FIPS 140-2 Non-Proprietary Security Policy

McAfee Database Security Sensor Cryptographic Module Version 1.0

Document Version 1.1

November 19, 2013

Prepared For:

McAfee, Inc.

2821 Mission College Blvd.

Santa Clara, CA 95054

www.mcafee.com

Prepared By:

Apex Assurance Group, LLC

530 Lytton Avenue, Ste. 200

Palo Alto, CA 94301

www.apexassurance.com

Abstract

This document provides a non-proprietary FIPS 140-2 Security Policy for the Database Security Sensor Cryptographic Module Version 1.0.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 About FIPS 140

Federal Information Processing Standards Publication 140-2 — Security Requirements for Cryptographic Modules specifies requirements for cryptographic modules to be deployed in a Sensitive but Unclassified environment. The National Institute of Standards and Technology (NIST) and Communications Security Establishment of Canada (CSEC) Cryptographic Module Validation Program (CMVP) runs the FIPS 140 program. The National Voluntary Laboratory Accreditation Program (NVLAP) accredits independent testing labs to perform FIPS 140 testing; the CMVP validates test reports for modules meeting FIPS 140 validation. *Validated* is the term given to a product that is documented and tested against the FIPS 140 criteria.

More information is available on the CMVP website at http://csrc.nist.gov/groups/STM/cmvp/index.html.

1.2 About this Document

This non-proprietary Cryptographic Module Security Policy for the Database Security Sensor Cryptographic Module Version 1.0 from McAfee provides an overview of the product and a high-level description of how it meets the security requirements of FIPS 140-2. This document contains details on the module's cryptographic keys and critical security parameters. This Security Policy concludes with instructions and guidance on running the module in a FIPS 140-2 mode of operation.

The McAfee Database Security Sensor Cryptographic Module Version 1.0 may also be referred to as the "module" in this document.

1.3 External Resources

The McAfee website (http://www.mcafee.com) contains information on McAfee products. The Cryptographic Module Validation Program website (http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2013.htm) contains links to the FIPS 140-2 certificate and McAfee contact information.

1.4 Notices

This document may be freely reproduced and distributed in its entirety without modification.

1.5 Acronyms

The following table defines acronyms found in this document:

Acronym	Term
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
API	Application Programming Interface
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CSEC	Communications Security Establishment Canada
CSP	Critical Security Parameter
DES	Data Encryption Standard
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard
GPC	General Purpose Computer
GUI	Graphical User Interface
HMAC	(Keyed-) Hash Message Authentication Code
KAT	Known Answer Test
MAC	Message Authentication Code
MD	Message Digest
NIST	National Institute of Standards and Technology
OS	Operating System
PKCS	Public-Key Cryptography Standards
PRNG	Pseudo Random Number Generator
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TRIPLE-DES	Triple Data Encryption Algorithm
TLS	Transport Layer Security
USB	Universal Serial Bus

Table 1 – Acronyms and Terms

2 McAfee Database Security Sensor Cryptographic Module Version 1.0

2.1 Cryptographic Module Specification

The module, the McAfee Database Security Sensor Cryptographic Module Version 1.0, is a software shared library that provides cryptographic services required by the McAfee Database Security Sensor. The Module's logical cryptographic boundary is the shared dynamic library files and their integrity check HMAC files, which are as follows:

- **Linux/Unix**:
  fipscanister.o
- **Windows**:
  fipscanister.lib

The module is a multi-chip standalone embodiment installed on a General Purpose Computer.

All operations of the module occur via calls from the McAfee applications and their respective internal daemons/processes. As such there are no untrusted services calling the services of the module, as APIs are not exposed.

2.1.1 Validation Level Detail

The following table lists the level of validation for each area in FIPS 140-2:

FIPS 140-2 Section Title	Validation Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
Electromagnetic Interference / Electromagnetic Compatibility	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

Table 2 – Validation Level by DTR Section

2.1.2 Approved Cryptographic Algorithms

The module's cryptographic algorithm implementations have received the following certificate numbers from the Cryptographic Algorithm Validation Program:

Algorithm	CAVP Certificate
AES	2571
Triple-DES (3-key 168 bit encryption)	1557
DSA	786
PRNG (ANSI X9.31 Appendix A.2.4 using AES)	1223
RSA (X9.31, PKCS #1.5, PSS)	1318
SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	2166
HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC- SHA-384, HMAC-SHA-512	1587

Table 3 – FIPS-Approved Algorithm Certificates

Note that DSA supports a key size of less than 1024 bits except when not in FIPS mode. The Module only provides functions that implement Diffie-Hellman primitives.

