TippingPoint Crypto Core
FIPS Object Module for OpenSSL
Non-proprietary FIPS 140-2 Security Policy
by Trend Micro Inc.

Versions 2.0.8 and 2.0.13

Document Version: 1.6

October 16, 2019

Acknowledgments

Trend Micro Inc. acknowledges that this document was derived from the OpenSSL FIPS 140-2 Security Policy document from the CMVP FIPS validation certificate #1747.

Modification History

Date	Modifications
2019-10-16	Disallowed FIPS 186-2 RSA KeyGen and SigGen.
2019-2-27	Added new platform. CentOS 5.6 running on an Intel Xeon E5-2620v4. Updated module name to better fit OEM preferences.
2017-11-14	Revised the RSA listings.
2017-08-31	Added new platforms. Linux 4.4 on 440T running on an Intel Core i3-3220, Linux 4.4 on 2200T running on an Intel Xeon E5-2620, Linux 4.4 on 8200TX running on an Intel Xeon E5-2648L v3, Linux 4.4 on 8400TX running on an Intel Xeon E5-2648L v3, Linux 4.4 on ESXi 6.5 running on an Intel Xeon E5-2698 v3. Added vendor affirmed configurations.
2015-03-04	Added new platforms. CentOS 5.6 on Intel Xeon E5-2620v3 and CentOS 5.6 on Intel Xeon E5-2690v3

References

Reference	Full Specification Name
[ANS X9.31]	Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)
[FIPS 140-2]	Security Requirements for Cryptographic modules, May 25, 2001
[FIPS 180-4]	Secure Hash Standard
[FIPS 186-4]	Digital Signature Standard
[FIPS 197]	Advanced Encryption Standard
[FIPS 198-1]	The Keyed-Hash Message Authentication Code (HMAC)
[SP 800-38B]	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
[SP 800-38C]	Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality
[SP 800-38D]	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
[SP 800-56A]	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
[SP 800-67R1]	Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
[SP 800-89]	Recommendation for Obtaining Assurances for Digital Signature Applications
[SP 800-90A]	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
[SP 800-131A]	Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths

Table of Contents

1. Introduction

This document is the non-proprietary security policy for the TippingPoint Crypto Core FIPS Object Module for OpenSSL Versions 2.0.8 and 2.0.13 hereafter referred to as the Module.

The Module is a software library providing a C-language application program interface (API) for use by other processes that require cryptographic functionality. The Module is classified by FIPS 140-2 as a software module, multi-chip standalone module embodiment. The physical cryptographic boundary is the general purpose computer on which the module is installed. The logical cryptographic boundary of the Module is the fipscanister object module, a single object module file named fipscanister.o (Linux®[1] /Unix®[2] and Vxworks®[3]) or fipscanister.lib (Microsoft Windows®[4]). The Module performs no communications other than with the calling application (the process that invokes the Module services).

Note that this Module is a rebranded Module based on OEM OpenSSL (Certs. #1747).

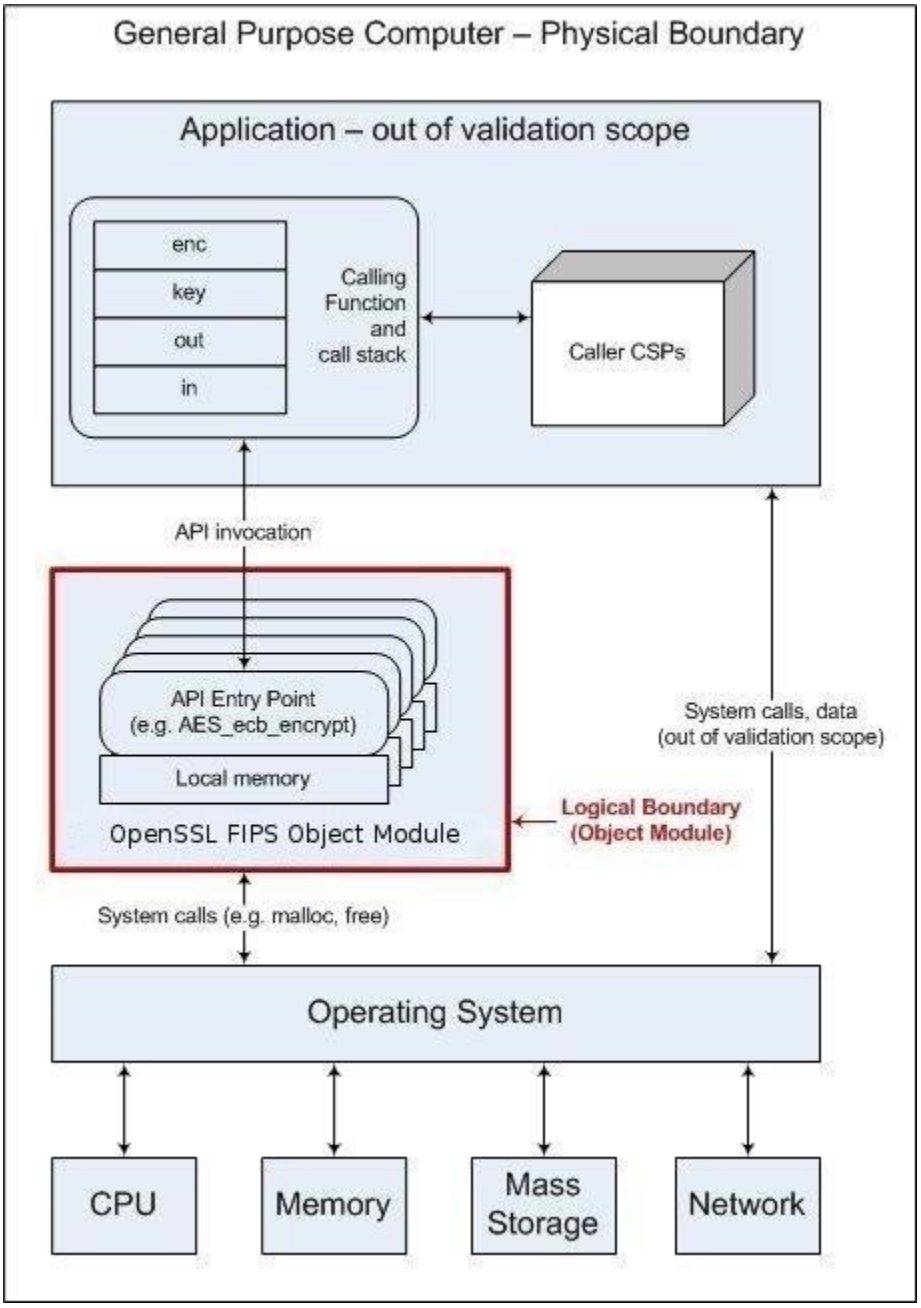
The FIPS 140-2 security levels for the Module are as follows:

