Download

$Revision: 1.19 $

by Nicolas Bélan

1. Revisions

Version	Date	Who	What
Revision 0.1.x, cvs 1.14	2004/05/19	Nicolas Bélan <nicolas<at>open-firewall.org>	First version, released and pretty clear
Revision 0.1.2, cvs 1.17	2004/06/18	Nicolas Bélan <nicolas<at>open-firewall.org>	Change with CMake version
Revision 0.1.3, cvs current	2004/07/06	Nicolas Bélan <nicolas<at>open-firewall.org>	Open Firewall Log API and Formatters, daemon first draft

2. Quick informations

We are still working on 0.2.0 release of *open-firewall-core* module and *open-firewall-plugin* module.
We may release it soon. A major change in this release is the usage of CMake makefile system (www.cmake.org), and the definition and usage of the log API. For beginners with CMake, take a look at CMake and Openfirewall chapter. Take a look at release 0.1.2 and next for cmake preview.

You can get release 0.1.3 here.
The gpg signature of 0.1.3 release is :

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.5 (MingW32)

iD8DBQBBdnAJ7Z+vPzyCxIcRAqvtAJ9HNXlzbsQz9goVkmdjKL/b9D22/QCfX+XY
yjz7Ra7dqiOl0o0S+T/0VT4=
=DH1B
-----END PGP SIGNATURE-----
```

3. CVS access

You are able to browse our CVS repository using viewcvs :
open-firewall-core module

To checkout entire module, just hit:

```
you@yourhost $ CVSROOT=:pserver:anonymous@cvs.sf.net:/cvsroot/open-firewall
you@yourhost $ export CVSROOT
you@yourhost $ cvs login <hit enter when password is asked>
you@yourhost $ cvs -z3 co open-firewall-core
you@yourhost $ cvs -z3 co open-firewall-plugins
```

Detailled informations about CVS can be found here.
Available modules are:

open-firewall-core
Main module
open-firewall-plugins
Plugin module, use it to make modules
docs
Various documentation, including this web site

Available releases are:

HEAD
Main development branch
OF_RELEASE_0_2_0
tag of 0.2.0 next release
OF_RELEASE_0_1_3
tag of 0.1.3 release
OF_RELEASE_0_1_2
tag of 0.1.2 release
OF_RELEASE_0_1_1
tag of 0.1.1 release
OF_RELEASE_0_1_0
tag of 0.1.0 release

The next main release will be OF_RELEASE_0_2_0, but this is possible that intermediate releases (0.1.x) are provided before.

To retrieve a particular release, do:

```
you@yourhost $ cvs -z3 co -rTAG open-firewall-core
```

where TAG is one of 'HEAD', 'OF_RELEASE_0_1_0' and so on ...

4. The Open Firewall Archive OpenPGP signature

Files placed on the Open Firewall website are OpenPGP signed.

This signature can be used to prove that a file, which may have been obtained from a mirror site or other location, really originated from the Open Firewall website.

Before you can do this, you must gpg --import the key below. This is my key. This key is also available from most common PGP key servers, such as http://wwwkeys.pgp.net:11371/pks/lookup?op=get&search=0x3C82C487

To import it from the keyserver using GnuPG, do:

```
        $ gpg --keyserver wwwkeys.pgp.net --recv-keys 0x3C82C487
```

Using GnuPG, verifying a signature look like this:

```
        $ gpg --verify archive-version.tar.gz.asc archive-version.tar.gz
        ...
```

Unless you have taken explicit steps to build a trust path to the Open Firewall Archives Verification Key, you should expect to see a warning message after gpg has verified the signature. You should not be alarmed by this warning:

```
        Could not find a valid trust path to the key.
        Let's see whether we can assign some missing owner trust values.

