1.Introdução à ataques web e a sua mitigação

1.1 Indicadores de códigos maliciosos

Identificar indicadores de códigos maliciosos é fundamental para garantir que os aplicativos sejam construídos com as proteções adequadas. O monitoramento contínuo protege a segurança do sistema ao longo do ciclo de vida. Isso envolve:

- **Análise de comportamento suspeito:** Monitorar o comportamento do software em busca de atividades anômalas, como acessos não autorizados ou operações suspeitas.
- **Assinaturas de malware:** Identificar padrões de código conhecidos de malware, como vírus, trojans e worms.
- **Análise de tráfego de rede:** Monitorar o tráfego de rede em busca de atividades suspeitas, como tentativas de exploração de vulnerabilidades ou comunicações não autenticadas.

1.2 Replay attacks

Os ataques de repetição, também conhecidos como ataques de replay, são uma classe de ataques cibernéticos em que um invasor intercepta e posteriormente retransmite dados de comunicação válidos, como mensagens, solicitações de autenticação ou transações, a fim de enganar um sistema ou obter acesso não autorizado.

Essa técnica explora a reutilização de informações legítimas para causar danos ou ganhar vantagem indevida. A implementação de medidas de segurança adequadas, como tokens únicos, carimbos de data e hora e autenticação multifator, é essencial para proteger sistemas e dados contra esses tipos de ataques. Os ataques de repetição geralmente ocorrem da seguinte maneira:

- **Interceptação:** O invasor intercepta uma comunicação legítima entre duas partes, como um cliente e um servidor. Isso pode ser feito por meio de escuta passiva em redes não criptografadas ou por meio da obtenção indevida de dados de sessão, tokens de autenticação ou outros identificadores.
- **Armazenamento:** O invasor armazena os dados interceptados, incluindo as mensagens originais, as solicitações de autenticação ou qualquer outra informação relevante.
- **Repetição:** O invasor retransmite os dados armazenados em momentos oportunos. Isso pode incluir reenviar uma solicitação de login, uma transação financeira ou outra ação, como se fosse o usuário legítimo.
- **Exploração ou acesso não autorizado:** O sistema alvo, incapaz de distinguir entre as transmissões originais e as repetidas, pode executar a ação solicitada pelo invasor, concedendo acesso não autorizado ou causando danos.

1.2.1 Mitigação de ataques de repetição

Para mitigar ataques de repetição, é importante implementar medidas de segurança adequadas:

- **Utilização de tokens únicos:** O uso de tokens únicos em cada solicitação ou transação pode evitar ataques de repetição. Os tokens são gerados pelo servidor e validados apenas uma vez. Depois de usados, eles não podem ser reutilizados.
- **Carimbo de data e hora:** Incluir um carimbo de data e hora em solicitações ou mensagens pode ajudar a detectar e rejeitar mensagens repetidas que chegam em um período de tempo inaceitável.
- **Controle de sessão:** Implementar um controle de sessão robusto, onde cada sessão é associada a um token de sessão único, pode impedir que um invasor reutilize sessões de autenticação ou dados de sessão roubados.
- **Criptografia:** Usar criptografia forte em comunicações pode dificultar a interceptação e a retransmissão de dados.
- **Autenticação multifator:** A autenticação multifator pode adicionar uma camada adicional de segurança, exigindo que o usuário forneça várias formas de autenticação, tornando mais difícil para um invasor replicar com sucesso um processo de autenticação completo.
- **Monitoramento de tráfego:** Realizar o monitoramento de tráfego para detectar padrões suspeitos de repetição ou atividades incomuns.
- **Expiração de sessão:** Configurar sessões para expirar após um determinado período de inatividade ou tempo, reduzindo o tempo em que um invasor pode reutilizar informações de sessão.

1.3 Session Hijacking

O sequestro de sessão, também conhecido como "session hijacking" ou "session fixation," é um tipo de ataque cibernético em que um invasor obtém controle não autorizado sobre a sessão de um usuário em um sistema ou aplicativo.

Uma sessão é um período de interação contínua entre um usuário e um sistema, como uma sessão de login em um site ou aplicativo. O objetivo do sequestro de sessão é assumir a identidade do usuário legítimo para realizar ações maliciosas em seu nome.

1.3.1 Técnicas de Session Hijacking

Existem várias técnicas que os invasores podem utilizar para realizar o sequestro de sessão:

- **Captura de cookies de sessão:** Os invasores podem capturar os cookies de sessão do usuário, que frequentemente contêm informações de autenticação, como tokens de sessão. Isso pode ser feito por meio de ataques de sniffing na rede, ataques de XSS (Cross-Site Scripting) ou malware no dispositivo do usuário.
- **Predição de ID de sessão:** Alguns sistemas geram IDs de sessão previsíveis, permitindo que invasores adivinhem ou gerem IDs de sessão válidos. Se o invasor souber ou puder prever o ID de sessão de um usuário, ele pode sequestrar a sessão.
- **Ataques MitM:** Em ataques MitM, um invasor intercepta as comunicações entre o usuário e o servidor, permitindo que ele capture informações de sessão, como senhas ou tokens de autenticação.

- **Ataques de Session Fixation:** Neste ataque, o invasor força um usuário a usar uma sessão de sessão predefinida, que o invasor já conhece e controla.

