Server.

Actions can be executed immediately or at a specified time, and can be set to recur at a specified frequency or when a certain policy is triggered.

1.7.3.4 Sensors

Sensors are the mechanism by which Tanium Clients determine the local values that they have, in order to process the Questions and Actions that reference them.

Sensors may be defined with a variety of evaluation methods including WMI, VBScript, PowerShell, and Bash Shell. Sensors are defined for a variety of hardware, software, networks, environments, operating systems, and other characteristics of the computer. By default, hundreds of Sensors are supplied with the product, and thousands of Sensors are available from the Tanium community. In addition, the system easily allows for the creation of new Sensors or modification of existing Sensors by administrators with appropriate rights in the enterprise.

The Sensor messages, similar to the Settings and Actions messages, are digitally signed by the Server, and validated locally before acceptance by Clients.

1.7.3.5 Settings

The Tanium Platform has a number of settings that control the behavior of components in the network. Examples include bandwidth caps on the Clients, timing on retries for different types of messages, and other “configurable” options. Tanium enables administrators to change these settings through the peer-to-peer environment.

1.7.4 Other Message Types

Tanium Platform communication is entirely message based. In addition to the messages that drive the components of the system mentioned above, there are additional message types that are generated, as described below.

1.7.4.1 Strings

When Clients provide Answers to Questions, and when the Server expresses Filters or other Sensor properties to the Clients, they do so using hashes in order to ensure that those Question or Answer contributions are fixed and reasonable length. When a Client sees a new Sensor result that needs to

be displayed in a human-readable format (for example, in the Tanium Console), the Client generates a new String message that encapsulates the hash value/string pairing and sends it to the Server.

1.7.4.2 String Retries

Similar to String Messages (but less frequently needed), String Retries allow the Server to gather Strings that Clients believe it has already received, but which it does not have record of receiving.

1.7.4.3 Keep Alives

Keep Alive messages are sent periodically on connections that are maintained for long periods of time, and are short messages whose primary purpose is to ensure that established connections have not been dropped by one side or the other.

1.7.5 Security and Management Rights

The Tanium security model relies on a digital signature that accompanies content as it travels around the network. This allows each Client to independently validate Sensors, Settings, and Actions that it receives legitimately originated at the Server. The signed content includes configurable expiration times and monotonically increasing IDs to prevent replay attacks. Signatures are generated using the Elliptic Curve Digital Signature Algorithm (EDDSA).

Note: All signing, as well as key management, is automatically performed by the Server component. No management of the key infrastructure is required by the enterprise.

This document uses the term “administrator” to indicate a user having privileges within the TOE that may include the ability to create new users. Login to the Console UI, where all changes are initiated, is done using the Active Directory credentials of the administrator, which are tied in Tanium to a particular user role. Access can be restricted to a subset of machines in the environment based on dynamic computer grouping. This enables Tanium to restrict a particular administrator, for example, to only be able to see Windows Servers in the UK running IIS. In addition, roles can be specified on a per-user basis to only permit viewing a subset of Questions in the environment, particular Dashboards, particular Sensors, or particular Actions. For example, a role for antivirus administrators could allow them to see answers to pre-formulated questions about Computer Names, AV Definition Levels, and Locations, and only take the few actions that are relevant to those properties.

Furthermore, to ensure that only particular administrators are able to create particular types of content, Tanium implements role-based editing rights. With a granularity that allows particular roles to have the ability to edit Sensors, Settings, Questions, and Users separately, the enterprise can segment those abilities to ensure appropriate change management controls.

1.7.6 Network Messaging Protocol

All connections in Tanium Platform are standard TCP/IP stream connections. However, communication in Tanium is Message oriented. To facilitate that, Tanium’s stream connections use a simple protocol to break the stream into messages.

1.7.7 Network Connection Protocol

1.7.7.1 Connection types

Most of the Connections in the Tanium Platform architecture are used for sending data in one direction only. Separate connections are used for sending and receiving data, even if they happen to be between the same two machines. These connections are called Outgoing connections and Incoming connections.

The Client will send messages on an Outgoing connection, and receive messages on an Incoming connection. For Peer-to-Peer communications, each Client will have two Outgoing connections (one Forward, one Backward) and will have one or more Incoming connections. Clients may accept Incoming connections from multiple peers, but will only establish Outgoing connections to one peer in each direction.

1.7.7.2 File Distribution

Most of the messages in Tanium Platform are only sent along the Forward Outgoing connection. The primary exception to this rule, are the messages used to handle File Distribution, which can flow in both directions.

1.7.8 Physical boundary

The TOE is a software TOE and includes:

1. The Tanium Server, typically installed on a Windows Server;
2. The Tanium Client, installed on each managed asset.

The computer hardware platforms that the TOE software is installed on are not part of the TOE.

The components of the TOE are installed on systems with resident operating systems, but the operating systems are not part of the TOE.

The Tanium Server requires an SQL Server database, but this does not form part of the TOE.

The following documentation provided to end users is included in the TOE boundary:

1. *Tanium Installation Guide*
2. *Tanium Console User Guide*
3. *Tanium SOAP API Guide*
4. *Tanium Common Criteria Evaluated Configuration Guide*

In order to comply with the evaluated configuration, the following software components should be used:

TOE COMPONENT	VERSION/MODEL NUMBER
TOE Software	Tanium Platform, comprised of Tanium Server 6.2 (build 6.2.314.3328) and Tanium Client 6.0 (build 6.0.314.1383)

Table 3 – Evaluated Configuration for the TOE

The following figure presents an example of an operational configuration. The shaded elements in the boxes represent the TOE components.

