MSZQ

民生证券股份有限公司企业标准

Q/MSZQ J002.1-2024

民生证券

信息系统架构评审标准

Information System Architecture Review Standards

（民生证信备字〔2024〕7号，2024年9月25日备案）

2024-9-25备案　　2024-9-25实施

民生证券股份有限公司 发布

目　　次

前　言

本文件由民生证券股份有限公司提出。

本文件由民生证券股份有限公司归口。

本文件起草部门：民生证券股份有限公司信息技术中心。

民生证券信息系统架构评审标准

1. 范围

本标准详细规定了民生证券股份有限公司在进行信息系统架构评审过程中所需提交的信息和必须着重考量的关键要素。架构评审的核心在于评估技术方案的合理性与系统的健壮性，涵盖但不限于以下几个方面：

（1）系统整体设计的综合性考量

（2）组件的选用及其在不同应用场景下的适用性

（3）灾难恢复计划与系统的高可用性保障措施

（4）数据备份机制和全面的信息安全策略

通过这些评审维度，确保所提出的技术方案能够全面契合公司信息技术架构的标准和要求。

本标准适用于民生证券股份有限公司在信息技术建设领域内的所有架构评审活动。

2. 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 22240-2020 信息安全技术 网络安全等级保护定级指南

3. 术语和定义

下列术语和定义适用于本文件。

3.1 零信任

零信任是一种安全模型，基于访问主体身份、网络环境、终端状态等尽可能多的信任要素对所有用户进行持续验证和动态授权。零信任与传统的安全模型存在很大不同，传统的安全模型通过“一次验证+静态授权”的方式评估实体风险，而零信任基于“持续验证+动态授权”的模式构筑企业的安全基石。

3.2 DevOps

DevOps 是 Development（开发） & Operations（运维）的组合，是软件研发的“工业革命”。DevOps 强调优化开发侧和运维侧之间的协同合作，提倡对软件研发全生命周期进行文化、工艺和流程上的重塑，将原来「手工作坊」、「闭门造车」的传统研发模式改造为「自动化」、「标准化」和「可视化」的现代研发模式。

3.3 单点登陆（SSO）

单点登录（SSO）是一种身份验证解决方案，可让用户通过一次性用户身份验证登录多个应用程序和网站。鉴于当今的用户经常直接从其浏览器访问应用程序，因此组织正在优先考虑改善安全性和用户体验的访问管理策略。SSO 兼具这两方面的优点，因为一旦验证身份，用户就可以访问所有受密码保护的资源，而无需重复登录。

3.4 权限稽核

权限稽核是指通过某些手段定时定期检查系统的权限是否分配

不合理。

3.5 SCA

SCA（Software Composition Analysis，软件成分分析）技术是Gartner定义的一种应用程序安全检测技术，该技术用于分析开源软件以及第三方商业软件涉及的各种源码、模块、框架和库等，以识别和清点开源软件的组件及其构成和依赖关系，并检测是否存在已知的安全和功能漏洞、安全补丁是否已经过时或是否存在许可证合规或兼容性风险等安全问题，帮助确保企业软件供应链中组件的安全。

3.6 IAST

IAST（Interactive Application Security Testing，交互式应用安全测试）技术，借助应用插桩（含主动及被动）、代理/VPN等手段，并运用全场景流量分析技术，实现了极高的漏洞检出率和极低的漏洞误报率。该技术通过污点跟踪这一核心方法，对数据流进行追踪与分析，有效识别并标记出可能包含安全风险的“污点数据”，并监控这些数据在应用程序中的流转过程，检查它们是否未经充分处理就直接流向了敏感操作点。 同时IAST能够精准定位到API 接口和代码片段，让传统功能测试人员在执行应用功能测试的同时，也能透明地完成这一任务，有效覆盖90%以上的中高危漏洞。

4. 系统功能介绍

在架构评审过程中，每个参与评审的系统都应提供一份精炼的业务背景描述和功能概览。系统文档应清晰阐述其旨在解决的业务需求和实现的核心功能。

对于采用分阶段开发模式的系统，必须详细说明整体的分期建设规划，并对当前阶段的建设目标和功能特性进行重点介绍，以确保评审团队能够全面理解系统的发展方向和当前阶段的关键成果。

