1.Segurança de aplicação

Como profissionais de TI, muitas vezes somos responsáveis por instalar **patches de segurança** em aplicativos que apresentam vulnerabilidades como **buffer overflow**, **injeção de SQL** ou outros tipos de falhas. O processo de criação de um aplicativo do zero é desafiador, e geralmente há um equilíbrio entre a velocidade de desenvolvimento e a segurança do aplicativo.

Muitos desses problemas no código do aplicativo são encontrados durante a **garantia de qualidade (QA - Quality Assurance)**. Esse processo de testes verifica não apenas a funcionalidade do software, mas também executa uma série de verificações de segurança. No entanto, se essas vulnerabilidades não forem detectadas nessa fase, há uma grande chance de que pesquisadores ou invasores as encontrem e, em alguns casos, consigam explorá-las.

Os desenvolvedores de aplicativos frequentemente utilizam **validação de entrada de dados** para garantir que qualquer informação inesperada inserida no sistema não seja interpretada incorretamente. Como há muitas maneiras de inserir dados em um aplicativo, seja por formulários, campos específicos ou textos livres, cabe ao desenvolvedor analisar e garantir que os dados inseridos estejam no formato esperado.

Por exemplo, se um campo solicita um **CEP**, ele deve aceitar apenas um número específico de caracteres e seguir o formato adequado. Se alguém inserir um valor que não segue esse padrão, o aplicativo deve detectar essa entrada inválida e solicitar ao usuário que corrija o erro.

Existe um processo automatizado para testar a segurança da entrada de dados em aplicativos, chamado fuzzing. Ferramentas de fuzzing inserem dados aleatórios nos campos do aplicativo para verificar como ele responde. Se o aplicativo se comportar de maneira inesperada, os desenvolvedores precisam revisar e corrigir a forma como a validação de entrada está sendo feita.

Se você já precisou solucionar problemas em navegadores da web, provavelmente já ouviu falar de **cookies**. Cookies são pequenos fragmentos de informação armazenados no navegador, usados para rastreamento de sites, personalização de conteúdo e gerenciamento de sessões de login.

Os cookies em si são apenas arquivos de dados — eles **não são executáveis** e **não contêm malware**. No entanto, as informações armazenadas dentro de um cookie podem ser valiosas para um invasor. Por essa razão, muitos navegadores utilizam **cookies seguros**, que só podem ser transmitidos por meio de conexões **HTTPS** criptografadas. Como os cookies não possuem segurança embutida, qualquer dado armazenado neles pode ser facilmente lido. Portanto, os desenvolvedores devem evitar armazenar informações sensíveis dentro dos cookies.

Para testar a segurança de um aplicativo antes de seu lançamento, os desenvolvedores podem usar um processo chamado **análise de código estática**, também conhecido como **SAST (Static Application Security Testing)**. O código do aplicativo é inserido em um analisador estático, que verifica automaticamente se há vulnerabilidades como **buffer overflow**, **injeções de banco de dados** e outras falhas.

No entanto, essa análise não é perfeita. Algumas vulnerabilidades de segurança não podem ser detectadas apenas observando o código-fonte. Por exemplo, um sistema pode incluir **criptografia**, mas se a implementação da criptografia for inadequada, pode abrir brechas de segurança que não serão detectadas por um analisador estático.

Além disso, os resultados da análise podem conter **falsos positivos**, então um desenvolvedor precisa revisar cuidadosamente os alertas gerados para determinar se são realmente problemas.

Aqui está um exemplo de saída de um **analisador de código estático**:

- Ele pode indicar que, no arquivo **filetest.c**, linha **32**, há um problema com a função **gets()**, pois ela **não verifica estouro de buffer**.
- O analisador pode recomendar o uso de **fgets()** em vez de **gets()** para evitar esse problema.
- O desenvolvedor, então, deve revisar todo o código e aplicar as correções recomendadas antes que o aplicativo seja distribuído.

Cada vez que instalamos um aplicativo em nosso sistema, há o risco de que ele contenha **malware** embutido. Para garantir que o software que estamos instalando é seguro e não foi alterado desde sua criação, usamos um processo chamado **assinatura de código**. A **assinatura de código** responde a duas perguntas essenciais:

1. **O aplicativo foi modificado desde que saiu do desenvolvedor?**
2. **Esse aplicativo realmente veio do desenvolvedor original?**

Para responder a essas perguntas, os desenvolvedores assinam digitalmente seus aplicativos antes de distribuí-los. Esse processo utiliza **criptografia assimétrica**, onde uma **autoridade certificadora (CA)** assina a chave do desenvolvedor, permitindo que ele autentique seu código.

Quando um usuário instala o aplicativo, o sistema operacional verifica essa assinatura digital. Se algo estiver errado com a validação da assinatura, o sistema exibirá um alerta informando que o aplicativo pode ter sido modificado.

Outro recurso importante para segurança de aplicativos é o uso de **sandboxing**. Quando um aplicativo é executado dentro de um **sandbox**, ele só tem acesso aos dados e recursos necessários para seu funcionamento, sem comprometer outras partes do sistema.

O **sandboxing** pode ser aplicado em diferentes cenários:

- Durante o **desenvolvimento**, os programadores trabalham em um ambiente isolado para evitar que o código afete sistemas de produção.
- Em sistemas operacionais, como em **máquinas virtuais**, onde cada VM é separada das demais.
- Em **dispositivos móveis**, onde o sistema operacional isola aplicativos uns dos outros para evitar que um app malicioso acesse dados sensíveis de outros aplicativos.

Por exemplo, em um **smartphone**, um navegador pode acessar seus favoritos salvos, mas, por padrão, **não** pode acessar suas fotos ou mensagens. Isso impede que um invasor, explorando uma falha do navegador, roube informações privadas armazenadas no telefone.

Muitos desenvolvedores também implementam **monitoramento de segurança** dentro dos aplicativos, permitindo rastrear tentativas de ataques, como **injeção de SQL** ou exploração de vulnerabilidades conhecidas. Esse monitoramento gera **logs detalhados**, que podem ser analisados para detectar possíveis ameaças antes que causem danos significativos.

Se algo incomum acontecer no aplicativo — como um **aumento repentino no tráfego de dados** ou **tentativas de acesso anormais** —, essas atividades serão registradas no monitoramento de segurança, ajudando a identificar e responder rapidamente a possíveis ataques.