DOI: 10.15514/ISPRAS-2026-38(4)-19

Об оценке и метриках защищенности операционных систем

В.В. Кулямин, ORCID: 0000-0003-3439-9534 <kuliamin@ispras.ru>
А.К. Петренко, ORCID: 0000-0001-7411-3831 <petrenko@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
Московский государственный университет имени М.В. Ломоносова,
Россия, 119991, Москва, Ленинские горы, д. 1.
Национальный исследовательский университет «Высшая школа экономики» (НИУ ВШЭ),
101000, Россия, г. Москва, ул. Мясницкая, д. 20.

Аннотация. Наряду с задачами обеспечения защищенности программных систем и исследованием методов анализа корректности и устойчивости к вредоносным воздействиям встают задачи оценки и сопоставления степени защищенности программных систем. В общей постановке эти задачи, вероятно, не имеют решения, которое получило бы признание и имело бы практическую пользу. Практически полезной может быть постановка задачи, где сужается класс сопоставляемых систем и намечаются потенциальные сценарии использования соответствующих метрик и методов оценки. Данная статья предлагает подход к разработке методов оценки защищенности операционных систем как программного продукта и рассматривает некоторые сценарии их использования. Предлагаемый подход по сути является таксономией для предметной области проблем, методов и средств защиты операционных систем, то есть предлагает скорее качественные оценки, а не количественные, однако наличие достаточно полной и хорошо структурированной таксономии существенно упрощает разработку количественных метрик, что показано в недавних публикациях, приведенных в обзоре.

Ключевые слова: защищенные операционные системы; процессы безопасной разработки программного обеспечения; архитектуры операционных систем; техники усиления защиты программного обеспечения.

Для цитирования: Кулямин В.В., Петренко А.К. Об оценке и метриках защищенности операционных систем. Труды ИСП РАН, 2026, том 38, вып. 4, часть 2, стр. 67-80. DOI: 10.15514/ISPRAS-2026-38(4)-19.

Благодарности. Авторы выражают глубокую признательность участникам круглого стола «Методики оценки степени защищенности операционных систем», который проходил в ИСП РАН в июне 2026 года: Александру Попову (Positive Technologies), Дмитрию Пономареву (ФОБОС-НТ), Анне Мелеховой, Александру Аршинову, Сергею Рогачеву (Лаборатория Касперского), Юрию Солоделову (ГосНИИАС), Сергею Аносову и Валерию Кащееву (Открытая мобильная платформа), Владимиру Тележникову (Astra Linux), Денису Медведеву (Базальт СПО) и Евгению Баскову, Денису Ефремову, Сергею Игнатову, Вадиму Мутилину и Алексею Хорошилову (ИСП РАН). Многие соображения участников круглого стола о метриках оценки защищенности, способах их использования и подходах к регламентации этой деятельности были учтены при написании данной статьи. Авторы выражают отдельную благодарность Сергею Игнатову и Денису Ефремову за их вклад в расширение обзора литературы.

On the assessment and metrics of operating system security

V.V. Kuliamin, ORCID: 0000-0003-3439-9534 <kuliamin@ispras.ru>
A.K. Petrenko, ORCID: 0000-0001-7411-3831 <petrenko@ispras.ru>

V.P. Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia.
National Research University «Higher School of Economy»,
20, Myasnitskaya st., Moscow, 101000, Russia.

Annotation. Along with the tasks of ensuring the security of software systems and the study of methods for analyzing correctness and resistance to malicious influences, the tasks of evaluating and comparing the degree of security of software systems arise. In the general formulation, these tasks probably do not have a solution that would be recognized and would have practical benefits. It can be practically useful to formulate a problem where the class of comparable systems is narrowed down and potential scenarios for using appropriate metrics and evaluation methods are outlined. This article offers an approach to the development of methods for assessing the security of operating systems as a software product and examines some scenarios for their use. The proposed approach is essentially a taxonomy for the subject area of problems, methods and means of protecting operating systems, that is, it offers qualitative rather than quantitative assessments, however, the presence of a sufficiently complete and well-structured taxonomy greatly simplifies the development of quantitative metrics, as shown in recent publications cited in the review.

Keywords: secure operating systems; secure software development processes; OS architectures; hardening.

For citation: Kuliamin V.V., Petrenko A.K. On the assessment and metrics of the security of operating systems. Proceedings of the ISP RAS, 2026, volume 38, issue 4, part 2, pp. 67-80. DOI: 10.15514/ISPRAS-2026-38(4)-19.

