Arquitetura de segurança

Exemplo de análise e aplicação dos conceitos

Arquitetura e segurança

Existem diversos cenários de ataques que demandam uma solução arquitetural para proteger o sistema. Essa interdependência (segurança e arquitetura) e o uso das técnicas apresentadas até aqui (modelagem e requisitos) para detectar e responder a incidentes será apresentado nessa sessão.

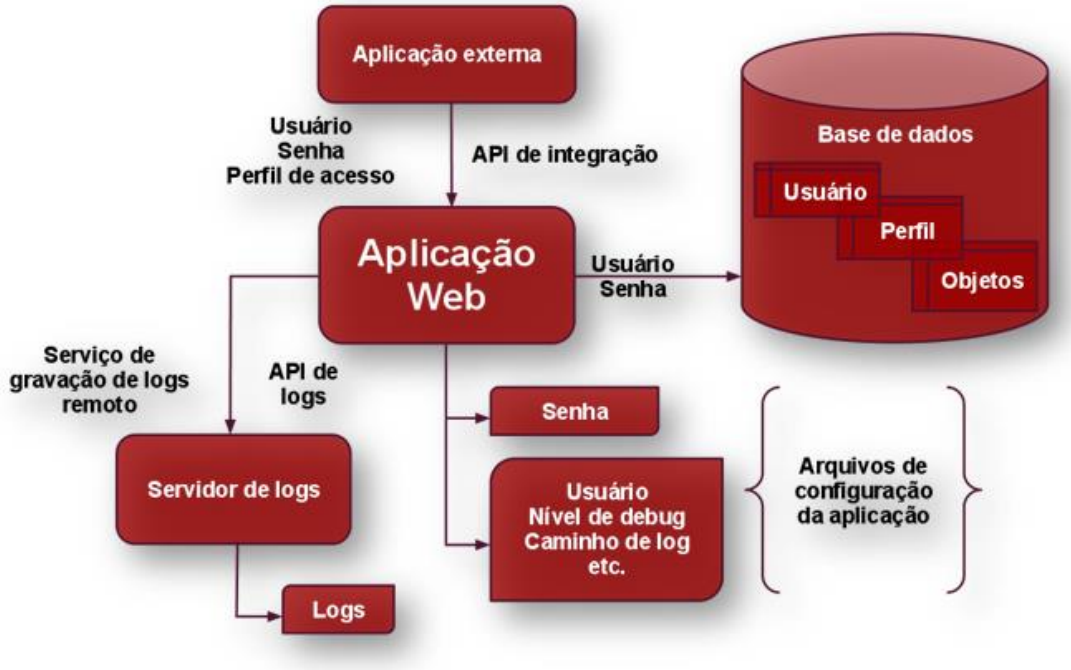
Problemas classicamente endereçados por arquitetura

- Interconexão de sistemas e serviços (Middlewares, Gateways)
- Inventário de componentes (libs, frameworks, componentes, design patterns a serem seguidos) e componentização.
- Organização do ecossistema de sistemas
- Dependências tecnológicas e obsolescência do parque.
- Redundância e alta disponibilidade

Arcabouço de segurança

Arquitetura alvo

Tomemos a arquitetura ao lado como base para iniciar o estudo. Note que a aplicação Web do exemplo persiste seus dados em um banco de dados, utiliza algumas APIs para expor algumas funcionalidades e registra tudo o que ocorre em um sistema de logs externo.



Arcabouço de segurança

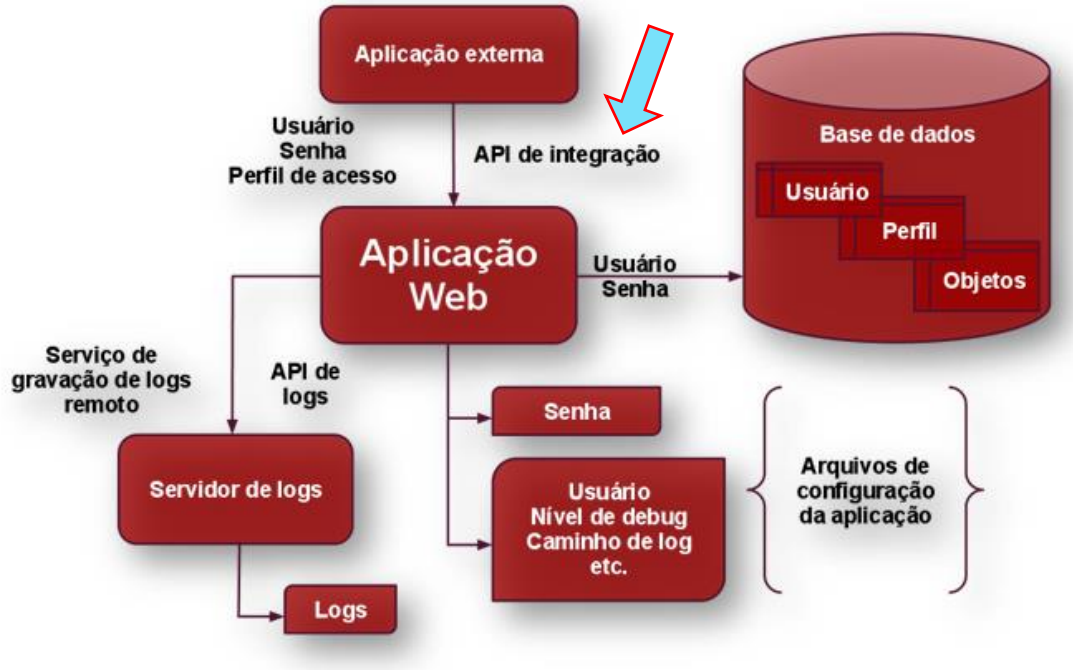
Arquitetura alvo

Interconexões

Escolhemos nesse exemplo a exposição de certos dados e funcionalidades através de **APIs** para que uma aplicação externa consuma esses dados.

Outra opção adotada em arquiteturas sem a preocupação de segurança é conceder o acesso direto ao banco de dados para que a aplicação externa se conecte.

Por que criar a complexidade de uma API, sua exposição, ciclo de vida, gestão em contraponto a um simples acesso?



