Такая методика позволит не только упростить анализ шифров на основе задачи о рюкзаке впоследствии, но и выявить недостаточную стойкость к анализу у существующих систем.

СПИСОК ЛИТЕРАТУРЫ:

1. *Adleman L.* On Breaking Generalized Knapsack Public Key Cryptosystems // Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing. M., 1983. С. 402—412.
2. *Shamir A.* A Polynomial-time Algorithm for Breaking the Basic Merkle — Hellman Cryptosystem // Proceedings of the IEEE Symposium on Foundations of Computer Science. M., 1982. С. 145—152.
3. *Lai M. K.* Knapsack Cryptosystems: The Past and the Future. 2001.
4. *Merkle R., Hellman M.* Hiding Information and Signatures in Trapdoor Knapsacks // IEEE Transactions on Information Theory. Vol. IT-24. M., 1978. С. 525—530.
5. *Shamir A., Zippel R.* On the Security of the Merkle — Hellman Cryptographic Scheme // IEEE Transactions on Information Theory, vol. IT-26. M., 1998. С. 339—340.
6. *Kasahara M.* Construction of New Classes of Knapsack Type Public Key Cryptosystem Using Uniform Secret Sequence. M., 2012. — 8 с.
7. *Noro K., Kobayashi K.* Knapsack Cryptosystem on Elliptic Curves. M., 2009. — 6 с.
8. *Осипян В.* Разработка математических моделей систем передачи и защиты информации. М., 2006. — 371 с.
9. *Su P. C., Lu E. H., Henry K. C.* A Knapsack Public-key Cryptosystem Based on Elliptic Curves Discrete Logarithm // Applied Mathematics and Computation 168. M., 2005. С. 40—46

С. К. Марфенко, В. О. Чуканов

ПРИМЕНЕНИЕ МОДИФИЦИРОВАННОЙ ЭВРИСТИЧЕСКОЙ МОДЕЛИ НАДЕЖНОСТИ ДЛЯ ОЦЕНКИ УСТОЙЧИВОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В программно-аппаратных системах, которые обеспечивают постоянную возможность для работы клиентов, любой простой или сбой в работе может привести к критическим последствиям. В брокерских приложениях, системах автоматизации бизнес-процессов, где критически важным показателем является коэффициент готовности (вероятность нахождения системы в работоспособном состоянии в произвольный момент времени), каждый отказ ПО приводит к материальным потерям, потерям в качестве и другим потерям для той организации, которая предоставляет такую услугу [1]. По этой причине атака с целью нарушить целостность системы и лишить ее работоспособности может оказаться достаточно серьезной угрозой. Необходимо внимательно подходить к задаче предотвращения такого рода негативных воздействий на систему. Очевидно, что с повышением качества ПО и его надежности уменьшится количество отказов, как случайных, так и запланированных, которые вызываются злоумышленником. Для оценки устойчивости системы к таким атакам целесообразно применять аппарат моделей надежности программного обеспечения, который позволяет сделать оценки некоторых количественных характеристик ПО: количества ошибок, наработки на отказ, вероятности безотказной работы, коэффициента готовности.

В рамках данного доклада рассматривается модифицированная эвристическая модель надежности ПО как аппарат для определения показателей устойчивости. В основе этой модели лежит классическая интуитивная схема: рассматриваемый программный комплекс тестируется

независимо двумя группами тестировщиков. По результатам тестирования первая группа обнаруживает N_a ошибок, вторая обнаруживает N_b ошибок. Очевидно, что эти множества ошибок пересекаются и их пересечение дает множество ошибок N_{ab}, найденных как первой, так и второй группой (схема пересечения и вхождения множеств ошибок показана на рис. 1) [2].

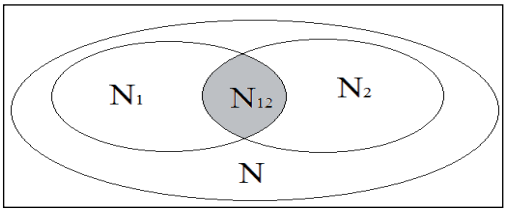

Рис. 1. Множество ошибок программы

Исходя из соображений независимости эффективности тестирования от того множества ошибок, которые исследуются первой или второй группой, можно сделать следующее суждение. Доля ошибок, найденных первой группой тестирования из всего множества ошибок, равна доле ошибок, найденных первой группой тестирования из множества ошибок, найденных второй группой [3]:

$N_a/N = N_{ab}/N_b$.

Отсюда можно получить общее количество ошибок в программном комплексе:

$N = (N_a*N_b)/N_{ab}$.

Эта схема не учитывает различий между разными классами ошибок и их влияния на работоспособность системы. Для того чтобы обойти это ограничение, предлагается использовать модифицированный алгоритм. В рамках этой модели все множество ошибок программного обеспечения разбивается на G групп по уровню их критичности и времени восстановления:

$N_1(t_{r1})$, $N_2(t_{r2})$, ..., $Ng(t_{rg})$.