Acknowledgements. The authors express their deep gratitude to the participants of the round table "Methods for assessing the degree of security of operating systems", which was held at the ISP RAS in June 2026: Alexander Popov (Positive Technologies), Dmitry Ponomarev (PHOBOS-NT), Anna Melekhova, Alexander Arshinov, Sergey Rogachev (Kaspersky Lab), Yuri Solodelov (GosNIIAS), Sergey Anosov and Valery Kashcheev (Open Mobile Platform), Vladimir Teleznikov (Astra Linux), Denis Medvedev (Basalt PDF) and Evgeny Baskov, Denis Efremov, Sergey Ignatov, Vadim Mutilin and Alexey Khoroshilov (ISP RAS). Many considerations about security assessment metrics, how to use them, and approaches to regulating these activities were taken into account when writing this article.
In addition, special thanks to Sergey Ignatov and Denis Efremov for their contribution to the expansion of the literature review.

1. Введение. О предмете исследования

Обеспечение защищенности программных и программно-аппаратных комплексов со временем становится все более важным. По этой причине расширяется набор методов и технологий разработки и поддержки программных систем, в частности технологий, которые поддерживают разработку безопасного программного обеспечения (РБПО). Разработка и внедрение таких технологий идет параллельно с развитием регуляторной базы. Необходимость таких стандартов, по крайней мере в тех областях, где информационная безопасность и устойчивость программно-аппаратных систем критичны, также не вызывает сомнений.

При этом наряду с задачами обеспечения защищенности программных систем и исследованием методов анализа корректности и устойчивости к вредоносным воздействиям встают задачи оценки и сопоставления степени защищенности программных систем. В общей постановке эти задачи, вероятно, не имеют решения, которое получило бы признание и имело бы практическую пользу. Практически полезной может быть постановка задачи, где

сужается класс сопоставляемых систем и намечаются потенциальные сценарии использования соответствующих метрик и методов оценки. В данной работе мы рассматриваем только методы оценки защищенности операционных систем. Операционные системы (ОС) в целом также являются слишком широким классом программного обеспечения (ПО), сюда входят небольшие ОС для встроенных систем (характерный размер 10-20 тыс. строк на языке программирования Си), ядра универсальных ОС (например, текст ядра ОС Linux сейчас содержит более 40 млн строк на Си) и дистрибутивы ОС Linux или Windows, которые содержат сотни миллионов строк программ на разных языках программирования. Кроме того, даже в рамках группы ОС примерно одинакового размера могут быть ОС, построенные по разным архитектурным принципам (например, монолитные и микроядерные), которые сопоставлять в плане обеспечения защищенности сложно (возможно, стоило бы ограничить рассмотрение теми компонентами кода, которые могут быть выполнены в привилегированном режиме). Мы намеренно оставляем границы объекта оценки несколько размытыми, чтобы можно было вести речь о сопоставлении ОС, построенных на достаточно разных принципах.

При оценке защищенности ОС речь идет не о защищенности конкретной инсталляции системы с фиксированными (или слабо меняющимися) окружением и внутренним наполнением. Защищенность ОС означает возможность (а также аккуратность и трудоемкость/стоимость) защиты разнообразных систем на ее основе с варьирующимися внешним окружением и внутренним наполнением (приложениями и данными) в различных сценариях их применения. Разнообразие конкретных атак, которым такие системы могут подвергаться очень велико, успешность противостояния им в большой мере зависит от корректности конфигурации систем защиты, аккуратных действий пользователей, регулярных усилий по администрированию, а также своевременной и продуманной поддержки, оказываемой разработчиками ОС. Для оценки защищенности ОС, развернутой в конкретном окружении программно-аппаратного комплекса, вполне применим метод на основе оценки общей трудоемкости или общей стоимости успешной атаки. Кроме того, в контексте развернутой ОС защищенность в большой мере определяется не ОС как программный продукт, а ОС как объект эксплуатации. Понятно, что важную роль в обеспечении защищенности в этом случае будут играть квалификация персонала и зрелость процессов эксплуатации и поддержки.

В рамках нашей работы представляется вполне корректным сфокусироваться на характеристиках защищенности ОС как программного продукта или как платформы, на основе которой при конкретной инсталляции в конкретном окружении ОС работает и эксплуатируется. Тем самым мы будем анализировать возможности, которые предоставляет ОС-платформа для обеспечения защиты, и оценивать, насколько сложны и эффективны данные средства.

2. О выборе метрик оценки защищенности

Начнем с обсуждения желательных свойств метрик защищенности.

- Адекватность.

  Метрика должна быть построена на основе показателей (и рассуждений), имеющих как можно более прямое отношение к защищенности и демонстрирующих какой-то важный ее аспект. Чем более далеко измеряемое свойство от свойств защиты (хотя, возможно, оно и влияет на них), тем тщательней стоит рассматривать более близкие альтернативы, и тем более аккуратной и продуманной должна быть аргументация о включении его в рассмотрение в связи с защищенностью (в том числе, и об отсутствии полезных альтернатив).

- Простота и понятность.

  Довольно часто, рассуждая о метриках, требуют, чтобы они были «простыми и

понятными». Однако простота и понятность измеряющего инструмента во многом определяется измеряемыми им свойствами (их сложностью) и предметами (их характеристиками и масштабами). Линейка, измеряющая длину не слишком больших и не слишком маленьких объектов проще, чем микрометр для более точного обмера достаточно маленьких предметов, а тот гораздо проще, чем интерферометр для измерения космических объектов и расстояний между ними.

