4. *Sun L., Ebringer T., Boztas S.* Hump-and-dump: efficient generic unpacking using an ordered address execution histogram. Department of Computer Science and Software Engineering, The University of Melbourne, Australia, 2008.
5. Программа для определения навесной защиты PEId. URL: http://cracklab.ru/download.php?action=get&n=NTYw.
6. Модуль для определения навесной защиты PESniffer из состава PETools. URL: http://cracklab.ru/download.php?action=get&n=MTU1.
7. Модуль для нахождения оригинальной точки входа Dream Of Every Reverser. URL: http://deroko.phearless.org/doer.html.
8. Модуль для нахождения оригинальной точки входа OEPFinder by Human. URL: http://www.exetools.com/forum/printthread.php?t=8841.
9. Программа для редактирования ресурсов Restorator. URL: http://www.bome.com
10. Язык для скриптов Lua. URL: http://www.lua.org.

С. В. Запечников (к. т. н., доцент)
Московский инженерно-физический институт (государственный университет)

КОНТРОЛЬ ЦЕЛОСТНОСТИ ФУНКЦИОНАЛЬНЫХ РЕСУРСОВ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ В РАСПРЕДЕЛЕННОЙ СРЕДЕ

В работе решается задача контроля функциональных ресурсов средств защиты информации (СЗИ). К контролируемым ресурсам относятся фрагменты файловой системы (ФС) и баз данных (БД), а также области оперативной и программируемой памяти, в которых содержатся объекты, реали-зующие функциональность СЗИ. Принципы решения задачи основаны на самоконтроле и взаимном контроле целостности выделенного множества объектов узлами распределенной компьютерной системы (РКС), функционирующими в составе СЗИ.

Введение

Корректная реализация протоколов и алгоритмов, обеспечивающих безопасность информации в компьютерных системах (КС), возможна лишь при условии обеспечения целостности самих ресурсов КС, ответственных за выполнение этих действий. Несмотря на наличие целого ряда программно-аппаратных средств контроля целостности ПО общего назначения, эта задача по-прежнему остается актуальной именно для СЗИ. Поскольку сами средства контроля целостности ресурсов с полным основанием могут быть отнесены к СЗИ, в решении этой задачи возникает специфика, связанная по сути с необходимостью обеспечивать самоконтроль и собственную безопасность СЗИ. Частным случаем рассматриваемой задачи является обеспечение целостности ресурсов средств криптографической защиты информации и средств управления криптографическими ключами, которые должны сохранять стойкость в условиях частичного разрушения и (или) компрометации системы.

Под функциональными ресурсами СЗИ будем понимать любые объекты КС, в которых содержатся объекты, реализующие те или иные функции СЗИ. Понятие «функциональные ресурсы» в рамках настоящей работы используется в противоположность понятию «информационные ресурсы», обозначающему данные, над которыми или с помощью которых СЗИ выполняют свои функции. Методы контроля целостности информационных ресурсов достаточно традиционны и хорошо разработаны: к ним, прежде всего, относятся коды, обнаруживающие и исправляющие ошибки, функции хэширования. Информационные ресурсы выступают для средств контроля целостности абсолютно пассивными и независимыми друг от друга объектами. Напротив, функциональные ресурсы организованы в систему, их число может быть весьма значительно, они распределены по множеству узлов РКС, а дополнительную сложность в их контроль вносит тот факт, что при выполнении процедур контроля они должны продолжать выполнять свои основные функции.

1. Постановка задачи и пути ее решения

Основные требования к методам решения задачи контроля функциональных ресурсов определяются следующими факторами.

1. Средства контроля целостности функциональных ресурсов не должны содержать централизованных элементов, таких, что выход из строя одного из них приводит к нарушению работоспособности средств контроля.

2. Контролируемая система может состоять из некоторого количества автономно функционирующих под управлением операционной системы узлов РКС, связанных между собой коммуникационными средствами; на каждом из них поддерживается ФС.

3. Контролируемыми элементами системы являются фрагменты ФС, содержащие исполняемый код программных модулей СЗИ и данные, критичные для функционирования системы, фрагменты системы директорий и БД, а также дампы областей оперативной и перезаписываемой (EEPROM) памяти с загруженным в них кодом программных модулей; количество подлежащих контролю объектов может быть весьма значительным.

