Имитационное моделирование процессов взаимодействия субъектов единого информационного пространства позволит проанализировать процессы взаимодействия субъектов, определить риск потери информации, риск получения неполной или недостоверной информации, риск получения дезинформации, риск использования информации другим субъектом, а также выявить и исследовать влияние различных факторов, определяющих качество организации и функционирования механизмов, обеспечивающих информационную безопасность субъектов.

СПИСОК ЛИТЕРАТУРЫ:

1. *Андрианов В. В.* Обеспечение информационной безопасности бизнеса. М.: ЦИПСиР: Альпина Паблишерс, 2011. — 373 с.
2. *Астахов А. М.* Искусство управления информационными рисками. М.: ДМК Пресс, 2010. — 312 с.
3. *Арямов А. А.* Общая теория риска: юридический, экономический и психологический анализ: монография. М.: РАП. Валтерс Клуверс, 2010. — 208 с.
4. *А. А. Малюк, В. С. Горбатов, В. И. Королев и др.* Введение в информационную безопасность: Учебное пособие для вузов. М.: Горячая линия — Телеком, 2011. — 288 с.
5. *Носова Н. С.* Конкурентная стратегия компании или маркетинговые методы конкурентной борьбы. М.: ИТК «Дашков и К», 2010. — 255 с.
6. *Рубин Ю. Б.* Теория и практика предпринимательской конкуренции: Учебное пособие. М.: Маркет ДС, 2010. — 604 с.

Д. В. Гуров, В. В. Гуров, М. А. Иванов

ИСПОЛЬЗОВАНИЕ МОДЕЛЕЙ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОЦЕНКИ ЗАЩИЩЕННОСТИ ПРОГРАММНОГО КОМПЛЕКСА

Защищенность — это совокупность свойств программного средства, характеризующая его способность предотвращать несанкционированный доступ (НСД), как случайный, так и умышленный, к программам и данным, а также степень удобства и полноты обнаружения результатов такого доступа или действий по разрушению программ и данных [1]. НСД может быть осуществлен при наличии определенных уязвимостей в программном коде. Во многом понятие защищенности перекликается с понятием надежности программного средства, одним из основных свойств которого является устойчивость к ошибке, т. е. способность поддерживать определенный уровень качества функционирования в случаях программных ошибок или нарушения определенного интерфейса.

Вопросы обеспечения надежности программного обеспечения рассматриваются уже достаточно давно и имеют в своем активе большой набор моделей и методик. В то же время проблема обеспечения защищенности возникла гораздо позже. Поэтому интересно рассмотреть возможность использования моделей надежности ПО применительно к оценке его защищенности. Отладка ПО проводится до тех пор, пока интенсивность появления потока ошибок не снизится до приемлемой для данного применения величины, т. е. среднее время наработки на отказ будет достаточно большим. Поэтому многие модели надежности ориентируются на анализ времени появления очередной ошибки (модель Джелинского—Моранды и др. [2]).

В то же время можно утверждать, что если программа попадет в руки злоумышленника на достаточно продолжительное время, то она будет взломана. Поэтому оценка ее защищенности может проводиться по степени ее стойкости, т. е. по длительности периода, на протяжении которого она противостоит НСД. А этот показатель определяется количеством уязвимостей, оставшихся в программе после этапа ее тестирования. Следовательно, использовать модели

надежности ПО, основанные на временных характеристиках потока ошибок, в данном применении нецелесообразно. Для этой цели следует использовать другой класс моделей, например модель Миллса [2]. Эта модель строится на твердом статистическом фундаменте. Сначала программа «засоряется» некоторым количеством известных ошибок. Эти ошибки вносятся в программу случайным образом, а затем делается предположение, что для ее собственных и внесенных ошибок вероятность обнаружения при последующем тестировании одинакова и зависит только от их количества. Тестируя программу в течение некоторого времени и отсортировывая собственные и внесенные ошибки, можно оценить N — первоначальное число ошибок в программе, а также установить доверительный уровень этого прогноза.

Предположим, что в программу было внесено S ошибок, после чего началось тестирование. Пусть при тестировании обнаружено n собственных ошибок и v ошибок из числа внесенных. Тогда оценка для N по методу максимального правдоподобия будет равна:

$$N = \frac{n}{v} * S .$$

Процесс внесения ошибок в настоящее время является самым слабым местом данной модели, поскольку предполагается, что для собственных и внесенных ошибок вероятность обнаружения одинакова (но неизвестна). Из этого следует, что внесенные ошибки должны быть «типичными» образцами ошибок. Это обеспечить довольно сложно, так как у разработчиков программы и людей, которые вносят в нее заведомо известные ошибки, стиль программирования, квалификация и другие аспекты обычно не совпадают.

