1.Escaneamento de vulnerabilidades

As varreduras de vulnerabilidades são utilizadas para determinar se um sistema pode estar suscetível a um tipo de ataque. Esse processo não executa um ataque real no sistema operacional; caso contrário, ele seria um teste de penetração. Em vez disso, a varredura verifica se há possibilidade de um ataque explorando uma vulnerabilidade específica em um sistema. Um exemplo simples de varredura de vulnerabilidade é um escaneamento de portas, que identifica quais portas estão abertas e quais estão fechadas em um sistema. O fato de uma porta estar aberta não significa necessariamente que ela seja vulnerável a um ataque, mas indica um possível ponto de entrada que pode ser explorado por um invasor.

As varreduras de vulnerabilidades são utilizadas para encontrar possíveis brechas antes que os atacantes as descubram. Os motores dessas ferramentas são altamente eficientes para identificar todos os sistemas existentes dentro de um determinado **subnet IP**. Embora muitas vezes associemos essas varreduras a invasores externos tentando acessar dispositivos internos, também é fundamental realizar escaneamentos dentro da própria rede, pois um ataque pode ser conduzido por um agente interno.

Uma das dificuldades comuns ao lidar com varreduras de vulnerabilidades é a grande quantidade de dados gerados. Nem todas as informações encontradas são precisas, e muitas podem ser **falsos positivos**, o que exige uma análise detalhada dos relatórios para determinar quais vulnerabilidades são reais e quais podem ser descartadas.

Ao visualizar os relatórios de uma varredura, é possível notar que as vulnerabilidades são categorizadas por gravidade. Algumas são consideradas **críticas**, enquanto outras são classificadas como **médias, baixas** ou apenas **informativas**. Um exemplo de vulnerabilidade crítica detectada em um escaneamento seria uma falha em um gerador de números aleatórios do OpenSSL, que resultaria em **chaves SSH fracas**. Essa vulnerabilidade ocorre devido a um problema no gerador de números aleatórios da biblioteca OpenSSL, o que a torna um alvo prioritário para correção.

Outra vulnerabilidade comum é a detecção de um sistema operacional **Unix sem suporte**, indicando que esse sistema não recebe mais atualizações de segurança e pode estar exposto a ataques. Esses tipos de vulnerabilidades exigem atenção imediata, pois representam riscos elevados para a organização.

Todos os problemas listados nos relatórios devem ser verificados antes que qualquer alteração seja feita no sistema. Se as vulnerabilidades forem confirmadas, especialmente as de maior gravidade, elas devem ser corrigidas por meio do **processo de controle de mudanças**, garantindo que o sistema ou as aplicações sejam devidamente atualizadas.

Os desenvolvedores de software também podem verificar vulnerabilidades em seus códigos por meio de **análise estática de segurança de aplicações (SAST - Static Application Security Testing)**. Esse tipo de software examina o código-fonte de um aplicativo para identificar possíveis falhas, como estouro de buffer, injeção de banco de dados e outras vulnerabilidades que podem ser detectadas diretamente no código.

Embora os analisadores de código estático sejam bastante eficazes para encontrar problemas evidentes no código, algumas vulnerabilidades podem passar despercebidas. Isso ocorre porque um analisador de código não consegue entender completamente **como determinadas tecnologias foram implementadas**. Por exemplo, uma falha na implementação de autenticação ou um erro na configuração de criptografia pode não ser detectado apenas analisando o código-fonte.

Assim como ocorre com as varreduras de vulnerabilidades de rede, os resultados dos analisadores de código também precisam ser revisados cuidadosamente para identificar falsos positivos. No relatório gerado pelo analisador, é necessário distinguir quais recomendações são realmente aplicáveis ao código e quais podem ser ignoradas.

Além da análise estática, os desenvolvedores também podem realizar **análises dinâmicas**, um processo conhecido como **fuzzing**. Esse método consiste em inserir dados aleatórios em um aplicativo para observar sua resposta. O objetivo do fuzzing é fazer com que o aplicativo se comporte de maneira inesperada, como travamentos, mensagens de erro ou falhas de exceção. Essas reações podem indicar possíveis vulnerabilidades na forma como o código lida com a entrada de dados.

O fuzzing também pode ser chamado de **injeção de falhas, testes de robustez** ou outros termos semelhantes. Essa análise dinâmica pode revelar falhas que não seriam detectadas apenas pela inspeção do código-fonte.

A técnica de fuzzing não é recente. Um dos primeiros fuzzers foi criado em 1988 como parte de um projeto acadêmico na **Universidade de Wisconsin**. O professor Barton Miller e seus alunos desenvolveram o **Fuzz Generator**, uma ferramenta para testar a confiabilidade de programas de sistemas operacionais. Desde então, diversos motores de fuzzing foram criados para diferentes sistemas e linguagens de programação.

Os fuzzers modernos são quase sempre **automatizados**, pois precisam testar milhares de variações de entrada para identificar falhas. Se você quiser experimentar um motor de fuzzing, pode baixar o **Basic Fuzzing Framework (BFF)** da **Carnegie Mellon Computer Emergency Response Team (CERT)**, disponível no site **professormesser.link/bff**.

Durante a instalação de aplicativos em **Windows, macOS ou Linux**, os pacotes de software geralmente são fornecidos como um arquivo executável ou um conjunto

de arquivos compactados. Sempre que um aplicativo é instalado, é fundamental garantir que o software seja confiável.

O primeiro passo para verificar a confiabilidade de um pacote é confirmar se ele foi baixado diretamente do fabricante ou se veio de uma fonte terceirizada. Caso tenha sido obtido de terceiros, existe a possibilidade de que um invasor tenha modificado o pacote para incluir malware ou explorar uma vulnerabilidade.

Ao instalar um aplicativo que não foi devidamente verificado, o usuário pode estar introduzindo involuntariamente uma ameaça ao seu sistema. Se houver dúvidas sobre o conteúdo de um pacote de software, é recomendável testá-lo primeiro em um **ambiente isolado**, como um laboratório virtual, antes de implantá-lo em um ambiente de produção.

Uma vez que o software tenha sido devidamente analisado e considerado seguro, ele pode ser instalado com confiança no ambiente de produção, garantindo a segurança do sistema e dos dados da organização.