RELATÓRIO 6 – CLIENT SIDE EXPLOITATION – PARTE 3

Lucas Loscheider Reis Muniz – lucaslc01@hotmail.com

Path Transversal

Também conhecido como “directory traversal”,”dot-dot-slash”,”backtracking”, de acordo com o site OWASP.org, esse ataque objetiva acessar arquivos e diretórios que estejam armazenados fora da pasta raiz da web. Manipulando variáveis que referenciam arquivos com “../” e variações na URL por meio de requisições de página, é possível acessar arquivos e diretórios armazenados em um sistema. Podendo existir em diversas linguagens de programação, o atacante consegue também executar comandos e comprometer um sistema.

Para prevenir, é importante não armazenar arquivos importantes na pasta raiz web; para servidores windows, a pasta raiz web não deve ser colocada no disco do sistema, para prevenir transversal path; preferir trabalhar sem o input de usuário quando usar chamadas de arquivos de sistemas; usar índices em vez de nomes e localizações dos locais dos arquivos nas linguagems de programação para referenciar;

Server-side template injection

Segundo o portswigger, é quando um atacante é capaz de usar sintaxe nativa de um documento para injetar payloads malicioso no próprio template, de forma a concatenar inputs nos templates ao invéz de os passar ao servidor como informação. Com isso o atacante consegue executar código remoto por meio de payloads, tomando controle total do servidor e o usando para realizar ataques na infraestrutura interna, além de ganhar acesso a arquivos importantes.

A melhor forma de prevenir essa vulnerabilidade é não permitir que os usuários modifiquem ou submetam novos templates. Também usar ferramentas de template “logic-less” como por exemplo o Mustache, usado para HTML, arquivos de configuração e códigos fontes, o qual substitui clausulas e loops por tags. E por último executar o código dos usuários usando um ambiente sandbox(tecnologia que usa servers virtuais para testar códigos em ambientes isolados), por exemplo Lua, onde módulos e funções potencialmente perigosas foram removidos.

Information Disclosure

É quando um site não intencionalmente revela informação sigilosa para seus usuários, como dados de outros usuários, informações de comércio e detalhes técnicos da sua estrutura. Como consequência teremos pontos de ataque expostos para usuários maliciosos e perda de confiança no site. Também um atacante pode obter essas informações interagindo com o site de uma forma maliciosa.

Para prevenir isso, devemos analisar separadamente cada caso, já que existem variadas formas de ocorrer esse vazamento. Em vista disso, temos algumas recomendações gerais, como alertar aos desenvolvedores sobre as informações que podem ser sigilosas e que aparentemente não são, mas que poderiam contribuir para um ataque; usar mensagens de erro genéricas para não indicar aos atacantes pistas sobre o comportamento não esperado; sempre entender as configurações de tecnologia de terceiros usadas nos próprios sites.

Insecure Deserialization

De acordo com o site acunetix.com, é uma vulnerabilidade que ocorre quando data não confiável é usada para abusar da lógica da aplicação, executar ataque DoS (negação de serviço) ou até mesmo executar código arbitrário ao ser deserializado.
Na serialização, um processo converte um objeto em um formato o qual pode ser salvo em disco, ou enviado, seja pela internet ou pelo próprio computador. Essa serialização pode ser em binário, texto estruturado, como XML, JSON, sendo estes dois os mais comuns em aplicações web. Já na deserialização, é o oposto, ou seja, transformar data serializada, vinda de um arquivo ou da rede, em um objeto. Tendo isso em vista, podemos ter desenvolvedores supondo que usuários não saibam ler ou manipular uma informação justamente por ela estar serializada em binário, mas isso é errôneo, já que uma simples pesquisa online permite explorar esses objetos.
Para prevenir, é preciso evitar a deserialização do input de um usuário. Também implementar uma assinatura digital, por exemplo, antes do processo de deserialização para verificar que um dado não foi forjado.

OS Command Injection

É uma vulnerabilidade web que permite um atacante a executar comandos arbitrários de Sistema Operacional (OS) no servidor, o qual roda uma aplicação, comprometendo essa aplicação e todas suas informações. Esse ataque se difere do Code Injection, porque enquanto que no code injection o atacante adiciona seu próprio código que será executado pela aplicação, no Command Injection o atacante extende funcionalidades padrões da aplicação, as quais serão executados comandos do sistema, sem a necessidade de injetar um código.

Para se denfender podemos:

- evitar chamar comandos SO diretamente, utilizando funções de biblioteca embutidas que não podem ser manipuladas para realizar outras tarefas além das que se pretende fazer;
- As aplicações devem rodar usando os mais baixos previlégios necessários para atingir suas finalidades.

High-level logic vulnerability

São vulnerabilidades as quais se aproveitam de formas de usar um fluxo de processamento legítimo de uma aplicação, causando resultados negativos para a organização. Por exemplo, adicionando um produto no carrinho de compras e interceptando essa requisição online pelo BurpSuite, para alterar o preco do produto e sua quantidade, e assim conseguir finalizar a compra com créditos restantes no site.

Para se prevenir:

- O desenvolvedor nunca pode confiar nos usuários. A partir disso, deve-se criar uma aplicação a qual considere se acesso em outros meios além do browser;
- Qualquer tipo de validação deve acontecer pelo servidor, e não pelo cliente;
- Manter boas práticas de programação, com comentários e explicações do código.

Laboratórios

De tods os laboratórios propostos, não fiz apenas o laboratório “Lab: 2FA broken logic ” porque na parte de força bruta pelo burp para identificar o código de autenticação de 2 fatores, ele estava demorando bastante, levando em consideração o intervalo escolhido por mim de 0000- 9999, para encontrar a senha correta. Estava levando + de 2 horas para achar. Então acabei desistindo de fazer essa prática. Em vista disso, aparecia uma mensagem, antes de iniciar a força bruta pelo burp, de que a versão gratuita apresentava uma certa limitação no número de tentativaz por vez de executar o ataque de força bruta.

O resto dos laboratórios podem ser encontrados abaixo:

Basic server side template injection code context: https://youtu.be/Ujuf5rWl4hI

File path traversal, traversal sequences stripped non recursively: https://youtu.be/dpnpLYPN1pc

File path traversal, traversal sequences stripped with superfluous URL decode: https://youtu.be/2ASxkLI7x7w

Information disclosure in error messages: https://youtu.be/DMLvC6Oqr48

Lab: High level logic vulnerability: https://youtu.be/ElgAZIz3-ZI

Modifying serialized objects: https://youtu.be/0Sm2ymTYt7Y

OS command injection, simple case: https://youtu.be/9KMebF-ZY8Q