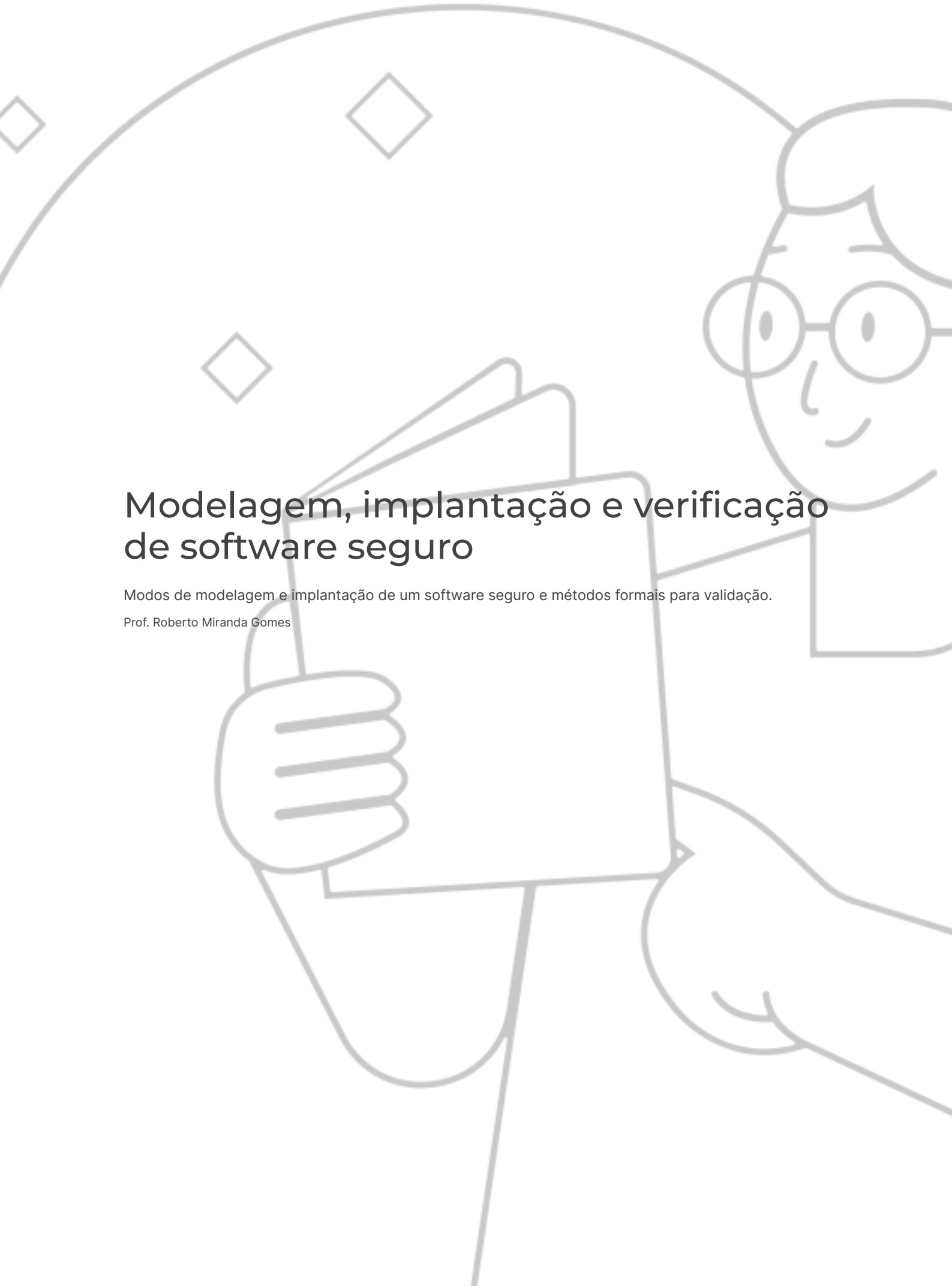


Modelagem, implantação e verificação de software seguro

Modos de modelagem e implantação de um software seguro e métodos formais para validação.

Prof. Roberto Miranda Gomes

Propósito

Conhecer os conceitos de desenvolvimento de software seguro relacionados à modelagem, verificação e implantação de software é fundamental para desenvolver sistemas seguros.

Objetivos

- Reconhecer conceitos empregados na avaliação de segurança de software.
- Identificar as etapas para a modelagem de ameaças.
- Reconhecer as etapas para implantação de software seguro.
- Identificar modelos formais de segurança de software.

Introdução

A segurança de software é uma área relevante na tecnologia da informação, pois é responsável por dar garantias mínimas de proteção de dados e disponibilidade de sistemas contra ameaças de ataques cibernéticos. A mentalidade de segurança deve permear o ciclo de vida completo de desenvolvimento de software seguro, introduzindo medidas protetivas específicas para cada fase.

Boas práticas, padronizadas por processos estruturados, devem ser empregadas para gestão de risco e modelagem de ameaças. Diversas organizações propõem arcabouços estruturados para cobrir essas questões práticas de desenvolvimento de software seguro, sendo excelentes fontes de consulta para todo profissional envolvido com a área.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A importância do software seguro

Confira, neste vídeo, a importância da segurança de software e os seus conceitos fundamentais. Veja também bugs, vulnerabilidades e exploits, e entenda como eles podem ser explorados por invasores para roubar informações.

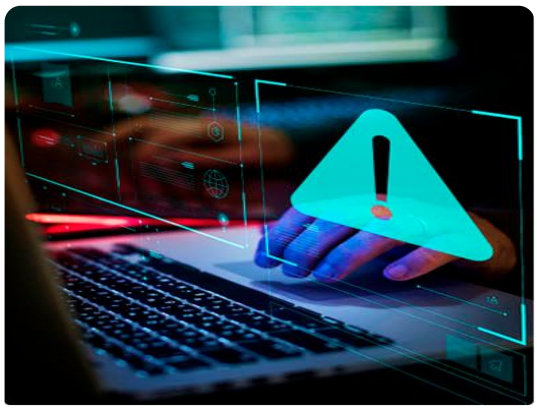
> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Você já viu um exploit em ação? Parece mágica a forma como um pedaço de código é executado rapidamente e toma o acesso de uma máquina que roda um programa complexo, escrito por uma equipe de especialistas e vendido e usado no mundo todo já por alguns anos.

O processo pode parecer algum tipo de magia computacional, tendo como ingredientes uma grande confusão de bits e bytes. Mas quando você der uma olhada no que acontece por debaixo dos panos e perceber como, de fato, funciona, certamente, simpatizará com o criador do código, que teve muita persistência para testar ações (algumas delas bem inusitadas) até encontrar um caminho capaz de furar o uso normal do sistema, ultrapassar os mecanismos de controle e tomar conta da máquina.

O cenário apresentado aqui é muito factível, por exemplo, em um caso de ataque do tipo buffer overflow, que ocorre quando um programa tenta armazenar mais dados em uma área de armazenamento temporário na memória (buffer) do que ele pode suportar, o que faz com que o excesso de dados “transborde” e seja armazenado em outras áreas da memória. Isso pode permitir que um invasor execute um código malicioso ou manipule o programa de maneiras favoráveis a, potencialmente, tomar o controle total do sistema.

Se você conhece um pouco de software, sabe que todos eles possuem bugs. Todo software, em algum momento após seu lançamento, vai apresentar uma falha, mau funcionamento diante do inesperado (uma ação executada pelo usuário nunca imaginada pelos programadores do sistema) ou até mesmo frente ao esperado. As possibilidades de buracos existentes em um sistema complexo são inúmeras! Seja por descuido ou desconhecimento, os programadores constroem sistemas com falhas, isso é um fato. Quanto maior e mais complexo o software, mais provável a existência de falhas que abram brechas a serem exploradas.



Sistema apresentando falha.

No contexto da segurança de software, alguns conceitos são de extrema importância, como a terminologia de base para compreensão de aspectos práticos. Vamos a alguns desses conceitos!

Vulnerabilidades

Confira, neste vídeo, os conceitos teóricos relacionados com a exploração de vulnerabilidades e aprenda a solucionar os bugs e as falhas de um software.

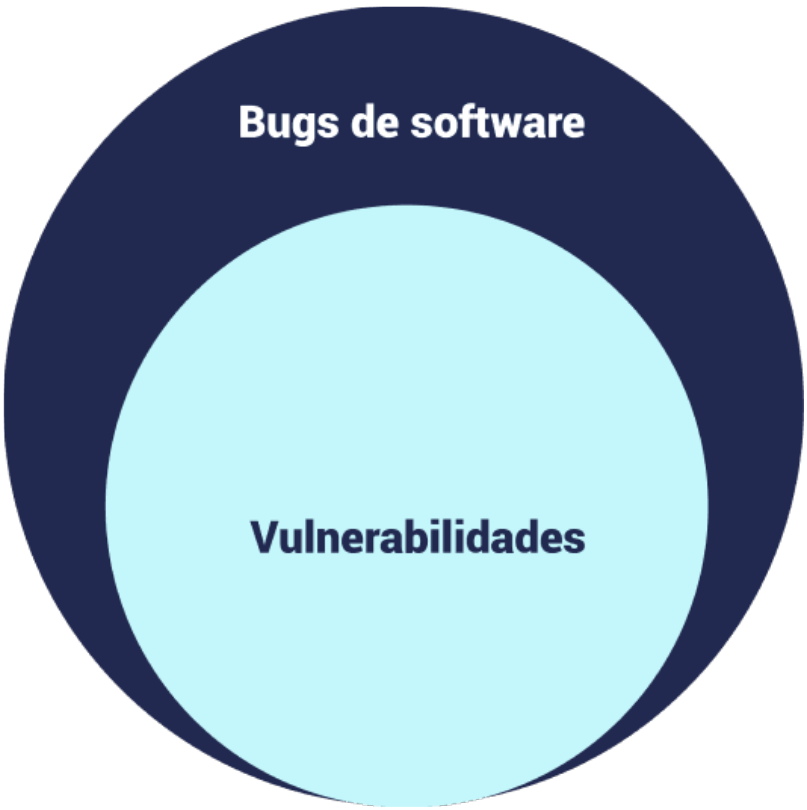
Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Em termos simples, são os pontos fracos de um sistema, os quais os invasores podem aproveitar a seu favor.

