




Desenvolvimento de Software Seguro

Parte III: Técnicas de Protecção

Segurança de Software

2019
Nuno Santos



Onde estamos

- ▶ **Parte I: Enquadramento e protecção (2 aulas)**
 - ▶ Conceitos de segurança de software, mecanismos básicos de segurança
- ▶ **Parte II: Vulnerabilidades (3 aulas)**
 - ▶ Buffer overflows, corridas e validação de entradas, vulnerabilidades na web e em bases de dados
- ▶ **Parte III: Técnicas de protecção (3 aulas)**
 - ▶ Auditoria e teste de software, análise estática de código, protecção dinâmica, desenvolvimento de software seguro

▶ 2 SS - Nuno Santos 2019



Plano para esta aula

- ▶ Validação de entradas
- ▶ Segurança no desenvolvimento de software



TÉCNICO
LISBOA

Validação de entradas



Motivação

- ▶ Nunca confiar no input
- ▶ Mas o input é necessário, por isso: validar!

- ▶ Validação – garantir que o input satisfaz o seu:
 - ▶ Tipo (ex., contém apenas certos caracteres)
 - ▶ Limites do comprimento (ex., 4 caracteres)
 - ▶ Síntaxe (ex., apenas algarismos)
 - ▶ O maior problema são os metacaracteres, pelo que nos vamos focar nesses



▶ 5

SS - Nuno Santos

2019



Onde efectuar a validação de entradas?

- ▶ **1º princípio:** a validação dos dados tem que ser feita sempre que os dados atravessam uma barreira de confiança
- ▶ ou seja, quando os dados atravessam:
 - ▶ A superfície de ataque da aplicação
 - ▶ Uma barreira de confiança dentro de uma aplicação
 - ▶ Para aplicações multi-tier, a validação tem que ser feita em cada tier (e.g., aplicações web)

▶ 6

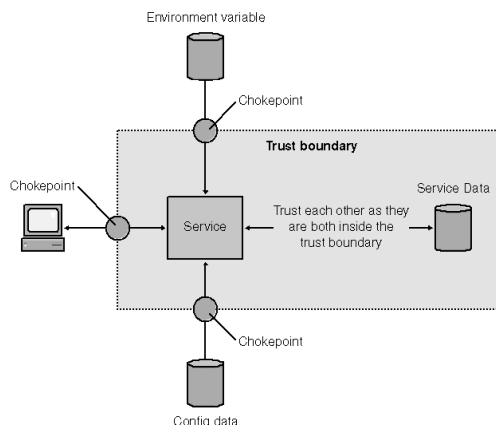
SS - Nuno Santos

2019



Onde efectuar a validação de entradas? (cont.)

- **2º princípio:** tem de existir um pequeno conjunto de pontos de estrangulamento (**chokepoints**) bem definidos onde as validações são feitas



► 7

SS - Nuno Santos

2019



Estratégias de validação de entradas

1 - White listing – aceitar o que sabemos que é bom

```
if (inp.match(regular expression that defines good
input) == false)
    error();
```

- Se espera um código postal, aceitar apenas o código postal (tipo, tamanho, e sintaxe). Se não, rejeitar

```
public String validateZipCode(String zipcode) {
    return (Pattern.matches("^\\d{4}(-\\d{3})?$",
        zipcode)) ? zipcode : '';
}
```

► 8

SS - Nuno Santos

2019



Estratégias de validação de entradas (cont.)

2- **Black listing** – rejeitar o que sabemos que é mau

```
if (inp.match(regular expression that defines bad input)
    == true)
    error();
```

- ▶ Se não espera ver strings JavaScript, rejeitar strings que o contém

```
public String removeJavascript(String input) {
    Pattern p = Pattern.compile("javascript",
        CASE_INSENSITIVE);
    p.matcher(input);
    return (!p.matches()) ? input : '';
}
```

- ▶ **Má estratégia, viola princípio de *fail-safe defaults***

▶ 9

SS - Nuno Santos

2019



Estratégias de validação de entradas (cont.)

3 – **Sanear**

- ▶ Eliminar ou codificar (aka quote ou escape) caracteres por forma a tornar o input seguro
- ▶ Ex.: quote apóstrofes

```
public String quoteApostrophe(String input) {
    return str.replaceAll("[']", "&rsquo;");
}
```

- ▶ **Mesmo problema que “rejeitar o que sabemos ser mau”...**
- ▶ **Mas existem boas bibliotecas para saneamento, pelo que é uma boa opção (de facto, é muito usado)**

▶ 10

SS - Nuno Santos

2019



Muitas vezes não é fácil...

