SECURING YOUR CONFIGURATION

Improper security configuration is one of the most commonly overlooked risks to your technology stack. If you leave your servers unsecured hackers can find vulnerable access points through simple Google searches.

RISKS

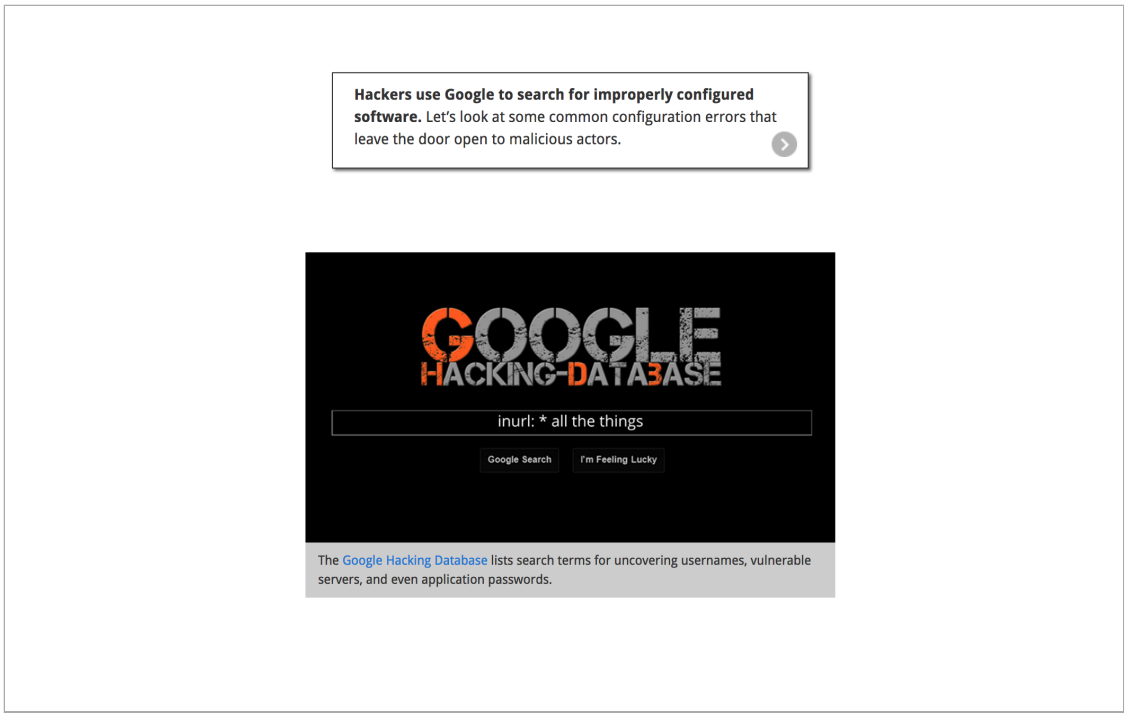
PREVALENCE **COMMON**

EXPLOITABILITY **EASY**

IMPACT **HARMFUL**

If an attacker can access your system via insecure configuration, they could:

- **Steal data.** Attackers frequently steal sensitive client data like email addresses, passwords or credit card numbers.
- **Infect your servers.** Compromised servers are often used to host spam-bots or other types of malware.
- **Abuse the trust your users have in your site.** If a hacker can serve content under your security certificate, they have a reliable way to infect others.



Want to see this vulnerability in action? Take our exercise. Learn how systems are typically left unsecured.

PROTECTION

Securing your software settings depends on understanding your software and enforcing best practices through good process management. You should:

- **Automate your build process.** An ad-hoc build processes makes it easy for insecure software settings to slip through. Make sure you have a scripted, repeatable build process, so you know what software (and what versions) you are running at any given time.
- **Review new software components and disable default credentials as soon as possible.** Each new library, toolkit and server introduces new security risks - ensure these are considered during your code reviews.
- **Clearly separate code and configuration.** Environment-specific and sensitive configuration should be stored outside the codebase - either in configuration files or dedicated systems (like databases). Hard-coded credentials and backdoors open your site to being compromised.
- **Create dedicated accounts with appropriate privileges.** Access to production servers and databases should follow *the principle of least privilege*. Users and processes should only have the permissions they require to function, and any escalation of privileges (for example, during release windows,) should be temporary and subject to formal review process.

- **Script your deployment process.** Make sure deployment to staging and production systems is done through a repeatable, scripted process. You should know what version of your code is running on each environment - and be able to vouch that each environment is running the appropriate configuration. After each release, perform (at least a cursory) “smoke-test” to ensure the correct software and configuration got deployed.
- **Segregate environments.** Production and staging environments should use different sets of credentials, since they will typicllay have different access levels. Try to ensure there is no network access between environments, so attackers cannot move sideways between environments with different access levels.
- **Add extra security for administrative systems.** If possible, avoid opening your administrative tools to the internet at large. Prescribe a secure password policy for administrators, and ensure your team knows to take security seriously. Implement multi-factor authentication if feasible. Make sure you know who has access to what system, and have a plan in place for when access has to be revoked (for instance, if team-members leave).

RELATED VULNERABILITIES

INFORMATION LEAKAGE PASSWORD MISMANAGEMENT