MISRA C развивается для решения проблем безопасности в сегодняшнем встроенном программном обеспечении

exponenta.ru

LDRA Software Technology

экспонента
ЦЕНТР ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И МОДЕЛИРОВАНИЯ

Введение в новые правила и требования к их соблюдению

Руководства MISRA C, разработанные и поддерживаемые MISRA, определяют подмножество языка C, подходящего для разработки любого приложения с высокими требованиями к целостности или высокой надежности. Первоначально разработанные для содействия использованию языка C в критически важных для безопасности приложениях в автомобильной промышленности, стандарты MISRA получили широкое признание в безопасных, жизненно важных и критически важных применений в аэрокосмической, телекоммуникационной, медицинской, оборонительной, железнодорожной, и других отраслях. Обновления на протяжении многих лет включали расширения и улучшения, помогающие снизить риски, связанные с программным обеспечением, для критически важных приложений, позволяя программистам тратить больше времени на кодирование и меньше времени на усилия по обеспечению соблюдения стандарта. Хотя подмножество языка MISRA имеет прочные связи с рынком критически важных приложений, оно стало проверенным подходом для кодирования лучших практик для любых встроенных систем. По мере развития индустрии встроенного программного обеспечения возникают угрозы для жизни, оборудования и бизнеса.

В то время как рекомендации MISRA разработаны, чтобы помочь разработчикам писать высококачественный код, который по своей природе более безопасен. Повышенная осведомленность отрасли о рисках безопасности привела к появлению новых рекомендаций по их устранению.

Новые рекомендации по безопасности реагируют на потребности подключенной отрасли

Риски безопасности значительно увеличились с появлением Интернета вещей и повышением требований к безопасности встроенных устройств. Даже те устройства, которые изначально не были предназначены для подключения к сети, вероятно, добавят эту функциональность по мере развития продуктов. Это делает многие из этих продуктов уязвимыми для злоумышленных захватов хакерами, а также непреднамеренного раскрытия личных или критически важных данных из-за слабых или небезопасных методов кодирования. Комитет MISRA признает, что безопасность не может быть запоздалой мыслью о развитии. Она должна быть принята во внимание с самого начала с использованием передовых методов кодирования и правил, предназначенных для защиты безопасности и защиты OEM-производителей и конечных пользователей. Аналогичные выводы сделали и другие организации.

После публикации MISRA C: 2012, комитет, отвечающий за поддержание стандарта C, опубликовал Руководства ISO/IEC 17961:2013 C language Security Guidelines. В ответ комитет MISRA опубликовал Поправку 1 (Amendment 1) к MISRA C: 2012, которая предназначена для поддержки этих новых требований к безопасности. Поправка является улучшением и полностью совместима со всеми существующими изданиями руководств MISRA и станет стандартным подходом для всех будущих изданий руководств MISRA.

MISRA C: 2012 Amendment 1 была выпущена, чтобы помочь разработчикам избежать практики кодирования, которая может привести к уязвимостям системы безопасности и написать код, который более понятен и удобен в сопровождении. Следуя этим дополнительным рекомендациям, разработчики могут более тщательно проанализировать свой код и могут показать регулирующим органам то, что они применяют безопасные методы кодирования. Это особенно важно в таких отраслях, как автомобилестроение, где угрозы безопасности привели к жестким требованиям OEM-производителей
к разработчикам, в части доказательства того, что их программное обеспечение соответствует самым высоким стандартам функциональной и информационной безопасности.

MISRA C: 2012 Amendment 1 устанавливает 14 новых руководящих принципов для безопасного кодирования C для улучшения охвата проблем безопасности, обозначенных Руководствами ISO C Secure. В некоторых из этих рекомендаций рассматриваются конкретные проблемы, связанные с использованием «ненадежных» данных - хорошо известной уязвимости безопасности. Конкретные примеры и их объяснения помогают уточнить важность этой поправки.

Небезопасные примеры кодирования и относящиеся к ним правила

Пример 1: не открывайте дверь, чтобы выдать свой пароль.

