УДК 004.41:004.056
MSC 68N25, 68N30

Кибериммунный подход к разработке. Иллюстрация применения на базе микросервисной архитектуры

С. П. Соболев

Санкт-Петербургский государственный университет,
Российская Федерация, 199034, Санкт-Петербург, Университетская наб., 7–9

Для цитирования: *Соболев С. П.* Кибериммунный подход к разработке. Иллюстрация применения на базе микросервисной архитектуры // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2024. Т. 20. Вып. 1. С. 52–61. https://doi.org/10.21638/11701/spbu10.2024.105

По мнению автора, обеспечение информационной защиты систем должно начинаться уже на этапе проектирования, а не перед внедрением в эксплуатацию. Кроме того, автор также придерживается точки зрения, что абсолютная защита современных информационных систем невозможна в принципе, вместо попыток равномерной защиты на уровне периметра безопасности системы необходимо определить компоненты, компрометация которых приводит к нанесению неприемлемого для заказчика системы ущерба. Разработчику следует сосредоточить усилия на качественной реализации и защите именно таких подсистем. Однако основой для такого подхода является наличие встроенных в систему защищенных механизмов разделения и контроля взаимодействия подсистем. В статье описывается, каким образом встроенная защищенность от уязвимостей и атак может быть реализована на примере широко используемого класса микросервисных архитектур.

Ключевые слова: кибериммунитет, проектирование систем, микросервисы, системная инженерия, конструктивная информационная безопасность.

1. Введение. Традиционный подход к разработке программного обеспечения фокусируется на реализации полезных функций, таких как, например, обработка данных или управление оборудованием. Обеспечение информационной защиты оставляется на последний этап и зачастую проводится по остаточному принципу, несмотря на повсеместное ужесточение законодательства в этой области. С одной стороны, требования к информационной безопасности (ИБ) в большинстве случаев размыты (требуется «безопасность вообще»), а с другой — для реализации сложных механизмов не хватает компетенций разработчиков, которые поэтому стараются затратить минимальные ресурсы на реализацию, тем более, что заказчики все равно не смогут оценить качество защиты и не будут готовы существенно доплатить для получения усиленной безопасности, поскольку никаких гарантий невозможности взлома разработчики дать не смогут. Это приводит к реактивному подходу в разработке механизмов защиты: делается только то, что прописано в явном виде, требуется для сопряжения с внешними системами или направлено на устранение замечаний, выявленных заказчиком. Использование системного подхода и соответствующих стандартов безопасности типа ГОСТ Р МЭК 62443-3-3-2016 [1] является редким исключением. Кроме того, это высокоуровневый стандарт, который не дает конкретных рецептов.

2. Кибериммунный подход. Кибериммунный подход, набирающий популярность в последнее время и методологически разрабатываемый компанией «Лаборатория Касперского» [2] совместно с сообществом, предполагает комбинацию следующих основных подходов [3, 4]:

1) конструктивную безопасность (secure by design);

2) безопасность как код (security as a code).

Кроме того, отправной точкой проектирования кибериммунных систем является выявление ключевых активов проектируемой системы и нежелательных событий (риски, ущербы), которые могут причинить ущерб данным активам.

Исходя из такого анализа, формулируются цели безопасности для разрабатываемой системы, а архитектура системы должна обеспечивать достижение этих целей. Некоторые аспекты не могут быть адресованы на уровне разрабатываемой системы, тогда они выносятся в предположения безопасности и, как правило, рассматриваются в качестве целей безопасности для других систем.

К таким предположениям безопасности относится физическая защищенность системы — на уровне информационной защиты трудно противостоять злоумышленнику, который имеет физический доступ к системе и может, например, сломать процессор молотком или перерезать провода. Такого рода риск должен рассматриваться на уровне службы безопасности объекта, обеспечивающей защиту физического периметра, если это применимо.

Формулировка целей и предположений безопасности (ЦПБ) — критически важный этап, поскольку итоговый документ становится по существу контрактом между заказчиком системы и разработчиками.

После согласования ЦПБ архитектор должен предложить такую архитектуру системы, с которой он сможет логически обосновать достижение целей безопасности. Диаграмма связей ролей и артефактов этапа проектирования показана на рис. 1.

