nuBuilder Security Considerations

nuBuilder Security

- Personal logins
- Permission roles
- Restrict access to forms, reports, php functions
- Users cannot run server-side scripts without permission
- Users cannot modify SQL queries

Further security measures

Among others, the following are also supported natively (or through documented implementation procedures):

- Restrict access to own records
- Create Password Policies
- Use 2FA
- Implement IP restrictions
- Log Login activities
- Log other activities
- Audit Table

General Open Source Software security considerations

The following is also true for **nuBuilder**.

WordPress example

- WordPress is open source an free too and powers nearly 40% of all websites on the internet.
- It has hundreds of thousands of theme and plugin combinations out there.
- It is not surprising that **vulnerabilities** exist and are constantly being **discovered**.
- However, WordPress usually gets a **bad rap** for being prone to security vulnerabilities and inherently not being a safe platform to use for a business.
- More often than not this is due to the fact that users keep following **industry-proven security worst-practices** like:
    - Using outdated WordPress software,
        - nulled plugins,
        - poor system administration, credentials management, and
        - lack of necessary Web and security knowledge.
    - Enabling lots of properties, themes, plugins and making it easier to use for end users without IT background, creates among others, lots of
        - gaps,
        - bugs,
        - vulnerabilities,
        - breaches

Dependent Applications Issues

- The Web server needs to be configured for security too.
- It must be running on a limited user account.

Take these with a **pinch of salt**, especially, as latest is not generally the greatest across major versions and many cutting edge fixes themselves could have serious bugs and can potentially break installations:

- Keep your "**PHP up to date**".
- Fix security-related updates in the past several years.
- The "system" as a whole needs to be continually updated.
- If it's a Linux system (any OS for that matter), it's important to "**keep everything up to date**", but especially the "**kernel**".

If it ain't broke, don't fix it may apply in some circumstances only.

Open Source takeover by Commercial entities

Taking over trusted Open Source Applications (PHP, MySQL after certain versions) has it's ramifications for the end user / sysadmin like:

- forcing users into their "latest" versions
- doctored benchmarks measuring a fish's ability to climb a tree
- hiding / removing fixed versions available before control
- forced update channel URLs to create a gradient to unknowlingly follw "the path"
- injection of "paid experts" to obfuscate with esoteric programming constructs
- hijacking unserstanability for the self-taught programmer

Those in management quickly learn the addage - "If you can't beat them, join them and then **scuttle them**".

Webserver's exposure calibration

Security countermeasures include:

- getting a valid SSL/TLS certificate,
- configuring TLS to use only strong cipher suites, and
- configuring **php.ini** for security

While there is a lot involved, the general mitigation principle is to

- disable dangerous PHP functions.

Open Source Application Preferences

- reliability
- proven continued functionality
- having a strong community