Table 1: Security Level of Security Requirements

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	2
Finite State Model	1
Physical Security	NA
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	NA

[1] Linux is the registered trademark of Linus Torvalds in the U.S. and other countries
[2] UNIX is a registered trademark of The Open Group
[3] Vxworks is a registered trademark owned by Wind River Systems, Inc
[4] Windows is a registered trademark of Microsoft Corporation in the United States and other countries

Figure 1: Module Block Diagram



2. Tested Configurations

Table 2: Tested Configurations (B = Build Method; EC = Elliptic Curve Support). The EC column indicates support for prime curve only (P), or all NIST defined B, K, and P curves (BKP).

#	Operational Environment	Processor	Optimizations (Target)-	EC	B
1.	Android 2.2 (HTC Desire)	Qualcomm QSD 8250 (ARMv7)	NEON	P	U2
2.	Android 2.2 (Dell Streak)	Qualcomm QSD 8250 (ARMv7)	None	P	U2
3.	Microsoft Windows 7 32 bit	Intel Celeron (x86)	None	BKP	W2
4.	uClinux 0.9.29	ARM 922T (ARMv4)	None	BKP	U2
5.	Fedora 14	Intel Core i5 (x86)	AES-NI	BKP	U2
6.	HP-UX 11i (hpux-ia64-cc, 32 bit mode)	Intel Itanium 2 (IA64)	None	BKP	U1
7.	HP-UX 11i (hpux64-ia64-cc, 64 bit mode)	Intel Itanium 2 (IA64)	None	BKP	U1
8.	Ubuntu 10.04	Intel Pentium T4200 (x86)	None	BKP	U2
9.	Android 3.0	NVIDIA Tegra 250 T20 (ARMv7)	None	P	U2
10.	Linux 2.6.27	PowerPC e300c3 (PPC)	None	BKP	U2
11.	Microsoft Windows 7 64 bit	Intel Pentium 4 (x86)	None	BKP	W2
12.	Ubuntu 10.04 32 bit	Intel Core i5 (x86)	AES-NI	BKP	U2
13.	Linux 2.6.33	PowerPC32 e300 (PPC)	None	BKP	U2
14.	Android 2.2	OMAP 3530 (ARMv7)	NEON	BKP	U2
15.	DSP Media Framework 1.4	TI C64x+	None	BKP	U2
16.	VxWorks 6.8	TI TNETV1050 (MIPS)	None	BKP	U2
17.	Linux 2.6	Broadcom BCM11107 (ARMv6)	None	BKP	U2
18.	Linux 2.6	TI TMS320DM6446 (ARMv4)	None	BKP	U2
19.	Linux 2.6.32	TI AM3703CBP (ARMv7)	None	BKP	U2
20.	Solaris 10 32bit	SPARC-T3 (SPARCv9)	None	BKP	U2
21.	Solaris 10 64bit	SPARC-T3 (SPARCv9)	None	BKP	U2
22.	Solaris 11 32bit	Intel Xeon 5675 (x86)	None	BKP	U2

#	Operational Environment	Processor	Optimizations (Target)-	EC	B
23.	Solaris 11 64bit	Intel Xeon 5675 (x86)	None	BKP	U2
24.	Solaris 11 32bit	Intel Xeon 5675 (x86)	AES-NI	BKP	U2
25.	Solaris 11 64bit	Intel Xeon 5675 (x86)	AES-NI	BKP	U2
26.	Oracle Linux 5 64bit	Intel Xeon 5675 (x86)	None	BKP	U2
27.	CascadeOS 6.1 32bit	Intel Pentium T4200 (x86)	None	BKP	U2
28.	CascadeOS 6.1 64bit	Intel Pentium T4200 (x86)	None	BKP	U2
29.	Ubuntu 10.04 32bit	Intel Pentium T4200 (x86)	None	BKP	U1
30.	Ubuntu 10.04 64bit	Intel Pentium T4200 (x86)	None	BKP	U1
31.	Oracle Linux 5	Intel Xeon 5675 (x86)	AES-NI	BKP	U2
32.	Oracle Linux 6	Intel Xeon 5675 (x86)	None	BKP	U2
33.	Oracle Linux 6	Intel Xeon 5675 (x86)	AES-NI	BKP	U2
34.	Solaris 11 32bit	SPARC-T3 (SPARCv9)	None	BKP	U2
35.	Solaris 11 64bit	SPARC-T3 (SPARCv9)	None	BKP	U2
36.	Android 4.0	NVIDIA Tegra 250 T20 (ARMv7)	None	P	U2
37.	Linux 2.6	Freescale PowerPC-e500	None	BKP	U2
38.	Apple iOS 5.1	ARMv7	None	BKP	U2
39.	WinCE 6.0	ARMv5TEJ	None	BKP	W2
40.	WinCE 5.0	ARMv7	None	BKP	W2
41.	Android 4.0	OMAP 3	NEON	P	U2
42.	NetBSD 5.1	PowerPC-e500	None	BKP	U2
43.	NetBSD 5.1	Intel Xeon 5500 (x86)	None	BKP	U2
44.	Windows 7 64-bit	Intel Core i5-2430M (x86)	AES-NI	BKP	W2
45.	Android 4.1	TI DM3730 (ARMv7)	None	P	U2
46.	Android 4.1	TI DM3730 (ARMv7)	NEON	P	U2
47.	Android 4.2	Nvidia Tegra 3 (ARMv7)	None	P	U2
48.	Android 4.2	Nvidia Tegra 3 (ARMv7)	NEON	P	U2
49.	Windows Embedded Compact 7	Freescale i.MX53xA (ARMv7)	NEON	BKP	W2
50.	Windows Embedded Compact 7	Freescale i.MX53xD	NEON	BKP	W2