«Действительность значений, полученных от внешних источников, должна быть проверена»

Этот пример контролирует, какие данные могут быть отправлены внешним источникам, что стало критическим, поскольку все больше и больше устройств подключены друг к другу или к Интернету.

Если код написан неправильно, внутренние данные могут быть открыты для внешних источников. Это правило защищает от уязвимостей «Heartbleed», в которых хакеры могли извлекать данные, в том числе пароли, через Интернет в открытом виде, тщательно обрабатывая сообщения для извлечения данных.

Следующий пример кода может быть опасным, потому что ввод данных пользователя не проверяется.

В этом примере пользовательский ввод снова выводится на экран, однако на самом деле это, скорее всего, будет использоваться в программе после чтения. Злоумышленник может использовать эту уязвимость, введя вредоносный код или значения. Одним из вариантов этого будет то, что числовое значение будет использоваться для доступа к буферу; злоумышленник сможет использовать это для доступа за пределы буфера, потенциально читая текстовые пароли или другие конфиденциальные данные с сетевого сервера.

```
void f1( void )
{
  char input [ 128 ];
  ( void ) scanf ( "%s", input );
  printf ( "%s", input );   /* Non compliant */
}
```

Пример 2: используйте правильную функцию для правильной задачи.

«Функция стандартной библиотеки memcmp не должна использоваться для сравнения строк с нулевым окончанием»

```
extern char buffer1[ 12 ];
extern char buffer2[ 12 ];

void f1 ( void )
{
  strcpy ( buffer1, "abc" );
  strcpy ( buffer2, "abc" );
  if ( memcmp ( buffer1, buffer2, sizeof ( buffer1 ) ) != 0 )
  {
    /* The strings stored in buffer1 and buffer2 are reported to be
     * different, but this may actually be due to differences in the
     * uninitialised characters stored after the null terminators.

     */
  }

}
```

Memcmp предназначена для сравнения блоков памяти и не предназначена для сравнения строк, таких как пароли. Поскольку она возвращает больше, чем просто true и false, ее можно использовать для отображения паролей в системах баз данных.

Следующий пример кода использует memcmp для сравнения строк, в то время, как должен был использоваться strcmp.

Пример 3: если кто-то может контролировать, как вы говорите, они могут заставить вас сказать все, что они хотят.

«Указатель, возвращаемый функциями стандартной библиотеки asctime, ctime, gmtime, localtime, localeconv, getenv, setlocale или strerror, не должен использоваться после последующего вызова той же функции»

«Сообщения локали» управляют форматированием строк, определяя, как «говорит» программа. Если хакеры смогут контролировать, как форматированы строки, они смогут заставить их "говорить" все, что захотят, включая управление любыми программами, с которыми взаимодействует первая программа. Это означает, что эти строки могут использоваться для выполнения произвольных команд и захвата системы.

Следующий пример кода может работать не так, как ожидалось, потому что второй вызов setlocale может привести к тому, что строка, на которую ссылается «res1», будет такой же, как и ссылка «res2». Помимо правильности, контроль «локали» использовался в эксплойтах, в которых строки были переформатированы для облегчения выполнения команд на удаленном компьютере.

```
void f1( void )
{
  const char *res1;
  const char *res2;

  res1 = setlocale ( LC_ALL, 0 );
  res2 = setlocale ( LC_MONETARY, "French" );

  printf ( "%s\n", res1 ); /* "res1" may NOT be related to the 'C' locale. */
}
```

