Red Hat Enterprise Linux 6.2 Kernel Crypto API Cryptographic Module v2.0

FIPS 140-2 Security Policy

Version 1.6

Last Update: 2013-02-08

Contents

1 Cryptographic Module Specification

This document is the non-proprietary security policy for the Red Hat Enterprise Linux 6.2 Kernel Crypto API Cryptographic Module, and was prepared as part of the requirements for conformance to Federal Information Processing Standard (FIPS) 140-2, Level 1.

The following section describes the module and how it complies with the FIPS 140-2 standard in each of the required areas.

1.1 Description of Module

The Red Hat Enterprise Linux 6.2 Kernel Crypto API Cryptographic Module is a software only cryptographic module that provides general-purpose cryptographic services to the remainder of the Linux kernel. The Red Hat Enterprise Linux 6.2 Kernel Crypto API Cryptographic Module is a software only, security level 1 cryptographic module, running on a multi-chip standalone platform.

The following table shows the overview of the security level for each of the eleven sections of the validation.

Security Component	Security 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
EMI/EMC	1
Self Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

Table 1: Security Levels

The module has been tested on the following multi-chip standalone platforms:

Manufacturer	Model	O/S & Ver.
HP	ProLiant DL585	Red Hat Enterprise Linux 6.2 (Single User Mode)
IBM	HS22	Red Hat Enterprise Linux 6.2 (Single User Mode)

Table 2: Tested Platforms

This cryptographic module provides main services with the Linux kernel. To perform integrity verification, additional software is used as outlined in the following list.

The list of components and the cryptographic boundary of the composite module is defined as follows:

- The Linux Kernel with the version of the RPM file of 2.6.32-220.4.2.el6.
- The module integrity check is performed by the Red Hat Enterprise Linux utility sha512hmac which uses the NSS library to obtain the cryptographic mechanisms. The name and version of the RPM file providing this utility is hmaccalc-0.9.12-1.el6.
- Network Security Service (NSS), a separately validated cryptographic module (FIPS 140-2 validation certificate #1837) provides cryptographic algorithms and security functions for sha512hmac application, which performs the integrity validation of the Linux kernel binary. The sha512hmac application uses the NSS module in accordance with the Security Rules stated in the NSS Cryptographic Module Version 3.12.9.1 Security Policy.
- The configuration of the FIPS mode is provided by the dracut-fips package with the version of the RPM file of 004-284.el6_3.1. This RPM version is provided with RHBA:2012-1318 accessible on the Red Hat Network.

1.2 Description of Modes of Operations

The module must always be configured as outlined in section 9.1. The module can be operated in FIPSmode or non-FIPS mode depending on what cryptographic functions are invoked.

In FIPS mode the kernel will support the following Approved functions:

- AES (Certs #1968, #1969, #1970, #1971, #1972), which is implemented in software and also with AES-NI, but only one of these implementations is enabled when the module is loaded
- Triple-DES (Certs #1278, #1279)
- SHA-1, 224, 256, 384, 512 (Certs #1725, #1726)
- HMAC SHA-1, 224, 256, 384, 512 (Certs #1187, #1188)
- RNG (X9.31) using AES 128 when called with fips(ansi_cprng) (Certs #1033, #1034, #1035, #1036, #1037)
- DSA (Certs #628, #629)

When used for the integrity check of the module, NSS will support the following Approved functions:

- DSA (Certs #634, #635)
- HMAC SHA-512 (Certs #1199, #1200)

The Linux Kernel Crypto API implements the following non-approved algorithms, which shall not be used in the FIPS mode of operation and can only be used in non-FIPS mode:

- DES
- Triple-DES in CTR mode
- XTS using AES192; although technically usable, CAVS currently does not provide a means to test it
- RNG (X9.31) when called with either stdrng or ansi_cprng

The module also implements cipher algorithms other than those listed above. These ciphers are technically unavailable. When calling these ciphers, the module returns an error.

