ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
СПбГУТ))) Сентябрь 2018 Том 6 № 3 ISSN 2307-1303 http://www.itt.sut.ru
TELECOM IT
September 2018 Vol. 6 Iss. 3 ISSN 2307-1303 http://www.itt.sut.ru

УНИФИКАЦИЯ ВЗАИМОДЕЙСТВИЯ КИБЕРФИЗИЧЕСКИХ СИСТЕМ С ИСТОЧНИКАМИ ДАННЫХ

Д. С. Левшун[1*]

[1] СПИИРАН, Санкт-Петербург, 199178, Российская Федерация
* Адрес для переписки: levshun@comsec.spb.ru

Аннотация
Взаимодействие устройств на основе микроконтроллеров с источниками данных, а именно различными датчиками, считывателями, извещателями и оповещателями и другими внешними электронными компонентами, можно условно разделить на следующие типы: инициализация, считывание, запись, событие и деинициализация. При этом важно отметить, что в зависимости от сложности подключаемого источника данных, те или иные типы взаимодействия могут остаться незадействованными. **Предмет исследования.** В данной работе представлена разработка, отладка и тестирование унифицированного взаимодействия киберфизических систем с источниками данных. **Метод.** Техническая реализация основана на принципах модульности и поуровневости. Отладка и тестирование осуществлялась на основе генератора трафика, функциональные возможности которого можно условно разделить на две части: нагрузочное тестирование и имитацию работы реальной системы. **Основные результаты.** Нагрузочное тестирование позволило выявить ошибки реализации, связанные с передачей сообщений минимального размера и переполнения буферов, а также некорректную обработку отдельных символов ASCII при их передаче по Serial. Имитация работы реальной системы позволила выявить ошибки реализации отдельных механизмы корреляции событий безопасности и сформировать корректное отображение данных в пользовательском интерфейсе системы. Также имитация работы системы позволила значительно ускорить процесс получения дампов трафика, передаваемого между микроконтроллерами системы, в том числе специализированных. **Практическая значимость.** В результате проделанной работы, была получена подключаемая библиотека, представляющая собой унифицированную реализацию взаимодействия устройств платформы Arduino с ранее задействованными в рамках работы нашей лаборатории источниками данных. Это позволило ускорить обмен опытом между сотрудниками лаборатории, унифицировать код, а также ускорить процесс разработки новых экспериментальных стендов и прототипов систем.

Ключевые слова
киберфизическая система, унифицированное взаимодействие, генератор трафика, микроконтроллер.

Информация о статье
УДК 004.056.53
Язык статьи – русский.
Поступила в редакцию 10.07.18, принята к печати 03.09.18.

Ссылка для цитирования: Левшун Д. С. Унификация взаимодействия киберфизических систем с источниками данных // Информационные технологии и телекоммуникации. 2018. Том 6. № 3. С. 28–37.

UNIFICATION OF CONNECTION BETWEEN CYBER-PHYSICAL SYSTEMS AND SOURCES OF DATA

D. Levshun[1*]

[1] SPIIRAS, St. Petersburg, 199178, Russian Federation
* Corresponding author: levshun@comsec.spb.ru

Abstract—Interaction of microcontroller-based devices with data sources (different sensors, readers, detectors, alarms and other external electronic components) can be conditionally divided into the following types: initialization, read, write, event and de-initialization. It is important to note that, depending on the complexity of the data source being connected, certain types of interaction may remain unused. **Research subject.** In this paper, the development, debugging and testing of unified interaction of cyberphysical systems with data sources is presented. **Method.** Technical implementation is based on the principles of modularity and level-by-level development. Debugging and testing was carried out on the basis of a traffic generator, the functionality of which can be conditionally divided into two parts: load testing and simulation of the real system behavior. **Core results.** Load testing allowed to identify implementation errors associated with the transmission of messages of minimum size and buffer overflow, as well as incorrect processing of individual ASCII characters when transmitted through Serial. Simulation of the real system allowed to identify errors in the implementation of individual mechanisms of correlation of security events and to produce a correct display of data in the user interface of the system. Also, the simulation of the system allowed to significantly speed up the process of obtaining traffic dumps transmitted between the microcontrollers of the system, including specialized ones. **Practical relevance.** As a result of the work done, the connected library was obtained, which is a unified implementation of the interaction of Arduino-based devices with previously used data sources within the framework of our laboratory. This allowed us to accelerate the exchange of experience among the laboratory staff, to unify the code, and to speed up the process of developing new experimental stands and prototypes of the systems.

