SYSTEM CONSULTANCY SERVICES SDN BHD
36, Jalan Wangsa Delima 6
Pusat Bandar Wangsa Maju (KLSC)
53300 Kuala Lumpur

OpenSSL 1.1.1g-NC2.vpn+ v2.1.9 ISO/IEC 19790 Level 1 Non-Proprietary Security Policy

Version 1.2.0

1 Contents

List of Table

List of Figure

Change History Issue

No	Related Pages/ Paragraph	Date	Comment
1.0.0	All	18 April 2022	Final Release. This document is derived from NC2.vpn_plus Security Policy v1.0.3. The name of the document is changed due to the change of the Module name. Additional changes are based on the findings of the testing laboratory during the testing and validation of the Module. These include the refinement, addition, and removal of the paragraphs and figures where it is out of the validation boundary.
1.1.0	Section 1.1	17 June 2022	Update the references current version and date.
	Table 2		Remove the calling application from the list as this is not tested.
	Table 6		Add the ‘Configuration’ service and rearrange the table to make it consistent with Table 9.
	Paragraph 33 a)		Change the path or location of the files from “/usr/local/opensense/lib” to “/usr/local/lib” and “/usr/local/bin” to “/usr/local/bin/seltest”.
	Table 7		Update the status output based on the Table 9 and APD document.
	Table 9		Add the details of the ‘Configuration’ service. Update the control input, control output, and status output of the ‘Show Module version and status” service. Update the SSP of the “Symmetric encrypt/ decrypt and zeroize of the symmetric SSP”

			Update the “Hash “Compare” control output and status output. Add “Self-tests (integrity test and conditional self-test)” that details the integrity test and conditional self-test services.
1.2.0	Section 1.1	26 July 2022	Update the references current version and date.
	Figure 1		Update the boundary of the Module validation which include the **openssInc2**.
	Paragraph 12		Add new sub paragraph 12e) which explain the **openssInc2**.
	Paragraph 32		Add new sub paragraph 32a)iv) as **openssInc2** command is required to provide show Module version and status service.
	Table 7		Update the status output based on the update in Table 9 and APD document.
	Table 9		Update the data input, control input, and status output of the “Self-tests (integrity test and conditional self-test)” services consistent with the changes in APD document.

Reference and Document

Reference	Document	Comment

List of Abbreviation and Definition used

Abbreviation	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher Block Chaining

Abbreviation	Definition
CFB	Cipher Feedback
CO	Crypto Officer
CSP	Critical Security Parameter. Security related information whose disclosure or modification can compromise the security of a cryptographic module.
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
FSM	Finite State Model
HTTPS	Hypertext Transfer Protocol Secure
IV	Initialization Vector
KAT	Known Answer Test
MyCV	Malaysian Cryptographic Validation Scheme
MySEAL	*Senarai Algoritma Kriptografi Terpercaya Negara*
OFB	Output Feedback
OpenSSL	The open-source cryptographic library implementation
OS	Operating System
PSP	Public security parameter. Security related public information whose modification can compromise the security of a cryptographic module.
RAM	Random Access Memory
RSA	Rivest, Shamir and Adleman
SCS	System Consultancy Services Sdn Bhd
SHA	Secure Hash Algorithm
SSL	Secure Socket Layer transport layer protocol
SSP	Sensitive Security Parameters. Critical security parameters (CSP) and public security parameters (PSP).
System Administrator	Users that are allowed to perform both configuration and monitoring of the Module.
TLS	Transport Layer Security

Abbreviation	Definition
Zeroisation	Method of destruction of stored data and unprotected SSPs to prevent retrieval and reuse

1. ABOUT THIS DOCUMENT

1 This document is a non-proprietary, ISO/IEC 19790 Level 1 Security Policy for System Consultancy Services (SCS) Sdn Bhd's OpenSSL 1.1.1g-NC2.vpn+ v2.1.9 (hereafter referred to as the 'Module'). This policy outlines the functionality provided by the Module, gives high-level details on the means by which the Module meet the ISO/IEC 19790 Level 1 security requirements, and how to operate the module in ISO/IEC 19790 compliant manner.

2 This Security Policy deals specifically with operation and implementation of the Module based on the requirements of the ISO/IEC 19790 standard and the Malaysian Cryptography Validation (MyCV) Scheme.

1.1. REFERENCES

[1] OpenSSL 1.1.1g-NC2.vpn+ v2.1.9 Finite State Model, v1.2.0, 26 July 2022

[2] OpenSSL 1.1.1g-NC2.vpn+ v2.1.9 Functional Specification, v1.1.1, 26 July 2022

[3] OpenSSL 1.1.1g-NC2.vpn+ v2.1.9 Life Cycle Assurance, v1.1.1, 26 July 2022

[4] OpenSSL 1.1.1g-NC2.vpn+ OpenSSL v2.1.9 Algorithm Parameter Definition, v1.2.0, 26 July 2022

[5] NC2.vpn+ Administration Guide, v1.0.4, 28 June 2021

[6] OpenSSL 1.1.1g-NC2.vpn+ v2.1.9 Vendor Testing, v1.0.0, 11 April 2022

1.2. DOCUMENT ORGANISATION

3 This document is organised into the following major sections:

a) Section 1 provides the introductory material for the security policy document as well as the brief description of the module.