#	Operational Environment	Processor	Optimizations (Target)-	EC	B
		(ARMv7)			
51.	Android 4.0	Qualcomm Snapdragon APQ8060 (ARMv7)	NEON	BKP	U2
52.	Apple OS X 10.7	Intel Core i7-3615QM	None	BKP	U2
53.	Apple iOS 5.0	ARM Cortex A8 (ARMv7)	NEON	BKP	U2
54.	OpenWRT 2.6	MIPS 24Kc	None	BKP	U2
55.	QNX 6.4	Freescale i.MX25 (ARMv4)	None	BKP	U2
56.	Apple iOS 6.1	Apple A6X SoC (ARMv7s)	None	BKP	U2
57.	eCos 3	Freescale i.MX27 926ejs (ARMv5TEJ)	None	BKP	U2
58.	Ubuntu 13.04	AM335x Cortex-A8 (ARMv7)	None	BKP	U2
59.	Ubuntu 13.04	AM335x Cortex-A8 (ARMv7)	NEON	BKP	U2
60.	Linux 3.8	ARM926 (ARMv5TEJ)	None	BKP	U2
61.	Apple iOS 6.0	Apple A5 / ARM Cortex-A9 (ARMv7)	None	BKP	U2
62.	Apple iOS 6.0	Apple A5 / ARM Cortex-A9 (ARMv7)	NEON	BKP	U2
63.	Linux 2.6	Freescale e500v2 (PPC)	None	BKP	U2
64.	AcanOS 1.0	Intel Core i7-3612QE (x86)	None	BKP	U2
65.	AcanOS 1.0	Intel Core i7-3612QE (x86)	AES-NI	BKP	U2
66.	AcanOS 1.0	Feroceon 88FR131 (ARMv5)	None	BKP	U2
67.	FreeBSD 8.4	Intel Xeon E5440 (x86)	None	BKP	U2
68.	FreeBSD 9.1	Xeon E5-2430L (x86)	None	BKP	U2
69.	FreeBSD 9.1	Xeon E5-2430L (x86)	AES-NI	BKP	U2
70.	ArbOS 5.3	Xeon E5645 (x86)	None	BKP	U2
71.	ArbOS 5.3	Xeon E5645 (x86)	AES-NI	BKP	U2
72.	Linux ORACLESP 2.6	ASPEED AST-Series (ARMv5)	None	BKP	U2
73.	Linux ORACLESP 2.6	Emulex PILOT 3 (ARMv5)	None	BKP	U2
74.	FreeBSD 9.2	Xeon E5-2430L (x86)	None	BKP	U2

#	Operational Environment	Processor	Optimizations (Target)-	EC	B
75.	FreeBSD 9.2	Xeon E5-2430L (x86)	AES-NI	BKP	U2
76.	FreeBSD 10.0	Xeon E5-2430L (x86)	None	BKP	U2
77.	FreeBSD 10.0	Xeon E5-2430L (x86)	AES-NI	BKP	U2
78.	FreeBSD 8.4 32-bit	Xeon E5440 (x86)	None	BKP	U2
79.	VMware Horizon Workspace 2.1 x86 under vSphere ESXi 5.5	Intel Xeon E3-1220	None	BKP	U2
80.	VMware Horizon Workspace 2.1 x86 under vSphere ESXi 5.5	Intel Xeon E3-1220	AES-NI	BKP	U2
81.	QNX 6.5 on ARMv4	Freescale i.MX25 (ARMv4)	None	BKP	U2
82.	CentOS 5.6	Xeon E5-2620v3	None	BKP	U2
83.	CentOS 5.6	Xeon E5-2690v3	None	BKP	U2
84.	Linux 4.4 on 440T	Intel Core i3-3220	SSE2	BKP	U2
85.	Linux 4.4 on 2200T	Intel Xeon E5-2620	AES-NI	BKP	U2
86.	Linux 4.4 on 8200TX	Intel Xeon E5-2648L v3	AES-NI	BKP	U2
87.	Linux 4.4 on 8400TX	Intel Xeon E5-2648L v3	AES-NI	BKP	U2
88.	Linux 4.4 on VMware ESXi 6.5 (VTPS)	Intel Xeon E5-2698 v3	SSE2	BKP	U2
89.	CentOS 5.6	Intel Xeon E5-2620 v4	None	BKP	U2

See Appendix A for additional information on build method and optimizations. See Appendix C for a list of the specific compilers used to generate the Module for the respective operational environments.

2.1 Vendor Affirmed Configurations

The module can execute in additional operational environments, each composed from a combination of the following hardware platforms and hypervisors. The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment that is not listed on the validation certificate. The following hardware platforms are Vendor affirmed:

- Any **Intel Xeon based** server hardware platforms supported by the below mentioned hypervisors

The following hypervisors are Vendor affirmed:

- KVM – Redhat Enterprise Linux
- VMWare ESXi

3. Ports and Interfaces

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).

Table 3: Logical interfaces

Logical Interface Type	Description
Control input	API entry point and corresponding stack parameters
Data input	API entry point data input stack parameters
Status output	API entry point return values and status stack parameters
Data output	API entry point data output stack parameters

As a software module, control of the physical ports is outside 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 is single-threaded and in error scenarios returns only an error value (no data output is returned).

4. Modes of Operation and Cryptographic Functionality

The Module supports only a FIPS 140-2 Approved mode. Tables 4a and 4b list the Approved and Non-approved but Allowed algorithms, respectively.

Table 4a: FIPS Approved Cryptographic Functions

Function	Algorithm	Options	Cert #
Random Number Generation; Symmetric key generation	[SP 800-90A] DRBG[5] Prediction resistance supported for all variations	Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), with and without derivation function	157, 229, 264, 292, 316, 342, 485 540, 739, 1601, C 418
Encryption, Decryption and CMAC	[SP 800-67]	3-Key Triple-DES TECB, TCBC, TCFB, TOFB; CMAC generate and verify	1223, 1346, 1398 1465, 1492 1522 1695 1742 1868, 2495, C 418
	[FIPS 197] AES	128/256 XTS; 128/ 192/256 ECB, CBC, OFB, CFB	1884,

5 [5] For all DRBGs the "supported security strengths" is just the highest supported security strength per [SP800-90A] and [SP800-57].