1.3.2 Proteção contra Session Hijacking

O sequestro de sessão é uma ameaça significativa à segurança cibernética que pode permitir que invasores assumam o controle de sessões de usuário. A implementação de medidas de segurança adequadas, como criptografia, autenticação multifator podem proteger contra esse tipo de ataque.

Além disso, o gerenciamento seguro de cookies de sessão é uma boa prática de prevenção do sequestro de sessão. Medidas que podem ser adotadas:

- **Criptografia:** Use criptografia para proteger as comunicações entre o usuário e o servidor, impedindo que terceiros capturem informações de sessão.
- **Autenticação multifator:** Implemente a autenticação multifator para adicionar uma camada extra de segurança à autenticação, tornando mais difícil para os invasores assumirem o controle de sessões.
- **IDs de sessão aleatórios:** Certifique-se de que os IDs de sessão sejam aleatórios e não previsíveis. Isso torna mais difícil para os invasores adivinharem ou gerarem IDs de sessão válidos.
- **Tempo de expiração de sessão:** Defina um tempo de expiração para sessões inativas, de modo que as sessões sejam encerradas automaticamente após um período de tempo específico.
- **Validação de origem:** Verifique a origem das solicitações para garantir que elas venham de fontes legítimas. Isso pode incluir verificação de referências (HTTP referer) e origens permitidas (CORS - Cross-Origin Resource Sharing).
- **Gerenciamento seguro de cookies de sessão:** Utilize cookies seguros (Secure flag), implemente uma política de Same-Site para controlar quais solicitações podem enviar cookies de sessão, revogue e emita novos tokens de sessão após eventos de autenticação ou quando um usuário alterar suas credenciais e, por fim, monitore e registre atividades de sessão para detectar comportamentos anômalos

1.4 Cross-Site Request Forgery (CSRF)

O Cross-Site Request Forgery (***CSRF***) é um tipo de ataque cibernético que explora a confiança que um site ou aplicativo tem em um usuário autenticado. Nesse tipo de ataque, um invasor engana um usuário legítimo para que execute ações não intencionais em um site ou aplicativo no qual o usuário já está autenticado.

O ataque ocorre quando o invasor convence o usuário a fazer uma solicitação HTTP maliciosa, sem o conhecimento ou consentimento do usuário, que é então processada pela aplicação web como uma ação legítima do usuário autenticado.

Um cenário hipotético

- Um usuário autentica-se em um aplicativo bancário online e obtém um cookie de sessão.
- O usuário visita um site malicioso enquanto ainda está autenticado no aplicativo bancário.
- O site malicioso incorpora um código HTML ou JavaScript que faz uma solicitação HTTP para transferir fundos da conta do usuário no aplicativo bancário para a conta do invasor, sem que o usuário perceba.
- Como o navegador do usuário está enviando automaticamente o cookie de sessão ao aplicativo bancário, a solicitação maliciosa é tratada como se fosse do usuário legítimo, e os fundos são transferidos sem o consentimento do usuário.

1.4.1 Prevenção e mitigação de ataques CSRF

Para prevenir e mitigar os ataques CSRF, é fundamental implementar medidas de segurança que garantam que as solicitações sejam legítimas e originárias de fontes confiáveis. As seguintes medidas que podem ser implementadas:

- **Token Anti-CSRF (CSRF token):** Uma das técnicas mais eficazes para mitigar ataques CSRF é o uso de tokens anti-CSRF. Um token CSRF é um valor único gerado pelo servidor e associado a uma sessão de usuário. Esse token é incluído em formulários ou em solicitações HTTP e é verificado pelo servidor para garantir que a solicitação seja legítima. Esses tokens são eficazes porque um atacante não pode prever ou obter o valor correto do token CSRF de um usuário legítimo, mesmo que consiga enganar o usuário para que ele faça uma solicitação maliciosa.
- **Origens de referência:** Os navegadores modernos suportam o cabeçalho HTTP Origin, que pode ser usado para verificar se uma solicitação é originada de um domínio legítimo. O servidor pode verificar o cabeçalho Origin para garantir que as solicitações venham de fontes confiáveis.
- **Verificação de origem:** Configurar cookies com a política Same-Site ajuda a impedir que cookies sejam enviados em solicitações CSRF. Isso limita a origem das solicitações que podem ser consideradas autênticas.
- **Requerer confirmação de ações críticas:** Para ações críticas, como transferências de fundos ou alterações de senha, os aplicativos podem exigir que os usuários confirmem a ação, mesmo que já estejam autenticados.
- **Tempo de expiração de sessão:** Implementar um tempo de expiração de sessão curto pode reduzir o risco de ataques CSRF, pois os tokens CSRF têm uma janela de oportunidade limitada para serem usados.

1.5 Clickjacking

O ***Clickjacking*** é um tipo de ataque cibernético em que um invasor engana um usuário para que clique em algo diferente do que ele percebe. Isso geralmente é feito ocultando ou mascarando elementos de uma página web real por trás de elementos de outra página. **Quando o usuário clica em algo que ele acredita ser inofensivo ou legítimo, na verdade, ele está clicando em algo malicioso ou executando uma ação indesejada sem o seu**

conhecimento ou consentimento. Os ataques de clickjacking são executados por meio de várias técnicas.

