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 authentications with loose comparison vulnerable?
  - YES, if we follow this threat model
- In real world, are they exploitable as a black box attack?

Probability of exploitability

- Compute the how likely they can be exploited
  - In normal case, a 64-bit, attacker gets chance about 1/(2^64)
  - Under such setting, attacker might easily exploit, e.g., password start fom ”0e”
- Randomly generate password or choose from database
  - Check the distribution that is in our attack string format
  - Conditionally, under certain vulnerable setting, the possibility to exploit

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.
- Static analysis can locate them, but human experience is needed to verify
  - A Wordpress CVE bypassed **IF condition** to install third-party plugins, which eventually allowed remote code execution
- If we can observe implicit type conversion happens dynamically!
  - Inter-string comparison is actually performed as inter-number comparison
- Enhance PHP Zend engine to check to perceive variable types
  - Log runtime type information before and during loose comparison
  - Check and evaluate on CVEs the verify the system works to notify developer (might as a defense)

- Other languages if they have same problem of implicit type conversion.