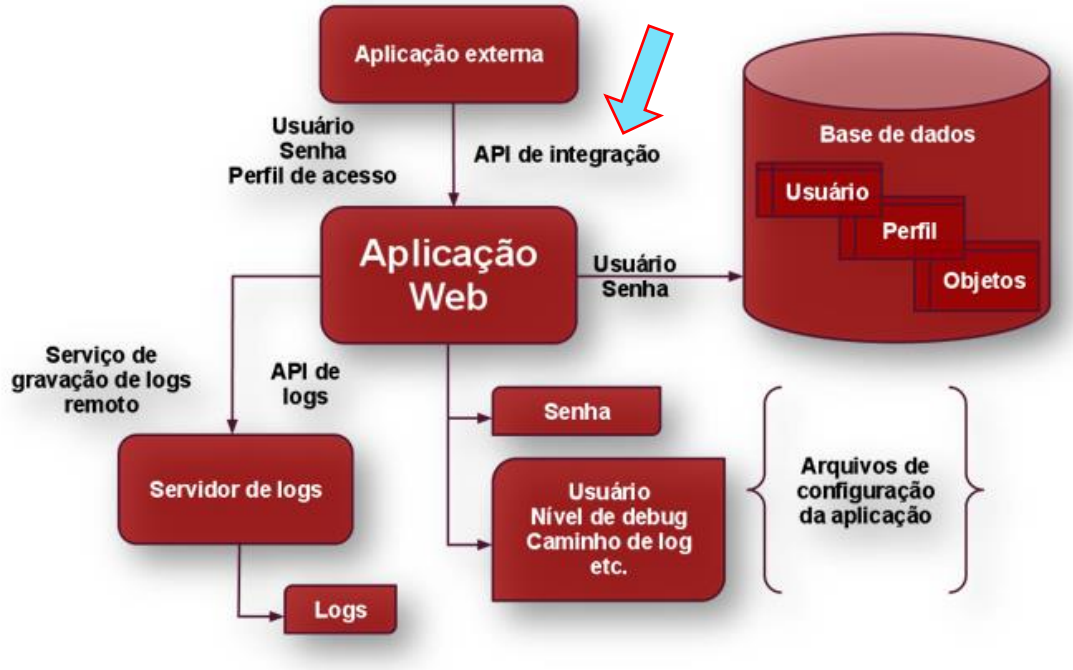
Arcabouço de segurança

Arquitetura alvo

Interconexões

A medida que a aplicação externa fizesse acesso diretos aos dados, o tipo do dado, a semântica do dado, tudo passaria a ser compartilhado entre as aplicações, acoplando-as e dificultando a manutenção delas. Ambas teriam que ser refatoradas caso o modelo de dados fosse alterado.

Toda a lógica de controle de acesso teriam que ser reescritos nas duas aplicações. Ou seja, a probabilidade de um vazamento seria proporcional dificuldade de se invadir a aplicação mais vulnerável.



Arcabouço de segurança

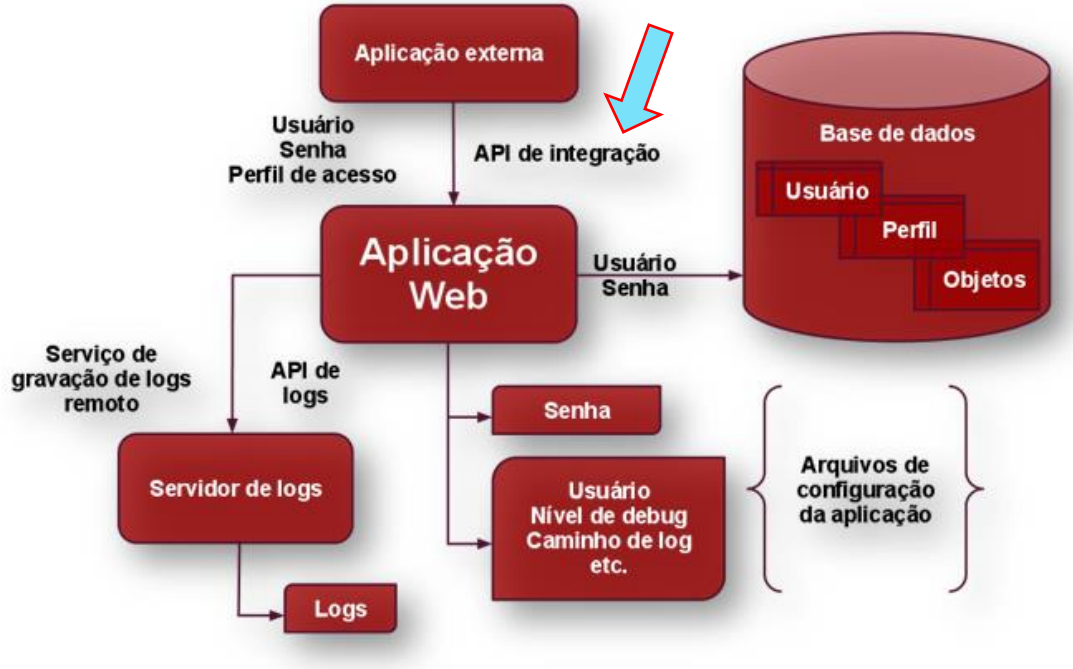
Arquitetura alvo

Assuma a aplicação Web como o sistema de RH e que a aplicação externa é o "aniversariantes do mês da intranet".

O acesso a base de dados foi concedido diretamente ao "aniversariantes do mês" que faz uma chamada pesada ao banco cada vez que a intranet é aberta.

Com muitos acessos, o sistema do RH enfrenta lentidão e aciona-se os DEVs para "responder ao incidente"!