Keywords—cyber-physical system, unification of connection, traffic generator, microcontroller.

Article info
Article in Russian.
Received 10.07.18, accepted 03.09.18.

For citation: Levshun D.: Unification of Connection Between Cyber-Physical Systems and Sources of Data // Telecom IT. 2018. Vol. 6. Iss. 3. pp. 28–37 (in Russian).

Введение

Архитектура комплексной системы безопасности состоит из нескольких основных частей (рис. 1): источники данных (*1*, *2*); модуль сбора данных (*3*); модуль обработки событий (*4*); модуль аналитической обработки данных и визуализации (*5*) [1].

Модуль сбора данных подвергает процессам нормализации и предобработки сырые гетерогенные исходные данные. При этом под нормализацией понимается процесс преобразования исходных данных в единый формат. Предобработка исходных данных позволяет изменить типы полей, убрать несущественные данные,

сформировать новые поля. Важно отметить, что исходные данные могут быть условно разделены на данные физического и кибернетического уровней.

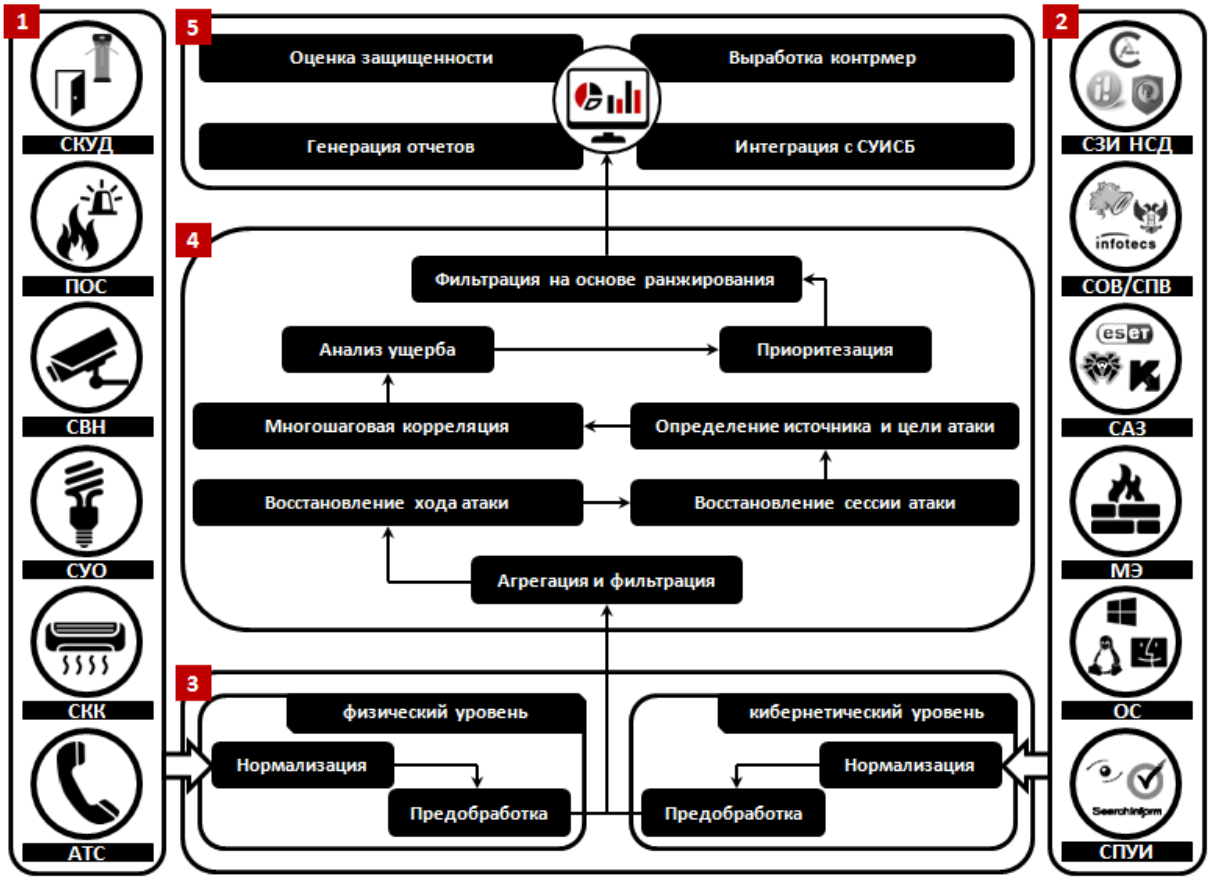

Рис. 1. Архитектура комплексной системы безопасности

Преимущества объединения исходных данных от систем обеспечения физической и кибернетической безопасности заключаются в снижении рисков, связанных с проведением сложных, растянутых во времени и происходящих на разных уровнях киберфизических атак. Это достижимо за счет применения корреляции данных, поступающих от гетерогенных источников [2]. Это даст возможность процессу корреляции событий безопасности в модуле обработки событий (*4*) автоматически формировать инциденты, выявлять аномальную активность, а также определять сценарии атак, выявление которых при использовании современных систем защиты возможно только на этапе расследования [3]. Кроме того, подобная система позволит снизить нагрузку на человека-оператора за счет аналитической обработки данных и визуализации (*5*).

1 Разработка унифицированного взаимодействия с источниками данных