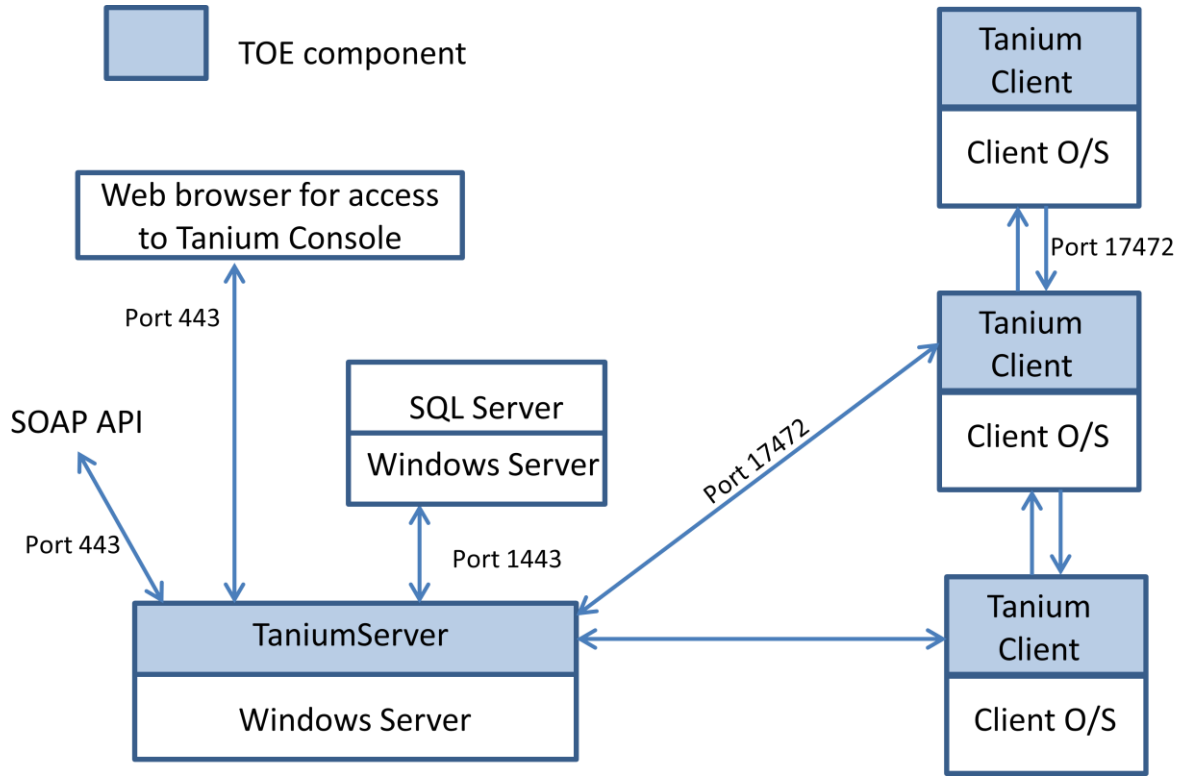


Figure 2 – TOE Boundary

1.7.9 Hardware and software supplied by the IT environment

The TOE consists of a set of software applications. The hardware, operating systems and all third party support software (e.g. SQL DBMS) on the systems on which the TOE executes are excluded from the TOE boundary.

The TOE requires the following hardware and software configuration on the Tanium Server platform.

COMPONENT	REQUIREMENTS
Hardware	x86-64 or Itanium server
Operating System	Windows Server 2008 R2 SP1 or 2012 R2 (64 bit)

Table 4 – Tanium Server Platform Requirements

The TOE requires the following supporting resources on external platforms:

COMPONENT	REQUIREMENTS
Active Directory domain controller	Windows Server 2008 R2 SP1 or 2012 R2 (64 bit)
Database support	Microsoft SQL Server 2008 R2 SP3 or 2012 SP2 (64 bit) installed on Windows Server 2008 R2 SP1 or 2012 R2 (64 bit)
Tanium Console access	Flash-enabled web browser

Table 5 – Tanium Server Environment Requirements

The supported platforms for the Tanium Client are:

COMPONENT	MINIMUM REQUIREMENTS
Operating system	Windows Server 2008 R2 SP1, Windows Server 2012 R2, Windows 7, Windows 8
Network	TCP/IP network connection

Table 6 – Managed System Platforms

1.7.10 Logical boundary

The logical boundaries of the TOE are defined by the functions provided by the TOE and are described in the following sections.

TSF	DESCRIPTION
Security audit	The TOE maintains an audit trail of administrator actions.
Cryptographic support	The TOE provides cryptographic services for message signing and verification, for hashing and token generation, and for protection of administrative sessions.
User data protection	The TOE provides functions to interrogate and configure computers on a network by transmission and receipt of messages. Messages are routed between the computers on a peer-to-peer basis, allowing their distribution and retrieval to proceed without the need for a direct connection to each computer. The client component of the TOE can execute commands on a network computer to gather information or perform actions.

TSF	DESCRIPTION
Management	The TOE provides a GUI, accessed through a browser, and protected by TLS. This interface allows access to management functions to control access to and operation of the TOE.
Identification and authentication	The TOE requires administrators to identify and authenticate themselves before accessing the TOE software. No action can be initiated before proper identification and authentication. Tanium Console users use their Active Directory or Windows Server credentials, coupled with management rights granted within Tanium, to log in to the Tanium Console.
Resource utilisation	The TOE can select alternative routes for the transmission of messages to bypass network computers that are unavailable.
Protection of the TSF	The TOE provides a digital signature for messages sent from server to client to ensure that modified and spoof messages can be detected and rejected.

Table 7 – Logical Boundary Descriptions

1.7.11 TOE data

TOE data consists of both TSF data and user data (query and action request data). TSF data consists of authentication data, security attributes, cryptographic keys and other generic configuration information. Security attributes enable the TSF to enforce the security policy. Authentication data enables the TSF to identify and authenticate users.

1.7.12 Items not within the scope of the evaluation

Use of a Tanium Zone Server.

SQL database installed on the same platform as the Tanium Server.

Installation of the client using Active Directory/Group Policy Object, software distribution systems such as System Center Configuration Manager or Tivoli, login scripts, and Network Access Control (NAC) policies.

1.8 Rationale for non-bypassability and separation of the TOE

The TOE is an application that executes on top of an underlying hardware system in conjunction with a Windows operating system. Responsibility for non-bypassability and separation are split between the TOE, the OS and the hardware IT Environment.

All access to objects in the TOE IT environment is validated by the IT environment security policies before they can succeed. An attacker will not be able to access any of the TOE security functions or

any of the TOE files or directories. Arbitrary entry into the TOE is not possible, and therefore the TSF is protected against external interference by untrusted objects.

The TOE provides strictly controlled functionality to the users within the TSC. By limiting access through role based access control, the TSF is protected from corruption or compromise from users within the TSC. The TOE interfaces are separated into 2 categories – security enforcing and security supporting. Security enforcing interfaces invoke the TSF and ensure that all enforcement functions complete successfully before allowing the user invoked action to proceed. Security supporting interfaces ensure that the TSF cannot be interfered with via those interfaces (i.e. they are isolated from the TSF). The security enforcing role is separate from the security supporting role and each role has its own unique set of privileges associated with it. Multiple simultaneous users (and roles) are supported.

The hardware and OS provide separate process spaces in which the TOE executes; these process spaces are protected from interference from other processes except through the defined TOE interfaces. Processes are separate from each other, each with their own memory buffer and it is impossible for one process to directly access the memory of another. The OS and hardware support non-bypassability by ensuring that all access to resources designated to be encrypted is mediated through the TOE.

2 Conformance claims

2.1 Common Criteria conformance claim

The TOE is Common Criteria Version 3.1 Revision 4 (September 2012) Part 2 extended and Part 3 conformant at Evaluation Assurance Level 2 and augmented by ALC_FLR.2 – Flaw Reporting Procedures.

2.2 Protection Profile conformance claim

The TOE does not claim conformance to any Protection Profile.

3 Security problem definition

In order to specify the nature of the security problem that the TOE is intended to solve, this section describes the following:

- Known or assumed threats to the assets against which specific protection within the TOE or its environment is required.
- Organisational security policy statements or rules with which the TOE must comply.
- Assumptions about the security aspects of the environment and/or of the manner in which the TOE is intended to be used.

This chapter identifies assumptions as A.*assumption*, threats as T.*threat* and policies as P.*policy*.