5. 系统基本信息

系统在架构评审前应提供以下基本信息并录入 CMDB 系统。如果一个系统可以划分为另一个系统的子系统，优先推荐归类为子系统，尽量减少一级系统的数量。

【系统名称】【子系统名称】【信息系统状态】【系统负责人】【负责人备岗】【信息系统描述】【所属部门】【开发类型】【供应商名称】【系统是否访问互联网】【等保测评等级】【备份能力等级】【系统使用对象】【所属团队】

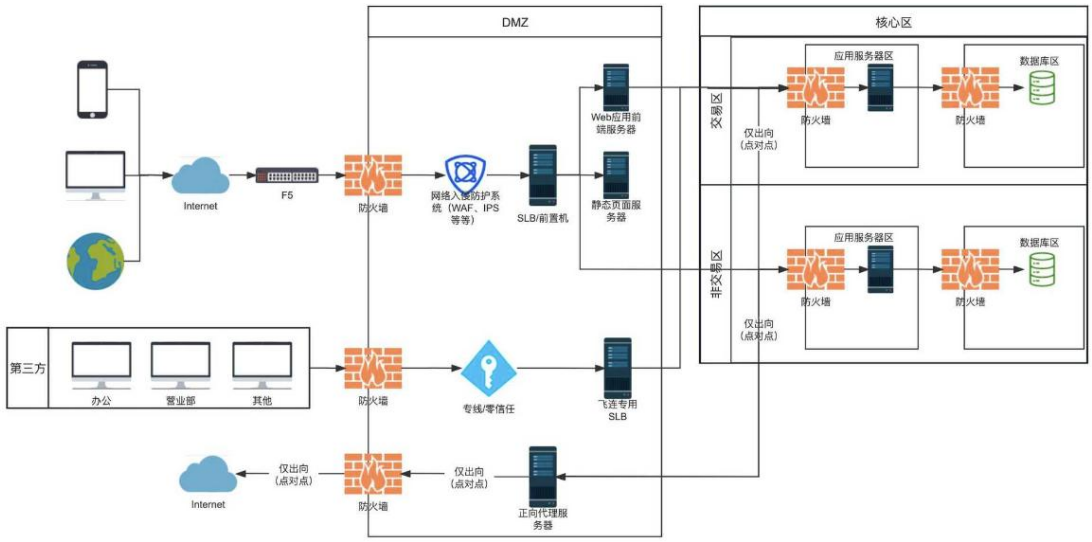
6. 通用架构评估

参与架构评审的系统应提供以下信息。

【逻辑架构图/网络拓扑图】【前中后台采用的高可用方案】【系统使用的中间件（缓存，消息队列等）以及使用场景】【灾备环境的架构图以及与主环境切换场景及切换方式】【是否是 Java 应用】【使用的数据库类型及版本】【部署模式：物理机部署、虚拟机部署，或者是容器化部署】【关联系统信息】。

参与架构评审的委员需根据公司的 IT 管理规范结合系统实现的业务场景进行架构评估。具体评估过程中遵循以下原则。

（1）架构图应能清晰体现组件的部署区域，应用组件间的调用链路及每台服务器部署的应用清单，具体参考下图示例。



（2）系统中各组件应实现高可用部署，不能实现的需说明具体原因。

（3）中间件优先采用 PAAS 平台上的云服务，如云服务不能满足应用系统的需求可自行搭建。

（4）等保三级的系统必须搭建灾备环境，其他系统可以根据业务对系统高可用的要求灵活选择是否需要搭建灾备环境。有灾备环境的，需详细描述灾备切换的方案。

（5）如果是 Java 应用程序，建议采用 Oracle Open JDK 8 / 17，如有特别原因需要采用 Oracle JDK 的话，请采用 JDK8u211 之前的版本。

（6）系统部署优先推荐使用国产服务器和操作系统，中间件和数据库推荐使用技术自主可控的版本。

7. DevOps 系统接入评估

在架构评审过程中，评审委员需对系统的研发管理策略进行细致评估，以确定是否需将研发流程整合至 DevOps 平台中。这一整合对

于提升研发效率、加强流程透明度和实现持续改进至关重要。

对于自主研发的系统，其需求管理、测试管理、持续集成、持续交付以及制品管理等关键流程，均应无缝接入 DevOps 平台。这不仅适用于内部开发的项目，同样适用于那些提供源代码的外部采购系统。