Есть достаточно давно сложившийся консенсус, что свойство защищенности программной системы в целом достаточно комплексно, и аккуратно оценить его можно только на основе анализа большого количества конкретных характеристик. Поэтому очень наивным представляется ожидание от значимых метрик защищенности наглядности и простоты для не слишком погруженного в данную область человека.

Что разумно ожидать и требовать, так это возможности объяснить достаточно грамотному в области защиты ПО человеку (по сути – эксперту), что именно данная метрика измеряет и как именно. Так, чтобы он мог согласиться, что это вполне разумно измерять в связи с защищенностью, и мог, при наличии необходимого технического инструментария, сам собрать нужные данные и провести соответствующие измерения. Рационально достижимое свойство «простоты» выражается ровно в этом требовании (как бы разочаровывающе это ни выглядело для не-экспертов).

- Комплексность.

  Как уже отмечалось, защищенность – довольно сложное свойство, и для его оценки нужно рассматривать много разнородных конкретных характеристик системы. Поэтому опять же, довольно наивно ожидать, что его можно выразить одним числом во всей полноте. Гораздо разумнее предположить, что метрика защищенности в целом будет выражаться как набор чисел, оценивающих различные важные ее аспекты (не сводимые друг к другу).

  При этом однако, даже отдельные (но не совсем конкретные) аспекты защищенности, типа устойчивости к атакам или способности поддерживать конфиденциальность данных, имеют схожую природу, и поэтому тоже лучше описываются наборами чисел. Однако, сведение оценки защищенности к набору из многих десятков различных чисел выглядит непрактично – ориентироваться в таких наборах будет слишком тяжело. Поэтому разумно в некоторых местах использовать интегральные метрики, объединяющие несколько разнородных показателей в одно число. Их использование, однако, порождает новые проблемы.

  - Шкалы сводимых в интегральную метрику показателей должны быть сопоставимы. Вычисление среднего от двух чисел, одно из которых меняется от 1 до 10^6, а другое от 3 до 5 дает малозначимые значения. Хотя довольно часто разнородные вещи на практике удобнее оценивать по несколько различающимся шкалам.
  - Используемые показатели должны быть независимы, по крайней мере, важно, чтобы корреляция между ними отсутствовала (хотя для конкретной ОС некоторые из них могут оказаться связанными из-за используемых средств разработки или специфической архитектуры), иначе вклад таких коррелированных показателей в общую интегральную метрику будет слишком высок и затмит влияние других важных факторов.
  - Сведение вместе разнородных показателей обычно основывается на вычислении среднего, однако, поскольку они разнородны (и представляют как бы величины с разными единицами измерения), для вычисления среднего арифметического необходимы веса.

Расстановка весов в наборе показателей представляет собой отдельную проблему: разные эксперты могут отстаивать различные наборы весов, часто вполне обоснованно (потому что они по-разному оценивают важность вкладов отдельных показателей в общую метрику). Иногда в разных контекстах полезно использовать разные наборы весов, переоценивая значимость того или иного показателя в общую защищенность в зависимости от сценариев использования. Однако, используя немного отличающиеся наборы весов довольно часто можно получить различные значения итоговой метрики (особенно важно, что ее значения для двух разных систем могут оказаться по-разному упорядочены), что ставит под сомнение ее объективность.

- Иногда можно избавиться от весов, используя в качестве интегрального значения не среднее арифметическое, а среднее геометрическое (заменяя сложение умножением, которое более осмыслено для разнородных чисел). Однако при этом часто требуется перемасштабирование отдельных показателей (удобнее обычно считать 0 базовым значением для случаев, когда какая-либо защита в данном аспекте отсутствует, но использование произведений требует считать таким значением менее удобное 1) и возможность вносить произвол за счет немного различных шкал разных показателей по-прежнему остается.

3. Обзор работ по метрикам защищенности ОС

Обзор [1] дает одну из самых полных картин используемых метрик безопасности систем, при этом рассматриваются системы вообще, без фокуса на каких-либо типах программно-аппаратных систем. Данная статья, это практически единственная попытка ввести количественные метрики для оценки защищенности, в других работах предлагают только качественные оценки. При этом заметим, что авторы обзора сами указывают на то, что метрики они выбрали субъективно и вопросы про обоснованность метрик не исследовались.

В нем все метрики безопасности разбиваются на 4 группы.

- Метрики уязвимостей (vulnerability metrics). Они измеряют количество уязвимостей, трудоемкость их использования и пр.

  Большое количество таких метрик не применимы к ОС, однако метрики, связанные с индивидуальными уязвимостями [2,3] могут быть достаточно полезны (хотя их слишком много для прямого практического использования).