1.3 The cryptographic module provides multiple implementations of AES given in the following bullet list:

- AES using AES-NI Intel instruction set when the aesni-intel kernel module is loaded (the kernel module can only be used if the underlying processor provides the AES-NI instruction set)

- AES implemented with streamlined assembler code when the aes-x86_64 kernel module is loaded
- AES implemented with regular C code when the aes-generic kernel module is loaded

Note, if more than one of the above listed kernel modules are loaded, the respective implementation can be requested by using the following cipher mechanism strings with the initialization calls (such as crypto_alloc_blkcipher):

- aesni-intel kernel module: "__aes-aesni"
- aes-x86_64 kernel module: "aes-asm"
- aes-generic kernel module: "aes-generic"

The AES cipher can also be loaded by simply using the string "aes" with the initialization call. In this case, the AES implementation whose kernel module is loaded with the highest priority is used. The following priority exist:

1. aesni-intel
2. aes-x86_64
3. aes-generic

For example: If the kernel modules aesni-intel and aes-asm are loaded and the caller uses the initialization call (such as crypto_alloc_blkcipher) with the cipher string of "aes", the aesni-intel implementation is used. On the other hand, if only the kernel modules of aes-x86_64 and aes-generic are loaded, the cipher string of "aes" implies that the aes-x86_64 implementation is used.

The discussion about the naming and priorities of the AES implementation also applies when cipher strings are used that include the block chaining mode, such as "cbc(aes-asm)", "cbc(aes)", or "cbc(__aes-aesni)".

The module maintains a process flag to indicate that the module is in FIPS 140-2 Approved mode. The flag is provided in the file /proc/sys/crypto/fips_enabled. If this file contains a value of 1, the module is operational in FIPS 140-2 approved mode.

1.4 Cryptographic Module Boundary

The Red Hat Enterprise Linux 6.2 Kernel Crypto API Cryptographic Module physical boundary is defined by the surface of the case of the platform. The logical module boundary is depicted in the software block diagram and is embodied by the Linux kernel and the kernel modules implementing the ciphers in /lib/modules/$(uname -r)/kernel/crypto (please note that only the modules implementing the approved mechanisms are available at runtime) and /lib/modules/$(uname -r)/kernel/arch/x86/crypto.

The integrity check mechanism provided by the sha512hmac application and the NSS library are part of the cryptographic module but not depicted as they are only used during load time of the module.