1.5.1 Etapas de um ataque de clickjacking

- O invasor cria uma página web maliciosa que contém um elemento oculto, como um botão ou um campo de entrada, que executa uma ação indesejada, como transferir fundos de uma conta.
- O invasor posiciona essa página maliciosa sobre uma página legítima que o usuário deseja interagir, como uma página de login de um banco.
- A página maliciosa é ajustada de forma que o elemento oculto coincida visualmente com um elemento legítimo da página subjacente, como um botão de "Login" ou "Transferência de Fundos."
- Quando o usuário clica no que parece ser o botão de login, na verdade, ele está clicando no elemento oculto da página maliciosa, que executa a ação indesejada, como transferir fundos, sem o conhecimento do usuário.

1.6 Prevenção de clickjacking

As medidas de segurança descritas a seguir podem prevenir ataques de clickjacking:

- **Uso de cabeçalhos HTTP como X-Frame-Options:** Os cabeçalhos HTTP, como o X-Frame-Options, podem ser configurados para controlar como as páginas web podem ser incorporadas em iframes. Configurar o cabeçalho X-Frame-Options com a opção "DENY" ou "SAMEORIGIN" ajuda a evitar que uma página seja incorporada em iframes não autorizados, reduzindo o risco de clickjacking.
- **Frame-Busting JavaScript:** O uso de JavaScript na página web pode ajudar a detectar se a página está sendo exibida em um iframe e, em seguida, redirecionar o navegador para a página original, interrompendo a tentativa de clickjacking.
- **Políticas de segurança de conteúdo:** As políticas de segurança de conteúdo (Content Security Policy - CSP) podem ser configuradas para controlar quais origens são permitidas para incorporar conteúdo na página. Isso ajuda a impedir a incorporação maliciosa de páginas em iframes.
- **Validação visual:** Os desenvolvedores podem adicionar validação visual à página para verificar se elementos ocultos não estão sendo sobrepostos ou mascarados por elementos maliciosos. Essa validação pode ser feita por meio de técnicas como verificação de visibilidade de elementos.

1.7 SSL Strip

O SSL Strip é um tipo de ataque cibernético que visa interceptar o tráfego criptografado entre um usuário e um servidor, descriptografá-lo e redirecioná-lo para um servidor malicioso. O nome "SSL Strip" se origina do fato de que o ataque geralmente é direcionado a conexões ***SSL/TLS*** (***Secure Sockets Layer/Transport Layer Security***), que são usadas para proteger as comunicações online.

O SSL Strip expõe as comunicações sensíveis dos usuários e para mitigar esse tipo de ataque, é fundamental implementar HTTPS rigoroso, usar políticas de segurança como HSTS e educar os usuários sobre como verificar a autenticidade dos certificados SSL/TLS. Essas medidas ajudam a proteger a privacidade e a segurança das comunicações online. O funcionamento do SSL Strip envolve as seguintes etapas:

- **Interceptação:** Um invasor posiciona-se entre o usuário e o servidor de destino, interceptando o tráfego entre eles. Isso pode ser realizado em redes Wi-Fi públicas não seguras, em redes comprometidas ou por meio de malware no dispositivo do usuário.
- **Redirecionamento:** O invasor redireciona as solicitações do usuário para um servidor malicioso, que atua como um intermediário entre o usuário e o servidor de destino legítimo.
- **Descriptografia:** O servidor malicioso descriptografa o tráfego que chega até ele, removendo a criptografia SSL/TLS. permite ao servidor malicioso acessar o conteúdo das comunicações, como senhas, informações de login ou dados confidenciais.
- **Encaminhamento:** Após a descriptografia, o servidor malicioso encaminha as solicitações para o servidor de destino real, mantendo o usuário inconsciente de que suas comunicações foram comprometidas. Em alguns casos, o servidor malicioso pode modificar o conteúdo das respostas do servidor de destino antes de enviá-las de volta ao usuário, possibilitando ataques de injeção de conteúdo.

1.7.1 Mitigação de ataques de remoção de SSL

A seguir estão algumas medidas de segurança que podem auxiliar na mitigação dos ataques de remoção de SSL.

- **Implementação de HTTPS estrito:** Os sites devem implementar HTTPS rigoroso, configurando seu servidor web para redirecionar automaticamente todas as solicitações HTTP para HTTPS. Isto garante que todas as páginas e recursos sejam carregados usando conexões seguras. Também deve implementar um certificado SSL/TLS confiável e configurá-lo corretamente no servidor. Isso evitará que os atacantes tenham a oportunidade de remover o SSL
- **HSTS (HTTP Strict Transport Security):** A implementação da política HSTS ajuda a garantir que os navegadores sempre se conectem a um site por meio de HTTPS, mesmo que o usuário digite "http://" na barra de endereços. Isso reduz a exposição a ataques de remoção de SSL.
- **Certificados de segurança e validação do usuário:** Os usuários devem ser educados sobre a importância de verificar a validade dos certificados SSL/TLS. Eles devem aprender a procurar sinais de segurança, como o ícone de cadeado e a conexão "https://" na barra de endereços. Além disso, devem ser incentivados a não ignorar avisos de certificado inválido em seus navegadores.
- **Rede segura:** Evite o uso de redes Wi-Fi públicas não seguras para acessar informações confidenciais. Use uma conexão VPN (Virtual Private Network) em redes públicas para proteger suas comunicações.
- **Segurança do dispositivo:** Mantenha dispositivos e navegadores atualizados com as últimas correções de segurança e use software antivírus e anti-malware confiáveis.