Базовыми обеспечивающими технологиями кибериммунных систем являются:

1) **MILS** (multiple independent levels of security) — изоляция частей системы на домены безопасности [3];

2) **FLASK** (flux advanced separation kernel) — тотальный контроль взаимодействия доменов [4].

Главный приоритет при разработке кибериммунных систем — минимизация доверенной кодовой базы, т. е. кода, которому, с одной стороны, вынужденно доверяем, поскольку он является критичным для достижения целей безопасности, а с другой — доверенный код по этой же причине очень тщательно тестируем, чтобы из категории «вынужденно доверенного» перевести его в категорию «благонадежного». «Благонадежный код» — это тот код, в результатах корректной работы которого мы уверены.

Разделение на домены позволяет комбинировать в одной системе доверенный и недоверенный коды.

• *Недоверенный код* — тот, в котором изначально сомневаемся и допускаем его взлом. Он может быть поверхностно протестирован или вообще не тестируется на безопасность, чтобы ни происходило в этом коде, так как это не нарушит цели безопасности системы.

• *Доверенный код* — тот, который максимально тщательно тестируем, используя различные технологии, вплоть до формальной верификации, если это приемлемо по затратам, потому что компрометация данного кода может нарушить достижение целей безопасности.

Очевидно, механизмы изоляции доменов должны быть достаточно хорошо реа-

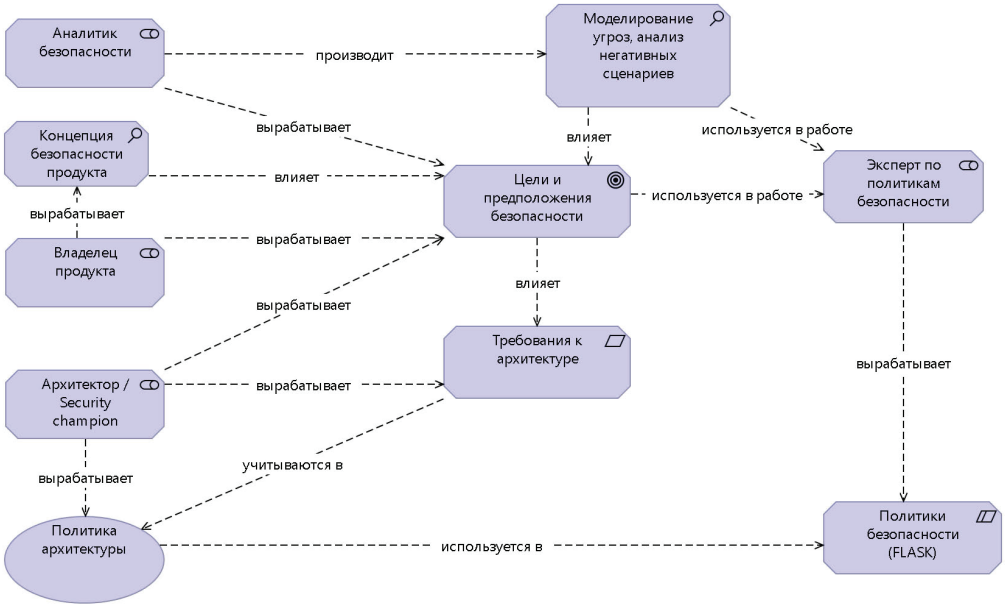

Рис. 1. Связь ролей и артефактов этапа проектирования

лизованы, чтобы успешный взлом недоверенных доменов не поставил под угрозу доверенные. Механизм разделения является частью ядра системы и должен тестироваться так же, как и доверенный код.

Связь ролей и сущностей этапа разработки и тестирования показана на рис. 2.