1.4.1 Hardware Block Diagram

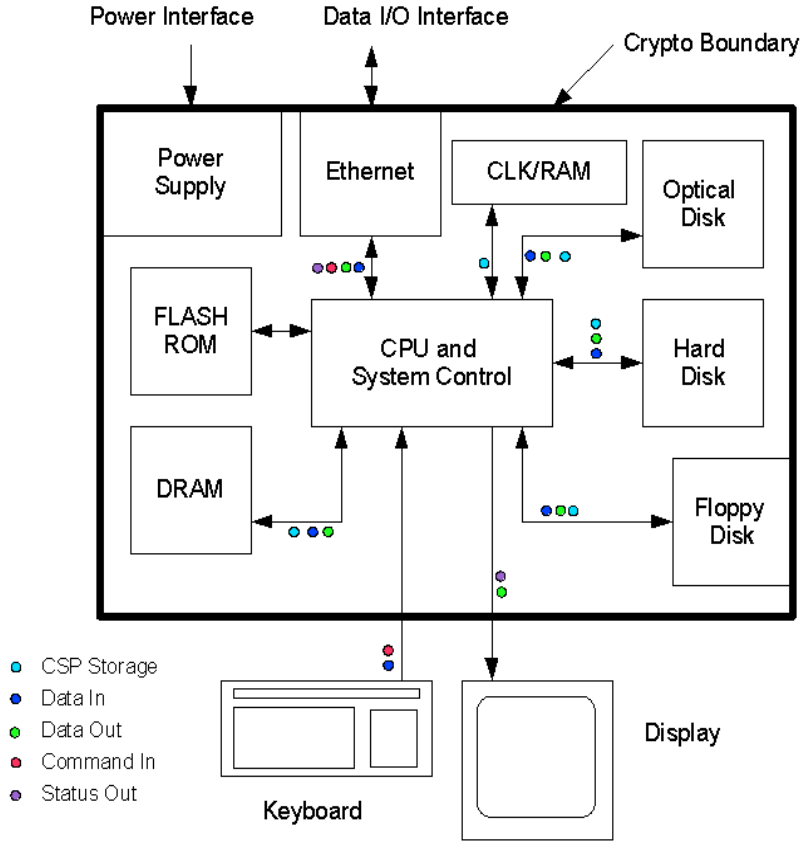

Figure 1: Hardware Block Diagram

This figure illustrates the various data, status and control paths through the cryptographic module. The physical boundary consists of the opaque enclosure surrounding the COTS PC system. The module consists of standard integrated circuits, including processors, and memory. The module does not include any security-relevant, semi- or custom integrated circuits or other active electronic circuit elements. The physical module includes power inputs and outputs, and internal power supplies. The cryptographic boundary contains only the security-relevant software elements that comprise the module.

1.4.2 Software Block Diagram

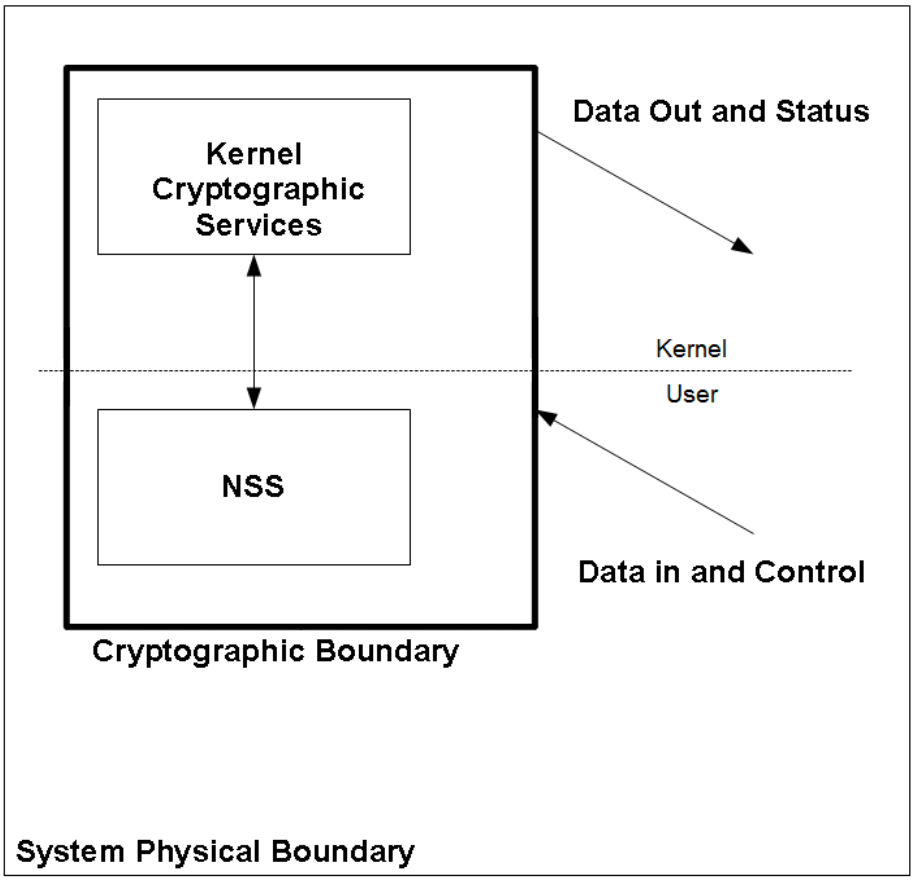

Figure 2: Software Block Diagram

1.5 Red Hat Enterprise Linux 6.2 Cryptographic Modules and FIPS 140-2 Certification

A set of kernel cryptographic libraries, services and user level cryptographic applications are certified at FIPS 140-2 level 1, providing a secure foundation for vendor use in developing dependent services, applications, and even purpose built appliances that may be FIPS 140-2 certified.