b) Section 2 provides the security requirements detail based on the ISO/IEC 19790 security requirements. This section is breaks into several sub-sections as follows:

i) Security Level Summary

ii) Cryptographic Module Specification

iii) Cryptographic Module Interfaces

iv) Roles, Services and Authentication

v) Software/ Firmware Security

vi) Operational Environment

vii) Physical Security

viii) Non-invasive Security

ix) Sensitive Security Parameters Management

x) Self-tests

xi) Life-cycle Assurance

xii) Mitigation of Other Attacks

2. SECURITY REQUIREMENTS

2.1. Security Level Summary

4 The Module meets the ISO/IEC 19790 Level 1 validation as shown in **Table 1**.

Table 1: Summary of ISO/IEC 19790 security requirements and compliance levels

Security Requirement	Compliance Level
Cryptographic Module Specification	1
Cryptographic Module Interfaces	1
Roles, Services and Authentication	1
Software/Firmware Security	1
Operational Environment	1
Physical Security	N/A
Non-invasive Security	N/A
Sensitive Security Parameter Management	1
Self-tests	1
Life-cycle Assurance: – Configuration Management – Design – Finite State Model (FSM) – Development – Testing – Delivery and Operation – Guidance	1
Mitigation of Other Attacks	N/A

2.2. Cryptographic Module Specification

2.2.1. Module Type

5 The Module is a software cryptographic module whose purpose is to provide the cryptographic services to applications running in the user space of the underlying HardenedBSD operating system through C Application Program Interface (API). The application (NC2.vpn+) is use to provide a secure tunnel, that is encrypted, between two points in a network. This is to ensure that any data sent over the internet is encrypted and private.

6 The Module was tested and can be run on the system requirements as presented in **Table 2** below.

Table 2: Tested Platform

Appliance	
Software (the Module)	OpenSSL 1.1.1g-NC2.vpn+ v2.1.9
Hardware	ABP-3000-8665U16
Operating System	FreeBSD v13.0 (HardenedBSD)
Web-based GUI User	
Web Browser	Microsoft Edge 44 Mozilla Firefox 64 Google Chrome 71

2.2.2. Cryptographic Boundary

7 Since the Module is defined as a software cryptographic module, it possesses both a physical cryptographic boundary and logical cryptographic boundary.

2.2.2.1. Physical Cryptographic Boundary

8 As a software module, the Module must rely on the physical characteristic of the computing platform. The physical embodiment of the computing platform on which it runs is a multi-chip standalone module.

9 The physical boundary of the Module is defined by the hard enclosure around the computing platform on which it runs. The computing platform consists of the integrated circuits of the system board, processor, RAM, hard disk, device casing, power supply, and fans. **Table 3** identifies the minimum hardware specification of the computing platform.

Table 3: Minimum Hardware Specification of the Computing Platform

Component	Specification
Processor	Intel based processor
Memory (RAM)	16GB
Storage	128GB storage; preferred industrial SSD type of storage

10 The computing platform and operating system of the operational environment which the software executes in are external to the defined software module boundary.

2.2.2.2. Logical Cryptographic Boundary

11 **Figure 1** illustrates the physical and logical cryptographic boundary of the Module where it shows a logical boundary of the Module (which is highlighted in red) and its relation to the hardware and operating system (physical boundary).

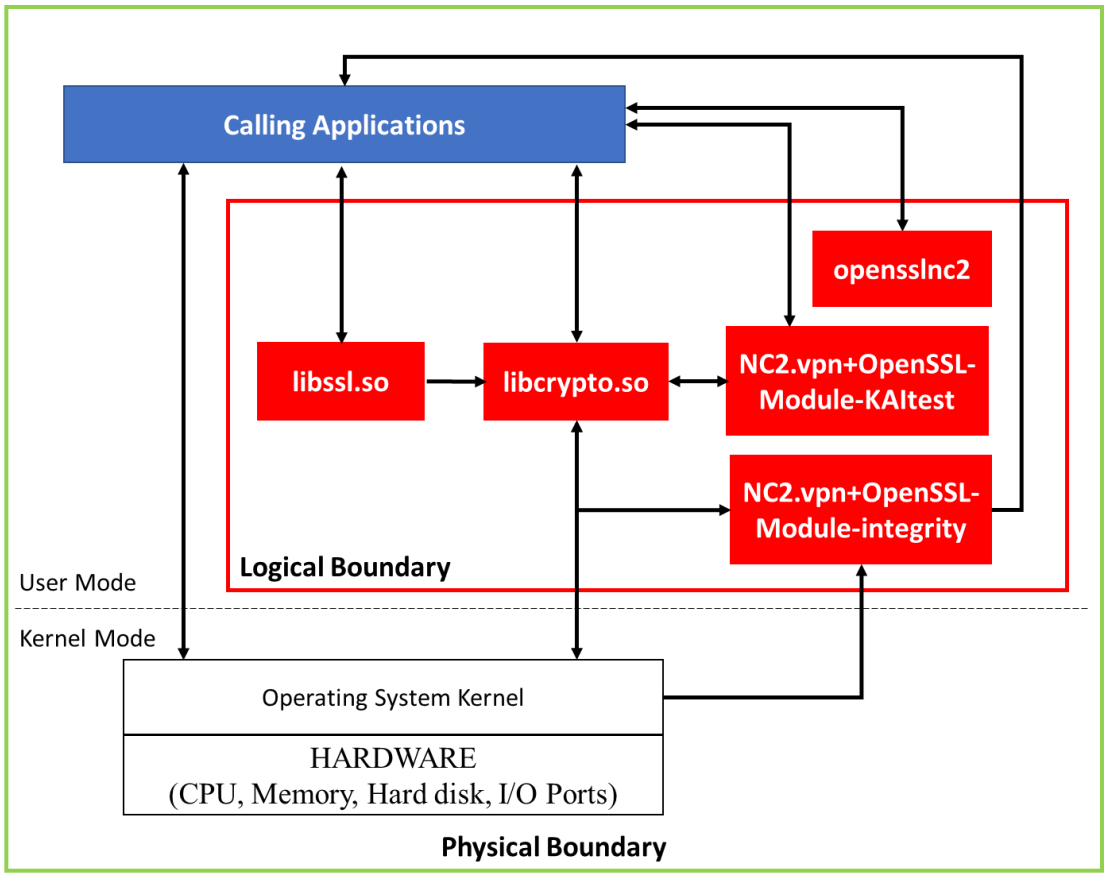

Figure 1: Module's Physical and Logical Cryptographic Boundary

12 The logical boundary of the Module consists of:

a) **NC2.vpn+OpenSSL-Module-integrity**. This function is responsible for performing of the integrity test of the Module, and determination of pass or fail. If failed, the Module shall enter an error state and shall not run the NC2.vpn+ (calling application) until the integrity test has been repeated and successfully passed. NC2.vpn+OpenSSL-Module-integrity will be called after the operating system had been loaded successfully.

b) **NC2.vpn+OpenSSL-Module-KAItest (Kripto Algorithm Integrity)**. This function is responsible for performing of the self-tests of the Module, and determination of pass or fail. If failed, the Module shall enter an error state and the VPN function will not be running until the relevant self-test has been repeated and successfully passed.