Function	Algorithm	Options	Cert #
	[SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM [SP 800-38E] XTS	1, CFB 8, CFB 128, CTR; CCM; GCM; CMAC generate and verify	2116, 2234, 2342, 2394, 2484, 2824, 2929, 3281, 4703, C 418
Message Digests	[FIPS 180-4]	SHA-1, SHA-2 (224, 256, 384, 512)	1655, 1840, 1923, 2019, 2056, 2102, 2368, 2465, 2719, 3850, C 418
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	1126, 1288, 1363, 1451, 1485, 1526, 1768, 1856, 2078, 3115, C 418
Digital Signature and Asymmetric Key	[FIPS 186-2] RSA	SigVer9.31(1024/1536/2048/3072/4096 with all SHA-2 sizes except SHA-224), SigVerPKCS1.5 (1024/1536/2048/3072/4096 with all with all SHA-2	960, 1086,

Function	Algorithm	Options	Cert #
Generation		sizes), SigVerPSS (1024/1536/2048/3072/4096 with all with all SHA-2 sizes) **Note**: Users of this library should keep DRBG as the random function when using these RSA options.	1145, 1205, 1237, 1273, 1477, 1535, 1678, 2563, C 418
	[FIPS 186-4] RSA	SigGen9.31 (2048 SHA(256 , 384 , 512)) (3072 SHA (256 , 384 , 512)), SigGenPKCS1.5 (2048 SHA (224 , 256 , 384, 512)) (3072 SHA(224 , 256 , 384 , 512)), SigGenPSS (2048 SHA (224, 256, 384, 512)) (3072 SHA (224), 256, 384, 512)) SigVer9.31 (1024/2048/3072 with all SHA-2 sizes except SHA-224), SigVerPKCS1.5 (1024/2048/3072 with all SHA-2 sizes), SigVerPSS (1024/2048/3072 with all SHA-2 sizes)	1535, 1678, 2563, C 418
	[FIPS 186-4] DSA	PQG Gen, Key Pair Gen, Sig Gen (2048/3072 with all SHA-2 sizes) PQG Ver, Sig Ver (1024/2048/3072 with all SHA sizes]	589, 661, 693, 734, 748, 764, 853, 870, 938, 1245, C 418
	[FIPS 186-2] ECDSA	PKG: CURVES(P-224 P-384 P-521 K-233 K- 283 K-409 K-571 B-233 B-283 B-409 B-571) PKV: CURVES(P-192 P-224 P-256 P-384 P-521 K-163 K-233 K-283 K-409 K-571 B-163 B-233 B-283 B-409 B-571)	270, 315, 347, 383, 394, 413, 496

Function	Algorithm	Options	Cert #
		PKG: CURVES(P-224 P-384 P-521) PKV: CURVES(P-192 P-224 P-256 P-384 P-521)	264, 378
	[FIPS 186-4] ECDSA	PKG: CURVES(P-224 P-256 P-384 P-521 K- 224 K-256 K-384 K-521 B-224 B-256 B-384 B- 521 ExtraRandomBits TestingCandidates) PKV: CURVES(ALL-P ALL-K ALL-B) SigGen: CURVES(P-224: (SHA-224, 256, 384, 512) P-256: (SHA-224, 256, 384, 512) P-384: (SHA-224, 256, 384, 512) P-521: (SHA-224, 256, 384, 512) K-233: (SHA-224, 256, 384, 512) K-283: (SHA-224, 256, 384, 512) K-409: (SHA- 224, 256, 384, 512) K-571: (SHA-224, 256, 384, 512) B-233: (SHA-224, 256, 384, 512) B-283: (SHA-224, 256, 384, 512) B-409: (SHA-224, 256, 384, 512) B-571: (SHA-224, 256, 384, 512)) SigVer: CURVES(P-192: (SHA-1, 224, 256, 384, 512) P-224: (SHA-1, 224, 256, 384, 512) P- 256: (SHA-1, 224, 256, 384, 512) P-384: (SHA-1, 224, 256, 384, 512) P-521: (SHA-1, 224, 256, 384, 512) K-163: (SHA-1, 224, 256, 384, 512) K-233: (SHA-1, 224, 256, 384, 512) K-283: (SHA- 1, 224, 256, 384, 512) K-409: (SHA-1, 224, 256, 384, 512) K-571: (SHA-1, 224, 256, 384, 512 B-163: (SHA-1, 224, 256, 384, 512) B-233: (SHA- 1, 224, 256, 384, 512) B-283: (SHA-1, 224, 256, 384, 512) B-409: (SHA-1, 224, 256, 384, 512) B-571: (SHA-1, 224, 256, 384, 512))	270, 315, 347, 383, 394, 413, 496, 528, 634, 1161, C 418
		PKG: CURVES(P-224 P-256 P-384 P-521) PKV: CURVES(ALL-P) SigGen: CURVES(P-224: (SHA-224, 256, 384, 512) P-256: (SHA-224, 256, 384, 512) P-384: (SHA-224, 256, 384, 512) P-521: (SHA-224, 256, 384, 512)) SigVer: CURVES(P-192: (SHA-1, 224, 256, 384, 512) P-224: (SHA-1, 224, 256, 384, 512) P- 256: (SHA-1, 224, 256, 384, 512) P-384: (SHA-1, 224, 256, 384, 512) P-521: (SHA-1, 224, 256,384, 512))	264, 378
ECC CDH (CVL)	[SP 800-56A] (§5.7.1.2)	All NIST defined B, K and P curves except sizes 163 and 192	12, 24, 36, 53, 71, 85, 260, 331, 464,

Function	Algorithm	Options	Cert #
			1346, C 418
		All NIST defined P curves except size 192.	10, 49

The Module supports only NIST defined curves for use with ECDSA and ECC CDH. The Module supports two operational environment configurations for elliptic curve; NIST prime curve only (listed in Table 2 with the EC column marked "P") and all NIST defined curves (listed in Table 2 with the EC column marked "BKP").

Table 4b: Non-FIPS Approved But Allowed Cryptographic Functions

Category	Algorithm	Description
Key Agreement	EC Diffie-Hellman	Legacy (untested) Diffie-Hellman scheme using elliptic curve, supporting all NIST defined B, K and P curves.
Key Encryption, Decryption	RSA	The RSA algorithm may be used by the calling application for encryption or decryption of keys. No claim is made for SP 800-56B compliance, and no CSPs are established into or exported out of the module using these services.