Однако применение данного подхода к анализу защищенности программы оказывается более плодотворным по сравнению с его классическим использованием, поскольку для обычных ошибок сведения об их качествах достаточно пространны, а разработанная авторами в [3] классификация уязвимостей позволяет выделить типовые ситуации, характеризующие тот или иной класс уязвимостей, и внести определенное количество уязвимостей каждого класса в исходный текст программы.

Применительно к оценке защищенности использование модели Миллса будет выглядеть следующим образом. Внесем в программу S уязвимостей, в том числе s_i уязвимостей i-го типа ($S = \sum_{i=1}^{m} s_i$), где m — число типов уязвимостей. Тогда если в процессе тестирования обнаружено n собственных уязвимостей, в том числе n_i уязвимостей i-го типа ($n = \sum_{i=1}^{m} n_i$), и v внесенных уязвимостей, в том числе v_i внесенных уязвимостей i-го типа ($v = \sum_{i=1}^{m} v_i$), то количество собственных уязвимостей i-го типа, изначально присутствовавших в программе, составит

$$N_i = \frac{n_i}{v_i} * s_i ,$$

а общее количество уязвимостей N, которое содержалось в программе до начала тестирования, составит:

$$N = \sum_{i=1}^{m} N_i = \sum_{i=1}^{m} (\frac{n_i}{v_i} * s_i).$$

Количество собственных уязвимостей i-го типа, оставшихся в программе по окончании тестирования, будет равно:

$$k_i = N_i - n_i = \frac{n_i}{v_i} * s_i - n_i .$$

Общее количество собственных уязвимостей, оставшихся в программе по окончании тестирования, при этом составит:

$$K = N - n = \sum_{i=1}^{m} (\frac{n_i}{v_i} * s_i) - n.$$

Эту величину можно использовать для решения второго вопроса — оценки меры доверия к модели. Предположим, что в программе имеется не более k собственных уязвимостей, и внесем в нее еще S уязвимостей. Будем тестировать программу до тех пор, пока не будут обнаружены

все внесенные уязвимости. По окончании тестирования подсчитаем число n обнаруженных собственных уязвимостей, содержавшихся в программе. Уровень значимости C, определяющий меру доверия к модели и вероятность того, что «модель будет правильно отклонять ложное предположение», вычисляется по следующей формуле [2]:

$$C = \begin{cases} 1, & при\ n > k \\ \dfrac{S}{S+k+1} & при\ n \le k \end{cases}.$$

Так, если мы предполагаем, что после первоначального этапа тестирования в программе осталось не более 2 уязвимостей ($k = 2$), то для получения уровня значимости $C = 0{,}9$ в нее необходимо внести 27 уязвимостей и найти их все.

Однако тестирование до момента нахождения всех внесенных уязвимостей представляется достаточно сложной задачей. В случае, когда на этапе тестирования найдено только j из S внесенных ошибок ($j \le S$), для оценки достоверности результата можно воспользоваться другой формулой, полученной в работе [4]:

$$C = \begin{cases} 1, & при\ n > k \\ \binom{S}{j-1} / \binom{S+k+1}{k+j} & при\ n \le k \end{cases},$$

где выражение вида $\binom{a}{b}$ вычисляется следующим образом:

$$\binom{a}{b} = \frac{a!}{b!*(a-b)!}.$$

С помощью полученной зависимости можно определить, например, количество уязвимостей, которые следует внести для получения требуемой достоверности, либо другие необходимые для анализа параметры.

В нашем случае задача усложняется тем, что нужно учитывать количество j_i найденных за время тестирования уязвимостей из первоначально внесенных S_i уязвимостей каждого из m типов. В этом случае достоверность оценки получаемого количества собственных уязвимостей i-го типа, изначально присутствовавших в программе, следует оценивать индивидуально:

$$C_i = \begin{cases} 1, & при\ \ n_i > k_i \\ \binom{S_i}{j_i - 1} / \binom{S_i + k_i + 1}{k_i + j_i}, & при\ \ n_i \le k_i \end{cases},$$

где n_i — количество найденных собственных уязвимостей i-го типа, k_i — предполагаемое количество уязвимостей i-го типа.

