

УДК 004.312.466

Б. В. СИВКО, магистр технических наук, Белорусский государственный университет транспорта, г. Гомель

ОПРЕДЕЛЕНИЕ ФУНКЦИИ БЕЗОПАСНОСТИ ПРИ ВЕРИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ ЖЕЛЕЗНОДОРОЖНОЙ АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

Рассмотрены формализация и определение функции безопасности при доказательстве корректности программного обеспечения микропроцессорных систем железнодорожной автоматики и телемеханики. Приведены способы поиска и выбора функции безопасности на основании технического задания, ограниченности ресурсов, используемой стратегии обеспечения безопасности и общих требований к характеристикам системы.

Показано, что определение доказываемой функции безопасности влияет как на способность к нахождению ошибок в программном обеспечении во время верификации, так и на качество аппаратно-программного комплекса в целом в случае использования функции на этапах разработки.

Развитие и совершенствование микроэлектронной базы расширяет возможности применения железнодорожной автоматики и телемеханики, позволяя реализовывать и предоставлять более широкую функциональность для эксплуатируемых систем. Однако проектирование современных микропроцессорных устройств и их внедрение на железнодорожном транспорте требует удовлетворения соответствующего уровня безопасности, который не может быть получен с помощью подходов, используемых в релейной схемотехнике, из-за чего необходимы принципиально отличные методы и способы решения проблемы. В настоящее время отсутствуют универсальные и общепризнанные способы доказательства безопасности микроэлектронных систем железнодорожной автоматики и телемеханики (СЖАТ).

В эксплуатируемых аппаратно-программных комплексах (АПК) программное обеспечение (ПО) является неотъемлемым компонентом, влияющим на безопасность системы в целом. Так как ни один из существующих методов не гарантирует полного отсутствия программных ошибок, то практикуется комплексный подход, заключающийся в применении ряда методов и средств для повышения безопасности и надежности системы на всех этапах её жизненного цикла.

Одним из возможных способов поиска ошибок и улучшения качества ПО является доказательство корректности, которое относится к формальным методам (*Formal Methods*) и успешно используется для верификации устройств СЖАТ на Белорусской железной дороге [1–3]. Данный способ может быть применен как для готового ПО, так и на ранних этапах разработки всего АПК, но в любом случае одним из первых шагов верификации является определение функции безопасности, подлежащей проверке на корректность [4].

Разработка и эксплуатация ПО, а также ряд исследований говорят о том, что чем позже обнаруживается ошибка, тем сложнее выявить её и исправить и тем больше проблем она может принести [5, 6]. При этом исправление допустимых до проектирования ошибок обходится в десятки раз дороже, чем исправление ошибок этапов реализации [7, 8]. Определение функции безопасности для проведения верификации относится к формализации решаемой задачи, является спецификаци-

ей по отношению к доказательству корректности и обладает теми же свойствами, что и постановка требований при разработке ПО. Потенциальные ошибки, допускаемые при определении данной функции, негативно влияют на качество верификации и могут приводить к искажению результатов доказательства корректности и, как следствие, его полному пересмотру.

На рисунке 1 показана последовательность этапов анализа на безопасность с определением функции безопасности.



Рисунок 1 – Последовательность этапов анализа на безопасность

Процесс валидации независим от последующей верификации: после определения свойств, подлежащих проверке, начинается доказательство корректности, которое работает с тем, что ему задано функцией безопасности. Если допускаются ошибки на данном этапе либо не принимаются во внимание особенности поведения, влияющие на безопасность, то это непосредственным образом влияет на качество последующей верификации.

Кроме того, какие бы эффективные и диверситетные методы и средства не использовались во время доказательства корректности, они не в состоянии выявить и исправить проблемы, созданные во время планирования, так как они работают с одинаковой спецификацией и только конечный пользователь может указать на ошибку, допущенную при составлении требований.

Одной из проблем валидации является определение условий, подлежащих проверке. Особенности данного процесса таковы, что после формализации нет однозначного критерия и уверенности в том, что утвержденная доказываемая функция является необходимой и достаточной [9]. Во время последующей разработки или анализа на безопасность может быть выяснено, что заданные рамки слишком строги и доказательство корректности провести невозможно, или, напротив, слишком слабы, из-за чего уменьшается вероятность нахождения ошибок в ПО.