The Module implements the following services which are Non-Approved per the SP 800-131A transition:

Table 4c: FIPS-Non-Approved Cryptographic Functions

Function	Algorithm	Options	Cert #
Random Number Generation; Symmetric key generation	[SP 800-90] DRBG	Dual EC DRBG	157, 229, 264, 292, 316, 342, 485
Random Number Generation; Symmetric key generation	[ANS X9.31] RNG	AES 128/192/256	985, 1087, 1119, 1166, 1186, 1202, 1278, 1292, 1314, 1351
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	KeyGen, SigGen9.31, SigGenPKCS1.5, SigGenPSS (All key sizes and SHA sizes)	960, 1086, 1145, 1205,

Function	Algorithm	Options	Cert #
			1237, 1273, 1477, 1535, 1678, 2563
	[FIPS 186-2] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)	589, 661, 693, 734, 748, 764
	[FIPS 186-4] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)	589, 661, 693, 734, 748, 764
	[FIPS 186-2] ECDSA	PKG: CURVES(P-192 K-163 B-163 P-224 P-384 P-521 K-233 K- 283 K-409 K-571 B-233 B-283 B-409 B-571) SIG(gen): CURVES(P-192 P-224 P-256 P-384 P-521 K-163 K-233 K-283 K-409 K-571 B-163 B-233 B-283 B-409 B-571)	270, 315, 347, 383, 394, 413, 496
			264, 378
	[FIPS 186-4] ECDSA	PKG: CURVES(P-192 K-163 B-163) SigGen: CURVES(P-192: (SHA-1, 224, 256, 384, 512) P-224:(SHA-1) P-256:(SHA-1) P-384: (SHA-1) P-521:(SHA-1) K-163: (SHA-1, 224, 256, 384, 512) K-233:(SHA-1) K-283:(SHA-1) K-409:(SHA-1) K-571:(SHA-1) B-163: (SHA-1, 224, 256,	270, 315, 347, 383, 394, 413
			264,

Function	Algorithm	Options	Cert #
		384, 512) B 233:(SHA-1) B-283: (SHA-1) B-409:(SHA-1) B-571:(SHA-1))	378
ECC CDH (CVL)	[SP 800-56A] (§5.7.1.2)	All NIST Recommended B, K and P curves sizes 163 and 192	12, 24, 36, 53, 71, 85
			10, 49

These algorithms shall not be used when operating in the FIPS Approved mode of operation.

EC Diffie-Hellman Key Agreement provides a maximum of 256 bits of security strength. RSA Key Wrapping provides a maximum of 256 bits of security strength.

The Module requires an initialization sequence (see IG 9.5): the calling application invokes `FIPS_mode_set()`[6], which returns a "1" for success and "0" for failure. If `FIPS_mode_set()` fails then all cryptographic services fail from then on. The application can test to see if FIPS mode has been successfully performed.

The Module is a cryptographic engine library, which can be used only in conjunction with additional software. Aside from the Module use of the NIST defined elliptic curves as trusted third party domain parameters, all other FIPS 186-3 assurances are outside the scope of the Module, and are the responsibility of the calling process.

4.1 Critical Security Parameters and Public Keys

All CSPs used by the Module are described in this section. All access to these CSPs by Module services are described in Section 4. The CSP names are generic, corresponding to API parameter data structures.

Table 4.1a: Critical Security Parameters

CSP Name	Description
RSA SGK	RSA (2048 to 16384 bits) signature generation key

[6] The function call in the Module is `FIPS_module_mode_set()` which is typically used by an application via the `FIPS_mode_set()` wrapper function

CSP Name	Description
RSA KDK	RSA (2048 to 16384 bits) key decryption (private key transport) key
DSA SGK	[FIPS 186-4] DSA (2048/3072) signature generation key
ECDSA SGK	ECDSA (All NIST defined B, K, and P curves except sizes 163 and 192) signature generation key
EC Diffie-Hellman Private	EC Diffie-Hellman (All NIST defined B, K, and P curves except sizes 163 and 192) private key agreement key
AES EDK	AES (128/192/256) encrypt / decrypt key
AES CMAC	AES (128/192/256) CMAC generate / verify key
AES GCM	AES (128/192/256) encrypt / decrypt / generate / verify key
AES XTS	AES (256/512) XTS encrypt / decrypt key
Triple-DES EDK	Triple-DES (3-Key) encrypt / decrypt key
Triple-DES CMAC	Triple-DES (3-Key) CMAC generate / verify key
HMAC Key	Keyed hash key (160/224/256/384/512)
Hash_DRBG CSPs	V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength)
HMAC_DRBG CSPs	V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input(length dependent on security strength)
CTR_DRBG CSPs	V (128 bits) and Key (AES 128/192/256), entropy input (length dependent on security strength)
CO-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for Crypto Officer role authentication
User-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for User role authentication

Authentication data is loaded into the module during the module build process, performed by an authorized operator (Crypto Officer), and otherwise cannot be accessed.

The module does not output intermediate key generation values.

Table 4.1b: Public Keys

CSP Name	Description
RSA SVK	RSA (1024 to 16384 bits) signature verification public key
RSA KEK	RSA (2048 to 16384 bits) key encryption (public key transport) key

CSP Name	Description
DSA SVK	[FIPS 186-4] DSA (1024/2048/3072) signature verification key or [FIPS 186-2] DSA(1024) signature verification key
ECDSA SVK	ECDSA (All NIST defined B, K and P curves) signature verification key
EC Diffie-Hellman Public	EC Diffie-Hellman (All NIST defined B, K and P curves) public key agreement key

For all CSPs and Public Keys:

Storage: RAM, associated to entities by memory location. The Module stores DRBG state values for the lifetime of the DRBG instance. The module uses CSPs passed in by the calling application on the stack. The Module does not store any CSP persistently (beyond the lifetime of an API call), with the exception of DRBG state values used for the Modules' default key generation service.

Generation: The Module implements SP 800-90A compliant DRBG services for creation of symmetric keys, and for generation of DSA, elliptic curve, and RSA keys as shown in Table 4a. The calling application is responsible for storage of generated keys returned by the module.

Entry: All CSPs enter the Module's logical boundary in plaintext as API parameters, associated by memory location. However, none cross the physical boundary.

Output: The Module does not output CSPs, other than as explicit results of key generation services. However, none cross the physical boundary.

Destruction: Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs. In addition, the module provides functions to explicitly destroy CSPs related to random number generation services. The calling application is responsible for parameters passed in and out of the module.

Private and secret keys as well as seeds and entropy input are provided to the Module by the calling application, and are destroyed when released by the appropriate API function calls. Keys residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. All API functions are executed by the invoking calling application in a non-overlapping sequence such that no two API functions will execute concurrently. An authorized application as user (Crypto-Officer and User) has access to all key data generated during the operation of the Module.

In the event Module power is lost and restored the calling application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed.