Взаимодействие устройств на основе микроконтроллеров с датчиками, считывателями, извещателями и оповещателями и другими внешними электронными компонентами можно условно разделить на следующие типы: инициализация, считывание, запись, событие и деинициализация (рис. 2).

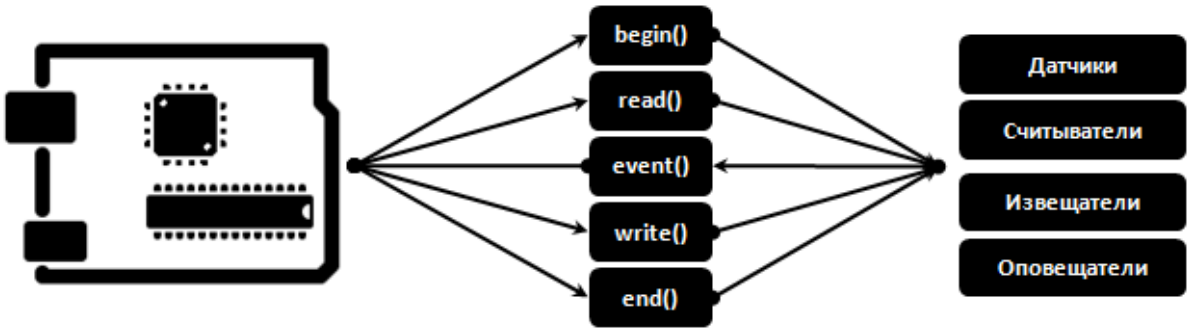

Рис. 2. Взаимодействие с источниками данных

Инициализация (в исходном коде, как правило, выражается через функцию begin()): процесс, позволяющий сформировать в прошивке устройства на основе микроконтроллера необходимые структуры данных для работы с источником данных, а также активировать соответствующие PIN[1] устройства в необходимом режиме.

Считывание (в исходном коде, как правило, выражается через функцию read()): процесс, позволяющий устройствам на основе микроконтроллеров через прошивку осуществлять периодический (в соответствии с исходным кодом) опрос источника данных с целью извлечения показаний датчиков или событий.

Запись (в исходном коде, как правило, выражается через функцию write()): процесс, позволяющий устройствам на основе микроконтроллеров через прошивку осуществлять запись на источник данных с целью изменения его конфигурации, режима работы или частоты передачи данных.

Событие (в исходном коде, как правило, выражается через функцию event()): процесс, позволяющий устройствам на основе микроконтроллеров через прошивку получать информацию от источников данных, связанную, например, с изменением показаний датчика или возникновением события (данный тип взаимодействия во многом аналогичен чтению (read()), однако инициатором взаимодействий выступает не само устройство, а источник данных).

Деинициализация (в исходном коде, как правило, выражается через функцию end()): процесс, позволяющий устройствам на основе микроконтроллеров через прошивку корректно завершить взаимодействие с ранее подключенным источником данных (освободить память от ранее задействованных структур данных и вернуть ранее задействованные PIN в исходное состояние).