3.1 Threats

The following are threats identified for the TOE and the information the TOE protects. The TOE itself has threats and the TOE is also responsible for addressing threats to the environment in which it resides. The assumed level of expertise of the attacker for all the threats is unsophisticated.

The TOE addresses the following threats:

THREAT	DESCRIPTION
T.BAD_CONFIG	An attacker may exploit vulnerabilities on network computers that are not configured in line with organizational policies.
T.SLOW_DEFEND	Information about the state of network computers may not be available in a timely manner, preventing the rapid detection of policy violations and distribution of updates, allowing vulnerabilities to be exploited by an attacker.
T.UNAUTH_INFO	An unauthorized person may gain access to information about network computers, allowing vulnerabilities to be exploited by an attacker.
T.BAD_UPDATES	An unauthorized person may modify the configuration of network computers to compromise the security of those computers.

Table 8 – Threats Addressed by the TOE

3.2 Organisational security policies

There are no organisational security policies applicable to the TOE.

3.3 Assumptions

This section describes the security aspects of the environment in which the TOE is intended to be used. The TOE is assured to provide effective security measures in a co-operative non-hostile environment only if it is installed, managed, and used correctly. The following specific conditions are assumed to exist in an environment where the TOE is employed. If the TOE is placed in an operational environment that does not meet these assumptions, the TOE may not be able to provide all of its security functionality. Assumptions can be on physical, personnel and connectivity of the operational environment.

ASSUMPTION	DESCRIPTION
A. PLATFORMI&A	Active Directory or Windows Server credentials are configured for administrators of the TOE.
A.TRAINED_USERS	Authorized administrators are appropriately trained and follow all appropriate guidance documentation.
A.TIME	The IT Environment will provide reliable timestamps for use by the TOE.

Table 9 – Assumptions

4 Security objectives

The security objectives for the Target of Evaluation (TOE) and for the operational environment are derived from the threats and assumptions in Section 3.

4.1 Security objectives for the TOE

The IT security objectives for the TOE are addressed below. The TOE has to meet these objectives by satisfying the security functional requirements.

OBJECTIVE	DESCRIPTION
O.IDAUTH	Administrators must be identified and authenticated before being allowed access to TOE functions.
O.ROLES	Administrators must be able to control access to TOE functions based on assigned roles.
O.ACCOUNT	Administrators must be accountable for their actions.
O.INFO	The TOE must provide and control the means to establish the configuration of computers on a network.
O.ACTION	The TOE must provide and control the means to execute actions on computers on a network.
O.GOOD_ORIGIN	The TOE must provide the means to validate the origin of requests for action.
O.CONTACT	The TOE must be able to contact all available computers on the network that are registered to the TOE.
O.PROTECT	The TOE must protect all sensitive configuration data transmitted between the TOE and the administrator.

Table 10 – TOE Security Objectives

4.2 Security objectives for the operational environment

The security objectives for the operational environment are addressed below:

OBJECTIVE	DESCRIPTION
OE.PASSWORD_STRENGTH	An authorized administrator must ensure that passwords conform to appropriate guidance from the organization using the TOE.
OE.PLATFORM_I&A	The Operational Environment must provide individual user identification and authentication mechanisms that operate independently of the TOE.
OE.TRAINED_USERS	Authorized users and administrators must be properly trained and follow all guidance for securing the TOE and authorization factors.
OE.TIME	The TOE environment must provide reliable timestamps to the TOE.

Table 11 – Operational Environment Security Objectives

4.3 Security objectives rationale

This section provides the summary that all security objectives are traced back to aspects of the addressed assumptions and threats. The following table provides a high level mapping of coverage for each threat and assumption.

	T.BAD_CONFIG	T.SLOW_DEFEND	T.UNAUTH_INFO	T.BAD_UPDATES	A.PLATFRMI&A	A.TRAINED_USERS	A.TIME
O.IDAUTH			X	X	X		
O.ROLES			X	X			
O.ACCOUNT			X	X			
O.INFO	X	X	X				
O.ACTION	X			X			
O.GOOD_ORIGIN			X	X			
O.CONTACT	X	X					
O.PROTECT			X	X			
OE.PASSWORD_STRENGTH			X	X			
OE.PLATFORM_I&A			X	X	X		
OE.TRAINED_USERS	X					X	
OE.TIME							X

Table 12 – Mapping of Assumptions, Threats, and OSPs to Security Objectives

The following table provides detailed evidence of the mapping between security objectives and assumptions.

THREATS, POLICIES, AND ASSUMPTIONS	ADDRESSED BY	RATIONALE
T.BAD_CONFIG	O.INFO O.ACTION O.CONTACT OE.TRAINED_USERS	The authorized administrator must be able to query all available clients (under their control) on the network to determine their configuration and to be able to distribute any necessary actions to perform software or settings updates to all available clients. The administrators must have the

THREATS, POLICIES, AND ASSUMPTIONS	ADDRESSED BY	RATIONALE
		necessary training and follow the guidance to competently complete these actions.
T.SLOW_DEFEND	O.INFO O.CONTACT	The TOE must provide a method of communicating with all registered clients on the network (to query the configuration and provide updates) in a quick manner that does not present any single points of failure preventing transmission of messages around the network.
T.UNAUTH_INFO	O.IDAUTH O.ROLES O.ACCOUNT O.INFO O.GOOD_ORIGIN O.PROTECT OE.PASSWORD_STRENGTH OE.PLATFORM_I&A	The TOE must ensure only authorized administrators are able to access the management functions, and that administrators are held accountable for their actions. To effectively achieve this, the underlying TOE platform must enforce the use of strong passwords to authenticate the user to the platform and then to pass through the authorization to the TOE. When the administrator communicates with the server and questions are distributed within the network the communication must be protected from unauthorized modification.
T.BAD_UPDATES	O.IDAUTH O.ROLES O.ACCOUNT O.ACTION O.GOOD_ORIGIN O.PROTECT OE.PASSWORD_STRENGTH OE.PLATFORM_I&A	The TOE must ensure only authorized administrators are able to access the management functions, and that administrators are held accountable for their actions. To effectively achieve this, the underlying TOE platform must enforce the use of strong passwords to authenticate the user to the platform and then to pass through the authorization to the TOE. When the administrator communicates with the server and actions are distributed within the network the communication must be protected from unauthorized modification.
A. PLATFORMI&A	O.IDAUTH OE.PLATFORM_I&A	The underlying TOE platform must enforce the identification and authentication of the user to the platform and then to pass through the authorization to the TOE.
A.TRAINED_USERS	OE.TRAINED_USERS	The administrators must have the necessary training and follow the guidance to competently complete these actions.

THREATS, POLICIES, AND ASSUMPTIONS	ADDRESSED BY	RATIONALE
A.TIME	OE.TIME	The underlying platform must provide a reliable timestamp for the TOE to use to record date/time of events in the audit trail.

Table 13 – Rationale for Mapping of Threats, Policies, and Assumptions to Objectives

5 Extended Components Definition

5.1 Introduction

This section provides definitions for CC Part 2 extended components that are used within this ST.