The certification is performed at FIPS 140-2 level 1, a software only certification that does not make any claims about the hardware enclosure. This allows vendors to develop their own higher level FIPS-certified modules using cryptographic modules.

The following cryptographic modules are included in the RHEL6.2 certification:

- Kernel Crypto API - a software only cryptographic module that provides general-purpose cryptographic services to the remainder of the Linux kernel
- Disk Volume Encryption - provides disk management and transparent partial or full disk encryption; Partial disk encryption encrypts only one or more partitions, leaving at least one partition as plaintext.
- Libgcrypt- supplies general cryptographic support for the Red Hat Enterprise Linux user space
- OpenSSL - a software library supporting cryptographic algorithms for general use by vendors
- OpenSSH-Server - supplies cryptographic support for the SSH protocol
- OpenSSH-Client - supplies cryptographic support for the SSH protocol
- Openswan - provides the IKE protocol version 1 key agreement services required for IPSec

1.5.1 Platforms

The certification was performed on a 64-bit system, which are capable of executing both 32 and 64-bit code concurrently. Vendors can "transfer" the FIPS 140-2 certificate to the other similar platforms, provided the binary is only recompiled without changing the code. This is called a vendor assertion.

1.5.2 Modes of Operations

Any vendor-provided FIPS 140-2 certified cryptographic modules and services that use the RHEL 6.2 underlying services to provide cryptographic functionality must use the RHEL 6 services installed and configured as outlined in the security policy document. Only the appropriately configured system ensures that FIPS 140-2 required self tests are executed and that ciphers are restricted to those that have been FIPS 140-2 certified by independent testing.

When invoking one of the non-approved ciphers which are still technically callable as listed above, the module implicitly transitions into non-FIPS mode. As all keys and Critical Security Parameters (CSP) handled by the module are ephemeral and there are no keys and CSPs shared between any functions, the module transitions back into FIPS mode immediately when invoking one of the approved ciphers. A re-invocation of the self-tests or integrity tests is not required.
Even when using this FIPS 140-2 non-approved mode, the module configuration ensures that the self-tests are always performed during initialization time of the module.
This section is given as guidance to developers to ensure the use cases conform with the initialization of the module. Although the modules operates in FIPS mode when following the guidance, the initialization of the FIPS 140-2 module may consist of several phases which may be preempted by developers, but not administrators or users. The following explanation describes the initialization phase to ensure the FIPS 140-2 module initializes correctly.

If dm-crypt is used to encrypt a disk or partition, particularly when other modules may store cryptographic keys or other CSPs (critical security parameters) there, it must be in FIPS 140-2 approved mode as described in dm-crypt Security Policy.

Since the kernel is in the approved mode, the remaining RHEL6.2 FIPS services (Crypto API, OpenSSL, Openswan, OpenSSH) start up in the approved mode by default. (Note that Openswan uses NSS for its cryptographic operations and NSS must explicitly be put into the approved mode with the modutil command.)

The approved mode for a module becomes effective as soon as the module power on self tests complete successfully and the module loads into memory. Self tests and integrity tests triggered in RHEL6.2 at startup or on the first invocation of the crypto module:

- kernel: during boot, before dm-crypt becomes available via dracut

- user space: before the user space module is available to the caller (i.e., for libraries during the initialization call, for applications: at load time)

See each module security policy for descriptions of self tests performed by each module and descriptions of how self test errors are reported for each module.

2 Cryptographic Module Ports and Interfaces

The physical ports of the Module are the same as the computer system on which it is executing.

FIPS Interface	Physical Port	Module Interface
Data Input	API	API input parameters
Data Output	API	API output parameters
Control Input	Kernel Command Line	API function calls
Status Output	Kernel log file (via dmesg)	API return codes
Power Input	PC Power Supply Port	Physical Power Connector

Table 3: Ports and Interfaces

3 Roles, Services and Authentication

3.1 Roles

Role	Services (see list below)
User	Encryption, Decryption, Random Numbers
Crypto Officer	Configuration, Encryption, Decryption, Random Numbers

Table 4: Roles

The Module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both User and Crypto Officer roles. 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. No further authentication is required. The Crypto Officer can install and initialize the Module.

