Pergunta P1.1

De acordo com os estudos realizados, a cada 1000 linhas de *source code*, o número de bugs de qualquer software varia entre 5 e 50 (limites inferior e superior), alguns dos quais são vulnerabilidades (não se sabe ao certo a percentagem de bugs que origina vulnerabilidades).

Tendo o software do Facebook 60 milhões de linhas de *source code*, estima-se que este tenha entre 300 mil e 3 milhões de bugs, alguns dos quais são vulnerabilidades (não se sabe ao certo).

O software de automóveis têm 100 milhões de linhas de *source code*, das quais possui entre 500 mil a 5 milhões de bugs.

O software Linux 3.1 possui 15 milhões de linhas de *source code*, com um número de bugs entre 75 mil a 750 mil.

Por fim, todos os serviços Internet da Google tem 2 biliões de linhas de *source code*, contendo entre 10 milhões e 100 milhões de bugs.

Pergunta P1.2

Considere-se as seguintes vulnerabilidades possíveis de ser exploradas nos seguintes contextos:

1. **Desenho do projeto**
    a. Levantamento errado de requisitos;

O levantamento incorreto de requisitos aquando especificação, implica que as fases seguintes de desenvolvimento refletirão estes mesmos erros. Um exemplo prático poderá estar relacionado com os níveis hierárquicos de acesso a informações cruciais, onde caso esta distinção não seja explícita inicialmente, nas fases seguintes não será implementada no projeto. Este problema poderá ser fácilmente resolvido fazendo a revalidação dos requisitos, bem como assegurar de que o desenvolvimento vai ser efetuado de acordo com estes.

    b. Utilizadores não alteram as variáveis definidas por defeito;

Um exemplo desta vulnerabilidade pode ser expressa pela inalteração das credenciais de autenticação de um determinado serviço ou aplicação. Um exemplo disto poderá ser a utilização de passwords definidas por defeito para acessos a sistemas de bases de dados. A solução é extremamente simples, visto que apenas é necessário aos utilizadores responsáveis por estas, a alteração dos dados de acesso.

2. **Codificação da aplicação:**
    a. Utilização de API's desatualizadas (ou fontes desconhecidas);

Este problema afeta todo o *software* desenvolvido que contenha fontes de código externa, *APIs*, bibliotecas, entre outros. Esta vulnerabilidade pode ser expressa pela inexistência de atualizações relativamente a bibliotecas antigas. Um exemplo prático desta vulnerabilidade passa por bibliotecas ou *APIs* de segurança, onde é necessária a manutenção do código de modo a que estas estejam sempre o mais atualizadas possível. Os mesmos problemas podem ser causados pela utilização de código de fontes desconhecidas, onde, caso não seja explícito o intuito do código, este não deverá ser utilizado.

    b. Transferência de dados importantes não cifrados;

Imagine-se que se pretende distinguir níveis de acesso a determinadas informações, é do interesse que estas estejam cifradas de alguma forma, caso contrário, apesar de uma correta implementação destes mesmos níveis de acesso, um utilizador mal intencionado que consiga ter acesso aos dados poderá ver esta mesma informação, suposta confidencial. Uma resolução simples para este problema passa por cifrar toda a informação, e possivelmente adaptar a segurança de cada cifra de acordo com cada nível, atribuindo uma maior segurança na informação mais crítica e garantir maior eficiência no tratamento da informação pública.

3. **Operacional:**
    a. Proteger sessões de utilizador com segurança extra (habilitar algum tipo de proteção de conta, dado um *timeout*);

Esta vulnerabilidade passa pela existência de computadores com acesso a informação crítica que não apresentam segurança de acesso à informação. Deste modo, apesar de um *software* não conter qualquer tipo de vulnerabilidade, o meio de acesso à informação apresenta falhas de segurança. De que nos servirá implementar níveis de acesso à informação, se todos os computadores de uma empresa não tiverem proteção a nível de contas de acesso a sessões de cada utilizador? De nada. Esta vulnerabilidade pode ser facilmente resolvida, implementando mecanismos de acesso seguro às contas (por ex.: palavras-passe) e até mecanismos que bloqueiam automaticamente a sessão dado um tempo de inatividade (proteções de ecrã / *screensavers*).

    b. Remoção de dados em formato analógico;

Este problema é expresso pela tendência de manter informação impressa nas secretárias de cada funcionário. Não fará sentido, mais uma vez, implementar mecanismos de segurança de informação, para que depois esta seja impressa e exposta numa folha de papel física, onde qualquer indivíduo que por esta passe tenha acesso à informação nela contida. Assim, sempre que possível deverão ser utilizados apenas acessos digitais, quando isto não for possível deveremos assegurar que se destrói a informação irreversivelmente, quando esta já não for necessária.

Pergunta P1.3

Uma vulnerabilidade dia-zero é uma vulnerabilidade que é desconhecida àqueles que teriam o interesse de a mitigar. É uma vulnerabilidade que é desconhecida do público em geral mas conhecida por um grupo restrito de pessoas.

Estas vulnerabilidades geralmente são pesquisadas por grupos organizados (instituições militares ou grupos de piratas informáticos) uma vez que ainda exigem alguns recursos, para identificar e criar formas de explorar vulnerabilidades que ainda são desconhecidas, com o objetivo de realizar ataques sem que a entidade a ser atacada se aperceba ou se possa defender dos mesmos.

Geralmente refere-se ao “dia-zero” como sendo o momento em que a entidade com a vulnerabilidade se apercebe da mesma podendo então passar a atuar no sentido da mitigação e/ou correcção da mesma.

Um ataque dia-zero permite atacar sistemas administrados por equipas competentes e com bons conhecimentos de segurança, uma vez que visam atacar vulnerabilidades ainda desconhecidas das mesmas.

Estas vulnerabilidades distinguem-se das outras no sentido em que a entidade que possui a vulnerabilidade desconhece a existência da mesma.