第34卷 第20期 计 算 机 工 程 2008年10月
Vol.34 No.20 Computer Engineering October 2008
·软件技术与数据库· 文章编号：1000—3428(2008)20—0066—03 文献标识码：A 中图分类号：TP311

基于数据融合的源代码静态分析

陈 超，李 俊，孔德光
(中国科学技术大学自动化系，合肥 230027)

摘 要：采用数据融合技术对源代码进行静态分析，实现可扩展的原型系统。对现有静态分析工具的分析结果进行解析和数据融合，并对相应的参数进行估计。为便于读取和分析输出结果，采用 XML 格式输出结果。对常用网络软件进行测试，结果表明相对于单个源代码分析工具，该技术有效地降低了误报率和漏报率。

关键词：漏洞检测；静态分析；数据融合；误报率；漏报率

Source Code Static Analysis Based on Data Fusion

CHEN Chao, LI Jun, KONG De-guang
(Department of Automation, University of Science & Technology of China, Hefei 230027)

【Abstract】This paper presents a new source code static analysis technology based on data fusion and implements an extended prototype system. It interprets and makes data fusion on the outcome of different static analysis tools and evaluates the corresponding parameters. To read and analyze easily, the outcome of the system adopts XML form. After testing common network software, experimental result shows that compared with single source code analysis, this technology can reduce false positives and false negatives effectively.

【Key words】vulnerability detection; static analysis; data fusion; false positives; false negatives

1 概述

软件静态分析主要是对源代码进行静态安全性分析，即通过在程序运行前分析源代码以尽早发现源程序存在的脆弱性和安全问题。评价静态分析工具的好坏有 2 个重要指标：(1)漏报率(false negatives)：程序中含有安全问题但没有找出的比率；(2)误报率(false positives)：报出程序原本不存在的安全问题的比率[1-2]。

对于已有的源代码静态分析工具，虽然各自的分析机理不同，但都有一个共同弱点，产生大量的误报和漏报[1-2]。主要由以下 2 点原因造成：

(1)静态分析本身具有局限性。Rice定理已经证明静态分析问题在最坏情况下是不可判定问题[1]；

(2)为简化设计，大多数工具的分析模型采用了保守策略，即采用了流不敏感和上下文不敏感的分析方式。

为了有效地降低漏报率和误报率，本文提出一种基于数据融合的源代码静态分析方法。

2 数据融合

不同源代码工具的工作原理和运行机制有很大不同。以词法工具为例，漏洞规则库的不同、定义的漏洞等级不同，都会使分析结果差异很大。文献[3]对 3 个具有代表性的词法工具Its4[4], Rats[5], Flawfinder[6]用相同的软件包进行测试。试验结果表明：Flawfinder发现的漏洞最多，但并不是完全覆盖了Rats, Its4 发现的漏洞部分；Rats和Its4 两者发现的漏洞数目相差不大，但还是有很多不相交的部分。由此可见，每个工具都有各自的优势，充分运用其优势，就能有效提升工具的性能和效率。

基于上述思想，本文把数据融合思想与源代码分析相结合，即将各个工具分析结果通过适当的方式进行分析融合，使漏洞结果之间互为补充。一方面，降低源代码静态分析的漏报率，尽可能多发现存在的漏洞；另一方面，通过对融合的结果进行等级评估，漏洞的误报率也会降低。

本文的研究主要基于 2 个假设：

(1)一个被多数工具识别的漏洞极有可能是漏洞；

(2)被所有工具误报的一个漏洞出现的概率相对较低。

设有 n 个静态分析工具，第 i 个静态分析工具的可信赖度权值为 $w(i)$，正确报告率是 $TP(i)$，则误报率是 $FP(i)=1-TP(i)$，漏报率是 $FN(i)$。$w(i)$的大小与 $FP(i)$和 $FN(i)$的大小成反比，即如果某个静态分析工具的 $FP(i)$和 $FN(i)$的值低，则它的 $w(i)$值就高。

如果不同静态分析的结果有可比性，即通过适当的转化变为一种统一的中间格式，则可以定义每条漏洞对应的估分随机变量 X，对某条漏洞为真实漏洞的概率进行评估与预测。由于多数工具识别的漏洞比少数工具识别的漏洞更有可能是漏洞，因此每条漏洞对应的估分随机变量 X 定义准则如下：(1)多个工具同时识别的漏洞的估分值高；(2)本身漏洞危险等级高则漏洞条目估分值高；(3)不同工具的分析结果以自身可信赖度权值 $w(i)$来体现估分的贡献度。