3.2 Services

See section 1.2 for the complete list of approved and non-approved cryptographic services. This table only covers the approved services.

R – The item is read or referenced by the service
W – The item is written or updated by the service
Z – The persistent item is zeroized by the service

Role	Service	CSP	Algo/Mode	API Calls	Access Type
User, Crypto Officer	AES encrypt/decrypt	128,192, 256 bit key	ECB, CBC, CTR, CTR(RFC3686), CCM	All API functions with the prefix of crypto_cipher_, crypto_ablkcipher_ and crypto_blkcipher_ crypto_free_ablkcipher crypto_has_ablkcipher ablkcipher_request_set_tfm ablkcipher_request_free ablkcipher_request_set_callback ablkcipher_request_set_crypt crypto_free_blkcipher crypto_has_blkcipher	R, W, Z
User, Crypto Officer	AES encrypt/decrypt	128, 256 bit key	XTS	All API functions with the prefix of crypto_cipher_, crypto_ablkcipher_ and crypto_blkcipher_	R, W, Z

Role	Service	CSP	Algo/Mode	API Calls	Access Type
				crypto_free_ablkcipher crypto_has_ablkcipher ablkcipher_request_set_tfm ablkcipher_request_free ablkcipher_request_set_callback ablkcipher_request_set_crypt crypto_free_blkcipher crypto_has_blkcipher	
User, Crypto Officer	Triple-DES encrypt/decrypt	2 Key & 3 Key	ECB, CBC	All API functions with the prefix of crypto_cipher_, crypto_ablkcipher_ and crypto_blkcipher_ crypto_free_ablkcipher crypto_has_ablkcipher ablkcipher_request_set_tfm ablkcipher_request_free ablkcipher_request_set_callback ablkcipher_request_set_crypt crypto_free_blkcipher crypto_has_blkcipher	R, W, Z
User, Crypto Officer	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	N/A	N/A	All API functions with the prefix of crypto_digest_, crypto_hash_ crypto_free_hash crypto_has_hash	R, W, Z
User, Crypto Officer	HMAC SHA-1, HMAC SHA-224, HMAC SHA-256, HMAC SHA-384, HMAC SHA-512	HMAC Key for each SHA length	N/A	All API functions with the prefix of crypto_hmac_ crypto_free_hash	R, W, Z
User, Crypto Officer	RNG using AES 128	Seed and Seed Key	N/A	All API functions with the prefix of crypto_rng_ crypto_alloc_rng crypto_free_rng	R, W, Z
User, Crypto Officer	AEAD	This is a combined	N/A	All API functions with the prefix of crypto_aead_ and	R, W, Z

Role	Service	CSP	Algo/Mode	API Calls	Access Type
		crypto protocol, that only supports approved algorithms that this module supports. Keys for each approved algorithm used will be associated as required.		aead_request_ crypto_free_aead	
User, Crypto Officer	DSA used for the integrity verification of kernel modules loaded at runtime of the Linux kernel – this DSA service is provided at runtime of the module	DSA private key	N/A	N/A	R, W, Z
Crypto Officer	DSA for integrity verification of the NSS library at boot time of the module	DSA private key	N/A	N/A	R, W, Z
Crypto Officer	HMAC SHA-512 for integrity verification of the sha512hmac application and the static Linux kernel binary at boot time of the module	HMAC Key	N/A	N/A	R, W, Z

Table 5: Services

The FIPS 140-2 module covers the static kernel binary and therefore provides a number of non-security services to callers within kernel space as well as user space. The following documents provide a description of these services:

- Linux system call API man pages provided in chapter 2 of the Linux man pages obtainable from git://github.com/mkerrisk/man-pages.git
- Linux kernel internals including interfaces between kernel components documented in the book ISBN-

13: 978-0470343432

- Linux kernel driver development documentation covering the kernel interfaces available for device drivers: ISBN-13: 978-0596005900

The non-security services actually available will be based on the installed configuration.

