Requisitos de segurança

Fundamentação de requisitos e construções de segurança em aplicações

Requisitos de segurança

Assumindo que o aluno pode não ter muita prática com a identificação de requisitos, soluções, arquiteturas clássicas de mercado para contornar a maioria dos problemas de segurança até hoje identificados, iremos detalhar aqui algumas das principais recomendações para compor a “caixa de ferramentas” de soluções do aluno.

Autenticação

Ao projetar a autenticação, não importa se você possui uma autenticação multifator em um canal se um invasor puder redefinir um acesso através um callcenter e respondendo a perguntas comumente conhecidas. Todos os métodos de autenticação devem ser fortes.

1. Uso de contas de sistemas operacionais dedicadas e de baixo privilégio para todos os componentes, serviços e servidores de aplicativos.
2. As comunicações entre os componentes do aplicativo, incluindo APIs, middleware e camadas de dados, devem ser autenticadas. Componentes devem ter o mínimo de privilégios necessários.
3. Uso de um único mecanismo de autenticação seguro, que pode ser estendido para incluir autenticação forte e possui registro e monitoramento suficientes para detectar violações de contas (abusos ou quebras de controle de acesso).
4. Todos os caminhos de autenticação e APIs de gerenciamento de identidades devem implementar o mesmo nível de força de autenticação.

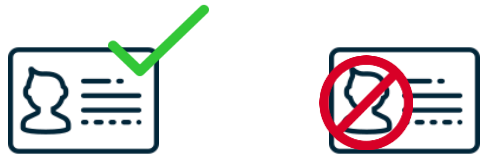
Autenticação

Garanta a existência de pontos de verificação de controle de acesso em funções dos sistemas. Nunca assuma que o controle de acesso implementado no lado “cliente” será respeitado.

Garanta a imposição do princípio do menor privilégio em funções, arquivos de dados, URLs, controladores, serviços e outros recursos.

Garanta que o aplicativo use um mecanismo de controle de acesso único e seguro para acessar dados e recursos protegidos. Todas as solicitações devem passar por esse mecanismo.

Garanta que o controle de acesso verifique a autorização do usuário para um item de recurso/dado ao invés de checar apenas o acesso funcional (usuário pode “ver holerite”, mas não qualquer holerite, apenas “o dele”).



Entrada e saída de dados

Nas principais arquitetura modernas, o ponto de garantia de confiança mudou. O termo "camada de serviço confiável" significa qualquer ponto de aplicação confiável, independentemente do local (antiga visão de rede externa versus rede interna), como um micro serviço, uma API sem servidor, uma API confiável em um dispositivo cliente que tenha inicialização segura , uma API de terceiros etc.

- Garanta que as entrada de dados definam claramente quais critérios devem ser atendidos para que o dado seja considerado integro e confiável.
- Valide as entradas de dados em uma camada de serviço confiável (antes disso o dado não deve ser tratado como confiável ou passível de processamento).
- Não use objetos serializados ao comunicar com clientes. Mas se for preciso, verifique a integridade dos objetos após a desserialização antes de confiar no conteúdo deles.
- Garanta que a codificação de saída seja específica para o interpretador ao qual a saída se destina (HTML encoding para dados enviados para o Browser, por exemplo).

Cifragem de dados

Os aplicativos precisam ser projetados considerando a cifragem robusta dos dados de acordo com sua classificação (protegendo o que é necessário). Alguns processos do ciclo de vida das chaves não podem ser negligenciados.

O gerenciamento de chaves criptográficas - se um ciclo de vida de chave criptográfica segue um padrão de gerenciamento de chaves, como o NIST SP 800-57 - todas as chaves e senhas são "substituíveis" e fazem parte de um processo bem definido evitando a perda de acesso aos dados cifrados.

Os aplicativos devem proteger as chaves e outros segredos usando cofres de senha (Vault) com todo processo de gestão de acesso a segredos.

Garanta que chaves simétricas, senhas ou segredos de API gerados por, ou compartilhados com, clientes são usados apenas para proteger segredos de baixo risco, como cifrar o armazenamento local ou usos efêmeros temporários, como ofuscação de parâmetros. Compartilhar segredos com clientes é "equivalente" a ter dados em texto claro.

Tratamento de erros, logs e auditoria

Garanta que exista um formato padrão e uma abordagem comum para tratamento de log usada em todo o sistema

Garanta que os logs sejam transmitidos com segurança para um sistema externo especialista, para análise, detecção e alerta de risco. Quando viável, como processo automatizado de contenção (bloqueio de uma conta de acesso, por exemplo).

Proteção de dados e privacidade

Todos os dados confidenciais devem ser identificados e classificados em níveis de proteção requerido.

Todos os níveis de proteção devem conter os requisitos de criptografia, requisitos de integridade, retenção, privacidade e outros requisitos de confidencialidade (LGPD/GDPR).

Anônimo

Pseudo anônimo

Informação pessoalmente identificável (PII)

Informação sensível pessoalmente identificável

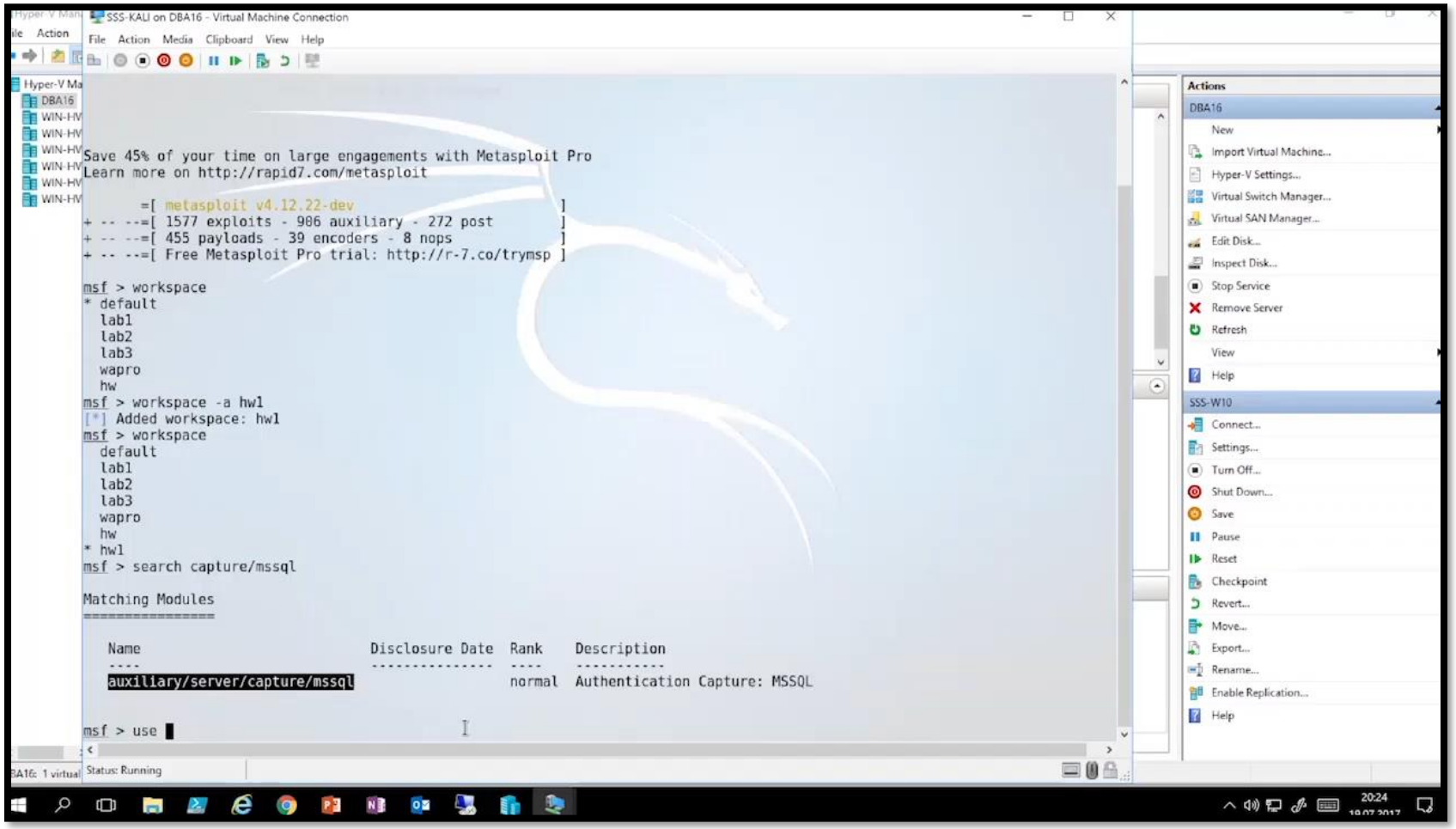
Comunicações entre os componentes

O aplicativo deve criptografar as comunicações entre os componentes, principalmente quando esses componentes estão em contêineres, sistemas, sites ou provedores de nuvem diferentes.

Os componentes do aplicativo devem verificar a autenticidade de cada componente em um canal de comunicação para evitar ataques de MiTM (Man in the Middle). Por exemplo, os componentes do aplicativo devem validar certificados e cadeias TLS.