对于某条漏洞 vul 而言，第 i 个工具的估分变量 $s(i)$ ($1\leq i\leq n$)将该工具 i 给漏洞 vul 条目的危险等级的高低映射到其给出的估分值上。因为相对于危险等级较低的漏洞，危险等级较高的漏洞更可能是漏洞，所以应设定更高的优先级别。Flawfinder 漏洞的危险等级分为 1~5 个等级。Rats 漏洞的危

基金项目：国家“863”计划基金资助项目(2006AA01Z449)
作者简介：陈 超(1983－)，男，硕士研究生，主研方向：信息安全；李 俊，副教授；孔德光，硕士研究生
收稿日期：2007-11-15 **E-mail：**jackchen@mail.ustc.edu

险等级分为 3 类：High, Medium, Default；以 5 分等级进行映射，则：*f*(High)=5, *f*(Medium)=3, *f*(Default)=1。Its4 的漏洞的危险等级分为 4 类：Some Risk, Risky Very Risky, Urgent；以 5 分等级进行映射，则：*f*(Some Risk)=1, *f*(Risky) =2, *f*(Very Risky)=4, *f*(Urgent)=5。

$w(i)$表示第 i 个工具对分析结果的可信赖度权值($1\le i\le n$)，$r(i)$表示第 i 个工具的实际估分($1\le i\le n$)。$vul(i)$=1 表示漏洞 $vul(i)$被第 i 个工具识别($1\le i\le n$)，满足：

$$E(X)=\sum_{i=1}^{n} r(i)w(i)$$

$$r(i)=\begin{cases} s(i) & vul(i)=1 \\ 0 & \text{其他} \end{cases}$$

$$\sum_{i=1}^{n} w(i)=1$$

$w(i)$值的大小可用 2 种方法进行调节：

(1)由用户交互指定；

(2)根据误报率和漏报率来进行评估。

设 X_i^j 和 Y_i^j 分别代表工具 i 对软件包 j 测试得到的误报率和漏报率。构建一个二维空间 $\{(X_i^j,Y_i^j): X_i^j\ge 0, Y_i^j\ge 0\}$。误报率和漏报率越低，源代码工具的可信赖度越高，即 $w(i)$值越高。工具在最好的情况下，误报率 *FP* 和漏报率 *NP* 为 0，在最坏的情况下，误报率 *FP* 和漏报率 *NP* 为 100%。基于这些特性，用欧式测度作为判定工具性能的主要标准。欧式测度代表软件包 j 工具 i 的可信赖度权值 $wj(i)$。$w(i)$代表工具 i 的对应于所有软件包的可信赖度权值，满足：

$$w^j(i)=\sqrt{\left(X_i^j-1\right)^2+\left(Y_i^j-1\right)^2}$$

$$w(i)=\frac{\sum_{j=1}^{m} w^j(i)}{\sum_{i=1}^{i=n}\sum_{j=1}^{j=m} w^j(i)}$$

估分变量 X 的数学期望 $E(X)$反映了 n 个工具对某条漏洞的分析结果，即数据融合后对某条漏洞的评估值。对于软件源码包 wu-ftpd-2.5.0 中的同一条漏洞 vul，以 Its4, Rats, Flawfinder 这 3 个工具为例，不同工具解析结果如下：

(1)Rats 工具分析结果

wu-ftpd-2.5.0/src/extensions.c:263: High: sprintf

(2)Its4 工具分析结果

wu-ftpd-2.5.0/src/extensions.c:263:(Very Risky) sprintf Non-constant format strings can often be attacked. Use a constant format string

(3)Flawfinder 工具分析结果

wu-ftpd-2.5.0/src/extensions.c:263 [4] (buffer) spritf: does not check for buffer overflow

在数据融合过程中，以误报率和漏报率进行评估的方法，得出 $w(1)$=0.32, $w(2)$=0.32, $w(3)$=0.36，其中，$r(1)$=5, $r(2)$=4, $r(3)$=4，得出 $E(X)$=4.32。