- Метрики защиты (defence metrics). Они измеряют силу и полноту используемых в системе защитных механизмов (для предотвращения, обнаружения и устранения/минимизации последствий атак) и трудоемкость их преодоления при атаках.

  Они в наибольшей степени подходят для оценки защищенности ОС как платформы.

- Метрики атак (attack metrics). Эти метрики измеряют количество, силу и опасность атак, направленных на систему. Они, в основном, собираются на основе статистики состоявшихся атак.

  По этой причине они малозначимы при оценке защищенности ОС как платформы – в этом контексте все успешные атаки превращаются в уязвимости и оцениваются метриками первой группы.

- Метрики состояния системы (situation metrics). Эти метрики предназначены для отслеживания текущего состояния конкретной системы в виде динамических характеристик: общего объема вредоносного трафика в сети, уровня проникновения злонамеренного кода, общего числа и серьезности зарегистрированных инцидентов, объема затрат на преодоление их последствий.

Они также неприменимы к оценке защищенности ОС как платформы.

Некоторое количество работ посвящено оценке защищенности именно операционных систем, многие из них далее исследуют возможности построения метрик защищенности на основе проводимого анализа. Эти работы можно разделить на две группы.

- Первые используют для оценки защищенности некоторый набор функций безопасности ОС, и далее строят оценку на основе того, насколько полно, аккуратно и эффективно ОС реализует эти функции.

  - Наиболее широко известные такие наборы – это функциональные компоненты и компоненты доверия из стандарта Common Criteria (ISO/IEC 15408) [4, 5].

    Поскольку стандарт имеет общий характер и не выделяет функциональность, специфичную для ОС, на его основе разработаны Профили защиты (Protection Profile) для ОС (пример см. в [6]), предлагающие систематизированные списки функций безопасности, реализация которых ожидается от ОС общего назначения. Эти профили национально-ориентированные, в разных странах могут использоваться разные (или явно используется профиль, созданный для другой страны).

    Однако набор функций безопасности из стандарта ориентирован на проверку соответствия стандарту и сертификацию, т.е., подтверждение того, что ОС подходит для применения в некоторой области или организациях, а не на целостную оценку защищенности. Поэтому в этом наборе присутствуют такие функции как аудит, аутентификация, защита пользовательских данных, управление службами безопасности, криптографические функции, и довольно мало упоминаются (в одном пункте в рамках Профиля защиты операционных систем общего назначения OSPP 4.3 [6]) важные для защищенности ОС функции, но сильно зависящие от реализации функции, связанные с обеспечением целостности системных и пользовательских данных, изоляцией различных компонентов ОС друг от друга и механизмами усиления защиты (hardening) от широко используемых уязвимостей.

  - Есть несколько работ, например, [7], где за основу принимаются свои списки (обычно иерархические, т.е., деревья) функций безопасности. Обычно их разрабатывают на базе функций из профилей защиты для ОС, с некоторыми дополнительными доработками (в частности, в [7] добавлены функции управления ресурсами и безопасного сохранения образа системы, а также восстановления из такого образа).

  - Аналогичный подход на основе набора функций безопасности принят и в руководствах по безопасности ОС, создаваемых командами их разработчиков [8-10]. В них основное внимание уделяется вопросам корректного использования функций безопасности, таких как аутентификация, различные варианты изоляции, поддержки целостности данных пользователей и самой ОС, различные механизмы управления доступом и пр. Попыток вводить и использовать какие-либо метрики безопасности в таких руководствах обычно нет.

  - Еще в двух работах [11,12] проводится сопоставление механизмов защиты ОС Android и iOS, в качестве основы для анализа защищенности в обоих случаях выбраны списки функций безопасности, более сфокусированные на механизмах защиты, предоставляемых в ОС. Кроме того, добавлено рассмотрение широко используемых уязвимостей и усилений защиты.

    Похожий (хотя и менее детальный) метод оценки защищенности для нескольких ОС использован в работе [13].

- В работах второй группы основой для анализа безопасности ОС (и построения метрик защищенности) являются широко распространенные и используемые классы уязвимостей. Примеры таких работ: [14-16].

  В [16] анализ проводится на основе более сложной техники – так называемого дерева атак-защит (attack-defence tree, attack-countermeasure tree) [17]. Здесь используются структурированные списки (деревья) атак и мер противодействия им. Начинают обычно с атак, затем в качестве дочерних вершин к вершине атаки вставляют вершины, представляющие меры противодействия им, далее – способы обхода этих мер и т.д., попеременно используя то атаки, то методы защиты от них. Подобная структура позволяет в компактной и удобной для анализа форме представить достаточно большой набор возможных уязвимостей/атак и мер защиты от них.

