SDL-软件安全设计初窥

— safer applications begin with secure design

— Cryin@insight-labs.org

摘要：本文详细介绍微软软件安全开发生命周期（SDL）相关概念，并讨论要遵循 SDL 过程所应执行的各种安全活动，其中着重对软件安全设计的原则进行探讨。并对 STRIDE 威胁建模方法进行深入介绍。

安全开发生命周期（SDL）是一个帮助开发人员构建更安全的软件和解决安全合规要求的同时降低开发成本的软件开发过程。 安全应用从安全设计开始，软件的安全问题很大一部分是由于不安全的设计而引入的，微软用多年的经验总结出了安全开发生命周期（SDL），并提出了攻击面最小化、STRIDE 威胁建模等多种方法辅助安全人员对软件进行安全设计。安全设计对于软件安全的重要性尤为可见。

一. 前言

1.1 SDL 介绍

安全开发生命周期（SDL）即 Security Development Lifecycle，是一个帮助开发人员构建更安全的软件和解决安全合规要求的同时降低开发成本的软件开发过程。 自 2004 年起，微软将 SDL 作为全公司的计划和强制政策，SDL 的核心理念就是将安全考虑集成在软件开发的每一个阶段:需求分析、设计、编码、测试和维护。从需求、设计到发布产品的每一个阶段每都增加了相应的安全活动，以减少软件中漏洞的数量并将安全缺陷降低到最小程度。安全开发生命周期 (SDL)[1] 是侧重于软件开发的安全保证过程，旨在开发出安全的软件应用。

1.2 SDL 安全活动

简单来说，SDL 是微软提出的从安全角度指导软件开发过程的管理模式，在传统软件开发生命周期（SDLC）的各个阶段增加了一些必要的安全活动，软件开发的不同阶段所执行的安全活动也不同，每个活动就算单独执行也都能对软件安全起到一定作用。当然缺少特定的安全活动也会对软件的安全性带来影响。

1. TRAINING	2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE	7. RESPONSE
1. Core Security Training	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan	Execute Incident Response Plan
	3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review	
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive	

图 1：微软 SDL 安全活动简图

我曾今有幸参加过微软安全专家 Michael Howard 及 Taha Mir 关于 SDL 及威胁建模的培训，作为《软件安全开发生命周期》一书的作者，Michael Howard 不只一次强调，安全培训是 SDL 最核心的概念，软件是由设计人员设计，代码是有开发人员编写。同样，大部分软件本身的安全漏洞也是由设计及编码人员引入，所以对软件开发过程中的技术人员进行安全培训这点至关重要。

[1] https://www.microsoft.com/en-us/SDL/process/design.aspx

可以看到在整个SDL周期中，除了安全培训这项活动，还在软件发布后增加了安全应急响应的相关活动，而目前国内大多数公司目前已经基本上具备了安全应急响应的活动和职能部门，同时包括安全编码规范、代码审计、渗透测试等安全活动也都已经基本具备甚至个别企业已经比较成熟。但在软件设计阶段的安全活动则相对较少，据我了解仅个别大型跨国企业才拥有安全设计等相关的安全活动。而根据微软多年的实践和经验，软件的安全问题很大一部分是由于不安全的设计而引入的。在设计阶段造成的安全缺陷在后期修复的成本和时间都相对较高。STRIDE威胁建模的创始人之一Taha Mir曾说过“safer applications begin with secure design”，即安全应用从安全设计开始，相应的微软SDL也提出了若干核心的安全设计原则，并提出了如攻击面最小化、STRIDE威胁建模等多种方法辅助安全人员对软件进行安全设计，本文就针对当前国内企业在软件设计阶段安全活动发展相对欠缺的安全设计进行探讨。

二. 安全设计核心原则

SDL安全设计核心原则:

- Attack Surface Reduction：攻击面最小化
- Basic Privacy: 基本隐私
- Least Privilege: 权限最小化
- Secure Defaults: 默认安全
- Defense in Depth：纵深防御
- Threat Modeling：威胁建模

2.1 攻击面最小化

攻击面是指程序任何能被用户或者其它程序所访问到的部分，这些暴露给用户的地方往往也是最可能被恶意攻击者攻击的地方。

攻击面最小化即是指尽量减少暴露恶意用户可能发现并试图利用的攻击面数量。软件产品的受攻击面是一个混合体，不仅包括代码、接口、服务，也包括对所有用户提供服务的协议。尤其是那些未被验证或者远程的用户都可以访问到

的协议，安全人员在攻击面最小化时首先要对攻击面进行分析，攻击面分析就是枚举所有访问入库、接口、协议一剂可执行代码的过程，从高层次来说，攻击面分析着重于:

- ✓ 降低默认执行的代码量
- ✓ 限制可访问到代码的人员范围
- ✓ 限定可访问到代码的人员身份
- ✓ 降低代码执行所需权限

常见的攻击面分析技巧如下表:

Higher Attack Surface	Lower Attack Surface
On by default	Off by default
Open socket	Close socket
UDP	TCP
Anonymous access	Authenticated user access
Constantly on	On as needed
Internet accessible	Local subnet accessible

表 1 攻击面分析常用技巧

攻击面最小化在微软的应用实践示例:

Windows	RPC 需要认证、防火墙默认打开
iis6.0、7.0	使用 Network service 权限运行，默认关闭.
Sql server 2005 /2008	xp_cmdshell 存储过程默认关闭，默认不开放远程链接
VS2005/2008	Web server 和 sql server 默认仅本地访问

表 2 攻击面最小化微软实践示例

2.2 基本隐私

用户使用软件时无可避免个人信息被收集、使用甚至分发，企业则有责任和义务建立保护个人信息的保护措施，抵御敌对攻击行为，确保用户基本隐私的安全性。隐私安全是建立可信任应用程序的关键因素。

在软件设计时考虑用户基本隐私的必要性及意义有:

- ✓ 履行法律规定和义务
- ✓ 增加客户的信赖
- ✓ 防止堵塞部署

对于特殊的软件或者全球性的产品，设计人员需要明确软件的行为及针对人群。尤其要考虑当地国家的法律法规，如美国儿童网路隐私保护法COPPA(Children's Online Privacy Protection Act)等，企业在开发产品、服务时有必要制定明确的隐私准则，对获取、记录用户隐私的相关产品需有明确的要求和指导建议。

Tips:

- ✧ 只收集程序必须用到的隐私数据，并明确告知用户并征得用户同意;
- ✧ 微软对于用户隐私数据如密码、口令等均需要加密存储，最低要求是sha256+salt，对于更高要求的则使用 PBKDF2 算法加密存储;

2.3 权限最小化

如果一个应用程序或网站被攻击、破坏，权限最小化机制能够有效的将潜在损害最小化。常见的权限最小化实践如:

- ✓ 普通管理员/系统管理员等角色管理
- ✓ 文件只读权限/文件访问权限等访问控制
- ✓ 进程/服务以所需最小用户权限运行

在进行软件设计时，安全设计人员可以评估应用程序的行为及功能所需的最低限度权限及访问级别，从而合理分配相应的权限。如果程序特定情况必须要较高级别的权限，也可以考虑特权赋予及释放的机制。即便程序遭到攻击，也可以将损失降到最低。

Tips:

- ✧ Windows 系统中网络进程、本地服务、用户进程的权限都较低且互相独立，分别为 NETWORK SERVICE、LOCAL SERVICE、user 权限，只有核心的重要进程实用 SYSTEM 权限;

- ✧ 最新版本的 Office 程序打开不可信来源的文档时，默认时不可编辑的，同时也是默认不可执行代码的，即使存在缓冲区溢出漏洞，也不会执行 shellcode 等恶意代码;

2.4 默认安全

默认安全配置在客户熟悉安全配置选项之前不仅有利于更好的帮助客户掌握安全配置经验，同时也可以确保应用程序初始状态下处于较安全状态。而客户可根据实际使用情况而决定应用程序安全与隐私的等级水平是否降低。

Tips:

- ✧ 在 Win 7 之后的 Windows 操作系统中，DEP(数据执行保护)默认是开启的。用户可设置选项改变 DEP 的状态;
- ✧ Win 10 默认启用安全防护软件 Windows Defender，用户可选择关闭;

2.5 纵深防御

与默认安全一样，纵深防御也是设计安全方案时的重要指导思想。纵深防御包含两层含义：首先，要在各个不同层面、不同方面实施安全方案，避免出现疏漏，不同安全方案之间需要相互配合，构成一个整体；其次，要在正确的地方做正确的事情，即：在解决根本问题的地方实施针对性的安全方案。

纵深防御并不是同一个安全方案要做两遍或多遍，而是要从不同的层面、不同的角度对系统做出整体的解决方案。

Tips:

- ✧ 针对 XSS 的防护，除了要对用户输入的特殊符号进行过滤，还要区分是否是富文本进而进行相应编码操作，在输入时过滤的同时在输出时也进行过滤操作。
- ✧ 即使做了十足的过滤、编码等安全防护，为了更一步确保缓解 XSS 攻击，Internet Explorer 6 SP1 为 Cookie 引入了一个新属性，这个属性规定，不许通过脚本访问 cookie。Web 站点程序对 Cookie 启用 HTTP-Only