Накопленный автором опыт верификации критически важных объектов информатизации микропроцессорных СЖАТ говорит о том, что определение доказываемой функции безопасности разрабатываемых и существующих АПК необходимо проводить на основании:

- используемой стратегии обеспечения безопасности; например, во время проектирования может быть использована стратегия применения логических элементов с несимметричными отказами (h_1 -надежные элементы), и система должна придерживаться её во время всего жизненного цикла [10];

- требований безопасности ко всей системе; например, верифицируемый компонент должен выдерживать заданные временные диапазоны сигналов взаимодействия с другими компонентами системы;

- требований безопасности к рассматриваемому АПК; например, система обязана сохранять инвариант и переходить в безопасное состояние в случае внутренних сбоев.

Таким образом, результатом верификации является доказательство того, что свойства рассматриваемого ПО или компонента выполняют требования безопасности к нему и к его окружению, а также согласованы с используемой стратегией обеспечения безопасности.

Определение доказываемой функции безопасности может проводиться на основании только технического задания (ТЗ) разрабатываемого АПК, но в этом случае имеется ряд потенциальных проблем:

- недостатки определения параметров безопасности, сформулированных в ТЗ, переносятся в определяемую функцию безопасности;

- формализация функции безопасности на основании ТЗ более сложна, нежели на основании требований безопасности.

Рассмотрение требований к системе в отношении стратегии обеспечения безопасности, безопасного внутреннего поведения и согласования взаимодействия с внешними компонентами позволяет быстрее и эффективнее разбивать доказываемую цельную функцию на

эквивалентные, но более простые функции, что уменьшает сложность верификации и улучшает её качество.

В процессе формализации функции безопасности возможно внесение изменений в спецификации для исключения недостатков формулировки параметров безопасности, определенных на этапе составления ТЗ.

Во время проведения валидации нельзя гарантировать необходимость и достаточность функции безопасности, но можно её изменить и повлиять на особенности проводимого доказательства корректности в дальнейшем.

Основными применяемыми автором способами изменения функции безопасности является её расширение (ослабление, более слабое определение) и сужение (усиление, более строгое определение).

Допустим, у нас есть множество функций безопасности f_i , каждая из которых зависит от вектора аргументов \bar{a} и имеет область значений истина/ложь (*true/false*). Тогда усилением функции f_1 является переход к такой функции f_2 , при котором выполняются условия (1) и (2):

$$\forall (f_2(\bar{a}) = \text{true}) \quad f_1(\bar{a}) = \text{true}; \quad (1)$$

$$\exists (f_1(\bar{a}) = \text{true}) \quad f_2(\bar{a}) = \text{false}. \quad (2)$$

Ослаблением функции f_3 является переход к такой функции f_4 , при котором выполняются условия (3) и (4):

$$\forall (f_3(\bar{a}) = \text{true}) \quad f_4(\bar{a}) = \text{true}; \quad (3)$$

$$\exists (f_4(\bar{a}) = \text{true}) \quad f_3(\bar{a}) = \text{false}. \quad (4)$$

Таким образом, усилением функции является переход от одной функции f_1 к другой функции f_2 таким образом, что всегда, когда истинна f_2 , то истинна и f_1 , но при этом существуют такие истинные значения f_1 , при которых f_2 ложна. Ослаблением является аналогичный обратный переход.

Рассмотрим пример трех функций безопасности, выбор которых может повлиять на доказательство корректности. Предположим, что имеется ПО АПК, работающее по замкнутому циклу, у которого каждое последующее выполнение тела цикла должно отличаться от предыдущего и для их отличия в памяти хранится идентификатор id . Будем считать, что число циклов конечно и каждый из них пронумерован последовательно во времени от 1 до n и, соответственно, существует множество идентификаторов $\bar{a} = \{id_1, id_2, \dots, id_n\}$.

Пример первой функции безопасности (5):

$$g_1(\bar{a}) = \text{true}, \forall (i \in N, i < n) \quad id_i \neq id_{i+1}. \quad (5)$$

Данная функция гарантирует отличие идентификатора от предыдущего и может быть использована для безопасного обновления входящей информации.

Вторая функция безопасности гарантирует уникальность идентификатора за все время работы АПК с момента его запуска и может быть использована для обновления информации, которое происходит не на каждом витке цикла (6):

$$g_2(\bar{a}) = \text{true}, \forall (i \neq j \in N; i, j \leq n) \quad id_i \neq id_j. \quad (6)$$

Следующая функция безопасности гарантирует, что каждый последующий идентификатор ровно на 1 больше предыдущего, и может быть использована для расчёта количества циклов между событиями (7):

$$g_3(\bar{a}) = \text{true}, \forall (i \in N, i < n) \quad id_{i+1} = 1 + id_i. \quad (7)$$

Функция g_3 более строгая, чем функция g_2 , которая в свою очередь, более строгая, чем g_1 . Также функция g_1 более слабая, чем g_2 , которая более слабая, чем g_3 .