  - Отдельно стоит отметить практические проекты с открытым исходным кодом, систематизирующие механизмы защиты ядра Linux. Проект Linux Kernel Defence Map [18] представляет собой граф (карту), связывающий классы уязвимостей, техники их эксплуатации и соответствующие защитные механизмы ядра Linux; по своей структуре он близок к деревьям атак-защит, хотя и не предполагает какой-либо числовой оценки. Инструмент kernel-hardening-checker [19] (ранее kconfig-hardened-check) автоматически проверяет конфигурацию ядра Linux – параметры Kconfig, командной строки ядра и sysctl – на соответствие рекомендациям по усилению защиты, собранным из таких источников, как KSPP, CLIP OS, grsecurity и собственных рекомендаций автора. Обе разработки не вводят собственных метрик защищённости и ограничены ядром Linux (а инструмент проверки даёт лишь бинарную оценку соответствия, не оценивая силу отдельных механизмов), однако они дают удобную и практичную основу для инвентаризации применяемых в ядре механизмов защиты.

Отдельная группа исследований посвящена возможности оценки защищенности систем с помощью оценки специфических рисков (киберрисков или рисков безопасности). В обзоре методов количественной оценки киберрисков [20], авторы предлагают причинно-следственную модель на базе моделирования структурных уравнений (structural equation modeling, SEM [21,22]). Подход описывает киберриск как взаимодействие латентных переменных – угрозы (Threat), подверженности воздействию (Exposure), безопасности (Security) и ущерба (Harm), – измеряемых с помощью наблюдаемых рефлексивных индикаторов. Реализация риска непосредственно не наблюдаема и оценивается косвенно через показатели ущерба. С помощью этой модели авторы систематизируют разрозненные эмпирические исследования из разных дисциплин (компьютерные науки, экономика, финансы, право).

В предложенной структуре угроза выступает единственным необходимым условием возникновения ущерба, тогда как подверженность воздействию служит положительным модератором, усиливающим её эффект, а безопасность – отрицательным модератором, ослабляющим его. Авторы показывают, что игнорирование факторов угрозы и подверженности воздействию в наивных регрессиях приводит к ложным корреляционным связям – например, к статистическим артефактам, при которых более высокие ИБ-бюджеты или применение обновлений ассоциируются с большей частотой компрометаций.

Довольно часто в итоге исследовательских работ пытаются получить инструмент/фреймворк для числовой оценки защищенности. Чтобы избежать очевидных вопросов к объективности приписываемых числовых значений, авторы пытаются использовать различные конструкции, типа байесовских оценок рисков/стоимости, теорий нечетких или серых систем и пр.

В контексте нашей постановки задачи мы берем за основу подходы, предложенные в Профиле защиты операционных систем общего назначения OSPP [6], в работе, посвященной

средствам защиты и модели угроз ОС Android [9], и в работе, посвященной сравнению защищенности ОС Android и iOS [11].

4. Общая схема оценки защищенности ОС

Интегральная оценка защищенности программной системы определяется набором конструктивных решений, нацеленных на обеспечение информационной безопасности, и эффективностью реализации этих решений. Пока открытым остается вопрос о структуризации системы характеристик защищенности ОС. Сейчас рассматриваются два альтернативных подхода (хотя между нет антагонистического противоречия):

- от угроз к анализу полноты и эффективности их парирования и
- от функций безопасности (или функциональных компонентов безопасности [4]) к анализу эффективности их реализации.

У каждого из этих подходов есть свои плюсы и минусы. Например, если следовать первому подходу, то можно упустить анализ полноты набора и эффективность реализации функций безопасности. Если следовать второму подходу, можно оставить без внимания анализ полноты средств защиты по отношению к угрозам, которые могут встретиться при эксплуатации.

В итоге приходится брать в рассмотрение и функции безопасности и список рассматриваемых угроз и контролировать полноту реализуемых средств защиты. При этом следует принять во внимание, что набор функций безопасности, так же, как и списки угроз, меняются, их нельзя зафиксировать на длительное время и для их фиксации нужно учитывать контекст использования объекта оценки. Однако набор видов техник защиты более консервативен. Для техник защиты можно сформулировать общий перечень характеристик, на основе которых определяется степень защищенности системы. Этот перечень приводится в Приложении.

5. От метрик к методике оценки защищенности

В конечном итоге мы хотели бы иметь оценку степени защищенности ОС, которая позволяет сравнить характеристики средств защиты в релизах одной ОС или в представителях разных (но сравнимых) ОС. Для получения работоспособной и практически полезной методики оценки защищенности ОС нужно разработать и проверить на пилотных экспериментах методы решения еще нескольких задач:

- построение профиля таксономии (частной таксономии) для учета особенности объекта оценки;
- квантификация базовых характеристик[1];
- агрегация базовых оценок в интегральную оценку подсистемы с учетом взаимных зависимостей механизмов защиты, когда наличие одних может делать избыточным требование реализации других.

