



SmartCard-Service

Акционерное общество «СмартКард-Сервис»

127106, г. Москва, Алтуфьевское шоссе, д. 1

Телефон: +7 (495) 981-12-10, 8 (800) 100-31-64, факс: +7 (495) 981-12-11

E-mail: reception@scserv.ru, site: www.scserv.ru

У Т В Е Р Ж Д Е Н О

Генеральный директор

АО «СмартКард-Сервис»

_____ В.А. Васильев

№ _____ «_____» _____ 20__ г.

Программа «Удостоверяющий центр CA.RUS»

ПРОЦЕДУРА ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ ПРИ РАЗРАБОТКЕ ПО

Файл: процедура обеспечения безопасности при разработке ПО.doc

Москва
2024

СОДЕРЖАНИЕ

1. Сведения о документе	3
2. Введение	4
3. Общие положения.....	5
4. Процессы обеспечения безопасности	6
ПРИЛОЖЕНИЕ 1	10
Контрольный лист по результатам анализа исходного кода	10
5. История изменений документа.....	12

1. Сведения о документе

Номер версии:	01.02
Дата выпуска:	18.12.2023 г.
Дата утверждения:	
Порядок обновления документа:	1 раз в год

2. Введение

- 2.1. Настоящий документ определяет требования к процессу разработки программного обеспечения с учетом требований информационной безопасности, в том числе при планировании, написании исходного кода, тестировании безопасности, анализе исходного кода, выпуске и поддержке.
- 2.2. Требования настоящего документа согласованны с требованиями стандартов Payment Card Industry Software Security Framework Secure Software Lifecycle Requirements and Assessment Procedures Version 1.1 (далее - PCI SSF SLC) и Payment Card Industry Software Security Framework Secure Software Requirements and Assessment Procedures Version 1.2.1 (далее - PCI SSF SSS)
- 2.3. Требования, определенные в данном документе, обязательны для выполнения всеми сотрудниками Компании АО «СмартКард-Сервис», участвующими в процессе разработки программного обеспечения.
- 2.4. Источники, используемые при подготовке документа:
 - cwe.mitre.org;
 - www.webappsec.org;
 - gotw.ca;
 - codeguru.com;
 - csrc.nist.gov.

3. Общие положения

- 3.1. Разработка программного обеспечения должна вестись с учетом международного опыта и лучших практик в области безопасного программирования, в том числе:
 - OWASP Application Security Principle;
 - OWASP Development Guide;
 - CERT Secure Coding Standart.
- 3.2. Требования информационной безопасности должны учитываться на всех стадиях создания ПО: при проектировании, написании исходного кода, анализе, тестировании, выпуске релиза, поддержке.
- 3.3. Разработка и тестирование ПО должно производиться с использованием тестовых данных (в том числе тестовых данных платежных карт). Использование реальных данных запрещено.
- 3.4. До выпуска релиза из ПО должны быть удалены как все тестовые данные (например, учетные записи, данные о продуктах и пр.), так и учетные записи используемые приложением (например, данные для взаимодействия с ОС, СУБД и пр.).

4. Процессы обеспечения безопасности

4.1. Блок-схема процесса обеспечения безопасности при разработке ПО приведена на Рис. 1.