- ▶ Exemplo: Quando input é inserido em comandos SQL, deve ser validado e / ou codificado
- ▶ Porque não validar apenas?
 - ▶ Por vezes não conseguimos remover todos os metacaracteres do input SQL
 - ▶ Ex: queremos remover todas as aspas? Se se o input for um nome, por exemplo O'Connor?



Segurança no desenvolvimento de software



Desenvolvimento de software

- ▶ Objectivos são por vezes contraditórios entre si, e com segurança
 - ▶ Funcionalidade
 - ▶ Usabilidade
 - ▶ Desempenho
 - ▶ Simplicidade
 - ▶ Time-to-market
- ▶ Windows Security Push was a shift-gears on Microsoft's tradeoff
 - ▶ "During February and March 2002, all feature development on Windows products at Microsoft stopped so that the complete Windows development team could analyze the product design, code, test plans, and documentation for security issues. Our team at Microsoft named the process the Windows Security Push. (...) and since then, the company has performed many other pushes across its product line including SQL Server, Office, Exchange, and others."

▶ 13

SS - Nuno Santos

2019



Selecionar a linguagem de programação

- ▶ Linguagem de programação tem impacto na segurança, pelo que deve ser seleccionado tendo este critério em conta
 - ▶ Geralmente outros factores têm mais peso: familiaridade, desempenho...
- ▶ C/C++ é rápido mas sujeito a vulnerabilidades
 - ▶ Ponteiros, falta de verificação de limites de arrays
- ▶ Java/C# não são perfeitas, mas limitam estes problemas
 - ▶ Sem ponteiros, limites de arrays verificados em runtime
 - ▶ Programas correm numa sandbox (implementa access control policies)
 - ▶ Mas existem outras vulnerabilidades, default policies, etc etc
- ▶ PHP é uma fonte de muitos problemas de segurança

▶ 14

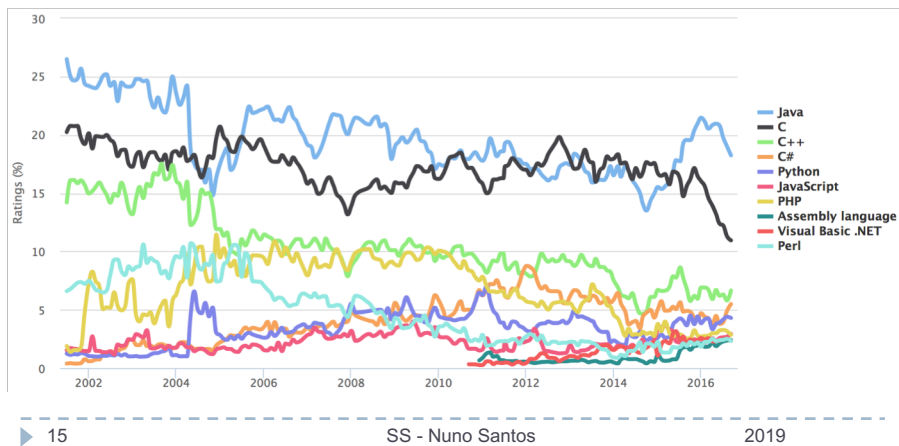
SS - Nuno Santos

2019



Mais popular, mais segura?

- ▶ TIOBE Programming Community Index www.tiobe.com (September 2015)
 - ▶ Dá indicação da popularidade das linguagens de programação
 - ▶ Não diz qual é a melhor linguagem de programação



Open source vs. closed source

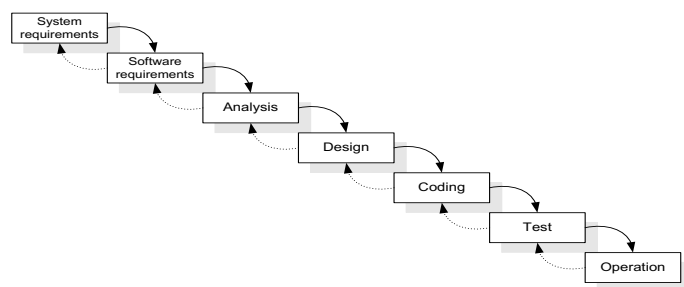
- ▶ Um debate permanente
- ▶ *“Open source is much better because many-eyeballs see the code and detect vulnerabilities”*
 - ▶ Mas olham de facto?
 - ▶ E conseguem ver?
- ▶ *“Closed source is much better because it is harder for the attacker to find vulnerabilities”*
 - ▶ No entanto muitas vulnerabilidades são encontradas em software comercial
 - ▶ Segurança por obscuridade não é uma boa ideia