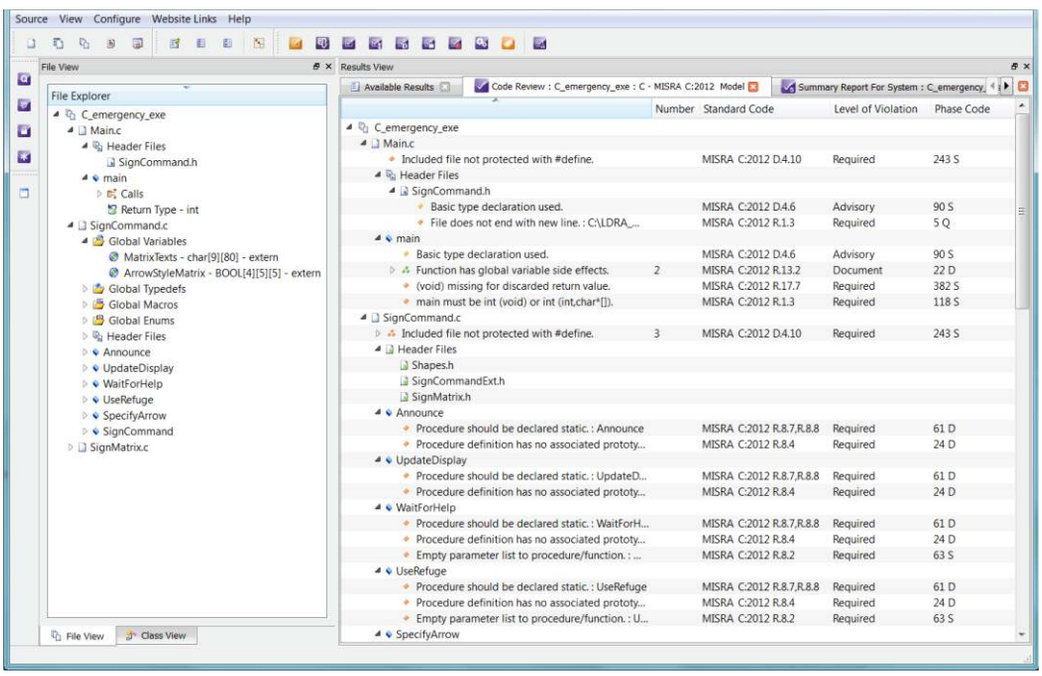
А вы уверены в соблюдении безопасности?

- Используете ли вы автоматизированные инструменты, которые могут тщательно проверять соответствие стандартам и доказывать это клиентам или регулирующим органам?
- Предоставляет ли ваш поставщик инструмента полную матрицу соответствия, чтобы вы точно знали, что проверяет инструмент?
- Может ли ваш инструмент также проверять другие стандарты надежности или безопасности, такие как CWE, CERT и отраслевые стандарты?
- Ваш инструмент также обеспечивает динамический анализ, чтобы убедиться, что нет «мертвого кода», который может повлиять на безопасность?
- У вас есть возможности полного модульного тестирования и сбора структурного покрытия для обеспечения эффективности вашего процесса тестирования?