Верифицировать более строгую функцию сложнее, чем более слабую, на это тратится большее количество ресурсов и не всегда это удается. Но если есть возможность, то рекомендуется доказывать корректность более сильной функции, так как это имеет следующие положительные эффекты:

- получаем более точное представление о том, как работает система, определяются свойства и поведение более строго;

- снижается анализируемая сложность функционирования и тем самым повышается вероятность нахождения ошибок;

- доказанные функции могут быть использованы в дальнейшем для более эффективного проведения других доказательств корректности рассматриваемого АПК.

Однако при анализе на безопасность возможно ослабление проверяемой функции при сохранении необходимого доказываемого уровня безопасности, что может быть необходимым в случае, когда невозможно доказать корректность в предложенном виде или отсутствуют ресурсы для проведения такого объема работ, но результат позволяет сделать заключение о безопасности ПО.

Таким образом, определение функции безопасности для проведения верификации является важным этапом анализа на безопасность, и её выбор представляет собой компромисс между имеющимися ресурсами и доказываемыми свойствами. Практика показывает, что избежать компромисса удастся только в случае, если систе-

ма подготовлена к тому, чтобы быть верифицируемой, когда задачи определения функции безопасности решаются до этапов разработки и проектирования, что возможно только для разрабатываемых и проектируемых АПК [3, 4].

Список литературы

- 1 **Сивко, Б. В.** Доказательство корректности блока телеуправления 16-1 диспетчерской централизации «Нёман» / Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2012. – № 1 (24). – С.18–21.
- 2 **Butler, R. W.** What is Formal Methods? / R. W. Butler // NASA LaRC Formal Methods Program, 2001.
- 3 **Харлап, С. Н.** Верификация программного обеспечения микропроцессорной светооптической светодиодной системы / С. Н. Харлап, Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2012. – № 1 (24). – С. 22–25.
- 4 **Сивко, Б. В.** Проектирование безопасного программного обеспечения микропроцессорных устройств автоматики и телемеханики / Б. В. Сивко // Проблемы безопасности на трансп. : тез. докл. VI Междунар. науч.-практ. конф., Гомель, 29–30 ноября 2012 г. / М-во образования Респ. Беларусь, М-во трансп. и коммуникаций Респ. Беларусь, Бел. ж. д., Белорус. гос. ун-т трансп.; редкол.: В. И. Сенько (отв. ред.) [и др.]. – Гомель, 2012. – С. 205.
- 5 **Fagan, M. E.** Design and code inspections to reduce errors in program development / M. E. Fagan // IBM Systems Journal. Vol. 15. Is. 3. September 1976. – P. 182–211.
- 6 **Boehm, B. W.** Software engineering / B. W. Boehm // IEEE Transactions on Computers 25: 1976. – 1226–1241.
- 7 **Тэллес, М.** Наука отладки / М. Тэллес; пер. с англ. Ю. Хсих. – М. : КУДИЦ-ОБРАЗ, 2003.
- 8 **Boehm, B. W.** (1988) Understanding and controlling software costs / B. W. Boehm, P. N. Papaccio // IEEE Trans. Software Eng. 14(10). October 1988. – 1462–1477.
- 9 **Gerhart, S. L.** Observations of Fallibility in Applications of Modern Programming Methodologies / S. L. Gerhart, L. Yelowitz // IEEE Trans. Software Eng. – Vol. 2. No. 3. – 1976. – P. 195–207.
- 10 **Сапожников, Вл. В.** Дискретные устройства железнодорожной автоматики и телемеханики / Вл. В. Сапожников, Ю. А. Кравцов. – М.: Транспорт, 1988.

Получено 27.12.2012

B. V. Sivko. Safety function definition during software verification of microprocessor interface rail systems.

Definition and formalization of safety function for software proof of correctness of microprocessor railway automation units has been considered. Are ways to select and search the safety function on the basis of terms of reference, limited resources, the strategy used to proof safety and general performance requirements of the system.

It is shown that the definition of proving safety function affects both the ability to find bugs in software during verification, and the quality of system in general, in the case of functions in development.