1.8 Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) é uma vulnerabilidade de segurança comum em aplicativos web que permite a um atacante injetar scripts maliciosos em páginas web visualizadas por outros usuários. Isso ocorre quando o aplicativo web não valida ou filtra adequadamente as entradas de dados fornecidas pelos usuários antes de exibi-las em uma página web.

Os ataques XSS exploram a confiança do navegador da vítima, que executa o código malicioso como se fosse parte legítima da página. Os ataques XSS também podem ser usados para roubar informações confidenciais, como cookies de sessão, ou para redirecionar os usuários para sites maliciosos.

1.8.1 Tipos de XSS

Há três tipos principais de XSS:

- **Refletido (Reflected XSS):** Neste tipo, o código malicioso é incorporado em um link ou em um campo de entrada, e a vítima é enganada para clicar no link ou acessar uma URL específica que contenha o código. O servidor web reflete o código de volta para a vítima, que o executa.
- **Armazenado (Stored XSS):** Neste caso, o código malicioso é armazenado no servidor, geralmente em um banco de dados, e é exibido para os usuários sempre que uma página específica é acessada. Comentários de fóruns ou campos de perfil em redes sociais são alvos comuns de ataques de XSS armazenados.
- **DOM-based (DOM-based XSS):** Esse tipo de XSS ocorre no lado do cliente, quando o código malicioso manipula o Document Object Model (DOM) da página web após o carregamento, sem necessariamente modificar o conteúdo no servidor. É mais difícil de detectar e mitigar, pois o código malicioso não viaja para o servidor.

1.8.2 Prevenção e mitigação de XSS

A prevenção e a mitigação de Cross-Site Scripting (XSS) são fundamentais para proteger aplicativos web contra essa vulnerabilidade comum. Isto significa que os esforços devem ser contínuos, pois as ameaças e as técnicas de ataque evoluem constantemente. Manter-se atualizado sobre as melhores práticas de segurança e monitorar seu aplicativo regularmente é essencial para proteger seus sistemas e os dados dos usuários contra ataques XSS de maneira eficaz.

1.9 SQL Injection

SQL Injection (***Injeção de SQL***) é uma das vulnerabilidades mais comuns e graves em aplicativos web que utilizam bancos de dados. Esse tipo de ataque ocorre quando um invasor insere instruções SQL maliciosas em campos de entrada, como formulários da web, para manipular consultas SQL executadas por um aplicativo.

O objetivo é explorar a falta de validação ou sanitização de dados, permitindo ao atacante acessar, modificar ou excluir dados do banco de dados, além de executar operações não autorizadas.

1.9.1 Como os ataques de SQL injection ocorrem

Os ataques de SQL Injection acontecem quando o aplicativo web incorpora diretamente dados não confiáveis em consultas SQL sem uma validação adequada. O invasor explora essa falta de segurança injetando instruções SQL maliciosas nos campos de entrada. O processo geralmente envolve os seguintes passos:

- **Identificação de vulnerabilidades:** O atacante procura campos de entrada, como caixas de pesquisa ou formulários de login, onde dados inseridos pelo usuário são diretamente incorporados em consultas SQL.
- **Inserção de instruções maliciosas:** O invasor insere instruções SQL maliciosas nos campos de entrada. Por exemplo, em vez de inserir um nome de usuário válido, o atacante pode inserir `' OR 1=1 –` em um campo de login.
- **Execução de instrução maliciosa:** O aplicativo web, sem uma validação adequada, incorpora a instrução SQL maliciosa na consulta e a executa no banco de dados. No exemplo acima, a consulta se tornaria `SELECT * FROM usuarios WHERE nome_usuario = '' OR 1=1 --'`.
- **Exploração:** Como o `1=1` sempre será verdadeiro, a consulta retornará todos os registros da tabela de usuários, permitindo ao invasor acessar informações de outros usuários ou até mesmo contornar a autenticação.

1.9.2 Exemplos de ataques de SQL injection

Um atacante insere `' OR 'a'='a` em um campo de login, enganando o aplicativo a pensar que um usuário válido está tentando fazer login, concedendo acesso não autorizado. Um invasor envia uma consulta SQL maliciosa que exclui todas as entradas de um banco de dados, resultando na perda de dados.

1.9.3 Prevenção e mitigação de SQL injection

A prevenção e a mitigação de SQL Injection são fundamentais para proteger aplicativos web contra esse tipo de ataque. Utilizar consultas parametrizadas, validar entradas de dados e adotar boas práticas de segurança são passos cruciais para garantir a integridade e a confidencialidade dos dados em um aplicativo.

- **Validação de entradas:** Valide e filtre rigorosamente todas as entradas de dados fornecidas pelos usuários. Certifique-se de que os dados sejam do tipo esperado e não contenham caracteres especiais não autorizados.
- **Consultas parametrizadas:** Consultas parametrizadas são uma das formas mais eficazes de prevenir SQL Injection. Use consultas parametrizadas ou consultas preparadas, dependendo da linguagem de programação e do sistema de gerenciamento de banco de dados (DBMS) que você está utilizando. Elas separam os valores dos parâmetros SQL, garantindo que os dados do usuário não sejam tratados como parte do comando SQL e assim, impede que os invasores injetem código malicioso nas consultas. A técnica correta é utilizar marcadores de posição (placeholders) na consulta SQL e inserir diretamente os valores de entrada no lugar dos placeholders. Essa abordagem separa os dados dos comandos SQL, prevenindo o SQL Injection.
- **Evite construção de consultas manuais:** Evite construir consultas SQL manualmente concatenando strings. Isso torna seu aplicativo vulnerável a injeções de SQL.
- **Princípio do menor privilégio:** Configure as permissões do banco de dados para que o aplicativo tenha apenas as permissões necessárias para realizar suas operações. Evite permissões de gravação quando somente a leitura é necessário.
- **Monitoramento e registros de auditoria:** Implemente monitoramento e registros de auditoria para detectar atividades incomuns ou tentativas de injeção de SQL em tempo real.