对于不提供源代码的外购系统，应确保供应商交付的所有制品均通过 DevOps 平台的制品库进行有效管理。这确保了制品的可追溯性、版本控制和整体研发流程的一致性。

8. 性能规划

参与架构评审的系统需提供系统关键性能指标，例如支持同时在线人数、主要业务并发数、主要查询响应时间、业务操作响应时间等。

参与架构评审的委员需深入分析这些性能指标，并结合系统所服务的业务场景，评估系统是否具备满足业务需求的性能水平。

对于被评定为重要级别的信息系统，必须开展压力测试。压力测试的目的是验证系统在高负载条件下的性能表现，并确保系统能够承受预期的最大用户访问量，同时保持服务的响应速度和稳定性。

9. 容量评估

参与架构评审的系统应提供系统需要的资源容量信息，具体包括以下几项。

（1）存储容量

（2）网络容量

（3）裸金属、虚拟机或容器资源的 CPU 和内存等容量

参与架构评审的委员需评估系统预计占用的带宽是否会影响已有系统的正常运行，系统申请的资源容量是否合理。此外，由于系统正式上线之前，对系统资源的估算和实际需求往往存在较大的偏差，故需要结合监控对系统资源进行持续运营评估，使用率低的资源及时进行回收。

10.备份能力评估

参与架构评审的系统应详细描述对数据备份能力的需求，常见的备份能力有以下几种。

（1）至少每天、周、月进行一次数据备份；

（2）不需进行数据备份；

（3）数据备份在本地、本地+异地；

（4）是否需要每季度数据备份进行一次有效性验证；

参与架构评审的委员需结合系统的重要性及数据丢失对系统正常运行的影响程度对备份能力进行评估。如果为重要信息系统要严格遵守《信息系统备份能力建设指引》中关于重要信息系统备份能力的要求。建议通过公司的统一备份系统进行备份。

参与评审的系统应详细描述对故障和重大灾难应对能力的要求。

参与架构评审的委员需结合系统的重要性以及系统故障对业务造成的影响评估系统应采用哪种系统备份模式以及采用哪种灾备建设模式。

常见的应用系统备份模式有：

（1）冷备；

（2）温备；

（3）热备；

（4）双/多活；

常见的灾备应对能力有：

（1）不需要建设异地灾备；

（2）应用冷/温/热备数据实时同步；

（3）异地双/多活；

11. 安全评估

参与评审的系统应满足以下安全规范。项目经理需要在系统建设之前将以下安全规范同步给组内开发人员或者软件系统供应商。

类别	评估项	安全要求
合法合规	数据来源合规	确保生产数据不使用通过爬虫、黑产买卖、未经第三方授权的数据
访问控制	权限矩阵评估	对权限矩阵进行说明，如正式员工、外包员工，权限，包括应用权限和数据库权限方面的设计等
	登陆的认证方式评估	对登录、注册、密码找回、领奖等 web 页面，存在爆破风险，需要明确认证方式单双因素以及具体组合，以及是否有图形码验证
	密码策略	密码策略应有一定的长度和复杂性，更安全的做法推荐： 1） 设定每 3 个月修改一次； 2） 不能使用过去曾经使用的 5 个密码； 3） 若用户密码由系统分配，用户首次登录系统或密码重置后第一次登录，密码必须被强制修改； 4） 密码修改时，必须认证旧密码 5） 密码应该由大小写字母、数字、字符中的若干种组合而成
	登陆策略	1） 对输错次数进行限制 2） 登陆失败时，不允许提示登陆失败原因（建议做法：显示账号或密码错误，验证码单独提示验证码错误）
	短信认证方式评估	采用短信认证的系统，应 1）采用限流措施 2） 明确时效性 3） 仅可一次性使用
	是否具备外网访问控制措施	1） 对于生产环境，生产网的核心层不应该与外网直接通信，一般通过 DMZ 层与外网通信；