Данное выражение позволяет, например, определить, какое количество уязвимостей следует искусственно внести в программу перед началом тестирования и обнаружить при тестировании, чтобы получить заданный уровень достоверности модели (рис. 1).

Используя полученные зависимости, можно, в частности, перед этапом тестирования вносить количество уязвимостей того или иного типа таким образом, чтобы получить наиболее достоверные результаты по тем уязвимостям, которые представляют наибольшую опасность, степень которой определяется экспертным путем или согласно данным, приводимым авторитетными в данной области организациями, см., например: [5].

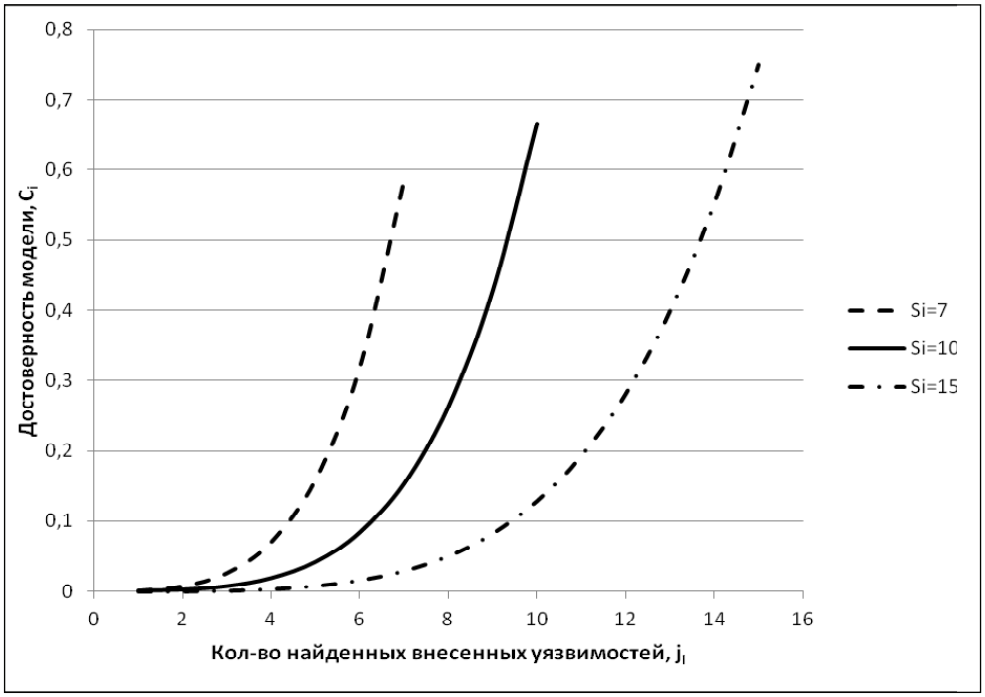

Рис.1. Зависимость достоверности модели (C_i) от количества найденных внесенных уязвимостей (j_i) при $k_i = 4$

СПИСОК ЛИТЕРАТУРЫ:

1. ГОСТ Р ИСО/МЭК 9126-93 «Оценка программной продукции. Характеристики качества и руководства по их применению».
2. *Майерс Г.* Надежность программного обеспечения. М.: Мир, 1980. – 360 с.
3. *Гуров В. В., Гуров Д. В., Иванов М. А., Шустова Л. И.* Технология безопасного программирования и особенности ее преподавания в вузе // Дистанционное и виртуальное обучение. 2010. № 9. С. 35–44.
4. *Teichroev D.* Survey of Languages for Stating Requirements for Computer-Based Information Systems // Proceedings of the 1972 Fall Joint Computer Conference. Montvale, N.J.: AFIPS Press, 1972, p.p. 1203–1224.
5. CWE/SANS Top 25 Most Dangerous Software Errors. URL: http://cwe.mitre.org/top25/.

В. В. Гуров, Д. В. Гуров, Г. Г. Новиков

РАЗРАБОТКА МЕТОДИКИ ОБУЧЕНИЯ БЕЗОПАСНОМУ ПРОГРАММИРОВАНИЮ

Подготовленность специалиста в технической области определяется совокупностью его знаний, умений и навыков («принцип ЗУН»). Это обусловливает необходимость не только дать студентам определенный набор знаний, но и сформировать у них определенные умения и навыки, характерные для осваиваемой области. Таким образом, в процессе обучения необходимо совмещать традиционные формы (лекции и семинарские занятия) с широким использованием технических средств.