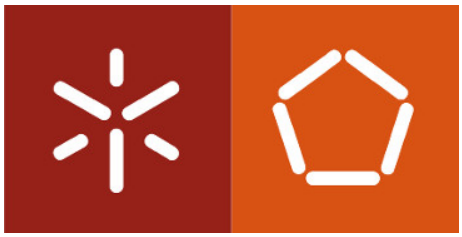
Engenharia de Segurança

27 de Abril de 2021

Grupo 7

a83899	André Morais
a84485	Tiago Magalhães

Prática 1 - Aula 07



Mestrado Integrado em Engenharia Informática
Universidade do Minho

Conteúdo

1 Vulnerabilidade de codificação

1.1 Pergunta 1.1

1.1.1 1)

As primeiras duas *Weakness* presentes no **The CWE Top 25 de 2020** são:

Rank	ID	Name	Score
1	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.82
2	CWE-787	Out-of-bounds Write	46.17

1.1.1.1 CWE-79

A primeira *weakness* é aplicável ao nível das *web pages* mais concretamente no *browser*, uma vez que está relacionada com o facto de o *software* não verificar corretamente o *input* antes de retornar o *output* que irá ser transmitido aos outros utilizadores através de páginas *web*.

Algumas consequências deste tipo de *weakness* são o de a página apresentar conteúdo executável malicioso pelo *browser*, tais como código *javascript*, *HTML Scripts*, *Flash*, entre outros, que poderão comprometer o utilizador.

As principais consequências afetam a:

- **Confidencialidade e o Controlo de Acessos** : Acontece quando um utilizador constroi um script do lado do cliente que realiza uma ação (como por exemplo enviar os *cookies* do site para um dado email). Este script irá ser carregado e corrido em todos os clientes que visitam o site. Desde que o pedido da página ao servidor tenha acesso às cookies, o script também terá.

- **Confidencialidade, Integridade e a Disponibilidade**: Acontece quando um utilizador consegue correr código arbitrário no computador de uma vítima quando o Cross-site Scripting é combinado com algumas falhas.

- **Confidencialidade, Integridade, Disponibilidade e o Controlo de acessos**: Este tipo de ataques baseia-se nos dois anteriores, diferenciando apenas o payload que chega ao servidor

1.1.1.2 CWE-787

Quanto à segunda *weakness* esta é mais suscetível em linguagens como o C, uma vez que é necessário o programador controlar a alocação de memória que poderá dar origem a que os dados sejam escritos fora do limite do *buffer*.

As principais consequências afetam a integridade e a disponibilidade, uma vez que se pode alterar dados na memória, bem como a aplicação *crashar*.

1.1.2 2)

Sendo o grupo 7, na posição 9 do **The CWE Top 25 de 2020** temos a *weakness*: **CWE-352**

CWE-352 : Cross-Site Request Forgery (CSRF) Esta *weakness* caracteriza-se por quando o serviço *web* não verifica se um pedido foi realmente realizado por quem o submeteu. Esta aplica-se ao nível das tecnologias *web*. As principais consequências são ao nível da confidencialidade, integridade, disponibilidade, não repúdio e controlo de acesso, uma vez que utilizador mal intencionado pode fazer passar por outro, alterar dados, ganhar privilégios e passar por mecanismos de proteção.

1.1.2.1 Exemplo de Código

```
<SCRIPT>
function SendAttack () {
form.email = "attacker@example.com";
// send to profile.php
form.submit();
}
</SCRIPT>

<BODY onload="javascript:SendAttack();">

<form action="http://victim.example.com/profile.php" id="form
   " method="post">
<input type="hidden" name="firstname" value="Funny">
<input type="hidden" name="lastname" value="Joke">
<br/>
<input type="hidden" name="email">
</form>
```

Este formulário contem campos escondidos e assim que o *browser* do utilizador carregue, a página não irá notar. Como a funcão *SendAttack*(), se encontra no atributo *onload* vai ser automaticamente executada quando o utilizador carrega a página. Assumindo que utilizador está *logado* no *website*, este terá uma sessão estabelecida válida, e irá dar *update* do seu *email* para o do atacante. Consequentemente o atacante terá acesso a esta identidade e poderá mandar mensagens pelo seu email.

1.1.2.2 Exemplos de CVE

- **CVE-2004-1703** : Add user accounts via a URL in an img tag
- **CVE-2009-3520** : modify password for the administrator
- **CVE-2009-3759** : web interface allows password changes or stopping a virtual machine via CSRF

1.2 Pergunta 1.2

1) Considerando que em cada 1000 linhas de codigo existem entre 5 a 50 bugs temos:

Software	SLOC	Limite Inferior de Bugs	Limite Superior de Bugs
Facebook	62M	0.31M	3,1M
Software de Automóveis	100M	0.5M	5M
Linux 3.1	15M	0.075M	0.75M

2) Não é possível estimar um número de vulnerabilidades, uma vez que não existe uma relação entre estas e o número de *bugs*.

1.3 Pergunta 1.3

1.3.1 Vulnerabilidades de projeto

- **CWE-20: Improper Input Validation**: O *software* recebe *input*, que não é corretamente validado. Para mitigar esta vulnerabilidade pode-se usar uma *framework* de validação.
- **CWE-36: Absolute Path Traversal**: O *software* utilizada *input* externo para construir um caminho que deveria estar dentro de uma diretoria restrita. Para mitigar é necessária validação de *input*.

1.3.2 Vulnerabilidades de codificação

- **CWE-129: Improper Validation of Array Index**: O *software* utiliza *input* não validado que quando calcular ou usar o índice de um *array* poderá ser um posição inválida do *array*. Para mitigar deve-se também seguir um estratégia de validação de *input*.
- **CWE-252: Unchecked Return Value**: O *software* não verifica o valor retornado de um método ou função, o que pode levar a condições

e estados não expectáveis. Para mitigar deve-se ter a certeza que se irá retornar um valor ou exceção em caso de erro.

1.3.3 Vulnerabilidades de operacional

- **CWE-209: Generation of Error Message Containing Sensitive Information**: Esta vulnerabilidade ocorre quando os erros de uma aplicação revelam informação sensível para um atacante. Para mitigar deve-se assegurar que as mensagens de erro apenas têm a informação mínima necessária.

- **CWE-5: J2EE Misconfiguration: Data Transmission Without Encryption**: Devido a esta vulnerabilidade, a informação enviada para uma *network* pode ser compremetida. Um atacante pode ser capaz de ler o mudar o conteúdo, se a informação for mandada em texto simples ou fracamente cifrado. Para evitar esta situação, todas as configurações, especialmente as de cifragem, devem ser verificadas antes de correr a aplicação

1.4 Pergunta 1.4

Uma vulnerabilidade dia-zero é uma vulnerabilidade que apenas é conhecida num meio restrito, e portanto, não são públicas nem têm CVE atribuído. Estas vulnerabilidades são muitas vezes vendidas no mercado negro por valores exorbitantes. As vulnerabilidades que não sejam de dia-zero é uma vulnerabilidade conhecida, isto é, é do conhecimento dos desenvolvedores e podem prevenir que seja explorada (Na grande maioria das vezes são *patched* antes de serem publicadas).