Ruby - Bug #20520

_FORTIFY_SOURCE=3 is not correctly respected

06/03/2024 10:33 AM - vo.x (Vit Ondruch)

Status:	Closed		
Priority:	Normal		
Assignee:	kjtsanaktsidis (KJ Tsanaktsidis)		
Target version:			
ruby -v:	ruby 3.3.1 (2024-04-23 revision c56cd86388) [aarch64-linux]	**Backport:**	3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN

Description

In Fedora, we are using following compilation options:

```
+ CFLAGS='-O2 -flto=auto -ffat-lto-objects -fexceptions -g -grecord-gcc-switches -pipe -Wall -Werr
or=format-security -Wp,-U_FORTIFY_SOURCE,-D_FORTIFY_SOURCE=3 -Wp,-D_GLIBCXX_ASSERTIONS -specs=/usr
/lib/rpm/redhat/redhat-hardened-cc1 -fstack-protector-strong -specs=/usr/lib/rpm/redhat/redhat-ann
obin-cc1  -mbranch-protection=standard -fasynchronous-unwind-tables -fstack-clash-protection -fno-
omit-frame-pointer -mno-omit-leaf-frame-pointer '
```

Please note that as of recently, there is included _FORTIFY_SOURCE=3. The problem is, that Ruby doing its configuration check:

```
checking whether -O3 -D_FORTIFY_SOURCE=2 is accepted as CFLAGS... yes
```

Includes another variant of _FORTIFY_SOURCE=2 into XFLAGS:

```
$ /usr/bin/make -O -j12 V=1 VERBOSE=1 'COPY=cp -p' -C redhat-linux-build
make: Entering directory '/builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/redhat-linux-build'
 BASERUBY = /builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/tool/missing-baseruby.bat
 CC = gcc
 LD = ld
 LDSHARED = gcc -shared
 CFLAGS = -O2 -flto=auto -ffat-lto-objects -fexceptions -g -grecord-gcc-switches -pipe -Wall -Werr
or=format-security -Wp,-U_FORTIFY_SOURCE,-D_FORTIFY_SOURCE=3 -Wp,-D_GLIBCXX_ASSERTIONS -specs=/usr
/lib/rpm/redhat/redhat-hardened-cc1 -fstack-protector-strong -specs=/usr/lib/rpm/redhat/redhat-ann
obin-cc1  -mbranch-protection=standard -fasynchronous-unwind-tables -fstack-clash-protection -fno-
omit-frame-pointer -mno-omit-leaf-frame-pointer  -fPIC
 XCFLAGS = -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -fstack-protector-strong -mbranch-protection=stan
dard -fno-strict-overflow -fvisibility=hidden -fexcess-precision=standard -DRUBY_EXPORT -I. -I.ext
/include/aarch64-linux -I/builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/include -I/builddir/bui
ld/BUILD/ruby-3.3.1-build/ruby-3.3.1 -I/builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/prism -I/
builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/enc/unicode/15.0.0
 CPPFLAGS =
 DLDFLAGS = -Wl,-z,relro -Wl,--as-needed   -Wl,-z,now -specs=/usr/lib/rpm/redhat/redhat-hardened-l
d -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1  -Wl,--build-id=sha1   -Wl,-soname,libruby.so.3.3
-fstack-protector-strong
 SOLIBS = -lz -lrt -lrt -lgmp -ldl -lcrypt -lm -lpthread
 LANG = C.UTF-8
 LC_ALL =
 LC_CTYPE =
 MFLAGS = -j12 -Otarget --jobserver-auth=fifo:/tmp/GMfifo10279 --sync-mutex=fnm:/tmp/GmbGoenG
 RUSTC = rustc
 YJIT_RUSTC_ARGS = --crate-name=yjit --crate-type=staticlib --edition=2021 -g -C lto=thin -C opt-l
evel=3 -C overflow-checks=on '--out-dir=/builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/redhat-l
inux-build/yjit/target/release/' /builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/yjit/src/lib.rs
gcc (GCC) 14.1.1 20240522 (Red Hat 14.1.1-4)
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
make: Leaving directory '/builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/redhat-linux-build'
make: Entering directory '/builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/redhat-linux-build'
gcc -O2 -flto=auto -ffat-lto-objects -fexceptions -g -grecord-gcc-switches -pipe -Wall -Werror=for
mat-security -Wp,-U_FORTIFY_SOURCE,-D_FORTIFY_SOURCE=3 -Wp,-D_GLIBCXX_ASSERTIONS -specs=/usr/lib/r
```

```
pm/redhat/redhat-hardened-cc1 -fstack-protector-strong -specs=/usr/lib/rpm/redhat/redhat-annobin-c
c1  -mbranch-protection=standard -fasynchronous-unwind-tables -fstack-clash-protection -fno-omit-f
rame-pointer -mno-omit-leaf-frame-pointer  -fPIC  -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -fstack-pr
otector-strong -mbranch-protection=standard -fno-strict-overflow -fvisibility=hidden -fexcess-prec
ision=standard -DRUBY_EXPORT -I. -I.ext/include/aarch64-linux -I/builddir/build/BUILD/ruby-3.3.1-b
uild/ruby-3.3.1/include -I/builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1 -I/builddir/build/BUIL
D/ruby-3.3.1-build/ruby-3.3.1/prism -I/builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/enc/unicod
e/15.0.0      -o dmyext.o -c /builddir/build/BUILD/ruby-3.3.1-build/ruby-3.3.1/dmyext.c

... snip ...
```