> Uma vulnerabilidade é uma falha específica gerada por desconhecimento ou descuido em um software, que possibilita que um invasor faça algo mal-intencionado; exponha ou altere informações confidenciais; interrompa ou destrua o funcionamento do sistema; ou assuma o controle de um sistema ou programa de computador.

Sem dúvida, você deve estar familiarizado com bugs de software. São erros, enganos ou descuidos em programas que resultam em comportamento inesperado e tipicamente indesejável. Quase todo usuário de computador, em algum momento, já perdeu um trabalho importante por causa de um bug de software. Nem todo bug cria vulnerabilidades, ou seja, em geral, as vulnerabilidades de software podem ser consideradas um subconjunto do fenômeno maior de bugs de software. Entenda melhor:



Relação entre bugs e vulnerabilidades.

Vulnerabilidades de segurança são aqueles bugs ou, genericamente, falhas que trazem uma brecha para que um usuário mal-intencionado possa se aproveitar para lançar ataques contra o software e os sistemas de suporte.

Todas as vulnerabilidades de segurança advêm de bugs de software, mas apenas alguns bugs de software acabam se configurando em vulnerabilidades de segurança. Uma falha deve proporcionar algum impacto ou liberar propriedades relevantes à segurança para ser realmente considerada um problema. Em outras palavras, a falha deve permitir que os invasores façam algo que, por meio de comportamento comumente esperado no uso do sistema, não seriam capazes de fazer.

Façamos um paralelo com dois termos de mais fácil compreensão:

Segurança

Confiabilidade

Tudo aquilo que é sabidamente seguro é automaticamente confiável, mas nem tudo que é confiável é necessariamente seguro. Usando isso como uma comparação útil, digamos que um programa confiável é aquele que está relativamente livre de bugs. Isso significa que o software raramente falha nas mãos do usuário, lidando inclusive com as condições excepcionais normalmente.

Se o programa foi escrito de forma a ser capaz de lidar com ambientes de execução incertos e entradas malformadas, podemos dizer que ele foi construído de maneira “defensiva”, e este deve ser um objetivo de todo programador. O programa deve ser capaz de se antecipar a algumas das falhas conhecidas e poder repelir um ataque direcionado por intrusos que estejam tentando manipular seu ambiente, introduzindo entradas para aproveitar alguma falhar e alcançar algum fim nefasto.

Confiabilidade e **segurança de software** não são a mesma coisa, mas compartilham objetivos semelhantes, pois ambos exigem estratégias de desenvolvimento que se concentram no extermínio de bugs de software. Veja o que diferencia esses dois tipos de software!

Software confiável

É aquele com baixa probabilidade de apresentar mau funcionamento ou falhas inesperadas. Isso inclui a garantia de que o software foi desenvolvido de acordo com padrões de qualidade, foi testado e validado de forma adequada, e pode ser confiável em termos de desempenho, disponibilidade e eficácia acerca dos objetivos para os quais foi projetado e construído.

Software seguro

É aquele que foi projetado e implementado com o objetivo de proteger o sistema de ataques e vulnerabilidades. Isso inclui a aplicação de técnicas de programação seguras, o uso de criptografia para proteger dados sensíveis, a implementação de controles de acesso para limitar o acesso a informações confidenciais, e outras medidas para reduzir os riscos de ataques e comprometimento do sistema.

Lembrando do caso do ataque de buffer overflow, o programador com mentalidade defensiva, para evitar esse tipo de ataque, deve usar técnicas de programação seguras, como verificar o tamanho dos dados armazenados em um buffer e garantir que ele esteja adequadamente dimensionado.

Explorando vulnerabilidades nos softwares

Confira, neste vídeo, a partir de exemplos, os conceitos teóricos relacionados com a exploração de vulnerabilidades em sistemas de computadores, redes e aplicativos.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A exploração de uma vulnerabilidade é realizada na prática por meio de um exploit, que é um software no qual o código tem a função de se aproveitar de uma falha existente dentro de um sistema, uma aplicação ou um serviço. Esse software, normalmente caracterizado por um pedaço de código, pode ser escrito em várias linguagens. Vejamos os tipos de linguagens utilizadas:

Python	C
C++	Assembly

O exploit carrega uma parte do código conhecida como payload. Em segurança de software, um payload (carga útil) refere-se a um código malicioso que é inserido em um sistema vulnerável para executar determinada ação. Essa ação pode ser de diversos tipos e depende do objetivo do invasor. Vamos conferir os tipos de ações que podem ser executadas!

- O roubo de informações.
- Os danos ao sistema.
- A instalação de outros softwares maliciosos.
- O controle remoto do sistema.

No contexto do uso de exploits, um payload geralmente é usado em conjunto com uma técnica de injeção de código para permitir que o atacante execute comandos no sistema comprometido. Isso pode ser feito, por exemplo, enviando um arquivo de exploit especialmente criado para o sistema alvo. Quando o exploit é executado com sucesso, o payload é então inserido no sistema e executado para realizar a ação maliciosa desejada.

Outro conceito relacionado com a prática da exploração de vulnerabilidades é o Shellcode, um conjunto de instruções e comandos criados para serem executados assim que o código é injetado. Deve ser encarado como um tipo específico de payload que é projetado para executar um shell (interface de linha de comando) no sistema comprometido. É geralmente escrito em linguagem Assembly e executado diretamente na memória do sistema comprometido.

Resumindo

O objetivo principal do shellcode é permitir que o invasor execute comandos no sistema comprometido, obtendo controle total sobre ele.

Relação entre vulnerabilidade, ameaça e risco

Confira, neste vídeo, os conceitos de vulnerabilidade, ameaça e risco na segurança de software, e entenda como a ameaça ocorre quando uma vulnerabilidade é explorada. A partir disso, você perceberá a importância do gerenciamento de risco na proteção contra ameaças.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Em termos teóricos, a avaliação de segurança de software é feita tomando por base uma análise de risco e o custo e retorno de cada uma das estratégias voltadas para lidar com cada risco. Então, precisamos apresentar os conceitos de ameaça e risco, que irão compor a avaliação de segurança.

Começando pela ameaça, é a situação que expõe o sistema a um evento ou a uma ação que pode explorar uma vulnerabilidade existente no software. As ameaças podem ser realizadas por pessoas mal-intencionadas, como hackers, ou por eventos não intencionais do próprio usuário comum. As ameaças podem explorar as vulnerabilidades para causar danos ao sistema, roubar dados ou informações, ou realizar outros tipos de ataques advindos de atividades maliciosas, bem como simplesmente expor dados ou travar um sistema, no caso de um evento não associado a más intenções. É importante reconhecer que a ameaça existe no encontro da vulnerabilidade com a exploração, ou seja, só existe realmente a ameaça quando existe uma vulnerabilidade que pode ser explorada. Observe o esquema:

Vulnerabilidade x Ameaça.

É possível que exista uma vulnerabilidade em um software e que ela não represente uma ameaça real. Isso pode ocorrer por vários motivos, como:

A vulnerabilidade requer acesso a recursos ou privilégios que não estão disponíveis para um invasor em potencial, ao menos pelos métodos mapeados na avaliação de segurança.

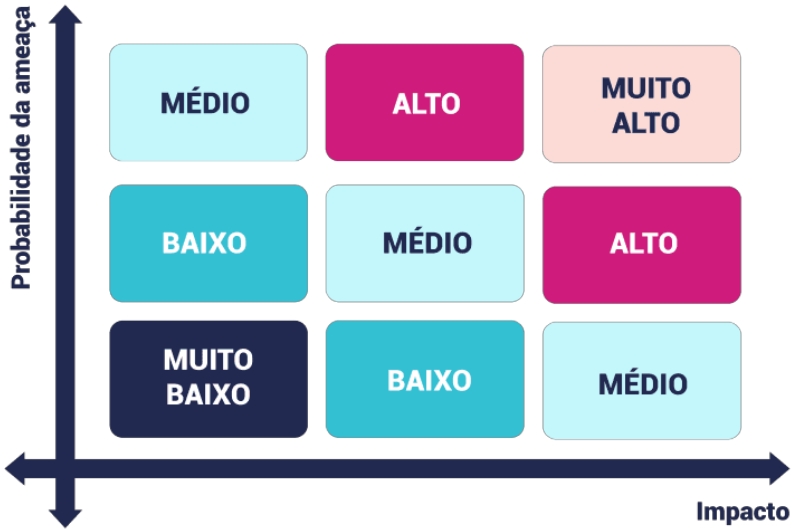
A vulnerabilidade é difícil ou impossível de explorar na prática, também na visão dos métodos mapeados, que deve considerar o estado da arte das técnicas de exploração.

Um exemplo prático seria uma vulnerabilidade em um software que requer acesso administrativo ao sistema para ser explorada. Nesse caso a ameaça é dependente da existência de ameaça do acesso administrativo. Se o sistema estiver configurado corretamente e o acesso administrativo for restrito a usuários confiáveis, a vulnerabilidade pode não representar uma ameaça real.

Resumindo

Não há ameaça real se as condições necessárias para sua exploração da vulnerabilidade não são atendidas. Portanto, é importante avaliar não apenas a existência de vulnerabilidades em um sistema, mas também as condições necessárias para sua exploração e o nível de risco associado a elas.

Enfim chegamos ao conceito de risco, que é a probabilidade de uma ameaça explorar uma vulnerabilidade para causar danos ao sistema. O risco deve ser calculado como um produto da probabilidade de uma ameaça ocorrer pelo tamanho do impacto que essa ameaça pode causar. Portanto, quanto maior a probabilidade de uma ameaça ocorrer e maior o impacto que essa ameaça pode causar, maior será o risco associado. Isso pode ser visualizado através de uma ferramenta conhecida como matriz de riscos. Confira:



Matriz de riscos.

Consolidando, vulnerabilidade é uma fraqueza no software, ameaça é a exploração dessa vulnerabilidade por uma ação ou evento e risco é a probabilidade de uma ameaça ocorrer e impactar o sistema. O gerenciamento de risco é importante na segurança de software para garantir a proteção do sistema contra ameaças conhecidas e desconhecidas.

Framework de avaliação

Confira, neste vídeo, um framework de ações voltadas para a avaliação de segurança de software.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A verificação de segurança de software envolve executar ações para avaliar e gerir riscos, sempre considerando as possíveis expectativas que os usuários têm, ou seja, respondendo o que seria considerado uma violação de segurança. Nem toda falha causa impactos na segurança, que é frequentemente descrita com base em três componentes, que podem ser descritos da seguinte forma:

Confidencialidade

É a capacidade de que a informação se mantenha privada apenas a quem detém direito de acessá-la.

