Programação Segura

Semana 5: Detecção de Vulnerabilidades – Técnicas, Ferramentas

Profª Denise Goya
Denise.goya@ufabc.edu.br – UFABC - CMCC

Técnicas de Detecção

- Vulnerabilidade em software pode ser detectada por meio de uma **variedade de técnicas**
- Nem sempre são eficazes como esperamos
- É preciso entender a técnica e sua limitação e confrontar com o tipo de vulnerabilidade que se deseja buscar:
  - Traçar os **Objetivos técnicos**

Objetivos Técnicos

- Identificar claramente o que se deseja rastrear em um código é o que estamos chamando de **objetivo técnico**
- O Instituto de Defesa dos Estados Unidos fez uma classificação útil dos objetivos, para confrontar com as técnicas

Objetivos Técnicos

Níveis:

- Objetivos
- Sub-objetivos

Table 4-1. Top-level Technical Objectives

1. Provide design & code* quality	
2. Counter known vulnerabilities	
3. Ensure authentication and access control* a. Authentication Issues b. Credentials Management c. Permissions, Privileges, and Access Control d. Least Privilege	
4. Counter unintentional-"like" weaknesses	1. Buffer Handling* 2. Injection* (SQL, command, etc.) 3. Encryption and Randomness* 4. File Handling* 5. Information Leaks* 6. Number Handling* 7. Control flow management* 8. Initialization and Shutdown [of resources/ components]* 9. Design Error 10. System Element Isolation 11. Error Handling* & Fault isolation 12. Pointer and reference handling*
5. Counter intentional-"like"/malicious logic* a. Known malware b. Not known malware	
6. Provide anti-tamper and ensure transparency	
7. Counter development tool inserted weaknesses	
8. Provide secure delivery	
9. Provide secure configuration	
10. Other	

Objetivos Técnicos

- Dada a grande diversidade dos objetivos, fica claro que as ferramentas capazes de tratar cada um deles deve ter abordagens diferenciadas

- O **Projeto** de vocês ficará focado no Objetivo
  - Contra **Fraquezas inseridas não intencionalmente** por programadores (4. *Counter unintentional-like weaknesses* do slide anterior)

Técnicas de Detecção

- Em alto nível, podemos categorizar três grandes conjuntos de técnicas de Detecção de Vulnerabilidades:
  - Estáticas
  - Dinâmicas
  - Híbridas

Técnicas de Detecção Estáticas

- Examinam o sistema ou o software **sem** executá-lo; pode incluir:
  - Análise de código-fonte
  - Análise de bytecodes
  - Análise de binários

Técnicas de Detecção Dinâmicas

- Examinam o sistema ou o software executando-o; podem:
  - Inserir entradas específicas e
  - Analisar os resultados e saídas
  - Analisar o comportamento

Técnicas de Detecção Híbridas

- Integram ambas abordagens
  - Estáticas
  - Dinâmicas

Análise Estática

- Incluem os seguintes Tipos:
  1. Modelo de Ataque
  2. Análise de código fonte
  3. Análise de binário/bytecode
  4. Detecção de código ofuscado
  5. Disassembler de binário/bytecode
  6. Inspeção manual (humana)
  7. outros subtipos . . .