Associated revisions

Revision 0ccb80d6bf57cd6e79ad622c024d3d0940ec6f3b - 06/11/2024 10:48 AM - KJ Tsanaktsidis

Extract hardening CFLAGS to a special $hardenflags variable

This changes the automatic detection of -fstack-protector,
-D_FORTIFY_SOURCE, and -mbranch-protection to write to $hardenflags
instead of $XCFLAGS. The definition of $cflags is changed to
"$hardenflags $orig_cflags $optflags $debugflags $warnflags" to match.

Furthermore, these flags are *prepended* to $hardenflags, rather than
appended.

The implications of doing this are as follows:

- If a CRuby builder specifies cflags="-mbranch-protection=foobar" at the ./configure script, and the configure script detects that -mbranch-protection=pac-ret is accepted, then GCC will be invoked as "gcc -mbranch-protection=pac-ret -mbranch-protection=foobar". Since the last flags take precedence, that means that user-supplied values of these flags in $cflags will take priority.
- Likewise, if a CRuby builder explicitly specifies "hardenflags=-mbranch-protection=foobar", because we *prepend* to $hardenflags in our autoconf script, we will still invoke GCC as "gcc -mbranch-protection=pac-ret -mbranch-protection=foobar".
- If a CRuby builder specifies CFLAGS="..." at the configure line, automatic detection of hardening flags is ignored as before.
- C extensions will *also* be built with hardening flags now as well (this was not the case by default before because the detected flags went into $XCFLAGS).

Additionally, as part of this work, I changed how the detection of
PAC/BTI in Context.S works. Rather than appending the autodetected
option to ASFLAGS, we simply compile a set of test programs with the
actual CFLAGS in use to determine what PAC/BTI settings were actually
chosen by the builder. Context.S is made aware of these choices through
some custom macros.

The result of this work is that:

- Ruby will continue to choose some sensible defaults for hardening options for the C compiler
- Distributors are able to specify CFLAGS that are consistent with their distribution and override these defaults
- Context.S will react to whatever -mbranch-protection is actually in use, not what was autodetected
- Extensions get built with hardening flags too.

[Bug #20154]
[Bug #20520]

Revision 0ccb80d6bf57cd6e79ad622c024d3d0940ec6f3b - 06/11/2024 10:48 AM - KJ Tsanaktsidis

Extract hardening CFLAGS to a special $hardenflags variable

This changes the automatic detection of -fstack-protector,
-D_FORTIFY_SOURCE, and -mbranch-protection to write to $hardenflags
instead of $XCFLAGS. The definition of $cflags is changed to
"$hardenflags $orig_cflags $optflags $debugflags $warnflags" to match.

Furthermore, these flags are *prepended* to $hardenflags, rather than
appended.

The implications of doing this are as follows:

- If a CRuby builder specifies cflags="-mbranch-protection=foobar" at the ./configure script, and the configure script detects that -mbranch-protection=pac-ret is accepted, then GCC will be invoked as "gcc -mbranch-protection=pac-ret -mbranch-protection=foobar". Since the last flags take precedence, that means that user-supplied values of these flags in $cflags will take priority.
- Likewise, if a CRuby builder explicitly specifies "hardenflags=-mbranch-protection=foobar", because we *prepend* to $hardenflags in our autoconf script, we will still invoke GCC as "gcc -mbranch-protection=pac-ret -mbranch-protection=foobar".
- If a CRuby builder specifies CFLAGS="..." at the configure line, automatic detection of hardening flags is ignored as before.
- C extensions will *also* be built with hardening flags now as well (this was not the case by default before because the detected flags went into $XCFLAGS).

Additionally, as part of this work, I changed how the detection of PAC/BTI in Context.S works. Rather than appending the autodetected option to ASFLAGS, we simply compile a set of test programs with the actual CFLAGS in use to determine what PAC/BTI settings were actually chosen by the builder. Context.S is made aware of these choices through some custom macros.

