```
From a258c17c6937f79529c8319a829310e09cdbd216 Mon Sep 17 00:00:00 2001
From: Reini Urban <rurban@cpanel.net>
Date: Wed, 25 Jan 2017 11:27:07 -0600
Subject: [PATCH] Storable: protect against classname len overflow

name lenghts can be max. I32 (hek_len), but are read from
system size integers. e.g. -1 you can cause a malloc fail => exit.

Even worse len > LG_BLESS didn't detect that so you could overwrite
the stack by malcrafted Storable files.
Found out by JD. RT #130635. No CVE since p5p believes local Storable
files are not exploitable.

(cherry picked from commit c82e80f6d88891738e2b5329723606b48347fa31)

Conflicts:
        .git-rr-cache
        Porting/Maintainers.pl
        dist/Storable/Storable.xs
        pod/perlcdelta.pod
        t/porting/customized.dat
---
 dist/Storable/ChangeLog    | 27 ++++++++++++++++++++++++++
 dist/Storable/README       | 16 +++++++---------
 dist/Storable/Storable.xs  | 35 ++++++++++++++++++++++++++++++++-------------
 dist/Storable/t/retrieve.t | 31 +++++++++++++++++++++++++++++++-
 4 files changed, 86 insertions(+), 23 deletions(-)

diff --git a/dist/Storable/ChangeLog b/dist/Storable/ChangeLog
index 038a0e791df0..11f6eee7011f 100644
--- a/dist/Storable/ChangeLog
+++ b/dist/Storable/ChangeLog
@@ -1,3 +1,30 @@
+Wed Jan 25 11:27:07 2017 -0600 Reini Urban <rurban@cpanel.net>
+    Version 3.05c
+
+        * Protect against classname len overflow on the stack
+        and 2x on the heap with retrieve_bless and retrieve_hook.
+        A serious security issue with malcrafted storable files or buffers,
+        but p5p accepts no CVE on Storable attacks. See RT #130635 (detected by JD).
+        * Fix NULL ptr SEGVs with retrieve_code and retrieve_other.
+        See RT #130098(JD)
+        * Fix the remaining 2-arg open calls
+
+Sat Jan 7 09:01:29 2017 +0100 Reini Urban <rurban@cpanel.net>
+    Version 3.04c
+
+        * fix printf types and warnings, esp. for 32bit use64bitint
+        * Change sv_setpvn(…, "…", …) to sv_setpvs(…, "…")
+
+Tue Jul 26 11:49:33 2016 +1000 Tony Cook <tony@develop-help.com>
+    Version 3.03c
+
+        * remove . from @INC when loading optional modules
+
+Sun Nov 20 18:06:45 2016 +0100 Reini Urban <rurban@cpanel.net>
+    Version 3.02c
+
+        * Fix -Wc++11-compat warnings, fix -Wchar-subscripts
+
 Fri Sep 16 01:32:59 2016 +0200 Reini Urban <rurban@cpanel.net>
     Version 3.01c

diff --git a/dist/Storable/README b/dist/Storable/README
```

```
 index c70b1003436f..f2fca638f119 100644
 --- a/dist/Storable/README
 +++ b/dist/Storable/README
 @@ -1,7 +1,7 @@
 -                          Storable 3.00
 +                          Storable 3.05c
                 Copyright (c) 1995-2000, Raphael Manfredi
                 Copyright (c) 2001-2004, Larry Wall
 -               Copyright (c) 2016, cPanel Inc
 +               Copyright (c) 2016,2017 cPanel Inc

  ----------------------------------------------------------------------
      This program is free software; you can redistribute it and/or modify
 @@ -48,7 +48,7 @@ To compile this extension, run:
  There is an embedded POD manual page in Storable.pm.

  Storable was written by Raphael Manfredi <Raphael_Manfredi@pobox.com>
 -Maintenance is now done by the perl5-porters <perl5-porters@perl.org>
 +Maintenance is now done by cperl, https://github.com/perl11/cperl

  Please e-mail us with problems, bug fixes, comments and complaints,
  although if you have complements you should send them to Raphael.
