链滴

人、工具、业务代码的之间的关系

作者：nanolikeyou
原文链接：https://ld246.com/article/1536722100173
来源网站：链滴

静态代码分析和人工审查的区别

当前技术层次上来看，源码市场上无非也就是那三四种种，Foritify，AppScan for code, Checkmarx coverity，google的叫做grrit\ errorprone\ shipshape，大公司可能有自己的工具，但是没商业化者没放出来，是外界没法接触到。Fortify是唯一向导界面直接出现j2ee的，别的都没有。java支持还算比较好的，框架之类的。coverity对c、c++的支持较好，但是其他语言就弱很多了，尤其php，框基本不怎么支持。国内新出来一款codepecker的软件，效果一般。实际公众中总是在误报、漏报、确配置三件事情上忙活，走开发规则、确保准确的配置、完善检测模型这三板斧。当然本身大量的误还是需要人工来verify，manual code review仍然是主题，尤其是对于增量代码更新来说，而且有些西天生更适合manual code review，有些则更适合automatic code scan，SDL中还是最好自动化结人工互补。人，工具，代码等这些之间，需要结合公司具体情况做权衡和平衡。SDL团队初建，没足的人手，那么人这块就没法保证，以工具优先，同样，代码的覆盖率同样也无法保证，只能说慢慢一步来。

代码防御原则

理论上来说，安全防御是需要进行纵深防御，而不仅仅是单单最外层一层进行防御。 首先最外层肯定要处理这些数据验证问题，其次各层系统之间的信任问题，最好也需要有相关的安全措施。但是由于本等，Secure in depth这个，实际和理论差距很大，SQL注入问题直接由数据库实现，但是最终由各种原因，数据库这边对SQL注入并未处理，仍然坚持自己只坚持数据库的功能，注入这种问题并不算处理,安全的问题，很大程度上是信任的问题。所以中间件说他信任上层应用的数据。那么实际上要研究下上层应用是否都是值得信任的。如果是复杂的系统，甚至有提供给第三方的，很难保证所有的界输入都已经被妥善处理 。但是对于中间件和数据库来说无法区分执行的命令是攻击命令还是正常命。这时候rasp这种东西就上了，这个情况很像消防制度、烟雾报警器、消防员灭火三者的关系。因为于不同的应用来说需要执行的命令是不同的，比如一个数据库管理的软件，就是要传送sql指令的，时候就没法过滤或者参数化。所以有时候这理论和实际理想情况差别很大。只能说是结合具体场景具分析。一般也就是最多在前面2、3层应用都要求对输入进行处理，再后面的系统，也是没法再要求了如果外层仅仅是个别自己可信任的应用提供数据给中间件，那么就实际效果来看，中间件不愿意处理说的过去。但是如果外层太多，就很难控制的住了。而且就咱们安全来说，能集中化处理的，就尽量中化处理，不然不管是工作量，还是出问题的概率，都不好办了。举个最简单的例子，某个系统对外供一个接口，那么如果外层就1、2个系统用，那么实际效果来看，那1、2个系统中都对数据进行了正验证，接口这边即使不验证，也是安全的。但是如果外层有几十个，甚至更多应用来用，或者说以后用的数量扩展了，那么接口这边如果不做集中化的验证，那就很复杂了（《阿里巴巴java开发手册》求对接口全部做验证》）。灵活，易用和安全，侧重点取个平衡。

SDL推行

如果要是没有上面高层的强力支持，很难推下去的，阻力太大。开发基本不关心安全不安全，关心的自己的任务能不能即使完成，pm也是更关注项目进度。而安全通常会给项目带来额外的资源消耗，论是人力还是时间。而互联网公司有很鲜明的特点： 1.
开发周期快 2. 产品开发多 3. SDL人员少 完整
SDL流程，恐怕是行不通的，SDL基本上可以说是100%会被整个开发团队反感，我们工作接触多的co e review和开发的code review还不是一回事，门槛更高，而且由于工具的太过于昂贵，一般公司没设立相应的职位或者职位继续不下去，从流程上看可以看看http://www.owasp.org.cn/owasp-projec/download/owasp-samm samm源码审计推进分步骤，似乎也挺适合互联网公司的，是个非大型专转件公司版的微软SDL，其实流程和要求和SDL已经差别不大了，还是1.
人员要求高 2. 成本大，和SDLC一样的缺点，它好在于有三个等级，可以慢慢一步步提
，而SDL则直接一步到位。目前的情况，其实SDL很难做到位，虽然一直在做，疲于数据安全检测、洞审计、修复、闭环。主要是项目周期短，发布快，项目又多，安全人员少，只能尽量走自动化路线但是像漏洞和代码分析，架构设计安全审计这些，好多东西自动化目前还无法办到。SDL除非是公司足够高的层次支持，使之成为像软件开发流程那样的基本工作流程才行。这里分为两种情况，将安全为产品的基本需求特性，卖软件的厂商，另外一种是开发的产品自用，两者对于安全性的初心是不一

的。

国内外区别

国内做安全是与国外相比是欧洲他们那边罚的太狠了，欧洲是相当于根据漏洞产生的数据、隐私安全面的影响，继而影响推动提升源码的安全性，一则是法律和罚款，人家有官方机构，每年都要进行PC的审计，二则是普通人对于隐私、自由方面十分看重，国内的隐私保护其实都是为了商业，似乎并不为了客户和法律。而且国外还有各种审计机构要审计，所以不得不重视，不得不严格，代码审计毕竟是漏洞挖掘工具，发现问题进而验证、推动整个占了安全人员很大的时间。至于建模、攻击面分析，发人员又不会，所以等于没有。这方面是国内外都遇到的问题。

在GDPR方面，分为隐私要求和安全要求：在评估阶段

- 发现并分类个人数据资产和受影响的系统
- 识别访问风险，通过设计支持隐私。

在设计阶段

- 创建安全参考架构
- 设计适合风险的TOM（如加密，伪化，访问控制，监控）

在数据控制层面

- 实施隐私增强控制（例如，加密，标记，动态屏蔽）
- 实施安全控制; 缓解访问风险和安全漏洞