OWASP 中国
The Open Web Application Security Project

OWASP low-code/no-code

低代码十大安全风险

前言

项目概述

低代码/无代码开发平台提供了一个通过图形用户界面创建应用软件，而不是传统的手工编码计算机程序的开发环境。这种平台减少了传统手工编码的规模，从而加快了商业应用程序的交付。

随着低代码/无代码开发平台激增以及被组织广泛使用，产业界提出了一个明确而紧迫的需求，即建立依赖此类平台开发的应用程序相关的安全和隐私风险意识。

“OWASP Top 10 Low-Code/No-Code Security Risks”（简称OWASP 低代码十大安全风险）项目的主要目标是为希望采用和开发**低代码（可视化少量代码开发）、无代码（可视化无需编程开发）**应用程序的组织提供帮助和指导。该指南提供了关于此类应用最突出的十类安全风险、所涉及的挑战以及如何克服这些风险与挑战的信息。

“OWASP 低代码十大安全风险”项目由Zenity提供支持。该项目中文版由OWASP中国社区翻译发布。

风险清单

LCNC-SEC-01：身份冒充
LCNC-SEC-02：授权滥用
LCNC-SEC-03：数据泄漏和意外后果
LCNC-SEC-04：身份验证和安全通信失效
LCNC-SEC-05：安全配置错误
LCNC-SEC-06：注入处理失效
LCNC-SEC-07：脆弱和不可信的组件
LCNC-SEC-08：数据和密钥处理失效
LCNC-SEC-09：资产管理失效
LCNC-SEC-10：安全日志记录和监控失效

意见建议

欢迎您参与项目开发和推广！您可以通过Issues · OWASP/www-project-top-10-low-code-no-code-security-risks · GitHub或邮件Michael Bargury至michaelb@zenity.io 提供以下内容，帮助优化完善项目：

✓提供脆弱性流行趋势数据；

✓提出改进建议。

有关中文版的意见或建议，可发送邮件至project@owasp.org.cn。

中文项目团队

组　长：肖文棣
副组长：张剑钟
翻　译：郭振新、呼和、黄圣超、刘国强、马伟、任博伦、王厚奎、王强、吴楠、张坤
审　核：王颉、杨文元

LCNC-SEC-01

身份冒充

风险评级

普遍性	可检测性	可利用性	技术影响
3	2	3	3

风险要点

无代码/低代码开发的应用程序可能内嵌任何应用程序用户隐式冒充的用户身份。这为权限提升创建了一条攻击路径，允许攻击者隐藏在另一个用户的身份背后来绕过传统的安全控制。

风险描述

无代码/低代码开发的应用程序可以冒充现有的用户身份，而不是使用自己的应用程序身份。嵌入式身份的权限可以属于应用程序创建者，也可以作为公共身份而被团队共享，例如数据库凭据。

应用程序身份的缺失会导致敏感数据暴露在无代码/低代码开发平台之外的监控系统。作为外部查看者，任何使用应用程序的用户都可以冒充应用程序的创建者，并且现有方法无法区分应用程序及其创建者。当应用程序使用不同的身份在不同的平台上运行时，这个问题变得更加严重。在这种情况下，一个用户可用于将文件存储在共享 SaaS上，而另一个用户可用于检索本地数据。

此外，身份嵌入在应用程序中，多个用户可以使用该应用程序。这为权限提升创建了一条直接的攻击路径，应用程序用户可以在其中获得正常情况下不应拥有的访问权限。

攻击场景示例

1	创客创建一个简单的应用程序来查看数据库中的记录。创客使用自己的身份登录数据库，创建嵌入在应用程序中的连接。用户在应用程序中执行的每个操作最终都会使用创客的身份查询数据库。恶意用户利用这一特性并使用该应用程序查看、修改或删除他们不具有访问权限的记录。数据库日志表明所有查询都是由单个用户（应用程序创客）进行的。
2	创客创建一个业务应用程序，允许公司员工根据他们的信息填写表格。为了存储表单响应，创客使用自己的个人电子邮件账户。用户无法知道该应用程序将他们的数据存储在创客的个人账户中。
3	创客创建业务应用程序并与管理员共享。创客将应用程序配置为使用其用户的身份。除了已知目的，该应用程序还使用其用户的身份来提升创客的权限。一旦管理员使用该应用程序，就会无意中提升了创客的权限。