        No path leading to one of our keys found.
```

```
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
```

5. Building Open-Firewall

5.1. Requirements

To build Open-Firewall core product, you need :

- Apache runtime library (APR), version 1.0.
- Apache runtime utility library (APU), version 1.0.
- CMake utility, version 2.0.2 or later (I use 2.1 development)

Referer to Build APR on Unix and Build APR on Win32.

5.2. CMake and Open-Firewall

I have used GNU automake/autoconf/autohead for first releases (0.1.x), but it is too complicated for me. In fact, it is working, but not enough user-friendly, and mainly, it can not generate windows native makefiles.

CMake can ! So, i have tested and adopted CMake utility for Open-Firewall. It has many good features, and it is working "not so bad".

For general description of CMake, take a look at cmake.org.

To install cmake, cmake installation guide.

5.3. Build targets

You can build Open-Firewall on UNIX or Windows platform.

FIXME (NB):

Only UNIX has the ability to reinject packet into TCP/IP stack ...

There are four targets available on Open-Firewall:

Debug
Debug mode, non optimized compiled code
Release
Release mode
RelWithDebInfo
Debug mode with optimized code
MinSizeRel
strip all code not used

To choose a particular target:

```
you@yourhost $ cmake -DCMAKE_BUILD_TYPE:STRING=Debug \
                     /your/path/to/open-firewall-core
```

Properties like CMAKE_BUILD_TYPE are propagated automatically to sub-modules, like open-firewall-plugins. So, building *core* in debug mode will affect how plugins are built too !

5.4. Building

5.4.1. Building on UNIX

After successfully installed cmake, you are able to build on UNIX.

Take a look at this important parameters:

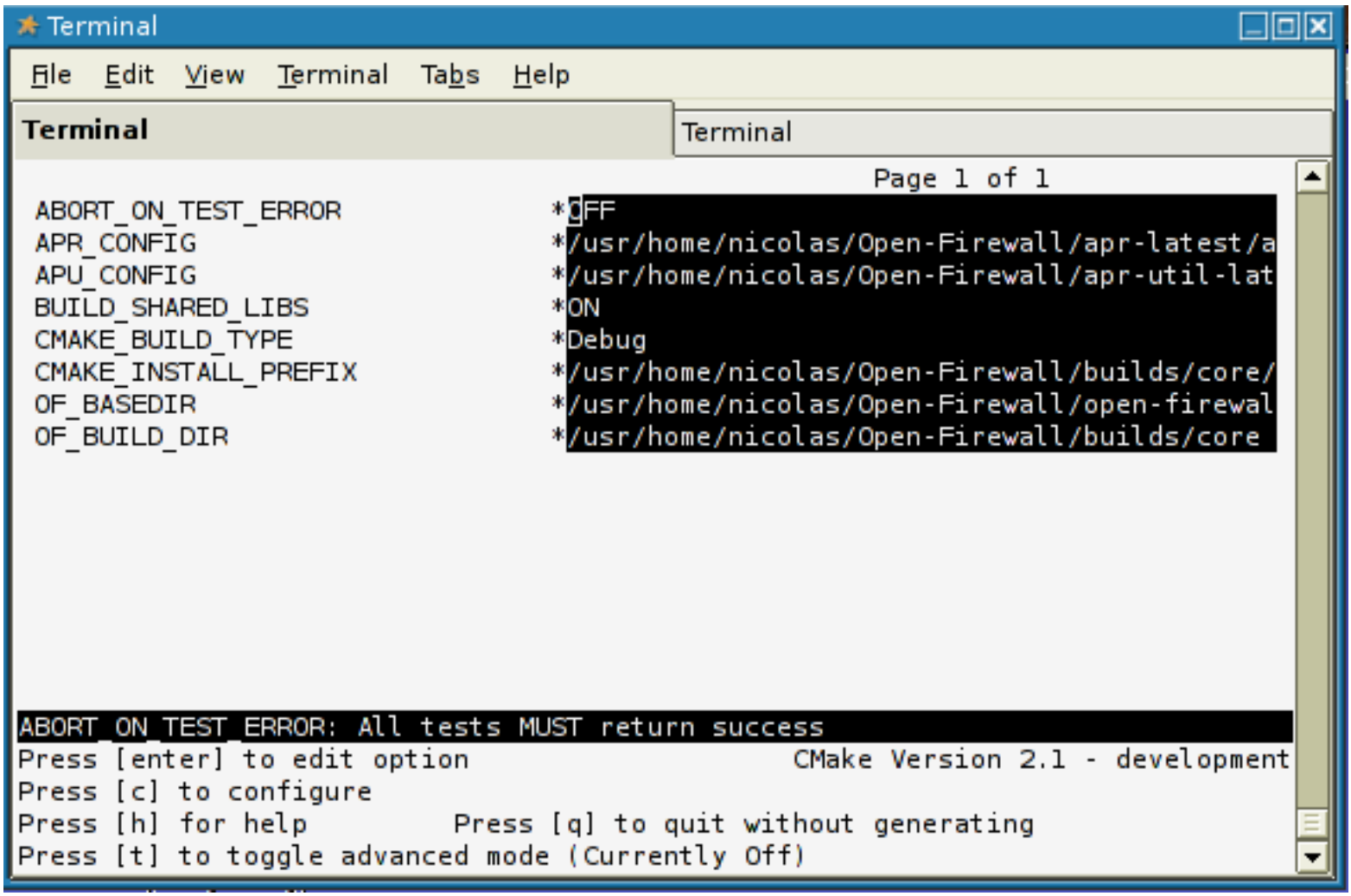
CMAKE_BUILD_TYPE
see Build targets
APR_CONFIG
Full path to apr-config shell scripts, in apr directory
APU_CONFIG
Full path to apu-config shell scripts, in apr-util directory
BUILD_SHARED_LIBS
Are we going to build .so (ON) or only static library linking (.a only). OFF means that APR/APR-UTIL are statically linked too!
CMAKE_BUILD_TYPE
see Build targets
OF_BASE_DIR
Location of open-firewall-core source directory
OF_BUILD_DIR
Location of open-firewall-core objects and binaries (often where you launched CMake)
ABORT_ON_TEST_ERROR
If ON, of_tests will call abort() is a test fails.
BUILD_DOCUMENTATION
If ON, cmake generate open-firewall-core.dox to enable doxygen in your source tree.

Note:
APR_CONFIG and APU_CONFIG should be set using full path names, not relative

You just have to do:

```
you@yourhost ~/builds $ /path/to/cmake/cmake \
                           -DAPR_CONFIG:STRING=/path/to/apr/apr-config \
                           -DAPU_CONFIG:STRING=/path/to/apr-util/apu-config \
                           -DCMAKE_BUILD_TYPE:STRING=Debug \
                           -DBUILD_SHARED_LIBS=ON
                           /path/to/open-firewall-core
you@yourhost ~/builds $ make
```

As on Windows, you can use *ccmake* instead of cmake (notice the 'c' at the beginning). This is a curses tool to set various parameters.



CMakeSetup for UNIX

Cache values are detailled here.

The first time CMake is started, no values are detailled.
Enter the correct path where source reside and where you want to build binaries, then hit 'c' (for configure).

Values are now on a red background: verify and adjust (if necessary) values, then re-hit 'c'.

You are now able to hit 'g' (for generate) to generate makefiles.

Currently, there are some known bugs with that procedure.

- If you do not install latests apr snapshosts, libtool let *.so into apr[-util]/.libs.
  You have to copy or link these libraries into apr source base directory (/path/to/snapshots/apr-latest/) :

```
me@host ~ $ cd /path/to/snapshots/apr-latest
me@host /path/to/snapshots/apr-latest $ ln -sf .libs/libapr-1.so.0
me@host /path/to/snapshots/apr-latest $ ln -sf .libs/libapr-1.0.so
```

- You have to do the same in apr-util library

To build:

```
you@yourhost ~/builds $ make
```

Files are generated into :

```
build-CMAKE_SYSTEM_NAME\bin\
build-CMAKE_SYSTEM_NAME\lib\
```

By default, if you do not specify CMAKE_BUILD_TYPE value, it is set to "Debug".

5.4.1.1. FreeBSD compilation sample

Options to make it compile on a FreeBSD 5.2.x

Note:
We are using uninstalled snapshots

```
me@myhost ~/builds $ /usr/home/nicolas/TESTS/CMake/bin/cmake \
                     -DAPR_CONFIG:STRING=${HOME}/Open-Firewall/apr-latest/apr-config \
                     -DAPU_CONFIG:STRING=${HOME}/Open-Firewall/apr-util-latest/apu-config \
                     -DCMAKE_BUILD_TYPE:STRING=Debug \
                     -DBUILD_SHARED_LIBS=ON
-- Check for working C compiler: gcc
-- Check for working C compiler: gcc -- works
-- Check for working CXX compiler: c++
-- Check for working CXX compiler: c++ -- works
-- Looking for include files HAVE_STDLIB_H
-- Looking for include files HAVE_STDLIB_H - found
-- Looking for include files HAVE_STDARG_H
-- Looking for include files HAVE_STDARG_H - found
-- Looking for include files HAVE_STRING_H
-- Looking for include files HAVE_STRING_H - found
-- Looking for include files HAVE_FLOAT_H
-- Looking for include files HAVE_FLOAT_H - found
-- Looking for include files HAVE_DLFCN_H
-- Looking for include files HAVE_DLFCN_H - found
-- Looking for include files HAVE_INTTYPES_H
-- Looking for include files HAVE_INTTYPES_H - found
-- Looking for include files HAVE_MEMORY_H
-- Looking for include files HAVE_MEMORY_H - found
-- Looking for include files HAVE_NETINET_IN_SYSTM_H
-- Looking for include files HAVE_NETINET_IN_SYSTM_H - found
-- Looking for include files HAVE_NETINET_IP_H
-- Looking for include files HAVE_NETINET_IP_H - found
-- Looking for include files HAVE_NETINET_IP_ICMP_H
-- Looking for include files HAVE_NETINET_IP_ICMP_H - found
-- Looking for include files HAVE_STDDEF_H
-- Looking for include files HAVE_STDDEF_H - found
-- Looking for include files HAVE_SYS_CDEFS_H
-- Looking for include files HAVE_SYS_CDEFS_H - found
-- Looking for include files HAVE_SYS_STAT_H
-- Looking for include files HAVE_SYS_STAT_H - found
-- Looking for include files HAVE_WINDOWS_H
-- Looking for include files HAVE_WINDOWS_H - not found.
-- Looking for include files HAVE_WINSOCK2_H
-- Looking for include files HAVE_WINSOCK2_H - not found.
-- Looking for chroot
-- Looking for chroot - found
-- Looking for setgid
-- Looking for setgid - found
-- Looking for setuid
-- Looking for setuid - found
-- Looking for _getcwd
-- Looking for _getcwd - not found
-- Looking for _getwd
-- Looking for _getwd - not found
-- Looking for getcwd
-- Looking for getcwd - found
-- Looking for getwd
```

```
-- Looking for getwd - found
-- Looking for chdir
-- Looking for chdir - found
-- Looking for _chdir
-- Looking for _chdir - found
-- Looking for ttyslot
-- Looking for ttyslot - found
-- Check if the system is big endian
-- Check if the system is big endian - little endian
-- Configuring done
-- Generating done

me@myhost ~/builds $ make
```

Files are generated into :