4. Секретный ключевой материал узлов РКС не может распространяться между узлами либо храниться за пределами узла-владельца в силу отсутствия средств защищенной передачи сообщений и в целях минимизации возможностей его компрометации.

5. Работоспособность средств контроля должна сохраняться в условиях частичного разрушения РКС: отказа части узлов РКС, обрыва части каналов связи.

Известные из литературы методы контроля целостности оперативной памяти, файловых систем и баз данных могут быть разделены на три группы в соответствии с используемыми средствами решения задачи. Первый подход основан на построении бинарного дерева, листьями которого являются проверочные величины для контролируемых объектов — дерева Меркле, впервые предложенного в работе [1]. Он характеризуется простотой, не требует хранения секретных данных, удобен для контроля ФС, но требует очень больших вычислительных затрат при изменении хотя бы одного контролируемого элемента. Второй подход основан на применении китайской теоремы об остатках (КТО) для контроля хэш-кодов областей оперативной памяти [2]. Данный метод позволяет контролировать большие области памяти, характеризуется высокой степенью параллелизма операций, но отличается значительной вычислительной сложностью и требует хранения секретной информации. Третий подход основан на применении однонаправленных функций, например RSA [3], для дистанционного контроля целостности больших массивов информации. Он характеризуется очень высокими вычислительными затратами, должен применяться отдельно к каждому элементу (файлу, БД) и не предоставляет возможностей централизованного их контроля.

В настоящей работе для решения задачи контроля целостности функциональных ресурсов СЗИ предлагается использовать новый метод, построенный на базе сочетания трех элементов: ранее известного метода построения дерева Меркле, предложенной в настоящей работе модификации метода, основанного на КТО, и доказательной регистрации («аудита») событий, связанных с контролем функциональных ресурсов.

Решение базируется на следующих принципах:

— регулярность проведения контроля;

— самоконтроль собственных ресурсов узлов РКС с формированием открытых общедоступных проверочных величин;

— взаимный контроль открытых проверочных величин узлами РКС;

— доказательная регистрация всех событий, связанных с контролем ресурсов, с заданной степенью гранулярности;

— концентрация программного кода, реализующего средства контроля целостности, в небольшом числе контролируемых и защищенных от внешних воздействий модулей либо аппаратная (микропрограммная) их реализация.

Для решения поставленной задачи используется двухступенчатая система контроля целостности ресурсов СЗИ. Она состоит из механизмов самоконтроля ресурсов отдельных узлов РКС и механизма взаимного контроля ресурсов узлами — носителями компонентов СЗИ.

Для самоконтроля используется метод, основанный на КТО, который предоставляет удобные возможности выборочной или полной проверки целостности контролируемых объектов, но требует хранения некоторого количества секретной информации на каждом из узлов. Разрушение этой секретной информации приведет к некорректному функционированию механизмов контроля только на скомпрометированном узле и не повлияет на другие узлы.

Для взаимного контроля применяется метод, основанный на построении дерева Меркле, который характеризуется существенно более высокими вычислительными затратами по сравнению с методом, основанным на КТО, но не требует формирования и хранения секретных данных. Разрушение средств взаимного контроля на любом из узлов РКС приводит к исключению их из механизма взаимного контроля, но не влияет на функционирование этих средств на всех остальных узлах.

2. Самоконтроль ресурсов узлов

Пусть $I_1, \ldots, I_n$ — идентификаторы контролируемых узлов РКС, на которых инсталлированы средства СЗИ. На каждом из них выделяется набор контролируемых объектов, таких как файлы, загрузочные сектора дисков, таблицы размещения файлов, области оперативной памяти с исполняемым кодом программ. Обозначим через $M^{i,j}$ множество контролируемых объектов j-й подсистемы i-го узла РКС. Оно может быть представлено в виде конечного числа объектов: $M^{i,j} = \{M_1^{i,j}, M_2^{i,j}, \ldots, M_{n_{ij}}^{i,j}\}$, где $M_k^{i,j} \in \{0,1\}^*$, $k = \overline{1,n_{ij}}$, n_{ij} — количество контролируемых объектов j-й подсистемы i-го узла. Пусть $H : \{0,1\}^* \to \{0,1\}^{l_1}$ — криптографическая хэш-функция с трудно обнаруживаемыми коллизиями. Обозначим через $h_k^{i,j} = H(M_k^{i,j})$ хэш-коды объектов с номерами $k = \overline{1,n_{ij}}$.