После накопления опыта в решении этих задач появится возможность системного проектирования фреймворка, который позволит механизировать или даже автоматизировать процесс сбора и интеграции оценок защищенности в конкретной системе. Первый опыт использования таксономии был предпринят Д. Ефремовым [23]. Он разработал частную методику для оценки средств усиления защиты (hardening) и применил ее к ядрам нескольких операционных систем. Его работу можно рассматривать как первый артефакт в коллекции оценок, упомянутой выше. В работе приводится краткая аргументация предложенных схем квантификации оценок и способ выбора весов при получении интегральной оценки. Автор

[1] Квантификацию (выражение качественных характеристик в количественной форме) можно проводить на основе коллекции уже оцененных ОС или отдельных механизмов/подсистем ОС.

отметил, что влияние возможных неточностей или необъективности при выборе весов достаточно слабо влияет на интегральную оценку, что говорит о работоспособности предложенной методики.

6. Заключение

Предложенная таксономия еще требует уточнения и развития. Достижению этой цели будут служить экспериментальные апробации. В ходе этих работ потребуется регулярное обновление таксономии, включение в рассмотрение новых техник защиты и новых видов угроз, выработка подхода к построению объективных методов оценки, что позволит снизить естественную вариативность оценок, получаемых от разных экспертов.

Открытым остается вопрос о сценариях использования измерения степени защищенности. Вероятными сценариями являются сравнение вариантов планируемых работ, связанных с защищенностью ОС. Некоторый инструмент, который помогал бы сопоставить такие варианты, был бы полезен архитекторам и руководителям работ, связанных с развитием и эксплуатацией ОС. Публикаций по разработке таких инструментов достаточно много, типичным примером является работа [24], однако, широкой популярностью такие инструменты пока не пользуются.

Другим вариантом использования метрик является аудит процессов разработки операционных систем, который нацелен на оценку зрелости компании-разработчика. Пока трудно сформулировать критерии качества измерений защищенности, но если эксперименты покажут, что объективные критерии удается сформулировать, то такого рода измерения вполне могут лечь в основу стандартов по безопасной разработке программ.

Список литературы / References

[1]. Pendleton M., Garcia-Lebron R., Cho J.-H., Xu S. A survey on systems security metrics. ACM Computing Surveys, 49(4), Article 62, 2016. DOI: 10.1145/3005714.

[2]. Forum of Incident Response and Security Teams: Common Vulnerability Scoring System (CVSS), v. 4.0. Available at: https://www.first.org/cvss, accessed 17.08.2026.

[3]. Common Vulnerabilities and Exposures List. (CVE). Mitre Corporation. Available at: https://www.cve.org/Resources/Media/Archives/OldWebsite/index.html, accessed 17.08.2026.

[4]. ISO/IEC 15408-2:2022. Common Criteria for Information Technology Security Evaluation. Part 2: Security Functional Components.

[5]. ISO/IEC 15408-3:2022. Common Criteria for Information Technology Security Evaluation. Part 3: Security Assurance Components.

[6]. Protection Profile for General Purpose Operating Systems (OSPP), v.4.3. National Information Assurance Partnership, 2022.

[7]. Afzali H., Mokhtari H. A Quantitative Model of Operating System Security Evaluation. In: Wyld D., Zizka J., Nagamalai D. (eds) Advances in Computer Science, Engineering & Applications. Advances in Intelligent Systems and Computing. Springer, 2012, vol. 167. DOI: 10.1007/978-3-642-30111-7_33.

[8]. Mayrhofer R., Stoep J.V., Brubaker C., Kralevich N. The Android Platform Security Model. ACM Transactions on Privacy and Security, 2021, 24(3), Article 19. DOI: 10.1145/3448609.

[9]. Mayrhofer R., Stoep J.V., Brubaker C., Hackborn D., Bonné B., Tuncay G.S., Jover, R.P., Specter M.A. The Android Platform Security Model (revised 2024). DOI: 10.48550/arXiv.1904.05572.

[10]. Apple Platform Security, March 2026. Available at: https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf, accessed 17.08.2026.

[11]. Garg S., Baliyan N. Comparative analysis of Android and iOS from security viewpoint. Computer Science Review, 40, Article 1000372, 2021. DOI: 10.1016/j.cosrev.2021.100372.

[12]. Khan S., Yusuf A., Haider M., Thirunavukkarasu K., Nand P., Imam Rahmani M.K. A Review of Android and iOS Operating System Security. Proc. of ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), Manama, Bahrain, 2022, pp. 67-72. DOI: 10.1109/ICETSIS55481.2022.9888847.

[13]. Sajid A., Shah M.A., Kamran M., Javaid Q., Zhang S. An Analysis on Host Vulnerability Evaluation of Modern Operating Systems. International Journal of Advanced Computer Science and Applications (IJACSA), 7(4):245-254, 2016. DOI: 10.14569/IJACSA.2016.070430.

[14]. Kaluarachchilage P.K.H., Attanayake C., Rajasooriya S., Tsokos C.P. An Analytical Approach to Assess and Compare the Vulnerability Risk of Operating Systems. International Journal of Computer Network and Information Security (IJCNIS), 12(2):1-10, 2020. DOI: 10.5815/ijcnis.2020.02.01.