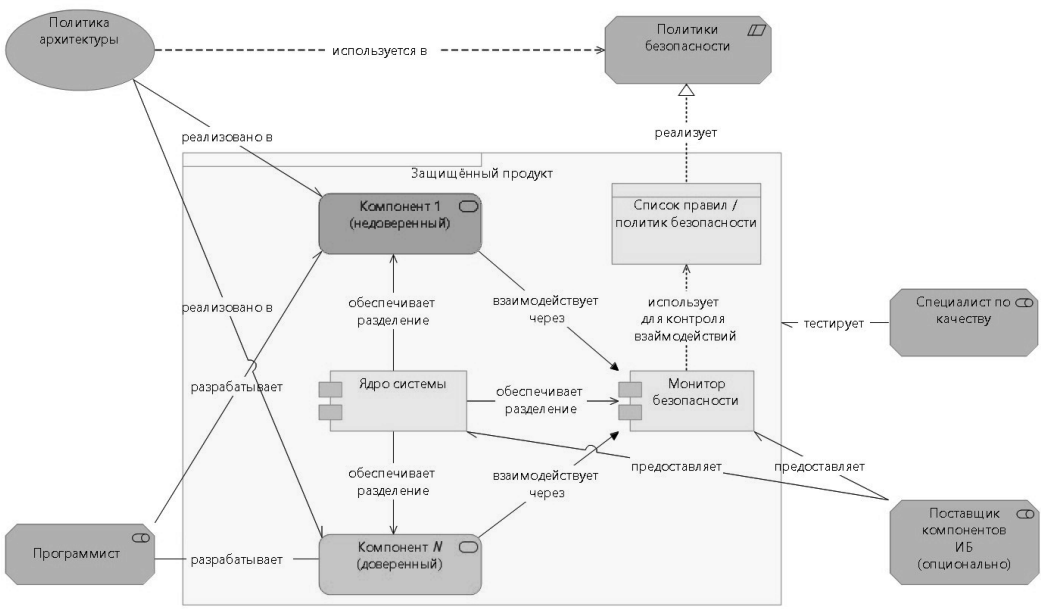

Рис. 2. Связь ролей и сущностей этапа разработки и тестирования

Любое взаимодействие доменов должно быть проверено на допустимость, это делается в специальном компоненте, называющемся *монитором безопасности*, с использованием правил, описывающих допустимое взаимодействие, которые именуем *политиками безопасности*. Применение политик безопасности позволяет отделить логику контроля безопасности от основного кода и контролировать все взаимодействия в системе централизованно.

3. Микросервисные архитектуры и кибериммунный подход.

3.1. Архитектурное решение. Декомпозиция систем на небольшие части, имеющие очень ограниченную функциональность, широко используется для разработки современного программного обеспечения, обладающего рядом преимуществ перед разработкой системы в виде монолита.

Кроме того, с развитием облачных платформ и инструментов виртуализации появилась возможность разрабатывать отдельные подсистемы (компоненты) с помощью разных технологий, наиболее подходящих для решения практических задач. Минимизация функций, реализуемых каждым компонентом, повышает контролируемость и гибкость разработки. Главным недостатком такого подхода является перерасход ресурсов (вычислительных мощностей, памяти для хранения и обработки данных), что для облачных платформ есть вопрос стоимости, а не реализуемости.

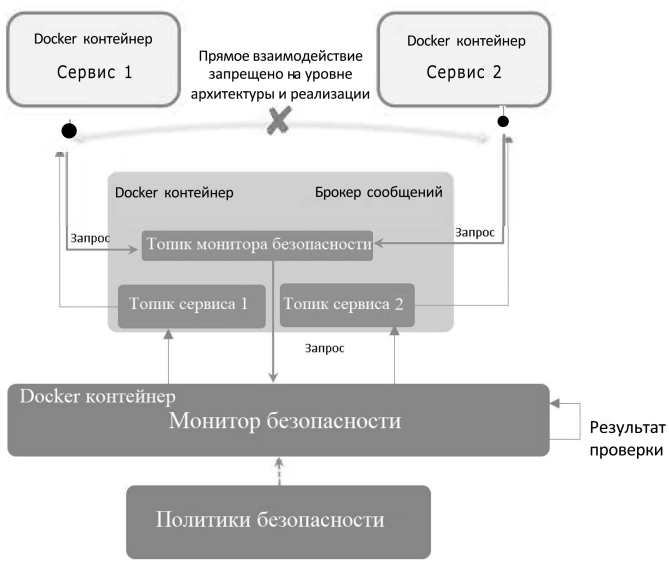

Рис. 3. Контролируемое взаимодействие микросервисов

Сочетание микросервисных систем и кибериммунного подхода органично: отдельные компоненты могут независимо выполняться на распределенных вычислительных мощностях, но за счет централизованного контроля взаимодействия подсистем будут достигаться цели безопасности, предъявляемые к системе.

