Avoid crash on crash report when a bad function pointer was called (#...11298)

Browse files

```
If Redis crashes due to calling an invalid function pointer,
the `backtrace` function will try to dereference this invalid pointer
which will cause a crash inside the crash report and will kill
the processes without having all the crash report information.

Example:

```
=== REDIS BUG REPORT START: Cut & paste starting from here ===
198672:M 19 Sep 2022 18:06:12.936 # Redis 255.255.255 crashed by signal: 11, si_code: 1
198672:M 19 Sep 2022 18:06:12.936 # Accessing address: 0x1
198672:M 19 Sep 2022 18:06:12.936 # Crashed running the instruction at: 0x1
// here the processes is crashing
```

This PR tries to fix this crash be:
1. Identify the issue when it happened.
2. Replace the invalid pointer with a pointer to some dummy function
   so that `backtrace` will not crash.

I identification is done by comparing `eip` to `info->si_addr`, if they
are the same we know that the crash happened on the same address it tries to
accesses and we can conclude that it tries to call and invalid function pointer.

To replace the invalid pointer we introduce a new function, `setMcontextEip`,
which is very similar to `getMcontextEip` and it knows to set the Eip for the
different supported OS's. After printing the trace we retrieve the old `Eip` value.
```

unstable (#11298)

MeirShpilraien committed on Sep 29

1 parent f106bee commit 0bf90d944313919eb8e63d3588bf63a367f020a3

Showing **1 changed file** with **58 additions** and **22 deletions**.