2.1.3 Non-Approved Cryptographic Algorithms

The module supports the following non-FIPS 140-2 approved but allowed algorithms:

- Diffie-Hellman with key sizes of 1024-10000 bits (key agreement; key establishment methodology provides between 80 and 219 bits of encryption strength)
- RSA encrypt/decrypt with key sizes of 1024-15360 bits (key wrapping; key establishment methodology provides between 80 and 256 bits of encryption strength)

2.2 Module Interfaces

The figure below shows the module's physical and logical block diagram:

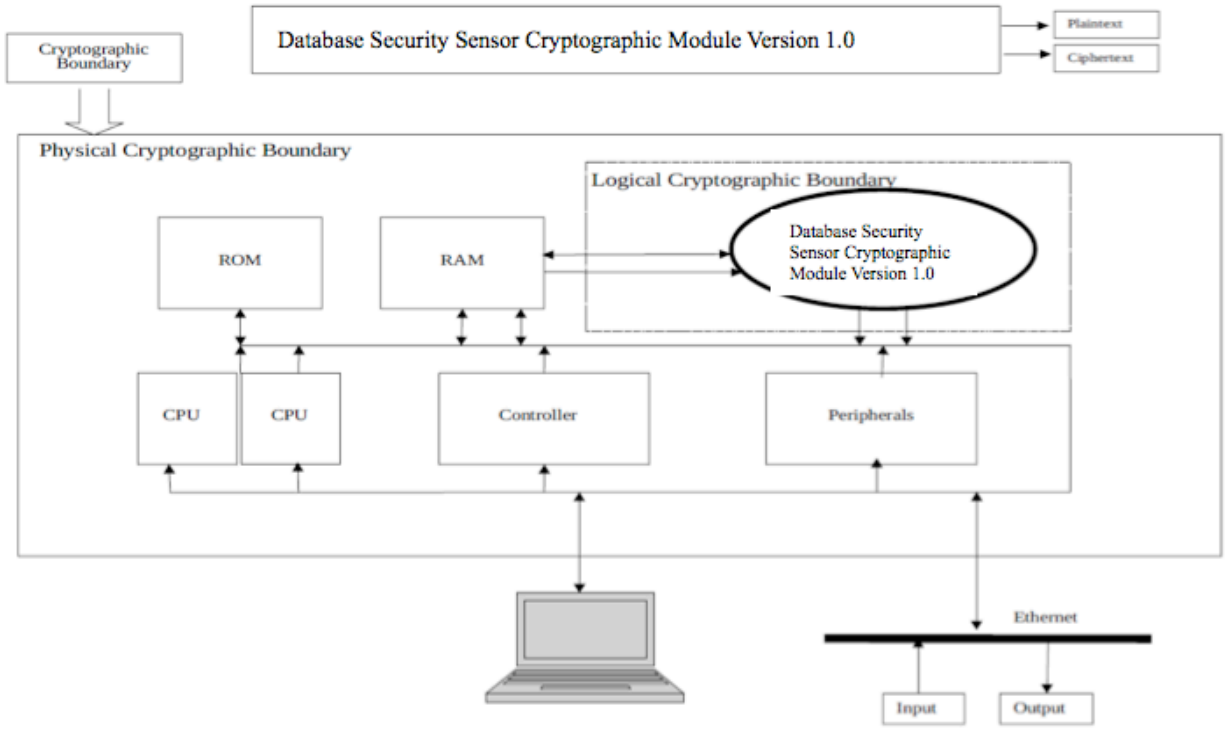

Figure 1 – Module Boundary and Interfaces Diagram

The physical ports of the Module are the same as the computer system on which it is executing. The logical interface is a C-language application program interface (API). The Data Input interface consists of the input parameters of the API functions. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the actual API functions. The Status Output interface includes the return values of the API functions.

FIPS 140-2 Interface	Logical Interface	Module Physical Interface
Data Input	Input parameters of API function calls	Ethernet Ports
Data Output	Output parameters of API function calls	Ethernet Ports
Control Input	API function calls	Keyboard, Serial port, Ethernet port
Status Output	API return codes	Keyboard, Serial port, Ethernet port
Power	None	PCI Compact Power Connector

Table 4 – Logical Interface / Physical Interface Mapping

As shown in Figure 1 – Module Boundary and Interfaces Diagram and Table 5 – Module Services and Descriptions, the output data path is provided by the data interfaces and is logically disconnected from processes performing key generation or zeroization. No key information will be output through the data output interface when the module zeroizes keys. No key information is output during key generation.

2.3 Roles, Services, and Authentication

The Module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both Crypto-User and Crypto-Officer roles. As allowed by FIPS 140-2, the Module does not support user authentication for those roles. Only one role may be active at a time and the Module does not allow concurrent operators.

