




# Auditoria e Teste de Software

## 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

- ▶ Modelação de ameaças
- ▶ Teste de software: fuzzers
- ▶ Teste de software: scanners de vulnerabilidades



## Modelação de ataques / ameaças



## Motivação

- ▶ “All software projects are guaranteed to have one artifact in common – source code. Together with architectural risk analysis, code review for security ranks very high on the list of software security best practices.”
  - ▶ Brian Chess & Gary McGraw
- ▶ “during the Windows Security Push (...) [Feb-Mar 2002] we found that the most important aspect of the software design process, from a security viewpoint, is **threat modeling**”
  - ▶ Howard & LeBlanc



## Modelação de ataques / ameaças

- ▶ O que é modelação? Porque é importante em tantas áreas?
- ▶ O objectivo é modelar ataques / ameaças, ou seja, caracterizar como é que elas podem afectar um sistema
- ▶ Depois remediar os problemas encontrados
- ▶ Pode ser realizado durante o ciclo de desenvolvimento de software (recomendado) ou ser feito depois



## Benefícios de modelação de ataques

- ▶ **Mais importante: Ajuda a identificar os riscos**
  - ▶ Ajuda a compreender o funcionamento da aplicação
  - ▶ Ajuda a encontrar bugs
  - ▶ Pode ajudar novos membros da equipa a compreender como funciona a aplicação
  - ▶ Documenta a aplicação para outras equipas que a podem estar a utilizar como componente
  - ▶ Ajuda a equipa de teste a definir o que deve ser testado

▶ 7

SS - Nuno Santos

2019



## Passos para modelação de ataques

- ▶ **Passo 0:** Recolha de informação
- ▶ **Passo 1:** Decompor a aplicação em alvos de ataque – criar o modelo
- ▶ **Passo 2:** Identificar as vulnerabilidades
- ▶ **Passo 3:** Descrever os ataques / ameaças para cada alvo
- ▶ **Passo 4:** Classificar o risco para cada ataque

▶ 8

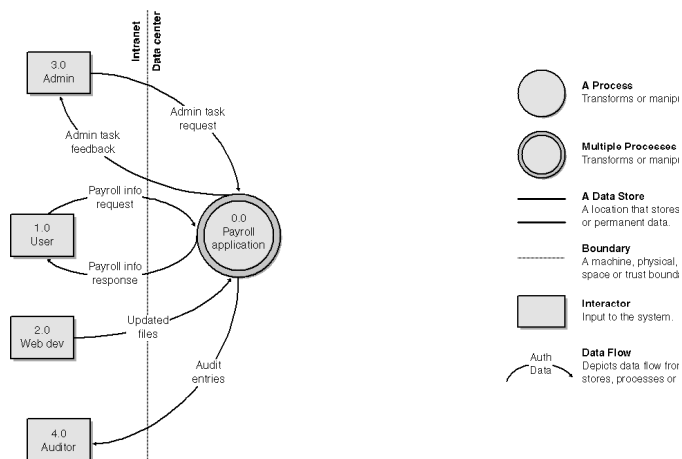
SS - Nuno Santos

2019



## Exemplo de decomposição de aplicação

### ► Caso de estudo: Aplicação para folha de pagamentos



► 9

SS - Nuno Santos

2019



## STRIDE taxonomy: Identificar vulnerabilidades

- **Spoofing identity**
  - Atacante apresenta-se como entidade válida
- **Tampering with data**
  - Modificação maliciosa de dados
- **Repudiation**
  - Negar que um determinado evento aconteceu
- **Information disclosure**
  - Exposição de informação a entidades que não era suposto
- **Denial of service**
  - Negação de serviço por algum componente a utilizadores / components
- **Elevation of privilege**
  - Aumento de privilégios do atacante

► 10

SS - Nuno Santos

2019



## DREAD: Classificação dos ataques / ameaças

- ▶ **Damage potential**
  - ▶ 10 se o atacante consegue ultrapassar todos os mecanismos de segurança
- ▶ **Reproducibility**
  - ▶ Quão fácil é realizar o ataque na prática?
- ▶ **Exploitability**
  - ▶ Quanto esforço e perícia é necessário para montar o ataque?
- ▶ **Affected users**
  - ▶ Se o ataque é bem sucedido quantos utilizadores são afectados?
- ▶ **Discoverability**
  - ▶ Quão fácil é descobrir a vulnerabilidade? Mais prudente: colocar a 10...

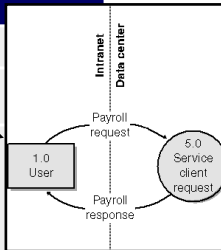
▶ 11

SS - Nuno Santos

2019



## Exemplo de resultado de auditoria

Attack Description	Malicious user views confidential on-the-wire payroll data	
Threat Target	Payroll Response (5.0 → 1.0)	
Threat Category	Information disclosure	
Risk	Damage potential: 8 Reproducibility: 10 Exploitability: 7 Affected users: 10 Discoverability: 10 Overall: 9	
Comments	Most likely attack is from rogue user using a protocol analyzer, because it's an easy attack to perform; the attack is passive and cheap in terms of time, effort, and money. The switch threat is important because many people think switched networks are secure from sniffing attacks when in fact they are not. If you think they are, take a look at "Why your switched network isn't secure" at <a href="http://www.sans.org">http://www.sans.org</a> .	

