Web 开发人员必备的安全检查列表 【已翻译100%】

英文原文：Web Developer Security Checklist

标签： <无>

王练 推荐于 9个月前 (共 9 段, 翻译完成于 06-08) 评论 3

参与翻译 (4人) : LeoXu, Tocy, 总长, 无若

开发在云上面运行的安全可靠的 Web 应用程序是**非常非常困难**的。要是你觉得这很容易，那要么是你的生活方式更高级，不用管这破事儿，要么就是之前已经趟过许多坑了，对此已经了然于心了。

如果你了解MVP这档子事儿，并且确信自己可以在一个月之内倒腾出一个既实用又安全的产品 – 着手进行“原型产品”的开发工作之前，请三思。在你看过下面的清单之后，你要认识清楚自己跳过了这其中许多的安全问题。至少，你要对潜在用户说实话，让他们知道这个产品还不是完备的，它只是一个不完全安全的原型。

这个清单很简单，而且也绝非完整的。我已经开发安全 Web 应用程序超过了14年，这份列表包含了我在这段艰辛中认识到的一些比较重要的问题。希望你在创建 Web 应用程序时会认真地考虑一下这些问题。

如果你还有可以添加到这份列表中的问题，欢迎评论。

数据库

- 可以的话，就使用加密来保护同用户相关的敏感数据，比如访问令牌、电子邮件地址或账单（不过这做将对精确匹配查找的查询请求有限制）。
- 如果你的数据库支持闲时的低成本加密（比如 AWS Aurora），就可以用这个来保护磁盘上的数据。要确保所有备份都是被加密存储的。
- 对数据库的访问用户帐户要赋予尽可能小的权限。不要使用数据库的 root 帐户，并且对未使用的帐户以及密码不正确的帐户进行检查。
- 使用专为此目的设计的密钥存储来存储和分发机密文件。不要在你的应用程序中硬编码这些数据。
- 只使用预备型的 SQL 来完全防止 SQL 注入攻击。例如：如果可以使用 NPM 的话，就不要使用 npm-mysql ，要使用支持预备型语句的 npm-mysql2 。

开发

- 确保你的软件的所有组件在推送到生产环境之前都进行过漏洞扫描。这里意思是 O/S，库以及包级别的漏洞都要扫描。这种工作应该使其在CI-CD过程中自动化。
- 对于开发环境系统安全性的警惕程序应该等同于生产系统。安全方面的考虑，要在安全且隔离的开发系统中构建软件。

认证

- 要确保使用了适当的加密（如bcrypt）对所有密码进行了散列处理。不要自己实现加密算法，而且要用相对较好的随机数据来初始化加密程序。
- 使用经过验证的最佳实践来实现登录、忘记密码以及其他诸如密码重置这样的组件。不要自己来重复造轮子——因为很难保证其在所有的场景中都不会出现问题。
- 使用简单但是足够使用的密码规则来引导和鼓励用户使用随机的长密码。

- 为你提供的所有服务的登录功能都加上多因素身份验证机制。

拒绝服务攻击保护

- 确保针对你所开放的 API 的 DOS 攻击不会对你的网站造成太大的负面影响。至少要在较慢的 API 路径以及像登录和生成令牌这样的跟身份验证相关的 API 路由上设置限速器。 可以考虑为前端 API 加上 CAPTCHA（图形验证码）来保护后端服务免受 DOS 攻击。
- 对用户提交的数据和请求的大小和结构进行比较完备的限制。
- 考虑通过全局缓存代理服务（如 CloudFlare）来缓解分布式拒绝服务攻击（DDOS）造成的负面影响。如果你遭遇了 DDOS 攻击就可以打开此操作或者其它类似功能，来进行 DNS 查找。

LeoX

翻译于 9

0人顶

顶 翻译得不错哦！

网络流量

- 对整个网站使用 TLS ，而不仅仅是登录表单和响应。 切勿在登录表单中使用 TLS。 传统上，使用 strict-transport-security 头来强制所有请求的 HTTPS 。
- Cookie 必须是 httpOnly 和 secure 的，并且限定路径和域。
- 使用 CSP 而不允许 unsafe-* 后门。 这是配置很痛苦，但值得。 使用 CSP 子资源完整性作为 CDN 内容。
- 客户端响应使用 X-Frame-Option、X-XSS-Protection 头。
- 使用 HSTS 响应强制仅限 TLS 访问。 将所有 HTTP 请求重定向到服务器上的 HTTPS 作为备份。
- 使用所有形式的 CSRF 令牌，并使用新的 SameSite Cookie 响应头，一劳永逸地修复较新的浏览器。