 @@ -69,9 +69,10 @@ Thanks to (in chronological order):
      Marc Lehmann <pcg@opengroup.org>
      Justin Banks <justinb@wamnet.com>
      Jarkko Hietaniemi <jhi@iki.fi> (AGAIN, as perl 5.7.0 Pumpkin!)
 -    Todd Rinaldo <toddr@cpanel.net>, JD Lightsey <jd@cpanel.net>
 -      for optional disabling tie and bless for increased security
 -    Reini Urban <rurban@cpanel.net> for the 3.00 >2G support and rewrite
 +    Todd Rinaldo <toddr@cpanel.net> and JD Lightsey <jd@cpanel.net>
 +      for optional disabling tie and bless for increased security.
 +    Reini Urban <rurban@cpanel.net> for the 3.0x >2G support and rewrite
 +    JD Lightsey <jd@cpanel.net>

  for their contributions.

 @@ -108,6 +109,3 @@ bring you this Storable release:
      Tim Bunce <Tim.Bunce@pobox.com>
      VMSperlers
      Yitzchak Scott-Thoennes <sthoenna@efn.org>
 -
 -If I've missed you out, please accept my apologies, and e-mail your
 -patch to perl5-porters@perl.org.
 diff --git a/dist/Storable/Storable.xs b/dist/Storable/Storable.xs
 index 3e4730950311..9a28a76180b9 100644
 --- a/dist/Storable/Storable.xs
 +++ b/dist/Storable/Storable.xs
 @@ -1105,7 +1105,7 @@ static const char byteorderstr_56[] = {BYTEORDER_BYTES_56, 0};
  #define SAFEPVREAD(x,y,z)                                       \
         STMT_START {                                            \
                 if (!cxt->fio)                                  \
 -                       MBUF_SAFEPVREAD(x,y,z);         \
 +                       MBUF_SAFEPVREAD(x,y,z);                 \
                 else if (PerlIO_read(cxt->fio, x, y) != y) {    \
                         Safefree(z);                            \
                         return (SV *) 0;                        \
 @@ -3672,8 +3672,8 @@ static int store_blessed(
         HV *pkg)
  {
         SV *hook;
 -       I32 len;
         char *classname;
 +       I32 len;
         I32 classnum;
```

```
        TRACEME(("store_blessed, type %d, class \"%s\"", type, HvNAME_get(pkg)));
@@ -3724,7 +3724,7 @@ static int store_blessed(
                } else {
                        unsigned char flag = (unsigned char) 0x80;
                        PUTMARK(flag);
-                       WLEN(len);                                      /* Don't BER-encode,
this should be rare */
+                       WLEN(len);                              /* Don't BER-encode, this
should be rare */
                }
                WRITE(classname, len);                          /* Final \0 is omitted */
        }
@@ -4325,11 +4325,18 @@ static SV *retrieve_blessed(pTHX_ stcxt_t *cxt, const char *cname)
        GETMARK(len);                   /* Length coded on a single char? */
        if (len & 0x80) {
                RLEN(len);
-               TRACEME(("** allocating %d bytes for class name", (int)len+1));
+               TRACEME(("** allocating %ld bytes for class name", (long)len+1));
+               if (len > I32_MAX)
+                        CROAK(("Corrupted classname length %lu", (long)len));
+                PL_nomemok = TRUE; /* handle error by ourselves */
                New(10003, classname, len+1, char);
+                PL_nomemok = FALSE;
+                if (!classname)
+                    CROAK(("Out of memory with len %ld", (long)len));
+                PL_nomemok = FALSE;
                malloced_classname = classname;
        }
-       SAFEPVREAD(classname, (SSize_t)len, malloced_classname);
+       SAFEPVREAD(classname, (I32)len, malloced_classname);
        classname[len] = '\0';          /* Mark string end */

        /*
@@ -4510,17 +4517,19 @@ static SV *retrieve_hook(pTHX_ stcxt_t *cxt, const char *cname)
                else
                        GETMARK(len);

-               if (len > I32_MAX) {