```
build-FreeBSD\bin\
build-FreeBSD\lib\
```

5.4.2. Building on Win32

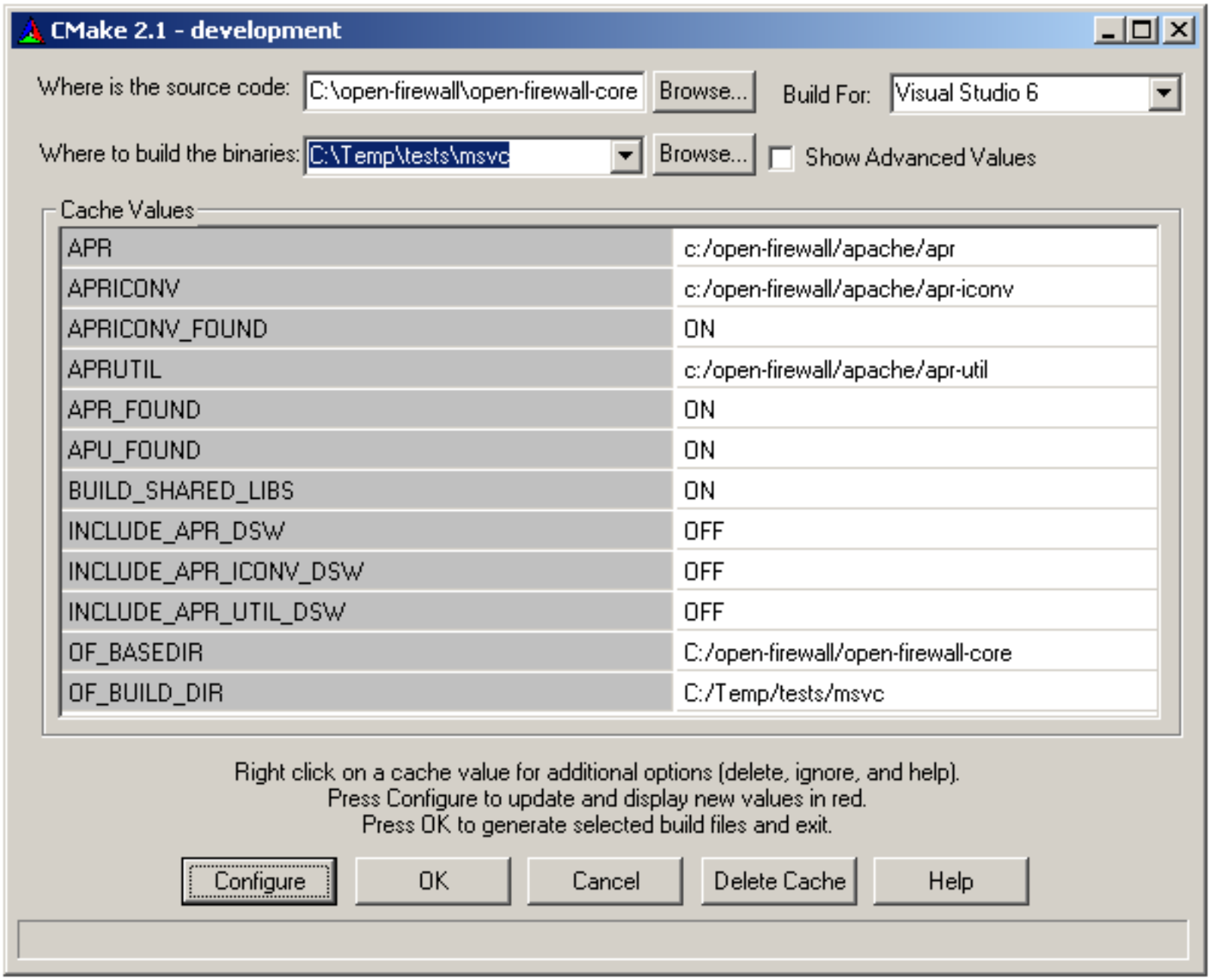
After successfully installed cmake, you are able to build on Windows.

Actually, you can compile with MSVC (6, 7 and 7.1), or using nmake and MS Visual C++ Toolkit 2003

5.4.2.1. Using MSVC

It is the easiest way: cmake will generate something called *Open-Firewall-Core.dsw*.

To configure, launch CMakeSetup in CMake bin directory (probably c:\Program Files\CMake\bin\CMakeSetup.exe). You see:



CMakeSetup for Win32

Cache values are detailled here.

The first time CMake is started, no values are detailled.
Browse where the source is and where you want to build binaries, then hit 'Configure'.

Values are now on a red background: verify and adjust (if necessary) values, then re-hit 'Configure'.

You are now able to click on 'Ok' to generate .dsw file.

To build:

```
c:\...\msvc> msdev Open-Firewall-Core.dsw /MAKE "ALL_BUILD - Win32 Debug"
```

You can also enter :

```
c:\...\msvc> msdev Open-Firewall-Core.dsw /MAKE "ALL_BUILD - Win32 Release"
c:\...\msvc> msdev Open-Firewall-Core.dsw /MAKE "ALL_BUILD - Win32 RelWithDebInfo"
c:\...\msvc> msdev Open-Firewall-Core.dsw /MAKE "ALL_BUILD - Win32 MinSizeRel"
```

Files are generated into :

```
build-Windows\bin\${CMAKE_BUILD_TYPE}
build-Windows\lib\${CMAKE_BUILD_TYPE}
```

By default, if you do not specify CMAKE_BUILD_TYPE value, it is set to "Debug".

5.4.2.2. Using Free tools

Microsoft released free compiler suite for Windows, called Microsoft Visual C++ toolkit 2003. It contains compiler and linker, and minimal headers. It does not contains nmake, but you can find it at ftp://ftp.microsoft.com/Softlib/MSLFILES/NMAKE15.EXE. There is no information about nmake redistribution or licensing, but it seems to be ok to get it.

You should get Microsoft SDK too and install it.

Then, configure open-firewall (CMakeSetup works too, but as you see, command line is working !):

```
c:\...\> mkdir nmake
c:\...\> cd nmake
c:\...\nmake> cmake -i -G"NMake Makefiles" \
                -DCMAKE_BUILD_TYPE:STRING=Debug \
                c:\path\to\open-firewall-core
-- Check for working C compiler: cl
-- Check for working C compiler: cl -- works
-- Check for working CXX compiler: cl
-- Check for working CXX compiler: cl -- works
-- Looking for include files HAVE_STDLIB_H
-- Looking for include files HAVE_STDLIB_H - found
-- Looking for include files HAVE_STDARG_H
-- Looking for include files HAVE_STDARG_H - found
-- Looking for include files HAVE_STRING_H
-- Looking for include files HAVE_STRING_H - found
-- Looking for include files HAVE_FLOAT_H
-- Looking for include files HAVE_FLOAT_H - found
-- Looking for include files HAVE_DLFCN_H
-- Looking for include files HAVE_DLFCN_H - not found.
-- Looking for include files HAVE_INTTYPES_H
-- Looking for include files HAVE_INTTYPES_H - not found.
-- Looking for include files HAVE_MEMORY_H
-- Looking for include files HAVE_MEMORY_H - found
-- Looking for include files HAVE_NETINET_IN_SYSTM_H
-- Looking for include files HAVE_NETINET_IN_SYSTM_H - not found.
-- Looking for include files HAVE_NETINET_IP_H
-- Looking for include files HAVE_NETINET_IP_H - not found.
-- Looking for include files HAVE_NETINET_IP_ICMP_H
-- Looking for include files HAVE_NETINET_IP_ICMP_H - not found.
-- Looking for include files HAVE_STDDEF_H
-- Looking for include files HAVE_STDDEF_H - found
-- Looking for include files HAVE_SYS_CDEFS_H
-- Looking for include files HAVE_SYS_CDEFS_H - not found.
-- Looking for include files HAVE_SYS_STAT_H
-- Looking for include files HAVE_SYS_STAT_H - found
-- Looking for include files HAVE_WINDOWS_H
-- Looking for include files HAVE_WINDOWS_H - found
-- Looking for include files HAVE_WINSOCK2_H
-- Looking for include files HAVE_WINSOCK2_H - found
-- Looking for chroot
-- Looking for chroot - not found
-- Looking for setgid
```