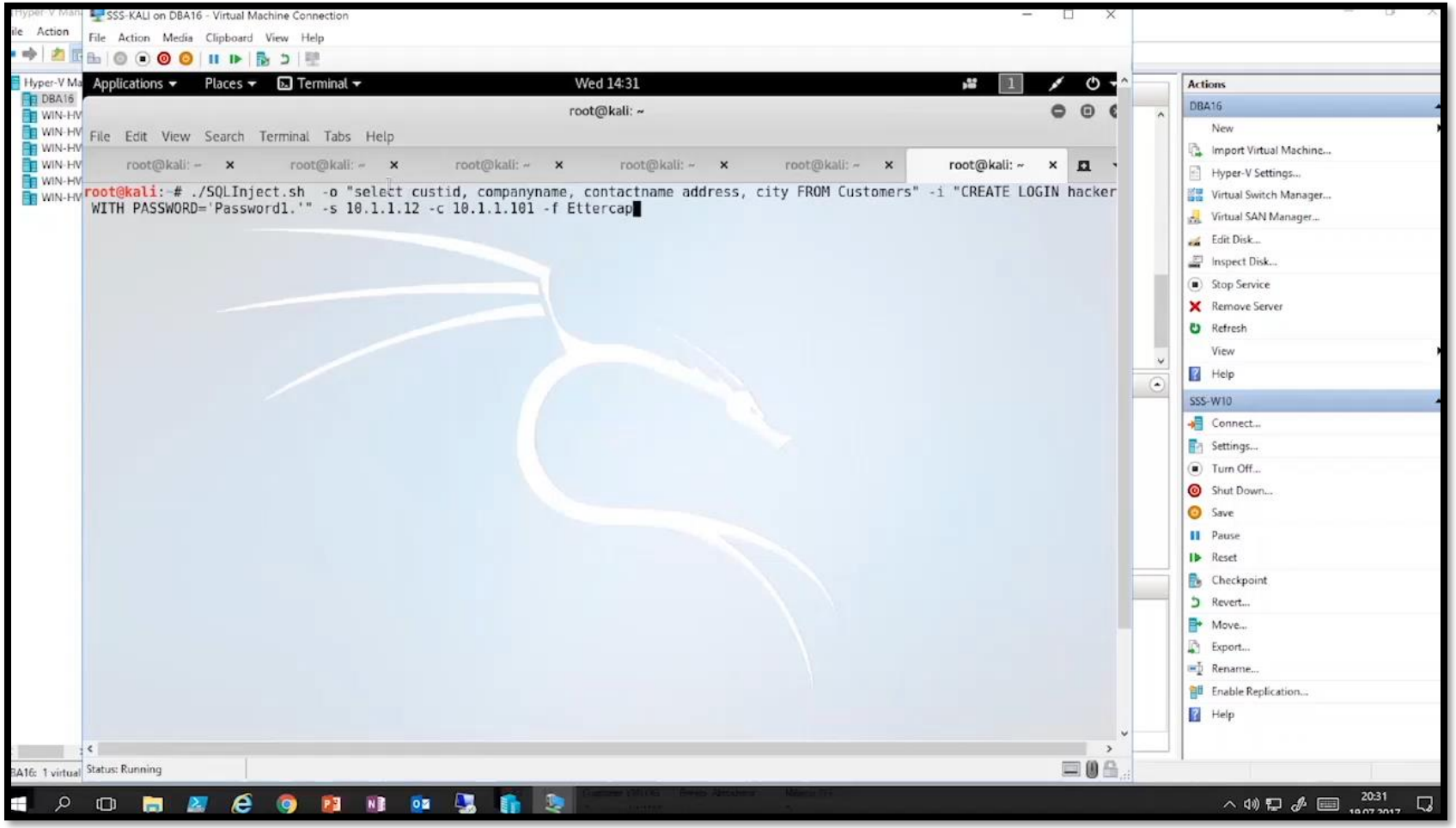
Comunicações entre os componentes

Seu acesso a bancos de dados é seguro?



Comunicações entre os componentes

Você se preocupa com configurações básicas de uso de funcionalidades de segurança nos componentes?

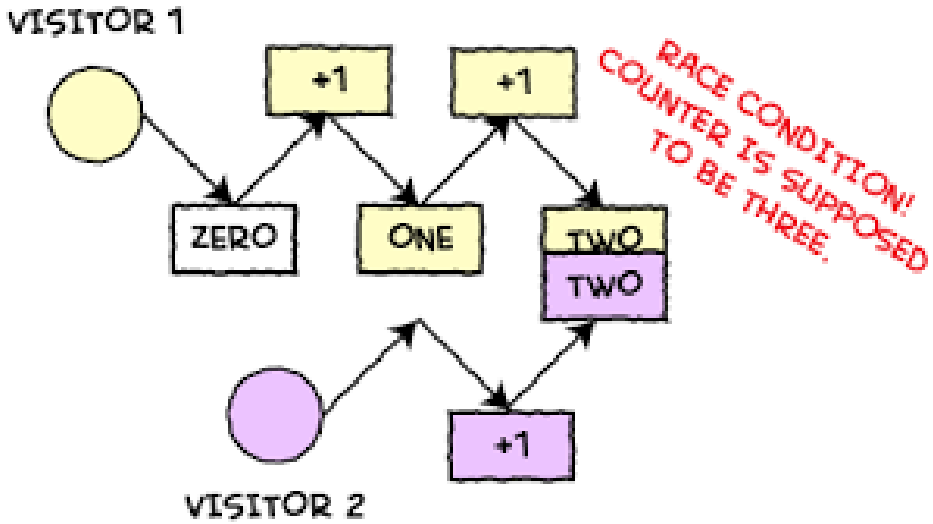


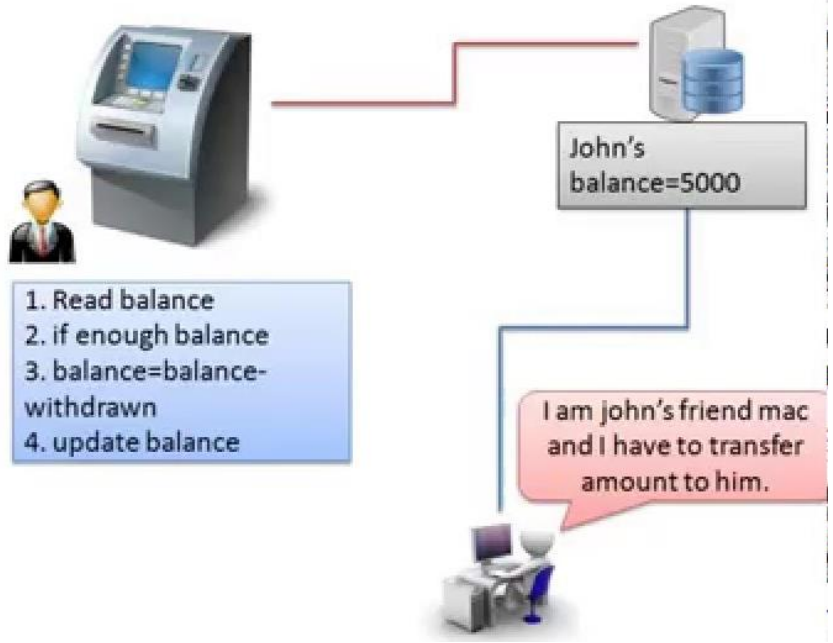
Lógica de negócio

Defina e documente todos os componentes do aplicativo em termos das funções comerciais ou de segurança que eles fornecem.

Garanta que todos os fluxos de lógica de negócios de alto valor, incluindo autenticação, gerenciamento de sessões e controle de acesso:

1. Sincronizem o estado da sessão entre fluxo concorrentes
2. São seguros e resilientes às condições de corrida entre o momento de verificação e o momento de uso da função.

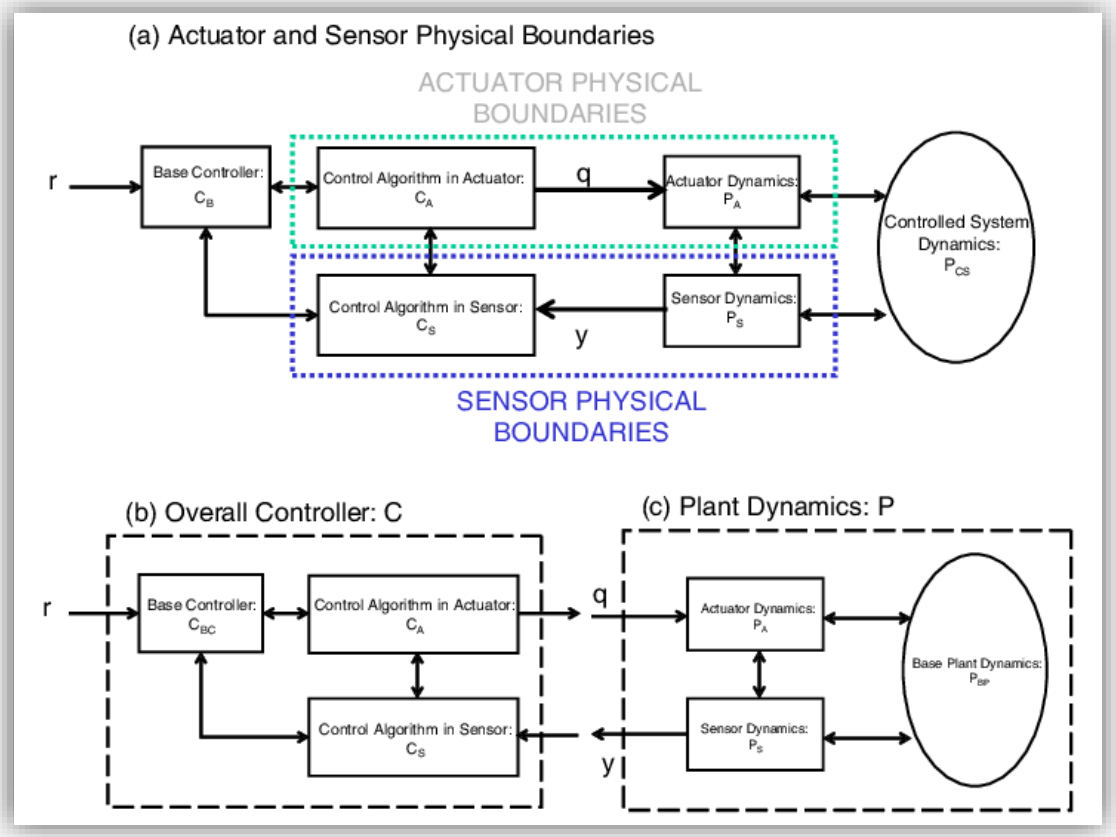




Hardenização de componentes

Garanta que as implantações de aplicativos protejam, empacotem e/ou isolem adequadamente o nível da rede para atrasar e impedir que invasores ataquem outros aplicativos, especialmente quando estão realizando ações confidenciais ou perigosas, como a desserialização de objetos

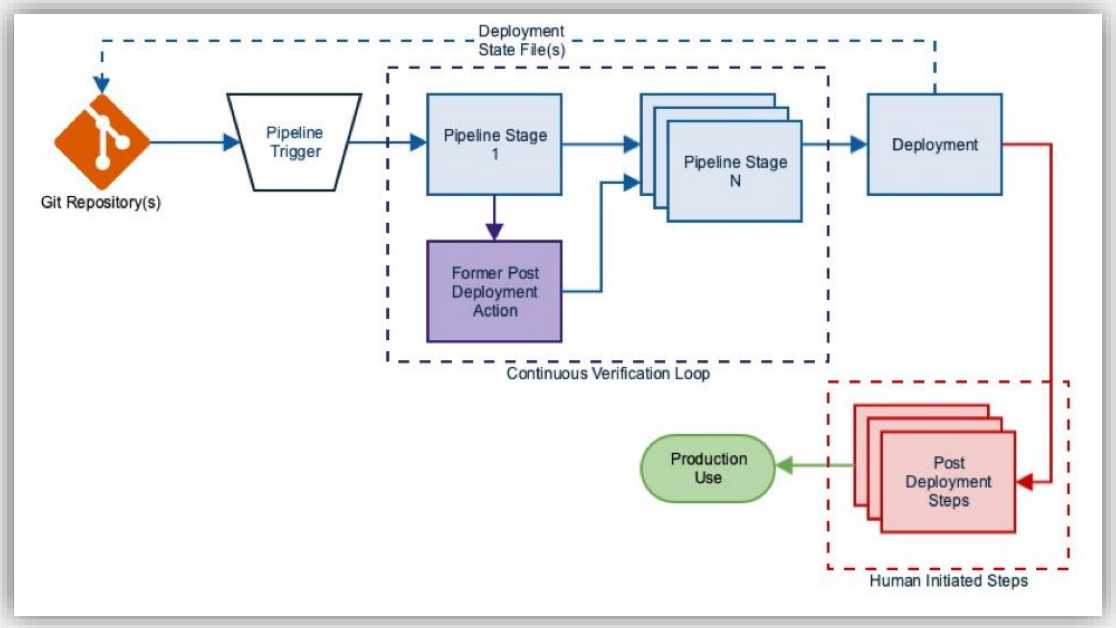
Garanta que o aplicativo não use tecnologias não suportadas, inseguras ou reprovadas, como plug-ins da NSAPI, Flash, ActiveX, Silverlight, applets Java no cliente.