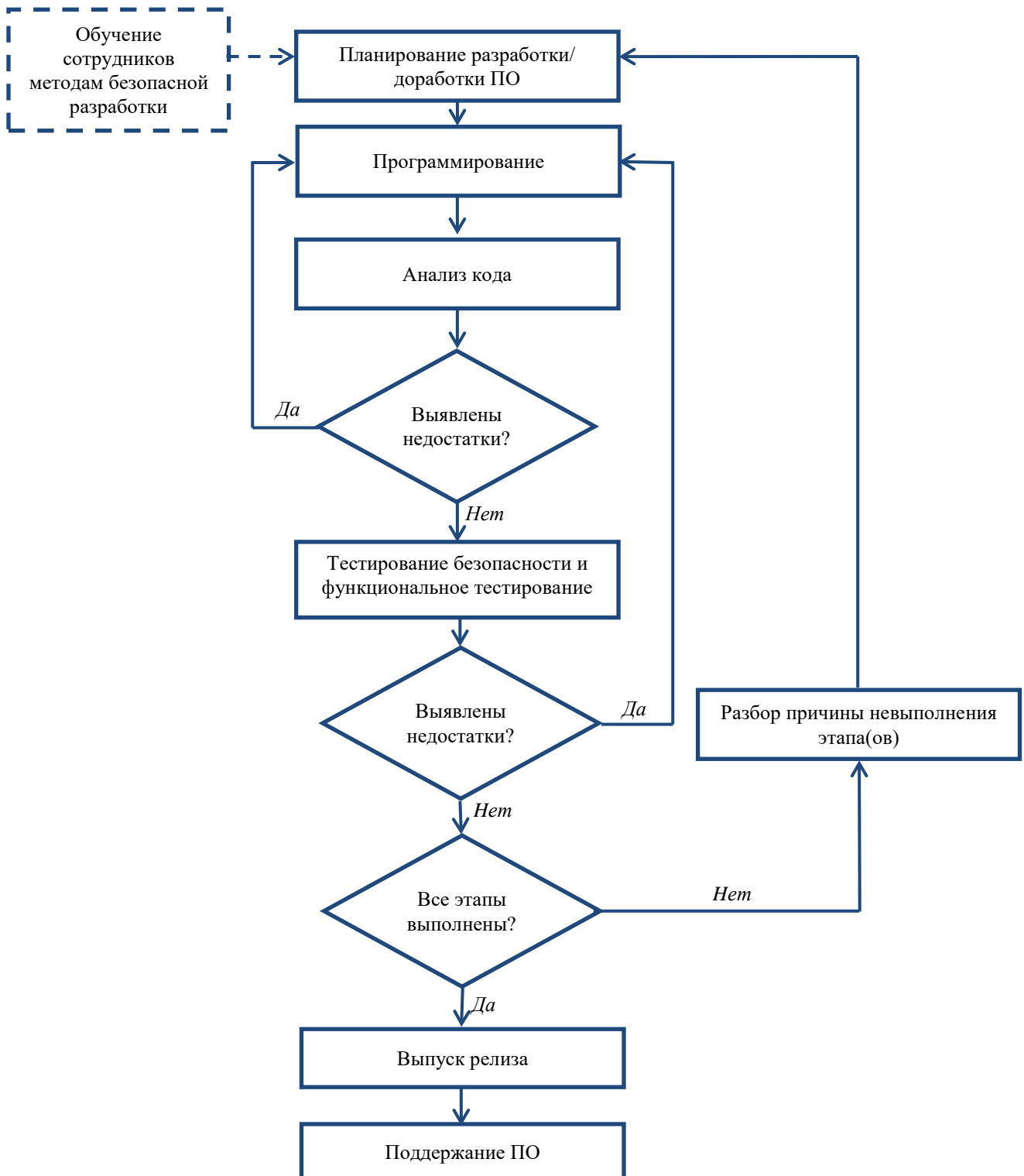


Рис. 1. Блок-схема процесса обеспечения безопасности при разработке ПО

- 4.2. Все сотрудники, участвующие в разработке программного обеспечения, должны проходить обучение приемам и практикам безопасного программирования до начала работ, а также дополнительное ознакомление с новыми видами уязвимостей и практик защиты от них ежегодно.
- 4.3. Обучение может проходить в рамках ознакомления с руководящей документацией, ознакомлений с почтовыми рассылками и информацией внешних ресурсов, групповых или индивидуальных занятий, внешних курсов.
- 4.4. При планировании разработки/доработки ПО необходимо учитывать требования Стандарта, в том числе:
 - требования к парольной политике;
 - использования безопасных механизмов аутентификации;
 - шифрования критичных данных и данных аутентификации, передаваемых по сетям общего пользования;
 - обеспечения защиты при хранении данных;
 - требования к протоколированию событий.
- 4.5. Необходимо документировать процесс обработки данных платежных карт (в том числе и критичных данных авторизации) в памяти платежного приложения.
- 4.6. Необходимо провести оценку рисков, сформировать модель угроз и при необходимости выработать меры по снижению выявленных рисков. Результаты оценки рисков должны использоваться на всех стадиях создания программного обеспечения.

При разработке моделей угроз необходимо учитывать:

- мотивы, которые могут быть у злоумышленника для атаки на программное обеспечение;
- слабые места в программном обеспечении, которые злоумышленник может попытаться использовать;
- возможность использования выявленных слабых мест;
- влияние успешной атаки;
- всю кодовую базу;
- все программные входы/выходы;
- все потоки процессов/данных;
- все границы доверия;
- все точки принятия решений.

Результаты выполнения оценки рисков необходимо фиксировать в соответствующих отчётах, при этом должны обеспечиваться:

- достаточность обоснований решений митигирования конкретных угроз или недостатков конструкции;
- достаточность обоснований всех остаточных рисков.