Важно отметить, что в зависимости от сложности подключаемого источника данных те или иные типы взаимодействия могут остаться незадействованными. Так, для работы с различными простыми датчиками (температуры, влажности, освещенности и т. п.), на устройствах на основе микроконтроллеров (например, платформы Arduino[2]) достаточно осуществить процесс инициализации используемых PIN и переменных для хранения показаний, а потом в бесконечном цикле осуществлять процесс их считывания через заданные промежутки времени.

[1] Официальная документация для работы с PIN Arduino-устройств. URL: https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode

[2] Официальный сайт для разработчиков устройств платформы Arduino. URL: http://arduino.cc/

Похожая ситуация возникает и при работе устройств на основе микроконтроллеров с различными внешними электронными компонентами, позволяющими направить на устройство сигнал в момент произведения над ними механического воздействия (тактовая кнопка, ключ, тумблер и т. п.). Для них также достаточно произвести только процесс инициализации используемых PIN и переменных для хранения показаний, чтобы начать процесс мониторинга их состояния. Однако вместо инициализируемого устройством процесса считывания, взаимодействие осуществляется по типу событие при замыкании электрической цепи в момент механического нажатия тактовой кнопки или изменения положения ключа / тумблера.

Взаимодействие типа запись необходимо для работы с более сложными устройствами, имеющими несколько режимов работы:

- считыватели RFID [4] и NFC [5], комбинированные датчики погоды или метеостанции и т. п.;
- элементы стендового моделирования: локомотивы, шлагбаумы, переезды, светофоры, сервоприводы и т. п.;
- различные системы: система отопления, вентиляции и кондиционирования, система контроля и управления доступом, система охранной и пожарной сигнализации и т. п.

Данный тип взаимодействия позволяет через прошивку устройства на основе микроконтроллера динамически управлять различными внешними электронными компонентами, меняя их режим работы. При этом управление может осуществляться как уже по заданному сценарию, так и оперативно, в зависимости от поступающего потока событий.

Отдельно стоит отметить взаимодействия типа деинициализация. Данный тип взаимодействия используется для устройств на основе микроконтроллеров, позволяющих осуществлять изменение их конфигурации без загрузки новой прошивки или отключения устройства от сети. Взаимодействие типа деинициализация позволяет освободить вычислительные ресурсы устройства за счет освобождения памяти от ранее задействованных структур данных, а также вернуть в исходное состояние задействованные ранее PIN устройства. Таким образом, после завершения процесса деинициализации, к устройству на основе микроконтроллера может быть подсоединен новый источник данных, процесс работы с которым уже будет необходимо инициализировать.

Техническая реализация унификации взаимодействия с источниками данных в рамках комплексной системы киберфизической безопасности осуществлена на уровне аппаратных интерфейсов (устройств платформы Arduino, объединенных в сеть на основе протокола I2C[3]).

Прошивка устройств платформы Arduino называется sketch и программируется на языке Wiring [6], представляющим собой адаптированную для микроконтроллеров версию языка C. При этом наиболее удобным решением с точки зрения программирования микроконтроллеров оказалась реализация собственной подключаемой библиотеки Sources.h.

[3] Официальная документация библиотеки Wire.h для работы с протоколом I2C. URL: https://www.arduino.cc/en/Reference/Wire

Структура библиотеки Sources.h представляет собой реализацию класса Source с публичными виртуальными функциями begin(), read(), event(), write() и end():

```
class Source {
 public:
  Source() {}
  void virtual begin() {} void virtual read() {}
  void virtual event() {} void virtual write() {}
  void virtual end() {}
};
```

При этом добавление взаимодействия с конкретным источником данных заключается в создании класса наследника Source:

```
class new_source : public Source {};
```

что в дальнейшем позволяет переопределить в рамках нового класса работу всех виртуальных функций.

Задача данной библиотеки – иметь унифицированную реализацию взаимодействия устройств платформы Arduino с ранее задействованными в рамках работы нашей лаборатории различными датчиками, считывателями, извещателями, оповещателями и системами. Это позволит ускорить обмен опытом между сотрудниками лаборатории, унифицировать код, а также ускорить процесс разработки новых экспериментальных стендов или прототипов систем.