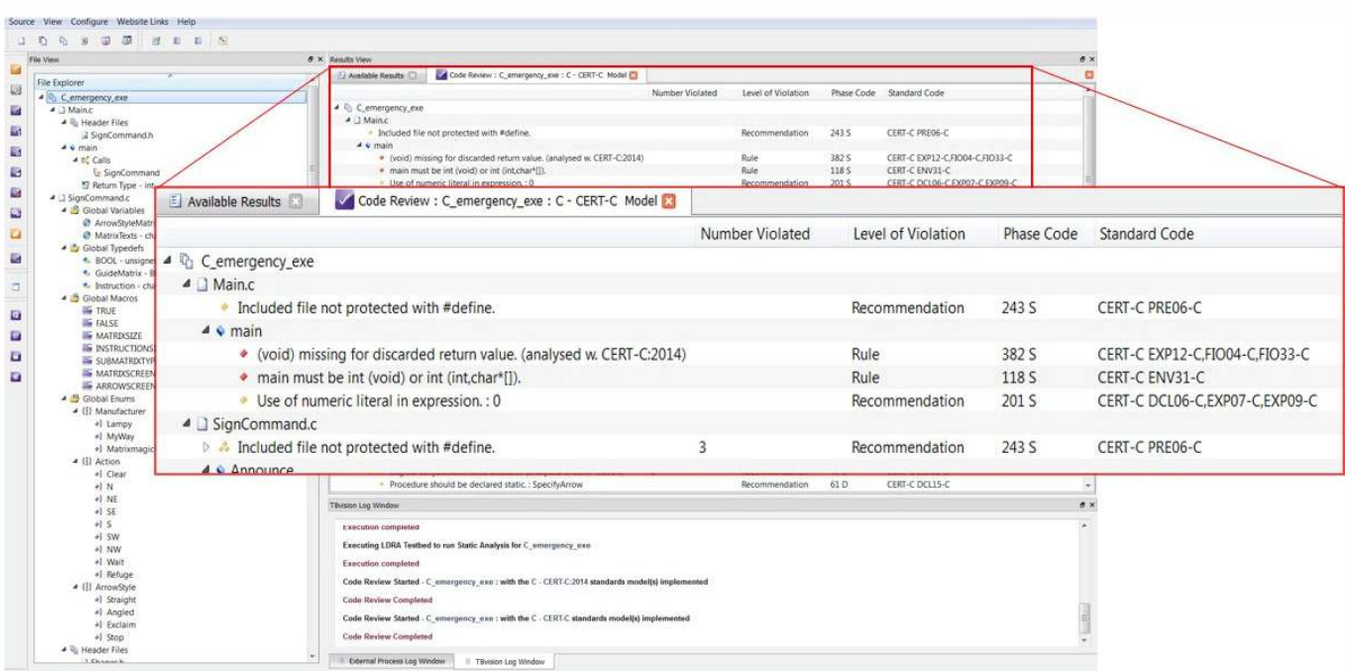
В дополнение к поправке, касающейся включения 14 новых правил, комитет MISRA также выпустил приложение MISRA C: 2012 Addendum 2, которое отображает общий охват MISRA C: 2012 ISO / IEC 17961: 2013 и дает дополнительные доказательства того, что MISRA C одинаково применим в среде, связанной с безопасностью, поскольку относится к надежности.

Новые рекомендации по безопасности реагируют на потребности подключенной отрасли

С тех пор, когда первые директивы MISRA C были впервые опубликованы, комитет признал, что иногда возникают обстоятельства, при которых невыполнимо или необоснованно выполнять все требования. Однако концепция одобренных нарушений, известных как отклонения, вызвала много споров, особенно при обсуждении их влияния на требование о соблюдении MISRA. Результатом было множество подходов, от политики «без отклонения» до свободного использования отклонений, использовавшихся просто для документирования несоблюдения стандарта.



Инспекция кода — выявление нарушений в системе.



Ощутимые выгоды TBsecure обеспечивают немедленную отдачу от инвестиций.

При поддержке крупных OEM-производителей, особенно из автомобильного рынка, где несоблюдение стандарта привело к широко освещенным нарушениям безопасности, комитет MISRA опубликовал руководство MISRA Compliance 2016, чтобы помочь разработчикам достичь и показать соответствие стандарту кодирования MISRA. Эта редакция дает более четкое руководство по использованию отклонений и определяет, что подразумевается под соответствием MISRA. Она также обеспечивает механизм для создания предварительно одобренных разрешений на отклонение и для подгонки классификации стандарта.

Соответствие может быть достигнуто и доказано путем проверки кода относительно матрицы соответствия MISRA C. Хотя теоретически это возможно было бы сделать вручную, сложность и количество сегодняшнего программного обеспечения, содержащего миллионы строк кода, которые теперь включены в автомобили, делают это необоснованным. Основным средством проверки кода по стандартам и рекомендациям является использование автоматических инструментов статического анализа, которые обнаруживают потенциальные недостатки безопасности на ранней стадии процесса разработки, чтобы разработчики могли их устранить до того, как код будет скомпилирован. Набор инструментов LDRA включает всестороннее покрытие новых правил MISRA C: 2012 Amendment 1.

Certificate Number FM 26376

+7 (495) 009-65-85

info@exponenta.ru

exponenta.ru

LDRA
Software Technology

экспонента
ЦЕНТР ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ
И МОДЕЛИРОВАНИЯ