Процесс внесения изменений в ПО должен включать формирование отчёта оценки влияния такого изменения на безопасность.

В том числе, необходимо выполнять оценку рисков для используемых компонентов программного обеспечения с открытым исходным кодом. Для таких компонентов должно быть подтверждено наличие поддержки (в т.ч. актуальных исправлений безопасности) – в противном случае, эти компоненты должны заменяться активно поддерживаемыми.

4.7. Разрабатываемое приложение должно:

4.7.1. Обеспечивать защиту паролей с использованием хеширования с уникальным значением «соли» для каждого значения;

4.7.2. Ограничивать доступ к необходимым функциям и (или) ресурсам и назначать встроенным учетным записям приложения минимальные права доступа:

- по умолчанию все учетные записи приложения/сервисов имеют доступ только к тем функциям и (или) ресурсам, которые необходимы для их работы;
- по умолчанию все учетные записи приложения/сервисов имеют минимальные права доступа к каждой функции и (или) ресурсу, необходимым учетной записи.

4.8. При написании программного кода необходимо руководствоваться рекомендациями, изложенными в настоящем документе, и оценивать влияние возможных уязвимостей, а именно:

- уязвимостей и слабостей в функциях программирования (<http://cwe.mitre.org/top25/index.html>);
- уязвимостей компилятора;
- уязвимостей общих библиотек ОС, используемых ПО.

4.9. Необходимо использовать следующие методы программирования:

- направленные на минимизацию возможного уровня привилегий для выполнения задач в среде выполнения;
- направленные на защиту от сбоев (любые сценарии выполнения, кроме явно определенных дизайном продукта, запрещены по умолчанию);
- направленные на учет всех возможных точек ввода пользовательских данных (включая различные способы ввода данных).

4.10. Любое изменение (в том числе добавление) программного кода приложения должно анализироваться на предмет безопасности и соответствия принятым в Компании практикам безопасного программирования.

4.11. Анализ исходного кода должен производиться сотрудниками, имеющими достаточный опыт в области безопасного программирования и не участвующим в написании анализируемого участка кода. По результатам анализа рекомендуется заполнять (но не обязательно) контрольный лист, форма которого приведена в Приложении 1.

4.12. При выявлении проблем в процессе анализа, исходный код передается на доработку, при этом доработанный код подлежит повторному полному анализу.

4.13. Тестирование безопасности разработанного ПО должно учитывать, как минимум, следующие уязвимости:

- инъекции кода (проверить входные данные и убедиться, что данные пользователя не могут изменить значения команд и запросов, применить параметризованные запросы и др.);
- переполнение буфера (проверить границы буфера и усечение строк ввода);
- небезопасное хранение материалов шифрования (проверить безопасность хранения данных платежных карт и авторизационных данных);
- небезопасные коммуникации (проверить обеспечение безопасности передаваемых данных платежных карт и авторизационных данных);
- некорректная обработка ошибок (предотвращение утечки информации через сообщения ошибках);

- уязвимости, выявленные на этапе анализа новых уязвимостей и при формировании модели угроз.
- 4.14. Результаты тестирования должны быть задокументированы.
- 4.15. При выявлении проблем в процессе тестирования, приложение передается на доработку, при этом доработанное приложение подлежит повторному полному тестированию.
- 4.16. При выпуске релиза ответственный за проект сотрудник должен убедиться:
- для каждого изменения произведен анализ исходного кода, тестирования функционала и безопасности ПО;
 - разработано уведомление для клиентов, описывающее возможное влияние на работоспособность ПО при его обновлении;
 - разработаны процедуры возврата («отката») на предыдущую версию, в случае неудачной установки нового релиза;
 - из ПО удалены все тестовые и временные учетные данные, конфигурационные данные, используемые при тестировании, тестовые номера карт и др. информация.
- 4.17. Версия выпущенного релиза должна определяться в соответствии с политикой релизов PCI SSF, приведенной в разделе 13. Версии и модификации документа «Общая информация».
- 4.18. Передача нового релиза клиенту осуществляется совместно с информацией о произведенных изменениях, новой версии программного обеспечения и при необходимости включать сведения о возможном влиянии на обновления, на процессы клиента, процедурах «отката» на предыдущую версию, обновленную версию документа «Руководство по внедрению стандарта PCI SSS».
- 4.19. При получении от клиента информации об обнаружении в разрабатываемом ПО недостатков, снижающих общий уровень безопасности приложения, и угроз данным платежных карт, необходимо в 5-дневный срок осуществить проверку наличия недостатков и, в случае подтверждения, в 15-дневный срок принять меры по их устранению.
- 4.20. Если устранение уязвимости в 15-дневный срок невозможно, необходимо разработать и передать клиентам временное решение, обеспечивающее снижение рисков от возможной реализации выявленной уязвимости.