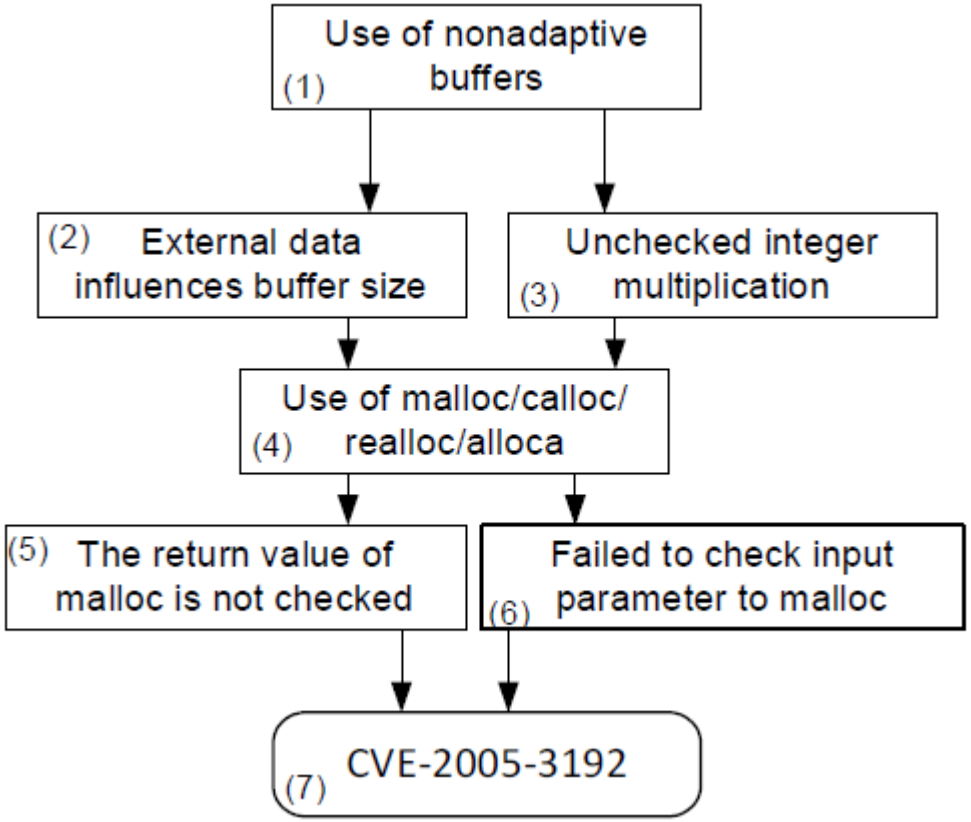
Análise Estática: Modelo de Ataque

- Modelos de ataque modelam a arquitetura do sistema a partir do ponto de vista de um atacante para encontrar fraquezas ou vulnerabilidades
- Exemplos:
  - *Vulnerability Cause Graph (VCG)*
  - *Security Goal Indicator Trees (SGIT)*
  - *Vulnerability Inspection Diagram* (VID)
  - *Security Activity Graphs* (SAGs)

Modelos por Grafos

- Para todas as técnicas mapeiam-se os objetos e suas relações por meio de grafos.
- Exemplo:
- Um **modelo de vulnerabilidade** que ajuda a ter melhor compreensão de **causa-efeito** é o de Grafo de Causa de Vulnerabilidade (ou VCG, *Vulnerability Cause Graph)*

VCG - *Vulnerability Cause Graph*



Análise Estática: Código-Fonte

- Há os seguintes subtipos:
  a) Avisos (warning flags) que podem ser emitidas por um compilador
  b) Analisador de qualidade de software, que apontam más práticas que levam a:
     - Funcionalidade precária
     - Desempenho ruim
     - Custo de manutenção
     - Fraqueza de segurança

Análise Estática: Código-Fonte

c) Analisador de fraquezas, que busca vulnerabilidades conhecidas; na literatura aparece com outros nomes como:

- “source code security analyzer,”
- “static application security testing” (SAST) tool,
- “static analysis code scanner,”
- “code weakness analysis tool”

Análise Estática: Código-Fonte

d) Analisador de fraquezas no código fonte, com configuração de **contexto**:

- Análogo ao anterior, porém pode ser configurado de acordo com o alvo a ser analisado

Análise Estática: Código-Fonte

e) Extrator de conhecimento em código fonte, para informações sobre arquitetura e design

f) Extrator de conhecimento em código fonte configurado para requisitos, para análise de sistemas específicos

Técnicas Análise Estática de Fonte

Algumas técnicas mais conhecidas:

1. Por **detecção de padrões** (pattern matching), em que palavras-chave são buscadas.
   - Ex: em C, a função **getc** é perigosa e deve ser rastreada ; Flawfinder é uma ferramenta que elabora este tipo de técnica

Técnicas Análise Estática de Fonte

Algumas técnicas mais conhecidas:

2. Por **Análise Léxica**: evolução da detecção de padrões, em que são consideradas características da linguagem (como sequências de padrões)
3. Por **parsing**, para análises mais elaboradas que a léxica, olhando toda a representação do código para sintaxe e semântica
   - Ex: usado para identificar SQL injection

Técnicas Análise Estática de Fonte

4. Análise de Fluxo de Dados, para determinar que valores possíveis uma variável pode assumir durante a execução; útil para análise de buffer overflow, de quebra de confidencialidade ou de integridade

Técnicas Análise Estática de Fonte

4. Análise de Fluxo de Dados

- Ex: existe um caminho que vai de um dado sensível (ou secreto) até um canal público? Isto é, o valor de uma variável sensível pode ser propagado ao longo da execução de um programa até ser passado como parâmetro para uma função do tipo printf, fput etc.
  - Caso sim: vulnerabilidade encontrada

Técnicas Análise Estática de Fonte

4. Análise de Fluxo de Dados

- Ex: existe um caminho que vai de uma entrada pública, não confiável, até uma operação sensível?
- Ex: há uma operação de leitura do tipo fscan cuja a entrada pode ser eventualmente propagada para dentro de um vetor cujos limites não são checados?
  - Caso sim: vulnerabilidade encontrada

Técnicas Análise Estática de Fonte

5. Análise de Fluxo de Dado Não-confiável (taint), é um caso especial de análise de fluxo de dados, em que o caminho não leva a uma função crítica, mas precisa ser monitorado até uma confirmação de que é confiável

Análise Estática

- Incluem os seguintes Tipos:
  1. Modelo de Ataque
  2. Análise de código fonte
  3. Análise de binário/bytecode
  4. Detecção de código ofuscado
  5. Disassembler de binário/bytecode
  6. Inspeção manual (humana) . . .

Análise Estática: Binários e Bytecodes

- Há os seguintes subtipos:
  a) scanner de vírus e spyware estáticos: são buscados padrões conhecidos de malware
  b) analisador de qualidade: são buscados padrões de más práticas nos binários
  c) analisador de fraquezas em bytecode: idem para identificar padrões conhecidos

Análise Estática: Binários e Bytecodes

- Há os seguintes subtipos:
  d) analisador de fraquezas em binários: idem para identificar padrões conhecidos
  e) análise de fluxo inter-aplicação, para examinar o fluxo de dados e de controle, para identificar suas interfaces de comunicação e as permissões e, possivelmente, detectar violações da política de segurança

Análise Estática: Binários e Bytecodes

- Há os seguintes subtipos:
  f) extratores: para identificar por exemplo strings que eram para serem de textos, mas que contêm binários ou bytecodes
  g) comparação do binário/bytecode com o manifesto de permissões da aplicação: para identificar inconsistências que possam levar a insegurança

Análise Estática: Código Ofuscado

- Detecção de código ofuscado:
  - código ofuscado pode ser usado para tentar burlar as ferramentas de detecção de vulnerabilidades
    - em geral, ofuscação de código é usado para proteger o código contra engenharia reversa de tecnologia proprietária ou produto crítico.

Análise Dinâmica

- Incluem os seguintes Tipos:
  1. Injeção de falha, em que o software/sistema é executado com injeção proposital de falhas; um mal tratamento de uma falha pode significar bug de segurança
  2. Fuzzing, em que o software/sistema é testado com várias entradas de dados aleatórias e com tamanhos variados.