预防措施

1. 提供与外部服务的连接时，请遵守最低权限原则。
2. 确保应用程序使用专用服务帐户而不是用户帐户。
3. 确保应用程序在其所有连接中使用单一一致的身份，而不是为每个连接使用不同的身份。
4. 确保维护适当的审计跟踪，以识别通过共享连接执行的操作背后的参与者。

LCNC-SEC-02

授权滥用

风险评级

普遍性	可检测性	可利用性	技术影响
3	3	3	3

风险要点

在大多数无代码/低代码的平台中，服务连接都是头等对象。这意味着应用程序、其他用户或整个组织之间的连接。应用程序也可能被共享给无权访问基础数据的用户。

风险描述

无代码/低代码应用程序不是建立在孤岛中的，而是集成在基于本地部署、软件即服务(SaaS)、平台即服务(PaaS)和云平台的跨组织堆栈上。大多数无代码/低代码平台都内置了大量连接器(即围绕API的包装器)，可轻松地实现快速连接。大多数无代码/低代码平台中，连接器和用户凭证形式的连接都是头等对象。这意味着可以在应用程序之间、与其他用户或与整个组织共享连接。

许多无代码/低代码平台通过查询和存储用户刷新令牌并随意重复使用来提高生产力并缩短交付时间，这导致滥用OAuth授权流程。这使业务用户无需考虑密钥或权限即可快速建立连接，同时，连接嵌入了难以监控或撤销的用户身份。尽管OAuth刷新令牌被设计为短期的，但是通常有效期为几个月甚至几年。因此，业务用户在一分钟内创建的连接可能会在无代码/低代码平台中持续很长时间，导致其他用户可以经常将这些连接用于与原始意图不同的目的。

攻击场景示例

1	创客创建一个连接到他们公司的电子邮件账户并且无意中点击了“与所有人共享”选项。组织中的每个用户，包括承包商和供应商，都可以访问创客公司的电子邮件账户。恶意用户触发“忘记密码”流程并使用连接来完成该过程，从而获得对账户的控制权。
2	创客创建一个简单的应用程序来查看数据库中的记录。该应用程序被配置成确保每一个用户只能查看相关的记录。然而，应用程序的配置方式是底层数据库连接与其用户隐式共享。应用程序用户可以直接使用数据库连接，获得对所有记录的完全访问权限。
3	管理员使用服务账户将应用程序连接到自己的源代码管理系统（即BitBucket）。配置的服务账户可以不受限制地访问所有存储库，以实现无缝集成。任何内部用户都可以滥用此连接来访问自己正常情况下无权访问的受限存储库。

预防措施

1. 禁用或监控隐式共享连接的使用。
2. 在提供对包含共享连接的环境的访问时，遵循最小特权原则。
3. 监控无代码/低代码平台的过度共享连接。
4. 教导业务用户了解连接共享的风险及其与凭证共享的关系。

LCNC-SEC-03

数据泄露和意外后果

风险评级

普遍性	可检测性	可利用性	技术影响
3	2	3	3

风险要点

无代码/低代码应用程序通常会跨多个系统同步数据或触发操作，这为数据跨越组织边界创造了一条攻击路径。这就意味着，在一个系统内的操作可能对另一个系统中造成意想不到的后果。

风险描述

无代码/低代码应用程序经常被用于多个系统之间的同步数据，或由于另一个系统的更改而触发其他系统的操作。作为数据移动的载体，无代码/低代码应用程序轻易就可以通过把数据移动到组织边界外部的另一个组织或个人账号而导致数据泄露。而当作为操作触发器，无代码/低代码应用程序可能通过将一个系统中的操作与另一个系统中的更改隐式耦合而造成意想不到的后果。此外，单个数据源可以连接和触发多个应用，从而导致难以预测和难以完全匹配的链式数据移动和操作触发。

攻击场景示例

1	创客配置了在其公司邮箱中收到的每一封新电子邮件时触发的自动化操作，该操作会自动向创客的个人电子邮件账户发送一封新的电子邮件，并从公司邮箱中收到的原始电子邮件中复制收件人、主题和正文。由于数据是复制到单独的邮箱而不是从公司邮箱转发的电子邮件，因此这个自动化操作可能绕过数据防泄漏（DLP）的控制。
2	场景1的创客配置了在两个SharePoint网站之间同步更改的自动化操作，因此站点A的每一个新文件都复制到站点B。用户2不小心将敏感文件写入到站点A，该敏感文件在用户2不知情的情况下同步复制给了站点B。用户2删除了站点A的敏感文件，然而该敏感文件仍会存在站点B上。