	（针对具有外网场景的应用）	2） 一般采用网络防火墙设备进行外网访问控制，原则是提供最小范围的通信 ip、协议、和端口信息。对于动态的通信地址（如：微信 api 服务器地址就是动态的），可以在程序级通过对方提供的专用接，来获取、配置和限制来源地址； 在应用层，采用域名（证书）检查、appkey 检查、access_token 检查等方式，确认通信对端的身份。
日志与审计	是否有日志留存	应该被记录的日志包括： 1） 系统中发生的与安全相关的活动（用户的登录和登出、口令的修改或重置 2） 特权账号的操作记录，如管理员对用户和角色的增加、删除、修改权限、修改角色、会话创建和结束； 3） 用户认证错误、用户侧提交的请求格式或语法错误（输入验证错误）或流量超限等）； 4） 服务的告警、故障、启动和终止 5） 系统发现的违规操作 6） 日志留存的预期效果是：在发生信息安全事件时，能够快速通过日志溯源到相关行为
	日志的保存期限、格式、安全性	日志的格式应该包括： 1） 用户 ID（如有）， 2） 日期和时间 3） 终端身份和位置（IP 和 MAC） 4） 事件名称 同时，日志中不应以明文方式记录敏感信息（如：密码、口令、会话 ID、手机号、身份证号、银行卡号等）
数据安全与敏感信息保护	公网是否加密传输（针对具有外网场景的应用）	数据在传输离开公司生产核心网后，根据项目的实际情况，建议使用 SSL、对称加密等方式保护数据。避免数据在非核心网传输被窃取、监听、或者中间人攻击。禁止使用 DES， 3DES ，MD2，MD4， MD5，SHA1 等已经淘汰的算法，禁止自行研制密码系统和加密算法，禁止支持 SSL3.0 及以下协议。
	会话管理措施	1）是否设置会话超时机制 2）同一用户的最大会话数。同一用户一般至多一个有效会话，少量场景可能会要求同一用户同时有几个有效会话。
	SessionID 安全设置	会话 cookie 中缺少 HttpOnly 属性会导致攻击者可以通过程序(JS 脚本、Applet 等)获取到用户的 cookie 信息,造成用户 cookie 信息泄露。因此如果 WEB 服务器生成并维护 SessionID（如 JSessionID, PHPSessID 等），需要设定属性为：HttpOnly 和 Secure
	错误与例外处理脱敏	作为潜在的信息泄露源，错误与例外的不应包含敏感信息（如系统内部结构、设计、配置信息）
	代码规范	代码中不可暴露明文密码，如数据库密码、服务器用户密码等，应加密放置在配置文件中，通过密钥进行连接
	开发测试环境数据脱敏	开发测试环境使用的数据，原则上采用公开数据，或构造出的测试数据，或经过部门认可脱敏后的数据，禁止直接使用生产数据
	压力测试计划	上线前是否需要进行压力测试 后续变更可能涉及压力测试的场景

	数据收集是否遵循最小化原则	本系统中收集的敏感数据，是否都有实际的业务应用场景
	公有云数据安全	对于《证券期货业数据分级分级指引》标准中，属于三级及以上分级的数据（如：交易客户的姓名、手机号、身份证号、交易口令、通信口令等），原则上不宜存贮于公有云（如：AWS, Azure, 阿里云）环境上。如果一定要上，则需要进行内部可行性讨论并确定具体的保护措施。
安全防控	移动 APP 加固	移动端上的 APP 处于不可信的环境中，面临设备丢失、程序被逆向、跟踪、调试和通信拦截等风险，需要考虑代码混淆与加壳保护，以增加攻击者逆向的难度。
	SQL 注入防控	使用参数化的查询语句（preparedstatement），对于（不可信来源的）外部传入参数，做占位符(?)和绑定操作(bind)；使用 ORM 框架（如：MyBatis），对于配置文件中，检查所有参数绑定的方法，要求用#进行绑定，如果有$绑定，则需要修改，实在无法修改的，需要确保在应用代码层保证参数的可信性；不可在有用户输入、不可信外部输入的情况下，采用动态 sql 拼接查询
	文件上传漏洞防控	尽量不提供文件上传功能，如有，需要采用防范措施并在服务端做好验证（如检查文件扩展名），确保前端用户或其它不可信来源上传的文件无法被 web 服务器解析成脚本文件；不要把文件保存在与应用程序相同的 Web 环境中，建议将文件保存在专用的文档服务器中并禁止脚本执行（如 PHP.ini 中可以配置）;确保上传的文件扩展名与实际的文件内容类型相匹配。
容器安全	Docker 版本保持最新	推荐使用最新版本的 docker，过时的版本容易受到安全攻击。新版本发布通常包含修补程序和错误修复程序，以解决旧版本的漏洞。如不能使用，提供不能使用原因和现有 docker 版本
	限制容器权限	应将容器的权限限制为仅运行其应用程序所需的权限。例如，可用 docker run --cap-drop ALL 删除容器所有特权，再使用 docker run –cap-add=XXX 向容器增加特权（建议先与开发方咨询）
	不要将容器的命名空间和宿主机进程的命名空间进行共享	建议不要将容器的命名空间和宿主机进程的命名空间进行共享，这样可以降低用户从容器中进入容器所在的宿主机的风险，进而降低同宿主机上其他用户的容器被攻击的风险。
	进程数限制	一般建议一个容器中只跑一个进程，容器中允许运行的进程数过多的话会带来 fork bomb 的风险，即攻击者可能会利用循环在容器中不断的新建进程，导致为容器分配的资源被耗尽，如果这个时候恰好又未限制容器可以使用的资源量或者限制的数值较大的话，可能会导致容器所在宿主机上的资源被耗尽，导致当前宿主机上的其他容器触发迁移。
	端口映射限制	为防止其他人通过容器内的特权端口(1024 以下的端口)攻击容器，比如开启 22 端口会增加容器中系统的账号信息被暴力破解的风险，所以一般建议容器只映射应用监听的端口。
	不将宿主机敏	有时我们会将宿主机上的一些公共的目录映射到容器中，防止每个容