1.10 XML Injection

XML Injection é uma vulnerabilidade que ocorre quando dados não confiáveis são inseridos em documentos XML sem validação adequada, levando a uma interpretação incorreta ou a execução de código malicioso. Isso pode ocorrer em aplicativos que analisam e processam dados XML, como serviços da web, aplicações que lidam com configurações ou qualquer sistema que utilize XML para troca de informações.

Os ataques de XML Injection acontecem quando um invasor insere dados manipulados em um documento XML de forma que o analisador XML interpreta os dados de maneira insegura. Isso pode levar a resultados indesejados ou até mesmo à execução de código malicioso.

- **Identificação da vulnerabilidade:** O atacante identifica campos de entrada ou parâmetros que são usados em documentos XML sem validação adequada.
- **Inserção de dados maliciosos:** O invasor insere dados manipulados, como entidades XML maliciosas, em um campo de entrada ou parâmetro.
- **Interpretação incorreta:** Quando o aplicativo processa o documento XML, o analisador XML interpreta as entidades maliciosas de forma insegura, resultando em um comportamento inesperado.

1.11 Prevenção e mitigação de XML injection

A prevenção e a mitigação de XML Injection são fundamentais para garantir a segurança de aplicativos web que lidam com dados XML. A principal medida para prevenir XML Injection é validar e filtrar cuidadosamente todas as entradas de dados que podem ser incorporadas em documentos XML. Isso envolve a seguinte abordagem:

- **Validação de entrada:** Certifique-se de que todos os dados de entrada (por exemplo, dados fornecidos pelo usuário ou dados provenientes de fontes externas) sejam validados quanto à conformidade com o formato esperado. Isso pode incluir o uso de expressões regulares ou validação baseada em esquemas XML para garantir que os dados estejam no formato correto.
- **Filtragem de dados:** Realize uma filtragem rigorosa de todos os caracteres especiais e metacaracteres que podem ser usados para manipular o XML, como <, >, &, ', e ". Esses caracteres devem ser substituídos ou escapados adequadamente para que não possam ser interpretados como marcações XML maliciosas.
- **Evitar concatenação de dados em documentos XML:** Não permita a concatenação direta de dados de entrada em documentos XML. Em vez disso, utilize bibliotecas ou métodos seguros para construir documentos XML, garantindo que os dados de entrada sejam tratados como dados, não como parte das marcações XML.
- **Uso de bibliotecas seguras:** Utilize bibliotecas e frameworks seguros para criar e manipular documentos XML. Essas bibliotecas geralmente têm medidas de segurança embutidas que ajudam a prevenir ataques de XML Injection.
- **Limitar privilégios:** Se possível, limite os privilégios das partes que processam os documentos XML, garantindo que elas tenham apenas as permissões necessárias para executar suas tarefas. Isso ajuda a reduzir o impacto de um possível ataque.
- **Monitoramento e registros:** Implemente monitoramento e registro de atividades para detectar e responder a tentativas de XML Injection. Isso pode incluir o acompanhamento de solicitações suspeitas e a análise de logs para identificar atividades maliciosas.
- **Treinamento e conscientização:** Treine desenvolvedores e equipes responsáveis pela segurança para entender os riscos associados ao XML Injection e as melhores práticas para evitá-lo.
- **Atualizações regulares:** Mantenha todas as bibliotecas, frameworks e software relacionados atualizados para garantir que quaisquer vulnerabilidades conhecidas sejam corrigidas.

1.12 LDAP injection

LDAP Injection é uma vulnerabilidade que ocorre quando dados não confiáveis são inseridos em consultas ***LDAP*** (***Lightweight Directory Access Protocol***) sem a devida validação, permitindo que um invasor manipule as consultas para acessar, modificar ou excluir informações armazenadas em um diretório LDAP, como registros de usuários.

Os ataques de LDAP Injection ocorrem quando um aplicativo incorpora dados de entrada não confiáveis em consultas LDAP sem validação adequada. Os passos típicos de um ataque de LDAP Injection incluem:

- **Identificação das vulnerabilidades:** O atacante identifica campos de entrada, como caixas de pesquisa ou formulários, onde os dados de entrada são diretamente incorporados em consultas LDAP.
- **Inserção de dados maliciosos:** O invasor insere dados manipulados, como caracteres especiais, em um campo de entrada.
- **Manipulação da consulta LDAP:** O aplicativo web incorpora os dados de entrada diretamente na consulta LDAP, sem validação adequada, permitindo que o invasor manipule a consulta.
- **Execução de consulta maliciosa:** A consulta LDAP maliciosa é executada no diretório LDAP, resultando em acesso não autorizado a informações ou em modificações indevidas.

1.12.1 Prevenção e mitigação de LDAP injection

A prevenção e a mitigação de LDAP Injection são fundamentais para garantir a segurança de aplicativos web que fazem consultas em diretórios LDAP. Adotar boas práticas de segurança é um passo essencial para proteger sistemas contra essas vulnerabilidades.