3.3 Operator Authentication

There is no operator authentication; assumption of role is implicit by action.

3.4 Mechanism and Strength of Authentication

No authentication is required at security level 1; authentication is implicit by assumption of the role.

4 Physical Security

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

5 Operational Environment

This module will operate in a modifiable operational environment per the FIPS 140-2 definition.

5.1 Policy

The operating system shall be restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded).

The kernel component that makes calls to the cryptographic module is the single user of the cryptographic module, even when the application is serving multiple clients.

The ptrace(2) system call, the debugger (gdb(1)) and strace(1) shall not be used. In addition, other tracing mechanisms offered by the Linux environment, such as ftrace or systemtap shall not be used.

6 Cryptographic Key Management

The module has a random number generator that returns key material for use in constructing cryptographic keys.

Key Generation – Key material generated by the module is ANSI X9.31 Appendix A.2.4 conformant.

Key Usage – Key usage is the responsibility of the calling program and is outside the scope of the module.

Key Storage – With respect to the module, all keys are considered ephemeral. Any storage of keys is the responsibility of the calling program and is outside the scope of the module.

Key Destruction – When a calling kernel components calls the appropriate API function that operation overwrites freed memory with 0s (please see the API document for full details).

6.1 Random Number Generation

The Module employs an ANSI X9.31 compliant random number generator for creation of keys. Note: the RNG seed is the tuple {V key DT}, where those values are defined in ANSI X9.31 Appendix A.2.4.

Based on AES 128, seed/seed key is to be provided by the caller, usually by obtaining bits via get_random_bytes(), which provides access to the Linux kernel hardware-based non-deterministic random number generator.

To ensure FIPS 140-2 compliant RNG operation, make the following declaration when initializing the DRNG:

```
struct crypto_rng *fips_rng = crypto_alloc_rng("fips(ansi_cprng)", 0, 0);
```

All subsequent RNG operations must use fips_rng as the struct crypto_rng *tfm argument. This declaration prevents the use of ansi_cprng and stdrng which are not allowed in a FIPS approved mode.

7 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

Product Name and Model: HP ProLiant Server DL585 Series
Regulatory Model Number: HSTNS-1025
Product Options: All
EMC: Class A

Product Name and Model: IBM BladeCenter HS22 Series
Regulatory Model Number: 09-EMCRTP-0008
Product Options: All
EMC: Class A

8 Self Tests

8.1 Power-Up 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. The module is single threaded during the self tests and will stop the boot procedure, and therefore any subsequent operation before any other kernel component can request services from the module.

The crypto officer with physical or logical access to the module can run the POST on demand by power cycling the module or by rebooting the operating system.

Self Test	Description
Mandatory power-up tests performed at power-up and on demand:	
Cryptographic Algorithm Known Answer Tests	Each cryptographic algorithm provided by the Linux Kernel crypto API (see section 1.2 for list of algorithms), is tested using a “known answer” test to verify the correct operation of the algorithm with the exception of DSA. For complete details of the known answer tests, please see the Red Hat CryptoAPI Functional Spec document. DSA implemented by the Linux Kernel crypto API is verified by the kernel integrity test but may not be used. NOTE: HMAC SHA-512 and DSA provided by the NSS library are tested before the NSS library makes itself available to the sha512hmac application. In addition, if the Intel AES-NI support is present and the dracut-fips-aesni RPM package (see section 9.1) is installed, the AES-NI implementation is self-tested with the same KAT vector as the other AES implementations.
Software Integrity Test	A DSA (provided by the NSS library) calculation is performed by the NSS library to verify the integrity of the NSS library binary file. An HMAC SHA-512 (provided by the NSS library) calculation is performed on the sha512hmac utility and static Linux kernel binary to verify their integrity. The Linux kernel crypto API kernel modules, and any additional code modules loaded into the Linux kernel are checked with the DSA implementation of the Linux kernel when loading them into the kernel to confirm their integrity. NOTE: The fact that the kernel integrity check passed, which requires the loading of sha512hmac with the self tests implies a successful execution of the integrity and self tests of sha512hmac (the HMAC is stored in /usr/lib/hmaccalc/sha512hmac.hmac). With respect to the integrity check of kernel modules providing the cryptographic functionality, the fact that the self test of these cryptographic modules are displayed implies that the integrity checks of each kernel module passed successfully.
Conditional tests performed, as needed, during operation:	
Continuous RNG	128 bits continuous testing is performed during each use of the approved RNG as

