Universidade do Minho
Escola de Engenharia

UNIVERSIDADE DO MINHO

Trabalho Prático – Aula 7

Vulnerabilidade de codificação

Autores:

Bruno Rodrigues: pg41066

Carlos Alves pg41840

1 ÍNDICE

QUESTÃO 1.1

QUESTÃO 1.1.1

CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer
O software executa operações num buffer de memória, mas pode ler ou gravar num local de memória que esteja fora do limite pretendido do buffer. Esta fraqueza pode ser vista em algumas linguagens: C, C++ onde é frequentemente prevalecente.

Em termos de consequências, afeta a disponibilidade, confidencialidade, integridade (ao executar códigos ou comandos não autorizados ou modificar memória). No caso de uma leitura ***out-of-bounds***, o invasor pode ter acesso a informações confidenciais. Se as informações confidenciais contiverem detalhes do sistema, como a posição atual dos buffers na memória, esse conhecimento poderá ser usado para criar novos ataques, possivelmente com consequências mais graves.

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
O software não neutraliza ou neutraliza incorretamente o input controlável pelo usuário antes de ser colocada na saída usada como uma página web que é veiculada a outros usuários.

As consequências mais comuns: confidencialidade, disponibilidade, integridade e do controlo de acesso. Isto pode ser alcançado por exemplo, por vezes pode ser possível executar código arbitrário na máquina da vítima quando o script entre sites é combinado com outras falhas. Ainda o ataque mais comum, envolve a divulgação de informações armazenadas nos cookies do utilizador.

CWE-20: Improper Input Validation
O produto não valida ou valida incorretamente as entradas que podem afetar o fluxo de controlo ou o fluxo de dados de um programa. Independente do idioma, prevalência indeterminada. Afeta Disponibilidade, Confidencialidade e Integridade. No caso da confidencialidade, um atacante pode ler dados confidenciais caso consiga controlar as referências dos recursos. Ao fornecer valores inesperados e causar uma falha no programa, como consumo excessivo de recursos vai afetando a Disponibilidade.

QUESTÃO 1.1.2

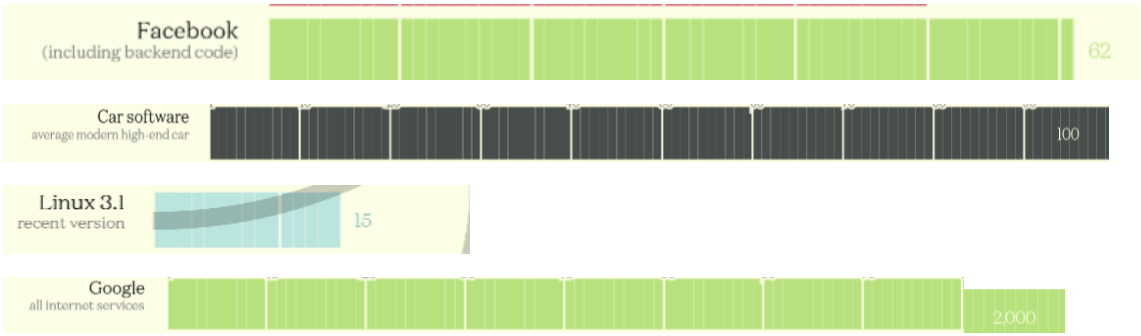
CWE-434: Unrestricted Upload of File with Dangerous Type
A *Weakness* em questão, é descrita como: o software permite que o atacante injete ou transfira arquivos de tipo perigosos que podem ser processados automaticamente no ambiente do mesmo. Essencialmente, esta fraqueza pode ser vista como a falta de restrições no tamanho ou no número de arquivos carregados, o que pode levar a um consumo excessivo de recursos.

Geralmente, esta fraqueza é causada desde o processo de arquitetura e design, pois não são tomadas as medidas necessárias de segurança.

A exploração bem-sucedida, levará primitivamente um impacto sobre a disponibilidade, confidencialidade e integridade. Isto poderá acontecer de fato, se a injeção ou execução de código arbitrário tornar-se possível se o arquivo carregado for interpretado e executado como código pelo destinatário, especialmente extensões *.asp* e *.php* carregadas em servidores web. É ainda classificada com avaliação MÉDIA no quesito da probabilidade de ser explorada.

