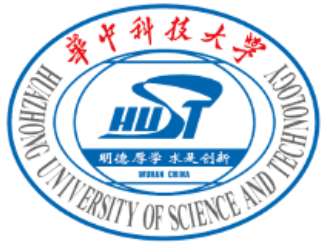


华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

网络空间安全学院

漏洞发现

网络空间安全学院 慕冬亮
Email : dzm91@hust.edu.cn

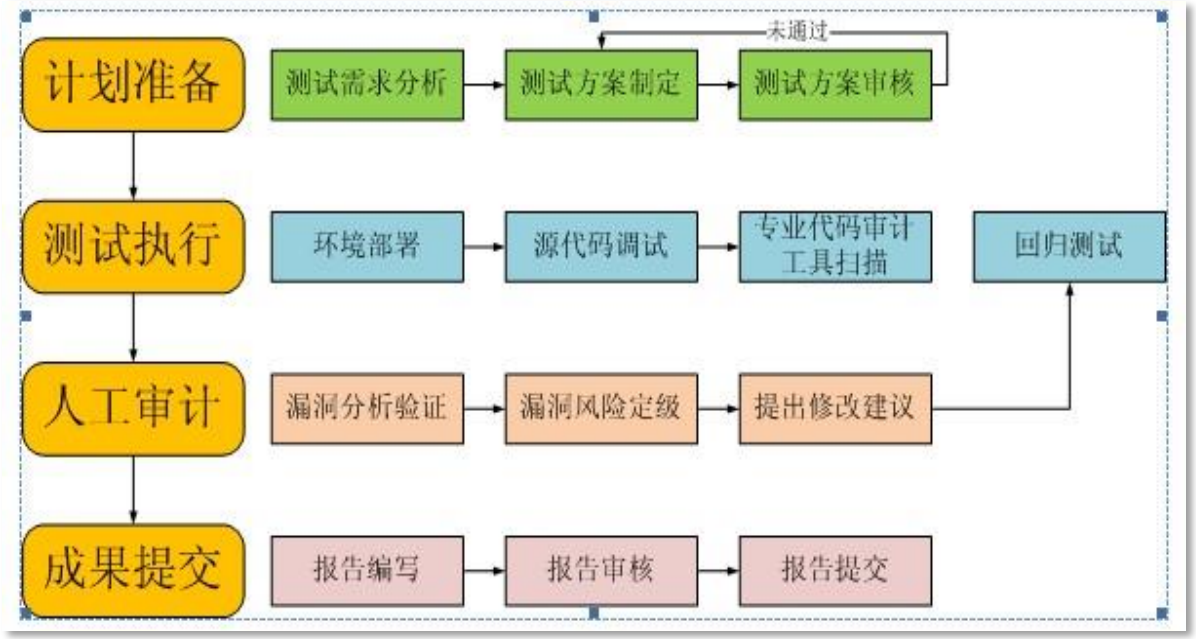
软件漏洞挖掘技术

- 源代码审计
- 静态分析工具
- 动态分析技术
- Fuzzing测试
- 面向二进制程序的逆向分析
- 基于补丁比对的逆向分析

源代码安全审计

源代码安全审计

- 词法分析和语法分析
- 构建控制流图、数据流图
- 审计危险函数
  - 字符串操作函数
  - 内存拷贝函数
  - 文件操作
  - 数据库操作
  - 进程操作
  - 安全检查函数
  - ……



源代码安全审计

- **程序切片**：程序切片（slice）是影响指定值的程序语句和判定表达式组成的语句集合，包括前向切片和后向切片。采用控制依赖和数据依赖实现。
  - 后向切片
    - 指构造一个集合affect(v/n)，使得这个集合由所有影响n点变量V的值的语句和谓词组成。
  - 前向切片
    - 指构造一个集合affect (v/n)，使得这个集合由受到n点的变量V的值的影响的所有语句和谓词构成。