Минимизация функционала отдельных (микро)сервисов способствует повышению гранулярности контроля и уменьшает негативные последствия от взлома, так как развитие атаки вглубь системы блокируется монитором безопасности (рис. 3).

К ключевым аспектам реализации таких систем относятся:

- логически обоснованная декомпозиция системы на микросервисы;

• техническая реализация взаимодействия микросервисов, обеспечивающая централизованный контроль;

• механизм реализации политик безопасности и сами политики безопасности.

Взаимодействие между компонентами системы может быть как синхронным (т. е. запрос — ожидание ответа — ответ), так и асинхронным (с использованием механизма обмена сообщениями). В последнем случае компоненты обрабатывают входящие сообщения и передают результат, даже, согласно их бизнес-логике, не ожидая ответа от сервиса-получателя. Взаимодействие через асинхронный обмен сообщениями является часто используемым подходом для распределенных систем, поскольку легче горизонтально масштабируется (т. е. допускает использование изменяемого количества обработчиков сообщений без увеличения вычислительных ресурсов, доступных каждому из обработчиков). Прием и доставку сообщений обеспечивают брокеры сообщений.

Отдельные компоненты могут выполняться в специальных виртуальных контейнерах, например может использоваться технология docker. В таком случае, если из-за взлома злоумышленник получил доступ к содержимому контейнера, то, в частности, удаление всех файлов в корневой папке приведет только к неработоспособности одного контейнера, а не всей системы. Перезапуск компонента в новом контейнере является типовым сценарием и в большинстве случаев может быть даже не замеченным другими сервисами.

3.2. Тестирование безопасности. Если правильная декомпозиция системы на сущности, использование обеспечивающих технологий (MILS, FLASK) — часть конструктивной безопасности, то описание допустимого взаимодействия компонентов в виде политик безопасности — это уже безопасность в виде кода.

Еще одной важной составляющей безопасности в виде кода является возможность ее тестирования. В отличие от применяемого на поздних этапах тестирования на проникновение, в кибериммунном подходе разрабатываются тесты негативных сценариев, направленные на динамическую проверку политик безопасности.

Наряду с обычными тестами поведения системы в предположении возможных атак снаружи при тестировании кибериммунных систем может изменяться поведение отдельных компонентов (атаки изнутри) для контроля достижимости целей безопасности.

Таким образом, при разработке кода можно до этапа финальных тестов встроить проверку систем безопасности, а их реализация в соответствии с целями безопасности позволяет логически обосновать защищенность системы при любых атаках, использовавших недоверенные компоненты.

4. Применение микросервисного инструментария для проектирования кибериммунных систем. Предложенный авторский инструментарий в настоящее время активно используется для популяризации кибериммунного подхода к разработке обучения студентов и специалистов.

Разработаны учебные примеры и задания, которые применяются и в рамках соревнований по проектированию кибериммунных систем. Рассмотрим в качестве учебного примера функциональный прототип кибериммунного устройства мониторинга оборудования (КУМО). Для учебного примера все внешние по отношению к КУМО интерфейсы заменим имитацией запросов в REST формате.

4.1. Концепция безопасности продукта. Ее формулирует заказчик разработки (или владелец продукта).

Кратко опишем назначения и применения продукта: КУМО собирает информа-

цию с производственных информационных систем, находящихся во внутренней информационной сети предприятия (Интранет), затем после обработки передает эту информацию для дальнейшего анализа пользователям, подключающимся к устройству через внешнюю сеть (Интернет)

Основные ценности и неприемлемые в отношении этих ценностей события приведены в табл. 1.

Таблица 1. **Ценности продукта, неприемлемые события**

Ценность	Неприемлемые события	Комментарии
Данные оборудования	Нарушение целостности, пропажа данных, неавторизованный доступ	
Результаты анализа данных	Аналогично	КУМО осуществляет поиск аномалий по сырым данным
Безопасность периметра внутренней сети	Проникновение во внутреннюю сеть через КУМО	

Верхнеуровневые сценарии работы устройства включают в себя:

• получение и анализ данных от внешних устройств, создание событий для передачи пользователю;

• передачу событий пользователю по запросу;

• обновление внутренней логики работы устройства.

