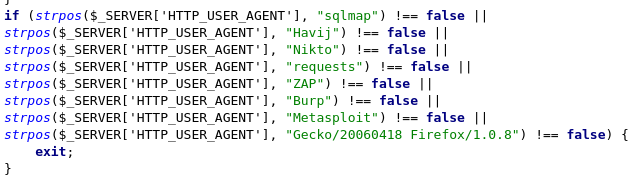
Mecanismos de confinamento encontrados

No ficheiro header.php aparece o seguinte código PHP:



Ora, como se pode observar, este código PHP previne que, se alguma ferramenta para testes de penetração e exposição de vulnerabilidades for utilizada, o sistema dá “exit”, impossibilitando essa mesma análise. Assim, diria-se que este é um dos métodos de confinamento utilizados.

Vulnerabilidades encontradas

1. Campos de login vulneráveis a SQL Injection e falta de verificações

Tentativa 1

Em primeiro lugar, o atacante tentou colocar uma pelica no campo de usermail do login, e apareceu o seguinte erro:

DB Error, could not query the database

MySQL Error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''''' at line 1

Ora, isto significa que o campo de login é vulnerável a SQL Injection!

O erro tem 5 pelicas seguidas, e podemos interpretar desta forma: ''' do campo do usermail, ou seja, início de string, 0 chars, fim de string, e início de string (sem fim); depois ainda tens mais 2 pelicas provenientes do campo da password (vazia) - 3 + 2 = 5.

Analisando igualmente o ficheiro login.php, em que ele primeiro verifica na base de dados se existe um username igual ao usermail introduzido, deduz-se que, por não haver um usermail igual a “vazio”, tenha então aparecido o erro no whireshark.

Tentativa 2

Desta vez, o atacante tentou colocar ' -- // no input reservado ao usermail; desta vez não deu nenhum erro de DB ou MySQL mas apareceu uma mensagem de “Invalid username, try again”.

Neste caso, ele ao colocar um comentário vai anular a componente do query destinada à password, pelo que é compreensível que o erro só se queixe do username/usermail; com a pelica introduzida no início ele fechou o query do usermail, e recebeu como "input" 0 chars tal como na primeira tentativa, mas como também foi colocado comentário na palavra-passe e por isso o campo da password foi ignorado e o query do usermail foi "fechado", então não houve erro de SQL Syntax.

O erro de username inválido (e não de sintaxe) deve ter aparecido por causa das verificações de html que são feitas do lado do cliente – o campo do usermail é do tipo “email”, logo é óbvio que um email igual a “vazio” é um email inválido.

Tentativa 3

Agora, ele tentou colocar no usermail ' OR 1=1 -- // e conseguiu entrar no sistema como admin, mesmo sem nenhuma palavra-passe.

Tal como na tentativa anterior, ele comentou o campo da password e, por isso, a parte do query destinada à palavra-passe vai ser novamente ignorada.

Para obter "dados", isto é, uma correspondência de uma conta na base de dados, alguma parte na query tem de dar valor lógico verdadeiro obrigatoriamente! ora, nada melhor do que garantir uma disjunção com uma condição sempre verdadeira... apesar do usermail introduzido ser "vazio", a condição "vazio" OR True vai dar True, possibilitando assim a entrada no sistema.

Tentativa 4

Desta vez, o indivíduo voltou a entrar na conta de administrador, mas sem recorrer a SQL Injection – ele recorreu a um email válido e conhecido (este está exposto numa das páginas do sistema) e a uma palavra-passe “típica” e “básica”, “abc”; assim, tudo indica que na tentativa anterior, e após ter entrado na conta e ser redirecionado para uma página onde pode criar novos posts e mudar as credenciais de acesso (nomeadamente a password – ver GET no whireshark), este tenha mudado a palavra-passe.

Tentativa 5

Bastante idêntica à tentativa anterior, diria-se que ele próprio voltou a mudar a palavra-passe para ser vazia, isto porquê:

- se fosse apenas ele recorrer a SQL injection, ao colocar uma pelica significava que ele ia estar à procura de uma conta com palavra-passe "vazia", o que é deveras impossível em condições ditas normais existir na base de dados uma conta com palavra-passe vazia;

- ele também não iria conseguir aceder à conta ao colocar na palavra-passe apenas uma pelica ' e colocando o usermail real, porque no ficheiro de login.php, a query do usermail e password é um AND, ou seja, ambos os literais têm de ser verdadeiros; ia dar verdadeiro que ele encontrou na base de dados um utilizador com aquele e-mail (até porque se ele não tivesse encontrado, ia dar erro de DB Error), mas como não existe uma palavra-passe vazia, ou melhor, a palavra-passe daquele usermail não era vazia, iria dar certamente "erro de password inválida" ou coisa do género.

Assim, deduz-se que ele voltou a mudar a palavra-passe, mas desta vez para “vazio”, evidenciando aqui outra vulnerabilidade – falta de verificações no input para a mudança da credencial de palavra-passe (neste caso, teria de verificar no mínimo se o conteúdo do input seria diferente de “vazio”).

Restantes tentativas

Nas restantes tentativas relativas à autenticação do atacante, ele recorreu sempre a um usermail real e no campo da palavra-passe voltou a recorrer a SQL Injection. Como já foi referido, após verificar que o usermail existe na base de dados, o query executado passa a ser:

$sql = "SELECT session FROM tblMembers WHERE username='" . $\_POST['usermail'] . "' AND password='" . $\_POST['password'] . "';";

Como é um AND, ambas as condições têm de ser verdadeiras; já sabemos que existe um usermail igual ao que foi introduzido, até porque ele é público e relativamente ao campo destinado à palavra-passe, ao colocar pelica ' ele está a "fechar" o campo da procura pela palavra-passe; mas como logo a seguir ele introduz uma condição verdadeira, a secção do SELECT destinada à password também vai dar verdadeira, possibilitando um reorno favorável da query e, assim, permitindo ao atacante entrar (de novo) na conta de administrador do sistema.