A solução: aumentar o processamento e/ou memória do banco de dados



Arcabouço de segurança

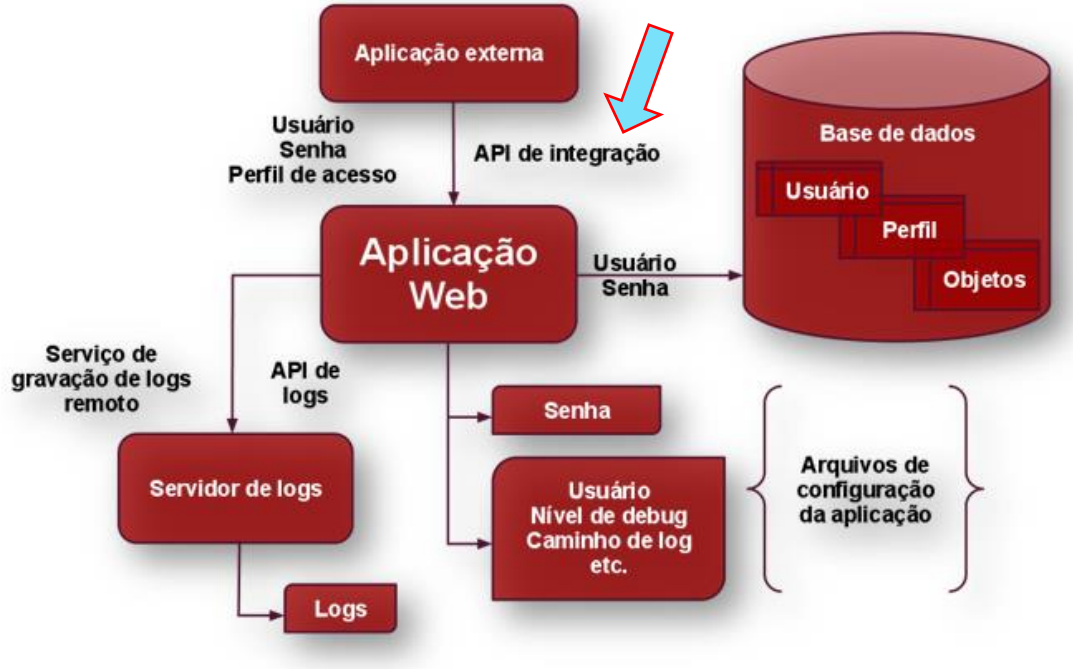
Arquitetura alvo

E se um atacante quisesse acessar e vazar a base de salários dos funcionários.

O quão óbvio e difícil seria atacar o sistema do RH?

Assuma que por erros de controle de acesso seja possível usar o acesso do "Aniversariantes do mês" para extrair todos os salários.

O atacante invadiria o "aniversariantes do mês", tomaria a credencial de acesso e, com sorte, **ela teria o nível de acesso necessário para extrair os dados de forma irrastreável**.



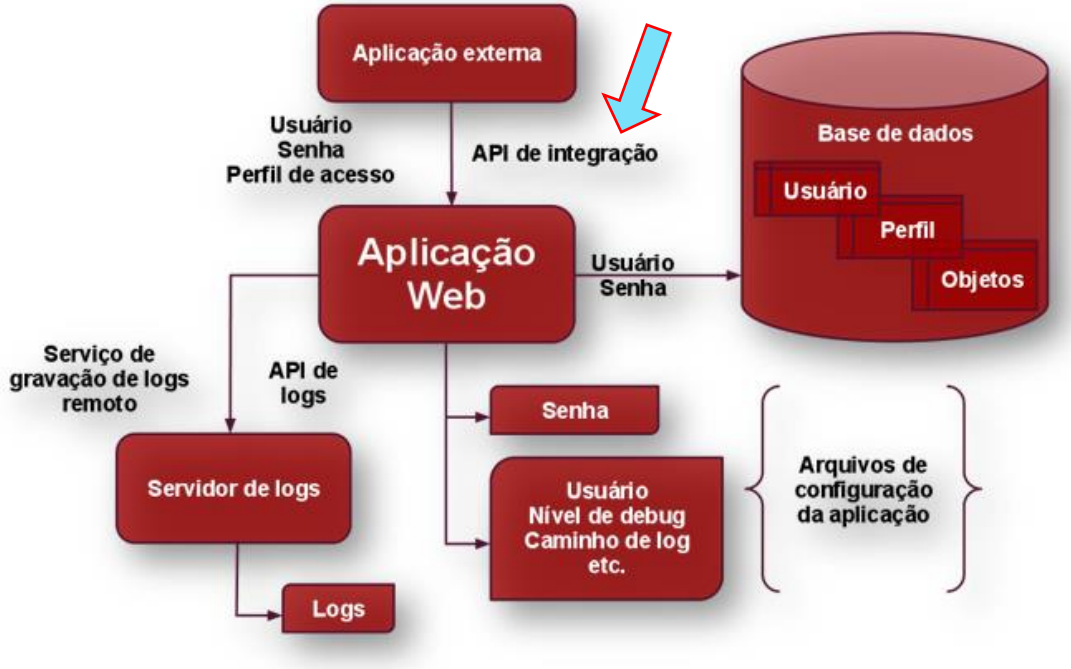
Arcabouço de segurança

Arquitetura alvo

Essa fragilidade arquitetural ocorre porquê, os acessos externos bypassam a lógica de controle de acesso nativa da aplicação proprietária dos dados;

O nível de exigência de segurança do "aniversariantes do mês" é muito menores porque não se previu aquele caminho de ataque;

Os logs de acessos aos dados não têm os mesmos critérios e controles da aplicação proprietária.