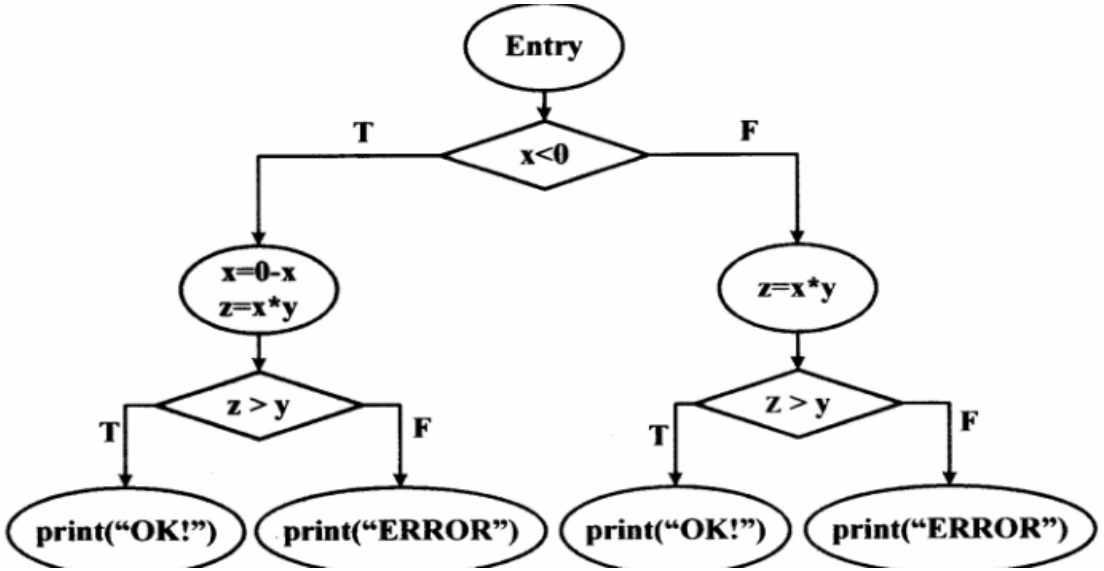
```
int i;
int sum =0;
int product =1;
for (i =0; i < N;++i) {
   sum = sum + i;
   product = product *i;
}
write(sum);
write(product);
```

- 静态切片
  - 不考虑程序的具体输入，计算对某个感兴趣点的变量有影响的语句和谓词集合
- 动态切片
  - 在某个给定输入的条件下，程序执行路径上所有对程序某个兴趣点上的变量有影响的语句和谓词集合。

源代码安全审计

- **符号执行**（Symbolic Execution）：通过分析程序来得到让特定代码区域执行的输入。
- 使用符号值作为输入（而非一般执行程序时使用的具体值），在达到目标代码时，分析器可以得到相应的路径约束，然后通过约束求解器来得到可以触发目标代码的具体值。

