A Mythril implementa cada verificação em módulos específicos localizados na pasta `analysis/module/modules`. Cada módulo registra *hooks* para instruções EVM relevantes (como os opcodes **JUMP**, **DELEGATECALL**, etc.) e inspeciona o estado simbólico nessas instruções. A lógica central está organizada nos seguintes pontos:

- **ArbitraryJump** (*arbitrary_jump.py*):
  Escuta chamadas a **JUMP/JUMPI**. Verifica se o destino pode ser controlado externamente, permitindo saltos arbitrários.

- **ArbitraryStorage** (*arbitrary_write.py*):
  Observa **SSTORE**. Checa se os parâmetros (endereço de armazenamento e dados) podem vir de inputs externos, indicando escrita arbitrária no armazenamento.

- **ArbitraryDelegateCall** (*delegatecall.py*):
  Analisa **DELEGATECALL**. Verifica se o endereço-alvo e os argumentos podem vir de uma fonte controlável, configurando execução arbitrária.

- **TxOrigin** (*dependence_on_origin.py*):
  Captura instruções que usam **ORIGIN**. Avalia se há dependência perigosa em **tx.origin** na lógica de acesso.

- **PredictableVariables** (*predictable_variables.py*):
  Geralmente registra *hooks* em instruções que acessam **block.number**, **block.timestamp**, etc. Testa se decisões dependem de valores fáceis de manipular.

- **Outros módulos**:
  Arquivos como *unprotected_selfdestruct.py*, *unprotected_ether_withdrawal.py*, *multiple_sends.py*, entre outros, seguem a mesma lógica. Eles registram *pre_hooks* ou *post_hooks* para opcodes relevantes (como **SELFDESTRUCT**, **CALL**) e verificam se as condições de acesso podem ser burladas.

O registro desses *hooks* é feito no **ModuleLoader** (*loader.py*) e despachado em tempo de execução simbólica (*security.py*), onde são percorridos todos os estados gerados pelo **LASER EVM** (*svm.py*).

Cada módulo, ao ser acionado, inspeciona as variáveis simbólicas (**constraints**, **global_state**) para detectar cenários de vulnerabilidade, como permissões, origens de parâmetros e caminhos de execução. Isso ocorre dentro dos métodos presentes em cada arquivo do diretório `analysis/module/modules`, que contêm as condições específicas para cada vulnerabilidade mencionada.

Ao todo, são **17 arquivos**, cada um dedicado a um tipo específico de vulnerabilidade.