 DOI: https://doi.org/10.12349/iser.v7i8.11561

Analysis and Practical Implementation of Cybersecurity Enhancement Strategies for Open-Source CMS Information Systems

Bin Long

Civil Aviation Chongqing Air Traffic Control Division, Chongqing, 401120, China

Abstract

This paper focuses on researching security risks in the secondary development and deployment processes of open-source CMS information systems. From multiple dimensions such as the operating system, open-source system configuration, code improvement, and system deployment, the article systematically analyzes potential security risks that open-source CMS systems may face in terms of network security and proposes corresponding hardening strategies. Simultaneously, using the LIN CMS system as a example, it elaborates on various security vulnerabilities discovered during penetration testing, code auditing, and self-discovery in the secondary development process and provides specific solutions. This paper offers strong guidance value for the security architecture design of open-source CMS information systems and the reinforcement of network security at the system framework level during secondary development.

Keywords

open-source; CMS system; network security; security reinforcement; LIN CMS

开源 CMS 信息系统的网络安全加固策略分析与实战

龙滨

民航重庆空管分局，中国 · 重庆 401120

摘　要

本文针对开源CMS信息系统在二次开发和部署过程中的安全风险问题进行研究。文章从操作系统、开源系统配置、代码改进和系统部署等多个维度，系统性地分析了开源CMS系统在网络安全方面可能存在的潜在风险，并提出相应的加固策略。同时以LIN CMS系统作为实战样例，详细阐述了在二次开发过程中在渗透测试、代码审计及自行发现的各类安全漏洞，并提供了具体的解决方案。本文对开源CMS信息系统的安全架构设计和二次开发时系统框架层面对网络安全加固具有较强的指导价值。

关键词

开源；CMS系统；网络安全；安全加固；LIN CMS

1 引言

CMS 系统也称内容管理系统，通常采用 B/S 架构和前后端分离的技术。随着开源信息技术的发展，涌现了一大批的开源 CMS 项目。由于采用这些项目功能全面、改进持续，能极大地降低开发维护成本，这使得很多公司和单位倾向于选择基于开源 CMS 系统进行二次开发。笔者在为单位开发内外网信息系统时，曾对不同架构的开源 CMS 系统进行过研究和内核改进，也多次主持对所二次开发的 CMS 系统进行了渗透测试及代码审计整改，积累了对 CMS 系统加固的经验。由于对开源信息系统二次开发的系统直接进行部署风险非常高，必须采取一些加固措施。在本文中，笔者将结合自身的理解，以及渗透测试和代码审计主要内容，对 CMS 信息进行加固的一些基本策略进行讨论，提供对开源 CMS 系统的安全加固建议。并以开源 LIN CMS 系统为例，探讨实战安全加固需要注意的问题。

【作者简介】龙滨（1976–），男，中国重庆人，硕士，高级工程师，从事空中交通管理通信、导航、监视研究。

2 开源 CMS 信息系统的安全加固

2.1 CMS 信息安全加固原则

笔者在开源信息系统网络安全方面结合实际经验，总结了以下 5 项原则：

默认安全配置修改原则：应避免使用开源系统在安全方面的默认配置，如默认管理员账号密码、默认数据库、默认秘钥等。

最小权限原则：开源系统默认的访问权限可能超出了

实际需求，可能导致访问越权，从而导致系统被恶意攻击。在设置用户或角色权限时应按最小权限原则分配。

强加密原则：登录用户、数据库应采用强密码，登录密码应进行加密强度验证并采用强加密算法，传输应采用加密协议，如 HTTPS、WSS 等，防止暴力破解和传输过程中的数据泄露。

可追溯原则：记录系统的所有操作，便于审计和排查问题。应开启 CMS 日志记录，并在系统设计上对数据库记录的历史访问。

隐私保护原则：应保护用户的隐私信息（如身份证号、电话号码、住址等）不泄露。

2.2 CMS 信息系统的加固

对开源 CMS 信息系统的安全加固不是孤立的，需要从 5 个层面，也就是网络安全防护设施层面、操作系统层面、开源 CMS 系统的安全配置层面、二次开发网络安全设计层面以及系统部署层面综合采取进行加固措施，形成组合加固策略，才能形成坚固的安全措施。