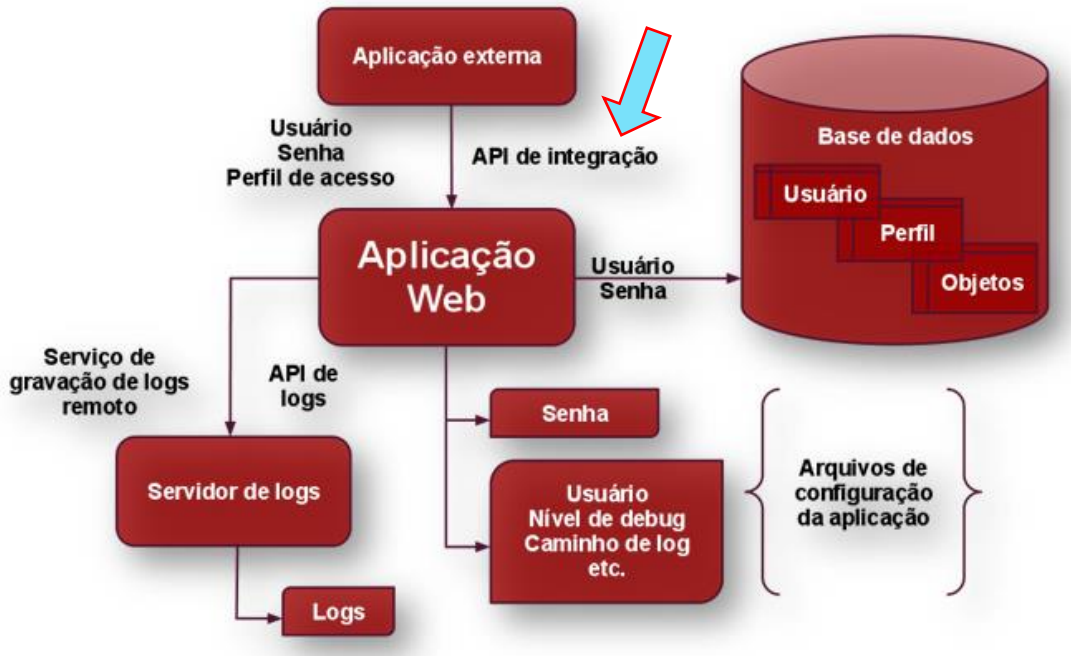
Arcabouço de segurança

Interconexões via APIs garantem:

Que todas as requisições passem pela lógica de controle de acesso da aplicação proprietária da base de dados.

Que existam logs de tudo que for necessário, pois é mais fácil entender a criticidade da informação pelo prisma da aplicação proprietária da base (“Sistema de RH” e não do “Aniversariantes do Mês”)

Controle de acesso - caso a aplicação externa se “comporte mal”, seu acesso a API pode ser revogado, mantendo o serviço disponível para outros consumidores e impedindo o acesso do ofensor (resposta a incidente seletiva).



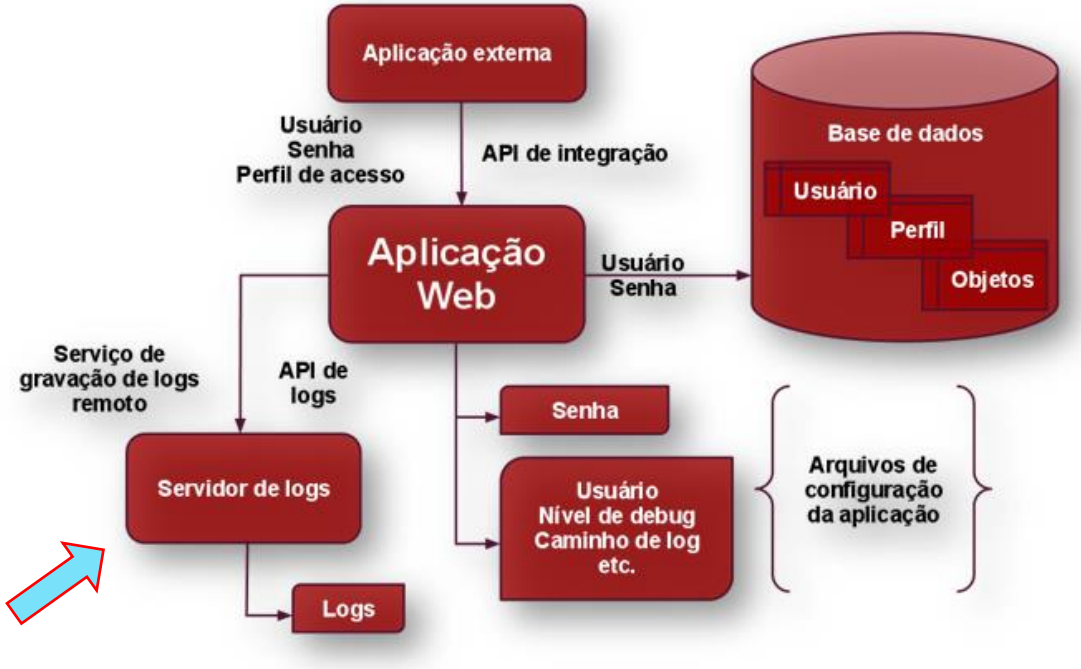
Arcabouço de segurança

Componentização

Os logs são registrados através de sistema de logs externos via APIs de escrita de logs que permitem apenas a escrita e não a leitura (one way).

Há vários riscos relacionados ao registro dos logs, mas vale destacar que muitos sistemas escrevem logs localmente e tem que administrar seu armazenamento, ciclo de vida etc. e esse trabalho acaba se replicando em dezenas de sistemas.

Entretanto, toda a gestão do ciclo de vida de logs deve, e pode, ser feita em um sistema centralizado.



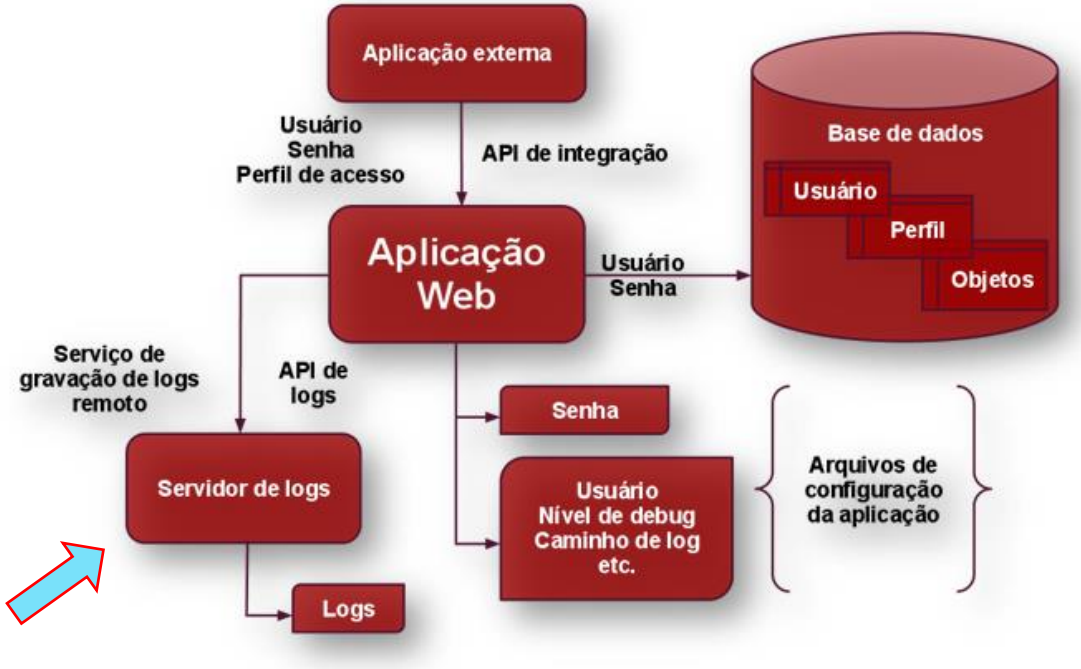
Arcabouço de segurança

Arquitetura alvo

Componentização

O log deve ser encarado como o mecanismo que a aplicação tem para "solicitar ajuda" para quem possa "ajudá-la". Ou seja, se você guarda logs localmente e ninguém os analisa, eles não têm serventia.