Integridade

É a capacidade de que a informação seja confiável e esteja correta.

Disponibilidade

É capacidade de que a informação esteja disponível no momento oportuno.

Considerando os três pilares que suportam as expectativas relacionadas à segurança de um sistema, podemos definir a avaliação de risco de segurança conforme as ações de: **identificar**, **avaliar** e **implementar** os

principais controles de segurança no software, com enfoque também na prevenção de defeitos e vulnerabilidades.

Se a verificação de segurança executa tais ações corretivas e preventivas, perpassando todo o ciclo de vida do software, desde o design até o final do suporte, podemos organizar, de forma enxuta, a gestão de riscos em quatro etapas, com respectivas demandas de atividade.

Gestão de risco

Envolve um processo estruturado para identificar, avaliar e mitigar os riscos em uma organização. Vejamos algumas etapas comuns envolvidas no processo de gestão de riscos!

1

Identificação

Etapa em que se deve levantar todos os ativos críticos da infraestrutura de tecnologia e então elencar as informações confidenciais criadas, armazenadas ou transmitidas por esses ativos. Para cada ativo, deve ser criado um perfil de risco.

2

Avaliação

Etapa em que se deve abordar os riscos de segurança identificados para cada um dos ativos críticos, avaliando-os. A avaliação deve analisar as relações entre ativos, cruzando ameaças e vulnerabilidades, e estabelecendo controles atenuantes. Após a avaliação, é necessário determinar como alocar tempo e recursos de maneira eficaz e eficiente para tratar os riscos.

3

Tratamento

Etapa em que é preciso definir uma abordagem de correção ou mitigação, aplicando efetivamente os controles de segurança para cada risco encontrado nos respectivos ativos. Os riscos que estiverem dentro de níveis aceitáveis podem ser aceitos, o que significa simplesmente deixá-los de lado. Outra abordagem é delegar a responsabilidade de terceiros, por vezes contratando um serviço, ou se preparar para os impactos apenas fazendo um seguro, por exemplo.

4

Prevenção

Etapa em que se deve implementar ferramentas e processos para minimizar a ocorrência de vulnerabilidades e a consequente existência de ameaças aos ativos.

A gestão de risco permite que uma organização visualize o portfólio de aplicativos de forma holística pelo ponto de vista de um invasor.

Métodos de verificação de software

Com foco na verificação de segurança, vamos “explodir” a etapa de avaliação da gestão de risco, pois é justamente sobre ela que recai a parte mais “mão na massa”, na qual os bits e bytes são vistos de perto, ou como dizemos no jargão da TI, “escovados”, a partir de alguns métodos. Vamos conferi-los!

Auditoria de software

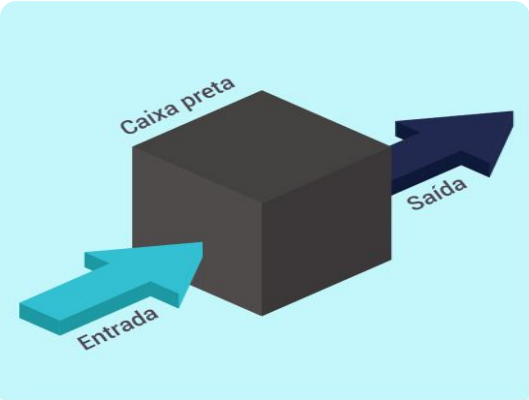
É o processo de análise de código (no formato fonte ou binário) para descobrir vulnerabilidades que possam ser exploradas por invasores. Com esse processo de análise sendo feita por **time amigo**, é possível identificar e fechar brechas de segurança, que, de outra forma, colocariam dados confidenciais e recursos de negócios em risco desnecessário.

Time amigo

Refere-se a um grupo de profissionais de segurança da informação, especialistas em auditoria de software, que são contratados pela própria empresa ou organização que desenvolveu o software a ser auditado. Esse time é considerado “amigo" porque trabalha em colaboração com a equipe de desenvolvimento do software, com o objetivo comum de identificar e corrigir quaisquer vulnerabilidades ou brechas de segurança presentes no código. Ao contrário de uma auditoria externa, em que um grupo de especialistas externos é contratado para testar a segurança do software, a auditoria realizada pelo time amigo permite maior transparência e cooperação entre equipes de segurança e desenvolvimento, facilitando a correção imediata das vulnerabilidades descobertas. Dessa forma, o time amigo é visto como um aliado no processo de garantir a segurança do software e proteger os dados confidenciais e recursos de negócios da empresa ou organização.

Teste de caixa preta

É um método de avaliação de um sistema de software manipulando apenas suas interfaces expostas. Normalmente, esse processo envolve a geração de entradas especialmente criadas que provavelmente farão com que o aplicativo execute algum comportamento inesperado, como falha ou exposição de dados confidenciais.



Teste da caixa preta.

O teste de caixa preta é tão somente injetar dados em uma aplicação e esperar que ocorra um comportamento inesperado. Não é possível saber exatamente o que o software está fazendo com os dados, portanto, podem existir centenas de caminhos de código que não serão explorados porque os dados injetados não acionam esses caminhos. Porém, por mais espantoso que possa parecer, muitas vulnerabilidades são encontradas por hackers testando o que acontece se lançar dados estranhos ao comportamento esperado como entrada. Felizmente, a auditoria de código pode ser combinada com o teste de caixa preta para maximizar a descoberta de vulnerabilidades em um período mínimo.

Verificando o aprendizado

Questão 1

Os bugs de software são muitas vezes introduzidos durante a fase de programação e podem possibilitar que um invasor tome o controle do sistema. Esse tipo de bug pode ser classificado como

A

ameaça.

B

risco.

C

vulnerabilidade.

D

exploit.

E

buffer overflow.

A alternativa C está correta.

Vulnerabilidade é uma fraqueza ou brecha no software que pode ser explorada por um atacante para comprometer a segurança do sistema. As vulnerabilidades podem ocorrer em diferentes níveis do software, como em código fonte, bibliotecas, protocolos de rede, configurações de sistema, entre outros.

Questão 2

Assinale a alternativa que apresenta o nome da parte maliciosa do código carregado juntamente com o restante, que é executado em um ataque bem-sucedido por ocasião da exploração de uma vulnerabilidade de software.

A

Payload

B

Exploit

C

Buffer overflow

D

Cross script connection

E

Overload

A alternativa A está correta.

Payload é o componente do exploit que é projetado para realizar uma ação maliciosa específica após a exploração bem-sucedida da vulnerabilidade. O payload pode ser usado para várias finalidades maliciosas, como instalação de malware, roubo de dados, controle remoto do sistema comprometido, entre outros. Ele pode ser projetado para ser executado automática ou manualmente pelo invasor.

Definição dos objetivos

Confira, neste vídeo, o conceito de modelagem de ameaças e a etapa de definição de objetivos. A partir disso, você verá a importância de identificar os objetivos do invasor, identificar possíveis ameaças e desenvolver contramedidas.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Modelar ameaças é uma forma de usar a identificação de vulnerabilidades para priorizar contramedidas. Vejamos uma modelagem de ameaças em cinco etapas:

Modelagem de ameaças.

Na primeira etapa, definir objetivos, devemos identificar os objetivos de segurança, definindo-os em estrito acordo com os requisitos. É uma etapa crítica no processo de modelagem de ameaças de segurança de software, pois envolve levantar os ativos que precisam de proteção, elencando as possíveis ameaças que podem prejudicá-los, ou seja, qualquer inobservância a algum ativo pode ter efeito dramático para os resultados.

A partir desse mapa, muito similar ao perfil de risco da primeira etapa da gestão de riscos e, tomando por base os requisitos de segurança estipulados, definem-se formalmente os objetivos de segurança a serem atingidos. De maneira mais específica, descrevemos os aspectos mais importantes dessa etapa. Vamos conferir!

Identificar as metas do sistema

Pode incluir a funcionalidade que o sistema deve fornecer, observando o que o sistema faz e para que e quem ele faz. É importante entender a finalidade do sistema e os objetivos de seus usuários a fim de identificar possíveis riscos de segurança.

Identificar ativos que precisam de proteção

Podem ser dados, hardware, software ou pessoal. Isso ajuda a entender o que precisa ser protegido de possíveis ameaças. É importante que nenhum ativo relevante fique de fora desse levantamento.

Identificar ameaças potenciais que podem prejudicar os ativos

Pode incluir ameaças externas (como hackers, malware e ataques de phishing) e ameaças internas (como erros de funcionários ou pessoas mal-intencionadas). Essa etapa deve considerar uma diversidade de perfis de especialistas em segurança, a fim de varrer a maior gama possível de ameaças conhecidas.

Entender a conformidade regulamentar

Pode ser necessário, dependendo do setor e do país onde o sistema será implantado, cumprir regulamentos ou padrões específicos. É importante identificar e entender esses regulamentos e padrões para garantir que o sistema seja projetado e implementado de acordo.

No geral, a primeira etapa da modelagem de ameaças é crucial para definir os objetivos de segurança do sistema e entender os riscos potenciais que podem afetar o sistema. Essa etapa ajuda a estabelecer uma base sólida para o restante do processo de modelagem de ameaças.

Diagramação dos fluxos de dados

Confira, neste vídeo, a etapa de diagramação dos fluxos de dados e entenda como essa técnica é utilizada para representar o movimento de informações dentro de um sistema ou processo.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A etapa de diagramação de fluxos visa criar um diagrama de fluxo de dados (DFD), ou seja, envolve a representação visual do sistema e de seus componentes, incluindo como os dados fluem entre eles. Vejamos alguns aspectos importantes!

Identificar os componentes do sistema (servidores, bancos de dados e interfaces de usuário)

Inclui o propósito de cada componente e como eles trabalham juntos para atingir os objetivos do sistema e construir uma visão clara sobre como o sistema funciona.

Definir fluxo de dados entre os componentes

Inclui identificar os diferentes tipos de dados, como entrada do usuário ou informações confidenciais, e como eles são transmitidos entre os componentes.

Identificar onde os dados são armazenados e como são protegidos

Inclui armazenamentos de dados internos, como bancos de dados, e externos, como armazenamento em nuvem.

Identificar limites de confiança

Inclui limites de confiança, que definem os limites entre diferentes componentes ou sistemas que possuem diferentes níveis de confiança. Por exemplo, um aplicativo da web pode ter um limite de confiança entre o navegador do cliente e o servidor da web.