Construção e entrega de componentes

Garanta que o pipeline de construção avise sobre componentes desatualizados ou inseguros e toma as ações apropriadas.

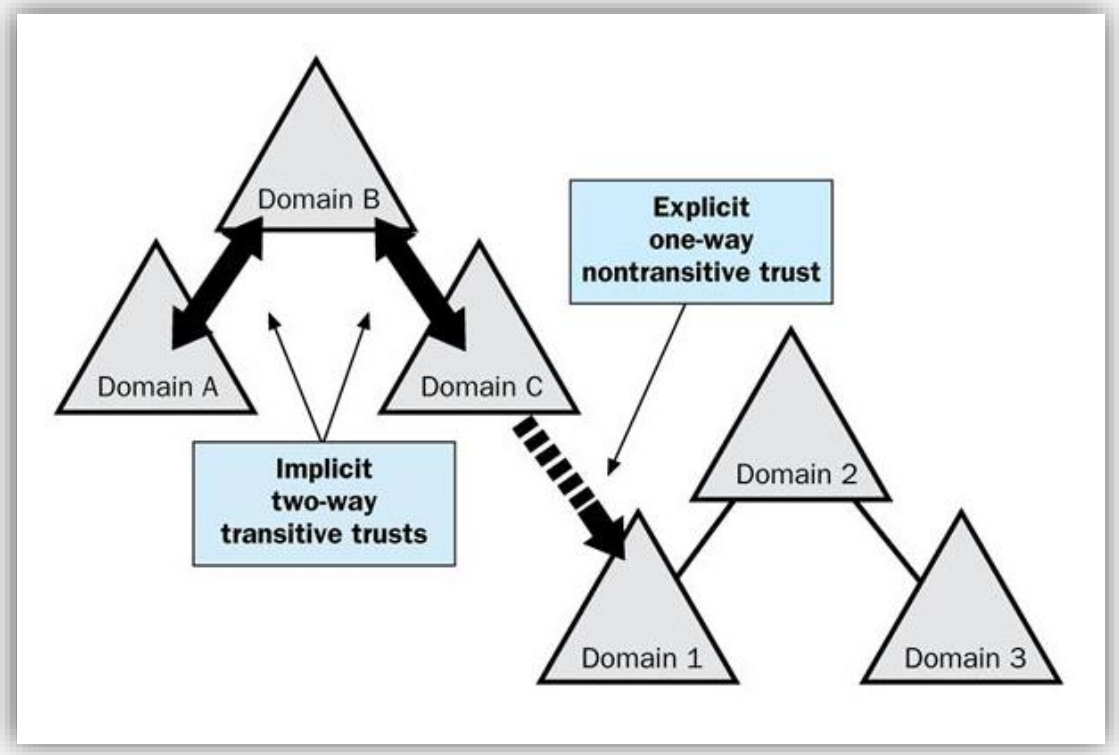
Garanta que o pipeline de construção contenha uma etapa de criação para criar e verificar automaticamente a implantação segura do aplicativo, especialmente se a infraestrutura do aplicativo for definida por software, como scripts de criação de ambiente em nuvem.



Componentes e relações de confiança

Segregue componentes de diferentes níveis de confiança por meio de controles de segurança bem definidos, regras de firewall, gateways de API, proxies reversos, grupos de segurança baseados em nuvem ou mecanismos semelhantes.

Verifique se a implantação de binários em dispositivos não confiáveis faz uso de assinaturas do binário, conexões confiáveis e endpoints verificados.



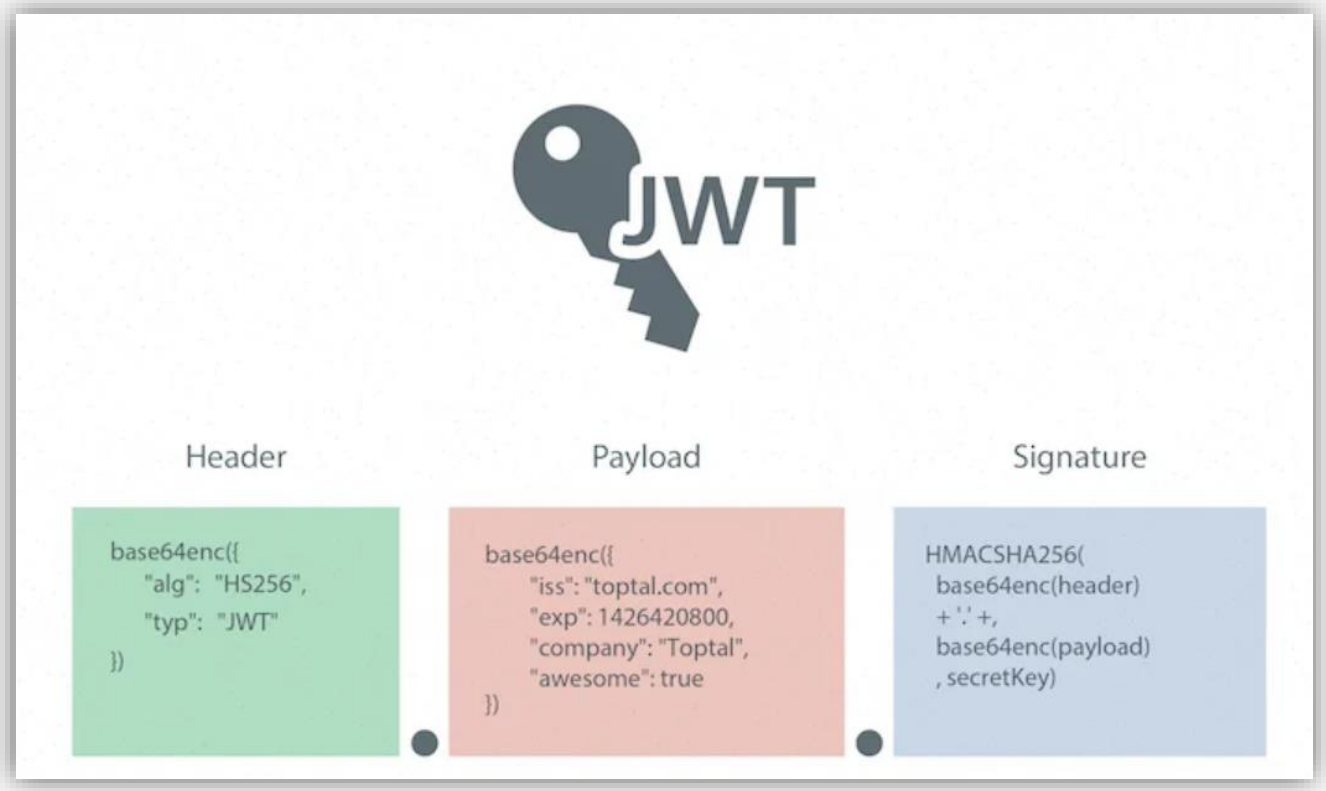
Um pouco mais sobre requisitos

Down the rabbit hole

Tratamos de alguns requisitos de forma bem abrangente e simples, mas questões complexas como processos de autenticação e controle de acesso, logs etc. merecem um detalhamento extra visando nivelar o conhecimento de arquitetos e desenvolvedores.

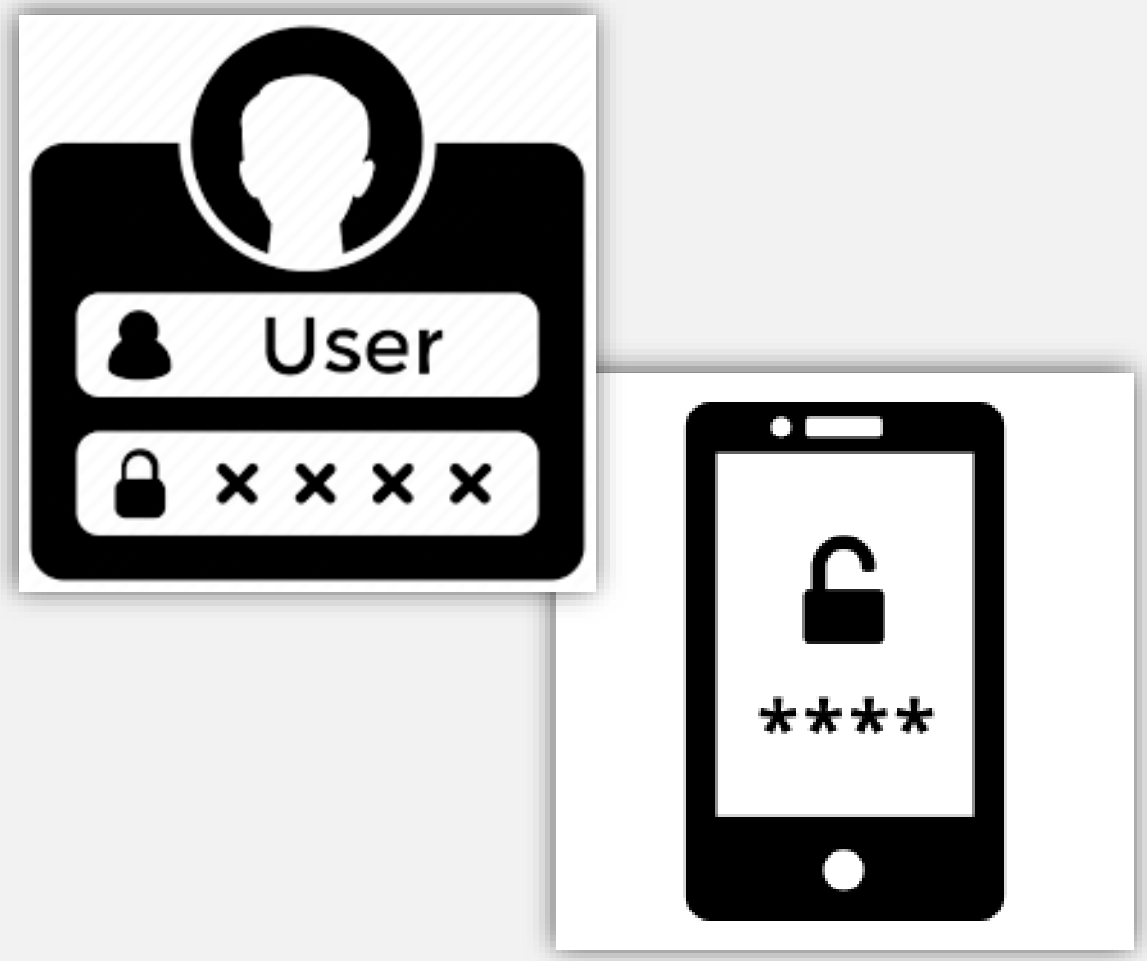
Autenticação

Autenticação é o ato de estabelecer ou confirmar que alguém (ou algo) é quem diz ser e que as declarações feitas por uma pessoa ou por um dispositivo estão corretas, resistem à falsificação de identidade e impedem a recuperação ou a interceptação de senhas.