[15]. Bhurtel M., Rawat D.B. Unveiling the Landscape of Operating System Vulnerabilities. Future Internet, 15(7):248, 2023. DOI: 10.3390/fi15070248.

[16]. Ali Z., Aslam N., Marotta A., Tiberti W., Cassioli D. A Systematic Review of Kernel-Level Security Mechanisms, Vulnerability Detection and Mitigation in Modern Operating Systems. Sensors, 26(8), 2452, 2026. DOI: 10.3390/s26082452.

[17]. Kordy B., Mauw S., Radomirović S., Schweitzer P. Foundations of Attack-Defense Trees. In: Degano P., Etalle S., Guttman J. (eds) Formal Aspects of Security and Trust. FAST 2010. LNCS 6561:80-95. Springer, 2011. DOI: 10.1007/978-3-642-19751-2_6.

[18]. Popov A. Linux Kernel Defence Map. Available at: https://github.com/a13xp0p0v/linux-kernel-defence-map, accessed 17.08.2026.

[19]. Popov A. kernel-hardening-checker: A tool for checking the security hardening options of the Linux kernel. Available at: https://github.com/a13xp0p0v/kernel-hardening-checker, accessed 17.08.2026.

[20]. Woods D.W., Böhme R. SoK: Quantifying cyber risk. In Proceedings of the 42nd IEEE Symposium on Security and Privacy (SP), 2021, pp. 211-228. DOI: 10.1109/SP40001.2021.00053.

[21]. Bollen K.A. Structural Equations with Latent Variables. John Wiley & Sons, New York, 1989. DOI: 10.1002/9781118619179.

[22]. Anderson J.C., Gerbing, D.W. Structural equation modeling in practice: A review and recommended two-step approach. Psychological Bulletin, 103(3), 411-423, 1988. DOI: 10.1037/0033-2909.103.3.411.

[23]. Ефремов Д.В. Сравнительная количественная оценка механизмов усиления защищенности ядер операционных систем. Труды ИСП РАН, 2026, том 38, вып. 4 (часть 2), стр. xx–xx. DOI: 10.15514/ISPRAS-2025-37(1)-1 / Efremov D.V. Comparative Quantitative Evaluation of Kernel Hardening Mechanisms in Operating Systems. Trudy ISP RAN/Proc. ISP RAS, 2026, vol. 38, issue 4 (part 2), pp. xx-xx (in Russian). DOI: 10.15514/ISPRAS-2025-37(1)-1. здесь пока подождем: эта статья публикуется в этом же выпуске.

[24]. Jonsson E., Pirzadeh L. A framework for security metrics based on operational system attributes. Proc. of 3-rd International Workshop on Security Measurements and Metrics, Banff, Canada, 2011, pp. 58-65. DOI: 10.1109/Metrisec.2011.19.

Информация об авторах / Information about authors

Виктор Вячеславович КУЛЯМИН – кандидат физико-математических наук, доцент кафедры Системного программирования ВМК МГУ, ведущий научный сотрудник ИСП РАН. Сфера научных интересов: программная инженерия, тестирование на основе моделей, формальные методы программной инженерии, формальные методы обеспечения безопасности.

Victor Viatcheslavovitch KULIAMIN – Cand. Sci. (Phys.-Math.), Associate Professor of System Programming Department, Faculty of Computational Mathematics and Cybernetics Moscow State University, Leading Researcher of the Institute for System Programming, Russian Academy of Sciences. Research interests: software engineering, model-based testing, formal methods of software engineering, formal methods of software security assurance.

Александр Константинович ПЕТРЕНКО – доктор физико-математических наук, профессор, заведующий отделом Технологий программирования ИСП РАН, профессор кафедр Системного программирования ВМК МГУ и ФКН НИУ ВШЭ. Научные интересы: формальные методы программной инженерии, операционные системы, языки спецификаций и моделирования, верификация, кибербезопасность.

Alexander Konstantinovich PETRENKO – Dr. Sci. (Phys.-Math.), Prof., Head of the Software Engineering Department at the Ivannikov Institute for System Programming, Russian Academy of Sciences, Professor of MSU and the Faculty of Computer Science, NRU HSE. Research interests: formal methods of software engineering, operating systems, specification and modeling languages, verification, cybersecurity.

Приложение. Схема оценки техник защиты ОС

Для каждой используемой в оцениваемой системе техники защиты или блокирования/ослабления некоторой угрозы нужно описать следующие элементы, выделяя особенности, влияющие на итоговую защищенность.