预防措施

1. 将平台连接限制在批准的服务列表中。
2. 将自定义连接器的创建限制为专职人员。
3. 监控平台以了解组织边界外的数据流，包括多跳路径。

LCNC-SEC-04

身份验证与安全通信失效

风险评级

普遍性	可检测性	可利用性	技术影响
2	3	2	2

风险要点

无代码/低代码应用程序通常通过业务用户设置的连接来连接到关键业务数据，这往往会导致不安全的通信。

风险描述

无代码/低代码应用程序通常采用跨内部部署、SaaS、PaaS以及云平台连接到关键业务数据。应用程序使用内置连接器，可以轻松连接到各种服务。连接提供了各种安全配置，包括通信协议、身份验证流程和使用的凭据类型。但在许多情况下，业务用户违反最佳实践和企业数据安全政策建立连接，这通常会导致安全风险。

攻击场景示例

1	创客创建了一个使用FTP连接的应用程序，并且没有勾选"加密"的复选框。由于应用程序与其用户之间的通信是加密的，因此应用程序的用户无法获悉自己的数据正在未加密的情况下进行传输。
2	创客使用管理员凭据来创建数据库连接并构建了一个应用程序，且应用程序使用该连接向用户显示数据。在这种情况下，尽管创客的计划是只允许用户通过应用程序进行只读操作，但用户也可以使用特权连接从数据库中写入或删除记录。

预防措施

1. 在生产环境中，将连接的创建限制为专职人员。
2. 监控平台是否存在不符合最佳做法的连接。
3. 通过培训使业务用户了解不安全通信的风险并且在做相关设置时让安全团队参与进来。

LCNC-SEC-05

安全配置错误

风险评级

普遍性	可检测性	可利用性	技术影响
3	2	3	3

风险要点

配置错误往往会导致匿名访问敏感数据或操作，以及不受保护的公共端点、密钥泄漏和过度共享。

风险描述

无代码/低代码平台提供了广泛的功能，其中一些功能控制着安全性和对特定用例支持之间的平衡。错误的配置通常会导致匿名用户也能访问敏感数据或操作，以及不受保护的公共端点、密钥泄漏和过度共享。此外，许多配置是在应用层面而不是租户层面，这意味着可以由业务用户而不是管理员进行设置。

攻击场景示例

1	创客创建一个应用程序，该应用程序公开了一个API端点。但是，该端点没有被配置为拒绝匿名访问。因此，攻击者扫描低代码/无代码平台的子域，找到该应用并窃取其底层数据。
2	创客创建由webhook触发的自动化操作，但未能用密钥保护该webhook。攻击者识别了webhook，从而可以随意触发自动化操作。该自动化操作可能是修改或删除数据。

预防措施

1. 阅读供应商的文档并遵循最佳实践指南。
2. 确保配置符合行业最佳实践。
3. 监测配置的偏移。
4. 为租户级的配置实施一个变更管理系统。

LCNC-SEC-06

注入处理失效

风险评级

普遍性	可检测性	可利用性	技术影响
2	2	2	2

风险要点

无代码/低代码应用程序以多种方式接收用户提供的数据，包括直接输入或从各种服务中检索用户提供的内容。此类数据可能会包含给应用程序带来风险的恶意有效载荷。

风险描述

无代码/低代码应用程序以多种方式接收用户提供的数据，包括直接输入或从各种服务中检索用户提供的内容。由于应用程序经常根据用户输入动态查询数据，因此会面临着基于注入攻击的重大风险。此外，由于应用程序可以以各种方式使用用户提供的内容，包括查询数据库、解析文档等，因此防止基于注入攻击必须考虑特定应用程序及其对用户数据的使用。

攻击场景示例

1	创客设置在新RSS订阅发布时将该订阅存储到SQL数据库中的自动化操作。控制该RSS订阅的攻击者利用该自动化操作向数据库中注入删除重要记录的命令。
2	创客创建了一个允许用户填写表单的应用程序。该应用程序将表单数据编码为CSV文件并将CSV文件存储在共享驱动器上。即使平台为SQL注入攻击清理了表单输入，但并没有针对Office宏攻击进行清理。攻击者利用这一点，输入一个在写入CSV文件的宏。用户打开CSV文件以分析用户表单，即可执行宏。