	感目录映射到容器中	器的镜像中反复配置，比如 dns 配置文件等。但不建议将宿主机上一些比较敏感的目录映射到容器中，比如宿主机上的内核配置文件，网卡配置文件等，防止用户在容器中对宿主机的关键配置进行篡改。
软件供应链安全	漏洞防御	禁止使用存在严重和高危漏洞的第三方组件，存在中危或低危漏洞的组件，需要结合实际使用场景和功能代码，进行安全性分析和评估，是否存在真实可利用等安全风险。
	许可证风险	分析调用的第三方组件的许可证类型，禁止使用未取得商用授权或未明确表示允许商用的第三方组件。
	活跃度评估	禁止使用 3 年内未发生过更新的第三方组件。
	供应商和约束	对于外购产品，供应商应通过合同等方式承诺所提供产品无严重已知漏洞、未使用存在严重和高危漏洞的第三方组件。如存在中危和低危漏洞的组件，需要结合实际使用场景和功能代码，进行安全性分析和评估，是否存在真实可利用等安全风险，同时，供应商对发现新漏洞需及时同步项目组，提供加固建议，并建立联合响应机制。供应商应提供成品软件物料清单（SBOM），并明确提供 SBOM 若存在真实性、授权、合规等纠纷时所需承担的责任。
安全检测	渗透测试	系统或应用上线前，必须进行全面的渗透测试，测试结果明确禁止存在严重和高危漏洞，若存在中危或低危漏洞，需要结合实际使用场景和功能代码，进行安全性分析和评估，是否存在真实可利用等安全风险。同时要求在渗透测试完成后，上线前系统或应用不可进行任何变动。
	漏洞扫描	系统或应用上线前，必须进行全面的服务器漏洞扫描，扫描结果明确禁止存在严重和高危漏洞，若存在中危或低危漏洞，需要结合实际使用场景，进行安全性分析和评估，是否存在真实可利用等安全风险。
	SCA 检测	系统或应用在上线前应进行 SCA 检测，扫描结果明确禁止使用存在严重和高危漏洞的第三方组件，存在中危或低危漏洞的组件，需要结合实际使用场景和功能代码，进行安全性分析和评估，是否存在真实可利用等安全风险。
	IAST 检测	自研应用系统应先在测试环境进行完整的功能测试，并要求必须在测试环境植入 IAST agent，系统或应用上线前对 IAST 检测结果报告进行安全审核，明确禁止存在严重和高危漏洞，若存在中危或低危漏洞，需要结合实际使用场景和功能代码，进行安全性分析和评估，是否存在真实可利用等安全风险。同时要求 IAST 运行测试环境和正式上线环境相同。