Analisar o fluxo de dados para identificar possíveis falhas de segurança

Inclui a identificação de áreas onde os dados podem ser vulneráveis ao acesso, à modificação ou à exclusão não autorizados.

Ao criar um diagrama de fluxo de dados, o modelador de ameaças obtém melhor compreensão da arquitetura do sistema e de como os dados são transmitidos entre seus componentes.

Resumindo

O entendimento do diagrama de fluxo ajuda a identificar possíveis riscos e vulnerabilidades de segurança que podem ser tratados nas etapas subsequentes do processo de modelagem de ameaças. É desejável que sejam empregadas ferramentas de modelagem com bom resultado gráfico, gerando documentação de qualidade.

Identificação das ameaças

Confira, neste vídeo, a identificação de ameaças como processo essencial para garantir a segurança e proteção de um sistema ou uma organização, e conheça alguns exemplos de ameaças cibernéticas.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A etapa de identificação de ameaças envolve o brainstorming de todas as possíveis ameaças ao sistema e seus componentes, incluindo ameaças internas e externas. Os diagramas de fluxos de dados devem ser utilizados como insumos para o brainstorming, pois trazem visibilidade sobre os possíveis pontos de falha observados a partir do acompanhamento do fluxo dos dados pelo sistema. Vejamos alguns aspectos importantes!

Identificar agentes (quem ou o que) de ameaça ao sistema

Inclui ameaças externas, como hackers ou malware, bem como ameaças internas, como funcionários com intenções maliciosas.

Identificar vetores de ameaças

Inclui a identificação de possíveis pontos de entrada, como conexões de rede, interfaces de usuário ou componentes de terceiros para identificar como um agente de ameaça pode obter acesso ao sistema ou a seus componentes.

Identificar ações de ameaças que um agente pode executar depois de obter acesso ao sistema

Inclui ações como roubo de dados confidenciais, modificação de configurações do sistema ou lançamento de ataques de negação de serviço.

Categorizar ameaças

Inclui priorizar quais ameaças devem ser abordadas primeiro. Uma vez identificadas todas as possíveis ameaças, elas devem ser categorizadas com base em sua gravidade e probabilidade de ocorrência.

Documentar ameaças identificadas junto com seu impacto potencial no sistema

Inclui a documentação usada na próxima etapa do processo de modelagem de ameaças para desenvolver contramedidas para lidar com as ameaças identificadas.

Durante os trabalhos de identificação, pode ser interessante utilizar um checklist estruturado de tipos de ameaça, incluindo:

1. Falsificação
2. Adulteração
3. Repúdio
4. Divulgação de informações confidenciais
5. Negação de serviço
6. Escalada de privilégio

Ao identificar todas as possíveis ameaças ao sistema, o modelador de ameaças obtém melhor compreensão dos possíveis riscos e vulnerabilidades que precisam ser resolvidos. Esse entendimento ajuda a garantir que o sistema seja projetado e implementado com a segurança em mente, reduzindo a probabilidade de ataques bem-sucedidos.

Mitigação das ameaças

Confira, neste vídeo, a mitigação das ameaças e entenda a sua priorização com base no impacto potencial, na avaliação das contramedidas, implementação e documentação. Por fim, observe como reduzir os ataques e fortalecer a segurança geral do seu sistema.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A etapa de mitigação envolve o desenvolvimento de contramedidas para lidar com as ameaças e vulnerabilidades identificadas, de forma a corrigir os problemas de segurança do sistema. Acompanhar a gestão de bug, considerando as ameaças identificadas, ajuda a configurar o fluxo de trabalho dessa etapa.

Isso ocorre com ainda mais eficiência se for apoiado no checklist estruturado sugerido na identificação das ameaças, pois cria uma padronização da tarefa, garantindo que nada seja esquecido. Vejamos alguns aspectos importantes dessa etapa!

Priorizar ameaças identificadas

Inclui o foco primeiro nas ameaças mais críticas que são medidas por meio do seu impacto potencial e probabilidade de ocorrência. É comum a utilização de três variáveis de avaliação como critério de priorização, a saber: impacto, gravidade e risco.

Avaliar a eficácia de cada contramedida

Inclui testar a contramedida, garantindo que ela aborde a ameaça identificada em um ambiente controlado, além de simular diferentes cenários de ataque.

Implementar contramedidas

Inclui a avaliação da eficácia de cada uma das contramedidas, para que sejam implementadas no sistema. Isto é: integrar as contramedidas no projeto do sistema e garantir que não interfiram na funcionalidade do sistema, como os controles de segurança, controles de acesso, criptografia ou sistemas de detecção de intrusão.

As contramedidas podem cobrir um ou mais tipos controles de segurança, dentre eles: controles físicos (câmeras de segurança), técnicos (como criptografia) ou administrativos (como políticas). Além disso, podem ter uma ou mais funções, como: preventiva, detectiva, corretiva, recuperativa e impeditiva.

Documentar contramedidas implementadas

Ao registrar essas contramedidas, garantimos um registro dos controles de segurança implantados no sistema. Dessa forma, podemos manter e atualizar as contramedidas conforme necessário.

Ao desenvolver e implementar contramedidas para abordar as ameaças identificadas, o modelador de ameaças reduz a probabilidade de ataques bem-sucedidos e fortalece a segurança geral do sistema. A manutenção contínua e as atualizações das contramedidas garantem que o sistema permaneça seguro diante das ameaças em evolução. A documentação amplia a consciência situacional e pode contribuir também para observação de possíveis lacunas ocasionadas por falta de cobertura das contramedidas.

Validação do modelo de ameaças

Confira, neste vídeo, o processo de validação do modelo de ameaças que é essencial na área de segurança da informação e que avalia a eficácia e a precisão para identificar e mitigar riscos de segurança em um sistema.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A etapa de validação envolve revisar e testar o modelo de ameaça para garantir que ele reflita com precisão os requisitos de segurança do sistema e seja eficaz na identificação e no tratamento de riscos de segurança. Geralmente acontece antes da implantação do sistema para garantir que os requisitos de segurança sejam

cumpridos, as hipóteses sejam validadas e os controles de segurança sejam avaliados. Vejamos algumas ações importantes!

1

Revisar modelo de ameaça e garantir precisão dos requisitos de segurança

Inclui revisar o diagrama de fluxo de dados, as ameaças identificadas e as contramedidas desenvolvidas para lidar com essas ameaças.

2

Testar contramedidas desenvolvidas e garantir a eficácia no tratamento das ameaças

Inclui testar o sistema em um ambiente controlado e simular diferentes cenários de ataque.

3

Resolver problemas identificados durante revisão e teste, atualizando modelo de ameaça e contramedidas

Inclui a adição de novas contramedidas ou o ajuste das existentes para lidar melhor com as ameaças identificadas.

4

Comunicar às partes interessadas resultados da validação do modelo de ameaça

Inclui documentar o modelo de ameaça atualizado e as contramedidas e fornecer treinamento para administradores de sistema e outro pessoal relevante.

Ao validar o modelo de ameaça, o modelador de ameaças garante que o sistema seja projetado e implementado com a segurança em mente, reduzindo a probabilidade de ataques bem-sucedidos. A validação e os testes contínuos ajudam a garantir que o sistema permaneça seguro diante das ameaças em evolução.

Verificando o aprendizado

Questão 1

Qual é a finalidade da primeira etapa da modelagem de ameaças de segurança de software?

A

Identificar possíveis falhas de segurança do sistema.

B

Definir formalmente os objetivos de segurança a serem atingidos.

C

Identificar limites de confiança entre diferentes componentes ou sistemas.

D

Analisar o fluxo de dados para identificar possíveis falhas de segurança.

E

Criar uma representação visual do sistema e seus componentes.

A alternativa B está correta.

A primeira etapa da modelagem de ameaças de segurança de software tem como objetivo definir os objetivos de segurança a serem atingidos. É uma etapa crítica no processo de modelagem de ameaças de segurança de software, pois envolve levantar os ativos que precisam de proteção, elencando as possíveis ameaças que podem prejudicá-los. Identificar as metas do sistema, os ativos que precisam de proteção e as ameaças potenciais que podem prejudicá-los são ações necessárias para que sejam formalmente definidos os objetivos de segurança a serem atingidos. As demais alternativas referem-se às etapas posteriores do processo de modelagem de ameaças.

Questão 2

A validação do modelo de ameaças é a última etapa do processo de modelagem de ameaças de software. Assinale a alternativa que apresenta três ações empregadas durante essa etapa.

A

Avaliar contramedidas; resolver problemas; comunicar resultados.

B

Testar contramedidas; modelar problemas; comunicar resultados.

C

Testar contramedidas; resolver problemas; aguardar resultados.

D

Testar contramedidas; resolver problemas; comunicar resultados.

E

Avaliar contramedidas; modelar problemas; comunicar resultados.

A alternativa D está correta.

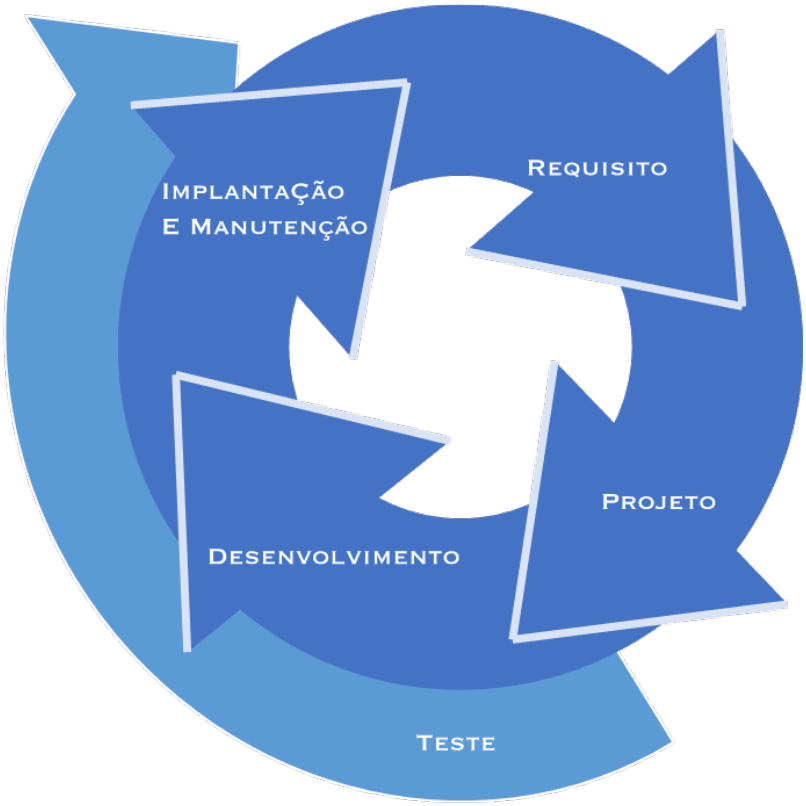
As ações empregadas na fase de validação do modelo de ameaças envolvem: testar contramedidas, visando garantir que elas sejam eficazes contra as ameaças levantadas; resolver problemas, que garante a validação do modelo de ameaças em si e pode gerar a necessidade de novas contramedidas, caso algum problema não seja resolvido; e comunicar os resultados, visando dar publicidade sobre o modelo para todas as partes interessadas.