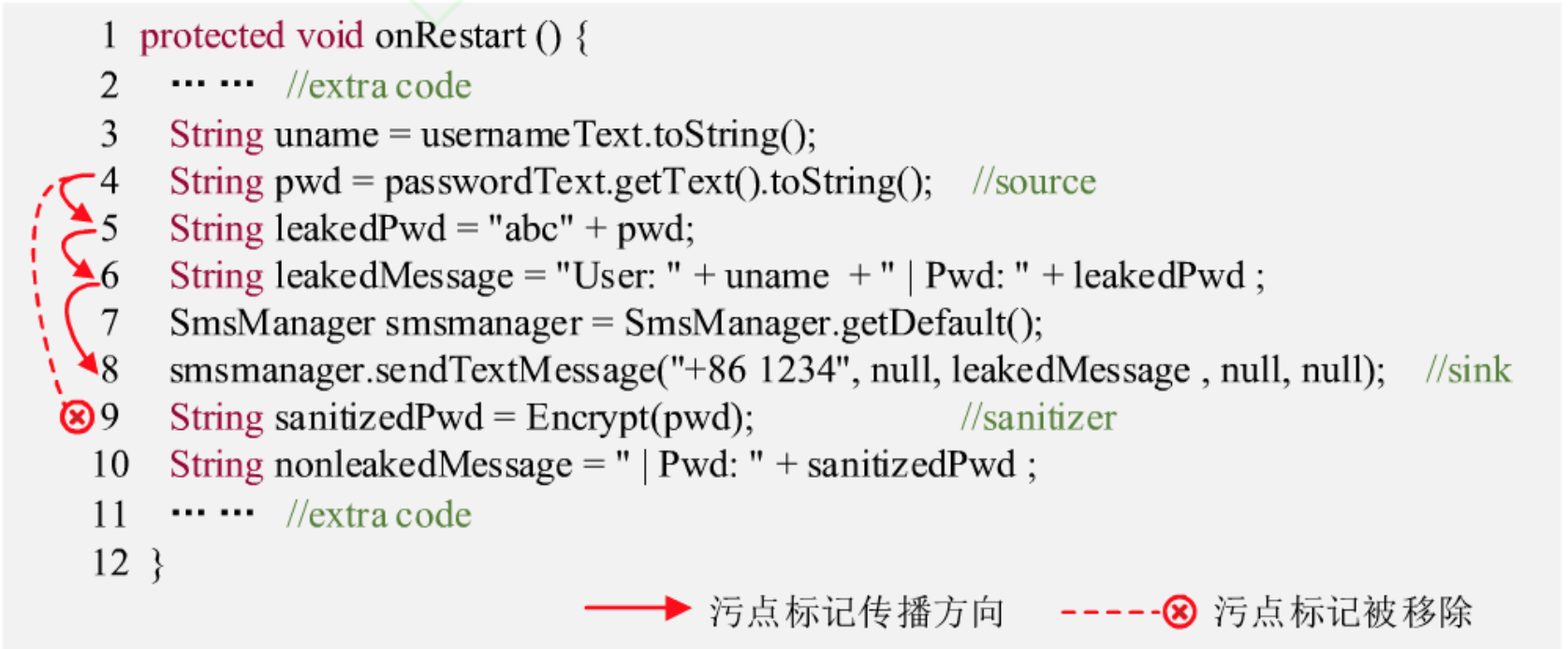
```
1   int Foo(int x, int y){
2     if(x < 0){
3       x=0-x;
4     }
5     int z = x*y;
6     if(z > y){
7       print("OK!");
8     }else{
9       print("ERROR");
10    }
11    return 0;
12  }
```