NC2.vpn+OpenSSL-Module-KAItest will be called by the calling application prior to the first operational use of the cryptographic algorithm. This is important to ensure that the cryptographic functions will be performed correctly. NC2.vpn+OpenSSL-Module-KAItest will also be called by libcrypto.so periodically during the operation of the VPN to ensure that the Module is performing correctly.

c) **libcrypto.so**. This is the core library for providing implementations of numerous cryptographic primitives. In addition, it provides a set of supporting services which are used by libssl.so, as well as implementations of protocols such as CMS (Cryptographic Message Syntax) and Online Certificate Status Protocol (OCSP).

d) **libssl.so**. This library depends on libcrypto.so and implements the TLS (Transport Layer Security) and DTLS (Datagram Transport Layer Security) protocols.

e) **openssInc2**. This command is use to show the Module status information such as the version, and operational mode of the Module. The command can be executed when the root privilege is enabled especially after the installation where the Module status information needs to be verified.

13 The Module provides cryptographic API for invocation of approved cryptographic functions from external applications.

2.2.3. Modes of Operations

14 This version of the Module supports MySEAL listed algorithm only. **Table 4** includes all MySEAL listed algorithms where the Module will be operating in an approved mode of operation once they are invoked by the calling application or function.

NOTE: An approved mode of operation shall be defined as the set of services which include at least one service that utilises a MySEAL listed algorithm.

15 The Module must be configured as described in Section 2.6.

Table 4: MySEAL Listed Algorithm

Cryptographic Primitives	Algorithm	Variant	Key Strength	Function	MyCV CAV Cert#
Symmetric Block Cipher	AES	Modes of operations: CBC, CFB, OFB Key sizes: 128, 192 and 256bit	Depend on the length of the cryptography key use	Encryption/ Decryption operations	MyCV-CAV001
Symmetric Block Cipher	CAMELLIA	Modes of operations: CBC, CFB, OFB Key sizes: 128, 192 and 256-bit	Depend on the length of the cryptography key use	Encryption/ Decryption operations	MyCV-CAV001

Asymmetric Cryptographic	DH	Modulus size: 2048	112	Key agreement	MyCV-CAV002
Cryptographic Hash Function	SHA2	Digest sizes: 512 bits	256	Hashing	MyCV-CAV001
Asymmetric Cryptographic	RSA-PSS 2048	Modulus size: 2048 SHA algorithm: SHA512	112	Digital signature (signing and verify operations)	MyCV-CAV002
Random Bit Generator	CTR DRBG	Modes: AES-128 and AES-192	Depend on the length of the cryptography key use	Random number generation	MyCV-CAV002

16 The RSA implementation includes known answer tests at start-up which consist of a sign and verify operation which is also a sign and verify operation when keys are generated.

17 The Module is not designed to support degraded functionality if the Module enters the error state. The VPN function will be shut down when the Module enters the error state. The VPN will be operated normally once the Module passes all the self-tests.

2.2.3.1. Approved Mode of Operation

18 The Module performs the following services, operations, or functions to operators:

a) The Module shall output the Module's name and the versioning information.

b) The Module shall output the current status. This may include the output of status indicators in response to a service request.

c) The Module shall initiate and run the pre-operational and conditional self-tests.

d) The Module shall use the MySEAL Listed Algorithm in the approved mode of operation.

e) The Module shall perform zeroization of the parameters (public/private keys, password etc.).

19 The Module utilises only MySEAL listed algorithm as specified in **Table 4**.

2.3. Cryptographic Module Interfaces

20 The Module's logical interfaces exist at a low level in the software as API functions.

21 As a software module, the Module does not have physical ports. Thus, the physical ports within the physical boundary are interpreted to be the physical ports of the hardware platform on which the Module runs and are directed through the interfaces provided by the Module. Control of the physical ports is outside of the Module scope; however, when the Module is performing self-tests, or is in an error state, all output on the logical data output interface is inhibited. The Module in error state returns only an error message that is the status output.

22 **Table 5** summarises the mapping of the Module's logical interfaces.

Table 5: Module's Logical Interfaces Mapping

Interface	Logical Interface
Data Input	The function calls or API that accept data input stack parameters for processing through their arguments.
Data Output	The function calls or API that return the data output stack parameters.
Control Input	The function calls or API that are used to control the operation of the Module
Control Output	The control commands that are used to control or indicate the state of operation of the Module.
Status Output	Return values for function calls or API; Module generated messages including successful and fail messages

2.4. Roles, Services, and Authentication

2.4.1. Roles

23 The Module implements Crypto Officer (CO) role only. This role is implicitly entered when performing system administration functions on the Module.

2.4.2. Authorised Services

24 All services implemented by the Module are listed in **Table 6** below, along with a description of services SSP access.

Table 6: Module's Services and SSP Access

Services	Role	Description
Configuration	CO	View/set/edit/delete Module configuration. Does not access SSPs.
Show Module version and status	CO	Function that provides Module status information such as version and the operation mode of the Module. Does not access SSPs.
Zeroize	CO	Functions that destroy SSPs. Automatically overwrite SSPs stored in allocated memory. All data output via the data output interface shall be inhibited while performing zeroization.
Symmetric encrypt/decrypt	CO	Used to encrypt or decrypt data. Executes using AES and CAMELLIA.
Hash	CO	Used to generate or verify data integrity with SHA512 (passed in by the calling API). Does not access SSPs.
Digital signature	CO	Used to generate or verify RSA-PSS digital signatures (passed in by the calling API).

