PHP Loose Comparison & Type Jugging

11th May 2020

Loose comparison -- *$a == $b*

- String type compared with String type
- String type can be evaluated as number if it forms certain format
    - All decimal digits plus “.”, “e”, “E”
    - Formally [-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?
- To make two unequal variables return true
    - Let them to be evaluated as 0, e.g., $a = “0e123”, $b = “0e456”
    - Alternatively, $a = “10.000000”, $b = “00000001e1”; $a = “1”, $b = “00001e0”

Loose comparison -- *$a == $b (contd.)*

- String type compared with Number type
- String is implicitly converted into Number type
  1. For strings fitting int/float format: Directly evaluate as numbers
  2. For strings starting with a legitimate int/float format string and then following with a random string: Evaluate front legitimate numeric string and discard the rest, e.g., “123abc” is evaluated as 123
  3. Others: Evaluates as 0
- To make two unequal variables return true
  - “123” == 123
  - “123abc” == 123
  - “abc” == 0

In the scenario of authentication $a == $b

- $a and $b are mostly String type
- Password can be either encrypted or just plain text
- Password can be persisted in database or not

- Our threat model is defined as “code allows a wrong password to successfully authenticate”
- Are all authentication with loose comparison vulnerable?
  - YES, if we follow this threat model

Other than authentication

- Loose comparison in other variable types can be a potential problem
  - String v.s. String
  - String v.s. Number
  - String v.s. Bool
  - Etc.
- If we can observe implicit type conversion appears!
  - Inter-string comparison is actually performed as inter-number comparison
- Static analysis can locate them, but human experience is needed to verify