12. 零信任接入评估

对于面向内部员工的 PC 端系统，应收缩到内网，通过堡垒机或者零信任进行访问；如下情况不建议对接零信任：

（1）如有敏感数据的下载和批量导出的系统，不建议对接零信任，而应直接放在堡垒机里面；

（2）重要信息系统（支持经营机构关键业务功能、如出现异常将对证券期货市场和投资者产生重大影响的信息系统）；

13. 统一认证评估

系统如面向公司员工内部使用且有登录认证需求的，需接入公司SSO进行统一单点登录认证。

14. 日志采集

系统日志应接入公司统一日志平台，日志平台提供日志统一采集存储及搜索分析能力。

15. 权限稽核

所有涉及用户角色及有权限分配管理能力的系统需接入权限稽核系统。

16. 等级保护基本信息

参与架构评审的系统应提供等级保护的基本信息，包括但不限于以下信息：

（1）敏感数据存储描述

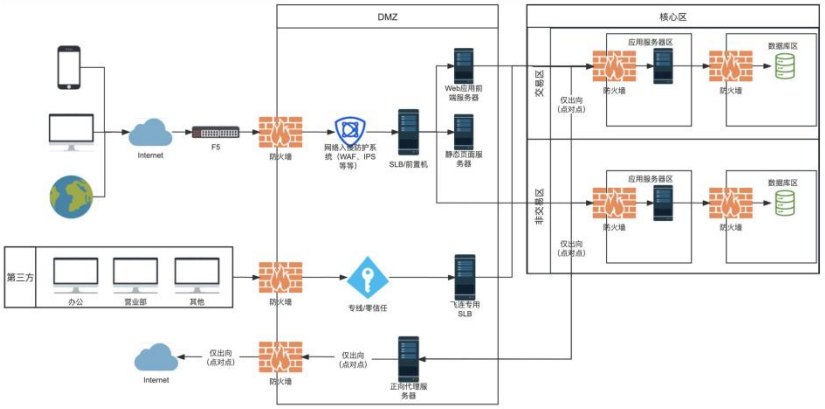
（2）等级保护定级评定

17. 附录

17.1 架构图示例

部署架构图需要尽量清晰体现组件的部署区域及边界，应用组件间的调用链路，每台服务器部署的组件应用。下图为公司网络架构规

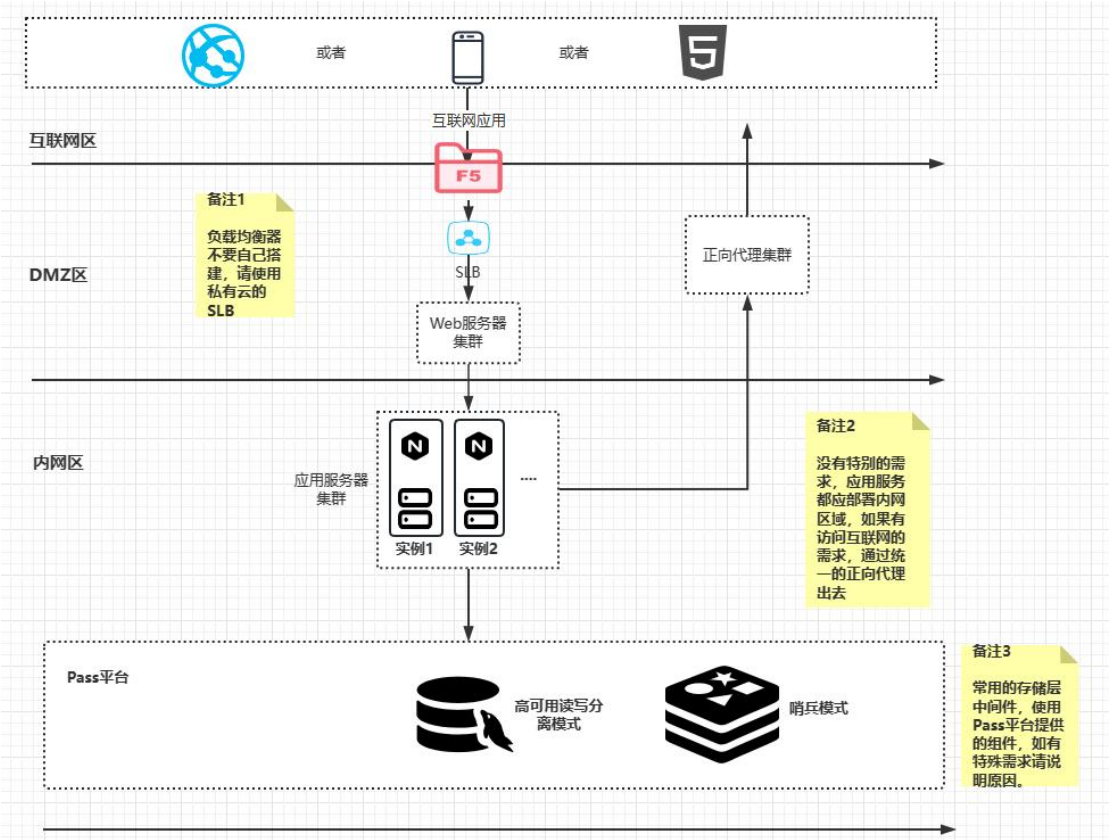
划图及要求：



注：
核心区内应用服务区和数据库区为广义表示，根据安全等级（现状已有）细分为不同安全级别的应用服务区和数据库区。默认同一类型区域内高安全级别可访问安全级别区域，低安全区域无法访问高安全区域。

1、边界设备（安全防护设备、负载均衡设备等）放置DMZ区域；互联网访问纯静态（无业务和数据交互）服务器放置DMZ区域；web应用前端服务器/SLB/前置机放DMZ区域；（提供互联网服务的机器，后台应用服务器禁止直接放置DMZ区）。
2、DMZ区也分为交易区和非交易区，不同区之间SLB/前置机等不互通。
3、核心区划分为交易区和非交易区两大区域。两大区域内再细化分为应用服务器区和数据区，应用服务器区和数据区边界均放置防火墙，默认出入向流量全关，根据业务或者访问需求进行点对点策略开放，禁止Any对Any情况。
4、DMZ区服务器禁止直接访问数据库区。同一应用服务器区或数据区内的网络不再经过防火墙，可互通。
5、应用服务器区需要第三方接入（营业部、上交所等等），通过专线或零信任等进行点对点策略开放接入访问。
6、内网服务器（核心区）单向出网访问，通过DMZ区正向代理服务器转发，点对点策略进行出网访问。