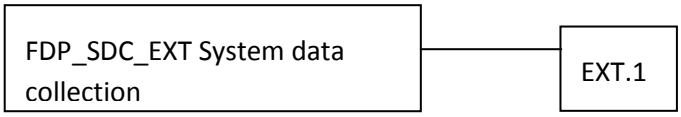
5.2 Extended components

5.2.1 User data protection: System data collection (FDP_SDC_EXT)

Family Behaviour

This family is added to the class FDP. This family deals with the generation and management of queries across a network.

Component Levelling



FDP_SDC_EXT.1 requires the ability to generate, distribute and execute queries on a network, and to retrieve and report on the results.

Management: FDP_SDC_EXT.1

The following actions could be considered for the management functions in FMT:

a) Control of permissions to carry out system data collection activities.

Audit: FDP_SDC_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: Distribution of a query.

FDP_SDC_EXT.1 System data collection

Hierarchical to: No other components.

Dependencies: None

FDP_SDC_EXT.1.1 The TSF shall distribute queries to network clients.

FDP_SDC_EXT.1.2 The TSF shall execute queries that have been distributed to network clients and collect the responses.

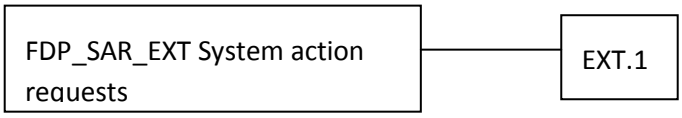
FDP_SDC_EXT.1.3 The TSF shall collate and report on responses to network queries.

5.2.2 User data protection: System action requests (FDP_SAR_EXT)

Family Behaviour

This family is added to the class FDP. This family deals with the distribution of action request across a network.

Component Levelling



FDP_SAR_EXT.1 requires the ability to distribute action requests to network computers, allowing execution of commands or deployment of packages.

Management: FDP_SAR_EXT.1

The following actions could be considered for the management functions in FMT:

a) Control of permissions to distribute system actions.

Audit: FDP_SAR_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: Distribution of an action.

FDP_SAR_EXT System action request

Hierarchical to: No other components.

Dependencies: None

FDP_SAR_EXT.1.1 The TSF shall distribute actions to network clients.

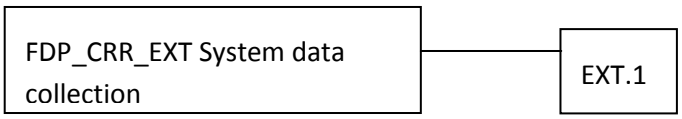
FDP_SAR_EXT.1.2 The TSF shall execute actions that have been distributed to network clients.

5.2.3 User data protection: System registration request (FDP_CRR_EXT)

Family Behaviour

This family is added to the class FDP. This family deals with the generation and response to registration requests from the network client component.

Component Levelling



FDP_CRR_EXT.1 requires the ability to generate and send registration requests from the client and receive responses from the server component.

Management: FDP_CRR_EXT.1

The following actions could be considered for the management functions in FMT:

a) None.

Audit: FDP_CRR_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: Distribution of a registration request

b) Minimal: Distribution of a settings message.

FDP_CRR_EXT.1 System data collection

Hierarchical to: No other components.

Dependencies: None

FDP_CRR_EXT.1.1 The TSF shall send a registration request to the server, including details of status statistics.

FDP_CRR_EXT.1.2 The TSF will respond with a Settings message.

6 Security Requirements

The security requirements that are levied on the TOE and the IT environment are specified in this section of the ST.

6.1 Security functional requirements

The functional security requirements for this Security Target consist of the following components from Part 2 of the CC, and those that were explicitly stated, all of which are summarized in the following table:

CLASS HEADING	COMPONENT	DESCRIPTION
Identification and authentication Class (FIA)	FIA_ATD.1	User attribute definition
	FIA_UAU.2	User authentication before any action
	FIA_UID.2	User identification before any action
	FIA_USB.1	User subject binding
Security management Class (FMT)	FMT_SMR.1	Security management roles
	FMT_MTD.1	Management of TSF data
	FMT_SMF.1	Specification of management functions
User data protection Class (FDP)	FDP_SAR_EXT.1	System action requests
	FDP_SDC_EXT.1	System data collection
	FDP_CRR_EXT.1	System registration request
	FDP_DAU.2	Data authentication with identity of guarantor
Cryptographic support (FCS)	FCS_CKM.1	Cryptographic key generation
	FCS_CKM.4	Cryptographic key destruction
	FCS_COP.1(1)	Cryptographic operation (DSA)
	FCS_COP.1(2)	Cryptographic operation (SHA)
	FCS_COP.1(3)	Cryptographic operation (AES)
	FCS_COP.1(4)	Cryptographic operation (RSA)
Resource utilization (FRU)	FRU_FLT.1	Degraded fault tolerance
Protection of the TSF Class (FPT)	FPT_FLS.1	Failure with preservation of secure state
Trusted path/channels (FTP)	FTP_TRP.1	Trusted path
Security Audit (FAU)	FAU_GEN.1	Audit Data Generation

Table 14 – TOE Security functional Components

6.1.1 Identification and authentication (FIA)

6.1.1.1 FIA_ATD.1 User attribute definition

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users: [*user name, group membership*]

6.1.1.2 FIA_UAU.2 User authentication before any action

FIA_UAU.2.1 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

6.1.1.3 FIA_UID.2 User identification before any action

FIA_UID.2.1 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

6.1.1.4 FIA_USB.1 User-subject binding

FIA_USB.1.1 The TSF shall associate the following user security attributes with subjects acting on behalf of that user: [

a. User name;

b. Group membership].

FIA_USB.1.2 The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: [*user security attributes are bound upon successful login with a valid user name*].

FIA_USB.1.3 The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: [*user security attributes do not change during a session*].

Application Note: *The TOE binds security attributes to subjects for Tanium sessions. Windows binds security attributes to subjects for workstation sessions. Permissions are determined by the union of all permissions in any permission set associated with a user. If the security attributes for a user are changed while that user has an active session, the new security attributes are not bound to a session until the next login.*

6.1.2 Security management (FMT)

6.1.2.1 FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles [*Administrator with assigned privileges*].

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

Application Note: *Administrators assigned to one of the following user types, are considered to be 'authorized administrators':*

- *Content Administrator*
- *Action Sensor Author*

- *Sensor Author*
- *Action Author*
- *Action Approver*
- *Action User*
- *Question Author*
- *Read-Only User*

The functions each user type is permitted to perform are defined in Table 15.

The authorized administrator permissions are restricted by Computer Group to which the user is assigned. i.e. the user is only able to perform the associated functions on those computers within the group(s) to which the user is assigned.

6.1.2.2 FMT_MTD.1 Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to [change-default, query, modify, delete, [*execute, import, deploy*]] the [

a) Global platform settings, notifications feature, user login IDs, console user groups and computer groups;

b) Dashboards;

c) Questions;

d) Actions;

e) Sensors;

f) *Action/package*]

to [*an authorized administrator*].

Application Note: An "authorized user" is considered to be an administrator, assigned to a user type, with permission to perform the associated action, as defined in Table 15.