Análise Dinâmica

3. Análise dinâmica de dado não-confiável (taint): similar ao caso estático, mas monitora as entradas até que elas cheguem nas funções sensíveis; pode reportar vulnerabilidades em que a validade do dado de entrada não foi de todo adequado
4. Sanitização dinâmica: é capaz de detectar situações em que a validação/sanitização da entrada é burlada

Análise Dinâmica

5. Ferramentas de gestão de redes e de segurança:
   - scanners de redes, sniffers, scanner de vulnerabilidades de rede, IDS (Intrusion Detection System), IPS (Intrusion Prevention System), Firewall, sistemas de log, entre outros

Técnicas de Detecção de Vulnerabilidade

Mapeamento das técnicas,
em função dos objetivos e adequação

Technical objective (high-level)								icability (all niques	Attack modeling	Warning flags	Source code quality analyzer	Source code weakness analyzer	Context-configured source code weakness analyzer	Source code knowledge extractor for arch/design coding standards – Extract design, architecture, mission layer	Requirements-configured source code knowledge extractor – Extract design, architecture, mission layer, to aid	Traditional virus/ spyware scanner
								N/A	$	$	$-$$	$$	$$$	$$$	$$$	$
								N/A	EE	E	E	EE	EEE	EEE	EEE	E
4. Counter unintentional-"like" weaknesses	Injection*	Buffer Errors	Incorrect Calculation of Buffer Size	[20]	CWE-131	-	x	✓	–	–	◑	✓	✓	–	–	–
			Classic Buffer Overflow	[3]	CWE-120	-	x	✓	–	–	–	✓	✓	–	–	–
			Other	-	CWE-119 (other children)	-	x	✓	–	–	◑	✓	✓	–	–	–
		Cross-Site Request Forgery (CSRF)		[12]	CWE-352	x	-	✓	–	–	–	◑	◑	–	–	–
		Cross-Site Scripting (XSS)		[4]	(same, child of CWE-20])	x	-	✓	–	–	–	✓	✓	–	–	–
		Code Injection	Unrestricted Upload of File with Dangerous Type	[9]	CWE-434	x	-	✓	–	–	–	✓	✓	–	◑	–
			Download of Code Without Integrity Check	[14]	CWE-494	x	x	✓	–	–	–	–	–	–	–	–
			Other code injection	-	CWE-34 (other)			✓	–	–	–	◑	◑	–	–	–
		Format String Vulnerability		(same)	CWE-134	-	x	✓	–	◑	–	✓	✓	–	–	–
		OS Command Injections		[2]	CWE-78	x	x	✓	–	–	–	✓	✓	–	–	–
		SQL Injection		[1]	CWE-89	x	-	✓	–	–	–	✓	✓	–	–	–
		Input Validation	URL Redirection to Untrusted Site ('Open Redirect') [child of CWE-20]		CWE-601	x	-	✓	–	–	–	✓	✓	–	–	–

Institute of Defense Analyses. **State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation.** Apêndice E: Matriz objetivo x técnica

Técnicas de Detecção de Vulnerabilidade

Technical objective (high-level)	Technical objective (lower-level)	Technical objective (lower-lower-level; Source for most: NVD)	Technical objective (fourth level, based on specific weaknesses)	Warning flags	Source code quality analyzer	Source code weakness analyzer	Context-configured source code weakness analyzer	Traditional virus/ spyware scanner	Bytecode weakness analysis (including disassembler + source code weakness analysis)	Binary weakness analysis - including disassembler + source code weakness analysis
4. Counter unintentional-"like" weaknesses	Buffer Handling*	Buffer Errors	Incorrect Calculation of Buffer Size	–	◑	✓	✓	–	✓	✓
			Classic Buffer Overflow	–	–	✓	✓	–	✓	✓
			Other	–	◑	✓	✓	–	◑	◑