Deve existir na solução um componente responsável por endereçar esses "pedidos de ajuda" da aplicação, pois ela pode não ser capaz de resistir sozinha aos ataques. Para isso é vital que os logs sejam íntegros e confiáveis e salvá-los localmente em arquivos não nos dá essa garantia.



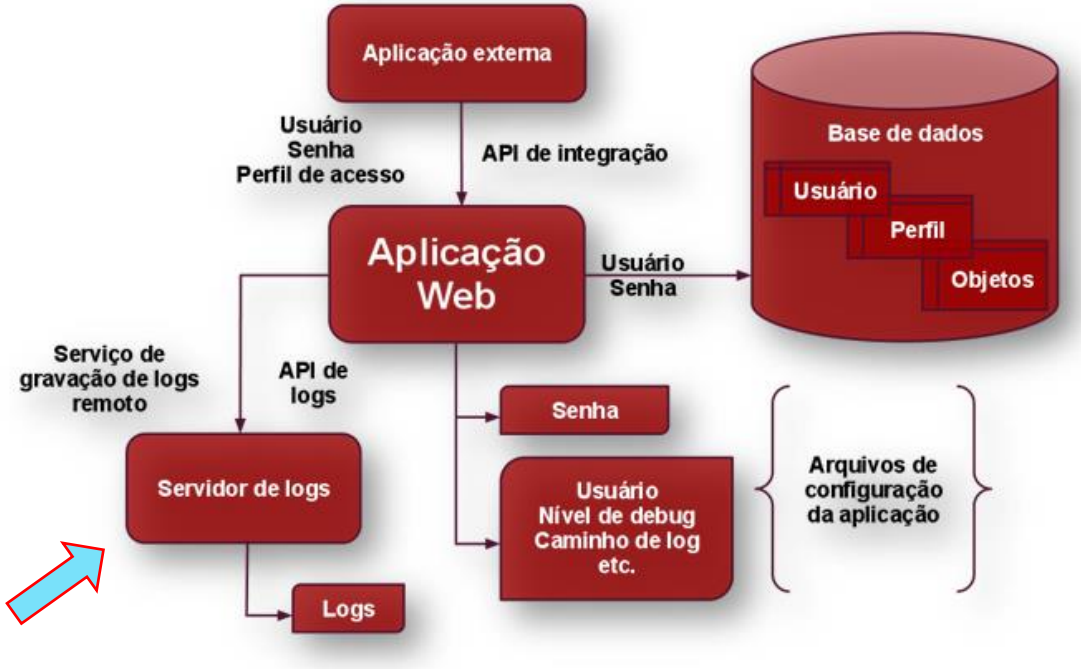
Arcabouço de segurança

Arquitetura alvo

Componentização

Por exemplo, uma aplicação está sob ataque de força bruta (estão tentando invadir contas da aplicação). A aplicação sozinha não consegue parar o atacante.

Se os logs são confiáveis e íntegros, pode-se coletá-los e processá-los em sistemas de correlação de logs para análise desse cenário (que vai acontecer em uma sexta-feira de carnaval às 18h quando ninguém da monitoração está realmente atento ao ataque).

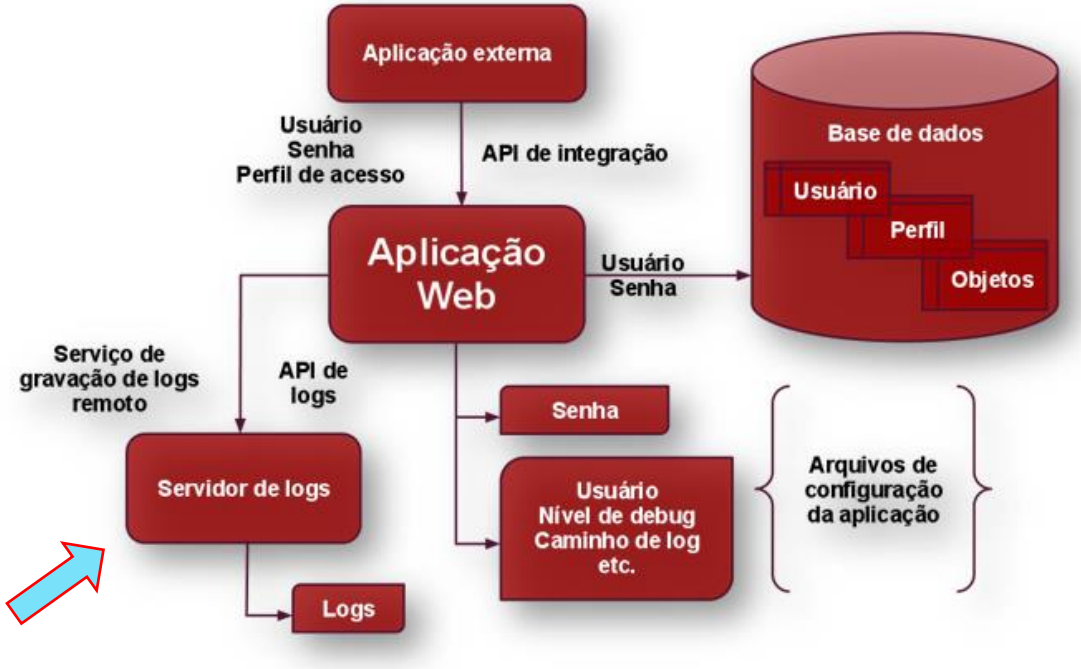


Arcabouço de segurança

Arquitetura alvo

Uma vez correlacionados os logs e identificado o ataque, uma resposta automatizada pode ser disparada, por exemplo, ativando um CAPTCHA durante o processo de login e aumentando sensivelmente o tempo entre uma tentativa de login e outra, chegando até a banir momentaneamente a origem do ataque (IP).