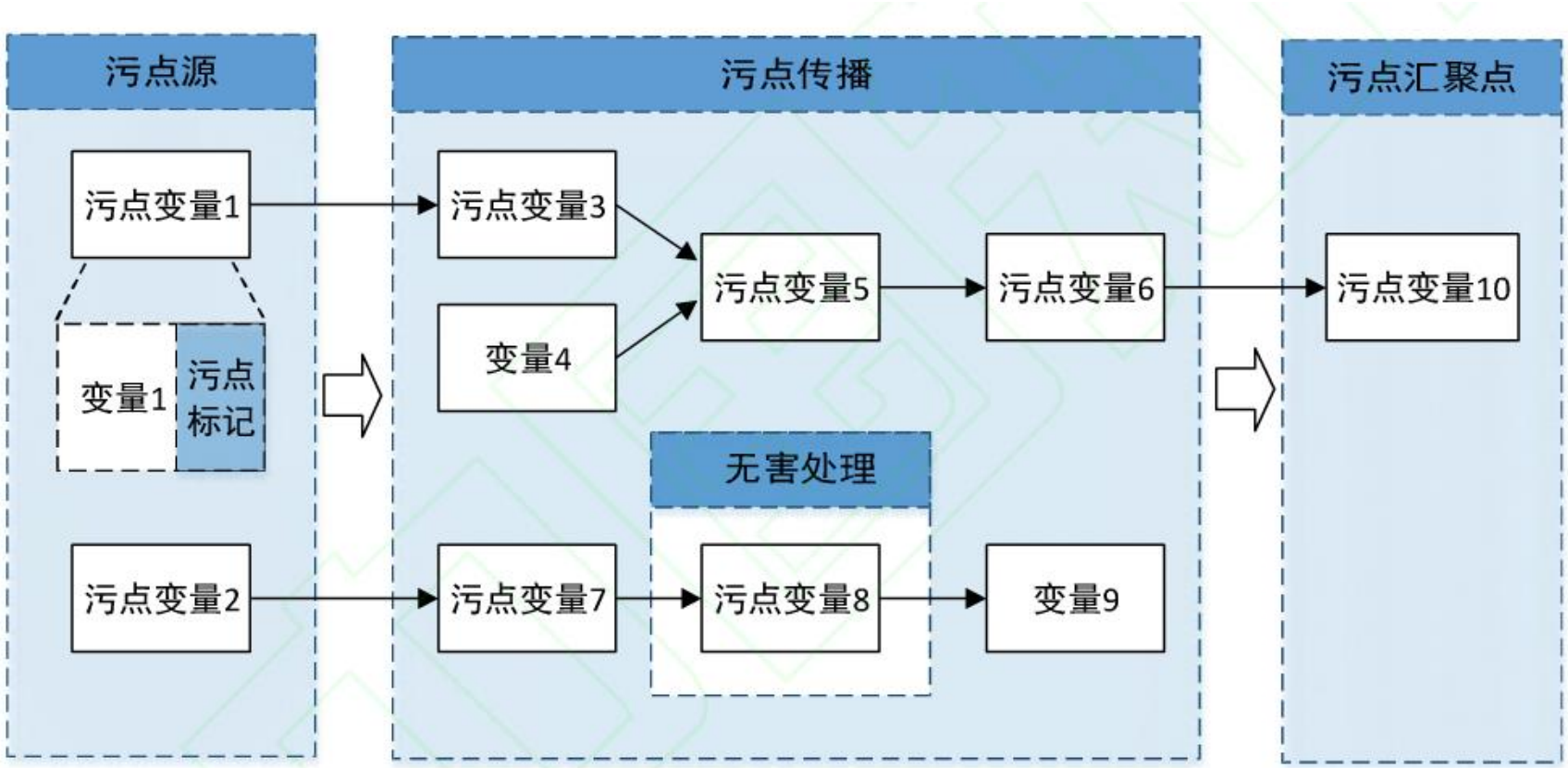
符号执行的缺陷是什么？

源代码安全审计

- 污点分析：把输入标记为Source（ Taint），把敏感函数或者敏感操作标记为Sink。



污点分析的处理过程

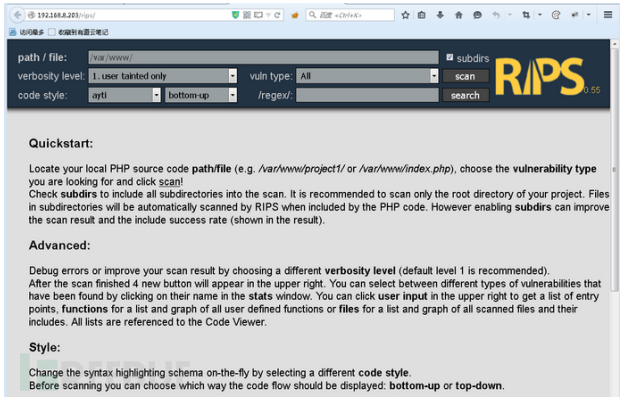

污点源
污点传播
污点汇聚点
污点变量1
变量1
污点标记
污点变量2
污点变量3
变量4
污点变量5
污点变量6
无害处理
污点变量7
污点变量8
变量9
污点变量10

静态分析工具

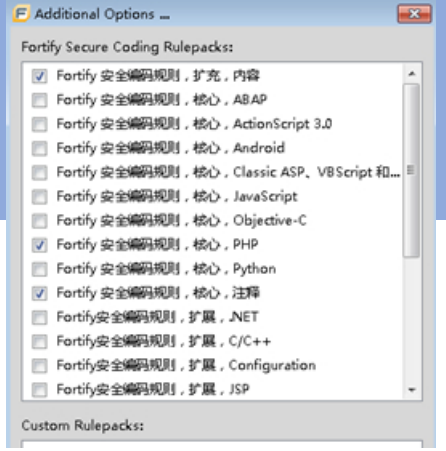
- 依据CVE公共漏洞字典表、OWASP十大Web漏洞，以及设备、软件厂商公布的漏洞库，结合软件设计规范或者编程规范对各种程序语言编写的**源代码**进行安全审计的工具。
  - 典型工具：RIPS、Fortify SCA、VCG等。

