Лекция 6. Тестирование безопасности

ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2014

Павел Степанов

Старший преподаватель

Кафедра компьютерной математики и программирования ГУАП

1. Содержание

- Важность безопасности
- Стандарты
- Инструменты

2. Безопасность

- Что такое уязвимость?
  - Повышение привилегий кода
  - Крэши и зависания
  - Утечки ресурсов, ведущие к отказу
  - Buffer/stack overrun и underrun
  - Исполнение чужого кода (injection)
  - Несанкционированный доступ к памяти
  - Ошибки защищенных протоколов
    - криптонеустойчивость
    - Изменение данных без расшифровки
    - “Человек посередине”

3. Где важна безопасность

- Платформы и системные библиотеки
- Финансовое ПО
- ПО, оперирующее личными данными
- Военное ПО
- Любое ПО, оперирующее чувствительными данными или вывод которого из строя чреват большими потерями, например, система управления АЭС

4. Основные дыры в безопасности

- Люди (чаще всего)
- Физический доступ
- Известные баги
- Неизвестные баги (реже всего)

5. Стандарты тестирования безопасности

- OSSTMM
- ISACA
- ISSAF
- NIST
- OWASP
- PROTOS
- ГОСТ Р ИСО/МЭК 15408

6. Общее у стандартов

- Вообще говоря, тестирование безопасности не ограничивается только кодом, оно должно быть комплексным
- Поэтому стандарты рассматривают весь спектр программного комплекса, вплоть до помещений и людей
- Ключевая задача стандарта дать оценку защищенности информационной системы и методику вычисления этой оценки

7. OSSTMM

- Open Source Security Testing Methodology Manual
- Вводит определение стандартного теста безопасности и процессов тестирования
- Метрика безопасности rav; калькулятор и формулы для ее вычисления
- охватывает понятия существенно более широкие, чем просто код (структуру и процессы организации, например, а также тестирование сотрудников)

8. ISACA

- Information System Audit and Control Association
  - ITAF – IT Audit Framework
  - COBIT – набор рекомендуемых практик для управления IT
- набор правильных шаблонов как надо делать, включающий безопасность и не только

9. ISSAF (ISSTF)

- sourceforge.net/isstf
- Опенсорсный набор определений и методологий для борьбы с уязвимостями

10. NIST

- National Institute of Standards and Technology, подразделение правительства США
- http://csrc.nist.gov/
- Cybersecurity framework – документ, описывающий кибербезопасность как часть модели управления рисками

11. OWASP

- Open Web Application Security Project
- Фреймворк тестирования веб-приложений на безопасность

12. PROTOS

- University of Oulu (Финляндия)
- Фреймворк тестирования безопасности протоколов

13. ГОСТ Р ИСО/МЭК 15408

- ГОСТ Р ИСО/МЭК 15408-1-2008 Введение и общая модель.
- ГОСТ Р ИСО/МЭК 15408-2-2008 Функциональные требования безопасности.
- ГОСТ Р ИСО/МЭК 15408-3-2008 Требования доверия к безопасности.
- ГОСТ Р ИСО/МЭК 15408 — «Общие критерии оценки безопасности информационных технологий»

14. Вывод по стандартам

- Наименьшее число проблем безопасности приходится именно на уязвимости кода
- Стандарты безопасности рассматривают безопасность как один из рисков
- Стандарты рассматривают все аспекты безопасности, не только код

15. Инструменты тестирования безопасности

- Статические анализаторы кода
- Динамические анализаторы кода
- Фаззинг
- Поиск шаблонов
- Мониторинг ресурсов и прав
- Найм независимых аналитиков

16. Статический анализ

- Простые анализаторы ищут антипаттерны программирования в коде
- Сложные анализаторы пытаются развернуть последовательность вызовов, приводящую к исполнению нежелательного кода
  - Такие статические анализаторы кода являются ноу-хау их владельцев
- Большая часть обнаруженных проблем – просто мусор. Но меньшая часть могут быть очень серьезными
- http://www.isecom.org/research/scare.html

17. Динамический анализ

- Инструментированный код следит, не произошло ли какое-нибудь потенциально опасное событие
- Не может найти проблему, которая не возникла в процессе тестирования ни разу

18. Fuzzing

- Фаззеры – специальный класс программных продуктов, тестирующих код путем подачи на вход искаженной информации
  - Первый класс фаззеров – искажают имеющиеся корректные данные
  - Второй класс фаззеров – пытаются предугадать некорректные данные на основе кода
  - Третий класс фаззеров – данные искажают люди
  - Простейший фаззер передает в код рандомные значения