Ou seja, toda resposta a incidentes começa com a coleta de logs em um componente arquitetural confiável. Sem esse componente, não há resposta real a incidente.

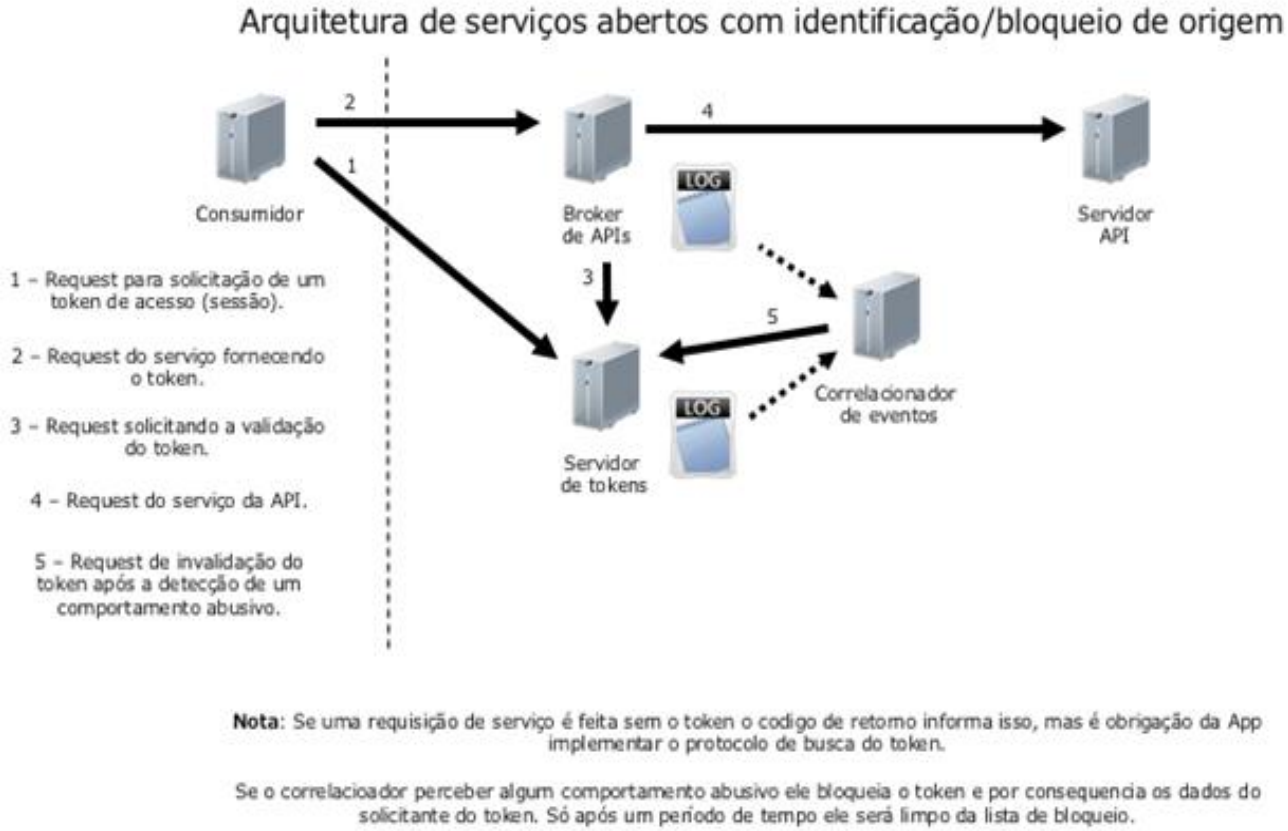


Arcabouço de segurança

Componentização

No exemplo ao lado ficam mais claros os componentes da resposta a incidentes automatizada e o papel dos logs nesse processo.

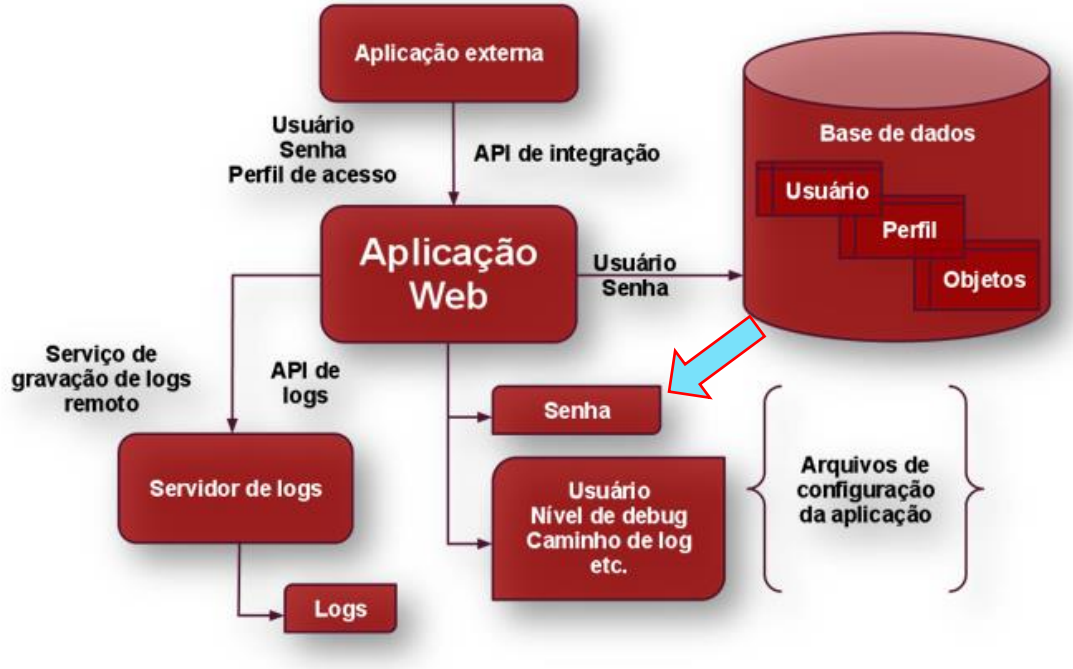
Apenas como nota: Perceba que no exemplo anterior falamos de um CAPTCHA que aparece apenas em momentos de "estresse" da aplicação, ou seja, a discussão deixa de ser se a aplicação deve ou não ter um CAPTCHA e passa a ser se ele está "ativado" ou não, pois os componentes de resposta a incidentes foram sensibilizados pelos logs.