Key agreement	CO	Used to perform key agreement primitives on behalf of the calling API (does not establish keys into the module).
Random number generation	CO	Used for random number and symmetric key generation.
Self-test	CO	Perform integrity test and conditional self-test. Does not access SSPs.

25 The Module does not implement a bypass capability.

NOTE: Bypass capability is the ability of a service to partially or wholly circumvent a cryptographic function or process.

26 The Module does not have self-initiated cryptographic output capability.

2.4.3. Authentication

27 The CO will not need to be authenticated to access the Module but the CO need to be authenticated to access the calling application (NC2.vpn+) via the web-based GUI interface over HTTPS (TLS) before allowing any actions by the CO.

2.5. Software/Firmware Security

28 The Module is implemented completely in hardware as specified in **Table 3**.

29 The Module integrity check will be done by the Module upon system start-up. Hash computation of the Module will be conducted and compared with the previous record. If the Module had been tampered, the hash value will not be the same and the integrity test will fail. The Module will enter the Error State if the integrity test fail (refer to OpenSSL 1.1.1g-NC2.vpn+ v2.19 Finite State Model document (Ref [1])). At this stage, the VPN function which is utilising the Module for cryptographic function will be disabled until the Module exit the Error State.

2.6. Operational Environment

30 The Module, which is a software module, rely on the computing platform and operating system on which it runs.

31 The Module operates in a modifiable operational environment as per ISO/IEC 19790:2012 Security Level 1 specifications.

32 The initial configuration of the computing platform will be performed once by SCS authorised personnel before the hard drive image being produce as an installer. The following shall be performed with the root privilege:

a) Copy the files to the following locations:

   i) libssl.so.11 to “/usr/local/lib/libssl.so.11”

   ii) libcrypto.so.11 to “/usr/local/lib/libcrypto.so.11 “

iii) NC2.vpn+OpenSSL-Module-integrity and NC2.vpn+OpenSSL-Module-KAItest to "/usr/local/bin/selftest"

iv) opensslnc2 to "/usr/local/bin/"

b) Run script ***selftest/setup.sh*** to automatically do the followings:

i) copy startup files required for Module-integrity and Module-KAItest,

ii) restart configd service,

iii) make new directory "/var/log/nc2vpn+" for logging, and

iv) compute hash for libssl.so, libcrypto.so, NC2.vpn+OpenSSL-Module-integrity binary, and NC2.vpn+OpenSSL-Module-KAItest binary.

33 Once the steps in paragraph 32 successfully finished, the hard drive image will be produced. The installation of the NC2.vpn+, which include the Module, will be performed using the hard drive image and no further configuration is needed for the operational environment of the Module.

34 Once installed, the information below can be verified:

a) the Module version by typing command:

opensslnc2 -v or ***opensslnc2 --version***

and the status output shall be:

"OpenSSL 1.1.1g-NC2.vpn+ v2.1.9"

b) the operational mode of the Module by typing command:

opensslnc2 -s or ***opensslnc2 --status***

and the status output shall be:

"Operating in MYCV approved mode"

or

"WARNING! Not in MYCV approved mode", the Module is not operating in the MyCV validated module properly if this status output is shown.

35 Although the Module is operated in a modifiable operational environment, the Module will ensure that the cryptography operations will not be interfered or changed by other services available in NC2.vpn+ such as firewall, IDS and others. The Module and the data output generated by the Module are stored in the file system separated from other services provided by NC2.vpn+. Moreover, the root access had been disabled before it is delivered to the customer to prevent privileged control of NC2.vpn+. Management of the NC2.vpn+ can only be access from the graphic user interface (GUI) by the authorised administrator.

2.6.1. Handling of Integrity and Self-Test Errors

36 All integrity and self-test will produce a log at file location /var/log/nc2vpn+/selftest.log. Logfile format is [Time] [Log level] [Thread] [Message]. [Log level] values could be “info” or “error”. Any failed test will produce [Log level] “error”. Any successful test will produce [Log level] “info”.

37 The log file which can be viewed by the CO via the web-based GUI interface over HTTPS (TLS). The CO can see the log from the menu **VPN -> OpenVPN -> Self Test** as per **Figure 2**.