Split Unified

80 src/debug.c

1123	1123	}
1124	1124	

```
1125  1125    #ifdef HAVE_BACKTRACE
1126        -   static void *getMcontextEip(ucontext_t *uc) {
      1126  +
      1127  +   /* Returns the current eip and set it to the given new value (if its not NULL) */
      1128  +   static void* getAndSetMcontextEip(ucontext_t *uc, void *eip) {
1127  1129    #define NOT_SUPPORTED() do {\
1128  1130        UNUSED(uc);\
      1131  +     UNUSED(eip);\
1129  1132        return NULL;\
1130  1133    } while(0)
      1134  + #define GET_SET_RETURN(target_var, new_val) do {\
      1135  +     void *old_val = (void*)target_var; \
      1136  +     if (new_val) { \
      1137  +         void **temp = (void**)&target_var; \
      1138  +         *temp = new_val; \
      1139  +     } \
      1140  +     return old_val; \
      1141  + } while(0)
1131  1142    #if defined(__APPLE__) && !defined(MAC_OS_X_VERSION_10_6)
1132  1143        /* OSX < 10.6 */
1133  1144        #if defined(__x86_64__)
1134        -     return (void*) uc->uc_mcontext->__ss.__rip;
      1145  +     GET_SET_RETURN(uc->uc_mcontext->__ss.__rip, eip);
1135  1146        #elif defined(__i386__)
1136        -     return (void*) uc->uc_mcontext->__ss.__eip;
      1147  +     GET_SET_RETURN(uc->uc_mcontext->__ss.__eip, eip);
1137  1148        #else
1138        -     return (void*) uc->uc_mcontext->__ss.__srr0;
      1149  +     GET_SET_RETURN(uc->uc_mcontext->__ss.__srr0, eip);
1139  1150        #endif
1140  1151    #elif defined(__APPLE__) && defined(MAC_OS_X_VERSION_10_6)
1141  1152        /* OSX >= 10.6 */
1142  1153        #if defined(_STRUCT_X86_THREAD_STATE64) && !defined(__i386__)
1143        -     return (void*) uc->uc_mcontext->__ss.__rip;
      1154  +     GET_SET_RETURN(uc->uc_mcontext->__ss.__rip, eip);
1144  1155        #elif defined(__i386__)
1145        -     return (void*) uc->uc_mcontext->__ss.__eip;
      1156  +     GET_SET_RETURN(uc->uc_mcontext->__ss.__eip, eip);
1146  1157        #else
1147  1158        /* OSX ARM64 */
1148        -     return (void*) arm_thread_state64_get_pc(uc->uc_mcontext->__ss);
      1159  +     void *old_val = (void*)arm_thread_state64_get_pc(uc->uc_mcontext->__ss);
      1160  +     if (eip) {
      1161  +         arm_thread_state64_set_pc_fptr(uc->uc_mcontext->__ss, eip);
      1162  +     }
      1163  +     return old_val;
1149  1164        #endif
1150  1165    #elif defined(__linux__)
1151  1166        /* Linux */
```

```
1152  1167        #if defined(__i386__) || ((defined(__X86_64__) || defined(__x86_64__)) && defined(__I
1153        -     return (void*) uc->uc_mcontext.gregs[14]; /* Linux 32 */
      1168  +     GET_SET_RETURN(uc->uc_mcontext.gregs[14], eip);
1154  1169        #elif defined(__X86_64__) || defined(__x86_64__)
1155        -     return (void*) uc->uc_mcontext.gregs[16]; /* Linux 64 */
      1170  +     GET_SET_RETURN(uc->uc_mcontext.gregs[16], eip);
1156  1171        #elif defined(__ia64__) /* Linux IA64 */
1157        -     return (void*) uc->uc_mcontext.sc_ip;
      1172  +     GET_SET_RETURN(uc->uc_mcontext.sc_ip, eip);
1158  1173        #elif defined(__arm__) /* Linux ARM */
1159        -     return (void*) uc->uc_mcontext.arm_pc;
      1174  +     GET_SET_RETURN(uc->uc_mcontext.arm_pc, eip);
1160  1175        #elif defined(__aarch64__) /* Linux AArch64 */
1161        -     return (void*) uc->uc_mcontext.pc;
      1176  +     GET_SET_RETURN(uc->uc_mcontext.pc, eip);
1162  1177        #else
1163  1178        NOT_SUPPORTED();
1164  1179        #endif
1165  1180    #elif defined(__FreeBSD__)
1166  1181        /* FreeBSD */
1167  1182        #if defined(__i386__)
1168        -     return (void*) uc->uc_mcontext.mc_eip;
      1183  +     GET_SET_RETURN(uc->uc_mcontext.mc_eip, eip);
1169  1184        #elif defined(__x86_64__)
1170        -     return (void*) uc->uc_mcontext.mc_rip;
      1185  +     GET_SET_RETURN(uc->uc_mcontext.mc_rip, eip);
1171  1186        #else
1172  1187        NOT_SUPPORTED();
1173  1188        #endif
1174  1189    #elif defined(__OpenBSD__)
1175  1190        /* OpenBSD */
1176  1191        #if defined(__i386__)
1177        -     return (void*) uc->sc_eip;
      1192  +     GET_SET_RETURN(uc->sc_eip, eip);
1178  1193        #elif defined(__x86_64__)
1179        -     return (void*) uc->sc_rip;
      1194  +     GET_SET_RETURN(uc->sc_rip, eip);
1180  1195        #else
1181  1196        NOT_SUPPORTED();
1182  1197        #endif
1183  1198    #elif defined(__NetBSD__)
1184  1199        #if defined(__i386__)
1185        -     return (void*) uc->uc_mcontext.__gregs[_REG_EIP];
      1200  +     GET_SET_RETURN(uc->uc_mcontext.__gregs[_REG_EIP], eip);
1186  1201        #elif defined(__x86_64__)
1187        -     return (void*) uc->uc_mcontext.__gregs[_REG_RIP];
      1202  +     GET_SET_RETURN(uc->uc_mcontext.__gregs[_REG_RIP], eip);
1188  1203        #else
1189  1204        NOT_SUPPORTED();
```

```
1190  1205      #endif
1191  1206  #elif defined(__DragonFly__)
1192        -     return (void*) uc->uc_mcontext.mc_rip;
      1207  +     GET_SET_RETURN(uc->uc_mcontext.mc_rip, eip);
1193  1208  #else
1194  1209      NOT_SUPPORTED();
1195  1210  #endif
1951  1966      }
1952  1967  }
1953  1968
      1969  + void invalidFunctionWasCalled() {}
      1970  +
      1971  + typedef void (*invalidFunctionWasCalledType)();
      1972  +
1954  1973  void sigsegvHandler(int sig, siginfo_t *info, void *secret) {
1955  1974      UNUSED(secret);
1956  1975      UNUSED(info);
1968  1987
1969  1988  #ifdef HAVE_BACKTRACE
1970  1989      ucontext_t *uc = (ucontext_t*) secret;
1971        -     void *eip = getMcontextEip(uc);
      1990  +     void *eip = getAndSetMcontextEip(uc, NULL);
1972  1991      if (eip != NULL) {
1973  1992          serverLog(LL_WARNING,
1974  1993          "Crashed running the instruction at: %p", eip);
1975  1994      }
1976  1995
1977        -     logStackTrace(getMcontextEip(uc), 1);
      1996  +     if (eip == info->si_addr) {
      1997  +         /* When eip matches the bad address, it's an indication that we crashed when call
      1998  +          * function pointer. In that case the call to backtrace will crash trying to acce
      1999  +          * won't get a crash report logged. Set it to a valid point to avoid that crash.
      2000  +
      2001  +         /* This trick allow to avoid compiler warning */
      2002  +         void *ptr;
      2003  +         invalidFunctionWasCalledType *ptr_ptr = (invalidFunctionWasCalledType*)&ptr;
      2004  +         *ptr_ptr = invalidFunctionWasCalled;
      2005  +         getAndSetMcontextEip(uc, ptr);
      2006  +     }
      2007  +
      2008  +     logStackTrace(eip, 1);
      2009  +
      2010  +     if (eip == info->si_addr) {
      2011  +         /* Restore old eip */
      2012  +         getAndSetMcontextEip(uc, eip);
      2013  +     }
1978  2014
1979  2015      logRegisters(uc);
1980  2016  #endif
```

```
2079  2115
2080  2116      serverLogFromHandler(LL_WARNING,"\n--- WATCHDOG TIMER EXPIRED ---");
2081  2117  #ifdef HAVE_BACKTRACE
2082        -     logStackTrace(getMcontextEip(uc), 1);
      2118  +     logStackTrace(getAndSetMcontextEip(uc, NULL), 1);
2083  2119  #else
2084  2120      serverLogFromHandler(LL_WARNING,"Sorry: no support for backtrace().");
2085  2121  #endif
```

0 comments on commit `0bf90d9`

Please sign in to comment.