Представим двоичные вектора длины l_2, полученные при вычислении хэш-кодов, в виде многочленов $h_k^{i,j}(z)$ над полем $Z_2[z]$, у которых $\deg h_k^{i,j}(z)=l_1$. Пусть $\rho = \{\rho_1(z), \rho_2(z), \ldots, \rho_q(z)\}$ — множество из $q \geq \max\limits_{i,j} n_{ij}$ неприводимых многочленов, таких, что $\deg \rho_i(z)=l_1$ для $\forall i=\overline{1,q}$. Для каждой пары (i, j) применим к множеству ρ отображение $\pi_{ij} : \rho \to \rho'_{ij}$, где $\rho'_{ij} = \{\rho_1(z), \rho_2(z), \ldots, \rho_{n_{ij}}(z)\}$. Эти отображения позволяют выбрать из множества всех неприводимых многочленов такое их количество, которого будет достаточно для вычисления контрольного кода контролируемого объекта.

Тестирование многочленов на неприводимость может быть выполнено методом, известным из работы [4]. Число неприводимых многочленов степени m над полем $Z_\rho[z]$ довольно велико и определяется по формуле: $N_\rho(m) \approx \rho^m/m$ [5. С. 155]. В связи с этим должно выполняться условие: $n_{ij} \leq N_\rho(m) = 2^{l_1}/l_1$, т. е., например, при использовании в качестве H хэш-функции с 256-битным хэш-кодом: $n_{ij} \leq 2^{256}/256 = 2^{248}$.

Далее применим КТО для многочленов к $h_k^{i,j}(z)$, $k=\overline{1,n_{ij}}$, в результате чего получим многочлен $x^{i,j}(z)$, такой, что $x^{i,j}(z) \equiv h_k^{i,j}(z) \bmod \rho_k(z)$, $\rho_k(z) \in \rho'_{ij}$, $k=\overline{1,n_{ij}}$, $\deg x^{i,j}(z) = l_2 < n_{ij}(l_1+1)$. Многочлен $x^{i,j}(z)$ сохраняется в защищенной памяти i-го узла РКС. КТО для многочленов порождает изоморфное отображение между $Z_2[z]/\rho_1(z)\ldots\rho_{n_{ij}}(z)$ и $Z_2[z]/\rho_1(z)+Z_2[z]/\rho_2(z)+\ldots+Z_2[z]/\rho_{n_{ij}}(z)$ [6. С. 405—407]. Поскольку преобразование КТО легко вычислимо в обе стороны, необходимо либо обеспечить секретность $x^{i,j}(z)$ в целях исключения подстановки противником хэш-кодов контролируемых объектов, либо использовать вместо бесключевой

хэш-функции H криптографическую хэш-функцию с ключом $H_{K_i} : \{0.1\}^* \times \{0.1\}^{|K_i|} \to \{0.1\}^{l_1}$ и обеспечивать секретность ключа K_i.

В дальнейшем целостность любого объекта $M_k^{i,j}$ может быть подтверждена путем проверки составленного для него сравнения КТО: $x^{i,j}(z) \equiv h_k^{i,j}(z) \bmod \rho_k(z)$.

При использовании КТО для целых чисел по попарно взаимно простым модулям для вычислений необходимо применять алгоритм Гаусса [5. С. 68, алгоритм 2.121] либо алгоритм Гарнера [5. С. 612, алгоритм 14.71]. Применение КТО для многочленов приводит к меньшим вычислительным затратам по сравнению с методами из работы [2], поскольку для вычислений может быть применен быстрый алгоритм для КТО по системе неприводимых многочленов [7. С. 289, алгоритм 10.22], который характеризуется сложностью порядка $O(M(n)\log n)$операций в поле, где $M(n)$ — время выполнения одной операции умножения, n— степень многочлена.

Далее из секретного многочлена $x^{i,j}(z)$ с помощью той же бесключевой хэш-функции H или хэш-функции с ключом H_{K_i} вычисляется открытый контрольный код $y^{i,j}(z)=h(x^{i,j}(z))$.