Technical objective (high-level)	Technical objective (lower-level)	Technical objective (lower-lower-level; Source for most: NVD)	Technical objective (fourth level, based on specific weaknesses)	Warning flags	Source code quality analyzer	Source code weakness analyzer	Context-configured source code weakness analyzer	Traditional virus/ spyware scanner	Bytecode weakness analysis (including disassembler + source code weakness analysis)	Binary weakness analysis - including disassembler + source code weakness analysis
4. Counter unintentional-"like" weaknesses	Injection*	Cross-Site Request Forgery (CSRF)		–	–	◑	◑	–	◑	◑
		Cross-Site Scripting (XSS)		–	–	✓	✓	–	✓	✓
		Code Injection	Unrestricted Upload of File with Dangerous Type	–	–	✓	✓	–	–	–
			Download of Code Without Integrity Check	–	–	–	–	–	–	–
			Other code injection	–	–	◑	◑	–	–	–
		Format String Vulnerability		◑	–	✓	✓	–	✓	✓
		OS Command Injections		–	–	✓	✓	–	✓	✓
		SQL Injection		–	–	✓	✓	–	✓	✓
		Input Validation	URL Redirection to Untrusted Site ('Open Redirect') [child of CWE-20]	–	–	✓	✓	–	✓	✓
			Other input validation	–	–	✓	✓	–	◑	◑

Detecção de Vulnerabilidade

- As técnicas existentes atendem a objetivos específicos: acompanhe a planilha anterior
- Nem todos os problemas de segurança são adequadamente detectáveis pelas ferramentas
  - em alguns casos, é preciso mais pesquisa:
    - detecção de malware desconhecido, análise de binários que não tenham os símbolos de depuração
  - em mobile: ferramentas ainda imaturas

Projeto: Análise de Código-Fonte

- Aberta a Atividade para entrega da Fase 2 do Projeto (para entrega até 10/março, 23:55):
  - **artigo preliminar**, já no formato da SBC
  - já com a descrição da ferramenta escolhida
  - busquem artigos técnicos que descrevam sua ferramenta e técnica associada
  - aproveitem as referências e (refs das refs) nas leituras recomendadas
  - incluam uma **planilha** análoga à anterior

Prova 1

- 10/março, na sala de aula
- inclui as técnicas e características da ferramenta escolhida para desenvolvimento do Projeto

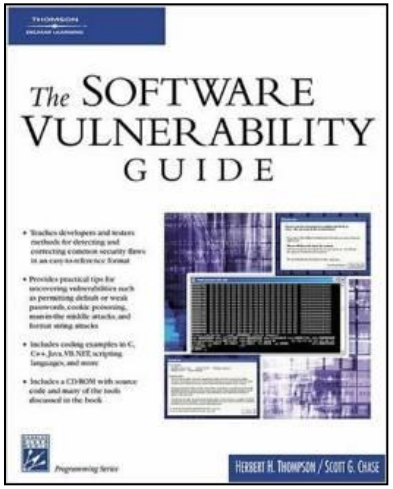
ESTUDO INDIVIDUAL

Leitura Recomendada

- Jimenez, Mammar, Cavalli. **Software Vulnerabilities, Prevention and Detection Methods: A Review**, First International Workshop on Security in Model Driven Architecture, 2009. http://www-lor.int-evry.fr/~anna/files/sec-mda09.pdf.
- Institute of Defense Analyses. **State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation**, 2014.https://www.ida.org/~/media/Corporate/Files/Publications/IDA_Documents/ITSD/2014/P-5061.ashx.
- Center for Assured Software - NSA. **CAS Static Analysis Tool Study - Methodology**, 2012. http://samate.nist.gov/docs/CAS%202012%20Static%20Analysis%20Tool%20Study%20Methodology.pdf.

Leitura Recomendada

- THOMPSON, H.; CHASE, SCOTT G.: "**The Software Vulnerability Guide**". Capítulo 3. Charles River Media, 2005



Exercício

1) Com base na planilha com o Mapeamento das técnicas de detecção em função dos objetivos e adequação, de

> Institute of Defense Analyses. **State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation**, Apêndice E., 2014

mapeie as caraterísticas da Ferramenta escolhida por seu grupo para desenvolvimento do Projeto.