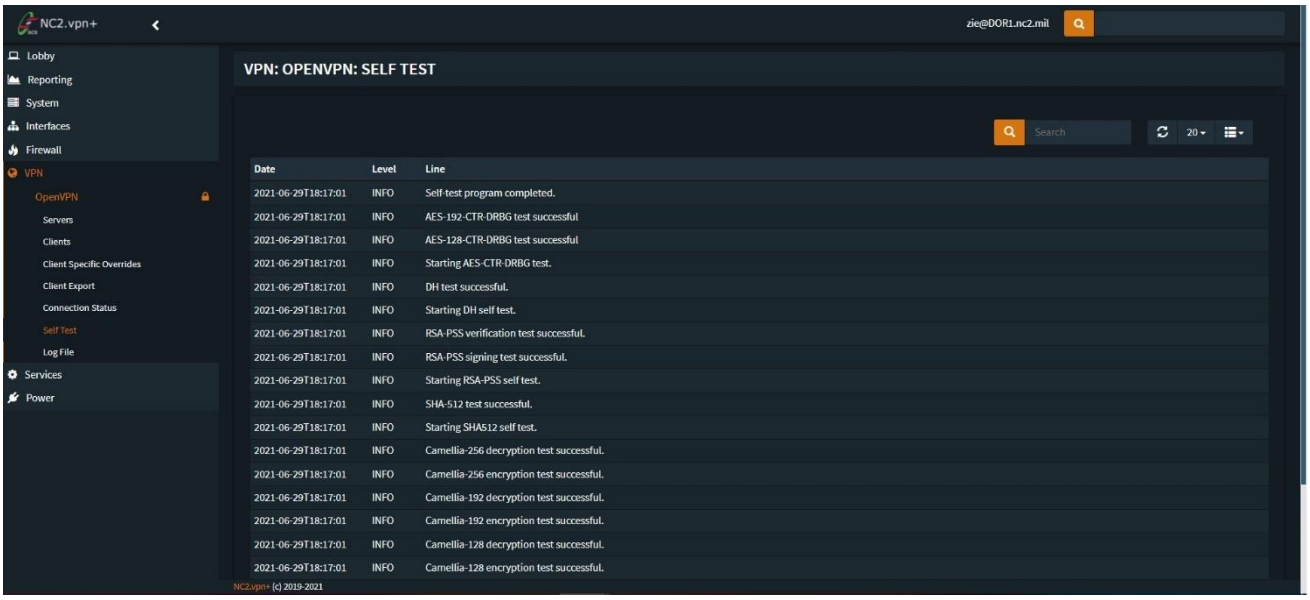

Figure 2: Self Test Log

38 The Module will log the messages in a log file either it is successful or fail. **Table 7** below summarises the testing indicators that are output via the Module’s Status Output interface.

Table 7: Self-Test Status Messages

Test	Status Output	
	Fail Message	**Successful Message**
Cryptographic algorithm self-test	Starting AES self test. AES-128 encryption test failed! AES-128 decryption test failed! AES-192 encryption test failed! AES-192 decryption test failed! AES-256 encryption test failed! AES-256 decryption test failed! Starting Camellia self test. Camellia-128 encryption test failed! Camellia-128 decryption test failed! Camellia-192 encryption test failed!	Starting AES self test. AES-128 encryption test successful. AES-128 decryption test successful. AES-192 encryption test successful. AES-192 decryption test successful. AES-256 encryption test successful. AES-256 decryption test successful. Starting Camellia self test. Camellia-128 encryption test successful. Camellia-128 decryption test successful. Camellia-192 encryption test successful.

	Camellia-192 decryption test failed! Camellia-256 encryption test failed! Camellia-256 decryption test failed! Starting SHA512 self test. SHA-512 test failed! Starting RSA-PSS self test. RSA-PSS test internal error code: {encryptioncode}. RSA-PSS signing test failed! RSA-PSS verification test failed! Starting DH self test. DH test failed! Starting AES-CTR-DRBG test. AES-128-CTR-DRBG test failed! AES-192-CTR-DRBG test failed! Self-test program completed. Shutting down VPN services!	Camellia-192 decryption test successful. Camellia-256 encryption test successful. Camellia-256 decryption test successful. Starting SHA512 self test. SHA-512 test successful. Starting RSA-PSS self test. RSA-PSS signing test successful. RSA-PSS verification test successful. Starting DH self test. DH test successful. Starting AES-CTR-DRBG test. AES-128-CTR-DRBG test successful AES-192-CTR-DRBG test successful Self-test program completed.
Software integrity check	File /usr/local/lib/libssl.so.11 does not exists! Shutting down VPN service. File /usr/local/lib/libcrypto.so.11 does not exists! Shutting down VPN service. File /usr/local/bin/selftest /NC2.vpn+OpenSSL-Module-integrity does not exists! Shutting down VPN service. File /usr/local/bin/selftest /NC2.vpn+OpenSSL-Module-KAItest does not exists! Shutting down VPN service. File /usr/local/bin/opensslnc2 does not exists! Shutting down VPN service. Hash record read error {0}: {1} Shutting down VPN service.	File opensslnc2 is intact. File libssl.so is intact. File libcrypto.so is intact. File Module-integrity is intact. File Module-KAItest is intact.

	Hash record of opensslnc2 not exists! Shutting down VPN service. Hash record libssl.so and libcrypto.so does not exists! Shutting down VPN service. Hash record of Module-integrity and Module-KAItest does not exists! Shutting down VPN service. Warning! File opensslnc2 integrity compromised. Shutting down VPN service. Warning! File libssl.so integrity compromised. Shutting down VPN service. Warning! File libcrypto.so integrity compromised. Shutting down VPN service. Warning! File Module-integrity integrity compromised. Shutting down VPN service. Warning! File Module-KAItest integrity compromised. Shutting down VPN service.	

39 If the Module is in error state, no output data exits the Module and no cryptographic services will be provided.

2.7. Physical Security

40 This requirement is not applicable to this type of module. Please refer to **Table 1**.

2.8. Non-invasive Security

41 This requirement is not applicable to due to no approved non-invasive attack mitigation test metrics defined in Annex F of the ISO/IEC 19790:2012. Please refer to **Table 1**.

2.9. Sensitive Security Parameters (SSPs) Management

42 The security requirements for SSPs management encompass the entire lifecycle of SSPs employed by the Module.

43 Sensitive Security Parameters (SSPs) consist of:

a) Critical Security Parameters (CSPs). Example of CSP includes secret and private cryptographic keys, authentication data such as passwords, PINs, certificates or other trust anchors. CSPs shall be protected from unauthorised access. NOTE: A CSP can be plaintext or encrypted.

b) Public Security Parameters (PSPs). Example of PSP includes public cryptographic keys, public key certificates, self-signed certificates, trust anchors, one-time passwords associated with a counter and internally held date and time. NOTE: A PSP is considered protected if it cannot be modified or if its modification can be determined by the module.

44 Since the Module is accessed via API functions call from a referencing application, the Module does not manage SSPs. In fact, for software, SSPs will be found in multiple locations external to the Module such as in application buffers, primary (RAM) memory, secondary disk storage, CPU registers, and on the system bus. In the case of networked client-server applications some SSPs will be found on both the client and server system and on the network infrastructure in between (Ethernet and WAN communication lines, routers, switches).