总长

翻译于 9

0人顶

顶 翻译得不错哦！

APIs

- 确保在公开 API 中无枚举资源。
- 确保在使用 API 时，用户已经被完全认证和授权。
- 在 API 中使用随机检查来检测潜在攻击的非法或异常请求。

校验和编码

- 为了实现快速用户响应，在客户端完成输入校验，但不要过于信任。在显示之前始终对用户输入进行校验和编码。
- 使用服务器上的白名单校验用户输入的最后一位。不要直接将用户内容插入到响应中。切勿在 SQL 语句或其他服务器端逻辑中使用不受信任的用户输入数据。

Tocy

翻译于 9

0人顶

顶 翻译得不错哦！

云端配置

- 确定所有的服务都最小化的开放端口。尽管通过增加不确定性没有实际的保护作用，但使用非标准的端口，将会提升攻击的难度。
- 主机的后端数据库和服务放在私人的 VPC 上，那么就不能让这些 VPC 在任何公网中可见。要注意的是，当配置 AWS 的安全策略时，对等的 VPC 一不注意就会让其服务在公网中可见。
- 隔离的逻辑服务应该放在不同的 VPC 上，并且要提供对等的 VPC 内置的通讯服务。
- 确保所有服务接收的数据都只是来自于最少的那个 IP 地址的集合。
- 要约束出站 IP 和 端口流量，以最小化 APT 和“通知”。
- 经常使用 AWS IAM 角色而不是使用 root 认证。
- 所有维护和开发人员，都应该使用最小化的访问权限。

无若

翻译于 9

1人顶

顶 翻译得不错哦！

- 根据计划，定期变更密码和访问秘钥。

基础设施

LeoX
翻译于 9
0人顶

顶 翻译得不错哦！

- 确保无需停机就能进行更新升级操作，同时确保可以以完全自动化的方式快速地更新软件。
- 要使用 Terraform 等工具来创建所有的基础设施，而不是通过云端控制台来进行。基础设施应该被定义为“代码”，并且能够一键重建。对于在云中手动创建任何资源的行为做到零容忍 ——Terraform 可以在操作之前对你的配置进行审核。
- 所有的服务要使用集中性的日志记录。永远也不应该通过 SSH 来访问或者检索日志。
- 除了一次性诊断，不要使用 SSH 进入服务。通常如果使用了 SSH，就意味着还存在着没有自动化执行的重要任务。
- 不要让 AWS 服务组上的 22 端口保持永久的打开状态。如果必须要使用 SSH 的话，那就只能使用公钥身份来进行验证，而不是密码。
- 创建稳固的主机，而不是修补和升级长寿的服务器。 （稳固的基础设施可以更安全）
- 使用入侵检测系统来最大限度地减少 APT。

操作

LeoX
翻译于 9
1人顶

顶 翻译得不错哦！

- 关闭无用的服务和服务器。最安全的服务器就是关掉的服务器。

测试

- 对你的设计和实现进行审查。
- 进行渗透测试—就是自己黑自己的系统，当然也要有除自己以外的别的什么人来测试一下。

培训

- 就关于安全社会工程领域中的潜在威胁和相关防护技术对员工（特别是高级员工）进行培训。

最后，要有计划

- 要有一个对你正要防御的威胁模型的描述，应该在里面列出潜在的威胁和行为者，并标明优先次序。
- 要有一个具备可操作性的安全事件计划，因为总有那么一天你会需要这个的。

评论(3)

Ctrl/CMD+Enter 发表评论

Andyfoo 来自 iPhone 发表于 2017-06-09 09:16
不全

多厘 发表于 2017-06-09 09:24
数据库部分 "只使用预备型的 SQL 来完全防止 SQL 注入攻击" 这个预备型应该翻译为"预处理"