Подобная процедура повторяется для контрольных кодов $y^{i,j}(z)$ каждой j-й подсистемы i-го узла. После применения к ним КТО для многочленов получим многочлен $x^i(z) \equiv y^{i,j}(z) \bmod \rho_k(z)$, $\rho_k(z) \in P_i'$, $k=\overline{1,n_i}$, $\deg x^i(z)=l_3<n_i(l_2+1)$, где $P_i'=\{\rho_1(z), \rho_2(z), ..., \rho_{n_i}(z)\}$, n_i — количество контролируемых подсистем i-го узла.

В дальнейшем целостность любой j-й подсистемы может быть подтверждена путем проверки составленного для нее сравнения КТО: $x^i(z) \stackrel{?}{\equiv} y^{i,j}(z) \bmod \rho_k(z)$.

Далее из секретного многочлена $x^i(z)$ с помощью той же бесключевой хэш-функции H или хэш-функции с ключом H_{K_i} вычисляется открытый контрольный код $y^i(z)=h(x^i(z))$, интерпретируемый впоследствии как двоичная строка $Y^i \in \{0,1\}^{l_3}$. Он может передаваться другим узлам РКС для взаимного контроля целостности.

Описанная выше процедура самоконтроля узла РКС должна осуществляться регулярно. Частота проверок определяется политикой безопасности КС. Факт получения величины Y^i при выполнении процедуры контроля в момент времени t должен фиксироваться системой доказательной регистрации событий, для чего эта величина вместе с меткой времени подписывается секретным ключом электронной цифровой подписи (ЭЦП) узла: $S_i(t)=\mathrm{Sign}_{sk_i}(Y^i,t)$, $i=\overline{1,n}$, и помещается в БД регистрируемых событий. Этого достаточно для доказательной регистрации фактов проверки целостности ПО на уровне отдельных узлов РКС.

При необходимости более глубокого контроля аналогичным образом может осуществляться доказательная регистрация фактов проверки целостности отдельных подсистем СЗИ на каждом узле. Для этого выработанные при проверке ПО подсистем контрольные коды подписываются секретными ключами ЭЦП подсистем СЗИ на данном узле: $S_{ij}(t)=\mathrm{Sign}_{sk_{ij}}(y^{i,j},t)$, $i=\overline{1,n}$, $j=\overline{1,m_i}$, где m_i — количество контролируемых подсистем i-го узла РКС. Глубина контроля может увеличиваться и далее, вплоть до уровня отдельных объектов: $S_{ijk}(t)=\mathrm{Sign}_{sk_{ijk}}(h_i^{i,k},t)$, $i=\overline{1,n}$, $j=\overline{1,m_i}$, $k=\overline{1,l_j}$, где l_j — количество контролируемых объектов j-й подсистемы i-го узла.

Доказательная регистрация событий позволяет в любой момент времени воспроизвести величину контрольного кода, полученного для каждого контролируемого объекта в момент времени t, и путем проверки ЭЦП убедиться в корректности процедуры контроля.

3. Взаимный контроль ресурсов узлов

На втором уровне для проверки аутентичности выработанных отдельными узлами РКС контрольных кодов предлагается использовать метод построения дерева Меркле. Пусть I_1, ..., I_n — узлы РКС, подлежащие контролю. Поименуем каждый из них элементом кода Грея $g_i^{(m)}$, где $m=\lceil \log_2 n \rceil$. Все элементы кода Грея различны [8], поэтому они могут служить идентификаторами узлов РКС. Выберем криптографическую хэш-функцию h с трудно обнаруживаемыми коллизиями и построим дерево Меркле следующим образом.

1. Поставим в соответствие каждому узлу I_i лист дерева Меркле $V_i^{(0)}$. Каждый из листьев дерева пометим величиной $g_i^{(m)}\|Y^i$, $i=\overline{1,n}$, что исключает вероятность совпадения величин, соответствующих различным листьям дерева.

2. Пометим дуги, исходящие из листьев $V_i^{(0)}$, символами $h(g_i^{(m)}\|Y^i)$, $i=\overline{1,n}$.