属性后，可确保即使发生 XSS 攻击，也可以阻止通过脚本访问 Cookie 的操作。

2.6 威胁建模

威胁建模是一种分析应用程序威胁的过程和方法。这里的威胁是指恶意用户可能会试图利用以破坏系统，和我们常说的漏洞并不相同。漏洞是一个特定的可以被利用的威胁，如缓冲区溢出、sql 注入等。

作为 SDL 设计阶段的一部分安全活动，威胁建模允许安全设计人员尽在的识别潜在的安全问题并实施相应缓解措施。在设计阶段把潜在的威胁发现有助于威胁的全面和更有效的解决，同时也有助于降低开发和后期维护的成本。威胁建模的一般流程如下:

- ✓ 与系统架构师及设计人员沟通，了解设计详情
- ✓ 使用成熟的威胁建模方法分析当前设计潜在的安全问题
- ✓ 提出安全建议及对潜在威胁的缓解措施
- ✓ 对安全设计进行验证并对整个设计方案进行回顾并再次确认

微软使用的威胁建模方法是 STRIDE 威胁建模方法。为了便于安全人员快速便捷的进行威胁建模，微软开发基于 STRIDE 威胁建模方法的 SDL Threat Modeling Tool[2]威胁建模工具，该工具可以帮助安全人员画数据流图、分析威胁、生成并导出威胁建模报告。

三. STRIDE 威胁建模方法

3.1 STRIDE 介绍

STRIDE 威胁建模是由微软提出的一种威胁建模方法，该方法将威胁类型分为 Spoofing（仿冒）、Tampering（篡改）、Repudiation（抵赖）、Information Disclosure（信息泄漏）、Denial of Service（拒绝服务）和 Elevation of Privilege

[2] https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx

（权限提升）。这六种威胁的首字母缩写即是 STRIDE，STRIDE 威胁模型几乎可以涵盖目前绝大部分安全问题。此外，STRIDE 威胁建模方法有着详细的流程和方法。

3.2 威胁建模流程

STRIDE 威胁建模的一般流程如下:

- ✓ 绘制数据流图
- ✓ 识别威胁
- ✓ 提出缓解措施
- ✓ 安全验证

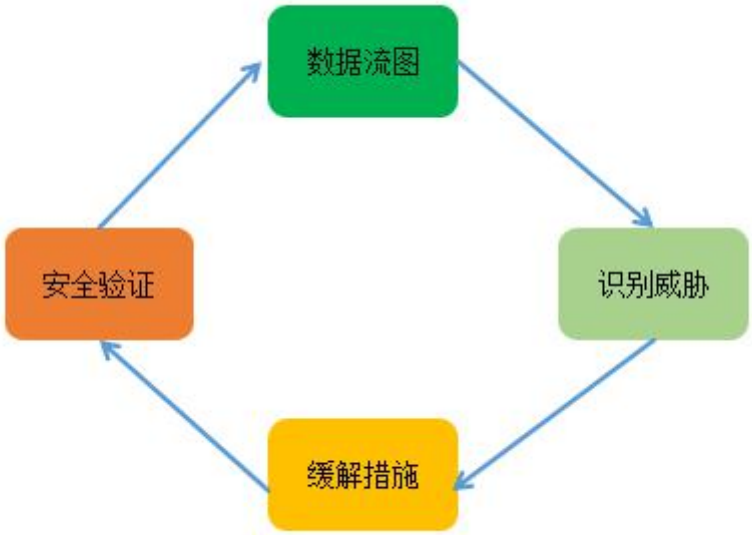

图 2: STRIDE 威胁建模流程

3.2.1 数据流图

数据流图(Data Flow Diagrams)包含外部实体(External Entity)、处理过程(Process)、数据流(Data Flow)、数据存储(Data Store)，安全人员与系统架构师及设计人员沟通，了解设计详情并画出数据流图后还需要标注信任边界（Trust Boundary），针对简单的 Web 应用的数据流图如下:

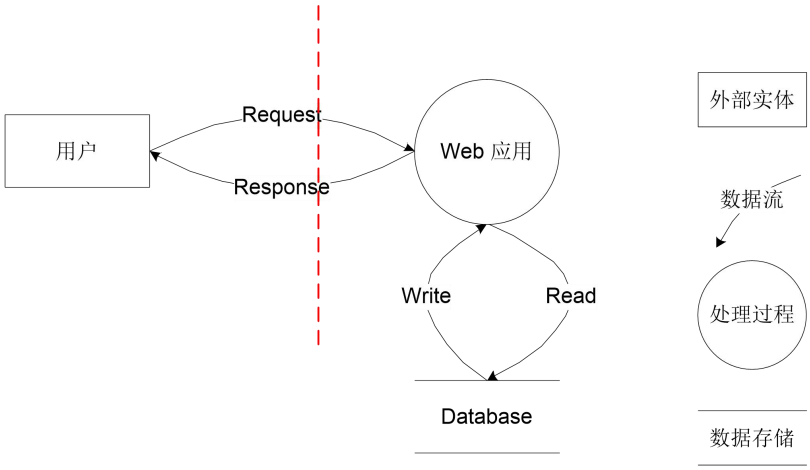

图 3: 数据流图示例及元素类型

3.2.2 识别威胁

STRIDE 威胁建模方法已经明确了每个数据流图元素具有不同的威胁，其中外部实体只有仿冒（S）、抵赖（R）威胁，数据流只有篡改（T）、信息泄露（I）、拒绝服务（D）威胁，处理过程有所有六种（STRIDE）威胁，存储过程有篡改（T）、信息泄露（I）、拒绝服务（D）威胁，但如果是日志类型存储则还有抵赖（R）威胁。具体可以对照如下表格进行威胁识别:

元素	S	T	R	I	D	E
外部实体	√		√			
处理过程	√	√	√	√	√	√
数据存储		√	?	√	√	
数据流		√		√	√	

表 3 数据流图元素对应的不同威胁

3.2.3 缓解措施

根据不同的数据流图元素及威胁，相应的缓解措施也不相同。如本文示例数据流图中外部实体用户的仿冒威胁，其缓解措施简单来说就是对用户身份进行认证。对于一个 Web 应用来说，缓解仿冒威胁不仅需要较强的认证机制，还需要

防止恶意攻击者用暴力破解、口令猜测等方法绕过认证从而造成仿冒用户的威胁。如果笔者来提出该用户仿冒威胁的缓解措施的话，详细措施如下:

- ✓ 对用户访问进行帐号密码、证书等身份认证;
- ✓ 用户帐号密码认证过程中，如果出现三次密码错误，则增加验证码机制。输入验证码且正确再进行身份认证;
- ✓ 当用户认证 5 次后仍然验证失败，则在 30 分钟内禁止该帐号登录;
- ✓ 用户密码必须包含数字、字母及特殊字符，且长度在 8 位以上，如果业务安全需要则增加密码过期机制，每隔 6 个月提醒用户修改密码;

在提出缓解措施时，有的时候不仅要考虑安全问题，同时也要考虑软件的易用性，所以不同的威胁，不同的应用场景。其缓解措施也要随之而改变以提高应用安全的同时也能给用户带来较好的交互体验。

微软对于常用的威胁给出了其常用的标准缓解措施，并在具体实施时已将常用的缓解方案及措施集成为独立的解决方案或者代码模块。可以方便同类应用直接使用。

威胁类型	**缓解措施**	**技术方案**
仿冒(S)	认证	Kerberos 认证 PKI 系统如 SSL / TLS 证书 数字签名
篡改(T)	完整性保护	访问控制 完整性校验
抵赖(R)	日志审计	强认证 安全日志、审计
信息泄露(I)	保密性	加密 访问控制列表
拒绝服务(D)	可用性	访问控制列表 过滤 热备份
权限提升(E)	授权认证	输入校验

		用户组管理 访问控制列表

3.2.4 安全验证

在威胁建模完成后，需要对整个过程进行回顾，不仅要确认缓解措施是否能够真正缓解潜在威胁，同时验证数据流图是否符合设计，代码实现是否符合预期设计，所有的威胁是否都有相应的缓解措施。最后将威胁建模报告留存档案，作为后续迭代开发、增量开发时威胁建模的参考依据。

四. 总结

SDL 的核心理念是将安全考虑集成在软件开发的每一个阶段:需求分析、设计、编码、测试和维护。从需求、设计到发布产品的每一个阶段每都增加了相应的安全活动，以减少软件中漏洞的数量并将安全缺陷降低到最小程度。本文重点介绍了设计阶段的安全活动指导思想及 STRIDE 威胁建模，但 SDL 的其它阶段的不同安全活动也同样对软件安全有着重要影响。同时本文介绍的安全设计原则仅为指导思想，安全设计人员还需要掌握一定的安全攻防知识，具备一定的安全攻防经验才能更好的设计出安全的方案及软件应用。另外根据笔者经验，在实际的安全设计工作中，对于不同软件及应用场景其面临的安全问题也不同。随着互联网时代发展，目前已经不在是单纯的软件时代了，类似移动端应用、智能硬件、云端、大数据平台等新形态的应用都面临的自身特有的安全问题。安全设计人员要考虑的也要更多，但安全设计的核心原则还是相差无几。由于篇幅及笔者经验有限，本文所述如有不妥之处可以与笔者联系交流。

五. 参考文献

[1] https://www.microsoft.com/en-us/SDL/process/design.aspx

[2] http://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx

[3] Introduction_to_Threat_Modeling

[4] Simplified Implementation of the SDL

[5] https://github.com/Cryin/Paper