QUESTÃO 1.2

QUESTÃO 1.2.1



Acima verifica-se a estimativa de linhas de código:

- **Facebook**: 62 000 000
- **Software de automóveis**: 100 000 000
- **Linux 3.1**: 15 000 000
- **Google(All services)**: 2 000 000 000

Em conformidade com a vídeo aula 7, "estima-se que qualquer pacote de software tem uma média de 5 a 50 bugs por cada 1000 SLOC (linha de código fonte), alguns dos quais vulnerabilidades".

$$Limite\ inferior = \frac{5 * nº\ de\ linhas\ de\ codigo\ fonte}{1000}$$

$$Limite\ superior = \frac{50 * nº\ de\ linhas\ de\ codigo\ fonte}{1000}$$

Com isto podemos estimar que:

- **Facebook**: entre 310 000 e 3 100 000
- **Software de automóveis**: entre 500 000 e 5 000 000
- **Linux 3.1**: entre 1 000 e 750 000
- **Google (All services)**: entre 10 00 0000 e 100 000 000

QUESTÃO 1.2.2

Em suma, não deverá ser possível estimar tais vulneráveis com o número de linhas de código fonte.

QUESTÃO 1.3

VULNERABILIDADES DE PROJETO

1. **CWE-272: Least Privilege Violation**

O nível de privilégio elevado necessário para executar operações como chroot() deve ser eliminado imediatamente após a execução da operação.

A solução baseia-se, essencialmente no princípio do menor privilégio ao atribuir direitos de acesso a entidades num sistema de software.

2. **CWE-331: Insufficient Entropy**

O software usa um algoritmo ou esquema que produz entropia insuficiente, deixando padrões ou agrupamentos de valores com maior probabilidade de ocorrer do que outros.

A solução pode ser conseguida durante a fase de implementação, onde se deve determinar a entropia necessária para fornecer adequadamente aleatoriamente e previsibilidade. Basta aumentar o número de bits de objetos, como por exemplo keys e seeds.

VULNERABILIDADE DE CODIFICAÇÃO

3. **CWE-128: Wrap-around Error**

Os erros de quebra automática ocorrem sempre que um valor é incrementado além do valor máximo para seu tipo e, portanto, "envolve" um valor muito pequeno, negativo ou indefinido.

Fornecer limites superiores como inferiores claros na escala de todos os protocolos é uma das potenciais formas de mitigar o problema.

4. **CWE-252: Unchecked Return Value**

O software não verifica o valor de devolvido de um método ou função, o que pode impedir a deteção de estados e condições inesperadas.

As soluções para este problema, podem ser realizadas durante a fase de implementação. Deve--se então verificar os resultados de todas as funções que retornam um valor e verificar se o valor é o pretendido. Alem disso, pode ser incrementado na projeção de uma função uma exceção em caso de erros.

VULNERABILIDADE OPERACIONAL

5. **CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')**

Sem a remoção ou citação suficiente da sintaxe SQL nas entradas controláveis pelo usuário, a consulta SQL gerada pode fazer com que essas entradas sejam interpretadas como SQL em vez de dados comuns do *user*.

Nas fases de Arquitetura e Design pode ser usada uma biblioteca que não permita que esta fraqueza ocorra ou usar mecanismos estruturados que imponham automaticamente a separação entre dados e código.

6. **CWE-134: Use of Externally-Controlled Format String**

Aqui um atacante pode modificar uma sequência de formatos controlada externamente, isso pode causar *buffer overflows*, negação de serviço ou problemas de representação de dados.

Neste exemplo, é possível mitigar o problema na fase: Requisitos, onde podemos escolher um idioma que não esteja sujeita a tal falha. Numa fase final de construção e compilação deveremos ter em atenção aos avisos do compilador.

QUESTÃO 1.4

A diferença entre as vulnerabilidades *zero-day* e as comuns é que as vulnerabilidades de dia zero são vulnerabilidades encontradas por um conjunto restrito de indivíduos, deste modo podem explora-las antes que a restante comunidade/terceiros tenha tempo para as explorar. Enquanto que as outras vulnerabilidades já são conhecidas por toda a comunidade.