3 数据融合源代码检测的系统设计

基于上述工作原理，本文实现了一个源代码静态分析工具的原型系统(简称 ISA)，该原形系统具有以下特点：

(1)集成多个分析工具的优点，不同结果之间相互补充；

(2)加入用户交互设定工具的可信赖度；

(3)采用 XML 格式形式输出，便于描述和解析以及数据的共享。

系统框架设计如图 1 所示。

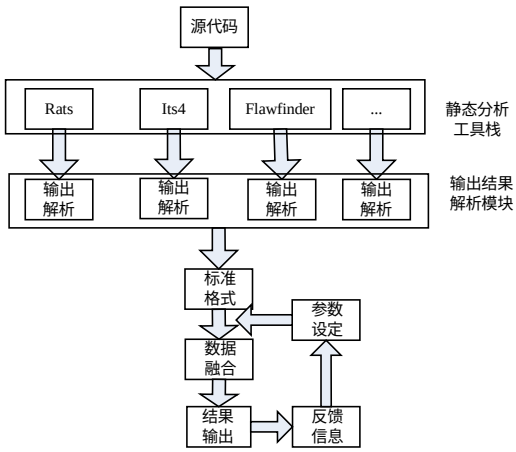

图 1　ISA 系统架构

(1)将程序源代码使用静态分析工具栈分析得到不同分析输出结果。静态分析工具栈是一个集成多个分析工具的框架。用户可根据相同的原理把其他静态分析工具加入到此栈中。

(2)输出结果解析主要方式是词法分析，针对工具分析结果输出的特定格式提取有效漏洞信息，以便后续数据融合模块使用。

(3)上述解析后转换成格式统一的数据结构形式，即一个 6 元组序列：(漏洞所在文件名，出现行号，危险等级，漏洞类型，引起出错的函数，漏洞危险性估分)。

(4)对若干个工具的分析结果，按前文原理中的准则进行数据融合。

(5)数据融合以后，调用相应漏洞集的排序模块，所有漏洞按照估分由高到低排列，并可以根据用户的交互或前文给定的算法指定信赖度权值，保留估分较高的漏洞以格式化的形式输出。一个漏洞条目 XML 描述如下所示。

```
<Vulnerabilities>
<Item>
        <File_in>extentions.c</File_in>
        <Row_NO>183</Row_NO>
       <Risk>5</Risk>
       <Type> bufferoverflow</Type>
       <Fun_name> printf</Fun_name>
        <Score> 4.32 </Score>
</Item>
</Vulnerabilities>
```

(6)将上述分析的结果与实际的漏洞分析结果进行对比，发现漏洞分布情况的不同，将表征两者差异的信息通过参数训练模块反馈给数据融合模块，调节 $w(i)$的权值，重新对 n 个结果集的数据进行融合，此过程迭代进行，直到将参数 $w(i)$训练到一个相对合适的比例为止，从而产生较优的结果。

(7)整个系统前端依赖于静态分析工具栈中的具体工具，工具不同，解析方式也不同，实现时将此部分独立出来，并将其余部分作为后端，通过将 ISA 代码编译成可执行文件的形式，书写脚本驱动整个系统可以在 Linux 和 Windows 平台上运行。

4 实验结果

本文选取wu-ftpd, Net-tools, Pure-ftpd几个软件包[3]进行测试，它们具有代表性并且都与网络应用相关。目前缓冲区溢出漏洞最常见且危害程度最大，因此，本文测试的漏洞类

型主要为缓冲区溢出。测试的实验环境是：Pentium 1.6 GHz CPU，256 MB内存，Linux(Red Hat9.0)。

表1中列出了不同源码包的特征比较（*Ver*代表软件的版本号，*LOC* 代表除去空行、注释后的源码包代码行数，*LOE* 代表存在实际真正漏洞的真正数目)。不同工具之间的漏报率的比较见表 2。不同工具之间的误报率的比较见表 3，其中，ISA 的阈值设定为 0.21。不同工具之间的效率比较见表 4。

表 1　不同源码包的特征比较

Program	*Ver*	*LOC*	*LOE*	*LOE*/*LOC*
wu-ftpd	2.5.0	13 582	64	0.47%
Net-tools	1.46	4 146	50	1.21%
Pure-ftpd	1.0.17a	25 230	1	3.96E-5

表 2　不同工具的漏报率比较　(%)

Program	ISA	Rats	Its4	Flawfinder
wu-ftpd	10.0	26.0	28.0	12.0
Net-tools	10.9	43.8	39.1	12.5
Pure-ftpd	100.0	100.0	100.0	100.0

表 3　不同工具的误报率比较　(%)

Program	ISA	Rats	Its4	Flawfinder
wu-ftpd	59.1	69.5	68.4	77.3
Net-tools	39.6	42.2	43.8	70.2
Pure-ftpd	100.0	100.0	100.0	100.0

表 4　不同工具的工作效率比较　(%)

program	ISA	Rats	Its4	Flawfinder
wu-ftpd	23.9	0	2.5	17.0
Net-tools	25.9	0	1.7	18.4
Pure-ftpd	0.0	0	0.0	0.0