A principal medida para prevenir LDAP Injection é utilizar consultas LDAP parametrizadas ou filtros de pesquisa LDAP seguros, uma vez que essas abordagens separam os dados de entrada das operações LDAP e evitam a interpretação maliciosa de caracteres especiais nos dados de entrada. Isso envolve a seguinte abordagem:

- **Consultas LDAP parametrizadas:** Ao construir consultas LDAP, evite a concatenação direta de dados de entrada nos filtros de pesquisa LDAP. Em vez disso, utilize parâmetros ou marcadores de posição nas consultas e insira os valores de entrada nos parâmetros. Isso permite que o servidor LDAP interprete os valores de entrada como dados, não como parte da consulta. O uso de consultas parametrizadas previne a interpretação maliciosa de caracteres especiais nos dados de entrada.
- **Filtros de pesquisa seguros:** Quando for necessário criar filtros de pesquisa LDAP manualmente, certifique-se de que os dados de entrada sejam filtrados e escapados adequadamente para evitar a injeção de caracteres especiais. Evite a concatenação direta de dados de entrada em filtros LDAP sem tratamento.
- **Validação de entrada:** Valide cuidadosamente todas as entradas de dados que podem ser usadas em consultas ou filtros LDAP quanto à conformidade com o formato esperado. Isso ajuda a garantir que apenas dados válidos e seguros sejam passados para as consultas LDAP.
- **Escape de caracteres especiais:** Se for necessário incluir dados de entrada diretamente em filtros LDAP, certifique-se de escapar adequadamente os caracteres especiais, como (,), *, \, e NULL, para que eles não sejam interpretados como metacaracteres LDAP.
- **Princípio do menor privilégio:** Garanta que as contas usadas para acessar o servidor LDAP tenham apenas as permissões necessárias para executar as operações de consulta. Isso ajuda a minimizar os riscos associados a um possível ataque de LDAP Injection.

- **Monitoramento e registro de log:** Implemente monitoramento e registro de atividades para detectar e responder a tentativas de LDAP Injection. Isso pode incluir o acompanhamento de solicitações suspeitas e a análise de logs para identificar atividades maliciosas.
- **Treinamento e conscientização:** Treine desenvolvedores e equipes responsáveis pela segurança para entender os riscos associados ao LDAP Injection e as melhores práticas para evitá-lo.
- **Atualizações regulares:** Mantenha o software relacionado ao LDAP, como servidores LDAP e bibliotecas de acesso LDAP, atualizado para garantir que quaisquer vulnerabilidades conhecidas sejam corrigidas.

1.13 Directory Traversal

Directory Traversal, também conhecido como Path Traversal, é uma vulnerabilidade de segurança que ocorre quando um invasor explora a falta de controle adequado sobre caminhos de arquivo ou diretório em um aplicativo web. Nesse tipo de ataque, o invasor tenta acessar arquivos e diretórios fora da área designada, potencialmente permitindo a visualização, leitura ou inclusão de informações sensíveis ou a execução de código malicioso.

Os ataques acontecem quando o aplicativo web não valida ou filtra adequadamente as entradas do usuário que contêm caminhos de arquivo ou diretório. Os passos típicos de um ataque de Directory Traversal incluem:

- **Identificação da vulnerabilidade:** O atacante identifica campos de entrada, como URLs ou parâmetros, onde caminhos de arquivo ou diretório são usados sem validação adequada.
- **Inserção de caminho manipulado:** O invasor insere um caminho de arquivo manipulado que contém sequências de caracteres que levam o aplicativo a acessar diretórios não autorizados.
- **Acesso aos arquivos/diretórios:** O aplicativo web incorpora o caminho de arquivo manipulado em uma operação de leitura ou inclusão de arquivo, permitindo que o invasor acesse informações fora da área designada.

1.13.1 Exemplos de ataques de Directory Transversal

Um invasor altera uma URL, substituindo o caminho original por ../etc/passwd para tentar ler o arquivo de senhas do sistema. Em um aplicativo de upload de arquivo, o atacante envia um arquivo com um nome que inclui ../ para tentar colocar o arquivo em um diretório sensível ou sobrescrever arquivos importantes.

1.13.2 Prevenção e mitigação de Directory Transversal

A prevenção e a mitigação de ataques de ***Directory Traversal*** são importantes aliados para garantir a segurança de aplicativos web. Validar e filtrar rigorosamente as entradas do usuário, utilizar caminhos relativos, configurar permissões adequadas no sistema de arquivos e implementar medidas de segurança são passos cruciais para evitar que invasores acessem ou modifiquem informações sensíveis ou executem código malicioso.

- **Validação estrita:** Valide e filtre rigorosamente todas as entradas de dados fornecidas pelos usuários que contêm caminhos de arquivo ou diretório. Certifique-se de que os caminhos sejam relativos e não permita caracteres especiais ou sequências que levem a travessias de diretório.
- **Utilização de caminhos relativos:** Sempre use caminhos relativos em vez de caminhos absolutos para acessar arquivos ou diretórios. Isso ajuda a evitar que os invasores acessem áreas não autorizadas do sistema de arquivos.
- **Restrições de acesso:** Configure permissões de sistema de arquivos adequadas para restringir o acesso a arquivos e diretórios apenas às operações necessárias.
- **Sanitização de caminhos:** Implemente uma função de sanitização que remova qualquer sequência ../ ou caracteres de escape de caminho antes de processar os caminhos.
- **Listas de permissões (whitelists):** Use listas de permissões que definam quais caminhos são permitidos e restrinjam todas as outras entradas.
- **Limitações de caracteres especiais:** Evite permitir caracteres especiais ou codifique-os de maneira adequada para que não sejam interpretados como parte de um caminho de diretório.