3. Построим внутренние вершины $V_1^{(1)}$, ..., $V_{\lceil n/2\rceil}^{(1)}$ дерева Меркле таким образом, что для $V_1^{(1)}$ входящими дугами являются $h(g_1^{(m)}\|Y^1)$ и $h(g_2^{(m)}\|Y^2)$, а исходящей — $h(h(g_1^{(m)}\|Y^1)\|h(g_2^{(m)}\|Y^2))$, для $V_2^{(1)}$ входящими дугами являются $h(g_3^{(m)}\|Y^3)$ и $h(g_4^{(m)}\|Y^4)$, а исходящей — $h(h(g_3^{(m)}\|Y^3)\|h(g_4^{(m)}\|Y^4)$, и т. д., для $V_{\lceil n/2\rceil}^{(1)}$ входящими дугами являются $h(g_{n-1}^{(m)}\|Y^{n-1})$ и $h(g_n^{(m)}\|Y^n)$, а исходящей — $h(h(g_{n-1}^{(m)}\|Y^{n-1})\|h(g_n^{(m)}\|Y^n))$, если n — четное, и $h(g_n^{(m)}\|Y^n)$, а исходящей — $h(h(g_n^{(m)}\|Y^n)$, если n — нечетное.

4. Будем повторять шаг (3), увеличивая каждый раз индекс k на единицу, где k — длина максимального пути от листьев дерева до вершин $V_1^{(k)}$, ..., $V_{\lceil n/2k\rceil}^{(k)}$, пока для некоторого $k\geq 1$ число вершин не станет равным двум. Если левая и правая дуги, входящие в какую-либо внутреннюю вершину $V_i^{(k)}$, помечены метками $h_{i_1}^{(k)}$ и $h_{i_2}^{(k)}$ соответственно, то исходящую из нее дугу помечаем меткой $h(h_{i_1}^{(k)}\|h_{i_2}^{(k)})$.

5. Построим корень дерева Меркле: если дуги, полученные для последних двух вершин $V_i^{(k)}$ на шаге (4), обозначены u_1 и u_2, то корневой вершине присвоим метку $R = h(u_1\|u_2)$.

Для каждого листа L_i существует единственный путь, соединяющий его с корнем. При условии обеспечения аутентичности метки корня R можно построить путь аутентификации, состоящий из последовательности $\lceil \log_2 n\rceil$ меток, присвоенных дугам вдоль этого пути. Проверка пути аутентификации в силу свойств дерева Меркле [1] гарантирует аутентичность каждого контрольного кода Y^i, соответствующего листу L_i. Если проверка хотя бы одной метки заканчивается с отрицательным результатом, величина Y^i не может быть признана аутентичной. Аутентичность метки корня R в рассматриваемой задаче предлагается обеспечивать за счет трех факторов: массовости рассылки по системе, периодического повтора и общедоступности.

Понятие об аутентичности контрольных кодов, обеспечиваемой деревом Меркле, совпадает с введенным в [9] понятием аутентичности как фиксации функциональных зависимостей между некоторыми величинами. В данном случае контролю подлежат не только Y^i, но и их соответствия идентификаторам узлов РКС, для чего выстраиваются функциональные зависимости первого рода в виде бинарного дерева.

Предложенный метод контроля функциональных ресурсов СЗИ реализуется с помощью двух алгоритмов: алгоритма формирования эталонных значений контрольных кодов, выполняемого однократно в начале жизненного цикла СЗИ либо в случае модификации его ПО, и алгоритма периодического контроля целостности функциональных ресурсов.

Алгоритм 1 *(Формирование эталонных значений контрольных кодов функциональных ресурсов СЗИ).*

1. Выбрать множество контролируемых объектов $M^{i,j}$ на узлах РКС с инсталлированными компонентами СЗИ. Пусть I — подмножество индексов узлов РКС, на которых расположены контролируемые объекты, J^i — подмножество индексов контролируемых подсистем на i-м узле.

2. Выбрать криптографическую хэш-функцию H с длиной хэш-кода l. Сгенерировать множество $P=\{\rho_1(z), \rho_2(z), ..., \rho_q(z)\}$, состоящее из $q \geq \max_i |J_i|$ неприводимых многочленов порядка l.

3. Для каждого контролируемого объекта $M_k^{i,j}$ вычислить хэш-код $h_k^{i,j}$, представить его в виде многочлена $h_k^{i,j}(z)$.