The User and Crypto Officer roles are implicitly assumed by the entity accessing services implemented by the Module. The Crypto Officer can install and initialize the Module. The Crypto Officer role is implicitly entered when installing the Module or performing system administration functions on the host operating system.

- User Role: Loading the Module and calling any of the API functions. This role has access to all of the services provided by the Module.
- Crypto-Officer Role: Installation of the Module on the host computer system. This role is assumed implicitly when the system administrator installs the Module library file.

2.3.1 Operator Services and Descriptions

The module supports services that are available to users in the various roles. All of the services are described in detail in the module's user documentation. The following table shows the services available to the various roles and the access to cryptographic keys and CSPs resulting from services:

Service	Role	CSP	Access
Symmetric encryption/decryption	User, Crypto Officer	AES Key, Triple-DES Key	read/write/execute
Key Transport	User, Crypto Officer	RSA Private Key	read/write/execute
Digital signature	User, Crypto Officer	RSA Private Key, DSA Private Key	read/write/execute
Symmetric key generation	User, Crypto Officer	AES Key, Triple-DES Key	read/write/execute
Asymmetric key generation	User, Crypto Officer	RSA Private Key, DSA Private Key	read/write/execute
Keyed Hash (HMAC)	User, Crypto Officer	HMAC Key	read/write/execute
Message digest (SHS)	User, Crypto Officer	None	read/write/execute
Random number generation	User, Crypto Officer	PRNG Seed and Seed Key	read/write/execute
Show status	User, Crypto Officer	none	execute

Service	Role	CSP	Access
Module initialization	User, Crypto Officer	none	execute
Self test	User, Crypto Officer	None	execute
Zeroize	User, Crypto Officer	All CSPs	

Table 5 – Module Services and Descriptions

2.3.2 Operator Authentication

As required by FIPS 140-2, there are two roles (a Crypto Officer role and User role) in the module that operators may assume. As allowed by Level 1, the module does not support authentication to access services.

2.4 Physical Security

The Module is comprised of software only and thus does not claim any physical security.

2.5 Operational Environment

The module operates on a general purpose computer (GPC) running on a modern version of Microsoft Windows, UNIX, or Solaris as a general purpose operating system (GPOS). For FIPS purposes, the module is running on this operating system in single user mode and does not require any additional configuration to meet the FIPS requirements.

The module was tested on the following platforms:

Platform #	Platform Details
1	Windows Server 2008 R2 64-bit with VMWare ESXi 4.0 running on a HP Proliant DL185 GS;
2	Windows Server 2008 64-bit with VMWare ESXi 5.0 running on a HP Proliant DL380 GS;
3	HP-UX 11.23 running on a HP RX2600 Server;
4	AIX 5.3 on a IBM 9115-305;
5	Solaris 9 running on a Sun UltraSPARC C-III
6	Red Hat Enterprise Linux 5.9 with VMWare ESXi 5.0 running on a Dell PowerEdge R510;
7	CentOS 5.5 with VMWare ESXi 5.0 running on a Dell PowerEdge R510; and
8	SUSE 11 patch 2 with VMWare ESXi 5.0 running on a Dell PowerEdge R510;

Table 6 – Tested Platforms

The following platforms are vendor affirmed:

- Windows Server 2008 Standard with Service Pack 2 or later
- Windows Server 2008 Datacenter with Service Pack 2 or later
- Windows Server 2008 R2 Standard
- Windows Server 2008 R2 Datacenter

Compliance is maintained for other versions of the respective operating systems family where the binary is unchanged.

The platforms used during testing met Federal Communications Commission (FCC) FCC Electromagnetic Interference (EMI) and Electromagnetic Compatibility (EMC) requirements for business use as defined by 47 Code of Federal Regulations, Part15, Subpart B. FIPS 140-2 validation compliance is maintained when the module is operated on other versions of the GPOS running in single user mode, assuming that the requirements outlined in NIST IG G.5 are met.

2.6 Cryptographic Key Management

2.6.1 Key Generation

The Module supports generation of DH, DSA, and RSA public-private key pairs. The Module employs an ANSI X9.31 compliant random number generator for creation of asymmetric and symmetric keys.

The developer shall use entropy sources that contain at least 128 bits of entropy to seed the RNG as the module is not capable of detecting randomness or quality of the seeding material provided.

2.6.2 Key Storage

Public and private keys are provided to the Module by the calling process, and are destroyed when released by the appropriate API function calls. The Module does not perform persistent storage of keys.

2.6.3 Key Access

An authorized application as user (the Crypto-User) has access to all key data generated during the operation of the Module.

2.6.4 Key Protection and Zeroization

Keys residing in internally allocated data structures can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Zeroization of sensitive data is performed automatically by API function calls for intermediate data items, and on demand by the calling process using Module provided API function calls provided for that purpose.

