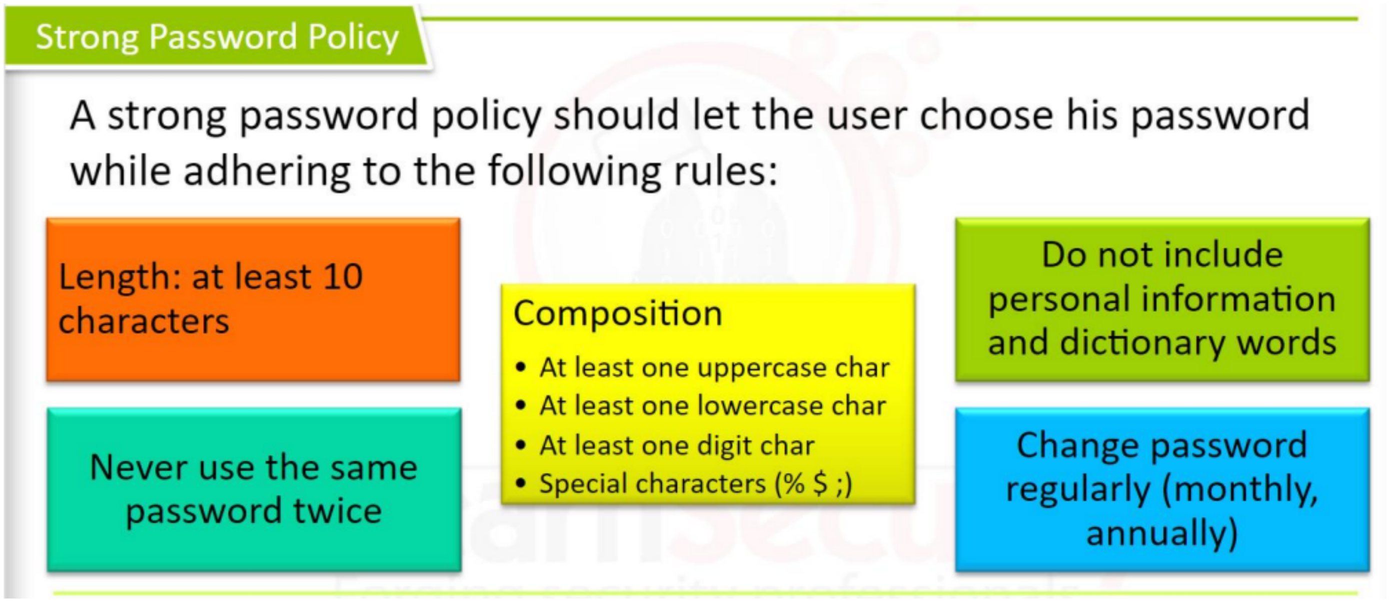
6. Module Authentication

Dictionary attacks

-> Weak Password policy
Mitigation:

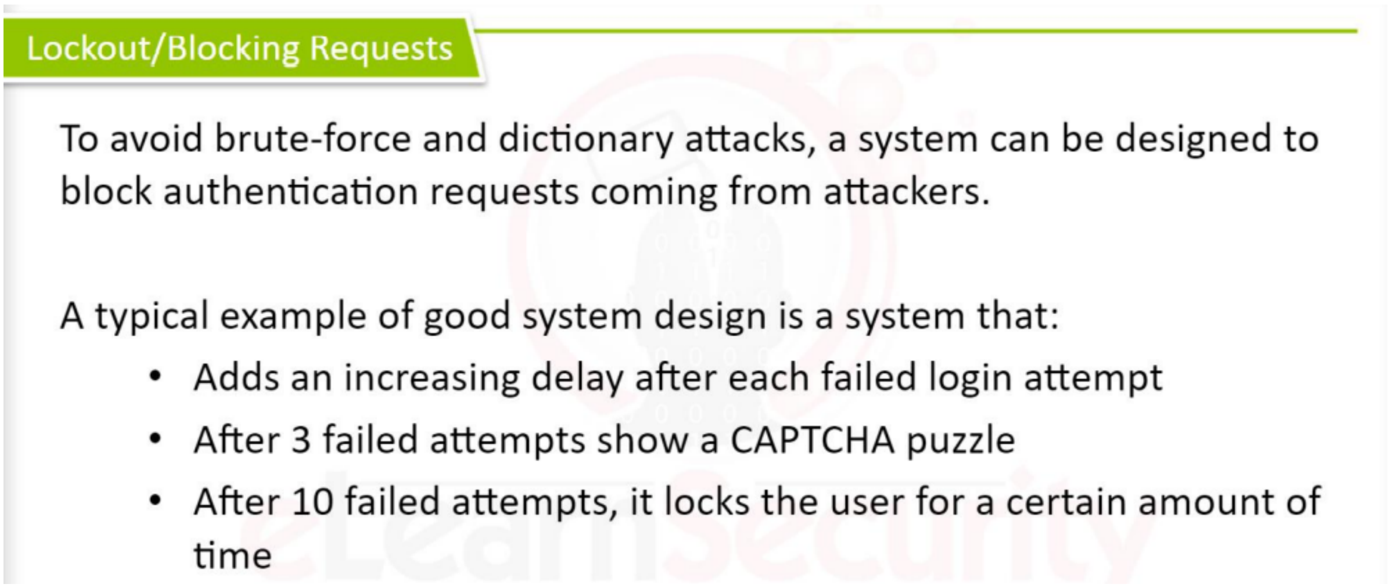


Strong Password Policy

A strong password policy should let the user choose his password while adhering to the following rules:

- Length: at least 10 characters
- Never use the same password twice
- Composition
  - At least one uppercase char
  - At least one lowercase char
  - At least one digit char
  - Special characters (% $;)
- Do not include personal information and dictionary words
- Change password regularly (monthly, annually)

-> password should not be stored in clear text



Lockout/Blocking Requests

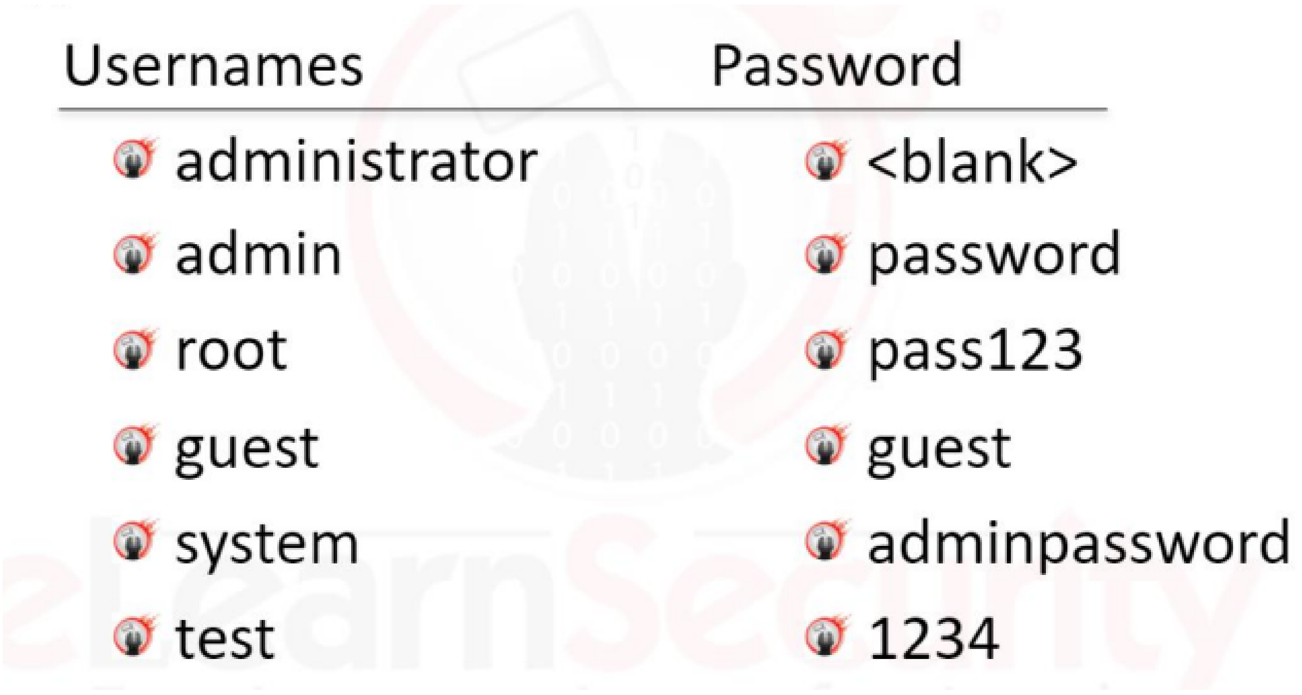
To avoid brute-force and dictionary attacks, a system can be designed to block authentication requests coming from attackers.