1.13.3 Exemplo prático de prevenção

Suponha que você tenha um aplicativo que permite aos usuários acessar arquivos de um diretório específico. Em vez de permitir que os usuários insiram caminhos diretamente, você pode criar uma função de validação que verifica se o caminho fornecido está dentro do diretório permitido e, em seguida, permite o acesso somente se a validação for bem-sucedida:

```
def validar_caminho(caminho):
    diretorio_permitido = '/caminho/permitido/'
    caminho_completo = os.path.abspath(os.path.join(diretorio_permitido, caminho))
    if caminho_completo.startswith(diretorio_permitido):
        return caminho_completo
    else:
        raise Exception("Acesso não autorizado ao diretório.")
```

Directory Traversal - Código.

1.14 Command Injection Attacks

Command Injection é um ataque que ocorre quando um invasor insere comandos maliciosos em campos de entrada ou parâmetros de um aplicativo web e, em seguida, faz com que o aplicativo execute esses comandos como parte de uma operação legítima.

Geralmente, os comandos são executados no sistema operacional em que o aplicativo web está hospedado. Os ataques de Command Injection podem permitir que os invasores executem comandos arbitrários, obtenham acesso não autorizado ao sistema e realizem ações maliciosas.

Os ataques de Command Injection acontecem se um aplicativo web incorpora dados de entrada não confiáveis diretamente em comandos do sistema operacional, sem validação ou sanitização adequada. Os seguintes passos descrevem um cenário típico de ataque:

- **Identificação da vulnerabilidade:** Em geral, o ataque de Command Injection é causado por uma vulnerabilidade relacionada à falta de validação e sanitização inadequada de dados de entrada, que são incorporados diretamente em comandos do sistema operacional. O atacante identifica campos de entrada ou parâmetros que são usados em comandos do sistema. Essa vulnerabilidade permite que um atacante injete comandos maliciosos que são executados pelo sistema como se fossem comandos legítimos. Ocorre na maioria das vezes em aplicativos que aceitam entrada de dados do usuário, como formulários da web ou campos de entrada, e passam esses dados diretamente para a execução de comandos do sistema operacional sem a devida validação e tratamento seguro.
- **Inserção de comandos maliciosos:** O invasor insere comandos maliciosos como parte dos dados de entrada. Por exemplo, em um campo de pesquisa, o atacante pode inserir ; ls para listar os arquivos do sistema.
- **Execução do comando malicioso:** O aplicativo web incorpora os comandos maliciosos na operação legítima e os executa no sistema operacional.
- **Exploração:** Os comandos maliciosos são executados, permitindo que o invasor acesse informações do sistema, modifique arquivos ou realize outras ações indesejadas.

1.14.1 Exemplos de Command Injection Attack

Um atacante insere ; rm -rf / em um campo de entrada de um aplicativo web que permite o upload de arquivos. Isso resulta na exclusão de todos os arquivos no sistema.

Em um sistema de gerenciamento de dispositivos, um invasor insere um comando malicioso que permite a reinicialização ou desativação de dispositivos críticos.

1.14.2 Prevenção e mitigação de Command Injection

A prevenção e a mitigação de ataques de Command Injection são essenciais para proteger aplicativos web contra a execução não autorizada de comandos maliciosos no sistema operacional. Validar entradas de dados, utilizar funções seguras para operações do sistema, sanitizar dados e restringir os privilégios do aplicativo são medidas importantes para evitar essa vulnerabilidade.

1.14.3 Exemplos práticos de prevenção de Command Injection

Em Python, é recomendável usar a biblioteca ‘subprocess’ para executar comandos do sistema de forma segura. Ela aceita uma lista de argumentos em vez de uma string única que contém todo o comando. Isso ajuda a evitar a injeção de comandos. import subprocess comando = ["ls", "-l", "/diretorio"] resultado = subprocess.run(comando, capture_output=True, text=True) print(resultado.stdout)

1.15 Server-Side Request Forgery (SSRF)

Server-Side Request Forgery (***SSRF***), em português "Forjamento de Requisições do Lado do Servidor", é uma vulnerabilidade de segurança que permite que um invasor faça com que um servidor execute requisições não autorizadas em outros recursos ou sistemas internos. Essa vulnerabilidade ocorre quando um aplicativo web permite que um usuário forje requisições a partir do servidor para outros servidores ou serviços, potencialmente expondo informações confidenciais ou explorando sistemas internos.

Os ataques de SSRF acontecem quando um aplicativo web não valida ou controla adequadamente as entradas do usuário que contêm URLs ou endereços IP usados em requisições feitas pelo servidor. Os seguintes passos descrevem um cenário típico de ataque de SSRF:

- **Identificação da vulnerabilidade:** O atacante identifica campos de entrada, como URLs, que são usados em requisições feitas pelo servidor.
- **Inserção de URL manipulada:** O invasor insere uma URL manipulada em um campo de entrada. Essa URL pode apontar para um servidor externo controlado pelo invasor ou para recursos internos não autorizados.
- **Execução da requisição:** O aplicativo web incorpora a URL manipulada em uma requisição feita pelo servidor e a envia para o servidor de destino.
- **Exploração:** O atacante pode explorar os resultados da requisição para obter informações sensíveis, explorar recursos internos, ou até mesmo usar a vulnerabilidade para ataques adicionais.