Контрольный лист по результатам анализа исходного кода

Номер релиза: _____ Номер изменения: _____

№	Название	Проверен
	Переполнения буфера	
C1.1	Переполнение буфера в стеке	<input type="checkbox"/>
C1.2	Переполнение буфера в куче	<input type="checkbox"/>
C1.3	Использование небезопасных функций (функций, не проверяющих размер буфера приемника при копировании)	<input type="checkbox"/>
	Нарушение границ массива (структуры)	
C2.1	Выход за границу при чтении (некорректное значение индекса или указателя)	<input type="checkbox"/>
C2.2	Выход за границу при записи (некорректное значение индекса или указателя)	<input type="checkbox"/>
C2.3	Некорректное вычисление индекса массива, основываясь на пользовательских данных	<input type="checkbox"/>
	Целочисленные переполнения	
C3.1	Ошибки преобразования числа со знаком к беззнаковому	<input type="checkbox"/>
C3.2	Ошибки преобразования числа беззнака к знаковому	<input type="checkbox"/>
C3.3	Неправильный расчет размера указателей при арифметических операциях	<input type="checkbox"/>
C3.4	Обрезание значений при приведении типов	<input type="checkbox"/>
	Ошибки при работе с C-строками	
C4.1	Ошибки обработки завершающего нуля при работе со строками	<input type="checkbox"/>
C4.2	Функции strncpy, strncat, snprintf не выставляют завершающего нуля, при достижении предела буфера приемника.	<input type="checkbox"/>
	Опечатки	
C5.1	Присваивание вместо сравнения	<input type="checkbox"/>
C5.2	Сравнение вместо присваивания	<input type="checkbox"/>
C5.3	Некорректное обозначение границ блока (в условиях и циклах)	<input type="checkbox"/>
C5.4	Отсутствие условия default в операторе switch	<input type="checkbox"/>
C5.5	Использование sizeof() над указателем	<input type="checkbox"/>
C5.6	Использование неинициализированных переменных	<input type="checkbox"/>
	Ошибки работы с критичными данными	
C6.1	Ошибки очистки буферов с критичными данными при освобождении	<input type="checkbox"/>
C6.2	Удаление участков кода, при оптимизации компилятором (например, удаление кода, для очистки пароля)	<input type="checkbox"/>
	Race Condition	
C7.1	Race Condition (ошибки синхронизации)	<input type="checkbox"/>

№	Название	Проверен
C7.2	Race Condition (при обработке сигналов или прерываний)	<input type="checkbox"/>
	Ошибки с динамической памятью	
C8.1	Двойное освобождение памяти	<input type="checkbox"/>
C8.2	Освобождение памяти, при существующих ссылках на этот участок памяти (утечка памяти)	<input type="checkbox"/>
C8.3	Отсутствие освобождения всех ресурсов, при обработке исключений	<input type="checkbox"/>
C8.4	Неправильный расчет размера буфера, при выделении памяти	<input type="checkbox"/>
C8.5	Отсутствие проверок при выделении памяти (с условием целочисленного переполнения)	<input type="checkbox"/>
	Другие ошибки	
C9	Дублирующееся значение ключа в ассоциативном массиве (хеше)	<input type="checkbox"/>
C10	Использование фиксированных адресов для указателей	<input type="checkbox"/>
C11	Разыменовывание нулевого указателя (отсутствие проверок возвращаемых данных)	<input type="checkbox"/>
C12	Ошибки форматной строки	<input type="checkbox"/>
C13	Неправильный расчет размера Unicode строки	<input type="checkbox"/>
C14	Некорректное назначение разрешений к критичному ресурсу	<input type="checkbox"/>

Ф.И.О. проверяющего: _____

Дата: «__» _____ 201_ г.

5. История изменений документа

Дата изменений	Версия док-та	Описание изменений
18.08.2021	01.00	Исходная редакция документа, разработанная с учетом требований PCI SSF (версия 3.2.1) и PA-DSS (версия 3.2)
29.08.2022	01.01	Внесены корректировки, учитывающие требования PCI DSS (версия 4.0), PCI SLC v.1.1 и PCI SSS v.1.1
18.12.2023	01.02	Внесены корректировки, учитывающие изменения стандарта PCI SSS v.1.2.1