19. Пример работы фаззера

- В тот момент, когда фаззер передаст на вход функции массив длины меньшей, чем 4, код выдаст исключение

```
void updateArray (int[] array){
    array[3]=5;
}
```

20. Поиск шаблонов

- Определенные фрагменты кода потенциально могут вносить уязвимости

21. Антипаттерн SQL Injection

- String sql = “select field1, field2 from table 1 where field3= ”+x;

22. Антипаттерн plain-text password

- Система авторизации должна сравнивать пароль пользователя с тем, который она считает правильным
- Популярной ошибкой является хранение эталонных паролей в открытом виде
- Вместо этого нужно хранить и пересылать только хеши паролей
- Последний раз встретил этот антипаттерн на mail.ru, когда регистрировал счетчик для своего сайта

23. Антипаттерн buffer overrun (underrun)

- Overrun – выделение буфера фиксированного размера и работа с ним без проверки на длину
- Underrun – обычно происходит со стеком или очередью, попытка выбрать значения из пустого стека или очереди
  - Особенно опасен в нативном коде
  - Опасность в возможности передачи управления в данные

24. Пример

```
public char* copy(char* anotherString){
    char* buffer = calloc(255,1);
    strcpy(anotherString, buffer);
    return buffer;
}

public char* copy2(char* anotherString){
    char[255] buffer;
    strcpy(anotherString, buffer);
    return &buffer;
}
```

25. Почему это проблема безопасности?

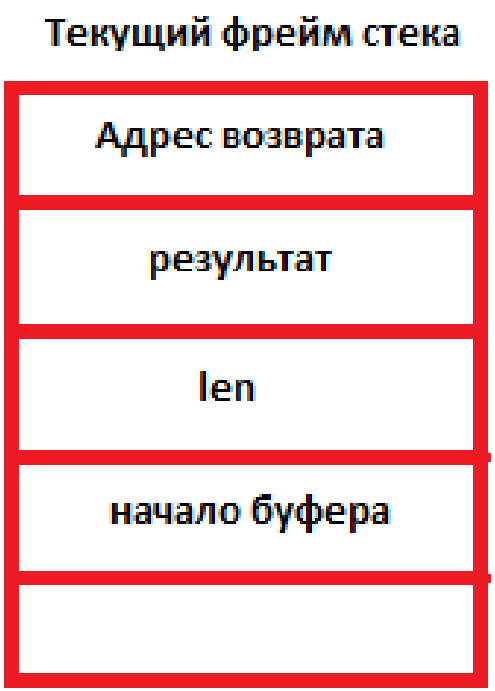
```
wchar_t replaceInCopy(wchar_t* string, int offset, wchar_t newChar) {
        int len = wstrlen(string);
        wchar_t buf[len+1];
        wstrcpy(string,buf);
        buf[offset] = newValue;
        return buf;
}
```

Грань между функциональными ошибками
и проблемами безопасности
иногда довольно тонка

26. Почему это проблема безопасности?

```
wchar_t replaceInCopy(wchar_t* string, int offset, wchar_t newChar) {
        int len = wstrlen(string);
        wchar_t buf[len+1];
        wstrcpy(string,buf);
        buf[offset] = newValue;
        return buf;
}
```

replaceInCopy(someString, -3, somewchar);



Вызывая эту функцию мы можем передать управление в произвольную точку памяти

27. Мониторинг ресурсов

- Утечки ресурсов и зависания могут проявляться при запусках кода обеспечения в бесконечном цикле
  - Если профилировщики показывают, что количество ресурсов постоянно возрастает, это может быть признаком проблемы

28. Мониторинг прав

- При выполнении системного кода если у пользователя повысились права, это может быть признаком уязвимости системного кода

29. Уязвимости протоколов

- Исследователи безопасности регулярно опровергают надежность тех или иных протоколов
  - SSL v3 “пудель” (октябрь 2014)
  - TLS “тройное рукопожатие” (март 2014)
  - RC4 атака (май 2013)
- Растут вычислительные мощности – ключи, считавшиеся ранее устойчивыми, более ими не являются

30. Кто еще находит дырки в безопасности

- Всевозможные конференции аналитиков безопасности
- Хакерские форумы

31. Уязвимости в Интернет

- XSS (межсайтовый скриптинг)
- CSRF (межсайтовая подделка запроса)
- Инъекция кода
- Обход авторизации
- Как найти?

32. Q&A