图 1 CMS 系统网络安全防护 5 个层面

2.2.1 网络安全防护设施加固

用户可根据实际情况安装防火墙、网络安全流量监测、日志审计、堡垒机、零信任、蜜罐等安全防护系统。本文主要对 CMS 系统加固进行论述，不做详述。

2.2.2 操作系统层面的安全加固

CMS 信息所部署的操作系统应避免采用默认账号，系统采用强密码，做好文件访问控制，加强用户分组权限控制，及时更新补丁，合理配置系统防火墙，安装病毒防护、木马防护软件等措施。本文不做详述。

2.2.3 开源 CMS 系统加固

对开源 CMS 系统加固，主要考虑两个方面，一是默认安全配置的加固，二是在进行二次开发设计的加固。

开源 CMS 系统的默认配置可能存在安全风险，比如系统默认超级用户名称、默认登录密码、默认数据库名称、默认数据库账号和密码、默认秘钥、默认用户、默认系统名称和路径、默认端口、token 默认过期时间、默认搜索关键字等。这些配置在部署前需要修改加固，一方面防止恶意用户通过猜测或暴力破解方式进行攻击，另一方面防止恶意用户通过研究开源系统的默认配置，从而导致系统被攻击。

以 LIN CMS 系统 Koa 后端为例，其默认配置如下：

表 1 LIN CMS 系统后端默认配置

项目	默认配置	修改建议
默认超级用户	root	更换超级用户名称
默认访客	guest	取消访客
默认用户登录密码	123456	强密码
默认数据库	lin-cms	自定义数据库名
默认数据库端口	3306	修改为其他端口
默认秘钥	此处省略	修改秘钥
默认 access token 过期时间	60*60 秒	可适当减少
默认 refresh token 刷新时间	60*60*24*30 秒	1 个月太长，须减少
默认后端端口	5000	修改为其他端口

对于 LIN CMS 系统 Vue 前端，其默认配置如下：

表 2 LIN CMS 系统后端默认配置

项目	默认配置	修改建议
前端默认端口	8080	更换端口
操作停滞时间 stagnateTime	60 * 60 * 1000	可减小

上述用户中，需要 root 和 guest 作为用户名，采用 123456 弱密码作为默认密码，容易被暴力破解。对系统端口和数据库端口进行修改，可以提升恶意用户猜测和发现难度；对默认秘钥进行重新设置，可以防止恶意用户利用开源秘钥对 token 进行解密；对 token 刷新和操作停滞过期时间进行合理设置，可以减少暴力破解窗口时间。

2.2.4 二次开发设计加固

二次开发设计加固策略：

开源 CMS 系统的发布方通常为提升评价和测试部署便利，会忽略掉一些安全方面的设计，需要用户在二次开发或正式部署时进行进行代码级加固。代码级加固方面需要考虑以下因素：

（1）是否采用默认采用未加密的传输协议（如：http、ws 等）。采用未加密的传输协议使得数据采用明文传输，存在网络抓包方式查看明文的风险。需修改为 https、wss 等加密协议进行传输。

（2）用户密码是否存在登录次数和锁定限制。恶意用户通常可以通过暴力破解猜测用户名和密码。应对措施包括延长登录频率、限制登录失败次数、设置登录失败锁定时间等。

（3）密码算法安全性是否够强，密码加密是否采用了默认的盐值。恶意用户通过某些信息获取到项目所基于开源项目的情况下，可以通过开源系统盐值和算法，逆向计算或正向密码验证，实现对加密的破解。

（4）是否会导致用户隐私信息暴露。在获取用户信息时，可以通过设置合适的 VO 字段或不同权限的 API 接口，避免获取该用户或该页面不需要的信息（如：身份证号、住

址、电话、权限等信息），特别是要避免前端显示一些调试信息，这些调试信息可能存在隐私风险。

（5）是否存在越权漏洞。比如：普通用户不能配置超级用户的权限。同级别用户不能获取对方的访问权限等，这需要在前端和后端方面进行精心设计和配置。

（6）是否存在 SQL 注入风险。前、后端采用 SQL 指令直接访问数据，未对字段能进行验证，将可能导致 SQL 注入风险。解决手段包括采用 ORM 框架、参数化查询、合法表名白名单方式进行数据访问等。