6.1.2.3 FMT_SMF.1 Specification of management functions

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions: [*All actions specified in the first column of Table 15*].

Actions	Administ-rator	Content Administ-rator	Action Sensor Author	Sensor Author	Action Author	Action Approver	Action user	Question Author	Read-Only User
Platform settings, Console Users, Computer Assignment									
Create or configure Global platform settings, Notifications feature, Console Users Login ID's, Console User Groups, & Computer Groups	✓								
Import Content and Manage Notifications	✓	✓							
Dashboards, Questions, Sensors									
Execute Dashboards	✓	✓	✓	✓	✓	✓	✓	✓	✓
Create Dashboards and view Dashboard details	✓	✓	✓	✓	✓	✓		✓	
Create Dashboard Groups, View Dashboard Group details, & Assign Dashboard Group Visibility	✓	✓							
Ask dynamic questions, save questions, and view saved question details	✓	✓	✓	✓	✓	✓	✓	✓	
Assign Default Action to Saved Question	✓	✓	✓						
Create Sensors and View Sensors Details	✓	✓	✓	✓					

Actions	Administ-rator	Content Administ-rator	Action Sensor Author	Sensor Author	Action Author	Action Approver	Action user	Question Author	Read-Only User
Action Groups, Actions, Packages									
Deploy Action/Package associated with a saved question, create Scheduled Actions & view all Action history	✓	✓	✓		✓	✓	✓		
Deploy any Action/Package, create Packages, and view Package details	✓	✓	✓						
View and manage any Scheduled Actions, create and manage Action Groups	✓	✓							
View and manage only Scheduled Actions created by the console user			✓		✓	✓	✓		

Table 15 – Available user types and associated permissions

6.1.3 User data protection (FDP)

6.1.3.1 FDP_SAR_EXT.1 System action requests

FDP_SAR_EXT.1.1	The TSF shall distribute actions to network clients.
Application Note:	In this component the TSF refers to the server component.
FDP_SAR_EXT.1.2	The TSF shall execute actions that have been distributed to network clients.
Application Note:	In this component the TSF refers to the client component.
Application Note:	The distributed actions are as defined in Section 7.3.2.3.

6.1.3.2 FDP_SDC_EXT.1 System data collection

FDP_SDC_EXT.1.1	The TSF shall distribute queries to network clients.
Application Note:	*In the above component the TSF refers to the server component.*
FDP_SDC_EXT.1.2	The TSF shall execute queries that have been distributed to network clients and collect the responses.
Application Note:	*In the above component the TSF refers to the client component.*
FDP_SDC_EXT.1.3	The TSF shall collate and report on responses to network queries.
Application Note:	*The distributed questions are as defined in section* 7.3.2.2.

6.1.3.3 FDP_CRR_EXT.1 Client registration requests

FDP_CRR_EXT.1.1	The TSF shall send a registration request to the server, including details of status statistics.
Application Note:	*In the above component the TSF refers to the client component.*
FDP_CRR_EXT.1.2	The TSF will respond with a Settings message.
Application Note:	*In the above component the TSF refers to the server component.*
Application Note:	*The registration request is described in section 7.3.2.1 and the settings message is described in section* 7.3.2.5*.*

6.1.3.4 FDP_DAU.2 Data authentication with identity of guarantor

FDP_DAU.2.1	The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of [*sensors, settings and actions*].
FDP_DAU.2.2	The TSF shall provide [*clients on a network*] with the ability to verify evidence of the validity of the indicated information and the identity of the user that generated the evidence.
Application Note:	*The evidence used as a guarantee of validity are the digital signatures detailed in FCS_COP.1.1(1) below. Action, sensor and setting messages transferred from*

server to client are protected from undetected modification by including a digital signature with all content originating from the server.

6.1.4 Cryptographic support (FCS)

6.1.4.1 FCS_CKM.1 Cryptographic key generation

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [*FIPS approved Random Number Generator*] and specified cryptographic key sizes [*128, 256 bits*] that meet the following: [*NIST SP 800-90 DRBG*].

6.1.4.2 FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [*zeroization*] that meets the following: [*FIPS 140-2 level 1*].

6.1.4.3 FCS_COP.1(1) Cryptographic operation (DSA)

FCS_COP.1.1(1) The TSF shall perform [*digital signature*] in accordance with a specified cryptographic algorithm [*ECDSA*] and cryptographic key sizes [*"NIST curve" P-521*] that meet the following: [*FIPS PUB 186-3 "Digital Signature Standard"*].

6.1.4.4 FCS_COP.1(2) Cryptographic operation (SHA)

FCS_COP.1.1(2) The TSF shall perform [*cryptographic hashing services*] in accordance with a specified cryptographic algorithm [*SHA-2(256, 512)*] and **message digest** sizes [256 bits, 512 bits] that meet the following: [*FIPS PUB 180-3 "Secure Hash Standard"*].

6.1.4.5 FCS_COP.1(3) Cryptographic operation (AES)

FCS_COP.1.1(3) The TSF shall perform [*encryption/decryption of administration session traffic*] in accordance with a specified cryptographic algorithm [*AES*] and cryptographic key sizes [128, 256 bits] that meet the following: [*FIPS PUB 197 "Advanced Encryption Standard"*].

6.1.4.6 FCS_COP.1(4) Cryptographic operation (RSA)

FCS_COP.1.1(4) The TSF shall perform [*encryption/decryption*] in accordance with a specified cryptographic algorithm [*RSA*] and cryptographic key sizes [2048 bits] that meet the following: [*RFC 2246*].

6.1.5 Resource utilisation (FRU)

6.1.5.1 FRU_FLT.1 Degraded fault tolerance

FRU_FLT.1.1 The TSF shall ensure the operation of [*message transmission around a network*] when the following failures occur: [*inaccessibility of a client on the network*].

6.1.6 Protection of the TSF (FPT)

6.1.6.1 FPT_FLS.1 Failure with preservation of secure state

FPT_FLS.1 The TSF shall preserve a secure state when the following types of failures occur: [*inaccessibility of a computer on a network*].

6.1.7 Trusted path/channels (FTP)

6.1.7.1 FTP_TRP.1 Trusted path

FTP_TRP.1.1 The TSF shall provide a communication path between itself and [remote] users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from [modification and disclosure].

FTP_TRP.1.2 The TSF shall permit [remote users] to initiate communication via the trusted path.

FTP_TRP.1.3 The TSF shall require the use of the trusted path for [initial user authentication and *protection of administration session traffic*].

6.1.8 Security Audit (FAU)

6.1.8.1 Audit Data Generation (FAU)

FAU_GEN.1.1 The TSF shall be able to generate an audit record of the following auditable events:

a. Start-up and shutdown of the audit functions;

b. All auditable events, for the [*not specified*] level of audit; and

c. [*all actions taken by the administrator;*

d. *distribution of a query;*

e. *distribution of an action;*

f. distribution of a registration request;

g. *distribution of a settings message*].

FAU_GEN.1.2 The TSF shall record within each audit record at least the following information:

a. Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and

b. For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [*no other audit-relevant information*].