-                       CROAK(("Corrupted classname length"));
-               }
-
-               if (len > LG_BLESS) {
-                        TRACEME(("** allocating %d bytes for class name", (int)len+1));
+                TRACEME(("** allocating %ld bytes for class name", (long)len+1));
+               if (len > I32_MAX) /* security */
+                        CROAK(("Corrupted classname length %lu", (long)len));
+                else if (len > LG_BLESS) { /* security: signed len */
+                        PL_nomemok = TRUE; /* handle error by ourselves */
                        New(10003, classname, len+1, char);
+                        PL_nomemok = FALSE;
+                        if (!classname)
+                            CROAK(("Out of memory with len %u", (unsigned)len+1));
                        malloced_classname = classname;
                }

-               SAFEPVREAD(classname, (SSize_t)len, malloced_classname);
+               SAFEPVREAD(classname, (I32)len, malloced_classname);
                classname[len] = '\0';          /* Mark string end */

                /*
@@ -6050,7 +6059,7 @@ static SV *retrieve_flag_hash(pTHX_ stcxt_t *cxt, const char *cname)
 static SV *retrieve_code(pTHX_ stcxt_t *cxt, const char *cname)
 {
 #if PERL_VERSION < 6
-    CROAK(("retrieve_code does not work with perl 5.005 or less\n"));
```

```
+       CROAK(("retrieve_code does not work with perl 5.005 or less\n"));
 #else
        dSP;
        I32 type, count;
diff --git a/dist/Storable/t/retrieve.t b/dist/Storable/t/retrieve.t
index 652118961c23..ccd907b8d6eb 100644
--- a/dist/Storable/t/retrieve.t
+++ b/dist/Storable/t/retrieve.t
@@ -1,12 +1,14 @@
 #!./perl
 #
 #  Copyright (c) 1995-2000, Raphael Manfredi
+#  Copyright (c) 2017, cPanel Inc
 #
 #  You may redistribute only under the same terms as Perl 5, as specified
 #  in the README file that comes with the distribution.
 #

 sub BEGIN {
+    unshift @INC, 'dist/Storable/t' if $ENV{PERL_CORE} and -d 'dist/Storable/t';
     unshift @INC, 't';
     unshift @INC, 't/compat' if $] < 5.006002;
     require Config; import Config;
@@ -19,7 +21,7 @@ sub BEGIN {


 use Storable qw(store retrieve nstore);
-use Test::More tests => 14;
+use Test::More tests => 20;

 $a = 'toto';
 $b = \$a;
@@ -54,4 +56,31 @@ is($d1, $d2);
 isnt($root->[1], undef);
 is(length $root->[1], 0);

+# $Storable::DEBUGME = 1;
+{
+    # len>I32: todo patch the storable image number into the strings, fake 2.10
+    # $Storable::BIN_MINOR
+    my $retrieve_blessed =
"\x04\x0a\x08\x31\x32\x33\x34\x35\x36\x37\x38\x04\x08\x08\x08\x11\xff\x49\x6e\x74\xff\x72\x6
e\x61\x6c\x73\x02\x00\x00\x00\x00";
+    my $x = eval { Storable::mretrieve($retrieve_blessed); };
+    # Long integer or Double size or Byte order is not compatible
+    like($@, qr/^(Corrupted classname length|.* is not compatible|panic: malloc)/, "RT
#130635 $@");
+    is($x, undef, 'and undef result');
+}
+
+{
+    # len>I32
+    my $retrieve_hook =
"\x04\x0a\x08\x31\x32\x33\x34\x35\x36\x37\x38\x04\x08\x08\x08\x13\x04\x49\xfe\xf4\xff\x72\x6
e\x61\x6c\x73\x02\x00\x00\x00\x00";
+    my $x = eval { Storable::mretrieve($retrieve_hook); };
+    like($@, qr/^(Corrupted classname length|.* is not compatible|panic: malloc)/, "$@");
+    is($x, undef, 'and undef result');
+}
+
+{
+    # len<I32, len>127: stack overflow
+    my $retrieve_hook =
"\x04\x0a\x08\x31\x32\x33\x34\x35\x36\x37\x38\x04\x08\x08\x08\x13\x04\x49\xfe\xf4\x7f\x72\x6
e\x61\x6c\x73\x02\x00\x00\x00\x00";
```

```
+    my $x = eval { Storable::mretrieve($retrieve_hook); };
+    is($?, 0, "no stack overflow in retrieve_hook()");
+    is($x, undef, 'either out of mem or normal error (malloc 2GB)');
+}
+
 END { 1 while unlink("store$$", 'nstore') }
```