Autenticação

"Usuário + senha" era a forma mais comum de autenticação fora dos sistemas de alta segurança. A autenticação multifator (MFA) era comumente aceita nos círculos de segurança, mas raramente exigida em outros lugares. À medida que o número de violações de senha aumentava, a ideia de que nomes de usuários eram de alguma forma confidenciais e senhas desconhecidas tornava muitos controles de segurança insustentáveis.



Autenticação

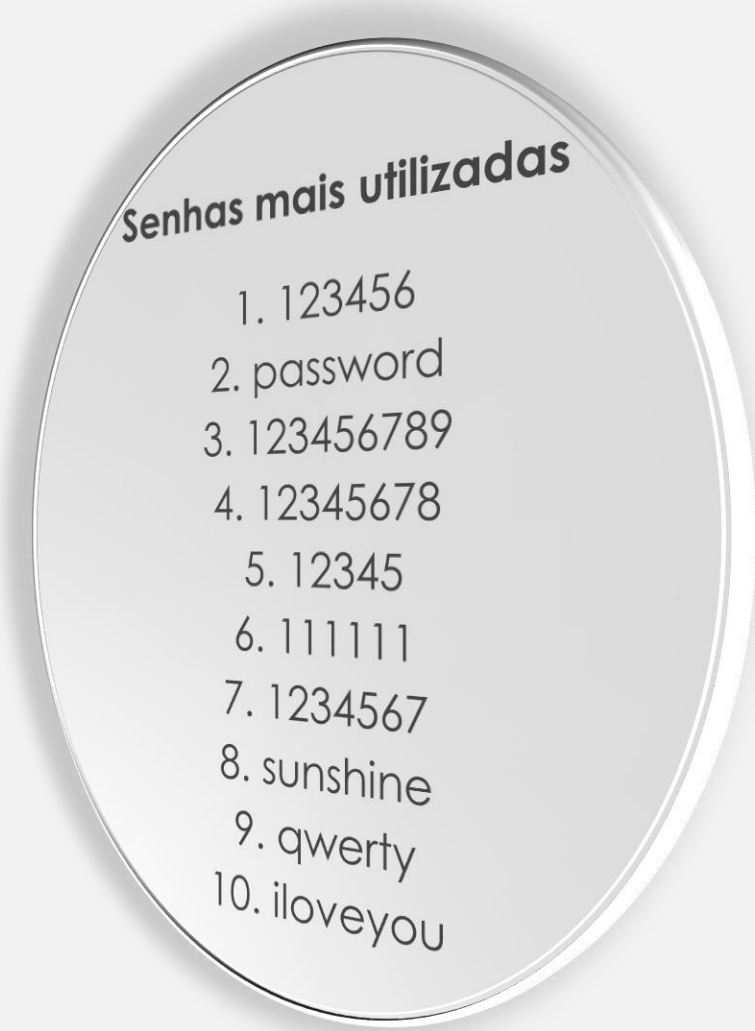
Por exemplo, o NIST 800-63 considera nomes de usuários e autenticação baseada em conhecimento (KBA) como informações públicas, SMS e notificações por e-mail como tipos de autenticadores "restritos" e senhas como pré-violadas.

Essa realidade torna os autenticadores baseados em conhecimento, recuperação de SMS e e-mail, histórico de senhas, complexidade e controles de rotação inúteis.

Senhas

Senhas, chamadas "Segredos Memorizados" pelo NIST 800-63, incluem senhas, PINs, padrões de desbloqueio e outros elementos de imagem e senhas. Eles são considerados "algo que você conhece".

Usar um único fator de autenticação esbarra nos bilhões de nomes de usuário e senhas válidos divulgados na Internet, nas senhas padrão ou fracas, nos rainbow tables e dicionários ordenados das senhas mais comuns.

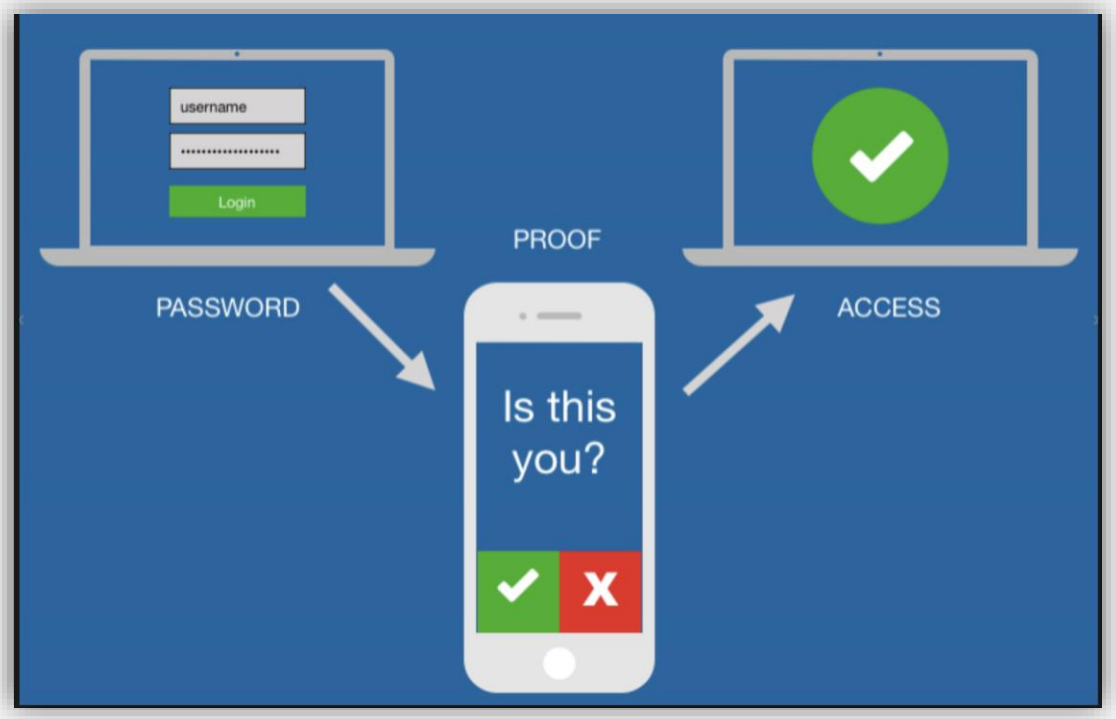


Identidades federadas

Os aplicativos devem incentivar os usuários a usar MFAs e reutilizar tokens que já possuem, como tokens FIDO ou U2F, ou vincular suas contas a CSP que forneça MFA.

Os provedores de serviços de credenciais (CSPs) fornecem identidade federada para os usuários. Os usuários podem ter mais de uma identidade, como uma identidade corporativa (Azure AD, Okta, Ping Identity ou Google) e uma identidade pessoal (Facebook, Twitter, Google etc.)

Deve-se avaliar o uso de identidades de usuários existentes de acordo com o perfil de risco e da "força" do CSP.



Gerenciadores de senhas

Os aplicativos devem checar se a senha de um usuário consta como “violada” em bases de senhas vazadas.

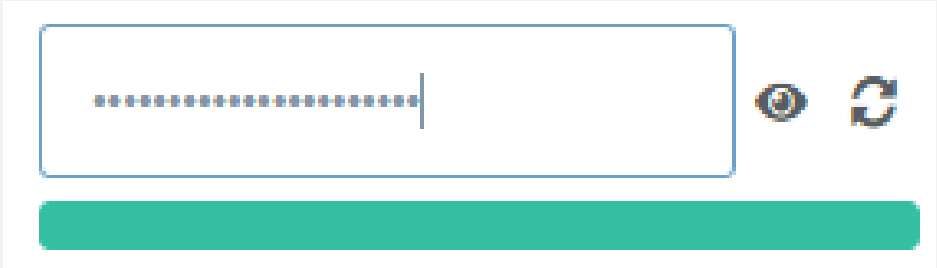
Se a senha for uma senha violada, o aplicativo deve exigir que o usuário defina uma nova senha “não-violada”. Se a senha tiver sido submetida no momento do login, um evento de alarme deve ser disparado para alertar sobre possíveis tentativas de quebra do controle de acesso, bloqueando a origem da tentativa maliciosa de acesso.

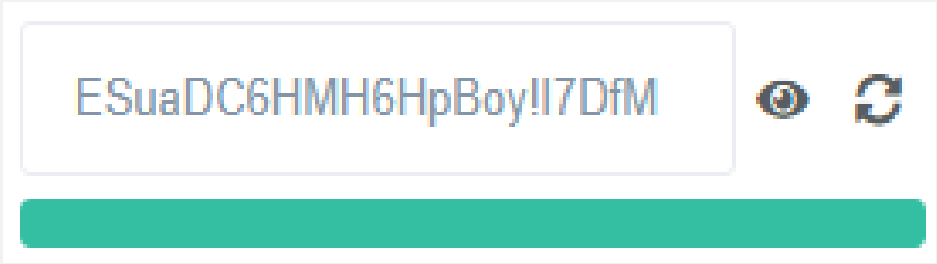
Um medidor de força da senha deve ser fornecido para ajudar os usuários a definir uma senha mais forte, bem como, uso de sistemas gerenciadores de senhas deve ser estimulado. Para isso, barreiras para a “cola” (ctrl+v) da senha devem ser removidas, incentivando a adoção de senhas realmente robustas e “fáceis de decorar” (keypass).