下面用一个简单的Web应用作为说明，实际架构图请按各自系统来，本示例只是为了说清楚部署/架构图中要体现上面的要素。



17.2 中间件选型建议

在进行中间件选型时，应遵循以下原则以确保技术选型与公司的战略目标和安全标准相一致。

原则一：优先利用现有私有云平台资源

对于需要数据库、缓存等中间件支持的系统，应首先考虑利用私有云平台提供的PAAS服务。这种做法不仅能够减少基础设施的重复建设，还能确保服务的一致性和可靠性。

原则二：优先选择技术自主可控的数据库解决方案

在自主研发系统的数据库选型过程中，优先推荐采用技术自主可控的数据库。对于可能涉及的外购系统，如果计划使用开源数据库，必须首先与供应商沟通，探讨切换至国产版本的可能性。若无法切换，必须提供充分的理由，并明确指出所使用的数据库类型及版本，同时确保其符合公司的安全基线。

原则三：Redis的具体使用建议

对于并发量较小的应用，建议采用Redis的哨兵模式来确保高可用性。而对于面临高并发挑战的应用，则应考虑部署Redis集群模式，以提供更强大的数据处理能力和扩展性。

17.3 架构冲突处理建议

在架构评审过程中，经常会出现一些IT系统的架构因为开发人员设计考虑不周、业务需求急迫、供应商系统定开不便等原因和架构评审规范存在冲突的情况。下面是一些冲突处理的原则。如果以下处理原则还是不能解决这些冲突，可以提交公司IT委员会进行决议。

优先考虑安全和合规性：对于高危漏洞以及合规风险，必须在系统上线前得到彻底解决。对于存在中低风险漏洞的系统，因为业务需求急迫等原因急于上线的，需由安全团队深入评估影响。

可维护性和扩展性：系统的架构应满足可维护和可扩展，业务需求紧迫的可以临时上线，但应将可维护和可扩展作为系统未来改造的重点。

高可用：系统的架构应该具有高可用的特性。不具备高可用特性的系统，需要在架构评审环节详细评估业务对系统宕机的可容忍性。

中间件选型：系统采用中间件的最终选型如果和默认选型建议有冲突的，需要在架构评审环节说明原因，评审委员需深入评估选型的合理性。

系统容量配置：初次上线的 IT 系统往往难以准确评估所需的资源配置，如 CPU、内存和存储等。运维人员常见的策略是申请大于实际需求很多的资源容量，导致出现资源浪费严重的情况。对于不能准确评估资源配置的系统，需采取资源节约的原则，同时借助监控，资源不够时再及时申请扩容。