

How to check if AES-NI is enabled for OpenSSL on Linux

Author : Dan Nanni

Categories : [Hardware](#), [Security](#)

Tagged as : [aes](#), [encryptionopenssl](#)

Question: I have a Linux server which has Intel AES-NI hardware capability. I would like to check whether currently installed OpenSSL can use AES-NI acceleration.

Intel Advanced Encryption Standard New Instructions (AES-NI) is a special instruction set for x86 processors, which is designed to accelerate the execution of AES algorithms. AES-based symmetric encryption is widely used in a variety of security applications and protocol implementations (e.g., IPSec, SSL/TLS, HTTPS, SSH). OpenSSL crypto library supports AES-based ciphers as well.

To support AES-NI, OpenSSL provides so-called EVP crypto APIs (e.g., `EVP_Decrypt/EVP_Encrypt`) which can automatically leverage hardware acceleration (if available) and fall back to software implementation (if not available), via a single interface. If you want to check whether currently installed OpenSSL supports AES-NI hardware acceleration, you can test using OpenSSL's EVP APIs.

Check if AES-NI is Available on CPU Processors

Before proceeding, first verify that current CPUs have the AES instruction set. For this you can inspect CPU flags as follows.

```
$ grep -m1 -o aes /proc/cpuinfo
```

```
aes
```

If the output shows `aes`, that means AES-NI engine is available on current CPUs.

Check if AES-NI is Enabled for OpenSSL

To check whether OpenSSL can leverage AES instruction sets, you can use OpenSSL's EVP APIs. When EVP APIs are called, they can automatically detect the presence of AES-NI and accelerate AES encryption computations using AES instruction sets. Thus you can compare AES performance with or without EVP functions. If AES-NI is available for OpenSSL, you will see significant performance boost when EVP functions are used.

Let's use OpenSSL's built-in speed test.

Ask Xmodulo

Find answers to commonly asked Linux questions

<http://ask.xmodulo.com>

To measure AES algorithm speed without AES-NI acceleration:

```
$ openssl speed -elapsed aes-128-cbc
```

```
$  
$ openssl speed -elapsed aes-128-cbc ← non-EVP APIs  
You have chosen to measure elapsed time instead of user CPU time.  
Doing aes-128 cbc for 3s on 16 size blocks: 18025424 aes-128 cbc's in 3.00s  
Doing aes-128 cbc for 3s on 64 size blocks: 5062620 aes-128 cbc's in 3.00s  
Doing aes-128 cbc for 3s on 256 size blocks: 1302396 aes-128 cbc's in 3.00s  
Doing aes-128 cbc for 3s on 1024 size blocks: 328926 aes-128 cbc's in 3.00s  
Doing aes-128 cbc for 3s on 8192 size blocks: 40843 aes-128 cbc's in 3.00s  
OpenSSL 1.0.2g 1 Mar 2016  
built on: reproducible build, date unspecified  
options:bn(64,64) rc4(16x,int) des(idx,cisc,16,int) aes(partial) blowfish(idx)  
compiler: cc -I. -I.. -I../include -fPIC -DOPENSSL_PIC -DOPENSSL_THREADS -D_REE  
NTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -m64 -DL_ENDIAN -g -O2 -fstack-protector-strong  
-Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -Wl,-Bsymbolic  
c-functions -Wl,-z,relro -Wa,--noexecstack -Wall -DMD32_REG_T=int -DOPENSSL_IA32  
_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1  
ASM -DSHA256 ASM -DSHA512 ASM -DMD5_ASM -DAES_ASM -DVPAES_ASM -DBSAES_ASM -DWHIR  
LPOOL_ASM -DGHASH_ASM -DECP_NISTZ256_ASM  
The 'numbers' are in 1000s of bytes per second processed.  
type          16 bytes    64 bytes    256 bytes   1024 bytes   8192 bytes  
aes-128 cbc    96135.59k   108002.56k  111137.79k  112273.41k   111528.62k  
$
```

To measure AES algorithm speed with AES-NI acceleration (via EVP APIs):

```
$ openssl speed -elapsed -evp aes-128-cbc
```

Ask Xmodulo

Find answers to commonly asked Linux questions

<http://ask.xmodulo.com>

```
$ openssl speed -elapsed -evp aes-128-cbc ← EVP APIs
You have chosen to measure elapsed time instead of user CPU time.
Doing aes-128-cbc for 3s on 16 size blocks: 91959080 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 64 size blocks: 25068627 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 256 size blocks: 6399970 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 1024 size blocks: 1602626 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 8192 size blocks: 201168 aes-128-cbc's in 3.00s
OpenSSL 1.0.2g 1 Mar 2016
built on: reproducible build, date unspecified
options:bn(64,64) rc4(16x,int) des(idx,cisc,16,int) aes(partial) blowfish(idx)
compiler: cc -I. -I. -I./include -fPIC -DOPENSSL_PIC -DOPENSSL_THREADS -D_REE
NTRANT -D_DSO_DLFCN -DHAVE_DLFCN_H -m64 -DL_ENDIAN -g -O2 -fstack-protector-strong
-Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -Wl,-Bsymboli
c-functions -Wl,-z,relro -Wa,--noexecstack -Wall -DMD32_REG_T=int -DOPENSSL_IA32
_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1
_ASM -DSHA256_ASM -DSHA512_ASM -DMD5_ASM -DAES_ASM -DVPAES_ASM -DBSAES_ASM -DWHIR
LPOOL_ASM -DGHASH_ASM -DECP_NISTZ256_ASM
The 'numbers' are in 1000s of bytes per second processed.
type          16 bytes    64 bytes    256 bytes    1024 bytes    8192 bytes
aes-128-cbc   490448.43k  534797.38k  546130.77k  547029.67k   549322.75k
$
```

The above two example outputs show encryption rates for different block sizes. You can see that AES speed with AES-NI acceleration is about five times higher than non-acceleration. This confirms that AES-NI is enabled for OpenSSL. If OpenSSL cannot leverage AES-NI for any reason, two outputs would show the same performance.