Отметим, что, так как структура библиотеки предполагает её постоянное расширение за счёт реализации взаимодействия с всё новыми источниками данных, то достаточно остро встаёт вопрос унифицированного наименования добавляемых источников данных, проверки реализаций взаимодействия на корректность, отсутствие ошибок и опечаток. В нашей лаборатории используется следующий способ наименования:

```
[interface]_[source]_[parameters],
1-wire_RFID_reader_125khz,
```

что позволяет отделить друг от друга источники данных одного типа, имеющие одинаковые параметры.

2 Отладка и тестирование унифицированного взаимодействия с источниками данных

Проектирование комплексной системы безопасности на уровне взаимодействия с источниками данных непосредственно связано с разработкой, отладкой и тестированием взаимодействия аппаратных интерфейсов, представляющих собой устройства на основе микроконтроллеров. В рамках текущей реализации экспериментального стенда, аппаратные интерфейсы объединены в сеть на основе протокола I2C с концентраторами системы в качестве главных устройств шины данных. При этом взаимодействие осуществляется на основе разработанного

нами протокола передачи данных, реализованного поверх базового протокола I2C.

Расширение функциональных возможностей базового протокола I2C с одной стороны позволяет удовлетворить требованиями к надежности и безопасности, а с другой накладывает дополнительные требования по разработке, отладке и тестированию. С целью облегчения решения возникающих задач, разработка новых функциональных возможностей велась модульно и поуровнево.

Под модульной разработкой рассматривается разбиение исходного кода расширенной версии протокола на отдельные независимые модули, выполняющие конкретную задачу. Подобное разбиение, с одной стороны, позволяет при пошаговой проверке разработанных алгоритмов оперативно выявить проблемный модуль и внести в него необходимые изменения без потенциальной возможности повлиять на работу других модулей. С другой стороны, подобное разбиение предоставляет возможность разработки каждого модуля изолированно друг от друга, что позволяет распараллелить работу программистов.

Поуровневая реализация расширенной версии протокола I2C начинается с разбиения процесса разработки протокола передачи данных на независимые друг от друга уровни (например, уровень адресации, уровень аутентификации, уровень шифрования, уровень передачи пакетов данных, уровень парсинга данных и т. д.). Независимость уровней означает, что если нижнеуровневый из них уже был разработан, отлажен и протестирован, то разработка следующего уровня (использующего функционал предыдущего) никак не отразится на его работоспособности. Подобный подход к разработке позволяет сократить время, направленное на отладку и тестирование разрабатываемого протокола передачи данных, упростить описание работы протокола, а также улучшает структуру исходного кода.

Важно отметить, что модульная и поуровневая реализация протокола передачи данные подразумевает не только отладку и тестирование отдельных модулей и уровней протокола, но и протокола в целом. Необходимость полной проверки связана с потенциальной возможность возникновения эмерджентных свойств – свойств, не присущих модулям или уровням по отдельности, однако возникающих при их взаимодействии.

Одним из возможных способов отладки и тестирования унифицированного взаимодействия с источниками данных является разработка генератора трафика. Единственный недостаток подобного решения – искусственность передаваемых по сети данных (необходимо обосновать, что генерируемый трафик соответствует настоящему в рамках проводимых экспериментов), однако для проверки функциональных возможностей протокола и параметров инфраструктуры сети данный недостаток не является существенным. Ключевыми преимуществами применения генератора трафика являются:

- возможность генерации пакетов данных любой длинны или наполнения в рамках заданных ограничений;
- возможность регулирования интенсивности потока данных;
- возможность отладки и тестирования в условиях отсутствия большого количества устройств или ограниченных ресурсов;
- возможность имитации реального потока данных по заранее заданному сценарию.

Разработанный генератор трафика представляет собой прошивку для устройства платформы Arduino (аппаратного интерфейса комплексной системы безопасности). И хотя изначально генератор трафика был разработан именно для комплексной системы безопасности, данная прошивка позволяет организовать процесс генерации трафика в любой киберфизической системе, поддерживающей подключение по протоколу I2C. Функциональные возможности разработанного генератора трафика можно условно разделить на две части: нагрузочное тестирование и имитацию работы реальной системы.