Usabilidade de senhas

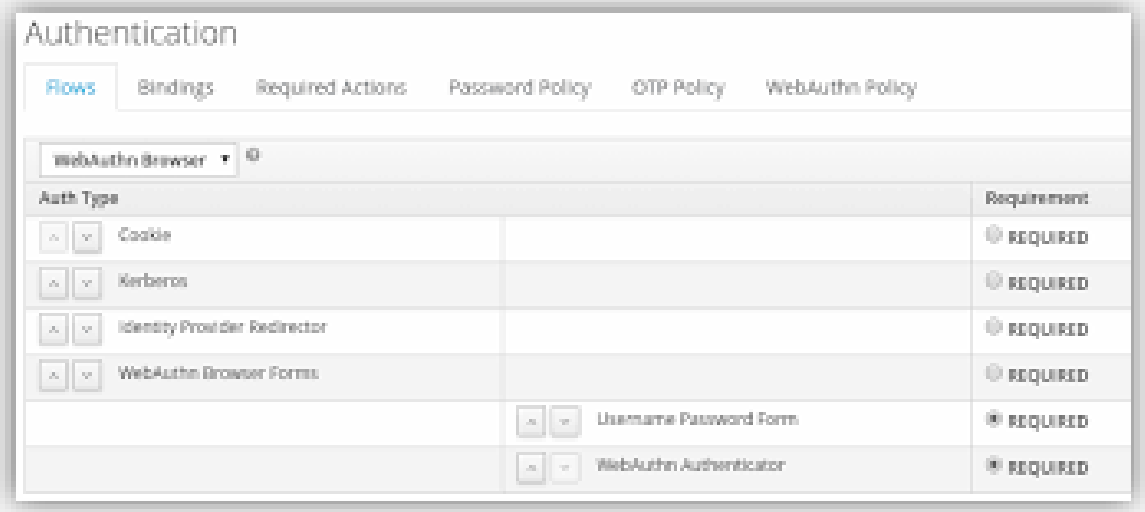
Verifique se o usuário pode optar por visualizar temporariamente toda a senha mascarada ou visualizar temporariamente o último caractere digitado da senha em plataformas que não têm isso como funcionalidade nativa.

O objetivo de permitir que o usuário visualize sua senha, ou veja o último caractere temporariamente, é melhorar a usabilidade da entrada de credenciais, particularmente em torno do uso de senhas longas e gerenciadores de senha que são longas por padrão.





Autenticação



Refatore os aplicativos para permitir autenticadores adicionais de acordo com as preferências do usuário, bem como permitir a retirada de autenticadores reprovados ou inseguros de “maneira ordenada”. O NIST considera e-mail e SMS como tipos de autenticador "restritos", e eles provavelmente serão removidos do NIST 800-63 em algum momento futuro. Os aplicativos devem planejar um roteiro que não exija o uso de email ou SMS.

CAPTCHAS

Verifique se os controles de anti-automação são eficazes na mitigação de ataques de teste de credenciais violadas (Força bruta).

1. Bloqueio de conta (pode ser temporário).
2. Bloqueio das senhas violadas e comuns.
3. Limitação de throughput e/ou atrasos crescentes entre tentativas.
4. CAPTCHA.
5. Restrições de endereço IP ou restrições com base em risco, como localização, primeiro login em um dispositivo, tentativas recentes de desbloquear a conta, ou similar.

Uso de SMS e e-mails

Autenticadores fracos (como sms e e-mail) limitam-se à verificação secundária e à aprovação de transações, e não como substituto de métodos de autenticação mais seguros. Os métodos mais fortes devem ser oferecidos antes de métodos fracos. Os usuários devem estar cientes dos riscos e medidas em vigor para limitar os riscos de comprometimento da conta.

Notificações devem ser enviadas aos usuários após redefinições de credenciais, alterações de e-mail ou endereço, login de locais desconhecidos.

O uso de notificações push é o preferível, mas o sms ou o e-mail pode ser usado, desde que nenhuma informação sigilosa seja divulgada na notificação.

Senhas vazaram...

Os processos de autenticação devem estar preparados para cenários de furto ou vazamento de credenciais (senhas).

As senhas devem ser armazenadas em um formato que seja resistente a ataques offline, ou seja, usando SALTs e HASHes fortes (criptografia de via única).

O SALT, que é secreto, é gerado usando um gerador de bit aleatório aprovado [SP 800-90Ar1] e forneça pelo menos a força de segurança mínima especificada na última revisão do SP 800-131A. Ele deve ser armazenado separado dos HASHes da senhas (por exemplo, em um dispositivo especializado como um módulo de segurança de hardware - HSM).

O SALT deve ter pelo menos 32 bits de comprimento e ser escolhido randomicamente para minimizar as colisões dele entre os hashes armazenados. Para cada credencial, um valor de SALT e o HASH resultante deverão ser armazenados.

Recuperação de acesso

- Não use perguntas secretas, elas são o caminho mais plausível de ataque e o ponto mais vulnerável da autenticação. Seu uso deve ser descontinuado.

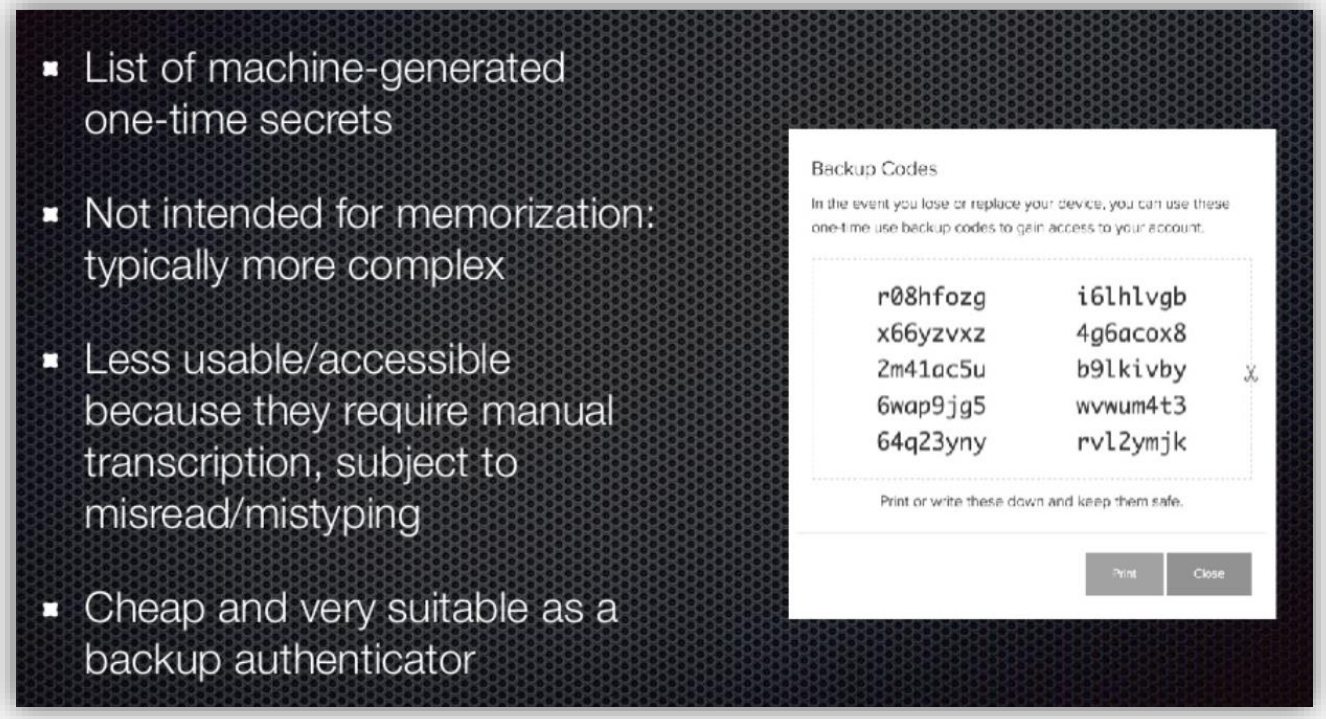
- Qualquer tentativa (com ou sem sucesso) de recuperação de senha deve disparar uma notificação para o usuário.

- Os mecanismos de recuperação da senha (esquecida) e outros caminhos de recuperação de acesso devem ser seguros (recuperação do OTP ou outro softtoken, uso de push para dispositivos móveis ou outro mecanismo de recuperação offline). Garanta que, se MFA for perdido, a prova de identidade será executada no mesmo nível usado durante a inscrição da conta (novo enrollment).

Look-up Secrets

Os Look-up secrets são listas pré-geradas de códigos secretos aleatórios. Esses códigos de consulta são usados uma vez e, depois de usa-los todos, a lista é descartada.

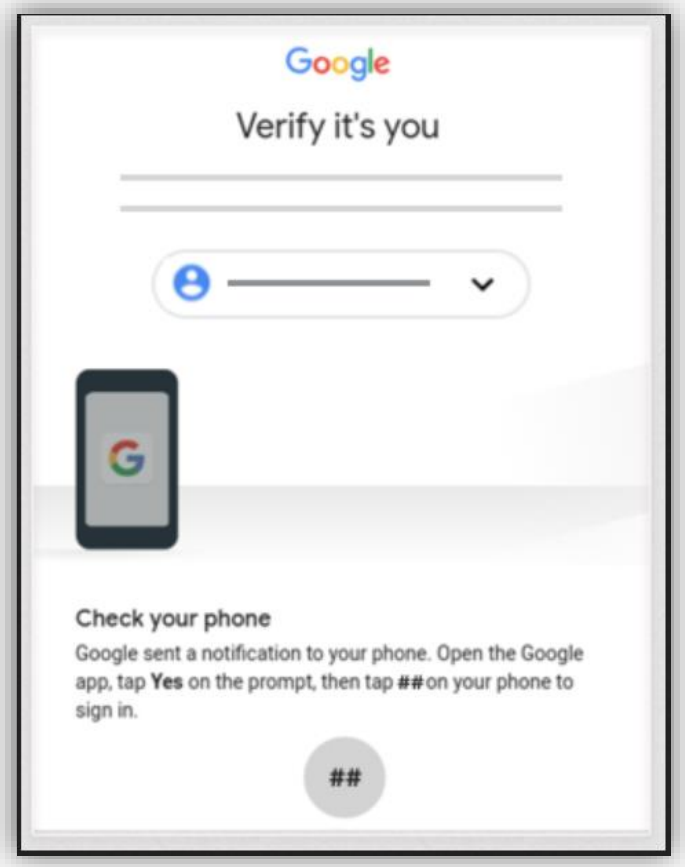
Estes códigos devem ter aleatoriedade suficiente (112 bits de entropia) e um salt exclusivo de 32 bits com um hash unidirecional seguro.



Out of Band Verifier

Os OOBV são meios de contato com o usuário através de um canal secundário seguro, por exemplo, enviando uma notificações push para dispositivos móveis. Este tipo de autenticador é considerado "algo que você tem".