Definindo os requisitos de segurança

Confira, neste vídeo, o ciclo de vida de desenvolvimento de software seguro e a etapa de definição dos requisitos de segurança.

> **Conteúdo interativo**
>
> Acesse a versão digital para assistir ao vídeo.

Chegou a hora de implementarmos o software, e é neste momento que entra o ciclo de vida de desenvolvimento seguro, encontrado na literatura pelas siglas SSDLC (*secure software development lifecycle*) ou SDL (*secure development lifecycle*). Confira as quatro etapas de teste:



Ciclo de vida de desenvolvimento seguro.

Na etapa de definição dos requisitos, que envolve capturar o escopo de funcionalidades do produto, são definidos objetivos e metas do projeto de software, traduzidos em requisitos funcionais e não funcionais que o software deve atender. É uma parte crítica do SDL porque define a base para todo o processo de desenvolvimento de software, sendo importante identificar os requisitos de segurança nesse estágio para garantir que a segurança seja integrada ao software desde o início. Isso pode ajudar a evitar que vulnerabilidades de segurança sejam introduzidas posteriormente no processo de desenvolvimento, quando se torna mais difícil e caro para corrigir.

Comentário

Os requisitos de segurança são geralmente identificados por meio de uma combinação de análise de risco, modelagem de ameaças e padrões e diretrizes de segurança. Os requisitos de segurança devem ser documentados e comunicados claramente a todas as partes interessadas, incluindo desenvolvedores, testadores e gerentes de projeto.

Vamos conhecer alguns exemplos de requisitos de segurança que podem ser identificados durante a fase de levantamento de requisitos. Vamos lá!

Requisitos de autenticação e controle de acesso

Especificam como os usuários são autenticados e autorizados a acessar o software e seus recursos.

Requisitos de proteção de dados

Especificam como os dados confidenciais serão protegidos, por exemplo, usando criptografia ou outras medidas de segurança.

Requisitos de registro e auditoria

Especificam como a atividade do sistema será registrada e auditada para detectar incidentes de segurança e rastrear a atividade do usuário.

Requisitos de conformidade

Especificam quaisquer requisitos regulamentares ou legais que o software deve atender, como a Lei Geral de Proteção de Dados (LGPD), ou outra norma associada ao negócio no qual o sistema irá operar.

O conceito-chave nesse passo é definir os aspectos de segurança do produto de maneira a incutir uma mentalidade de “segurança em primeiro lugar” e promover a conscientização do time de desenvolvedores. Ao identificar e documentar os requisitos de segurança, os desenvolvedores de software podem garantir que a segurança seja uma preocupação desde o princípio e que o software seja projetado e construído para atender aos padrões de segurança exigidos. Isso pode ajudar a reduzir o risco de violações de segurança, proteger dados confidenciais e manter a integridade e a disponibilidade do software.

Projetando software seguro

Confira, neste vídeo, a etapa projeto de software seguro, que envolve a identificação e mitigação de riscos de segurança desde o início do processo, garantindo a implementação de medidas adequadas para proteger o software contra ameaças.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A etapa de projeção de software seguro envolve desenhar tecnicamente os requisitos. A arquitetura do software é desenhada e as especificações técnicas são formalmente definidas empregando linguagens voltadas à modelagem.

Resumindo

O objetivo da fase de projeção de software é garantir que este seja projetado de forma a atender aos requisitos funcionais e não funcionais identificados na fase de definição dos requisitos, ao mesmo tempo em que integra considerações de segurança ao design.

As considerações de segurança são integradas à arquitetura geral do software e às especificações técnicas, o que inclui analisar possíveis riscos e vulnerabilidades de segurança identificados e projetar o software de forma a mitigar esses riscos. Vejamos algumas considerações de segurança que podem ser abordadas na fase de design. Confira!

1

Armazenamento e transmissão segura de dados

Projeta o software para armazenar e transmitir dados com segurança, como criptografar dados em trânsito e em repouso.

2

Controle de acesso

Projeta o software para controlar o acesso a recursos ou dados confidenciais, como por meio de controle de acesso baseado em função (*role-based access control* - RBAC) ou outros mecanismos de autenticação.

3

Validação de entrada

Projeta o software para validar a entrada de usuários e outros sistemas, de modo a evitar a exploração de vulnerabilidades de segurança, como injeção de SQL ou script entre sites (*cross-site scripiting* - XSS).

4

Protocolo de comunicação seguro

Projeta o software para usar protocolos de comunicação seguros, como *hypertext transfer protocol secure* (HTTPS), para proteger os dados transmitidos pela Internet.

5 Tratamento de erros

Projeta o software para lidar com erros e exceções de maneira segura, como registrar erros sem revelar informações confidenciais e tentar garantir o retorno ao estado anterior durante a detecção do erro. Essa abordagem é chamada de resiliência em software e visa tornar o sistema mais confiável.

Ao integrar considerações de segurança no projeto do software, os desenvolvedores podem criar um software menos vulnerável a ataques. Isso pode ajudar a evitar que vulnerabilidades de segurança sejam introduzidas posteriormente no processo de desenvolvimento e pode reduzir o risco de violações de segurança e perda de dados ou de disponibilidade. A fase de projeto prepara o terreno para a fase de desenvolvimento, na qual o software é realmente construído e codificado de acordo com as especificações técnicas definidas nessa etapa.

Desenvolvendo software seguro

Confira, neste vídeo, a etapa de software seguro, que cria aplicativos e sistemas desenvolvidos contra ataques e protege as informações dos usuários.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A etapa de desenvolvimento envolve escrever o código que implementa o desenho técnico e permite que o software seja desenvolvido com base nas especificações técnicas definidas na fase de projeto. O objetivo dessa fase é implementar o software de forma que atenda aos requisitos funcionais e não funcionais, e integre as considerações de segurança identificadas nas fases anteriores.

Desenvolvedor escrevendo codificação.

Durante a fase de desenvolvimento, os desenvolvedores escrevem o código, criam a documentação e executam tarefas de garantia de qualidade para assegurar que o software funcione conforme pretendido e atenda aos requisitos especificados. Práticas de segurança e técnicas de codificação seguras são seguidas durante essa fase para garantir que o software seja desenvolvido com a segurança em mente.

Alguns exemplos de práticas de codificação segura que podem ser seguidas durante a fase de desenvolvimento incluem todas as considerações abordadas na fase de projeto e a utilização de bibliotecas seguras, em vez de escrever o código do zero. O desenvolvedor pode utilizar bibliotecas seguras já existentes que tenham sido testadas e verificadas para evitar vulnerabilidades de segurança que abram brechas para ataques ao sistema.

Vamos relembrar as preocupações levantadas e desenhadas na fase de projeto, uma vez que devem ser levadas em consideração durante o desenvolvimento. Confira!

- Validação de entrada, inclusive com crítica não apenas ao formato, mas também levando em consideração a limitação adequada de tamanho, de forma a tentar evitar ataques de buffer overflow.

- Armazenamento e transmissão seguros de dados, considerando mecanismos criptográficos para proteção dos dados.

- Tratamento de erros e exceções de maneira segura que garantam a não exposição de informações críticas e confidenciais no caso de uma mensagem de erro.

- Controle de acesso com mecanismo de autenticação para dados confidenciais e recursos do sistema.

- Protocolos de comunicação seguros para proteger os dados transmitidos pela internet.

A fase de Desenvolvimento é uma parte crítica do SDL porque é onde o software é realmente construído e codificado. Ao seguir práticas e técnicas de codificação seguras durante esta fase, os desenvolvedores podem ajudar a fazer um software que atenda aos padrões de segurança, reduza o risco de tornando o software mais robusto em sua superfície de ataques, parte exposta a ameaças de hacker. Além de escrever código, os desenvolvedores também criam documentação e realizam tarefas de garantia de qualidade para assegurar que o software seja desenvolvido de acordo com as especificações técnicas e atenda aos requisitos identificados nas fases anteriores.

Conscientização e mentalidade de segurança é algo que os desenvolvedores de software seguro devem possuir. Portanto, oferecer treinamento e educação é algo desejável nas empresas de desenvolvimento. A varredura de código por um profissional diferente do que desenvolveu também contribui para a segurança do software. Existem ainda scanners que analisam código e podem detectar problemas durante a codificação.

Profissional executando varredura de código.

É importante criar um processo operacional repetível. Por exemplo, instruir os desenvolvedores a sempre executarem as ferramentas de varredura de segurança antes de enviar seu código, bem como realizar reciclagens periódicas de treinamento para melhores práticas de segurança.

Testando o software

Confira, neste vídeo, a etapa de teste de software seguro, que identifica e corrige vulnerabilidades de segurança. Veja também alguns exemplos de testes realizados para saber se o software é resiliente a ataques maliciosos.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

A etapa de teste de software envolve verificar se as funcionalidades se desenvolvam como esperado e dentro dos requisitos de segurança. O software é testado para garantir que funcione conforme o planejado, atenda aos requisitos funcionais e não funcionais e integre as considerações de segurança identificadas nas fases anteriores.

A fase de teste é essencial porque ajuda a identificar quaisquer problemas ou vulnerabilidades no software antes de sua implantação, realizando um apanhado geral sobre a qualidade de todas as atividades anteriores voltadas à segurança. Testes devem ser realizados durante todo o processo de desenvolvimento, desde testes unitários até testes de integração, finalizando com teste de sistema e teste de aceitação.

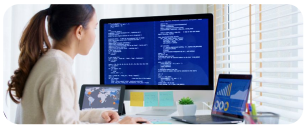
Confira alguns exemplos de testes que podem ser realizados:

Teste funcional

Testar o software para garantir que ele execute as funções a que se destina.