静态分析工具

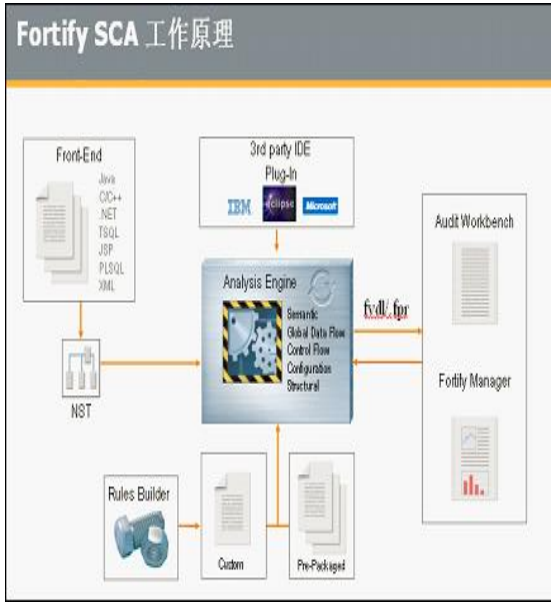
- RIPS
  - 能够检测XSS、SQL注入、文件泄露、本地/远程文件包含、远程命令执行以及更多种类型的漏洞。

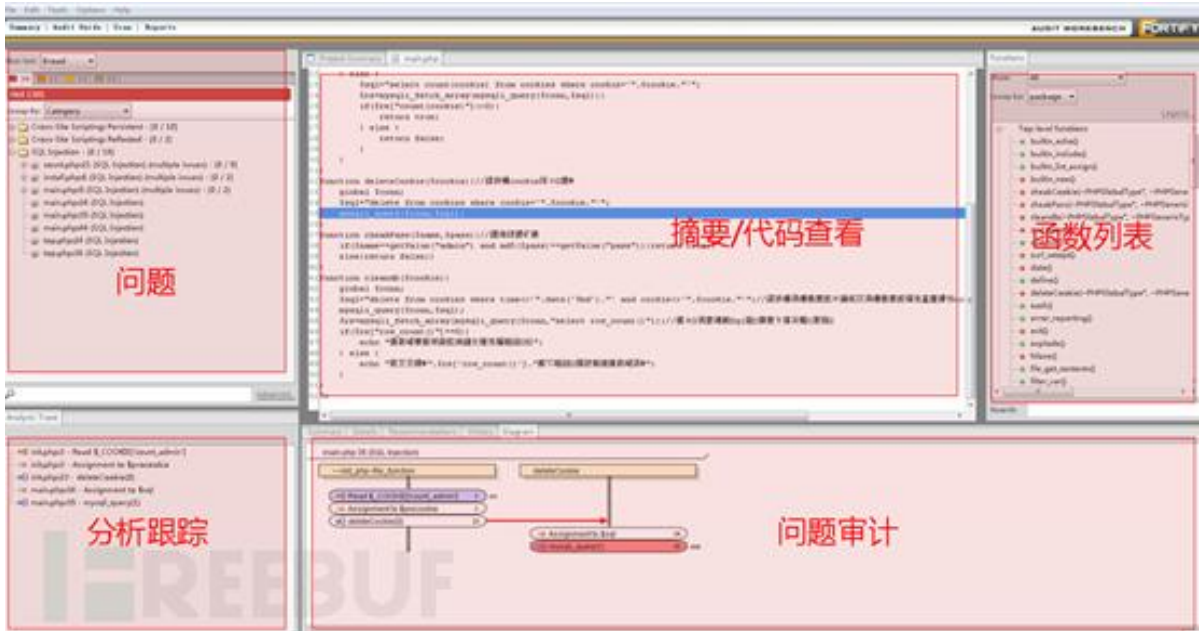


静态分析工具



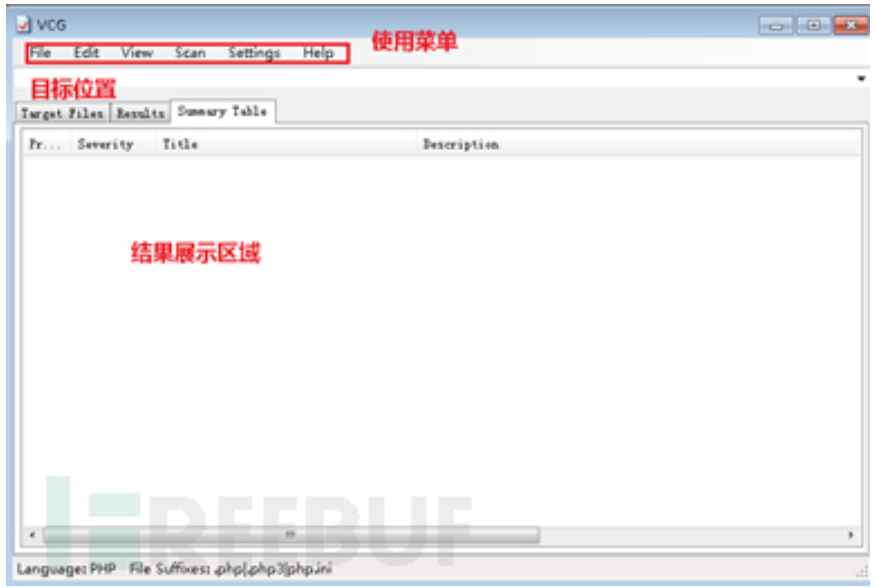
- Fortify SCA（商业软件）
  - 数据流引擎：跟踪,记录并分析程序中的数据传递过程所产生的安全问题
  - 语义引擎：分析程序中不安全的函数,方法的使用的安全问题
  - 结构引擎：分析程序上下文环境,结构中的安全问题
  - 控制流引擎：分析程序特定时间,状态下执行操作指令的安全问题
  - 配置引擎：分析项目配置文件中的敏感信息和配置缺失的安全问题
  - 特有的X-Tier™跟踪器：跨跃项目的上下层次,贯穿程序来综合分析问题