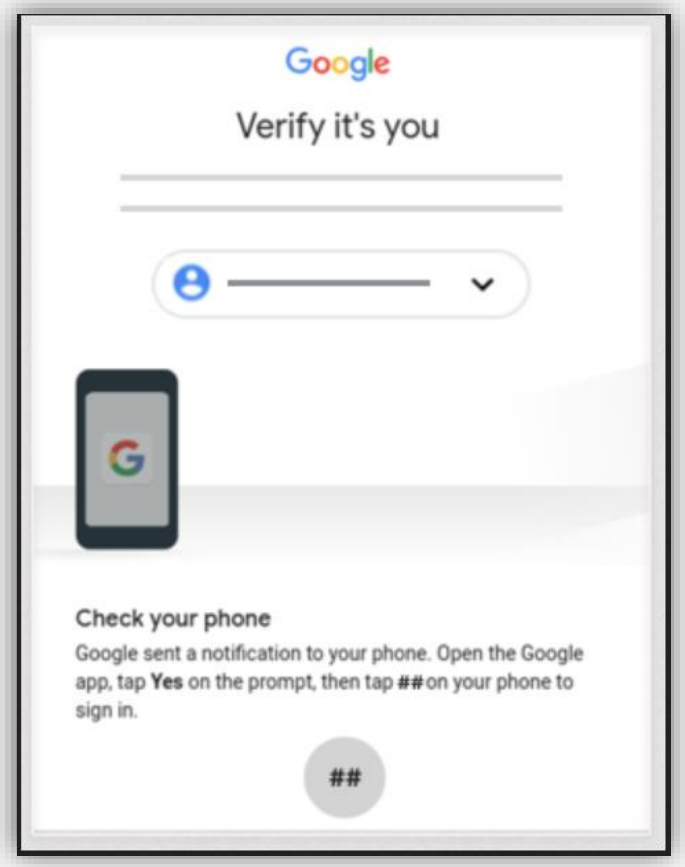
A mensagem contém um código de autenticação (normalmente um número aleatório de seis dígitos ou um diálogo de aprovação modal). Através do aplicativo insere-se o código de autenticação (canal principal) e compara o hash do valor recebido com do código de autenticação gerado. Se eles corresponderem, o usuário foi autenticado.



Out of Band Verifier

Alguns detalhes importantes sobre essa implementação são:

1. O Out of Band Verifier deve expirar as solicitações, códigos ou tokens após 10 minutos
2. Os códigos ou tokens de autenticação são utilizáveis apenas uma vez e somente para a solicitação de autenticação original
3. A aplicação executando o processo de autenticação e o OOBV devem se comunicar através de um canal seguro independente
4. O OOBV armazene apenas o hash do código de autenticação.

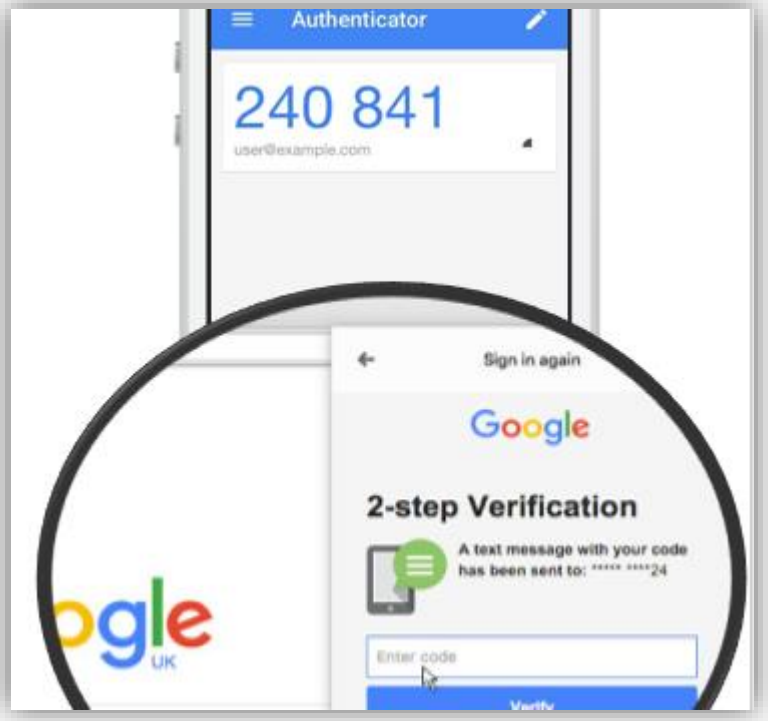


MFAs

Senhas OTPs (One Time Passwords) são tokens físicos ou softtokens que exibem um conteúdo efêmero pseudoaleatório. Algumas definições importantes são:

- Ciclo de vida adequado
  - Deve ser usado apenas uma vez em até X segundos após a geração (por exemplo, expiração em 30 segundos).
  - Deve ser revogado em caso de roubo ou perda a partir de qualquer sessão registrada.
- O mecanismo de hash usado deve ter as chaves protegidas (salt).
- O proprietário do OTP deve ser notificado de tentativas de abuso.

Nota: Autenticadores biométricos devem ser apenas fatores secundários em conjunto com algo que você tem ou algo que você conhece.



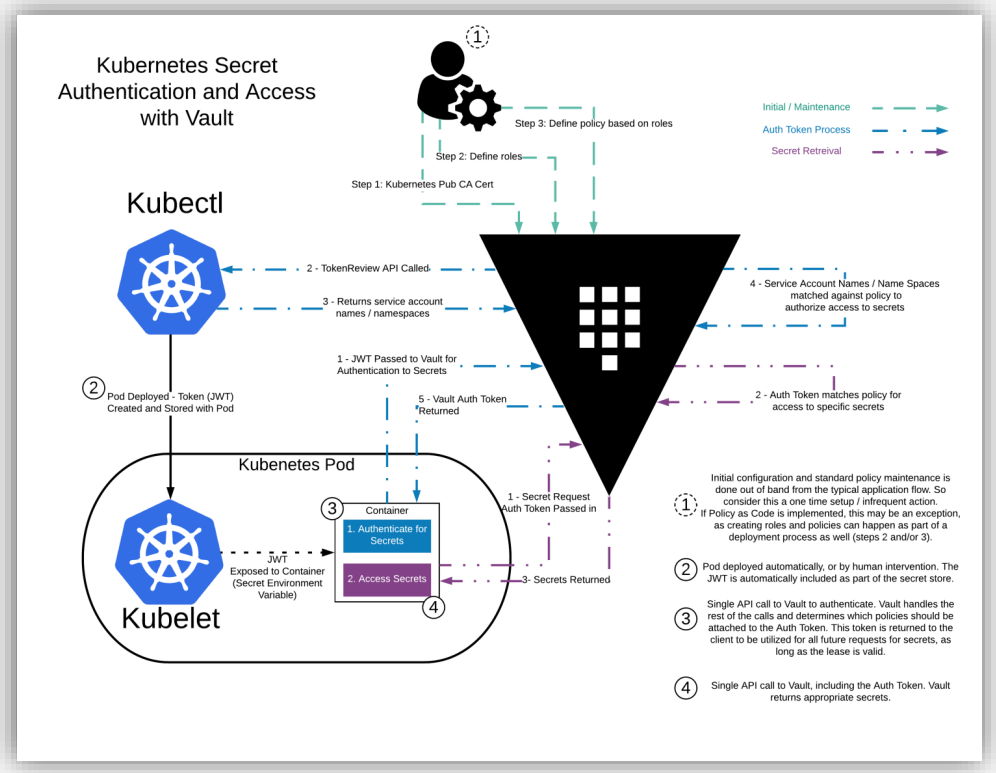
Contas de sistemas

Os segredos de integração precisam ser alteráveis.

Caso a opção de identificação seja uma senha, nunca usar senhas “default” ou padrão da companhia. As senhas devem ser armazenadas com proteção suficiente para impedir ataques de recuperação offline, incluindo o acesso ao sistema local.

Senhas, sementes, segredos e chaves de API devem suportar um ciclo de vida seguro, desde a geração, propagação, alteração e inativação.

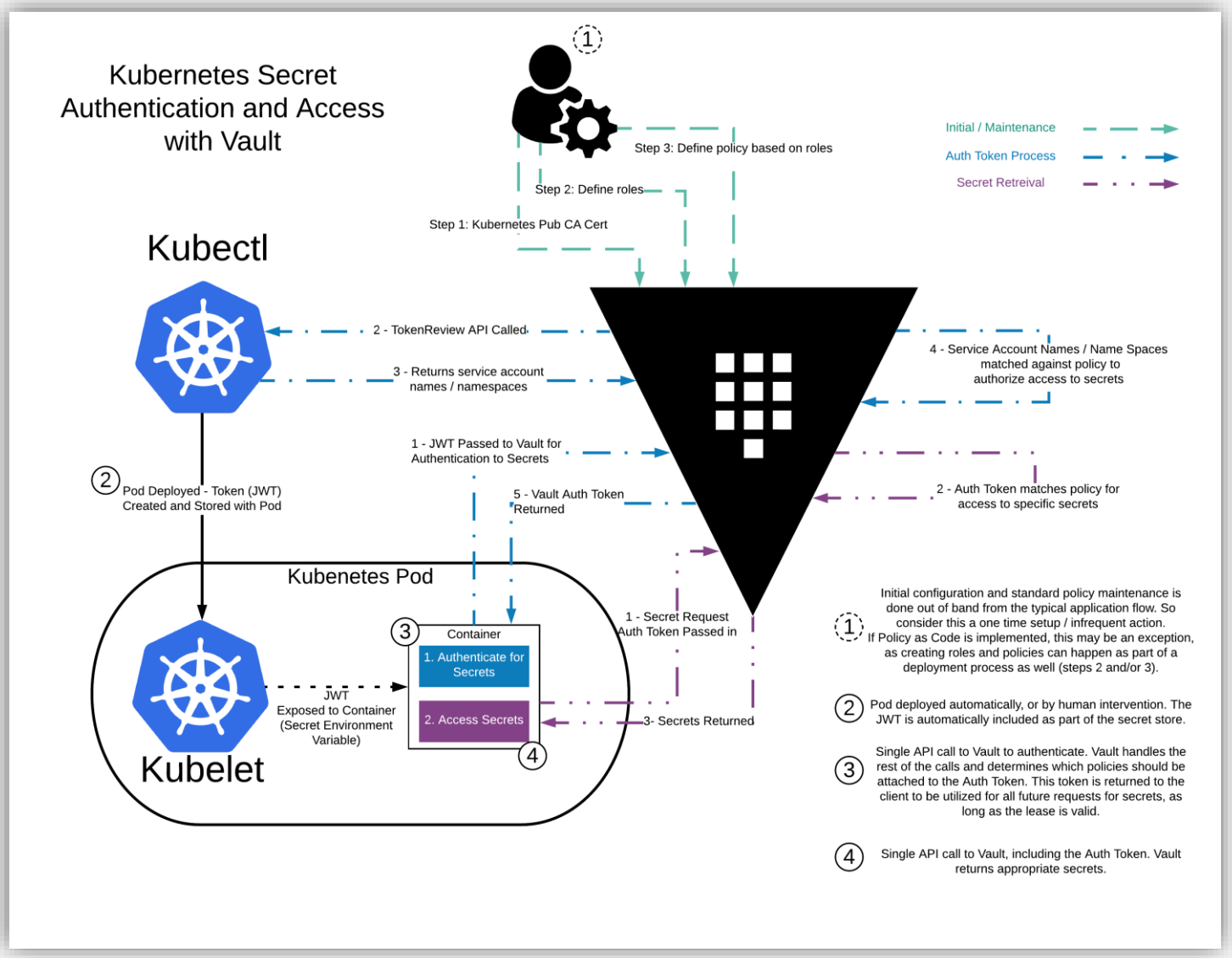
Devem ser armazenadas em um software key store seguro, de um hardware trusted platform module (TPM) ou de um hardware security module (HSM).