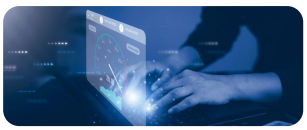
Teste de segurança

Testa o software em busca de vulnerabilidades e pontos fracos de segurança, como testes de penetração ou verificação de vulnerabilidades.



Teste de desempenho

Testa o software para garantir que ele funcione adequadamente sob cargas esperadas e cenários de uso, assegurando sua disponibilidade mesmo em condições de estresse dentro dos requisitos previstos.



Teste de usabilidade

Testa o software para garantir que seja fácil de usar e atenda às necessidades do usuário e que um comportamento inesperado do usuário não comprometa a confidencialidade, integridade ou disponibilidade do sistema.

Teste de compatibilidade

Testa o software para garantir que ele funcione conforme o esperado em diferentes plataformas, navegadores e dispositivos.

O teste deve ser um processo contínuo ao longo do ciclo de desenvolvimento, e os desenvolvedores devem usar uma variedade de ferramentas manuais e automatizadas, e técnicas de teste para garantir que o software seja utilizado de forma completa e precisa. Os resultados dos testes devem ser documentados e comunicados claramente a todas as partes interessadas, incluindo desenvolvedores, testadores e gerentes de projeto.

Comentário

Um SDL verdadeiramente ágil não segue o padrão de testes de um processo linear. Quando estamos falando em produção de software em modelos tradicionais, primeiro é preciso fazer tudo e só depois testar. Esse protocolo não pode ser seguido em um SDL ágil, porque novas versões são lançadas de forma rotineira.

A maioria das equipes hoje constrói algum tipo de **pipeline** com integração contínua e testes de software frequentes. Nesse modelo, há uma série de testes ocorrendo em vários estágios, alguns durante a

codificação, alguns após cada envio de código, alguns todas as noites e alguns testando o ambiente de produção ao vivo, por exemplo.

Pipeline

É uma abordagem de gerenciamento de projetos de software que envolve a automação de todo o processo de construção, testes e implantação de um aplicativo.

Dentro da etapa de requisitos, é importante mapear onde os testes de segurança devem ser executados, mas é possível introduzir testes adicionais antes da implantação, dependendo da estratégia de lançamento do sistema.

Implantando e fazendo manutenção do software

Confira, neste vídeo, a etapa de implantação e manutenção de software, que garante a confiabilidade, segurança e eficiência do software ao longo do tempo, proporcionando uma experiência positiva aos usuários.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Na etapa de implantação e manutenção, o software é implantado em ambiente de produção e mantido ao longo do tempo para garantir que continue atendendo aos requisitos funcionais e de segurança. Durante a fase de implantação, o software é liberado para produção, e a infraestrutura necessária é implementada para dar suporte ao software em um ambiente de produção. Isso pode envolver a configuração de servidores, configuração de bancos de dados e implementação de controles de segurança para proteger o software e os dados.

Depois que o software é implantado, manutenção e suporte contínuos são necessários para garantir que ele continue atendendo aos requisitos funcionais e de segurança. Isso pode envolver a execução de atualizações e patches regulares para solucionar vulnerabilidades de segurança ou correções de bugs, monitorar o software quanto a problemas de desempenho ou incidentes de segurança e fornecer suporte técnico aos usuários.

Desenvolvedor corrigindo bugs do software.

Confira alguns exemplos de atividades que podem ser realizadas durante a fase de implantação e manutenção.

Atualizações e patches de segurança

Envolve atualizar regularmente o software para solucionar quaisquer falhas ou pontos fracos que tenham sido descobertos posteriormente.

Monitoramento e resposta a incidentes

Envolve monitorar o software quanto a incidentes de segurança ou problemas de desempenho, e respondê-los de maneira oportuna e eficaz, inclusive interagindo com agências específicas da área para publicar os fatos ocorridos, ampliando a abrangência de conscientização sobre as falhas para outras organizações que utilizem o mesmo sistema por meio de alertas.

Suporte ao usuário

Envolve o fornecimento de suporte técnico aos usuários para garantir que eles possam usar o software de maneira eficaz e segura.

Ajuste de desempenho

Envolve a otimização do desempenho do software para garantir que ele funcione de maneira ideal sob as cargas esperadas e os cenários de uso.

Backups de dados e recuperação de desastres

Envolve a implementação de backups de dados e medidas de recuperação de desastres para garantir que os dados sejam protegidos e possam ser restaurados em caso de perda ou interrupção de dados.

Ao realizar manutenção e suporte contínuos durante a fase de implantação e manutenção, outros aspectos da segurança devem ser abordados, incluindo questões da infraestrutura que irá abrigar o sistema e os acessos a ela. Para os casos cada vez mais comuns de sistemas que rodam na nuvem, existe toda uma disciplina de segurança. Todas as atividades incluídas nessa etapa visam dar garantias mínimas de que o software continue atendendo às necessidades dos usuários.

Verificando o aprendizado

Questão 1

Durante a fase de desenvolvimento de software seguro, qual das seguintes práticas é importante para garantir a segurança do código?

A

Ignorar a validação de entrada.

B

Utilizar bibliotecas seguras.

C

Não aplicar controle de acesso.

D

Transmitir dados sem criptografia.

E

Não tratar erros e exceções.

A alternativa B está correta.

Utilizar bibliotecas seguras durante a fase de desenvolvimento de software é uma prática importante, pois permite que os desenvolvedores usem componentes já testados e verificados, evitando vulnerabilidades de segurança que possam ser exploradas em ataques.

Questão 2

SDL é a sigla em inglês para ciclo de vida de desenvolvimento de software seguro. Assinale a alternativa que melhor descreve a etapa de teste.

A

É realizada após o software ter sido desenvolvido.

B

É realizada nas integrações das partes do software.

C

É realizada nos múltiplos estágios da implementação.

D

É realizada apenas em ambiente de produção.

E

É realizada após a implantação.

A alternativa C está correta.

Os testes de software não ocorrem conforme um processo linear de atividades. Eles devem ser previstos ainda na fase de elicitação dos requisitos e realizados nos diversos estágios a partir do desenvolvimento, considerando testes unitários, de integração, e testes realizados após a implantação e a cada nova implementação de atualização do sistema.

OWASP

Confira, neste vídeo, a organização OWASP e o ciclo de vida de desenvolvimento de software seguro proposto por ela. Veja também como é realizada a definição dos requisitos de segurança, a criação de um modelo de ameaças e sua implementação e verificação antes do lançamento.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

The Open Worldwide Application Security Project (OWASP) é uma fundação sem fins lucrativos que, conforme sua missão, trabalha para melhorar a segurança do software. Promove projetos de software de código aberto, desenvolvendo lideranças locais nos diversos capítulos espalhados pelo mundo, e promove conferências educacionais e de treinamento, servindo de boa fonte para desenvolvedores e tecnólogos protegerem a web. A OWASP atua nos mais diversos temas relacionados à segurança das aplicações web.

O SDL proposto pela OWASP é uma estrutura para a construção de software seguro que consiste em uma série de etapas a serem seguidas durante o processo de desenvolvimento de software, desde o planejamento até o lançamento. Vamos conferir!

Definição dos requisitos de segurança

Nesta fase, os requisitos de segurança para o aplicativo são identificados e documentados, o que inclui a identificação de possíveis riscos e ameaças à segurança e a definição de objetivos de segurança que o aplicativo deve atender.

Criação de um modelo de ameaças

Nesta fase, o objetivo é identificar e documentar possíveis ameaças e vulnerabilidades à segurança. A modelagem de ameaças envolve a identificação de ameaças potenciais, como usuários mal-intencionados, conexões de rede inseguras ou vulnerabilidades no código.

Design seguro

Nesta fase, o desenho de projeto do aplicativo é revisado e os recursos de segurança são adicionados, o que inclui a implementação de controles de segurança, como controles de acesso, validação de entrada e criptografia.

Implementação

Nesta fase, o aplicativo é desenvolvido usando práticas de codificação seguras. Isso inclui seguir as diretrizes de codificação segura, que são orientações para que o software esteja apto a passar nos testes de segurança.

Verificação

Nesta fase, o aplicativo é testado quanto a vulnerabilidades de segurança usando várias técnicas de teste, como teste de penetração e varredura de vulnerabilidade.

Lançamento do software

Nesta fase, o aplicativo é liberado para produção. Isso inclui garantir que o aplicativo seja seguro e que os controles de segurança estejam em vigor para proteger o aplicativo e seus dados.

Pós-lançamento

Nesta fase, o aplicativo é monitorado quanto a vulnerabilidades e ameaças de segurança, o que inclui a realização de avaliações de segurança regulares, resposta a incidentes de segurança e atualização do aplicativo com correções (incluindo as correções segurança), conforme necessário.

SAFECode

Confira, neste vídeo, a organização SAFECode e o ciclo de vida de desenvolvimento de software seguro. Veja também as definições de segurança e projeto, prática de codificação, gestão de risco e suas falhas, testes, validação e planejamento.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Software Assurance Forum for Excellence in Code (SAFECode) é uma organização sem fins lucrativos, com atuação global, dedicada a aumentar a confiança em produtos e serviços de tecnologia da informação e comunicação por meio do avanço de métodos eficazes de garantia de software. A missão da SAFECode é promover as melhores práticas para desenvolver e fornecer software, hardware e serviços mais seguros e confiáveis.

A organização publicou um conjunto de diretrizes denominado “Práticas Fundamentais para o Desenvolvimento de Software Seguro”, no qual apresentam elementos essenciais de um programa de ciclo de vida de desenvolvimento seguro. O SDL proposto é composto de oito práticas fundamentais. Algumas serão descritas com mais detalhes, e outras apenas citadas. Vejamos!

Definição de controles de segurança da aplicação

O SAFECode usa o termo *application security controls* (ASC) para se referir aos requisitos de segurança. As entradas para o ASC devem incluir: princípios de design seguro; práticas de codificação seguras; requisitos legais e do setor com os quais o aplicativo precisa estar em conformidade (por exemplo, PCI – no setor de pagamentos, SCADA – na automação industrial ou LGPD – para proteção de dados pessoais); políticas e normas internas; incidentes e outros comentários; ameaças e riscos.