45 SCS (the Vendor) and the CO (operator/user) share a responsibility to ensure that the SSPs are always protected from unauthorised access. This protection will generally make use of the security features of the host computing platform and software which is outside of the cryptographic boundary defined for this Module.

46 **Table 9** in **Appendix A** list all the cryptographic keys and the SSPs used by the Module.

47 Key zeroization is performed by:

a) Shutting-down or rebooting the operating system or the host computer.

b) Reformatting and overwriting the mass storage device where the Module is installed on.

All data output via the data output interface shall be inhibited while performing zeroization.

2.10. Self-tests

48 The Module performs a number of pre-operational and conditional self-tests to ensure proper operation of the Module.

49 The failure of pre-operational and conditional self-tests causes the Module to enter Error State (see OpenSSL 1.1.1g-NC2.vpn+ v2.19 Finite State Model document (Ref [1]) for the details of the error log). All cryptographic operations are disabled until the Module successfully performs the self-tests. In practice, self-test failure means the application must exit and restarted. If the error persists, the Module must be reinstalled.

2.10.1. Pre-operational Self-test

50 Pre-operational self-test include software integrity test and run automatically upon system start-up or reboot by calling *NC2.vpn+OpenSSL-Module-integrity* function. The integrity test is performed using SHA512. Hash computation of the Module will be conducted and compared with the previous record. If the Module had been tampered, the hash value will not be the same and the integrity test will fail. The Module will enter the Error State if the integrity test fails. No cryptographic operation will be available until after successful execution of pre-operational self-test.

2.10.2. Conditional Self-test

51 In addition to the pre-operational self-test, the Module performs several conditional self-tests including Known Answer Tests (KATs) and periodic self-test.

NOTE KATs are tests where cryptographic value is calculated and compared with a stored previously determined answer.

52 Conditional self-tests are performed automatically and cannot be turned off.

53 **Table 8** list the algorithms that are tested in the conditional self-test.

Table 8: Conditional Self-tests

Algorithm	Conditional self-test
AES CBC	Encryption and decryption with key sizes 128, 192 and 256 bits
CAMELLIA CBC	Encryption and decryption with key sizes 128, 192 and 256 bits
SHA2	Message digest with SHA512 bits
RSA-PSS	Sign and verify test with RSA-PSS 2048 SHA512 bits
DH	Key agreement with 2048 bits
CTR-DRBG	Random number generation from known IV and encryption with AES key size 128 and 192 bits

2.11. Life-cycle Assurance

54 The Module and related documentations are managed in accordance with the established OpenSSL 1.1.1g-NC2.vpn+ v2.19 Life Cycle Assurance (Ref [3]).

55 Proper guidance for the system administrator to ensure secure management and operation of the Module is available in the NC2.vpn+ v2.19 Administration Guide (Ref [5]).

56 This is to give assurance that the Module is properly designed, developed, tested, configured, delivered, installed and disposed, and that the proper operator guidance documentation is provided.

2.12. Mitigation of Other Attacks

57 This requirement is not applicable to this Module because it does not implement counter measures towards special attacks. Please refer to **Table 1**.

Appendix A Module's Services and SSP Access

Table 9: Module's Services and SSP Access

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
Configuration – view/ set/ edit/ delete Module configuration	CO	N/A	N/A	Via NC2.vpn+ v2.1.9 management interface menu: VPN -> OpenVPN-> Server (https://[serverip:port number]/vpn_openvpn_server.php?act=edit&id=0)	N/A	N/A	Read Write Edit	N/A
Show Module version and status	CO	N/A	N/A	N/A	Command: opensslnc2 [-s status] [--status status] [-v version] [--version version] [-h] [--help] Example: opensslnc2 -v or opensslnc2 –version Example: opensslnc2 -s or	N/A	Read	"OpenSSL 1.1.1g-NC2.vpn+ v2.1.9" "Operating in MYCV approved mode" Or

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
					opensslnc2 –status Example: opensslnc2 -h or opensslnc2 –help			“WARNING! Not in MYCV approved mode” USAGE: opensslnc2 [-s status] [--status status] [-v version] [--version version] [-h] [--help]
Symmetric encrypt/ decrypt and zeroize of the symmetric SSP								
Encryption – Service context establishment	CO	N/A	Cipher Context	EVP_CIPHER_CTX *EVP_CIPHER_CTX_new(void)	N/A	N/A	N/A	N/A
Encryption – Initialise encryption context	CO	Cipher Context, Encryption method, Implementation, Encryption key, initialization vector	N/A	int EVP_EncryptInit_ex(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher, ENGINE *impl, const unsigned char *key, const unsigned char *iv)	N/A	Key (AES or CAMELLIA)	N/A	N/A
Encryption – Initialise decryption context	CO	Cipher Context, Encryption method,	N/A	int EVP_DecryptInit_ex(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher, ENGINE *impl,	N/A	Key (AES or CAMELLIA)	N/A	N/A

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
		Implementation, Encryption key, initialization vector		const unsigned char *key, const unsigned char *iv)				
Encryption – Delete context	CO	Cipher Context	N/A	void EVP_CIPHER_CTX_free(EVP_CIPHER_CTX *c)	N/A	N/A	Delete	N/A
Symmetric encryption	CO	Cipher Context, Plain text input, Length of plain text input	Cipher text result, Length of cipher text	int EVP_EncryptUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl, const unsigned char *in, int inl)	N/A	N/A	Read Write Execute	N/A
Symmetric decryption	CO	Cipher Context, Cipher text input, Length of cipher text input	Plain text result, Length of plain text	int EVP_DecryptUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl, const unsigned char *in, int inl)	N/A	N/A	Read Write Execute	N/A
Hash								