Module users (the calling applications) shall use entropy sources that meet the security strength required for the random number generation mechanism: 128 bits for the SP800-90A DRBG as shown in Table 2 (Hash_DRBG, HMAC_DRBG) and Table 3 (CTR_DRBG). This entropy is

supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

5. Roles, Authentication and Services

The Module implements the required User and Crypto Officer roles and requires authentication for those roles. Only one role may be active at a time and the Module does not allow concurrent operators. The User or Crypto Officer role is assumed by passing the appropriate password to the `FIPS_module_mode_set()` function.

The password values may be specified at build time and must have a minimum length of 16 characters. Any attempt to authenticate with an invalid password will result in an immediate and permanent failure condition rendering the Module unable to enter the FIPS mode of operation, even with subsequent use of a correct password.

Authentication data is loaded into the Module during the Module build process, performed by the Crypto Officer, and otherwise cannot be accessed.

Since minimum password length is 16 characters, the probability of a random successful authentication attempt in one try is a maximum of $1/256^{16}$, or less than $1/10^{38}$. The Module permanently disables further authentication attempts after a single failure, so this probability is independent of time.

Both roles have access to all of the services provided by the Module.

- User Role (User): Loading the Module and calling any of the API functions.
- Crypto Officer Role (CO): Installation of the Module on the host computer system and calling of any API functions.

All services implemented by the Module are listed below, along with a description of service CSP access.

Table 5: Services and CSP Access

Service	Role	Description
Initialize	User, CO	Module initialization. Does not access CSPs.
Self-test	User, CO	Perform self tests (FIPS_selftest). Does not access CSPs.
Show status	User, CO	Functions that provide module status information: Version (as unsigned long or const char *) FIPS Mode (Boolean) Does not access CSPs.

Service	Role	Description
Zeroize	User, CO	Functions that destroy CSPs: fips_drbg_uninstantiate: for a given DRBG context, overwrites DRBG CSPs (Hash_DRBGCSPs,HMAC_DRBGCSPs,CTR_DRBGCSPs) All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application.
Random number generation	User, CO	Used for random number and symmetric key generation. Seed or reseed an DRBG instance Determine security strength of DRBG instance Obtain random data Uses and updates Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs.
Asymmetric key generation	User, CO	Used to generate DSA and ECDSA keys: DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK There is one supported entropy strength for each mechanism and algorithm type, the maximum specified in SP800-90A
Symmetric encrypt/decrypt	User, CO	Used to encrypt or decrypt data. Executes using AES EDK, Triple-DES EDK (passed in by the calling process).
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. Executes using AES CMAC, Triple-DES, CMAC (passed in by the calling process).
Message digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest. Does not access CSPs.
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. Executes using HMAC Key (passed in by the calling process).
Key transport[7]	User, CO	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). Executes using RSA KDK, RSA KEK (passed in by the calling process).
Key agreement	User, CO	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the module). Executes using EC Diffie-Hellman Private, EC Diffie-

[7] "Key transport" can refer to a) moving keys in and out of the module or b) the use of keys by an external application. The latter definition is the one that applies to the TippingPoint Crypto Core OpenSSL.

Service	Role	Description
		Hellman Public (passed in by the calling process).
Digital signature	User, CO	Used to generate or verify RSA, DSA or ECDSA digital signatures. Executes using RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process).
Utility	User, CO	Miscellaneous helper functions. Does not access CSPs

6. Self-Test

The Module performs the self-tests listed below on invocation of Initialize or Self-test.
Table 6a: Power On Self Tests (KAT = Known answer test; PCT = Pairwise consistency test

Table 6a: Power On Self Tests (KAT = Known answer test; PCT = Pairwise consistency test)

Algorithm	Type	Test Attributes
Software integrity	KAT	HMAC-SHA1
HMAC	KAT	One KAT per SHA1, SHA224, SHA256, SHA384 and SHA512 Per IG 9.3, this testing covers SHA POST requirements.
AES	KAT	Separate encrypt and decrypt, ECB mode, 128 bit key length
AES CCM	KAT	Separate encrypt and decrypt, 192 key length
AES GCM	KAT	Separate encrypt and decrypt, 256 key length
XTS-AES	KAT	128, 256 bit key sizes to support either the 256-bit key size (for XTS-AES-128) or the 512-bit key size (for XTS-AES-256)
AES CMAC	KAT	Sign and verify CBC mode, 128, 192, 256 key lengths
Triple-DES	KAT	Separate encrypt and decrypt, ECB mode, 3-Key
Triple-DES CMAC	KAT	CMAC generate and verify, CBC mode, 3-Key
RSA	KAT	Sign and verify using 2048 bit key, SHA-256, PKCS#1
DSA	PCT	Sign and verify using 2048 bit key, SHA-384
DRBG	KAT	CTR_DRBG: AES, 256 bit with and without derivation function HASH_DRBG: SHA256 HMAC_DRBG: SHA256
ECDSA	PCT	Keygen, sign, verify using P-224, K-233 and SHA512. The K-233 self-test is not performed for operational environments that support prime curve only (see Table 2).
ECC CDH	KAT	Shared secret calculation per SP 800-56A §5.7.1.2, IG 9.6
X9.31 RNG	KAT	128, 192, 256 bit AES keys Note: This KAT is not performed in version 2.0.13 of the module. In version 2.0.8, although the module performs a self test for X9.31 RNG, this algorithm cannot be used in FIPS mode.

The Module is installed using one of the set of instructions in Appendix A, as appropriate for the target system. The HMAC-SHA-1 of the Module distribution file as tested by the CMT Laboratory and listed in Appendix A is verified during installation of the Module file as described in Appendix A.

The `FIPS_mode_set()`[8] function performs all power-up self-tests listed above with no operator intervention required, returning a "1" if all power-up self-tests succeed, and a "0" otherwise. If any component of the power-up self-test fails an internal flag is set to prevent subsequent invocation of any cryptographic function calls. The module will only enter the FIPS Approved mode if the module is reloaded and the call to `FIPS_mode_set()`[9] succeeds.

The power-up self-tests may also be performed on-demand by calling `FIPS_selftest()`, which returns a "1" for success and "0" for failure. Interpretation of this return code is the responsibility of the calling application.

The Module also implements the following conditional tests:

Table 6b: Conditional Tests

Algorithm	Test
DRBG	Tested as required by [SP800-90A] Section 11
DRBG	FIPS 140-2 continuous test for stuck fault
DSA	Pairwise consistency test on each generation of a key pair
ECDSA	Pairwise consistency test on each generation of a key pair
RSA	Pairwise consistency test on each generation of a key pair Note: FIPS 186-2 RSA KeyGen is disallowed.