16 SS - Nuno Santos 2019



Segurança no ciclo de desenvolvimento de SW

- ▶ Como é que a segurança deve ser introduzida no ciclo de desenvolvimento de software?
 - ▶ Desde o princípio e durante todo o ciclo
- ▶ Um processo de desenvolvimento popular: **cascata**



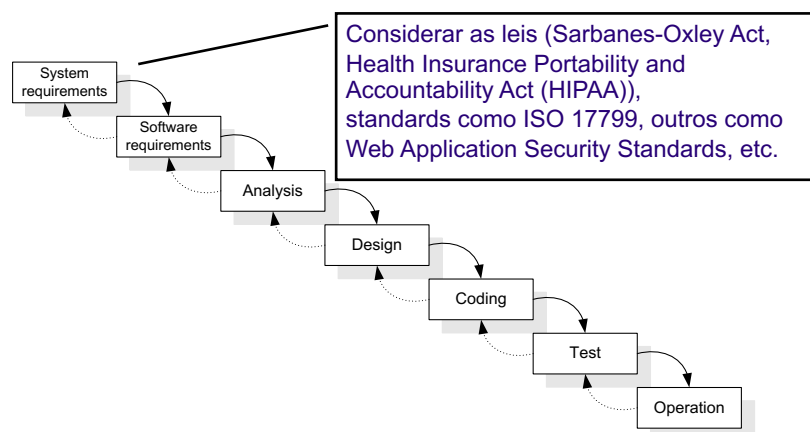
▶ 17

SS - Nuno Santos

2019



Modelo de desenvolvimento em cascata



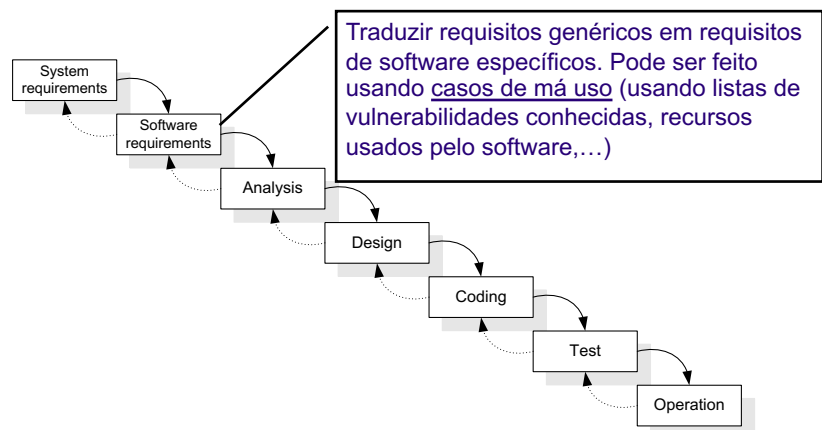
▶ 18

SS - Nuno Santos

2019



Modelo de desenvolvimento em cascata



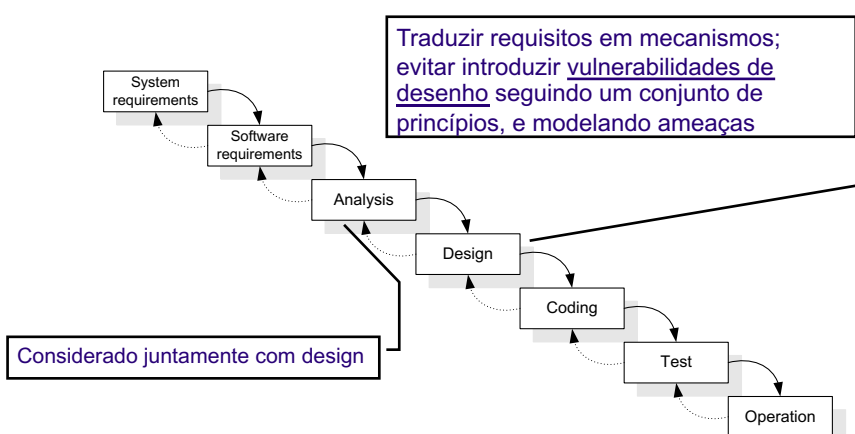
► 19

SS - Nuno Santos

2019



Modelo de desenvolvimento em cascata



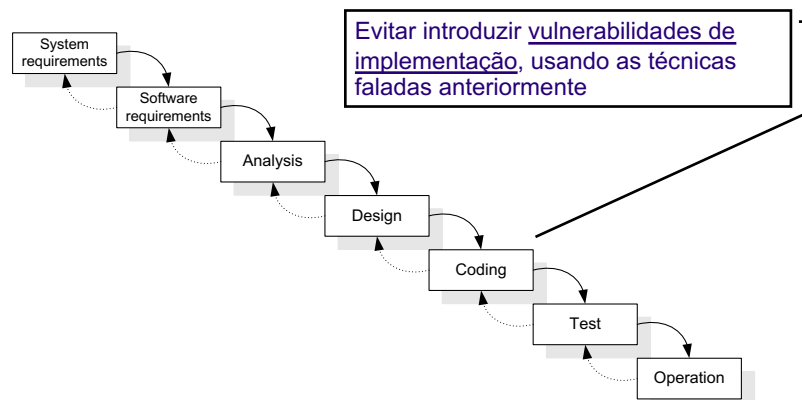
► 20

SS - Nuno Santos

2019



Modelo de desenvolvimento em cascata



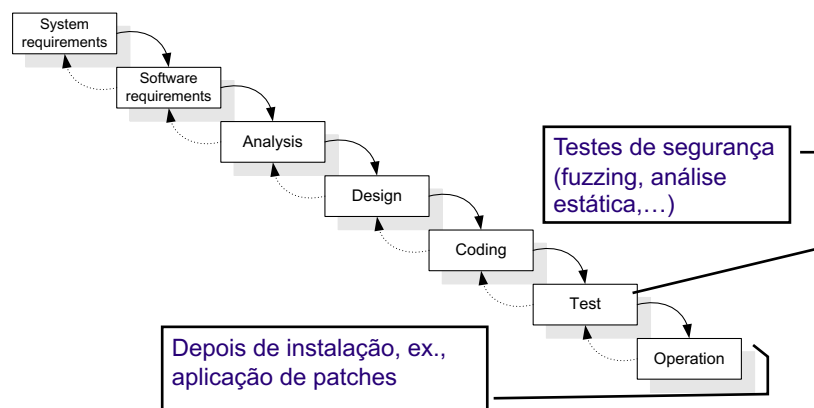
► 21

SS - Nuno Santos

2019



Modelo de desenvolvimento em cascata



► 22

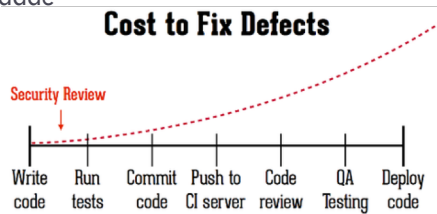
SS - Nuno Santos

2019



Considerar a segurança desde o início

- ▶ Mau princípio: “implementar primeiro, segurar depois” ☹
- ▶ Decisões de desenho no início têm impacto importante na segurança
 - ▶ Colocar segurança mais tarde pode ter custos maiores e implicar grandes revisões no desenho da aplicação
 - ▶ Pode implicar descartar funcionalidade que teve custos a implementar



▶ 23

SS - Nuno Santos

2019



Conclusões

- ▶ Validação e codificação de entradas é fundamental para garantir que dados não confiáveis possam comprometer potenciais vulnerabilidades no software
- ▶ As técnicas de protecção que foram discutidas, não devem ser vistas isoladamente, mas como parte de todo o processo de desenvolvimento seguro de software
- ▶ A segurança é uma faceta fundamental que deve ser tida em conta em todas as fases do desenvolvimento do software

▶ 24

SS - Nuno Santos

2019



Referências

- ▶ **Bibliografia**

- ▶ [Correia17] Capítulos 16 e 2

- ▶ **Próxima aula**

- ▶ Exame