Роли пользователей описаны в табл. 2.

Таблица 2. **Роли пользователей**

Роль	Назначение
Оператор	Чтение данных, чтение событий
Инженер	Изменение параметров работы системы

4.2. Цели и предположения безопасности (ЦПБ). ЦПБ формулируются совместно заказчиком и архитектором продукта, они еще не содержат технические решения, но оформляют условия «контракта» на разработку — архитектор должен предложить такую реализацию продукта, которая удовлетворяет согласованным целям безопасности с учетом предположений.

4.2.1. Цели безопасности. Они включают следующее:

1) только аутентифицированный оператор имеет доступ к критичным данным;

2) только аутентифицированный инженер может осуществить обновление параметров работы КУМО;

3) для обновления используется только целостная и аутентичная 'прошивка';

4) данные из внешней сети через КУМО ни при каких обстоятельствах не попадают во внутреннюю сеть.

4.2.2. Предположения безопасности. К ним относятся такие:

1) физическая безопасность устройства обеспечена;

2) только авторизованные пользователи имеют физический доступ к устройству;

3) аутентичные пользователи благонадежны (т. е. не пытаются намеренно причинить ущерб системе).

4.2.3. Политика архитектуры. Политика архитектуры — это архитектурная диаграмма, иллюстрирующая допустимые взаимодействия доменов безопасности,

уровень целостности данных и степень влияния доменов безопасности на возможность достижения целей безопасности при обозначенных предположениях безопасности.

Отправной точкой при построении политики архитектуры может использоваться функциональная архитектура системы, при этом все сущности и взаимодействия должны быть красными. Это эквивалентно тому, что все домены безопасности — недоверенные (т. е. могут быть скомпрометированы, могут произвольным образом изменить проходящие через них данные), а данные — низкоцелостные (могут содержать искаженную информацию, которую нельзя использовать без дополнительных проверок в доверенных сущностях).

В ходе анализа архитектор выявляет компоненты, которые могут критическим образом повлиять на имеющиеся цели безопасности, такие компоненты автоматически объявляются доверенными.

Нотация политики архитектуры такова, что исходящие из недоверенных сущностей (красных) данные обязаны считаться низкоцелостными, а из доверенных (желтых и зеленых) — высокоцелостными.

Доверенные сущности желтого цвета имеют специальное назначение — они повышают целостность данных, для чего реализуют соответствующие проверки. На этапе проектирования подробного описания проверок может не быть, они появятся на этапе детального проектирования и кодирования, а качество проверок должно подтверждаться тестированием.

На следующем шаге архитектор анализирует качество полученной политики архитектуры. Важной задачей архитектора является качественный анализ сложности и объема доверенной кодовой базы (trusted computing base — TCB). На этапе разработки и тестирования доверенные сущности должны проходить дополнительные проверки и максимально тщательно тестироваться. Особенно это важно для доверенных сущностей, повышающих целостность данных, так как они могут столкнуться с атаками изнутри, идущими от недоверенных (красных) сущностей.

Должно быть очевидно, что стоимость разработки кода доверенных сущностей качественно выше стоимости разработки кода, не относящегося к TCB. В идеале все сторонние библиотеки должны использоваться только в недоверенных сущностях.

В свою очередь, доверенные сущности должны быть преимущественно маленькими по объему, простыми по логике и содержать минимальное количество интерфейсов.

В случае, когда сложная и большая сущность в архитектуре объявлена доверенной, архитектору следует найти способ декомпозировать данную сущность на несколько более мелких и простых. В идеале при этом часть кода доверенной сущности должна быть выведена из TCB (т. е. перенесена в красные компоненты).

Пример политики архитектуры приведен на рис. 4. В кружках символами указана качественная оценка сложности и размера сущности:

• сложность — S (simple), M (medium), C (complex) — простой, средней сложности, сложный;

• размер кода — S (small), M (medium), L (large), XL (eXtra Large) — маленький, среднего размера, большой, очень большой соответственно.