6.2 Security assurance requirements

The assurance security requirements for this Security Target are taken from Part 3 of the CC. These assurance requirements compose an Evaluation Assurance Level 2 (EAL2) augmented by ALC_FLR.2. The assurance components are summarized in the following table:

CLASS HEADING	CLASS_FAMILY	DESCRIPTION
ADV: Development	ADV_ARC.1	Security architecture description
	ADV_FSP.2	Security-enforcing functional specification
	ADV_TDS.1	Basic design
AGD: Guidance Documents	AGD_OPE.1	Operational user guidance
	AGD_PRE.1	Preparative procedures
ALC: Lifecycle Support	ALC_CMC.2	Use of a CM system
	ALC_CMS.2	Parts of the TOE CM coverage
	ALC_DEL.1	Delivery procedures
	ALC_FLR.2	Flaw reporting procedures
ATE: Tests	ATE_COV.1	Evidence of coverage
	ATE_FUN.1	Functional testing
	ATE_IND.2	Independent testing - sample
AVA: Vulnerability Assessment	AVA_VAN.2	Vulnerability analysis

Table 16 – Security Assurance Requirements

6.3 CC component hierarchies and dependencies

This section of the ST demonstrates that the identified SFRs include the appropriate hierarchy and dependencies. The following table lists the TOE SFRs and the SFRs each are hierarchical to, dependent upon and any necessary rationale.

SFR	HIERARCHICAL TO	DEPENDENCY	RATIONALE
FIA_ATD.1	None	None	N/A
FIA_UAU.2	FIA_UAU.1	FIA_UID.1	Satisfied by inclusion of FIA_UID.2
FIA_UID.2	FIA_UID.1	None	N/A

SFR	HIERARCHICAL TO	DEPENDENCY	RATIONALE
FIA_USB.1	None	FIA_ATD.1	Satisfied
FMT_SMR.1	None	FIA_UID.1	Satisfied by inclusion of FIA_UID.2
FMT_MTD.1	None	FMT_SMF.1, FMT_SMR.1	Satisfied Satisfied
FMT_SMF.1	None	None	N/A
FDP_SAR_EXT.1	None	None	N/A
FDP_SDC_EXT.1	None	None	N/A
FDP_CRR_EXT.2	None	None	N/A
FDP_DAU.2	FDP_DAU.1	FIA_UID.1	Satisfied by inclusion of FIA_UID.2 (although not necessary as the proof of origin relates to the server rather than any individual)
FCS_CKM.1	None	FCS_CKM.2 or FCS_COP.1, FCS_CKM.4	Met by inclusion of FCS_COP.1 and FCS_CKM.4
FCS_CKM.4	None	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	Met using FCS_CKM.1
FCS_COP.1(all)	None	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1, FCS_CKM.4	Satisfied by inclusion of FCS_CKM.1 and FCS_CKM.4
FRU_FLT.1	None	FPT_FLS.1	Satisfied
FPT_FLS.1	None	None	N/A
FTP_TRP.1	None	None	N/A
FAU_GEN.1	None	FPT_STM.1	Although FPT_STM.1 is not included, the TOE Environment provides reliable timestamps to the TOE. An environmental objective states that the TOE will receive reliable timestamps, thereby satisfying this dependency

Table 17 – TOE SFR Dependency Rationale

6.4 Security requirements rationale

This section provides rationale for the Security Functional Requirements demonstrating that the SFRs are suitable to address the security objectives

6.4.1 Security functional requirements for the TOE

The following table provides a high level mapping of coverage for each security objective:

	O.IDAUTH	O.ROLES	O.ACCOUNT	O.INFO	O.ACTION	O.GOOD_ORIGIN	O.CONTACT	O.PROTECT
FAU_GEN.1			X					
FCS_COP.1(1) *DSA*						X		
FCS_COP.1(2) *SHA*						X		X
FCS_COP.1(3) *AES*								X
FCS_COP.1(4) RSA								X
FCS_CKM.1						X		X
FCS_CKM.4						X		X
FDP_DAU.2						X		
FDP_SAR.EXT.1					X			
FDP_SDC.EXT.1				X				
FDP_CRR_EXT.1				X	X			
FDP_DAU.2						X		
FIA_ATD.1	X	X	X					
FIA_UAU.2	X							
FIA_UID.2	X							
FIA_USB.1	X	X	X					
FMT_SMR.1		X						
FMT_MTD.1		X		X	X			
FMT_SMF.1		X		X	X			
FPT_FLS.1							X	
FRU_FLT.1							X	
FTP_TRP.1	X							X

Table 18 – Mapping of TOE SFRs to Security Objectives

The following table provides detailed evidence of coverage for each security objective:

OBJECTIVE	REQUIREMENTS THAT ADDRESS THE OBJECTIVE	SFR AND RATIONALE
O.IDAUTH	FIA_UAU.2 FIA_UID.2 FIA_ATD.1 FIA_USB.1 FTP_TRP.1	Before the administrator can access the TOE management functions they must be identified and authenticated (verified by the underlying operating system, i.e. using Active Directory) and validated against a TOE role.
O.ROLES	FIA_ATD.1 FIA_USB.1 FMT_SMR.1 FMT_MTD.1 FMT_SMF.1	The administrators of the TOE must be associated with a TOE role which defines the management functions the administrator is able to access.
O.ACCOUNT	FIA_ATD.1 FIA_USB.1 FAU_GEN.1	Use of the management functions must be audited to ensure the administrators are accountable for their actions.
O.INFO	FMT_MTD.1 FMT_SMF.1 FDP_SDC.EXT.1 FDP_CRR_EXT.1	The TOE must provide the administrators with the ability to register new clients on the network and to configure settings for the network clients. The TOE must provide the administrators with the ability to configure sensors and questions, and to distribute these around all registered clients on the network to determine the configuration of the network clients.
O.ACTION	FMT_MTD.1 FMT_SMF.1 FDP_SAR.EXT.1 FDP_CRR_EXT.1	The TOE must provide the administrators with the ability to configure actions, and to distribute these around all registered clients on the network to execute the actions on the network clients to update their configuration.
O.GOOD_ORIGIN	FDP_DAU.2 FCS_COP.1(1) DSA FCS_COP.1(2) SHA FCS_CKM.1 FCS_CKM.4	The TOE must sign messages sent from the server to the clients.
O.CONTACT	FPT_FLS.1 FRU_FLT.1	The TOE must ensure that if a network client becomes unavailable, the peers of that client must still be able to communicate with other clients in the network to ensure all messages are distributed within the network.
O.PROTECT	FCS_COP.1(2) SHA FCS_COP.1(3) AES FCS_COP.1(4) RSA FCS_CKM.1 FCS_CKM.4	The TOE must encrypt all communication between the TOE and the administrator (TOE console) to ensure confidentiality and integrity of the configuration data is maintained during transmission.