4*. Для каждого контролируемого объекта i-му узлу подписать его хэш-код вместе с меткой времени соответствующим секретным ключом подсистемы доказательной регистрации

событий: $S_{ijk}(t)=\mathrm{Sign}_{sk_{ijk}}(h_i^{j,k},t)$. Сохранить в БД подсистемы доказательной регистрации событий запись $\lfloor i,\ j,\ k,\ h_k^{i,j},\ t, S_{ijk}(t)\ \rfloor$.

5. Применить алгоритм КТО для многочленов ко всем $h_k^{i,j}(z)$ j-й подсистемы i-го узла для получения многочлена $x^{i,j}(z)$: $x^{i,j}(z)\equiv h_k^{i,j}(z)\bmod \rho_k(z)$, $\rho_k(z)\in P'_{ij}$, $k=\overline{1,|M^{i,j}|}$, $\deg x^{i,j}(z)=l_2<|M^{i,j}|(l_1+1)$, $P'_{ij}=\{\rho_1(z), \rho_2(z), ..., \rho_{|M^{i,j}|}(z)\}$. Сохранить $x^{i,j}(z)$ в защищенной памяти i-го узла. Вычислить $y^{i,j}(z)=\mathrm{h}(x^{i,j}(z))$.

6*. Для каждой j-й контролируемой подсистемы i-му узлу подписать контрольный код $y^{i,j}$ вместе с меткой времени соответствующим секретным ключом подсистемы доказательной регистрации событий: $S_{ij}(t)=\mathrm{Sign}_{sk_{ijk}}(y^{i,j},t)$. Сохранить в БД подсистемы доказательной регистрации событий запись $\lfloor i,\ j,\ k,\ h_k^{i,j},\ t, S_{ijk}(t)\ \rfloor$.

7. Применить алгоритм КТО для многочленов ко всем $y^{i,j}(z)$ i-го узла для получения многочлена $x^i(z)$: $x^i(z)\equiv y^{i,j}(z)\bmod \rho_k(z)$, $\rho_k(z)\in P'_i$, $k=\overline{1,|J_i|}$, $\deg x^i(z)=l_3<|J_i|(l_2+1)$, $P'_i=\{\rho_1(z), \rho_2(z), ..., \rho_{|J_i|}(z)\}$. Сохранить $x^i(z)$ в защищенной памяти i-го узла. Вычислить $y^i(z)=\mathrm{h}(x^i(z))$. Представить $y^i(z)$ в виде двоичного вектора $Y^i\in\{0,1\}^{l_3}$.

8. Каждому i-му узлу подписать контрольный код Y^i вместе с меткой времени соответствующим секретным ключом подсистемы доказательной регистрации событий: $S_i(t)=\mathrm{Sign}_{sk_i}(Y^i,\ t)$. Сохранить в БД подсистемы доказательной регистрации событий запись $\lfloor i,\ Y^i,\ t,\ S_i(t)\rfloor$.

9. Каждому i-му узлу из множества I разослать Y^i всем остальным узлам и принять Y^r от всех r-х узлов, $r\neq i$. Если от какого-либо r-го узла принято несколько различных значений Y^r или не принято ни одного значения, то он исключается из множества I.

10. Каждому узлу из множества I вычислить $h(g_r^{(m)}\|Y^r)$ для $\forall r\in I$ и построить дерево Меркле по методу, описанному выше.

11. Каждому узлу из множества I разослать R корневую метку дерева Меркле всем остальным узлам и принять значения корневой метки от всех других узлов из множества I. Если от какого-либо узла будет принято несколько значений метки, то принять в качестве корневой метки R то значение, которое встречается наибольшее число раз. Если от какого-либо узла не принято ни одного значения, то он исключается из множества I.

12. Каждому узлу из множества I подписать корневую метку R вместе с меткой времени своим секретным ключом подсистемы доказательной регистрации событий: $S_i^R(t)=\mathrm{Sign}_{sk_i}(R,\ t)$. Сохранить в БД подсистемы доказательной регистрации событий следующую запись: $\lfloor i,\ R,\ t,\ S_i^R(t)\rfloor$.

Конец алгоритма.