Projeto

O software deve incorporar recursos de segurança para cumprir, internamente, as práticas de segurança e, externamente, as leis ou os regulamentos. Além disso, o software deve resistir a ameaças conhecidas com base no ambiente operacional. A modelagem de ameaças, revisões de arquitetura e revisões de design podem ser usadas para identificar e corrigir falhas de projeto antes de partir para implementação do código-fonte.

Práticas de codificação segura

As vulnerabilidades não intencionais são introduzidas no código por erros do programador. Esses tipos de erros podem ser evitados e detectados usando padrões de codificação; selecionando das linguagens, estruturas e bibliotecas mais apropriadas, incluindo o uso de seus recursos de segurança associados; usando ferramentas de análise automatizadas; e revisando manualmente o código.

Gestão de risco de segurança inerente ao uso de componentes de terceiros

Os projetos de software são construídos usando componentes de código aberto e de proprietários terceiros. Cada um desses componentes pode ter vulnerabilidades já na adoção ou que apareçam depois. Uma organização deve manter um inventário desses componentes, usar ferramentas para verificar vulnerabilidades neles e ter um plano para responder quando novas vulnerabilidades forem descobertas.

Teste e validação

A organização deve aplicar testes automatizados estáticos e dinâmicos, além dos famosos testes de penetração realizado pelos hackers éticos. Nesse último, um especialista em exploração de vulnerabilidades é contratado e liberado legalmente para tentar encontrar e explorar vulnerabilidades do sistema, reportando cada uma delas.

Gestão de descobertas de falhas de segurança

As cinco primeiras práticas produzem artefatos que ajudam nas descobertas relacionadas à segurança do produto (ou falta dela). As descobertas devem ser rastreadas e ações devem ser tomadas para remediar as vulnerabilidades. Como alternativa, a equipe pode aceitar conscientemente o risco e, nesse caso, o risco deve ser rastreado, com uma classificação de gravidade; um plano de correção; um prazo de nova revisão.

Divulgação e resposta de vulnerabilidade

O fato de seguir um SDL formal não faz o produto se tornar perfeitamente seguro devido ao cenário de ameaças que muda constantemente. A organização deve desenvolver uma resposta de vulnerabilidade e processo de divulgação para ajudar a conduzir a resolução de vulnerabilidades descobertas externamente e para manter todas as partes interessadas informadas sobre o progresso.

Planejamento da implementação do desenvolvimento seguro

O SDL saudável e maduro deve incluir, além das sete práticas anteriores, uma integração dessas práticas no processo de negócios em toda a organização, incluindo gerenciamento de programas, gerenciamento de partes interessadas, planejamento de implantação, métricas e indicadores e um plano de melhoria contínua. A cultura, a expertise e o nível de habilidade da organização precisam ser considerados ao planejar a implantação de um ciclo de vida de software seguro.

Microsoft SDL

Confira, neste vídeo, o ciclo de vida de desenvolvimento de software seguro proposto pela Microsoft. Veremos também algumas definições importantes, como requisitos, métricas e relatórios. E finalmente, aprenda a executar a modelagem de ameaças e estabelecer requisitos de projeto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A Microsoft desenvolveu um SDL próprio, o Microsoft Security Development Lifecycle, para o processo de desenvolvimento de seus produtos e, desde 2006, vem disseminando esse conhecimento. Seu método contém diversas práticas que serão descritas resumidamente as mais importantes, e as outras apenas citaremos, tudo bem?

Fornecer treinamento

Toda a organização de desenvolvimento deve conhecer a perspectiva, os objetivos e as técnicas do invasor, não apenas as implicações comerciais de não criar produtos seguros, o que chega a ser intuitivo. Em muitos casos, profissionais como desenvolvedores, arquitetos de software, engenheiros de serviço, gerentes de programa, gerentes de produto e gerentes de projeto não tiveram no seu cabedal de educação formal aprendizado sobre segurança cibernética.

Desenvolvedora recebendo treinamento.

Ferramentas de segurança, linguagens de programação seguras e vetores de ataque estão em constante evolução, tornando o conhecimento obsoleto com grande velocidade. Portanto, o treinamento contínuo em segurança cibernética é essencial para organizações que desenvolvem software.

Definir requisitos de segurança

A empresa desenvolveu algumas técnicas para a elaboração sistemática de requisitos de segurança. Por exemplo, a engenharia de requisitos de qualidade de segurança (*security quality requirements engineering* - SQUARE), que é um processo de nove etapas que ajuda as organizações a criar segurança nos estágios iniciais do ciclo de vida da produção.

O framework *keep all objectives satisfied* (KAOS) para linguagem de especificação de requisitos baseado em objetivos tem como foco o conceito de antimodelos. Um antimodelo é construído abordando obstáculos maliciosos (chamados antiobjetivos) criados por invasores para ameaçar os objetivos de segurança de um sistema. Um obstáculo nega os objetivos existentes do sistema!

O *secure i** estende a estrutura de modelagem i* com modelagem e análise de compensações de segurança e alinha os requisitos de segurança com outros requisitos. É importante entender que os requisitos de segurança devem ser continuamente atualizados para refletir as mudanças na funcionalidade necessária, nos padrões e no cenário de ameaças.

Definir métricas e relatórios de conformidade

A equipe de gerenciamento deve entender e ser responsabilizada pelos níveis mínimos aceitáveis de segurança usando métricas de segurança. Um subconjunto dessas métricas pode ser definido como indicadores-chave de desempenho (KPIs) para relatórios gerenciais.

Executar modelagem de ameaças

A particularidade da Microsoft é empregar o que chamam de *spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege* (STRIDE) como check list estruturado.

Estabelecer requisitos de projeto

O design de recursos seguros envolve o cumprimento dos princípios de segurança atemporais, ou seja, conceitos praticamente imutáveis: simplicidade; controle de acesso baseado em permissão em vez de exclusão; segurança não baseada na obscuridade; princípio do privilégio mínimo; defesa em profundidade; entre outros princípios que norteiam os requisitos de projeto.

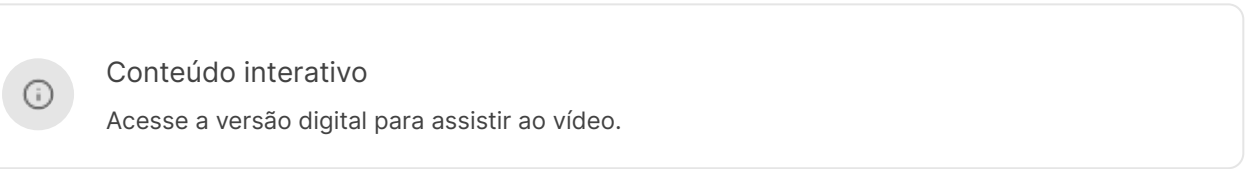
Outras práticas

As outras práticas que completam a estrutura de SDL da Microsoft são:

- Definição de uso de padrões de criptografia.
- Gerenciamento de riscos de segurança usando componentes de terceiros.
- Emprego apenas de ferramentas homologadas.
- Execução de teste estático de código, conhecido como static analysis security testing (SAST).
- Execução de teste dinâmico de código, conhecido como dynamic analysis security testing (DAST).
- Teste de penetração, o pen test.
- Estabelecimento de processo-padrão de resposta a incidentes.

NIST

Confira, neste vídeo, o famoso NIST e algumas publicações relevantes para nosso conteúdo, bem como alguns exemplos importantes.



O National Institute of Standards and Technology (NIST) oferece uma série de publicações que podem ser usadas como suporte formal aos esforços de desenvolvimento seguro. Vamos citar algumas, mas dentro de um universo gigantesco de informações disponíveis.

NIST CSF (Cybersecurity Framework)

O arcabouço de segurança cibernética fornece orientação às organizações para entender, gerenciar, reduzir e comunicar melhor os riscos de segurança cibernética. O framework propõe uma classificação de maturidade para as organizações subdividida em níveis.

Os níveis CSF funcionam como uma forma de fazer as organizações enxergarem e compreenderem o patamar que sua abordagem de segurança cibernética e processos em vigor para gerenciar esse risco está. Os níveis têm gradação crescente de rigor e sofisticação na descrição geral das práticas de gestão do risco de segurança cibernética.

NIST SP 800-39

O objetivo da Publicação Especial 800-39 é gerenciar riscos de segurança da informação nas operações organizacionais, ou seja, missão, funções, imagem e reputação, ativos organizacionais, indivíduos, dentre outros. Fornece uma abordagem estruturada, mas flexível, para gerenciar o risco de segurança da informação, considerando detalhes de avaliação, resposta e monitoramento contínuo do risco.

O texto põe foco na gestão de riscos técnicos de sistemas de TI com abordagem prescritiva e sugestiva. Inclui ameaças, vulnerabilidades, probabilidade e impacto, junto com monitoramento de controle e verificação de conformidade.

NIST SP 800-160

A publicação concentra-se na engenharia de resiliência cibernética, uma disciplina relativamente nova de engenharia de sistemas especializada em desenvolver sistemas seguros confiáveis e com capacidade de sobrevivência. Observada do ponto de vista da gestão de risco, a resiliência cibernética tenta reduzir o risco de dependência da empresa dos recursos cibernéticos.

A norma NIST SP 800-160 define a engenharia de resiliência cibernética e tem como objetivo construir e manter sistemas confiáveis com a habilidade de prever, resistir, recuperar-se e adaptar-se a situações desfavoráveis, estresses, ataques ou comprometimentos que envolvam ou sejam ativados por recursos cibernéticos.

NIST SP 800-53

É uma publicação bastante interessante em termos práticos para mitigação de risco, no âmbito da gestão de riscos e implementação de contramedidas no âmbito da avaliação de software seguro. Fornece um catálogo de controles de segurança e privacidade para sistemas de informação e organizações para proteger operações e ativos organizacionais e indivíduos de um conjunto diversificado de ameaças e riscos, incluindo ataques hostis, erros humanos, desastres naturais, falhas estruturais, entidades de inteligência estrangeiras e riscos de privacidade.

Os controles propostos são flexíveis e customizáveis e abordam diversos requisitos derivados de leis, regulamentos, políticas, padrões e diretrizes. O catálogo de controle aborda a segurança e a privacidade pela perspectiva da funcionalidade e de garantia. Abordar funcionalidade e garantia ajuda a garantir que os produtos de tecnologia da informação e os sistemas que dependem desses produtos sejam suficientemente confiáveis.

Outros exemplos

Vejamos outras publicações e ferramentas relacionadas que merecem ser citadas:

NIST SP 800-61

Guia para tratamento de incidentes de segurança de computadores.