Нагрузочное тестирование изначально было разработано для проверки реализованного поверх базового I2C протокола передачи данных. После отладки и тестирования отдельных модулей и уровней протокола было необходимо протестировать протокол передачи в целом. Изначально ставилась задача проверить, существуют ли ASCII символы или их комбинации, которые при передаче данных от аппаратного интерфейса до концентратора и сервера комплексной системы безопасности приведут к отказу в обслуживании или некорректной обработке данных.

Экспериментальный стенд был организован следующим образом: 3 устройства платформы Arduino в качестве аппаратных интерфейсов (2 *Arduino Mega*, 1 *Arduino Uno*); устройство платформы Arduino в связке с одноплатным компьютером Raspberry[4] в качестве концентратора (*Arduino Mega* и *Raspberry Pi* 3 *Model* B).

На двух аппаратных интерфейсах на основе Arduino Mega был запущен генератор трафика, настроенный на режим работы, при котором добавляемые в буфер события были случайной длины от 1 до 135 байт (короткие и длинные сообщения), при этом содержимое каждого байта события было случайным символом типа char от 0 до 255. Для аппаратного интерфейса на основе Arduino Uno длина событий была от 1 до 27 байт (только короткие сообщения). Поток данных, сформированный аппаратными интерфейсами, направлялся на Arduino часть концентратора для предварительной обработки, нормализации и дальнейшей передачи на Raspberry часть концентратора. В свою очередь, Raspberry часть концентратора также проверяла поступающие в генерируемом потоке данных события и записывала их во внутреннюю базу данных.

Организованный таким образом процесс отладки и тестирования унифицированного взаимодействия позволил выявить ошибки реализации, связанные с передачей сообщений минимального размера и переполнения буферов, а также некорректную обработку отдельных символов ASCII[5] при их передаче по Serial[6]. Ошибки реализации были устранены исправлением кода отдельных модулей, а передача специальных символов ASCII по протоколу Serial была решена с помощью кодирования и декодирования сообщений по стандарту Base64. Кроме того, нагрузочное тестирование, позволило оценить итоговую скорость передачи данных расширенной версии протокола I2C и сравнить её с базовой реализацией [7].

[4] Официальный сайт разработчиков устройств на основе одноплатных компьютеров платформы Raspberry. URL: https://www.raspberrypi.org

[5] Официальный сайт таблицы ASCII кодов. URL: http://www.asciitable.com

[6] Официальная документация для работы с Serial Arduino-устройств. URL: https://www.arduino.cc/reference/en/language/functions/communication/serial/

Имитация работы реальной системы на основе генератора трафика позволяет осуществлять отладку и тестирование подсистем анализа, обработки и визуализации данных. Встроенные алгоритмы позволяют имитировать срабатывания различных датчиков, считывателей, извещателей и оповещателей киберфизических систем, а также регулировать их частоту. При этом структура кода генератора трафика является модульной и поддерживает расширение возможных источников, а также усложнение сценариев генерации трафика, что может быть полезно для более точной имитации целевой системы.

Применение генератора трафика в режиме имитации работы реальной системы позволило выявить ошибки реализации в модуле анализа поступающих событий с целью выявления инцидентов безопасности (отдельные механизмы корреляции событий безопасности работали некорректно). Кроме того, данный режим работы позволил сформировать корректное отображение данных в пользовательском интерфейсе системы. Дополнительным преимуществом имитации работы является возможность проверки системы в условиях сильной нагрузки при отсутствии достаточного количества устройств на основе микроконтроллеров.

Также имитация работы системы позволила значительно ускорить процесс получения дампов трафика, передаваемого между микроконтроллерами системы. А с учетом наличия возможности задания определенных сценариев при генерации трафика, стало также возможным создавать специализированные дампы передаваемых данных, полезные для отладки и тестирования отдельных механизмов корреляции событий безопасности.

Заключение

Данная работа направлена на унификацию взаимодействия киберфизических систем с источниками данных. Данное взаимодействие, как правило, происходит на уровне устройств на основе микроконтроллеров и может быть условно разделено на следующие типы: инициализация, считывание, запись, событие и деинициализация.

Техническая реализация унификации взаимодействия с источниками данных основана на принципах модульности и поуровневости. И в рамках комплексной системы киберфизической безопасности была осуществлена на уровне аппаратных интерфейсов (устройств платформы Arduino, объединенных в сеть на основе протокола I2C).