	well as the non-approved hardware/hybrid RNG. This test is a “stuck at” test to check the RNG output data for repetition of a value.
On Demand Execution of Self Tests	
On Demand Testing	Invocation of the self tests on demand can be achieved by restarting the OS.

Table 6: Self Tests

A failure of any of the power up self tests panics the module. The only recovery is to reboot. For persistent failures, you must reinstall the kernel. See section 9.1 for details.

9 Guidance

This section provides guidance for the Cryptographic Officer and the User to maintain proper use of the module per FIPS 140-2 requirements.

9.1 Cryptographic Officer Guidance

Additional kernel modules not provided with the RHEL6.2 FIPS 140-2 validated kernel RPM package must not be loaded as the kernel configuration is fixed in approved mode.

To operate the kernel crypto module, the operating system must be restricted to a single operator mode of operation. (This should not be confused with single user mode which is runlevel 1 on RHEL. This refers to processes having access to the same cryptographic instance which RHEL ensures this cannot happen by the memory management hardware.)

Secure installation and Startup:

Crypto officers use the Installation instructions to install the module in their environment.

The version of the RPM containing the validated module is stated in section 1 above. The integrity of the RPM is automatically verified during the installation and the Crypto officer shall not install the RPM file if the RPM tool indicates an integrity error.

To bring the module into FIPS approved mode, perform the following:

1. Install the dracut-fips package:

```
# yum install dracut-fips
```

2. Recreate the INITRAMFS image:

```
# dracut -f
```

After regenerating the initrd, the crypto officer has to append the following string to the kernel command line by changing the setting in the boot loader:

```
fips=1
```

If /boot or /boot/efi resides on a separate partition, the kernel parameter boot=<partition of /boot or /boot/efi> must be supplied. The partition can be identified with the command `"df /boot"` or `"df /boot/efi"` respectively. For example:

```
$ df /boot
Filesystem         1K-blocks   Used       Available         Use%  Mounted on
/dev/sda1           233191     30454      190296            14%   /boot
```

The partition of /boot is located on /dev/sda1 in this example. Therefore, the following string needs to be appended to the kernel command line:

```
"boot=/dev/sda1"
```

FIPS 140-2 and AES-NI support

According to the kernel crypto API FIPS 140-2 security policy, the kernel crypto API module supports the AES-NI Intel processor instruction set as an approved cipher. The AES-NI instruction set is used by the kernel module.

In case you configured a full disk encryption using AES, you *may* use the AES-NI support for a higher performance compared to the software-only implementation.

To utilize the AES-NI support, the mentioned kernel module must be loaded during boot time by installing a plugin.

Before you install the plugin, you MUST verify that your processor offers the AES-NI instruction set by calling the following command:

```
cat /proc/cpuinfo | grep aes
```

If the command returns a list of properties, including the I<aes> string, your CPU provides the AES-NI instruction set. If the command returns nothing, AES-NI is not supported.

You MUST NOT install the following plugin if your CPU does not support AES-NI because the kernel will panic during boot.

The support for the AES-NI instruction set during boot time is enabled by installing the following plugin (make sure that the version of the plugin RPM matches the version of the installed RPMs!):

```
# install the dracut-fips-aesni package
rpm -Uhv dracut-fips-aesni-*.noarch.rpm
# recreate the initramfs image
dracut -f
```

The changes come into effect during the next reboot.

9.2 User Guidance

CTR and RFC3686 mode must only be used for IPSEC. It must not be used otherwise.