На диаграмме, в частности, видно, что сущность 4 («Обработчик некритичных данных») является сложной и объемной (CL — complex, large), поэтому в ней допустимы только некритичные данные, поскольку полностью протестировать содержащиеся в этой сущности алгоритмы может быть экономически нецелесообразно.

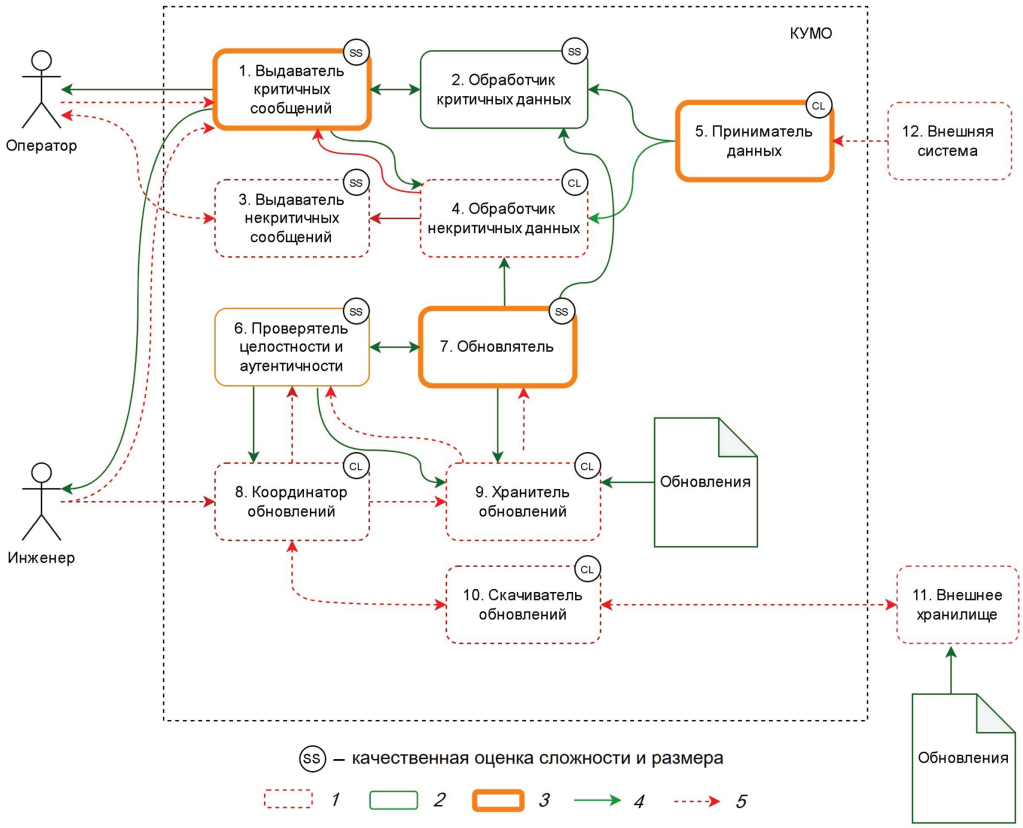

Рис. 4. Политика архитектуры для учебного примера

1 — недоверенный домен; *2* — доверенный домен; *3* — доверенный домен, повышающий целостность данных; *4* — высокоцелостные данные; *5* — низкоцелостные данные.

А критические данные будут анализироваться в сущности 2 («Обработчик критичных данных»), которая будет простой и маленькой.

Реализация кода КУМО должна быть организована таким образом, чтобы обеспечить тщательный контроль доверенных сущностей 1, 2, 5, 6, 7, тогда как для недоверенных сущностей 3, 4, 8, 9, 10, содержащих бóльшую часть кода, можно будет использовать преимущественно функционально-ориентированное тестирование.

5. Заключение. Кибериммунный подход к разработке является особенно важным в ситуации, когда необходимо в сжатые сроки разрабатывать сложные системы, особенно системы, содержащие большое количество сложной логики, собственного и стороннего кода, тестировать который одинаково тщательно не представляется возможным в первую очередь из-за экономических соображений.