Отладка и тестирование унифицированного взаимодействия киберфизических систем с источниками данных осуществлялась на основе разработанного генератора трафика. При этом функциональные возможности разработанного генератора трафика можно условно разделить на две части: нагрузочное тестирование и имитацию работы реальной системы.

Нагрузочное тестирование позволило выявить ошибки реализации, связанные с передачей сообщений минимального размера и переполнения буферов, а также некорректную обработку отдельных символов ASCII при их передаче по Serial.

Имитация работы реальной системы позволила выявить ошибки реализации отдельных механизмы корреляции событий безопасности и сформировать корректное отображение данных в пользовательском интерфейсе системы. Также

имитация работы системы позволила значительно ускорить процесс получения дампов трафика, передаваемого между микроконтроллерами системы, в том числе специализированных.

Литература

1. Desnitsky Vasily, Levshun Dmitry, Chechulin Andrey and Kotenko Igor. Design Technique for Secure Embedded Devices: Application for Creation of Integrated Cyber-Physical Security System // Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). Vol. 7. No. 2. June, 2016. pp. 60–80. URL: http://jowua.yolasite.com/vol7no2.php

2. Федорченко А. В., Левшун Д. С., Чечулин А. А., Котенко И. В. Анализ методов корреляции событий безопасности в SIEM-системах. Часть 1 // Труды СПИИРАН. 2016. Вып. 4 (47). С. 5–27.

3. Kotenko Igor, Levshun Dmitry, Chechulin Andrey. Event correlation in the integrated cyber-physical security system // Proceedings of the 2016 XIX IEEE International Conference on Soft Computing and Measurements (SCM-2016), IEEE, St. Petersburg, Russia, May 2016. pp. 484–486.

4. Finkenzeller K. RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication. 3rd Edition. John Wiley & Sons, 2010.

5. Corda A. NFC mobile communication device and NFC reader: пат. 8862052 США. 2014.

6. Ramon M. C. Arduino IDE and Wiring Language // Intel Galileo and Intel Galileo Gen 2. Apress, Berkeley, CA, 2014. pp. 93–143.

7. Levshun Dmitry, Chechulin Andrey, Kotenko Igor. A Technique for Design of Secure Data Transfer Environment: Application for I2C Protocol // The 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS2018). Saint-Petersburg, Russia, May 15–18, 2018. pp. 789–794.

References

1. Desnitsky, Vasily, Levshun, Dmitry, Chechulin, Andrey and Kotenko, Igor. Design Technique for Secure Embedded Devices: Application for Creation of Integrated Cyber-Physical Security System // Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). Vol. 7. No. 2. June, 2016. pp. 60–80. URL: http://jowua.yolasite.com/vol7no2.php

2. Fedorchenko, Andrey, Levshun, Dmitry, Chechulin, Andrey, Kotenko, Igor. An Analysis of Security Event Correlation Techniques in Siem-Systems. Part 1 // SPIIRAS Proceedings. 2016. Iss. 4 (47). pp. 5–27.

3. Kotenko, Igor, Levshun, Dmitry, Chechulin, Andrey. Event correlation in the integrated cyber-physical security system // Proceedings of the 2016 XIX IEEE International Conference on Soft Computing and Measurements (SCM-2016), IEEE, St. Petersburg, Russia, May 2016. pp. 484–486.

4. Finkenzeller, K. RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication. 3rd Edition. John Wiley & Sons, 2010.

5. Corda, A. NFC mobile communication device and NFC reader: serticiate 8862052 USA. 2014.

6. Ramon, M. C. Arduino IDE and Wiring Language // Intel Galileo and Intel Galileo Gen 2. Apress, Berkeley, CA, 2014. pp. 93–143.

7. Levshun, Dmitry, Chechulin, Andrey, Kotenko, Igor. A Technique for Design of Secure Data Transfer Environment: Application for I2C Protocol // The 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS2018). Saint-Petersburg, Russia, May 15–18, 2018. pp. 789–794.

Левшун Дмитрий Сергеевич – младший научный сотрудник, СПИИРАН, Санкт-Петербург, 199178, Российская Федерация, levshun@comsec.spb.ru

Levshun Dmitry – Research Assistant, SPIRAS, St. Petersburg, 199178, Russian Federation, levshun@comsec.spb.ru