In the event of a DRBG self-test failure the calling application must uninstantiate and re-instantiate the DRBG per the requirements of [SP 800-90A]; this is not something the Module can do itself.

Pairwise consistency tests are performed for both possible modes of use, e.g. Sign/Verify and Encrypt/Decrypt.

The Module supports two operational environment configurations for elliptic curve: NIST prime curves only (listed in Table 2 with the EC column marked "P") and all NIST defined curves (listed in Table 2 with the EC column marked "BKP").

[8] `FIPS_mode_set()` calls Module function `FIPS_module_mode_set()`

[9] `FIPS_mode_set()` calls Module function `FIPS_module_mode_set()`

7. Operational Environment

The tested operating systems segregate user processes into separate process spaces. Each process space is logically separated from all other processes by the operating system software and hardware. The Module functions entirely within the process space of the calling application, and implicitly satisfies the FIPS 140-2 requirement for a single user mode of operation.

8. Mitigation of other Attacks

The module is not designed to mitigate against attacks which are outside of the scope of FIPS 140-2.

Appendix A Installation and Usage Guidance

The test platforms represent different combinations of installation instructions. For each platform there is a build system, the host providing the build environment in which the installation instructions are executed, and a target system on which the generated object code is executed. The build and target systems may be the same type of system or even the same device, or may be different systems – the Module supports cross-compilation environments.

Please note, the following instructions are not applicable to the 2.0.13 version of the Module as the FIPS Object Module is preinstalled in the Trend Micro Inc. operational environments.

Each of these command sets are relative to the top of the directory containing the uncompressed and expanded contents of the distribution files *openssl-fips-2.0.8.tar.gz* (all NIST defined curves as listed in Table 2 with the EC column marked "BKP") or *openssl-fips-ecp-2.0.8.tar.gz* (NIST prime curves only as listed in Table 2 with the EC column marked "P"). The command sets are:

```
U1:
      ./config no-asm
      make
      make install

U2:
      ./config
      make
      make install

W1:
      ms\do_fips no-asm

W2:
      ms\do_fips
```

Installation instructions

1. Download and copy the distribution file to the build system.
   These files can be downloaded from *http://www.openssl.org/source* /.

2. Verify the HMAC-SHA-1 digest of the distribution file; see Appendix B. An independently acquired FIPS 140-2 validated implementation of SHA-1 HMAC must be used for this digest verification. Note that this verification can be performed on any convenient system and not necessarily on the specific build or target system. Alternatively, a copy of the distribution on physical media can be obtained from OSF[10]. Unpack the distribution

[10]For some prospective users the acquisition, installation, and configuration of a suitable FIPS 140-2 validated product may not be convenient. OSF will on request mail a CD containing the source code distribution, via USPS or international post. A distribution file received by that means need not be verified by a FIPS 140-2 validated implementation of HMAC-SHA-1. For instructions on requesting this CD see http://opensslfoundation.com/fips/verify.html.

```
   gunzip -c openssl-fips-2.0.8.tar.gz | tar xf -
   cd openssl-fips-2.0.8
or
   gunzip -c openssl-fips-ecp-2.0.8.tar.gz | tar xf -
   cd openssl-fips-ecp-2.0.8
```

4. Execute one of the installation command sets U1, W1, U2, W2 as shown above. No other command sets shall be used.

5. The resulting *fipscanister.o* or *fipscanister.lib* file is now available for use.

6. The calling application enables FIPS mode by calling the `FIPS_mode_set()` [11] function.

Note that failure to use one of the specified commands sets exactly as shown will result in a module that cannot be considered compliant with FIPS 140-2.

Linking the Runtime Executable Application

Note that applications interfacing with the FIPS Object Module are outside of the cryptographic boundary. When linking the application with the FIPS Object Module two steps are necessary:

1. The HMAC-SHA-1 digest of the FIPS Object Module file must be calculated and verified against the installed digest to ensure the integrity of the FIPS object module.

2. A HMAC-SHA1 digest of the FIPS Object Module must be generated and embedded in the FIPS Object Module for use by the `FIPS_mode_set()` [11] function at runtime initialization.

The `fips_standalone_sha1` command can be used to perform the verification of the FIPS Object Module and to generate the new HMAC-SHA-1 digest for the runtime executable application. Failure to embed the digest in the executable object will prevent initialization of FIPS mode.

At runtime the `FIPS_mode_set()`[11] function compares the embedded HMAC-SHA-1 digest with a digest generated from the FIPS Object Module object code. This digest is the final link in the chain of validation from the original source to the runtime executable application file.

Optimization

The “asm” designation means that assembler language optimizations were enabled when the binary code was built, “no-asm” means that only C language code was compiled.

[11] `FIPS_mode_set()` calls the Module function `FIPS_module_mode_set()`

For OpenSSL with x86 there are three possible optimization levels:

1. No optimization (plain C)
2. SSE2 optimization
3. AES-NI+PCLMULQDQ+SSSE3 optimization

Other theoretically possible combinations (e.g. AES-NI only, or SSE3 only) are not addressed individually, so that a processor which does not support all three of AES-NI, PCLMULQDQ, and SSSE3 will fall back to SSE2 optimization.

For more information, see:

- http://www.intel.com/support/processors/sb/CS-030123.htm?wapkw=sse2
- http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions- aes-ni/?wapkw=aes-ni

For OpenSSL with ARM there are two possible optimization levels:

1. Without NEON

2. With NEON (ARM7 only)

For more information, see http://www.arm.com/products/processors/technologies/neon.php

Appendix B Controlled Distribution File Fingerprint

Please note, the following instructions are not applicable to the 2.0.13 version of the Module as the FIPS Object Module is preinstalled in the Trend Micro Inc. operational environments.

The *TippingPoint Crypto Core FIPS Object Module for OpenSSL v2.0.8* consists of the FIPS Object Module (the *fipscanister.o* or *fipscanister.lib* contiguous unit of binary object code) generated from the specific source files.

For all NIST defined curves (listed in Table 2 with the EC column marked "BKP") the source files are in the specific special OpenSSL distribution *openssl-fips-2.0.8.tar.gz* with HMAC-SHA-1 digest of

```
7f486fbb598f3247ab9db10c1308f1c19f384671
```

Please contact Trend Micro Inc. for source distribution.