- ***Модель/алгоритм (что должно делаться) и реализация (что именно реализовано)***
  Это некоторое описание основных идей и элементов используемой техники.
  Особенно стоит выделить следующее:
  - Затрагивает ли реализация все компоненты, необходимые для блокирования целевых угроз, или только их часть.
    Если часть – какую, какие другие техники использованы в остальных местах.
  - Реализована ли оцениваемая техника на всех платформах и конфигурациях ОС, или только на части.
    Если на части, то чем именно она выделена (и почему), что делается на остальных платформах и конфигурациях.
  - Реализована ли желаемая идея/модель полностью, или реализация ограничена с точки зрения получаемой защиты (по сравнению с идеальным алгоритмом), в чем именно могут быть ограничения.
  - Какие уровни задействованы в реализации.
    - железо
    - компоненты bios/firmware
    - общая архитектура ядра
    - отдельные компоненты ядра и/или специальные компоненты ядра, нужные только для этой техники
    - драйверы
    - элементы конфигурации/политики защиты
    - специальные приложения
    - компоненты компилятора
    - др. средства разработки и сборки (анализаторы, санитайзеры, тесты, верификаторы и пр., не входящие в основной поток компиляции)
    - процессы эксплуатации редко [требуются особые действия при установке и администрировании, редкие действия по настройке и проверке] или процессы эксплуатации регулярно [особые действия требуются почти каждый раз при использовании связанных функций]
    - процессы разработки и сборки редко [при разработке связанного кода и его правке] или процессы разработки и сборки регулярно [почти при каждой сборке]
  - Насколько в результате реализация техники сконцентрирована в рамках одного-двух уровней (перечисленных выше) или, наоборот, рассредоточена по многим разным.
- ***Целевые угрозы***
  Это список угроз, для блокирования или минимизации ущерба от которых предназначена техника.
  - Угрозы должны формулироваться в максимально точном виде. Если этот вид отличается от некоторой общей формулировки угрозы, то указать, в чем, какие случаи остаются не закрытыми (или наоборот, в каких именно частных случаях блокируется/минимизируется угроза).
  - Указания на полноту блокировки для каждой угрозы (стоит приводить детальные пояснения, если возможно).
    - полностью

    - значительно/ущерб снижается значительно
    - частично/ущерб снижается средне
    - минимально/ущерб снижается незначительно
  - Указания на обеспечиваемые свойства безопасности (или цели безопасности), если их можно сформулировать в достаточно компактном и ясном виде (например, «только процессы, запущенные доверенными пользователями, могут модифицировать данный элемент памяти»).
- ***Слабости и возможности обхода***
  Это список возможных слабостей техники (включая слабости в компонентах, от которых она зависит, типа слабого генератора случайных чисел) и способов ее отключения/обхода.
  - Какие слабости и др. факторы могут не позволить технике достичь максимального уровня закрытия целевых угроз.
  - Каким образом можно убрать эти слабости/убрать или снизить негативное влияние факторов.
  - Какими способами можно отключить или обойти технику при установке/настройке/эксплуатации, какие привилегии для этого нужны (сборщик[deployer]/разработчик/админ безопасности/админ/пользователь со спец. возможностями/любой пользователь).
- ***Вопросы сложности***
  Это набор особенностей техники, влияющих на сложность и трудоемкость ее реализации.
  - Общая сложность (экспертная оценка, объем доп. кода).
  - Рассредоточенность реализации по различным уровням и компонентам (см. выше).
  - Зависимости от инструментов/др. компонентов на различных уровнях.
- ***Вопросы производительности***
  Это набор особенностей техники, влияющих на итоговую производительность ОС.
  - В какие моменты и насколько производительность может снижаться (экспертные оценки).
    - при сборке
    - при установке
    - при запуске
    - при настройке/конфигурировании/администрировании
    - при эксплуатации
  - На каких конкретно операциях/системных вызовах и насколько часто (может быть не на каждом вызове, а в зависимости от данных и состояния, может на каждом).
  - Измерения снижения производительности, на каких именно тестах.
- ***Вопросы разработки и поддержки***
  Это набор особенностей реализации техники с точки зрения используемых методов при ее реализации и внесения изменений и правок.
  Особенно стоит выделить следующее.
  - Какие методы и инструменты контроля качества использованы при разработке.
  - Насколько проведена верификация реализации техники.
    - Имеющиеся модульные/интеграционные/общие тесты.
    - Используемые статические методы/инструменты анализа.
    - Используемые динамические методы/инструменты анализа.
    - Формальная верификация (какая часть кода и свойств ею закрыта, какой

набор предположений использован).

- Оценка рисков необходимости и частоты правок (все сделано один раз и не исправлялось, правилось и уже давно заморожено, есть следы не слишком давних исправлений, регулярные обновления).
- Как построена цепочка внесения правок/изменений.
  - Участвует ли только разработчик ОС или есть другие участники (отвечающие за разные компоненты), сколько их вообще.
  - Возможность получения реакции от разработки/поддержки (все замерло уже, или активно поддерживается, или отвечают только при каком-то статусе запроса/запрашивающего).
  - Оценки времени внесения правок (от сообщения о проблеме/решения о внесении изменения до развертывания у конечного пользователя).