Arcabouço de segurança

Cofre de senhas

A prática de armazenar as credenciais de acesso a bancos de dados no código ou em arquivos de configuração não são capazes de suportar os ataques simplórios de localização desses segredos, portanto, o uso de cofres de senhas é extremamente necessário. Os segredos devem estar em componentes especializados para sua proteção e para viabilização de todo processo de controle de acesso, rotacionamento periódico e demais aspectos o ciclo de vida de uma chave.

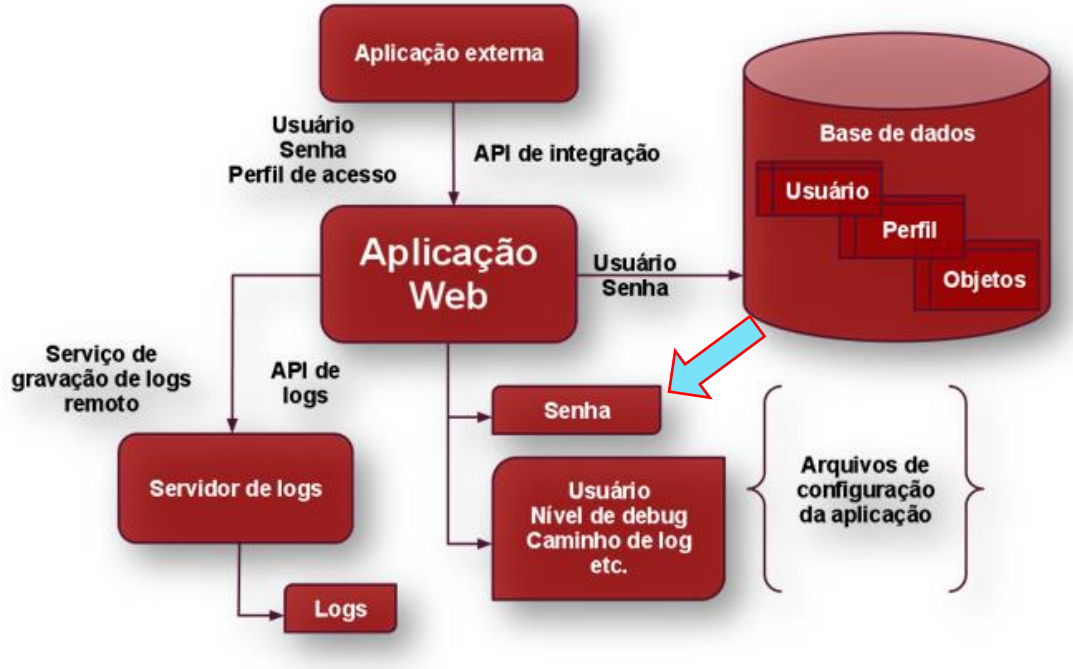


Arcabouço de segurança

Cofre de senhas

Senhas de acesso a APIs (ou mesmo aos SGBDs, chaves SSH, certificados e senhas para criptografia etc.), devem sofrer trocas periódicas. A aplicação deve suportar a troca do "segredo", criando uma nova senha a intervalos agressivos (por exemplo, 1 hora), quando possível.

Isso torna o vazamento de uma dessas credenciais quase que irrelevante. Adotando esse processo, nem o DBA possuirá a senha da aplicação por muito tempo, diminuindo ainda mais os riscos de acessos indevidos.

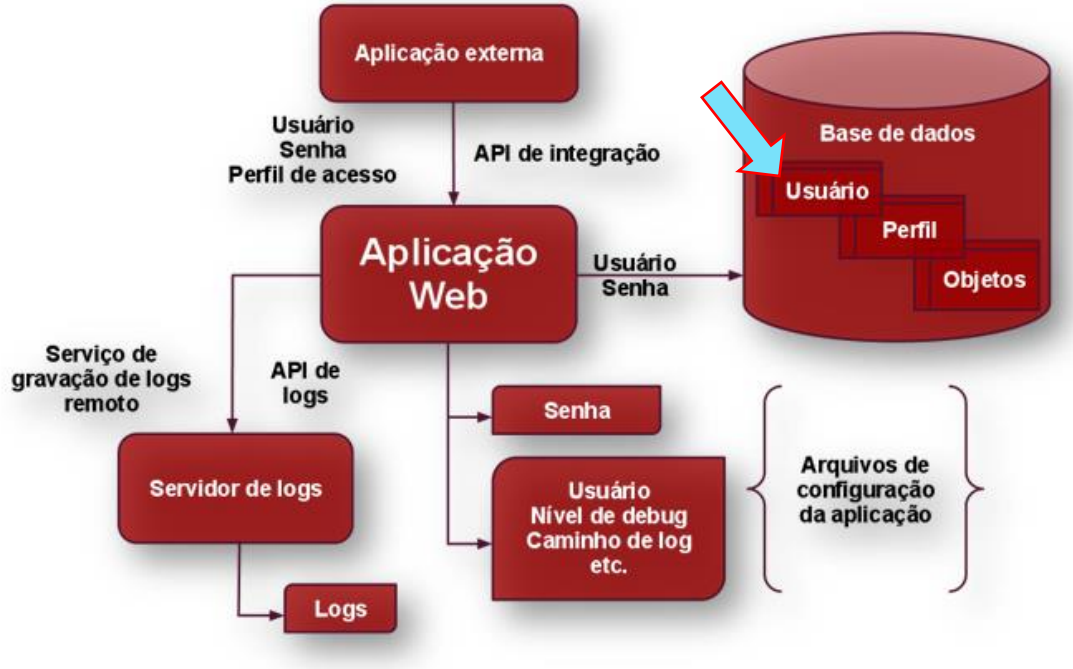


Arcabouço de segurança

Componente de controle de acesso

O principal engano é projetar o controle de acesso como um mero "usuário e senha" e esquecer do ciclo de vida das contas, quem as cria, como elas são recuperadas, quem tem autoridade para dizer que uma conta existe ou não, como identificar tentativas de quebra do controle de acesso, como se manter seguro, mesmo quando senhas e bases de hashes mundo a fora vazam etc.