NIST SP 800-122

Guia para proteção de confidencialidade de informações de identificação pessoal.

NIST SP 800-218

Mitigando o risco de vulnerabilidades de softwares pela adoção de um Framework de Desenvolvimento de Software Seguro.

O NIST também conta com diversas publicações relacionadas à criptografia, como técnicas de cifragem de blocos, algoritmos de função resumo (hash function) e os próprios algoritmos de criptografia em si, tanto de simétricos como assimétricos. O NIST já foi responsável por um concurso mundial para seleção do padrão de criptografia a ser adotado pelos EUA por ocasião da substituição do então algoritmo padrão, o *data encryption standard* (DES). Existem publicações que versam inclusive sobre criptografia pós-quântica, ou seja, o que acontecerá com a criptografia quando o computador quântico for efetivamente inventado e fabricado

em escala. O NIST preocupa-se em estar sempre na fronteira do conhecimento, sendo, portanto, importante fonte de informação de qualidade.

Normas ISO

Confira, neste vídeo, as normas ISO 27000, que estabelecem requisitos e diretrizes para diversas áreas de atuação, promovendo padronização e melhoria contínua nos processos, produtos e serviços das organizações.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

International Organization for Standardization (ISO) é uma entidade não governamental que desenvolve e publica padrões internacionais. A atuação do órgão é extremamente abrangente, passando por diversas áreas do conhecimento. Relacionado à gestão da segurança da informação, podemos relacionar toda a família de normas 27000. Com respeito ao sistemas de software, daremos um destaque especial para: ISO 15288, ISSO 16085, ISO 20004 e ISO 23643.

Família ISO 27000

É um conjunto de padrões internacionais que fornece diretrizes para sistemas de gerenciamento de segurança da informação (SGSI). Vejamos a família de normas ISO 27000!

1

ISO/IEC 27001 - Tecnologia da Informação – Técnicas de segurança – Sistemas de gestão de segurança da informação - Requisitos

Especifica os requisitos para um SGSI e fornece uma abordagem sistemática para gerenciar informações confidenciais da empresa para garantir sua confidencialidade, integridade e disponibilidade.

2

ISO/IEC 27002 - Tecnologia da Informação – Técnicas de segurança – Código de prática para controles de segurança da informação

Fornece um código de práticas para gerenciamento de segurança da informação e abrange uma ampla gama de controles de segurança, incluindo segurança física, segurança de rede, controles de acesso e gerenciamento de incidentes.

3

ISO/IEC 27003 - Tecnologia da informação — Técnicas de segurança — Sistemas de gestão da segurança da informação — Orientações

Fornece diretrizes para a implementação de um SGSI e inclui informações sobre as fases de planejamento, projeto e implementação.

4

ISO/IEC 27004 - Tecnologia da informação — Técnicas de segurança — Sistemas de gestão da segurança da informação — Monitoramento, medição, análise e avaliação

Fornece orientações sobre a medição e avaliação da eficácia de um SGSI, incluindo o uso de métricas e indicadores de desempenho.

5 ISO/IEC 27005 - Tecnologia da informação — Técnicas de segurança — Gestão de riscos de segurança da informação
Fornece diretrizes para gerenciamento de riscos relacionados à segurança da informação, incluindo avaliação, tratamento e comunicação de riscos.

6 ISO/IEC 27006 - Information technology – Security techniques – Requirements for bodies providing audit and certification of information security management systems
Apresenta diretrizes ao credenciamento de organizações que fornecem serviços de certificação para SGSI, garantindo que os organismos de certificação operem de forma eficaz e imparcial. No entanto, é uma norma ainda não traduzida/adaptada para o português.

A família de padrões ISO 27000 é essencial para organizações que lidam com informações confidenciais, incluindo dados financeiros, informações pessoais e propriedade intelectual. Ao implementar um SGSI com base nesses padrões, as organizações podem reduzir o risco de violações de segurança e garantir a confidencialidade, integridade e disponibilidade de suas informações.

Existe uma certificação ISO 27001 que é reconhecida globalmente e proporciona às organizações uma vantagem competitiva, aumentando sua credibilidade e reputação.

ISO/IEC 15288

É uma norma antiga (2009), mas que ainda não foi substituída e possui tradução/adaptação para o português. Descreve o ciclo de vida de sistemas a partir de processos que apoiam definição, controle e melhoria dos procedimentos associados ao ciclo de vida utilizados em uma organização ou em um projeto.

A ISO/IEC 15288 aborda o desenvolvimento de sistemas e configurações de hardware, software, dados, pessoas, processos, procedimentos, instalações, materiais e entidades envolvidas. Aplica-se a organizações que desempenham papéis tanto de comprador como de fornecedor, engloba todo o ciclo de vida de sistemas, incluindo concepção, desenvolvimento, produção, utilização, suporte e desativação de sistemas, e para a aquisição e fornecimento de sistemas, independentemente de ser realizado interna ou externamente à organização. Além disso, fornece um modelo de referência de processo definido por propósito e resultados esperados de processo, que decorrem da execução bem-sucedida das tarefas.

ISO/IEC 16085

É uma norma que fornece um padrão para profissionais responsáveis pelo gerenciamento de riscos associados a sistemas e software ao longo de todo seu ciclo de vida. Essa norma foi revisada recentemente para se alinhar com as atualizações de outros padrões relacionados, bem como para incluir novos conteúdos relacionados aos desafios de gerenciamento de riscos inerentes a grandes programas e projetos de engenharia de sistemas complexos.

ISO/IEC 20004

É uma norma que toma por base outras duas, realizando um refinamento de atividades definidas na ISO/IEC 18045. Fornece orientações mais específicas sobre a identificação, seleção e avaliação de vulnerabilidades para proporcionar uma avaliação (baseada na ISO/IEC 15408) de um sistema-alvo. A norma tomou como base a biblioteca ITIL (Information Technology Infrastructure Library), que é um conjunto de práticas e processos para gerenciamento de serviços de TI, o qual fez relativo sucesso no início dos anos 2000 com sua versão 2.

ISO/IEC 23643

É uma norma recente (2020), que especifica os requisitos para os fornecedores de software e fornece diretrizes para usuários e desenvolvedores de software seguro e ferramentas de verificação de segurança. Os usuários de tais ferramentas são desenvolvedores de software que precisam estar cientes das necessidades de segurança de software.

É interessante, pois apresenta casos de uso para segurança de software e ferramentas de verificação de segurança. Apresenta também categorias de ferramentas para segurança de software e fornece orientação e requisitos específicos de cada categoria para fornecedores e desenvolvedores de ferramentas.

A ISO é uma importante fonte de consulta para profissionais das mais diversas áreas, tendo algumas de suas normas obrigatoriedade de uso em um grande espectro de setores e indústrias.

Verificando o aprendizado

Questão 1

No contexto da segurança das aplicações de software, existem diversas entidades que se preocupam em fomentar práticas padronizadas para o ciclo de desenvolvimento de software seguro (SDL), seja para seu pessoal interno de desenvolvimento seja para desenvolver a área com estudos, eventos e publicações. Assinale a alternativa que apresenta três importantes órgãos que têm SDL publicado.

A

Microsoft - SAFECode - NIST

B

Microsoft - SAFECode - OWASP

C

NIST - SAFECode - OWASP

D

OWASP - SAFECode - ISO

E

Microsoft - OWASP - ISO

A alternativa B está correta.

O SDL envolve a identificação de possíveis ameaças e vulnerabilidades de segurança desde o início do processo de desenvolvimento de software e a adoção de medidas para mitigá-las. O processo normalmente inclui atividades como modelagem de ameaças, práticas seguras de codificação, análise de código, teste de penetração e teste de segurança e existem empresas desenvolvedoras que divulgam seu SDL e organismos preocupados em desenvolver tais práticas. Dentre eles, podemos citar: OWASP, SAFECode e Microsoft. O NIST e a ISO não definem SDLs. Eles possuem frameworks para segurança, mas sem definir um SDL.

Questão 2

As normas ISO são importantes fontes de consulta para várias áreas, incluindo negócios, tecnologia, qualidade, meio ambiente, saúde e segurança ocupacional. Essas normas fornecem diretrizes e padrões para ajudar empresas e organizações a melhorar a eficiência e a eficácia de suas operações. Na área de TI existe uma família de normas que trata de segurança da informação (SI), sendo fonte de consulta obrigatória para

profissionais que trabalhem com gestão de SI. Assinale a alternativa que corresponde ao número da família de normas.

A

9000

B

1500

C

1800

D

27000

E

33000

A alternativa D está correta.

A família de normas ISO 27000 aborda a gestão da segurança da informação em todos os aspectos, incluindo políticas de segurança, gestão de riscos, gestão de incidentes de segurança, gestão de continuidade de negócios, segurança física e lógica, entre outros.

Considerações finais

A verificação de software é um processo crucial que tem como objetivo garantir que o software atenda aos requisitos de segurança, confiabilidade, desempenho e qualidade. A respeito especificamente da segurança, a modelagem de ameaças é uma técnica utilizada para identificar e avaliar as possíveis ameaças e vulnerabilidades que podem comprometer o software.

O gerenciamento de risco é outra atividade importante no processo de verificação de segurança de software. Engloba a identificação de ativos, avaliação e tratamento dos riscos associados a medidas de prevenção.

O SDL é um processo de desenvolvimento de software que incorpora a segurança desde o início do ciclo de vida do desenvolvimento. A segurança incorporada tira proveito de ferramentas estruturadas, por exemplo, a modelagem de ameaças, composta das seguintes etapas: definição de objetivos; diagramação de fluxos de dados; identificação de ameaças; mitigação; e validação.

Em conjunto, a modelagem de ameaças, o gerenciamento de risco e o SDL formam uma abordagem abrangente para a verificação de software, ajudando o software a reduzir o número de falhas que se configurem como vulnerabilidades exploradas.

Explore +

Confira as indicações que separamos especialmente para você!

Pesquise:

- A base de dados de exploits e conheça os exploits já prontos orientados para determinadas plataformas e tipos de ataque, com o respectivo autor do código. Vale conferir!

- Em seus próprios portais eletrônicos, OWASP, SAFECode e Microsoft SDL. São excelentes caminhos para aprofundar o seu estudo!

Referências

KHAN, S. A.; KUMAR, R.; KHAN, R. **Software security**: concepts & practices. Boca Raton, FL: Chapman & Hall/ CRC Press, 2023.