Uso de cofre de senhas

Exemplo de um uso seguro de cofre de senhas

Em linhas gerais, o sistema de "instanciação" de containers injeta no container um acesso efêmero que dá acesso a credencial da aplicação rodando no container.



Gerencia de sessão

Com base nos requisitos da ISO 27002, exigia-se o bloqueio de sessões simultâneas.

O bloqueio de sessões simultâneas não é mais apropriado, não apenas porque os usuários modernos têm muitos dispositivos ou porque o aplicativo é uma API sem uma sessão do navegador, mas na maioria dessas implementações, o último acesso autenticado "ganha" o acesso, que geralmente é o acesso do atacante. Assim, precisamos mudar a maneira de impedir, atrasar e detectar ataques as sessões.

O ataque mais comum se inicia bloqueando a conta e tentando redefinir ou recuperar uma credencial. Então utiliza-se uma sessão ou token válido e injeta-se os dados do perfil da vítima. A falha ocorre na verificação inadequada do controle de acesso (autorização para efetivar aquela ação sobre aquele dado).

Todas as transações sensíveis tem que ser protegidas por uma autenticação válida. Garanta uma sessão de login válida ou exija uma reautenticação ou verificação secundária antes de permitir transações sensíveis ou modificações na conta.

Controle de acesso

Autorização é o conceito de permitir o acesso a recursos apenas àqueles autorizados a usá-los.

- As pessoas que acessam recursos possuem credenciais válidas para fazer isso.
- Os usuários estão associados a um conjunto bem definido de papeis e privilégios.
- O processo de autorização deve ser protegido contra replay ou adulteração.

Como implementar:

- O aplicativo impõe regras de controle de acesso em uma camada de serviço confiável (backend).
- Os atributos do usuário usados pelos controles de acesso não podem ser modificados pelos usuários finais.
- Negue acesso por padrão. Não pode existir nenhum acesso a recursos até que o acesso seja atribuído explicitamente.

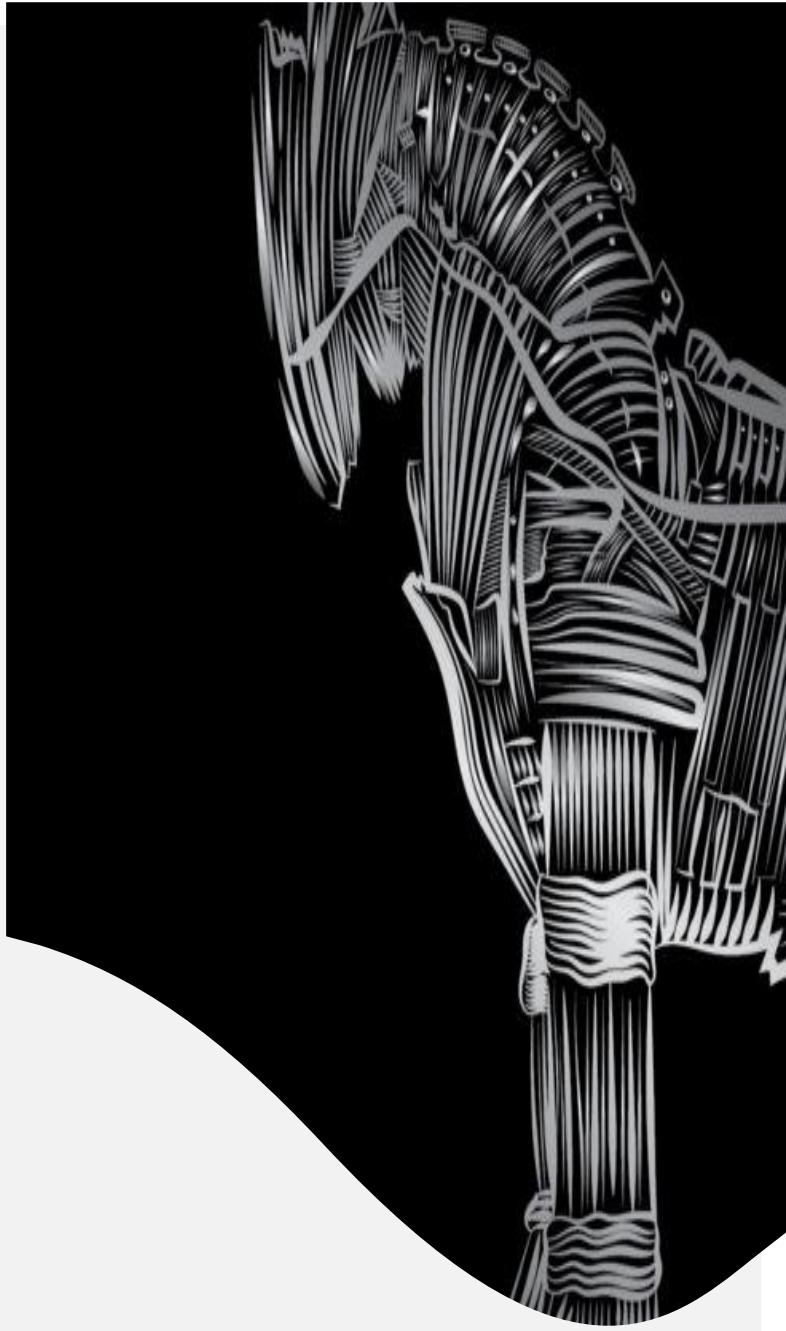
Detalhes da Autorização

Os controles de acesso tratam suas falhas (exceptions) com segurança.

Os dados confidenciais e as APIs estão protegidos, validando-se a autorização antes de qualquer outra ação.

O aplicativo utiliza um mecanismo anti-CSRF forte para proteger a funcionalidade autenticada (abuso do acesso)e um mecanismo anti-automação (bloqueio do acesso devido ao abuso).

As “funções administrativas” devem requerer MFA.



Logs de sistema

O principal objetivo do tratamento eventos é fornecer informações úteis para os administradores e equipes de resposta a incidentes. Não se trata de criar quantidades massivas de logs (ruídos), mas logs de alta qualidade (informações úteis).

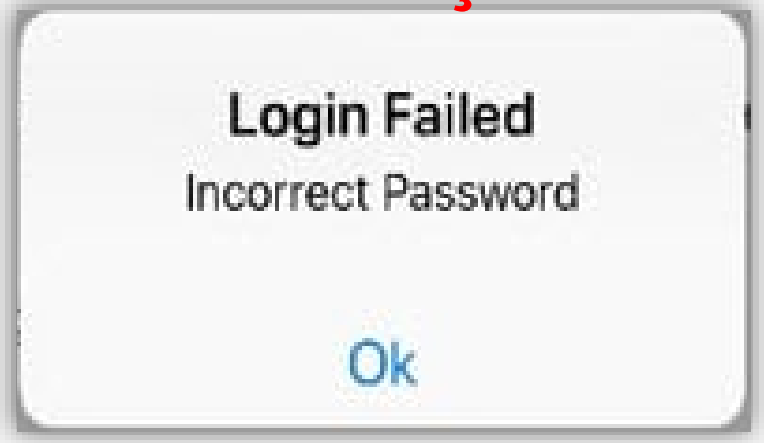
Logs de alta qualidade devem esclarecer 5 fatores: **Quem; Fez o quê; A partir de onde; Quando; Se teve sucesso no intento**. Isso deve incluir:

- Não coletar ou registrar informações confidenciais, a menos que seja especificamente necessário. O registro de dados privados ou confidenciais tornam os logs atrativos e, portanto, devem ser protegidos por criptografia. Jamais registre meios de pagamento ou credenciais.
- Garantir o controle de acesso aos logs.
- Garantir que os registros não sejam armazenados para sempre, mas tenham uma vida útil absoluta o mais curta possível.
- O aplicativo "encode" adequadamente os dados fornecidos pelo usuário para evitar injeções via logs (rodar comandos na interface que lê os logs).
- O horário correto dos eventos sejam registrados (timezone – considerar o uso abrangente do UTC).
- Sequenciar corretamente os eventos é imprescindível para análise de ataques.

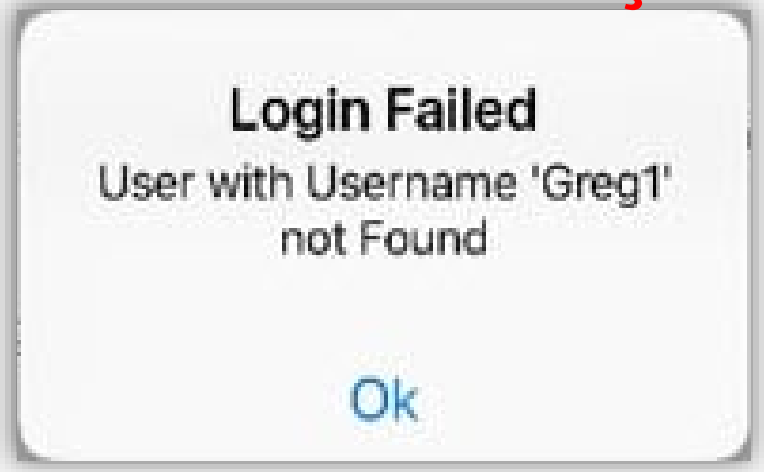
Tratamento de erros - alertas

O objetivo do tratamento de erros é permitir que o aplicativo forneça eventos de segurança relevantes para monitoramento, disparo de uma resposta a incidente. Ao registrar eventos relacionados à segurança, certifique-se de que haja um propósito para o registro e que ele seja de fato utilizado no correlacionamento de eventos (SIEM).