Only the process that creates or imports keys can use or export them. No persistent storage of key data is performed by the Module. All API functions are executed by the invoking process in a nonoverlapping sequence such that no two API functions will execute concurrently.

2.7 Self-Tests

The Module performs both power-up self tests at module initialization and continuous condition tests during operation. Input, output, and cryptographic functions cannot be performed while the Module is in a self-test or error state as the module is single threaded and will not return to the calling application until the power-up self tests are complete. If the power-up self tests fail, subsequent calls to the module will fail and thus no further cryptographic operations are possible.

2.7.1 Power-On Self-Tests

Power-on self-tests are executed automatically when the module is loaded into memory. The `FIPS_mode_set()` function verifies the integrity of the runtime executable using a HMAC SHA-1 digest computed at build time. If the digest match, the power-up self-tests are then performed. If the power-up self-test is successful, `FIPS_mode_set()` sets the `FIPS_mode` flag to `TRUE` and the Module is in FIPS mode.

TYPE	DETAIL
Known Answer Tests[1]	• AES encrypt/decrypt (all supported modes, all supported key sizes) • HMAC SHA-1 • HMAC SHA-224 • HMAC SHA-256 • HMAC SHA-384 • HMAC SHA-512 • SHA-1 • SHA-224 • SHA-256 • SHA-384 • SHA-512 • PRNG • RSA (Sign and verify using 2048 bit key, SHA-256) • Triple-DES encrypt/decrypt (all supported modes, all supported key sizes)
Pair-wise Consistency Tests, Sign/verify	• DSA (all supported modes, all supported key sizes)
Module integrity	• HMAC-SHA-1

Table 7 – Power-On Self-Tests

2.7.2 Conditional Self-Tests

The module implements the following conditional self-tests upon key generation, or random number generation (respectively):

TYPE	DETAIL

[1] Note that all SHA-X KATs are tested as part of the respective HMAC SHA-X KAT.

TYPE	DETAIL
Pair-wise Consistency Tests	• DSA • RSA
Continuous Test	Performed on approved PRNG

Table 8 – Conditional Self-Tests

2.7.3 Cryptographic Function

A single initialization call, `FIPS_mode_set`, is required to initialize the Module for operation in the FIPS 140-2 Approved mode. When the Module is in FIPS mode, all security functions and cryptographic algorithms are performed in Approved mode.

The FIPS mode initialization is performed when the application invokes the `FIPS_mode_set()` call which returns a "1" for success or a "0" for failure. The module will support either explicit FIPS mode initialization through the `FIPS_mode_set()` function or implicit initialization by querying the `/proc/sys/crypto/fips_enabled flag`. If the flag is set and the module is being initialized, it will automatically call `FIPS_mode_set(1)` during this initialization. Prior to this invocation the Module is uninitialized in non-FIPS mode by default.

The `FIPS_mode_set()` function verifies the integrity of the runtime executable using a HMAC SHA-1 digest which is computed at build time. If this computed HMAC SHA-1 digest matches the stored, known digest, then the power-up self-test (consisting of the algorithm-specific Pairwise Consistency and Known Answer tests) is performed. If any component of the power-up self-test fails, an internal global error flag is set to prevent subsequent invocation of any cryptographic function calls. Any such power-up self-test failure is a hard error that can only be recovered by reinstalling the module[2]. If all components of the power-up self-test are successful, then the module is in FIPS mode. The power-up self-tests may be performed at any time by reloading the module.

No operator intervention is required during the running of the self-tests.

2.8 Mitigation of Other Attacks

The Module does not contain additional security mechanisms beyond the requirements for FIPS 140-2 Level 1 cryptographic modules.

[2] The `FIPS_mode_set()` function could be re-invoked but such re-invocation does not provide a means from recovering from an integrity test or known answer test failure

3 Guidance and Secure Operation

The tested operating systems segregate user processes into separate process spaces. Each process space is an independent virtual memory area that is logically separated from all other processes by the operating system software and hardware. The Module functions entirely within the process space of the process that invokes it, and thus satisfies the FIPS 140-2 requirement for a single user mode of operation.

The Module is installed as appropriate to the target system. A complete revision history of the source code from which the Module was generated is maintained in a version control database.

Upon initialization of the Module, the module will run its power-up self tests. Successful completion of the power-up self tests ensures that the module is operating in the FIPS mode of operation.

As the Module has no way of managing keys, any keys that are input or output from applications utilizing the module must be input or output in encrypted form using FIPS approved algorithms.

The self-tests can be called on demand by reinitializing the module using the FIPS_mode_set() function call, or alternatively using the FIPS_selftest() function call.