***Алгоритм 2** (Периодический контроль целостности функциональных ресурсов СЗИ).*

1. Выбрать подмножество контролируемых объектов $M_*^{i,j}\subseteq M^{i,j}$ на узлах РКС с инсталлированными компонентами СЗИ. Пусть $I_*\subseteq I$ — подмножество индексов узлов РКС, на которых расположены контролируемые объекты, $J_*^i\subseteq J^i$ — подмножество индексов контролируемых подсистем на i-м узле.

2. Для каждого проверяемого объекта $M_k^{i,j}$ вычислить хэш-код $h_k^{i,j}$, представить его в виде многочлена $h_k^{i,j}(z)$.

3*. Для каждого проверяемого объекта i-му узлу подписать его хэш-код вместе с меткой времени соответствующим секретным ключом подсистемы доказательной регистрации событий: $S_{ijk}(t)=\mathrm{Sign}_{sk_{ijk}}(h_i^{j,k},\ t)$. Сохранить в БД подсистемы доказательной регистрации событий запись $\lfloor i,\ j,\ k,\ h_k^{i,j},\ t,\ S_{ijk}(t)\rfloor$.

4. Проверить выполнение сравнения КТО для всех $h_k^{i,j}(z)$ каждой j-й подсистемы каждого i-го узла из множества I_*:

$$x^{i,j}(z) \stackrel{?}{\equiv} h_k^{i,j}(z) \bmod \rho_k(z), \quad (1)$$

где $\rho_k(z) \in P'_{ij}$, $P'_{ij}=\{\rho_1(z), \rho_2(z), ..., \rho_{|M_{i,j}|}(z)\}$. Если для какого-либо объекта $M_k^{i,j}$ сравнение (1) не выполнено, то выдать сообщение о нарушении целостности объекта $M_k^{i,j}$ и исключить i-й узел из множества I_*.

5. Каждому i-му узлу из множества I_* извлечь из защищенной памяти секретный контрольный код $x^{i,j}(z)$ и вычислить $y^{i,j}(z)=h(x^{i,j}(z))$.

6*. Для каждой j-й контролируемой подсистемы i-му узлу подписать контрольный код $y^{i,j}$ вместе с меткой времени соответствующим секретным ключом подсистемы доказательной регистрации событий: $S_{ij}(t)=\mathrm{Sign}_{sk_{ij}}(y^{i,j},t)$. Сохранить в БД подсистемы доказательной регистрации событий запись $\lfloor i,\ j,\ y^{i,j},\ t,\ S_{ij}(t) \rfloor$.

7. Проверить выполнение сравнения КТО для всех $y^{i,j}(z)$ каждого i-го узла из множества I_*:

$$x^i(z) \stackrel{?}{\equiv} y^{i,j}(z) \bmod \rho_k(z), \quad (2)$$

где $\rho_k(z) \in P'_i$, $P'_i=\{\rho_1(z), \rho_2(z), ..., \rho_{|J_i|}(z)\}$. Если для какого-либо контрольного кода $y^{i,j}$ сравнение (2) не выполнено, то выдать сообщение о нарушении целостности ресурсов j-й подсистемы i-го узла и исключить его из множества I_*.

8. Каждому i-му узлу из множества I_* извлечь из защищенной памяти секретный контрольный код $x^i(z)$и вычислить $y^i(z)=h(x^i(z))$. Представить $y^i(z)$ в виде двоичного вектора $Y^i \in \{0,1\}^{l_3}$.

9. Каждому i-му узлу подписать контрольный код Y^i вместе с меткой времени соответствующим секретным ключом подсистемы доказательной регистрации событий: $S_i(t)=\mathrm{Sign}_{sk_i}(Y^i,t)$. Сохранить в БД подсистемы доказательной регистрации событий запись $\lfloor i,\ Y^i,\ t,\ S_i(t) \rfloor$.

10. Каждому i-му узлу из множества I_* разослать Y^i всем остальным узлам и принять Y^r от всех r-х узлов, $r \neq i$. Если от какого-либо r-го узла принято несколько различных значений Y^r или не принято ни одного значения, то он исключается из множества I_*.

11. Каждому узлу из множества I_* вычислить $h(g_r^{(m)} \| Y^r)$ для $\forall r \in I_*$ и пересчитать дерево Меркле по методу, описанному выше.