Alerta – Força Bruta



Alerta - Enumeração



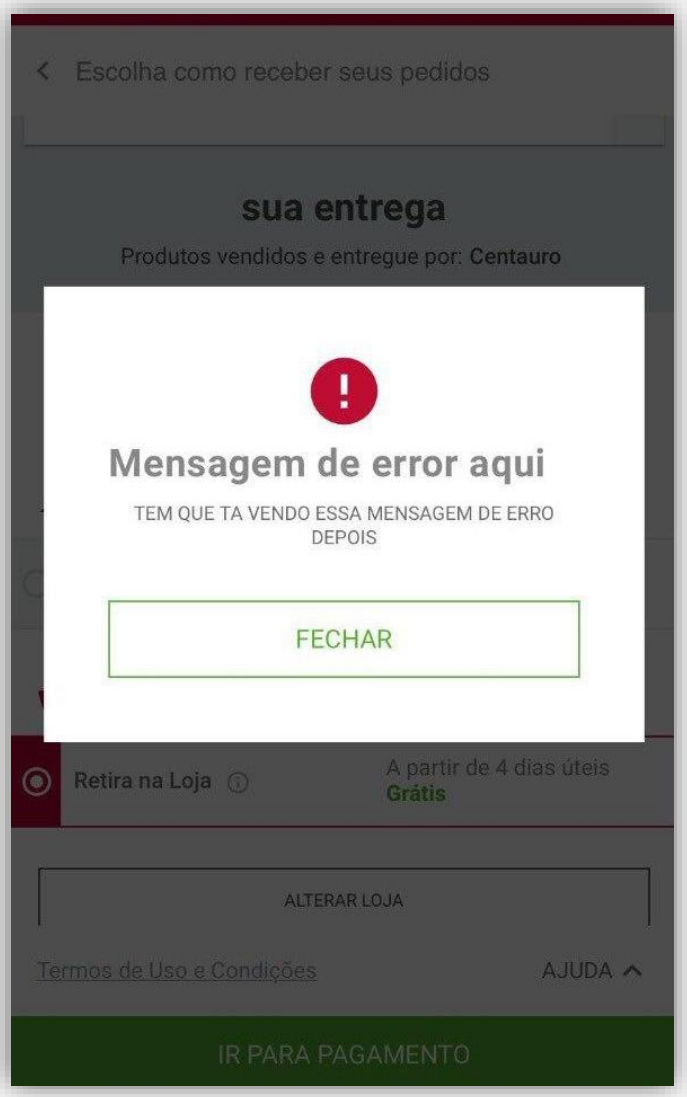
Tratamento de erros

Garanta que:

Uma mensagem genérica seja mostrada quando ocorre um erro (de segurança), com um ID único que a equipe de suporte pode usar para investigar o contexto do erro.

O tratamento de exceção seja usado no software para endereçar as condições de erro esperadas e inesperadas (um "último recurso" que captura todas as exceções não tratadas).

Nenhum tipo de exceção chegue a interface do usuário.



Arquivos e Recursos

Arquivos recebidos de fontes externas à aplicação não são confiáveis até que sejam validados. Arquivos “não confiáveis” obtidos de fontes “não confiáveis” devem ser armazenados de forma segura (permissionamento do arquivo) e fora da raiz do site.

Defina cotas de tamanho e número de arquivos por usuário. Não aceite arquivos que possam preencher todo o espaço de armazenamento(DOS).

Arquivos e Recursos

Proteja-se contra:

- RFD (reflected file download) validando ou ignorando nomes de arquivos enviados pelo usuário.
- Command Injection não os usados diretamente em APIs ou bibliotecas do SO.
- RFI e SSRF não os usando para acessar códigos e bibliotecas em servidores de terceiros.
- Vazamentos de dados liberando o download apenas as extensões necessárias, bloqueando todas as demais (.bak, .swp, .zip, .tar.gz, etc.).

Muito cuidado com:

- “Zip bombs" - pequenos arquivos de entrada que serão descompactados em arquivos grandes, esgotando assim os limites de armazenamento de arquivos.
- Validação de arquivos apenas pela extensão – Valide o conteúdo.
- Virus e Malwares – Valide todos os uploads.

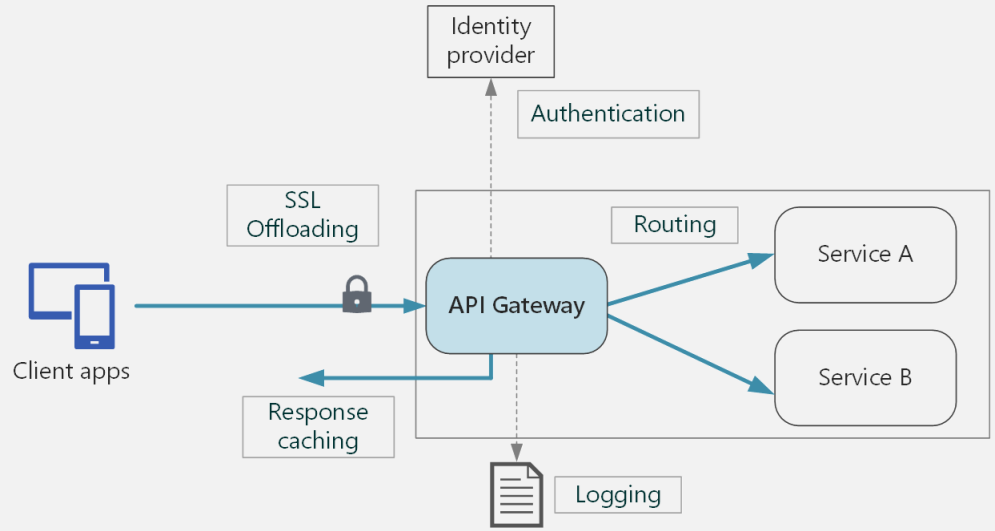
APIs

Todos os componentes do aplicativo usem componentes e parsers seguros para evitar ataques aos parses que exploram SSRF e RFI.

As URLs da API não exponham informações confidenciais, como a chave da API, os tokens de sessão, etc.

A checagem de autorização sejam efetuada em cada chamada no momento do acesso ao recurso.

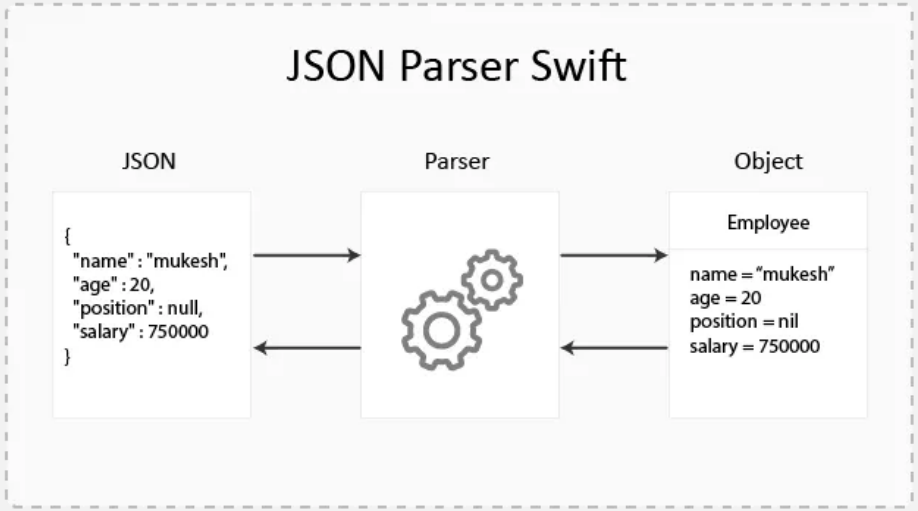
As solicitações que contêm tipos de conteúdo inesperados ou ausentes sejam rejeitadas apropriadamente (HTTP 406 inaceitável ou 415 tipo de mídia não suportado).



APIs - JSON

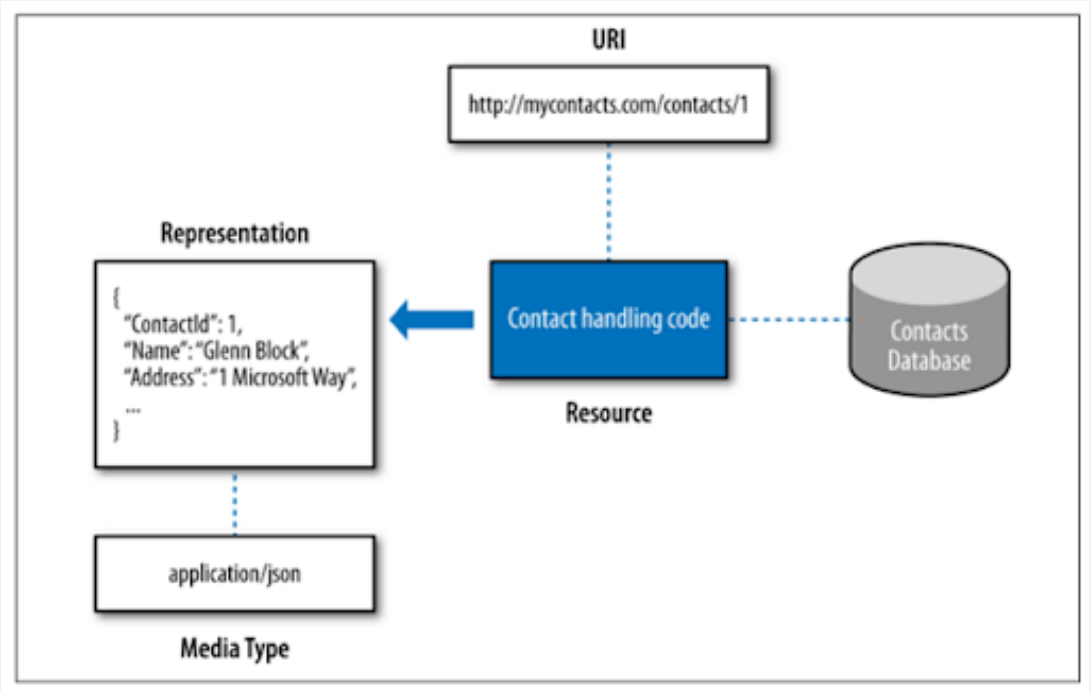
A validação do esquema JSON está em um estágio de rascunho de padronização (https://json-schema.org/). Considere o uso dessa estratégia:

- Validação formal de esquema JSON.
- Validação de estrutura do objeto JSON (identifique elementos ausentes ou extras).
- Verificação da integridade do objeto JSON usando métodos de validação de entrada padrão, como tipo de dados, formato de dados, comprimento etc.



APIs REST

- Os serviços que utilizam cookies devem se proteger contra CSRF (Triple or double submit cookie pattern, CSRF nonces ou checagem de cabeçalho ORIGIN).
- Os serviços devem estar protegidos contra chamadas excessivas, especialmente se a API não for autenticada.
- Verifique se o tipo de conteúdo recebido é esperado(application/xml ou application/JSON).
- Proteja a mensagem com (TLS). Assinaturas por mensagem podem fornecer garantia adicional.



APIs SOAP

- A validação do esquema XSD deve ocorrer para garantir um documento XML formado adequadamente, seguido pela validação de cada campo de entrada antes que qualquer processamento da mensagem ocorra.
- O payload da mensagem está assinada usando o WS-Security para garantir o transporte confiável entre o cliente e o serviço.