OBJECTIVE	REQUIREMENTS THAT ADDRESS THE OBJECTIVE	SFR AND RATIONALE
	FTP_TRP.1	

Table 19 – Rationale for Mapping of TOE SFRs to Objectives

6.4.2 Rationale for TOE Security assurance requirements

The general level of assurance for the TOE (EAL2 plus ALC_FLR.2) is consistent with current best commercial practice for IT development and provides a product that is competitive against non-evaluated products with respect to functionality, performance, cost, and time-to-market. The TOE assurance also meets current constraints on widespread acceptance, by expressing its claims against EAL2 augmented by ALC_FLR.2 from part 3 of the Common Criteria.

7 TOE Summary Specification

7.1 Management (SF.MGMT)

FMT_SMR.1, FMT_MTD.1, FMT_SMF.1

The TOE's Management Security Function provides functionality that enables an authorized administrator to configure and manage TOE components via the (Adobe Flash) Tanium Console. Authorized administrators can perform different actions according to the user types to which they are assigned.

The actions that each administrator role can perform are specified in Table 15.

7.2 Identification & authentication (SF.I&A)

FIA_UAU.2, FIA_UID.2, FIA_ATD.1, FIA_USB.1, FPT_TRP.1

Windows Server credentials are used to validate a Windows user is an authorized administrator of the Tanium Console and to determine the permission set (user type) the administrator is associated with. The TOE makes a call to Windows Active Directory to ensure the current OS user is identified and authenticated to a Windows role that is bound to a TOE authorized administrator user type. The TOE binds security attributes to subjects for Tanium Console sessions. Windows binds security attributes to subjects for workstation sessions. Permissions are determined by the union of all permissions in any permission set associated with an administrator. If the security attributes for an administrator are changed during an active session, the new security attributes are not bound to a session until the next login.

7.3 User data protection (SF.INFO)

FDP_SAR_EXT.1, FDP_SDC_EXT.1, FDP_CRR_EXT.1, FDP_DAU.2

7.3.1 Overview

The TOE provides the ability to register new network clients to be managed. The new client connects to the server (directly) to provide status statistics. The server then determines the peers in close (network) proximity to the client and responds with the locations of a number (10 by default) of peers to establish the client in the peer-to-peer network. The server also responds with the global configuration state and any necessary configuration information (these are digitally signed, see section 7.6).

Once the client has been established in the peer-to-peer network, it will receive all questions and actions distributed to the network from the server. The ring peer-to-peer network is broken into segments of approximately 100 workstations to provide a way for the server to contact all clients quickly (within a few seconds).

Administrators can then collect information about the clients using Questions and providing updates to client configurations using Actions. When a peer receives a Question it processes the Sensors

requested in the Question and adds its response to the Answer section of the message. If the same response has already been provided by an earlier client it will simply increment the count for that response. The client encapsulates the response in a hash value/string pairing and sends it to the Server.

If an action includes a software update, the update file is split into "shards"; small packets of maximum 64Kb. The shards are distributed through the ring and are cached throughout the ring for later retrieval and re-constitution by new clients.

7.3.2 Component Interactions

There are five major types of communication within Tanium:

- Registration
- Questions
- Actions
- Sensors
- Settings

7.3.2.1 Registration

The purpose of the registration process is to allow the Server to keep a record of which Clients exist in the environment, and where they are in relation to the other Clients that are currently online. It also serves as a way to "seed" the peer-to-peer network with new commands, such as the Questions, Actions, Settings, and Sensors that will be described in the following sections.

The registration interaction is initiated by the Clients on a schedule that can be configured using the Settings capability described later. The interaction is lightweight, and involves the Client connecting to the Server, and providing a number of status statistics (e.g. whether it has had trouble connecting to its peers, how many peers it has talked to recently, etc.), and a set of values that represent all of the information that the Client currently knows about the global configuration state (i.e. which Sensors have been defined, all Settings values, which Questions the Client has answered, and which Actions have been executed). The Server responds with the locations of a number of Clients that are near to the Client to help with the establishment of the peer-to-peer network. The Server also provides a confirmation that the global configuration state known by the Client is up-to-date, or that it is not. If it is not, the Server and Client determine which pieces of configuration information are out of date, and update the Client. All settings information provided to the Client is delivered digitally signed by the Server, and is validated by the Client using the public key installed with it.

All registration interactions occur over a single configurable port (by default 17472), and are initiated by the Client to the Server.

7.3.2.2 Questions

Administrators who wish to collect information about Clients in the environment can do so using the Question message. Upon creation, a Question is initially recorded into the SQL database on the Server, where it is noticed by the Tanium Server service. When the Tanium Server service determines that a new Question has been asked, it queues that Question for Clients that subsequently register.

When the next Client registers, and submits its hash values to determine whether it has seen all Questions in the network, the Server will respond that there is a new Question, and provides the Client with the definition of that Question.

The actual format of the Question message is divided into two parts. First, there is the Question definition, which provides the Clients all the information necessary to determine what is being asked. Questions like “Computer Names and IP Addresses of computers with the Running Application is Firefox.exe and the Region is Europe” can be encoded to allow the Clients to determine what pieces of data are being requested, and what Clients should be supplying them.

After the Question definition, there is the Answer section of the message. This is blank when the Question is delivered to the Client upon the registration.

The Client will process the Question definition locally to determine if should provide one or more answers based on the Question’s targeting filter. In the Question above, the Client would determine whether it matched the filter “the Running Application is Firefox.exe and the Region is EMEA”. If the Client determines that it matches the targeting, it calculates its Answer based on the Sensors requested in the Question definition. The Client will then add its Answer to the Answer section. Once it has completed that process, it will forward the message along to its nearest peer.

The next Client will examine the Question in the same way. After processing the Question definition, it will submit its answer in the Answer section as well. Note that if the Answer is the same as an answer that has been provided by another Client in the Answer section, the Client answering can simply increment a counter on that answer, allowing for efficient storage of the answers.

This process continues until either the Answer section is “full” (by default, 1000 unique answers have been provided), a certain number of Clients have touched the message (by default 100), or a Client who has already seen the Question and has already answered it is reached. In any of those cases, the Question and its Answers are reported to the Server, where the Answers are submitted into the database for viewing on the Console.

When subsequent Clients register with the Server, they will receive the Question if they have not yet seen it, based on the hash interaction described above. However, if they have already seen the Question, they will be notified that their state is up-to-date, and the Question will not be delivered to them.

Note that the Peer-to-Peer process occurs over a single configurable port (by default 17472), and the reporting of the “full” report is done from the last Client to the Server over a configurable port (by default 17472).

7.3.2.3 Actions

In addition to being able to collect data from the entire environment using Questions, Tanium also provides the ability to take action, or make modifications. The Tanium solution provides the ability to deploy Packages that consist of a command line call and may include a set of files needed by the configured command line call. For example, if an administrator needed to uninstall an application, patch a third-party application, or make changes to a Windows registry key, they can create a Package, upload any needed files by the Action, and deploy it to any or all machines in the environment.