12. Каждому узлу из множества I_* разослать корневую метку R дерева Меркле всем остальным узлам и принять значения корневой метки от всех других узлов из множества I_*. Если от какого-либо узла будет принято несколько различных значений метки или не будет принято ни одного значения, то он исключается из множества I_*.

13. Каждому узлу из множества I_* подписать корневую метку R вместе с меткой времени своим секретным ключом подсистемы доказательной регистрации событий: $S_i^R(t)=\mathrm{Sign}_{sk_i}(R,t)$. Сохранить в БД подсистемы доказательной регистрации событий следующую запись: $\lfloor i,\ R,\ t,\ S_i^R(t) \rfloor$. Выдать подтверждение целостности ресурсов.

Конец алгоритма.

Обозначенные звездочкой (*) шаги (4), (6) алгоритма 1 и шаги (3), (6) алгоритма 2 являются необязательными: их наличие либо отсутствие будет зависеть от принятой в СЗИ глубины контроля целостности ресурсов.

Подписываемые хэш-коды, контрольные коды и метки времени могут не сохраняться в БД подсистемы доказательной регистрации событий, если для их подписания применяется схема ЭЦП с восстановлением сообщений, например ISO/IEC 9796.

В случае модификации ПО на одном или нескольких узлах I_j осуществляется пересчет эталонных значений Y^i для этих узлов и, как следствие, пересчет контрольных величин вдоль тех ветвей дерева Меркле, которые содержат в себе путь от вершины, содержащей Y^i, до корня,

содержащего R. Для этого необходимо повторно выполнить алгоритм 1, причем для неизменяемых модулей СЗИ можно взять прежние эталонные значения контрольных кодов. Предполагая, что модификация ПО происходит сравнительно редко, а количество узлов РКС относительно невелико, процедура пересчета дерева Меркле не потребует чрезмерно больших вычислительных затрат. Такая процедура должна проводиться под контролем администратора системы и сопровождаться обязательной доказательной регистрацией всех составляющих ее событий.

4. Заключение

В работе предложен двухступенчатый метод контроля целостности функциональных ресурсов СЗИ, основанный на самоконтроле и взаимном контроле компонентами СЗИ своих подсистем, файлов, дампов оперативной памяти и иных контролируемых объектов. Метод самоконтроля основан на китайской теореме об остатках, метод взаимного контроля — на построении бинарного дерева Меркле из контрольных кодов, формируемых отдельными узлами системы. Разработаны алгоритмы, реализующие этот метод.

СПИСОК ЛИТЕРАТУРЫ:

1. *Merkle R. C.* A certified digital signature // Adv. in Cryptology — Proc. of CRYPTO'89, LNCS Vol. 435. Springer-Verlag, 1990. P. 218–238.
2. *Potlapally N. R.* Verifying data integrity with few queries to untrusted memory [электронный ресурс]. URL: http://eprint.iacr.org/2007/027.
3. *Ateniese G., Burns R., Curtmola R., Herring J., Kissner L., Peterson Z., Song D.* Provable data possession at untrusted stores [электронный ресурс]. URL: http://eprint.iacr.org/2007/202.
4. *Ben-Or M.* Probabilistic algorithms in finite fields // Proc. of the IEEE 22nd Annual Symposium on Foundations of Computer Science, 1981. P. 394–398.
5. *Menezes A. J., van Oorschot P. C., Vanstone S. A.* Handbook of Applied Cryptography [электронный ресурс]. URL: www.cacr.math.uwaterloo.ca/hac.
6. *Акритас А.* Основы компьютерной алгебры с приложениями: Пер. с англ. М.: Мир, 1994. — 544 с.
7. *Von zur Gathen J., Gernard J.* Modern computer algebra. Cambridge University Press, 1999.
8. *Ноден П., Китте К.* Алгебраическая алгоритмика (с упражнениями и решениями): Пер. с франц. М.: Мир, 1999.
9. *Запечников С. В.* Методы и алгоритмы, обеспечивающие аутентичность ключевого материала криптосистем в условиях воздействия дестабилизирующих факторов // Материалы X Международной научно-практической конференции «Информационная безопасность». Ч. 2. Таганрог: Изд-во ТТИ ЮФУ, 2008. С. 146–149.