（7）是否存在 XSS 跨站脚本攻击风险。XSS 攻击包含三种：存储型 XSS、反射型 XSS、基于 DOM 的 XSS。通常需要通过 CSP 多种手段综合手段进行防御，一些前端开发语言框架自身具备一定防止 XSS 的能力。

（8）是否存在 CSRF 跨站请求伪造风险。解决方案通常采用 CSRF 令牌机制、检查请求来源（Referer）、Same-Site Cookies 策略、敏感操作时再次身份验证等方式。

（9）是否存在 CORS 跨域资源共享风险。通常需要在服务器端合理设置响应头 CORS 相关字段（如：Access-Control-Allow-Origin 等）等方式解决。

（10）是否存在文件上传漏洞。文件上传漏洞是 Web 应用中因未对上传文件进行有效校验而产生的安全漏洞，攻击者可借此上传恶意文件控制服务器。防范方法主要包括：上传文件类型白名单、文件魔术头验证、上传文件大小控制、文件路径限制、上传文件重命名等。

（11）是否存在信息泄露风险（如：调试信息、报错信息、敏感路径、版本信息、网站地图信息等）风险。系统上线前应删除所有调试信息，敏感信息可采用加密方式存储。

（12）用户认证机制是否安全。比如：采用不安全的认证机制、不安全的加密算法、不安全的密码存储方式、不安全的密码重置流程等。需要深入系统内核代码，完善认证机制，采用更加安全的加密算法和密码存储方法。

（13）API 访问安全性不足。比如采用未经权限认证的 API 接口、API 参数未经校验、敏感信息未加密传输、API 访问权限设置不合理等。需要在后端精心设计和配置。

（14）依赖库存在漏洞。比如采用不安全的依赖库。著名的 log4j 漏洞事件就属于这一范畴。

二次开发设计加固实战：

以对 LIN CMS 进行二次开发系统为例，笔者通过对开源 LIN CMS 系统底层的研究，结合渗透测试、代码审计情况，发现该系统存在不少安全漏洞，并对源代码进行了加固：

（1）前端和后端采用未加密的 http 和 ws 协议进行传输。为此，笔者对 starter.js 文件进行修改，通过加入私钥和 SSL 证书，实现了 https 协议的传输。ws 协议主要用于消息处理，不影响使用，可进行屏蔽处理。

（2）未对密码登录次数进行限制和锁定。LIN CMS 系统设计为可无限次登录。为此，我们在后端对当前用户登录的次数进行限制，当指定时间内用户连续登录失败超过指定次数时，会锁定该用户一段时间，该方式能有效避免恶意用户短时间发起多次暴力破解尝试。

（3）未对密码强度进行验证，存在弱密码风险。在新建用户、用户修改密码、管理员重置密码时，增加了进行强度验证，新建或修改密码必须达到强度要求。

（4）密码加密采用默认的盐值，算法采用 SHA-1 和 1 次循环次数进行加密。这被证明是一种不安全的加密方式，我们修改了默认的加盐密码，重写了 password-hash.js 文件，采用更加安全的加密算法，提升了循环次数，大幅增加算法破解难度。通常我们可以设置循环次数为固定值，对于每次加密，也可以设计一个随机的循环次数，这样更加安全。

（5）系统后端 API 存在越权漏洞。原因开源代码和二次开发时一些 API 采用了不带权限的 put、get 方法，未进行权限验证。为此，我们在后端的所有 API 进行了检查，采用了带权限验证的 linPut、linGet 方法，并核实了所有 API 接口的 login-required、group-required 等权限。

（6）系统后端 API 存在 SQL 注入漏洞。主用原因主要有两点：

一是因前期调试方便，前端采用 SQL 指令传递到后端访问数据，后端直接执行该 sql 进行访问。对于这一类问题，我们将其改为 ORM 方式进行数据访问。

二是一些复杂的查询，无法通过 KOA 框架自带的 sequelize 框架进行查询，因此开发期间采用了 SQL 指令进行查询。对于这一类问题，由于无法通过 ORM 方式解决，我们在代码层面采用参数化查询、合法表名白名单方式进行了调整。

（7）存在 XSS 漏洞和上传文件漏洞。主要是上传的文件验证后缀合法性，对带后缀的文件没有进行格式校验。为此，我们在前端对文件类型进行指定，在后端对上传文件后缀指定了白名单，只允许指定的图片、压缩、pdf 等格式可以上传，并对每一种格式进行了魔术头验证，有效降低了 XSS 攻击和上传漏洞风险。