```
-- Looking for setgid - not found
-- Looking for setuid
-- Looking for setuid - not found
-- Looking for _getcwd
-- Looking for _getcwd - found
-- Looking for _getwd
-- Looking for _getwd - not found
-- Looking for getcwd
-- Looking for getcwd - found
-- Looking for getwd
-- Looking for getwd - not found
-- Looking for chdir
-- Looking for chdir - found
-- Looking for _chdir
-- Looking for _chdir - found
-- Looking for ttyslot
-- Looking for ttyslot - not found
-- Check if the system is big endian
-- Check if the system is big endian - little endian
-- Configuring done
-- Generating done
c:\...\nmake> nmake all
```

Files are generated into :

```
build-Windows\bin
build-Windows\lib
```

5.4.2.3. Win32 Cache values

Take a look at this important parameters (in cmake order):

APR
Location of apr 1.0 source directory
APRICONV
Location of apr-iconv 1.0 source directory
APRICONV_FOUND
Does this system has apr-iconv library ?
APRUTIL
Location of apr-util 1.0 source directory
APR_FOUND
Does this system has apr library ? "OFF" value indicates that Open-Firewall should not be built !
APU_FOUND
Does this system has apr-util library ? "OFF" value indicates that Open-Firewall should not be built !
BUILD_SHARED_LIBS
Are we going to build .dll and .lib (ON) or only static library linking (.lib only). OFF means that APR/APR-UTIL are statically linked too!
CMAKE_BUILD_TYPE
see Build targets
INCLUDE_APR_DSW
If ON, Open-Firewall-Core.dsw will load apr.dsp. This generates error at msvc startup that you can safely ignore.
INCLUDE_APR_ICON_DSW
If ON, Open-Firewall-Core.dsw will load apriconv.dsp. This generates error at msvc startup that you can safely ignore.
INCLUDE_APR_UTIL_DSW
If ON, Open-Firewall-Core.dsw will load aprutil.dsp. This generates error at msvc startup that you can safely

ignore.
OF_BASE_DIR
Location of open-firewall-core source directory
OF_BUILD_DIR
Location of open-firewall-core.dsw is built (and where Open-Firewall-Core.dsw is generated)
ABORT_ON_TEST_ERROR
If ON, of_tests.exe will call abort() is a test fails.

5.4.3. Builing on other system

6. Verifying Open-Firewall

You may verify the (minimal) stability of Open-Firewall with of_tests.
It is a little binary which load unitary tests of open-firewall, executes them and reports results.
For example, in pre-0.2.0 release on wind*ws:

```
c:\...\core> build-Windows\bin\of_tests.exe
Test started at 03:16:24 PM,
===================================
loading tests from C:/Temp/tests/msvc/build-Windows/lib/Debug

Suite: Test utils
Suite: Test plugin
Suite: Test network
Suite: Test log
Suite: Test config
Running 'Test config'
================================
test_config_of_config_get_plugin.c:57: ASSERTION FAILED : s(120003) != APR_SUCCESS
Running 'Test log'
================================
Running 'Test network'
================================
Running 'Test plugin'
================================
test_plugin_of_plugin_get_copyright.c:53: ASSERTION FAILED : ptr bogus bogus %p= NULL
Running 'Test utils'
================================
test_utils_of_utils_parse_command_line.c:92: ASSERTION FAILED : s1(70012) != s2(0)
=========================================
Totals
------
  # tests          : 78
  # run            : 59
  # errors         : 3
  # not implemented: 54
  # success        : 2
=========================================
```