预防措施

1. 清理用户输入，同时考虑应用程序将对该输入执行的操作。
2. 告知业务用户并使其了解到未经处理的用户输入的会带来的风险，这是平台无法自行解决的问题。

LCNC-SEC-07

脆弱和不可信的组件

风险评级

普遍性	可检测性	可利用性	技术影响
2	2	2	2

风险要点

无代码/低代码应用程序严重依赖于市场或web上现有组件，以及由开发人员构建的自定义连接器。这些组件通常是非托管的，缺乏可见性，并使应用程序面临基于供应链的风险。

风险描述

无代码/低代码应用程序严重依赖于市场或Web上的现成组件，包括数据连接器、小工具和子服务。在许多情况下，整个应用程序是由供应商构建的。第三方组件和应用程序往往成为那些希望危害大量客户的攻击者的目标。

此外，无代码/低代码应用程序往往通过自定义代码实现可扩展性。这些代码嵌入到应用程序中，在某些情况下，它们的安全性没有受到与专业代码应用程序同等程度的重视。

攻击场景示例

1	整个组织的创客都使用来自公开的脆弱的组件。每个使用该组件的应用程序都暴露在攻击下。管理员可能会发现很难找到受脆弱组件影响的应用程序。
2	开发人员创建一个自定义连接器，允许创客连接到内部业务API。自定义连接器在URL上传递身份验证令牌并向应用程序用户暴露身份验证密钥。

预防措施

1. 删除不使用的依赖项、不必要的特性、组件、文件和文档。
2. 持续关注这些应用程序使用的应用程序和组件版本，并扫描该清单以查找已废弃或易受攻击的组件。
3. 限制使用预先批准的公开组件。
4. 监控未维护或未为旧版本创建安全补丁的组件。

OWASP low-code/no-code

LCNC-SEC-08

数据及密钥处理失效

风险评级

普遍性	可检测性	可利用性	技术影响
3	2	3	3

风险要点

无代码/低代码应用程序通常将数据或密钥作为其“代码”的一部分进行存储，或者存储在平台提供的托管数据库中，而这些数据必须按照法规和安全要求进行适当的存储。

风险描述

无代码/低代码应用程序可以将数据作为其“代码”的一部分进行存储，或者存储在平台提供的托管数据库中。存储在由无代码/低代码供应商管理的数据库中的数据通常包含一些敏感数据，例如个人可识别信息（PII）和财务数据。应用程序创建者可以决定如何存储这些数据，然而管理员通常缺乏对此类托管数据库的可见性。在许多情况下，敏感数据违反监管要求未经加密存储就在不同地理位置之间传输。

此外，应用程序创建者经常会把密钥硬编码到“代码”中。无论是通过环境变量、配置还是代码，应用程序通常可以依靠硬编码的密钥来访问其他服务。对于这些硬编码的密钥，任何对该应用程序具有写入权限的用户都可以访问到，并且还可能通过客户端代码泄露给应用程序的使用者或者匿名用户。

另外，许多本机日志流混合了应用程序日志、指标和通过应用程序传递的敏感数据。在许多平台中，日志将包含应用程序默认使用的实际数据点。

攻击场景示例

1	创客创建一个业务应用程序，要求用户填写包含敏感数据的表单。应用程序使用平台提供的托管数据库来存储结果，然而由于所有其他创客默认使用托管数据库进行存储，因此其他创客都可以访问到这些敏感数据。
2	创客在创建的应用程序中使用了自定义API，并在代码中硬编码了访问该API的密钥。于是，其他创客也就可以直接访问到这些API密钥。此外，这些API密钥可能会泄漏到应用程序的客户端代码中，从而使用户也可以直接访问到这些密钥。

预防措施

1. 通过培训，提高业务用户与数据存储相关的合规性、隐私和风险的安全意识。
2. 加强对于无代码/低代码供应商提供的托管数据库、环境变量和配置的监控，及时发现不恰当存储的敏感数据。
3. 对于涉及敏感数据处理的应用程序，确保安全团队参与其中。

LCNC-SEC-09

资产管理失效

风险评级

普遍性	可检测性	可利用性	技术影响
3	2	3	2

风险要点

无代码/低代码应用程序易于创建开发，并且维护成本相对较低，这个特点使这些应用程序在保持活动状态的同时，组织也很容易弃用这些应用程序。此外，内部应用程序可以在不解决业务连续性问题的情况下迅速普及。