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
Hash – Hash context establishment	CO	N/A	Message Digest Context	EVP_MD_CTX *EVP_MD_CTX_new(void)	N/A	N/A	N/A	N/A
Hash – Initialise message digest context	CO	hash algorithm type, implementation	Message Digest Context	int EVP_DigestInit_ex(EVP_MD_CTX *ctx, const EVP_MD *type, ENGINE *impl)	N/A	N/A	N/A	N/A
Hash – add data to message digest context	CO	Data to be hashed, Size of “data” parameter in byte	Message Digest Context	int EVP_DigestUpdate(EVP_MD_CTX *ctx, const void *data, size_t count)	N/A	N/A	Read Write	N/A
Hash – compute message digest	CO	Message Digest Context	Message digest, size of message digest	int EVP_DigestFinal_ex(EVP_MD_CTX *ctx, unsigned char *md, unsigned int *size)	N/A	N/A	Read Write Execute	N/A
Hash – delete the message digest context	CO	Message Digest Context	N/A	void EVP_MD_CTX_free(EVP_MD_CTX *ctx)	N/A	N/A	Delete	N/A
Hash – Compare the hash value of 2 binaries sequences	CO	Binary sequence input 1,	0 if the two inputs match exactly,	int compare2bin(const unsigned char* hex1, const unsigned char* hex2, unsigned int len)	N/A	N/A	Read	N/A

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
Note: Module will enter self-tests state if the control output is 0 or error state if the control output is 1.		Binary sequence input 2, Length of hex1 and hex2 input	1 if binary sequences do not match					
Digital signature and zeroize of the SSP								
Digital signature – Fill EVP_PKEY structure with private key	CO	File pointer to RSA-PSS private key file, deprecated (**x argument), Call back function password for RSA-PSS secret key	EVP_PKEY structure	EVP_PKEY *PEM_read_PrivateKey (FILE *fp, EVP_PKEY **x, pem_password_cb *cb, void *u)	N/A	Password for RSA-PSS secret key (N/A if no password)	Read Write	N/A
Digital signature – Fill EVP_PKEY structure with public key	CO	File pointer to RSA-PSS public key file, deprecated (**x argument), Call back function,	EVP_PKEY structure	EVP_PKEY *PEM_read_PUBKEY(FILE *fp, EVP_PKEY **x, pem_password_cb *cb, void *u)	N/A	Password for RSA-PSS secret key (N/A if no password)	Read Write	N/A

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
		password for RSA-PSS secret key						
Digital signature – Get RSA key structure from EVP_PKEY	CO	EVP_PKEY structure that contains public or private key	RSA key structure that contains RSA-PSS public or private key	const RSA *EVP_PKEY_get0_RSA(const EVP_PKEY *pkey)	N/A	RSA-PSS private/public key	Read	N/A
Digital signature – apply PSS padding to hash input	CO	RSA private key structure, SHA512 hash of a file, Hash type, Salt length	Data output after applying padding to mHash,	int RSA_padding_add_PKCS1_PSS(RSA *rsa, unsigned char *EM, const unsigned char *mHash, const EVP_MD *Hash, int sLen)	N/A	N/A	Read Write	1 if successful, 0 if failed
Digital signature – RSA encrypt with private key (actual signing process)	CO	Length of data input *from in bytes (which is the length of RSA key in bytes), RSA-PSS private key, padding	PSS padded hash, Output of the signing process, Size of the signature data	int RSA_private_encrypt(int flen, unsigned char *from, unsigned char *to, RSA *rsa, int padding)	N/A	RSA-PSS private key	Read Write Execute	N/A

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
Digital signature – decrypt RSA-PSS signature (part of verifying process)	CO	Length of signature input, RSA-PSS signature input, RSA-PSS public key, padding	Decrypted RSA-PSS signature, Size of the recovered PSS padded message digest	int RSA_public_decrypt(int flen, const unsigned char *from, unsigned char *to, RSA *rsa, int padding)	N/A	RSA-PSS public key	Read Write Execute	N/A
Digital signature – RSA-PSS verify	CO	RSA public key, Original file digest, Decrypted RSA-PSS signature data, Salt length	Digest type, Size of the recovered message digest	int RSA_verify_PKCS1_PSS(RSA * rsa, const unsigned char * mHash, const EVP_MD * Hash, const unsigned char * EM, int sLen)	N/A	RSA public key	Read	N/A
Digital signature – delete the RSA key object	CO	RSA key object	N/A	**void RSA_free(RSA *rsa)**	N/A	N/A	Delete	N/A
Digital signature – RSA key generation	CO	size of key, Length of public exponent,	RSA key	RSA *RSA_generate_key(int bits, unsigned long e, void (*callback)(int, int, void *), void *cb_arg)	N/A	RSA public key	Read Write Execute	N/A

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
		Callback function for asynchronous call, Callback function arguments						
Key agreement and zeroize of the SSP								
DH – Create DH key context	CO	DH private key, Value NULL	DH key context	EVP_PKEY_CTX *EVP_PKEY_CTX_new (EVP_PKEY *pkey, ENGINE *e)	N/A	DH Private key	Read Write	N/A
DH – Initialize DH key context	CO	DH key context	N/A	int EVP_PKEY_derive_init (EVP_PKEY_CTX *ctx)	N/A	N/A	N/A	1 for success, <=0 for failure
DH – Set peer public key to DH key context	CO	DH key context, Other party public key	N/A	int EVP_PKEY_derive_set_peer(EVP_PKEY_CTX *ctx, EVP_PKEY *peer)	N/A	DH Private key, Other party public key	Read Write Execute	1 for success, <=0 for failure
DH – Derive shared secret	CO	DH key context	Derived shared secret, Length of key	int EVP_PKEY_derive(EVP_PKEY_CTX *ctx, unsigned char *key, size_t *keylen)	N/A	DH Private key, Other party public key	Read Write Execute	1 for success, <=0 for failure