There are three implementations of AES: aes-generic, aesni-intel, and aes-x86_64 on x86_64 machines. The additional specific implementations of AES for i386 and s390 are disallowed and not available on the test platforms.

The Linux Kernel Crypto API does implement DES, Triple-DES in CTR mode, XTS using AES192, and RNG (X9.31), when called with either stdrng or ansi_cprngr, which are Non-Approved algorithms but shall not be used .

When using the module, the user shall utilize the Linux kernel crypto API provided memory allocation mechanisms. In addition, the user shall not use the function copy_to_user() on any portion of the data structures used to communicate with the Linux kernel crypto API.

To ensure FIPS 140-2 compliant RNG operation, make the following declaration when starting the library:

```
struct crypto_rng *fips_rng = crypto_alloc_rng("fips(ansi_cprng)", 0, 0);
```

All subsequent RNG operations must use fips_rng as the struct crypto_rng *tfm argument. This declaration

prevents the use of ansi_cprng and stdrng which are not allowed.

The Linux kernel crypto API-provided DRNG requires a reset operation after the initial boot sequence. This reset operation must be called by providing the seed and seed key for the DRNG. The caller should use the function get_random_bytes() to obtain the data for the seed and seed key.

Only the cryptographic mechanisms provided with the Linux kernel crypto API are considered to be used. The NSS library, although used, is only considered to support the integrity verification and is not intended for general-purpose use with respect to this cryptographic module.

9.3 Handling Self Test Errors

Self test failure within the Kernel Crypto API module or the dm-crypt kernel module will panic the kernel and the operating system will not load.

Recover from this error by trying to reboot the system. If the failure continues, you must reinstall the software package being sure to follow all instructions. If you downloaded the software verify the package hash to confirm a proper download. Contact Red Hat if these steps do not resolve the problem.

The Crypto API module performs a power-on self test that includes an integrity check and known answer tests for the available cryptographic algorithms. It also runs conditional tests on the RNG whenever it is used and on asymmetric cryptographic keys whenever they are generated.

The kernel dumps self test success and failure messages into the kernel message ring buffer. Post boot, the messages are moved to /var/log/messages.

Use **dmesg** to read the contents of the kernel ring buffer. The format of the ringbuffer (**dmesg**) output is:

alg: self-tests for %s (%s) passed

Typical messages are similar to "alg: self-tests for xts(aes) (xts(aes-x86_64)) passed" for each algorithm/sub-algorithm type.

10 Mitigation of Other Attacks

No other attacks are mitigated.

11 Glossary and Abbreviations

AES	Advanced Encryption Specification
CAVP	Cryptographic Algorithm Validation Program
CBC	Cypher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cypher Feedback
CC	Common Criteria
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CVT	Component Verification Testing
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
FSM	Finite State Model
HMAC	Hash Message Authentication Code
LDAP	Lightweight Directory Application Protocol
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OFB	Output Feedback
O/S	Operating System
PP	Protection Profile
RNG	Randome Number Generator
RSA	Rivest, Shamir, Addleman
SAP	Service Access Points
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol

SOF	Strength of Function
SSH	Secure Shell
SVT	Scenario Verification Testing
TDES	Triple DES
TOE	Target of Evaluation
UI	User Interface

Table 7: Abbreviations

12 References

[1] Kernel Cryptographic Services User Guide

[2] FIPS 140-2 Standard, http://csrc.nist.gov/groups/STM/cmvp/standards.html

[3] FIPS 140-2 Implementation Guidance, http://csrc.nist.gov/groups/STM/cmvp/standards.html

[4] FIPS 140-2 Derived Test Requirements,http://csrc.nist.gov/groups/STM/cmvp/standards.html

[5] FIPS 197 Advanced Encryption Standard, http://csrc.nist.gov/publications/PubsFIPS.html

[6] FIPS 180-3 Secure Hash Standard, http://csrc.nist.gov/publications/PubsFIPS.html

[7] FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC), http://csrc.nist.gov/publications/PubsFIPS.html

[8] FIPS 186-3 Digital Signature Standard (DSS), http://csrc.nist.gov/publications/PubsFIPS.html