A typical example of good system design is a system that:

- Adds an increasing delay after each failed login attempt
- After 3 failed attempts show a CAPTCHA puzzle
- After 10 failed attempts, it locks the user for a certain amount of time

-> User enumeration Through errors from entering wrong credentials
Automation with burpsuite -> intruder

-> default credentials



Usernames	Password
administrator	<blank>
admin	password
root	pass123
guest	guest
system	adminpassword
test	1234

-> SessionID is predictable

Defense

-> Cache Browser Method Defense
Disable the autocomplete HTML attribute
`<input type="password" autocomplete="off"`

-> Cookie Method Defense
If the Cookie contains user credentials, the credentials have to be encrypted

-> Web Storage Method Defense
If the Web Storage contains user credentials, the credentials have to be encrypted

Password Reset function

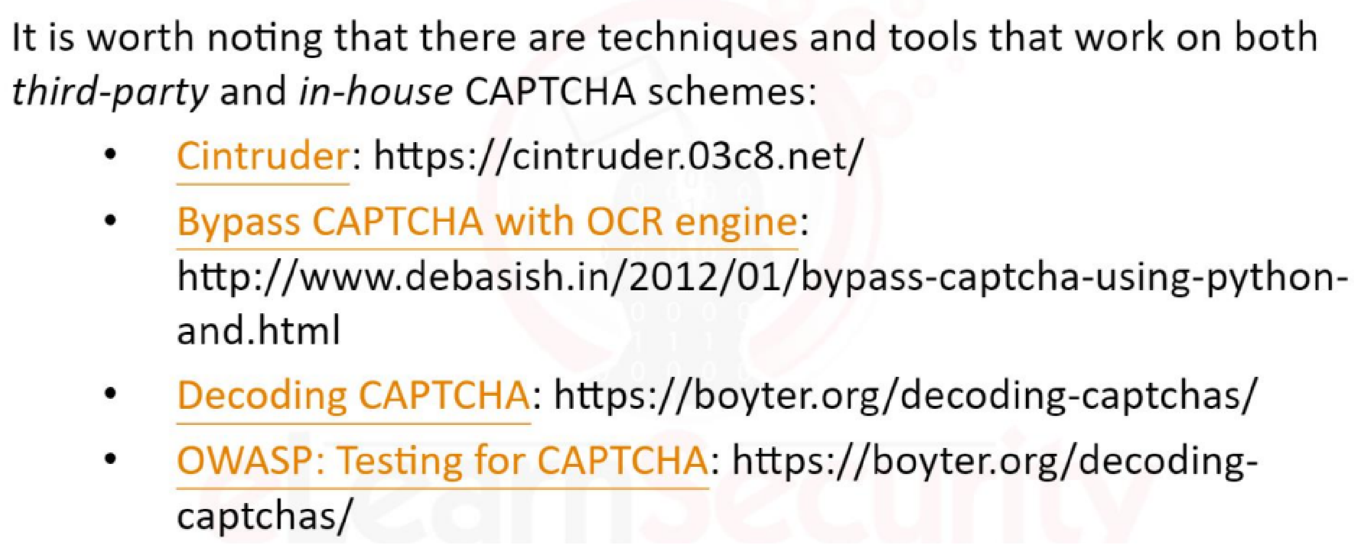
-> no rate limiting
-> Password Reset Link not expired
-> weak passwords are allowed
-> Password Reset Link is guessable

Logout Weakness

-> User logout but session still valid

Captcha

-> Implementing captcha means using third party code, which may lead to bypass authentication, XSS, SQLi



It is worth noting that there are techniques and tools that work on both *third-party* and *in-house* CAPTCHA schemes:

- Cintruder: https://cintruder.03c8.net/
- Bypass CAPTCHA with OCR engine: http://www.debasish.in/2012/01/bypass-captcha-using-python-and.html
- Decoding CAPTCHA: https://boyter.org/decoding-captchas/
- OWASP: Testing for CAPTCHA: https://boyter.org/decoding-captchas/

IDOR

Alreay known

Path Traversal

```php
<?php
$my_file = @$_GET['lang'] . '.html';
if (file_exists($my_file)){
    readfile($my_file);
}
```

According to the above code, the exploit should be `lang=/etc/passwd%00`. The %00 is a url encoded version of null character. it means an end of a string.