用 *TP* 表示工具检测出来的真正漏洞数目，*NUM* 表示工具检测出的输出结果条数。对于 ISA 来说，由于大多数的漏洞都被排在了输出结果的前面，因此设定可信赖度阈值对 ISA 的输出结果进行提取。用 *TP**代表指定 ISA 可信赖度阈值后系统检测出来的真正漏洞数目，*NUM**代表 ISA 指定可信赖度阈值后的输出结果条数，*RTP* 代表其他工具指定比例 *ratio* 后，前(*NUM***ratio*)条发现的真正漏洞数，误报率=1-*TP*/*NUM*，漏报率=1-*TP*/*LOE*，而对于 ISA，误报率=1-*TP**/*NUM**；对于工具的执行效率，其他工具没有进行相应的漏洞优先级排名，逐条扫描发现漏洞的概率较低，取 *ratio*=0.3, *efficiency*=*RTP*/(*NUM***ratio*)，对 ISA 来说，分析了输出结果的指定阈值后逐条扫描能够发现的漏洞数目，效率 *efficiency*=*TP**/*NUM**，通过训练和用户交互的调节，ISA 取得阈值为 0.21。

从上述数据中可以发现，与单个工具相比，ISA 的误报率和漏报率明显降低，而 Pure-ftpd 软件包由于源码包中只有 1 个漏洞，原有 3 个工具都没有发现，根据数据融合相应算法，此漏洞仍没有找出，因此，漏报率和误报率都是 100%。对于工作效率的比较，体现 ISA 对漏洞优先级排名的优势，使用户用较少的时间挖掘出较多的漏洞。从而可得出 ISA 系统的 3 个优点：(1)增加检测出来的漏洞数目，相继减少了漏报的数目；(2)在设定阈值的情况下，明显提高了正确率，降低了误报率；(3)通过对漏洞条目进行优先级排序，以较少的时间找出较多的漏洞数，提高了挖掘效率。

5　结束语

本文给出一种基于数据融合的用于漏洞挖掘的源代码静态分析方法，设计并实现了一个源代码静态集成分析系统，对多个静态分析工具的结果进行数据融合，综合了各个工具的优点，实现了较好的检测性能。

参考文献

[1] Chess B, McGraw G. Static Analysis for Security[J]. IEEE Security & Privacy, 2004, 2(4): 32-36.

[2] McGraw G. Software Security[J]. IEEE Security & Privacy, 2004, 2(2): 80-83.

[3] Pozza D, Sisto R. Comparing Lexical Analysis Tools for Buffer Overflow Detection in Network Software[C]//Proc. of Communication System Software and Middleware. New Delhi, India: [s. n.], 2006.

[4] Viega J, Bloch J T. ITS4: A Static Vulnerability Scanner for C and C++ Code[C]//Proc. of Annual Computer Security Applications Conference. [S. l.]: IEEE Computer Society Press, 2000.

[5] Rough Auditing Tool for Security (RATS)[EB/OL]. (2006-04-01). http://www.scans.org./top20.html.

[6] Flawfinder[EB/OL]. (2006-05-01). http://www.dwheeler.com/flawfinder.

(上接第 65 页)

```
MemoryStream mstream=
new MemoryStream((byte[])result);
//定义 MemoryStream 流对象以内存作为支持存储区读取数据
stream=mstream; }
catch (Exception ex){throw new Exception (ex.Message,ex);}
//捕获异常
finally{cn.Close();}}
const int buffersize = 1024 * 16;
byte[] buffer = new byte[buffersize];
//定义数据缓冲区
int count = stream.Read(buffer, 0, buffersize);
while (count > 0)
//从数据缓冲区循环读取数据流
{context.Response.OutputStream.Write(buffer, 0, count);
//把从缓冲区读取的图像流写入当前响应流中
count = stream.Read(buffer, 0, buffersize);
//对 count 重新计数}}
```

由此可知，采用内存流和自定义 HTTP 请求处理程序同步处理 HTTP Web 请求的方法可以在任意时刻对任意页面进行任意 Web 请求处理，同时减少了程序的代码量，提高了 I/O 操作的速度和效率以及数据的访问效率。

4　结束语

本文以图像数据为例，采用内存流存取数据，并启用自定义 HTTP 请求处理程序同步处理 HTTP Web 请求，在.NET 平台上使用 C#编程实现对 Oracle 数据库 BLOB 数据的高效存储和读取。本方法对声音、视频、文档、动画等其他 BLOB 类型数据同样适用，具有实际的参考价值和意义。

参考文献

[1] 杨　云. ASP.NET 典型系统开发[M]. 北京: 人民邮电出版社, 2006.

[2] 高晓兵. 基于数据仓库的质量信息系统关键技术研究[D]. 西安: 西北工业大学, 2005.

[3] Kauffman J, Matsik B. Beginning ASP.NET Databases Using C#[M]. [S. l.]: Wrox Press Ltd., 2002.