Множества не пересекаются: для любой пары {i,j} верно $N_i \cap N_j = Ø$. При этом слияние этих множеств дает N — все ошибки программного обеспечения.

Для каждой из групп справедливы соображения простой интуитивной модели:

$N_k = (N_{ak}*N_{bk})/N_{abk}$ — для любого k от 1 до L.

Таким образом, для любой группы можно посчитать количество ошибок в исходной программе N_k и после их исправления $N'_k = N_k - N_{ak} - N_{bk} + N_{abk}$.

Следующая задача состоит в том, чтобы определить набор весовых коэффициентов $(w_1,w_2,...,w_g)$, по одному на группу. Для каждой из групп этот коэффициент показывает вероятность того, что данная ошибка приведет к отказу и соответствующему такому типу отказов простою системы:

$w_k \in [0;1]$.

Искомые значения могут быть приняты равными 1 для ошибок, гарантированно приводящих к отказу, могут быть получены на основе моделирования с входными наборами, используемыми с заданной или произвольной частотой. Полученные значения могут быть сами по себе использованы при построении систем высокой доступности. При оценке времени максимального простоя системы, исходя из общего количества ошибок i-го типа (N_i), вероятности отказа при возникновении такой ошибки (w_i), времени восстановления системы после ошибки такого класса (t_{ri}) и допущения, что каждая ошибка будет сразу же после обнаружения исправляться.

$\omega = (T_{общ} - (N_1*w_1*t_{r1} + N_2*w_2*t_{r2} + ... + N_l*w_l*t_{rg}))/T_{общ}$, где $T_{общ}$ — общее время эксплуатации системы.

Также общее количество ошибок некоторого типа может использоваться для определения коэффициентов в моделях надежности с более мощным математическим аппаратом и, в конечном счете, для определения вероятности безотказной работы и времени наработки на отказ.

Новизна предложенного подхода заключается в модификации существующей модели: разбиение ошибок на группы и введение весовых коэффициентов. В результате эффективность такого рода расчетов будет заметно выше, так как значимость каждой ошибки будет учтена и ошибки, не имеющие принципиального значения, будут отделены с точки зрения влияния на количественные характеристики надежности от тех, которые являются критическими.

СПИСОК ЛИТЕРАТУРЫ:

1. *Саттон М., Грин А., Амини П.* Fuzzing. Исследование уязвимостей методом грубой силы. М.: Символ-Плюс, 2009. — 560 с.
2. *Чуканов В. О.* Надежность программного обеспечения и аппаратных средств систем передачи данных атомных электростанций: Учебное пособие. М.: МИФИ, 2008. — 180 с.
3. *Майерс Г.* Надежность программного обеспечения. М.: Мир, 1980. — 350 с.

В. С. Матвеева, А. В. Мамаев

КРИМИНАЛИСТИЧЕСКИЙ ПОДХОД К АНАЛИЗУ ВРЕМЕННЫХ АТРИБУТОВ ФАЙЛОВ В ОПЕРАЦИОННОЙ СИСТЕМЕ СЕМЕЙСТВА MICROSOFT WINDOWS И ФАЙЛОВОЙ СИСТЕМЕ NTFS

Подмену временных атрибутов файлов можно сделать как вручную, переводя системные часы или внося изменения в служебные структуры файловой системы, так и с помощью специальных программ, которых существует достаточное количество. Но интересно то, что эти же функциональные возможности также прописывают у ряда вредоносных программ с целью ввести в заблуждение пользователя и отнести файл скорее к системному, чем к подозрительному. Таким образом, при просмотре свойств файла в ОС будут отображаться подмененные сведения. Но не все так просто с точки зрения компьютерной криминалистики. Для распознания факта подмены используются особенности файловой системы NTFS под управлением ОС семейства Microsoft Windows.

В файловой системе NTFS временные атрибуты файлов содержатся в файловой записи для каждого файла в главной файловой таблице (далее — MFT). И как ни странно, у файла их ровно 8, а не 3, как мы привыкли. За временные атрибуты отвечают две структуры: $STANDARD_INFORMATION и $FILE_NAME, каждая из которых содержит дату и время создания файла, последнего изменения файла, последнего доступа к файлу, а также дату и время последнего изменения сведений в файловой записи.

При подмене временных атрибутов файлов изменяют сведения в структуре $STANDARD_INFORMATION, так как именно она отображается пользователю в ОС. При этом структура $FILE_NAME остается неизменной и используется криминалистами для распознания факта подмены и восстановления оригинальных атрибутов файла. Однако это не так-то просто, так как существует большое количество особенностей, на основании которых изменяются временные атрибуты файла в зависимости от ОС и производимых действий. Поэтому проведена серия тестов для разных ОС семейства Microsoft Windows по выявлению условий изменения атрибутов в обеих