The result of this work is that:

- Ruby will continue to choose some sensible defaults for hardening options for the C compiler
- Distributors are able to specify CFLAGS that are consistent with their distribution and override these defaults
- Context.S will react to whatever -mbranch-protection is actually in use, not what was autodetected
- Extensions get built with hardening flags too.

[Bug #20154]
[Bug #20520]

Revision 0ccb80d6 - 06/11/2024 10:48 AM - KJ Tsanaktsidis

Extract hardening CFLAGS to a special $hardenflags variable

This changes the automatic detection of -fstack-protector, -D_FORTIFY_SOURCE, and -mbranch-protection to write to $hardenflags instead of $XCFLAGS. The definition of $cflags is changed to "$hardenflags $orig_cflags $optflags $debugflags $warnflags" to match.

Furthermore, these flags are *prepended* to $hardenflags, rather than appended.

The implications of doing this are as follows:

- If a CRuby builder specifies cflags="-mbranch-protection=foobar" at the ./configure script, and the configure script detects that -mbranch-protection=pac-ret is accepted, then GCC will be invoked as "gcc -mbranch-protection=pac-ret -mbranch-protection=foobar". Since the last flags take precedence, that means that user-supplied values of these flags in $cflags will take priority.
- Likewise, if a CRuby builder explicitly specifies "hardenflags=-mbranch-protection=foobar", because we *prepend* to $hardenflags in our autoconf script, we will still invoke GCC as "gcc -mbranch-protection=pac-ret -mbranch-protection=foobar".
- If a CRuby builder specifies CFLAGS="..." at the configure line, automatic detection of hardening flags is ignored as before.
- C extensions will *also* be built with hardening flags now as well (this was not the case by default before because the detected flags went into $XCFLAGS).

Additionally, as part of this work, I changed how the detection of PAC/BTI in Context.S works. Rather than appending the autodetected option to ASFLAGS, we simply compile a set of test programs with the actual CFLAGS in use to determine what PAC/BTI settings were actually chosen by the builder. Context.S is made aware of these choices through some custom macros.

The result of this work is that:

- Ruby will continue to choose some sensible defaults for hardening options for the C compiler
- Distributors are able to specify CFLAGS that are consistent with their distribution and override these defaults
- Context.S will react to whatever -mbranch-protection is actually in use, not what was autodetected
- Extensions get built with hardening flags too.

[Bug #20154]
[Bug #20520]

History

#1 - 06/03/2024 10:34 AM - vo.x (Vit Ondruch)

I'll try to workaround this by --disable-fortify-source, I hope it won't have another side effects.

#2 - 06/07/2024 03:30 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Assignee set to kjtsanaktsidis (KJ Tsanaktsidis)

#3 - 06/11/2024 11:05 AM - Anonymous

- Status changed from Open to Closed

Applied in changeset git|0ccb80d6bf57cd6e79ad622c024d3d0940ec6f3b.

Extract hardening CFLAGS to a special $hardenflags variable

This changes the automatic detection of -fstack-protector, -D_FORTIFY_SOURCE, and -mbranch-protection to write to $hardenflags instead of $XCFLAGS. The definition of $cflags is changed to "$hardenflags $orig_cflags $optflags $debugflags $warnflags" to match.

Furthermore, these flags are *prepended* to $hardenflags, rather than appended.

The implications of doing this are as follows:

- If a CRuby builder specifies cflags="-mbranch-protection=foobar" at the ./configure script, and the configure script detects that -mbranch-protection=pac-ret is accepted, then GCC will be invoked as "gcc -mbranch-protection=pac-ret -mbranch-protection=foobar". Since the last flags take precedence, that means that user-supplied values of these flags in $cflags will take priority.
- Likewise, if a CRuby builder explicitly specifies "hardenflags=-mbranch-protection=foobar", because we *prepend* to $hardenflags in our autoconf script, we will still invoke GCC as "gcc -mbranch-protection=pac-ret -mbranch-protection=foobar".
- If a CRuby builder specifies CFLAGS="..." at the configure line, automatic detection of hardening flags is ignored as before.
- C extensions will *also* be built with hardening flags now as well (this was not the case by default before because the detected flags went into $XCFLAGS).

Additionally, as part of this work, I changed how the detection of PAC/BTI in Context.S works. Rather than appending the autodetected option to ASFLAGS, we simply compile a set of test programs with the actual CFLAGS in use to determine what PAC/BTI settings were actually chosen by the builder. Context.S is made aware of these choices through some custom macros.

The result of this work is that:

- Ruby will continue to choose some sensible defaults for hardening options for the C compiler
- Distributors are able to specify CFLAGS that are consistent with their distribution and override these defaults
- Context.S will react to whatever -mbranch-protection is actually in use, not what was autodetected
- Extensions get built with hardening flags too.

[Bug #20154]
[Bug #20520]