（8）CORS 配置不合理。原因是系统采用了默认的 CORS 配置，未对请求来源进行限制。为此，我们在后端采用 koa-cors 组件，对 CORS 配置进行了调整，只允许指定的域名进行访问。

（9）用户认证机制安全性不足。LIN CMS 系统采用 JWT 进行用户认证，JWT 方式由于是服务器分发 token，本地并不存储分发的 token，用户登录退出之后，token 过期时间可能还没结束。由于此时 token 仍然有效，攻击者可通过截获 token 进行攻击。为尽量减少代码改动，并保证系统安全，我们自己设计了一种 session 记录同 JWT 结合的方式，在用户登录、主动退出、重新登录、token 自动失效时对 token 进行记录或失效处置。通过上述方式，有效避免了用户 token 过期的问题，提升了系统安全性。

2.2.5 部署加固

部署加固策略分析：在开源 CMS 信息系统部署过程中，也可能存在一些安全问题：

（1）部署工具本身存在漏洞。一些旧版本的部署工具可能存在漏洞，如 node.js、nginx、Apache Tomcat 等，需要及时更新。

（2）部署配置不合理。部署过程中，需要对服务器的配置进行合理的调整，如：更换默认 80 或 8080 等端口、采用 https 等加密协议、合理配置反向代理等，避免出现安全漏洞。

（3）部署工具应设置 40x、50x 等错误页面，正式部署前删除网站地图文件。

（4）系统报错或文件可能泄露系统敏感信息。在部署过程中，可能会出现一些报错信息和包含网站敏感信息的文件，这些信息和文件可能包含系统的路径、版本等敏感信息，导致信息泄露，需要进行屏蔽或删除。

部署加固实战：笔者在对 LIN CMS 系统二次开发的系统在部署过程中，也遇到一些安全问题：

（1）对于自主开发的企业微信端，采用 hbuiderx 进行编译测试，发现存在 vite 版本过低且存在漏洞。后来采用其他的部署工具，解决了该问题。

（2）由于测试版本采用 hbuiderx 的编译部署，系统的报错信息中包含系统的路径信息，存在信息泄露的风险。在更换部署工具后，该问题也得到了解决。

（3）存在信息泄露风险。由于 LIN CMS 前后端系统采用 node.js 后端，在 node_modules 目录下存在不少 .js.map 文件，系统编译过程中也会生成该后缀文件。由于攻击者可能根据这类文件获取到网站地图，存在网站文件路径、第三方插件信息泄露的风险，我们在部署前，删除了所有 .js.map 文件。

3 总结和展望

开源系统的安全性一直是一个重要的研究领域。本文针对开源 CMS 系统的网络安全可能存在的问题进行了分析，并提出改进要点，同时以 LIN CMS 系统为例，剖析了该开源系统的安全性漏洞，并提出相应的解决手段。从实战的二次开发代码改动来看，安全加固需要深入理解系统内核，而且需要对系统网络安全有较深的理解。

随着互联网网络安全技术的发展，相关攻击、渗透技术层出不穷，信息系统防守的难度也在不断增加。开源 CMS 信息系统的安全不仅需要二次开发人员的网络安全意识和技术能力的提升，更需要开源贡献者的共同努力，从开源代码层面不断提升信息系统的安全性，减少二次开发人员的网络安全加固难度和部署风险。由于采用开源信息系统为基础进行二次开发的范围非常广，笔者呼吁政府和相关主管部门加强对开源信息系统支持和监管，出台一些更加有利于开源信息系统网络安全的指导意见。

参考文献

[1] 陈立, 刘洋. 开源软件供应链安全威胁建模与防护[J]. 网络与信息安全学报, 2022, 8(5): 34-43.

[2] 张伟. 织梦CMS安全性评估及加固措施[J]. 计算机工程与应用, 2024, 60(5): 89-97.

[3] 周敏, 吴峰. 基于行为分析的CMS系统动态安全加固模型[J]. 计算机工程, 2025, 51(3): 112-120.

[4] 陈立. RESTful API渗透测试与CMS接口防护[J]. 信息网络安全, 2025, 25(2): 56-63.