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
DH – delete DH key context	CO	DH key context	N/A	void EVP_PKEY_CTX_free(EVP_PKEY_CTX *ctx)	N/A	N/A	Delete	N/A
DH – key generation: DH parameters	CO	specify generate parameters, Types of algo, length of key	output file where DH parameter is stored	DH parameters was generated using openssl command line interface: openssl genpkey -genparam -algorithm DH -pkeyopt dh_paramgen_prime_len:2048 -out dhparam.pem	N/A	DH parameter	Read Write Execute	N/A
DH – Generate private and public key based on the DH parameters	CO	DH parameters	output file where key is stored	DH private and public keys were generated using openssl command line interface: openssl genpkey -paramfile dhparam.pem -out dhkeyA.pem	N/A	DH keys	Read Write Execute	N/A
DH – Extract public key from pem file	CO	DH key file	output file where public key is stored	Extract the public key using openssl command line interface:	N/A	DH keys	Read Write Execute	N/A

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
				openssl pkey -in dhkeyA.pem -pubout -out dhpubA.pem				
Random number generation and zeroize of the SSP								
DRBG – Create key context	CO	N/A	Cipher context	EVP_CIPHER_CTX *EVP_CIPHER_CTX_new(void)	N/A	N/A	N/A	N/A
DRBG – Initialize encryption context	CO	Cipher context, Encryption method, Implementation, Specifies encryption key, Specifies initialization vector	N/A	int EVP_EncryptInit_ex(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher, ENGINE *impl, const unsigned char *key, const unsigned char *iv)	N/A	Seed key (key & iv)	N/A	N/A
DRBG – Delete context for random number generator	CO	Cipher context	N/A	EVP_CIPHER_CTX_free(EVP_CIPHER_CTX *c)	N/A	N/A	Delete	N/A

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
DRBG – random number encryption	CO	Cipher context, Plain text input, Length of plain text input	Cipher text result, Length of cipher text	int EVP_EncryptUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl, const unsigned char *in, int inl)	N/A	Seed key (key & iv)	Read Write Execute	N/A
Self-tests (integrity test and conditional self-test)								
Pre-operational self-test include software integrity test run automatically upon system start-up or reboot by calling NC2.vpn+-OpenSSL-Module-integrity.py executable file.	CO	N/A	N/A	NC2.vpn+-OpenSSL-Module- integrity.py.	N/A	N/A	N/A	<u>Successful Messages:</u> File opensslnc2 is intact. File libssl.so is intact. File libcrypto.so is intact. File Module-integrity is intact. File Module-KAItest is intact. <u>Failure Message:</u> File /usr/local/lib/libssl.so.11 does not exists! Shutting down VPN service. File /usr/local/lib/libcrypto.so.11 does not exists!

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
								Shutting down VPN service. File /usr/local/bin/selftest /NC2.vpn+OpenSSL-Module-integrity does not exists! Shutting down VPN service. File /usr/local/bin/selftest /NC2.vpn+OpenSSL-Module-KAItest does not exists! Shutting down VPN service. File /usr/local/bin/opensslnc 2 does not exists! Shutting down VPN service. Hash record read error {0}: {1} Shutting down VPN service. Hash record of opensslnc2 not exists! Shutting down VPN service.

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
								Hash record libssl.so and libcrypto.so does not exists! Shutting down VPN service. Hash record of Module-integrity and Module-KAItest does not exists! Shutting down VPN service. Warning! File opensslnc2 integrity compromised. Shutting down VPN service. Warning! File libssl.so integrity compromised. Shutting down VPN service. Warning! File libcrypto.so integrity compromised. Shutting down VPN service. Warning! File Module-integrity integrity compromised.

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
								Shutting down VPN service. Warning! File Module-KAItest integrity compromised. Shutting down VPN service.
Conditional self-tests including Known Answer Tests (KATs) and periodic self-test (every 20 minutes). These tests run automatically by calling NC2.vpn+-OpenSSL-Module-KAItest.py executable file and cannot be turned off.	CO	N/A	N/A	NC2.vpn+-OpenSSL-Module- KAItest.py.	N/A	N/A	N/A	<u>Successful message:</u> Starting AES self test. AES-128 encryption test successful. AES-128 decryption test successful. AES-192 encryption test successful. AES-192 decryption test successful. AES-256 encryption test successful. AES-256 decryption test successful. Starting Camellia self test. Camellia-128 encryption test successful.

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
								Camellia-128 decryption test successful. Camellia-192 encryption test successful. Camellia-192 decryption test successful. Camellia-256 encryption test successful. Camellia-256 decryption test successful. Starting SHA512 self test. SHA-512 test successful. Starting RSA-PSS self test. RSA-PSS signing test successful. RSA-PSS verification test successful. Starting DH self test. DH test successful.

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
								Starting AES-CTR-DRBG test. AES-128-CTR-DRBG test successful AES-192-CTR-DRBG test successful Self-test program completed. <u>Failure message:</u> Starting AES self test. AES-128 encryption test failed! AES-128 decryption test failed! AES-192 encryption test failed! AES-192 decryption test failed! AES-256 encryption test failed! AES-256 decryption test failed! Starting Camellia self test. Camellia-128 encryption test failed!

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
								Camellia-128 decryption test failed! Camellia-192 encryption test failed! Camellia-192 decryption test failed! Camellia-256 encryption test failed! Camellia-256 decryption test failed! Starting SHA512 self test. SHA-512 test failed! Starting RSA-PSS self test. RSA-PSS test internal error code: {encryptioncode}. RSA-PSS signing test failed! RSA-PSS verification test failed! Starting DH self test. DH test failed! Starting AES-CTR-DRBG test.

Services	Role	Data Input	Data Output	Control Input	Control Output	SSP	Access Type	Status Output
								AES-128-CTR-DRBG test failed! AES-192-CTR-DRBG test failed! Self-test program completed. Shutting down VPN services!

-- END OF DOCUMENT --