Четкая последовательность действий от задумывания продукта до тестирования с фокусом на обоснованном достижении целей безопасности позволяет на практике внедрять кибериммунный подход к разработке для создания защищенных систем любого уровня сложности.

Дальнейшая работа может вестись в направлении инструментальной поддержки процесса проектирования, в частности инструментов автоматической верификации и оценки качества политики архитектуры.

При использовании моделеориентированной разработки код сущностей может быть в некоторых случаях автоматически сгенерирован прямо из модели, что упростит контроль соответствия реализации проектной документации.

Литература

1. ГОСТ Р МЭК 62443-3-3-2016. Сети промышленной коммуникации. Безопасность сетей и систем. Ч. 3-3. Требования к системной безопасности и уровни безопасности. М.: Стандартинформ, 2016. 62 с.

2. *Конструктивная информационная безопасность*. М.: Лаборатория Касперского, 2023. 2 с. https://os.kaspersky.ru/blog/security-by-design

3. *DeLong R. J., Rudina E.* MILS architectural approach supporting trustworthiness of the IIoT solutions: IIC whitepaper. Boston: Industrial Internet Consortium, 2021. 94 p.

4. *Spencer R., Smalley S. D., Loscocco P., Hibler M., Andersen D. G., Lepreau J.* The Flask security architecture: system support for diverse security policies. Washington: USENIX Security Symposium, 1999. 17 p.

Статья поступила в редакцию 25 июня 2023 г.
Статья принята к печати 26 декабря 2023 г.

Контактная информация:

Соболев Сергей Павлович — канд. техн. наук, ст. преп.; s.sobolev@spbu.ru

Cyber immune development approach. Microservices based illustration

S. P. Sobolev

St. Petersburg State University, 7–9, Universitetskaya nab., St. Petersburg, 199034, Russian Federation

For citation: Sobolev S. P. Cyber immune development approach. Microservices based illustration. *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, 2024, vol. 20, iss. 1, pp. 52–61. https://doi.org/10.21638/11701/spbu10.2024.105 (In Russian)

The author believes, that ensuring the information security of systems should begin at the design stage, rather than after implementation is over and verification starts. Also the author supports the point of view, that it is impossible to guarantee absolute information security, eliminate all defects and vulnerabilities from code once and forever. It is more realistic to assume that there are software defects are present in the inner permimeter of any system and the main question is how critical this issue will be for the systems customer assets. Developers shall focus their efforts in design and implementation in such a way that probability of successful attacks compromising system security objectives through critical code is minimal. Also such critical parts are defined and optimised for size and complexity during design phase, separated from non-critical parts, substantial efforts are invested in high qualirty implementation and thorough testing of such critical parts. The article describes how built-in protection against vulnerabilities and attacks can be illustrated using microservices-based architecture.

Keywords: cyber immunity, systems engineering, systems design, microservices, secure software development.

References

1. GOST R MEK 62443-3-3-2016. *Seti promyshlennoi kommunikatsii. Bezopasnost' setei i sistem. Ch. 3-3. Trebovaniia k sistemnoi bezopasnosti i urovni bezopasnosti* [GOST R IEC 62443-3-3-2016.

Industrial communication networks. Network and system security. Pt 3-3. System security requirements and security levels]. Moscow, Standardinform Publ., 2016, 62 p. (In Russian)

2. *Konstruktivnaia informatsionnaia bezopasnost'* [*Security by Design*]. Moscow, Kaspersky Lab. Publ., 2023, 2 p. https://os.kaspersky.ru/blog/security-by-design (In Russian)

3. DeLong R. J., Rudina E. *MILS architectural approach supporting trustworthiness of the IIoT solutions*. IIC whitepaper. Boston, Industrial Internet Consortium, 2021, 94 p.

4. *Spencer R., Smalley S. D., Loscocco P., Hibler M., Andersen D. G., Lepreau J.* The Flask security architecture: System support for diverse security policies. Washington, USENIX Security Symposium Publ., 1999, 17 p.

Received: June 25, 2023.
Accepted: December 26, 2023.

Author's information:

Sergey P. Sobolev — PhD in Engineering, Senior Lecturer; s.sobolev@spbu.ru