▶ 12

SS - Nuno Santos

2019

## Fuzzers

## Teste de software usando fuzzers

- ▶ “Operating system facilities, such as the kernel and utility programs, are typically assumed to be reliable. In our recent experiments, we have been able to crash 25-33% of the utility programs on any version of UNIX that was tested.”
  - ▶ Barton P. Miller, Lars Fredriksen e Bryan So, An Empirical Study of the Reliability of UNIX Utilities, CACM, Dec. 1990
- ▶ Estes autores utilizaram uma ferramenta chamada *fuzz*
  - ▶ A ferramenta conseguia reproduzir as condições que faziam crashar os programas
- ▶ Fuzzers do “brute force search for vulnerabilities”
  - ▶ Injectam input aleatório na aplicação, e vê se a aplicação crasha (ou ocorre outra violação de segurança)
- ▶ Hoje o fuzzing é muito usado para testes de segurança
  - ▶ Muitas vulnerabilidades recentes foram descobertas utilizando fuzzing



## Elementos para vectores de fuzzing

- ▶ Pensar em ataques conhecidos: BOs, SQL inj., XSS,...
- ▶ Metacaracters
  - ▶ HTML < >
  - ▶ SQL - ; ' "
  - ▶ OS . / %00 \* | ' '
  - ▶ Web server ../ %00
  - ▶ C / C++ %00
- ▶ Exemplos
  - ▶ String longa: 10K A's
  - ▶ String muito longa: 100K A's
  - ▶ Format strings %n%n%n...
  - ▶ Caminhos ../../../../../../../etc/passwd
  - ▶ Caminhos ../../../../../../../etc/passwd%00
  - ▶ Caracteres invulgares: metacaracters e outros
  - ▶ XSS <script>alert(document.location);</script>

▶ 15

SS - Nuno Santos

2019



## Exemplo de vectores de fuzzing

- ▶ Para testar vulnerabilidades de SQL injection

```
' or 1=1--  
" or 1=1--  
' or 1=1 /*  
or 1=1--  
' or 'a'='a  
" or "a"="a  
) or ('a'='a  
Admin' OR '  
' or 1 in (select @@version)--  
' union all select @@version--
```

▶ 16

SS - Nuno Santos

2019





## Tipos de fuzzers

Em termos de conhecimento do alvo:

- ▶ **Fuzzers magros** – ferramentas simples com pouco conhecimento sobre a aplicação a testar; enviam input aleatório inválido para o alvo
- ▶ **Fuzzers gordos** – podem gerar input que é válido (aceite pelo parser) mas irregular; melhor cobertura

Em termos de especialização:

- ▶ **Especializados** – implementados para tipo específico de aplicação, protocolo de rede, formato de ficheiro
- ▶ **Genéricos** – pode ser aplicado a um espectro alargado de alvos (*fuzzer frameworks*, ex. SPIKE)



## Scanners de vulnerabilidades



## Scanners de vulnerabilidades

- ▶ Método clássico para procurar vulnerabilidades em redes de computadores
  - ▶ COPS, SATAN, Nessus, OpenVAS, LANguard
- ▶ Web vulnerability scanners – para aplicações web
  - ▶ AKA web application scanners, web application security scanners, web application vulnerability scanners



## Scanners de vulnerabilidades para a web

- ▶ NIST definiu 14 classes de vulnerabilidades que devem ser detectadas
  - ▶ XSS, SQLI, command injection, XML injection, HTTP response splitting, file inclusion, direct reference to objects, CSRF, improper information disclosure, broken authentication / weak session management, session fixation, insecure communication, failure to restrict URL access
- ▶ Mas não todas as vulnerabilidades (impossible), só as classes mais bem conhecidas



## Scanners de vulnerabilidades vs. fuzzers

- ▶ Web vulnerability scanners também “injectam ataques”, de forma parecida com os fuzzers
  - ▶ Fuzzers – fazem uma procura exaustiva, pelo que procuram por novas vulnerabilidades
  - ▶ Scanners – utilizam ataques conhecidos e heurísticas para procurar sobretudo por vulnerabilidades conhecidas
- ▶ Web vulnerability scanners são muitas vezes ferramentas comerciais
  - ▶ Acunetix WVS, IBM Rational AppScan, HPWebInspect

▶ 21

SS - Nuno Santos

2019



## Conclusões

- ▶ Modelação de ameaças tem por objectivo modelar de forma exaustiva as vulnerabilidades do software e analisar o impacto de potenciais ataques
- ▶ Para descoberta de vulnerabilidades desconhecidas em software, não muito utilizadas ferramentas de teste chamadas fuzzers
- ▶ Para determinar a existência de vulnerabilidades conhecidas em aplicações, utilizam-se os scanners de vulnerabilidades, especialmente no âmbito de aplicações web

▶ 22

SS - Nuno Santos

2019



## Referências e próxima aula

---

### ► Bibliografia

- [Correia17] Capítulos 12 e 13

### ► Próxima aula

- Análise Estática e Protecção Dinâmica