1.15.1 Prevenção e mitigação de SSRF

- **Validação estrita de entradas:** Valide e filtre rigorosamente todas as entradas de dados fornecidas pelos usuários que contêm URLs ou endereços IP, garantindo que elas sejam seguras e não permitam que URLs não autorizadas sejam acessadas.
- **Uso de listas de permissões (whitelists):** Mantenha uma lista de recursos ou URLs permitidos e restrinja todas as outras entradas a esses recursos autorizados.
- **Limitação de privilégios:** Configure o aplicativo web para executar com privilégios mínimos, limitando sua capacidade de acessar recursos internos ou servidores externos não confiáveis.
- **Restrição de saída de rede:** Bloqueie ou restrinja a capacidade do aplicativo web de fazer requisições para recursos externos ou servidores não autorizados. Esta é uma prática importante para mitigar vulnerabilidades de SSRF. Isso pode ser alcançado configurando regras de firewall, bloqueando portas não utilizadas ou implementando medidas de segurança de rede, como VLANs ou proxies reversos.
- **Utilização de URLs relativas:** Sempre que possível, utilize URLs relativas em vez de URLs absolutas para garantir que as requisições sejam direcionadas apenas a recursos dentro do escopo apropriado.
- **Monitoramento de logs:** Implemente registros de auditoria para monitorar atividades incomuns ou tentativas de SSRF em tempo real.

1.16 Ataques Man-in-the-Middle web

O ataque Man-in-the-Browser é uma técnica usada por cibercriminosos para interceptar e manipular comunicações entre um usuário e um aplicativo web. Para se proteger contra esse tipo de ataque:

- **Criptografia:** Utilize comunicações seguras por meio de criptografia para proteger os dados em trânsito.
- **Autenticação multifator:** Exija autenticação adicional, como senhas de uso único ou reconhecimento biométrico, para garantir a identidade do usuário.
- **Monitoramento de atividade suspeita:** Detecte comportamentos incomuns durante a interação do usuário com o aplicativo, como tentativas de acesso a áreas restritas ou mudanças não autorizadas nas configurações.

1.16.2 Resiliência em sites de operações

A resiliência de um site de operações refere-se à capacidade desse site de manter a continuidade operacional, mesmo diante de eventos adversos, falhas ou desastres. Existem diferentes abordagens para criar resiliência em um site de operações, classificadas geralmente como: ***Hot***, ***Warm*** e ***Cold***. Cada uma dessas categorias define o nível de preparação e prontidão para a restauração de serviços após uma interrupção. As redes empresariais geralmente fornecem resiliência no nível local. Um local alternativo de processamento ou recuperação é um local que pode fornecer o mesmo nível de serviço (ou similar).

Um site de processamento alternativo pode estar sempre disponível e em uso, enquanto um site de recuperação pode levar mais tempo para ser configurado ou ser usado apenas em caso de emergência. A escolha entre hot, warm ou cold depende dos requisitos de negócios, do orçamento disponível e da tolerância ao tempo de inatividade. Cada abordagem oferece um equilíbrio diferente entre custo e tempo de recuperação.

As operações são projetadas para fazer failover no novo site até que o site anterior possa ser colocado online novamente. Failover é uma técnica que garante que um componente, dispositivo, aplicativo ou site redundante possa assumir de forma rápida e eficiente a funcionalidade de um ativo que falhou.

Existem diferentes abordagens e suas características para criação de resiliência de site, elas são:

- **Site de operações hot:** A infraestrutura, todos os sistemas e dados necessários estão ativos e em funcionamento. Um hot site geralmente envolve a duplicação exata de todos os sistemas, aplicativos e dados do site principal para o site de contingência. Isso é alcançado por meio de tecnologias de replicação em tempo real. Em caso de falha no site principal, a transição para o hot site é praticamente instantânea, minimizando o tempo de inatividade. Suas características incluem: hardware e software totalmente configurados; dados sincronizados em tempo real entre o site principal e o site de contingência; rápido tempo de recuperação, praticamente sem tempo de inatividade percebido.

- **Site de operações warm:** Normalmente, parte da infraestrutura e dos dados já está configurada, mas algumas atividades manuais podem ser necessárias para ativar completamente o site. Um warm site pode envolver a configuração de parte da infraestrutura antes do incidente, mas pode exigir intervenção manual para estar completamente operacional. Dados podem ser sincronizados regularmente, mas atrasos podem ocorrer, dependendo da frequência das atualizações. Entre suas características estão: parte da infraestrutura está pré-configurada; dados podem não estar totalmente sincronizados, mas são atualizados regularmente; tempo de recuperação mais rápido do que um site "cold", porém mais lento do que um "*hot*".
- **Site de operações cold:** Não há infraestrutura operacional nem dados em tempo real. Um cold site requer configuração manual extensiva. Isso pode envolver a instalação de hardware, software e a restauração de dados. Geralmente, é mais econômico do que as opções hot e warm, mas o tempo de recuperação é significativamente mais longo. Suas características incluem: nenhum hardware ou software configurado; dados podem estar desatualizados ou não disponíveis; maior tempo de recuperação, pois requer configuração manual e restauração de dados.