Package files are distributed and cached in pieces, or "shards," which allows the Tanium system to distribute Package files efficiently throughout the "ring" of peer Clients. When a Client begins downloading a file, completed shards are immediately passed along to its peer. Also, as each Client receives shards, it will select random shards and cache them locally in a self-cleaning, size-limited cache so that there is a high statistical likelihood that any Package file can be reconstituted from cached shards found in the peer ring. Subsequently, when a Client sees a new Action that requires files that are cached within the peer ring, it simply requests the shards needed to reconstitute the files from its peers without having to download the file or files again from the Server. Any shards that are not already available in the peer ring will be requested once from the Server, and re-cached in the ring at that time.

Each Action and Package is signed by the Server using a FIPS 140-2 certified 512-bit ECC cryptography algorithm, and validated by the Client prior to any Action execution. When a Client receives an Action or Package from the Server, it is validated to ensure authenticity using the public key that was installed with the Client upon initial installation, which matches the private key resident on the Server.

Actions can be executed immediately or at a specified time, and can be set to recur at a specified frequency or when a certain policy is triggered.

7.3.2.4 Sensors

Sensors are the mechanism by which Tanium Clients determine the local values that they have, in order to process the Questions and Actions that reference them. Sensors come with a name, an evaluation method, a script, and a maximum evaluation age that determines how long a sensor's result is valid before needing to be re-evaluated.

Sensors may be defined with a variety of evaluation methods including WMI, VBScript, PowerShell, and Bash Shell. Sensors are defined for a variety of hardware, software, networks, environments, operating systems, and other characteristics of the computer. By default, hundreds of Sensors are supplied with the product, and thousands of Sensors are available from the Tanium community. In addition, the system easily allows for the creation of new Sensors or modification of existing Sensors by administrators with appropriate rights in the enterprise.

The Sensor messages, similar to the Settings and Actions messages, are digitally signed by the Server, and validated locally before acceptance by Clients. Similar to the other signed messages, after the

first Client downloads it, the Client validates the Sensor message, absorbs the new or changed Sensor into its Sensor cache, and then forwards it around the ring until it reaches a Client that has already seen the Sensor message, at which point that Client kills the message.

7.3.2.5 Settings

The Tanium system has a number of settings that control the behavior of components in the network. Examples include bandwidth caps on the Clients, timing on retries for different types of messages, and other "configurable" options. Tanium enables administrators to change these settings through the peer-to-peer environment.

When a setting is first changed, the change is made in the database, and is pushed out to the Clients as part of the registration process, signed with the ECC key described above. The first Client that sees the Setting will validate the signature on the Setting message, and if it is valid, will proceed to change its local settings accordingly. It will then forward the setting on to the next Client in the peer-to-peer ring, who will do the same, and so on. The Setting message will be killed when it reaches a Client that has already seen that Setting update.

7.3.3 Other Message Types

Tanium communication is entirely message based. In addition to the messages that drive the components of the system mentioned above, there are additional message types that are generated, as described below.

7.3.3.1 Strings

When Clients provide Answers to Questions, and when the Server expresses Filters or other Sensor properties to the Clients, they do so using hashes in order to ensure that those Question or Answer contributions are fixed and reasonable length. When a Client sees a new Sensor result that needs to be displayed in a human-readable format (for example, in the Tanium Console), the Client generates a new String message that encapsulates the hash value/string pairing and sends it to the Server.

String Messages contain the Sensor that generated the string, the hash of the string, and the string itself. String Messages are sent by the Client that generated them to their forward peers to ensure that redundant string/hash pairs are not supplied to the Server. If enough Clients (default 10 Clients ahead of the Client that originally generated the String Message) determine that they have not seen the string/hash pairing being reported, the String Message is sent on to the Server. If any Client recognizes that the string/hash pairing in the String Message has already been reported, the message is killed and not reported to the Server.

7.3.3.2 String Retries

Similar to String Messages (but less frequently needed), String Retries allow the Server to gather Strings that Clients believe it has already received, but which it does not have record of receiving. If the Server is trying to correlate a hash value to its string but the hash doesn't have a corresponding string in the Server's database, the Server will send a String Retry message to the Clients. Each Client

will examine their own String State table which contains the string/hash values the Client has seen. If a Client can match the hash to a string, it will send the string up to the Server.

7.3.3.3 Keep Alives

Keep Alive messages are sent periodically on connections that are maintained for long periods of time, and are short messages whose primary purpose is to ensure that established connections have not been dropped by one side or the other. Keep Alive messages are sent on a configurable interval, which defaults to once every three seconds. Note that both Client-to-Client and Server-to-Client connections can have Keep Alive messages fed through them if the components believe that the connection should not be transient.

7.4 Protection of the TSF (SF.PTSF)

FPT_FLS.1, FDP_DAU.2

Each Action and package is digitally signed by the server, as detailed in section 7.6.

If a client fails to communicate with the previous or next peer in the ring the client will then contact the next closest peer until a response is received. If all close peers are exhausted without receiving a response, the client will assume it is now operating as a ring of 1, e.g. as a VPN peer, and will communicate directly with the server. In this way, even if a client is removed from the ring it can still connect to the server to receive questions and actions, and to send information. Also, with the client removed the ring can still operate by its peers connecting to each other to re-form the ring.

7.5 Resource utilisation (SF.RU)

FRU_FLT.1

If a client fails to communicate with the previous or next peer in the ring the client will then contact the next closest peer until a response is received. If all close peers are exhausted without receiving a response, the client will assume it is now operating as a ring of 1, e.g. as a VPN peer, and will communicate directly with the server. In this way, even if a client is removed from the ring it can still connect to the server to receive questions and actions, and to send information. Also, with the client removed the ring can still operate by its peers connecting to each other to re-form the ring.

7.6 Cryptographic support (SF.CRYPTO)

FCS_CKM.1, FCS_CKM.4, FCS_COP.1(all), FDP_DAU.2, FTP_TRP.1

The TOE provides the following cryptographic services to protect the transfer of messages between the server and the client, and between the server and administrator:

a) TLS connection to the management console using AES 128/256 bit encryption. The keys are generated using NIST SP 800-90 DRBG FIPS approved Random Number Generator, and deleted by overwriting when no longer required;

b) NIST curve" P-521 digital signing and verification of messages sent from the server to the client;

c) SHA-2 hash generation in support of digital signature services.

All cryptographic services are provided by the Tanium Cryptographic Module (TaniumCryptoLibrary.dll) which is FIPS 140 validated [CMVP Certificate #2736]. This cryptographic module is used in both the Tanium Server and Client.

Keys are deleted by overwriting when no longer required using the cryptographic module.

The server key pair used for digital signature is persistent by default. Key rotation can be enabled to allow the enterprise to prevent brute-force cracking of the keys. The frequency of the key rotation is a setting controlled by the enterprise.

7.7 Security Audit (SF.AUDIT)

FAU_GEN.1, FIA_USB.1

The TOE generates audit records for the following events:

- Start-up and shutdown of the audit functions;
- All actions taken by the administrator;
- Distribution of a query;
- Distribution of an action;
- Distribution of a registration request;
- Distribution of a settings message.

For each event the following detail is recorded:

- Date and time of the event;
- Type of event (e.g. administrative action type or distribution of query/action/registration request/settings message);
- Subject identity (for all actions taken by the administrator);
- Outcome (success or failure) of the event.