Prover o serviço de controle de acesso de maneira segmentada é inexequível na prática (o custo seria impagável).

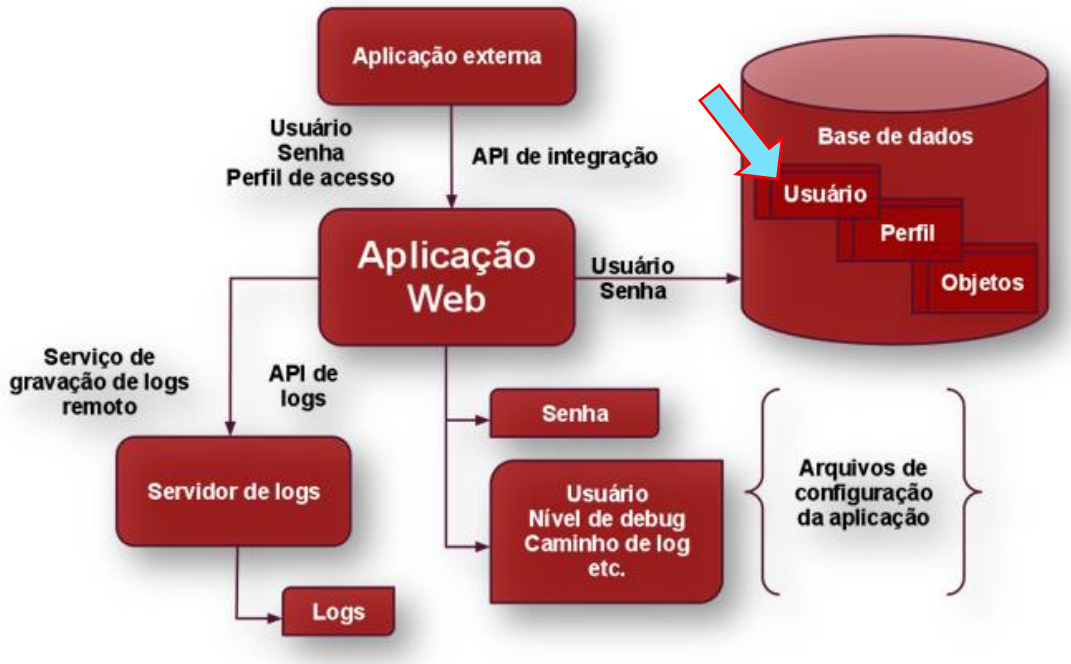


Arcabouço de segurança

O controle de acesso é um sistema e a melhor solução arquitetural é usar um sistema dedicado e integrar as diversas aplicações e um único sistema de controle de acesso altamente protegido e monitorado.

Esse sistema de controle de acesso deve prover:

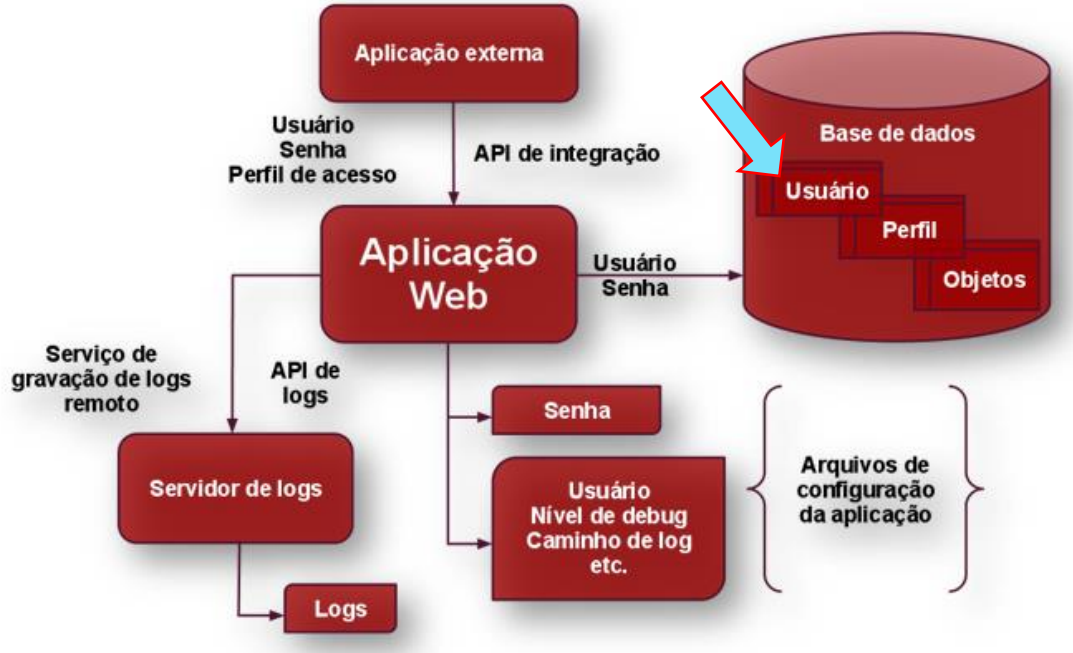
1. Solicitação de acesso;
2. Aprovação/reprovação dos acessos;
3. Revisão periódica de acessos;
4. Recuperação de contas.
5. Identificação e resposta a tentativas de quebra do controle de acesso (automatizar ações de contenção).



Arcabouço de segurança

Às aplicações, resta o controle de autorização em si, cabendo a aplicação mapear privilégios aos objetos do sistema que aquele usuário (papel) deve possuir.

Essa parte do problema é melhor endereçada com o controle de privilégios feitos nas aplicações, devidamente auditadas através de testes diretos (revisão de código ou pentest) ao controle de acesso, com o apoio de processos de SoD (Segregation of Duty) para alertar discrepâncias que viabilizem fraudes nos sistemas por um perfil ter "privilégios" demais em uma aplicação.



Arquitetura e segurança

Conclusão

Tratando-se de segurança e arquitetura, muitos tópicos se inter-relacionam (controle de acesso e monitoração) e recai, sobre as decisões arquiteturais, a obrigação de estruturar os vários componentes de uma solução para que a aplicação seja de fato segura. Não por si mesma, mas por todo o ecossistema de componentes devidamente estruturados, no qual um componente protege o outro, garantindo assim a segurança efetiva em camadas.