风险描述

无代码/低代码应用程序在保持活动状态同时，组织也容易弃用这些应用。究其原因如创建新应用程序的方便性、相对较低的维护成本以及这些应用程序通常由SaaS服务管理。这意味着组织中活动应用程序的数量往往会快速增长，流行的业务应用程序往往会发现自己没有所有者。

此外，大型组织内有用的内部业务应用程序，具有病毒式特点，可能导致应用程序由单个业务用户开发但被整个组织内的许多用户使用。无代码/低代码应用程序可能缺乏重要业务应用程序所期望的业务连续性有效措施，例如，拥有多个所有者、受IT部门监控或提供SLA。

攻击场景示例

1	创客创建的应用程序成为流行的内部工具。创客离开组织或从事其他角色。由于API更改，应用程序中断，业务中断。IT部门根本不了解这个应用程序，也无法帮助修复这个应用程序。
2	创客浏览平台市场并探索应用模板。每次单击都会创建一个带有面向外部URL的应用程序。用户很可能会忘记这些应用程序，这将造成业务数据暴露。这种情况再乘以创客的数量，导致废弃应用程序资产数量不断增加。

预防措施

1. 维护一个包含有应用程序、组件和用户的全面资产清单。
2. 删除或禁用不使用的依赖项、不必要的特性、组件、文件以及文档。

LCNC-SEC-10

安全日志记录和监控失效

风险评级

普遍性	可检测性	可利用性	技术影响
2	2	3	2

风险要点

无代码/低代码应用程序通常缺乏全面的审计跟踪，存在不生成日志或日志不足的问题，或者对敏感日志的访问没有严格管控。

风险描述

无代码/低代码应用程序通常依赖于供应商来生成日志和监视数据。在许多情况下，日志要么不足，要么没有收集，从而阻碍了安全调查，并且无法满足合规性要求。

此外，应用程序通常缺乏全面的审计跟踪，从而阻碍了变更管理流程和查询，很难找出是谁引入了一项变更。

攻击场景示例

1	应用日志已关闭。当发生违规尝试行为时，安全团队无法确定谁访问了该应用程序以及访问者尝试执行的操作。
2	业务关键型应用程序在发生变更后停止运行。由于发生了多个变更，每个变更都会导致应用程序更新，因此很难找到哪个创客引入了导致问题的特定变更。创客必须手动检查每个应用程序版本才能找到有问题的版本。由于每个应用程序“保存”都会转换为更新，因此更新的数量将使手动过程的成本过高。在某些平台上，创客只能查看应用程序的当前版本，因此创客将无法找到或恢复到稳定版本。

预防措施

1. 利用平台内置功能收集用户访问和平台审核日志。
2. 在适用的情况下，使用日志记录机制检测应用程序，以提供额外的可见性。
3. 通过配置平台以避免记录原始应用程序的数据，确保日志不包含敏感数据。

参考文献

风险名称	参考链接
LCNC-SEC-01 身份冒充	• https://www.darkreading.com/dr-tech/why-so-many-security-experts-are-concerned-about-low-code-no-code-apps
LCNC-SEC-02 授权滥用	• Low-Code Platforms Are the New Holy Grail for Hackers
LCNC-SEC-03 数据泄露和意外后果	• Low-Code Security and Business Email Compromise via Email Auto-Forwarding • Hackers Abuse Low-Code Platforms And Turn Them Against Their Owners
LCNC-SEC-04 身份验证与安全通信失效	• The CISO View: Protecting Privileged Access in RPA
LCNC-SEC-05 安全配置错误	• By Design: How Default Permissions on Microsoft Power Apps Exposed Millions • Microsoft Internal Security Best Practices: Secure Power Platform Development • Power Platform Landing Zones Reference Implementation
LCNC-SEC-06 注入处理失效	• A03:2021 – Injection, OWASP Top 10
LCNC-SEC-07 脆弱和不可信的组件	• CVE-2021-44228: Incident Report for Mendix Technology B.V. • A06:2021 – Vulnerable and Outdated Components, OWASP Top 10
LCNC-SEC-08 数据及密钥处理失效	• Establishing a DLP strategy
LCNC-SEC-09 资产管理失效	• Why So Many Security Experts Are Concerned About Low-Code/No-Code Apps
LCNC-SEC-10 安全日志记录和监控失效	• A08:2021 – Software and Data Integrity Failures, OWASP Top 10

感谢阅读

扫码加入