The `openssl` command from a version of OpenSSL that incorporates a previously validated version of the module may be used:

```
openssl sha1 -hmac etaonrishdlcupfm openssl-fips-2.0.8.tar.gz
```

Appendix C Compilers

This appendix lists the specific compilers used to generate the Module for the respective Operational Environments. Note this list does not imply that use of the Module is restricted to only the listed compiler versions, only that the use of other versions has not been confirmed to produce a correct result.

Table C: Compilers

#	Operational Environment	Compiler
1.	Android 2.2 (HTC Desire)	gcc 4.4.0
2.	Android 2.2 (Dell Streak)	gcc 4.4.0
3.	Microsoft Windows 7 32 bit	Microsoft 32-bit C/C++ Optimizing Compiler Version 16.00
4.	uClinux 0.9.29	gcc 4.2.1
5.	Fedora 14	gcc 4.5.1
6.	HP-UX 11i (hpux-ia64-cc, 32 bit mode)	HP C/aC++ B3910B
7.	HP-UX 11i (hpux64-ia64-cc, 64 bit mode)	HP C/aC++ B3910B
8.	Ubuntu 10.04	gcc 4.1.3
9.	Android 3.0	gcc 4.4.0
10.	Linux 2.6.27	gcc 4.2.4
11.	Microsoft Windows 7 64 bit	Microsoft C/C++ Optimizing Compiler Version 16.00 for x64
12.	Ubuntu 10.04 32 bit	gcc 4.1.3
13.	Linux 2.6.33	gcc 4.1.0
14.	Android 2.2	gcc 4.1.0
15.	DSP Media Framework 1.4	TMS320C6x C/C++ Compiler v6.0.13
16.	VxWorks 6.8	gcc 4.1.2
17.	Linux 2.6	gcc 4.3.2
18.	Linux 2.6	gcc 4.3.2
19.	Linux 2.6.32	gcc 4.3.2
20.	Solaris 10 32bit	gcc 3.4.3
21.	Solaris 10 64bit	gcc 3.4.3
22.	Solaris 11 32bit	gcc 4.5.2
23.	Solaris 11 64bit	gcc 4.5.2
24.	Solaris 11 32bit	gcc 4.5.2
25.	Solaris 11 64bit	gcc 4.5.2

#	Operational Environment	Compiler
26.	Oracle Linux 5 64bit	gcc 4.1.2
27.	CascadeOS 6.1 32bit	gcc 4.4.5
28.	CascadeOS 6.1 64bit	gcc 4.4.5
29.	Ubuntu 10.04 32bit	gcc 4.1.3
30.	Ubuntu 10.04 64bit	gcc 4.1.3
31.	Oracle Linux 5	gcc 4.1.2
32.	Oracle Linux 6	gcc 4.4.6
33.	Oracle Linux 6	gcc 4.4.6
34.	Solaris 11 32bit	Sun C 5.12
35.	Solaris 11 64bit	Sun C 5.12
36.	Android 4.0	gcc 4.4.3
37.	Linux 2.6	gcc 4.1.0
38.	Apple iOS 5.1	gcc 4.2.1
39.	WinCE 6.0	Microsoft C/C++ Optimizing Compiler Version 15.00 for ARM
40.	WinCE 5.0	Microsoft C/C++ Optimizing Compiler Version 13.10 for ARM
41.	Android 4.0	gcc 4.4.3
42.	NetBSD 5.1	gcc 4.1.3
43.	NetBSD 5.1	gcc 4.1.3
44.	Windows 7	Microsoft (R) C/C++ Optimizing Compiler Version 16.00 for x64
45.	Android 4.1	gcc 4.6
46.	Android 4.1	gcc 4.6
47.	Android 4.2	gcc 4.6
48.	Android 4.2	gcc 4.6
49.	Windows Embedded Compact 7	Microsoft C/C++ Optimizing Compiler Version 15.00.20720 for ARM
50.	Windows Embedded Compact 7	Microsoft C/C++ Optimizing Compiler Version 15.00.20720 for ARM
51.	Android 4.0	gcc 4.4.3
52.	Apple OS X 10.7	Apple LLVM version 4.2
53.	Apple iOS 5.0	gcc 4.2.1
54.	OpenWRT 2.6	gcc 4.6.3
55.	QNX 6.4	gcc 4.3.3

#	Operational Environment	Compiler
56.	Apple iOS 6.1	gcc 4.2.1
57.	eCos 3	gcc 4.3.2
58.	Ubuntu 13.04	gcc 4.7.3
59.	Ubuntu 13.04	gcc 4.7.3
60.	Linux 3.8	gcc 4.7.3
61.	Apple iOS 6.0	gcc 4.2.1
62.	Apple iOS 6.0	gcc 4.2.1
63.	Linux 2.6	gcc 4.4.1
64.	AcanOS 1.0	gcc 4.6.2
65.	AcanOS 1.0	gcc 4.6.2
66.	AcanOS 1.0	gcc 4.5.3
67.	FreeBSD 8.4	gcc 4.2.1
68.	FreeBSD 9.1	gcc 4.2.1
69.	FreeBSD 9.1	gcc 4.2.1
70.	ArbOS 5.3	gcc 4.1.2
71.	ArbOS 5.3	gcc 4.1.2
72.	Linux ORACLESP 2.6	gcc 4.4.5
73.	Linux ORACLESP 2.6	gcc 4.4.5
74.	FreeBSD 9.2	gcc 4.2.1
75.	FreeBSD 9.2	gcc 4.2.1
76.	FreeBSD 10.0	clang 3.3
77.	FreeBSD 10.0	clang 3.3
78.	FreeBSD 8.4	gcc 4.2.1
79.	VMware Horizon Workspace 2.1 x86 under vSphere	gcc 4.5.1
80.	VMware Horizon Workspace 2.1 x86 under vSphere	gcc 4.5.1
81.	QNX on ARMv4	gcc 4.3.3
82.	CentOS 5.6	gcc 4.1.2
83.	CentOS 5.6	gcc 4.1.2
84.	Linux 4.4 on 440T	gcc 5.3.0
85.	Linux 4.4 on 2200T	gcc 5.3.0
86.	Linux 4.4 on 8200TX	gcc 5.3.0
87.	Linux 4.4 on 8400TX	gcc 5.3.0

#	Operational Environment	Compiler
88.	Linux 4.4 on VMware ESXi 6.5 (VTPS)	gcc 5.3.0
89.	CentOS 5.6	gcc 4.1.2