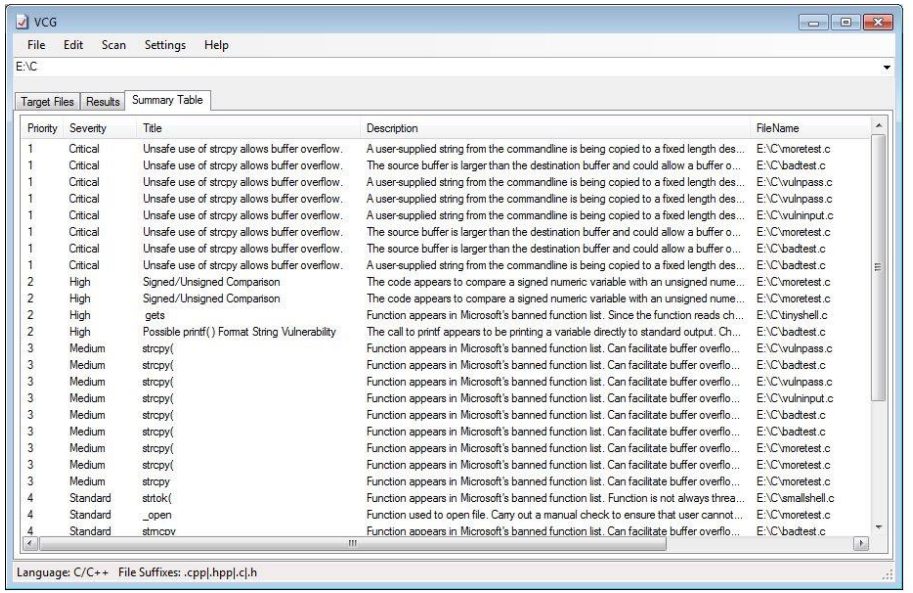




静态分析工具

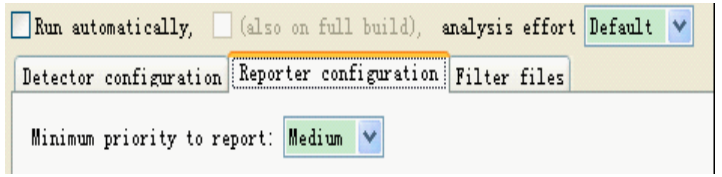
- VCG（VisualCodeGrepper）是一个基于字典的自动化源代码扫描工具，可以由用户自定义需要扫描的数据。它可以对源代码中所有可能存在风险的函数和文本做一个快速的定位。［简洁型，基于字典］





静态分析工具

- Findbugs-Java Bug pattern
  - 正确性（Correctness）：这种归类下的问题在某种情况下会导致bug，比如错误的强制类型



FindBugs
because it's easy

Docs and Info
FindBugs 2.0
Demo and data
Users and supporters
FindBugs blog
Fact sheet
Manual
Manual(ja/日本語)
FAQ
Bug descriptions
Bug descriptions(ja/日本語)
Bug descriptions(fr)
Mailing lists
Documents and Publications
Links

Downloads

FindBugs Swag

Development
Open bugs
Reporting bugs
Contributing
Dev team
API [no frames]

FindBugs Bug Descriptions

This document lists the standard bug patterns reported by FindBugs version 3.0.1.

Summary

Description	Category
BC: Equals method should not assume anything about the type of its argument	Bad practice
BIT: Check for sign of bitwise operation	Bad practice
CN: Class implements Cloneable but does not define or use clone method	Bad practice
CN: clone method does not call super.clone()	Bad practice
CN: Class defines clone() but doesn't implement Cloneable	Bad practice
CNT: Rough value of known constant found	Bad practice
Co: Abstract class defines covariant compareTo() method	Bad practice
Co: compareTo()/compare() incorrectly handles float or double value	Bad practice
Co: compareTo()/compare() returns Integer.MIN_VALUE	Bad practice
Co: Covariant compareTo() method defined	Bad practice
DE: Method might drop exception	Bad practice
DE: Method might ignore exception	Bad practice
DMI: Adding elements of an entry set may fail due to reuse of Entry objects	Bad practice
DMI: Random object created and used only once	Bad practice
DMI: Don't use removeAll to clear a collection	Bad practice
Dm: Method invokes System.exit(...)	Bad practice
Dm: Method invokes dangerous method runFinalizersOnExit	Bad practice
ES: Comparison of String parameter using == or !=	Bad practice
ES: Comparison of String objects using == or !=	Bad practice
Eq: Abstract class defines covariant equals() method	Bad practice
Eq: Equals checks for incompatible operand	Bad practice
Eq: Class defines compareTo(...) and uses Object.equals()	Bad practice
Eq: equals method fails for subtypes	Bad practice

FindBugs 1.2 demo and results

- http://findbugs.sourceforge.net

Application	Details		Correctness bugs		Bad Practice	Dodgy	KNCSS
	HTML	WebStart	NP bugs	Other			
Sun JDK 1.7.0-b12	All	All Small	68	180	954	654	597
eclipse-SDK-3.3M7-solaris-gtk	All	All Small	146	259	1,079	643	1,447
netbeans-6_0-m8	All	All Small	189	305	3,010	1,112	1,022
glassfish-v2-b43	All	All Small	146	154	964	1,222	2,176
jboss-4.0.5	All	All Small	30	57	263	214	178

KNCSS - Thousands of lines of non-commenting source statements

Bug categories

Correctness bug
: Probable bug - an apparent coding mistake resulting in code that was probably not what the developer intended. We strive for a low false positive rate.

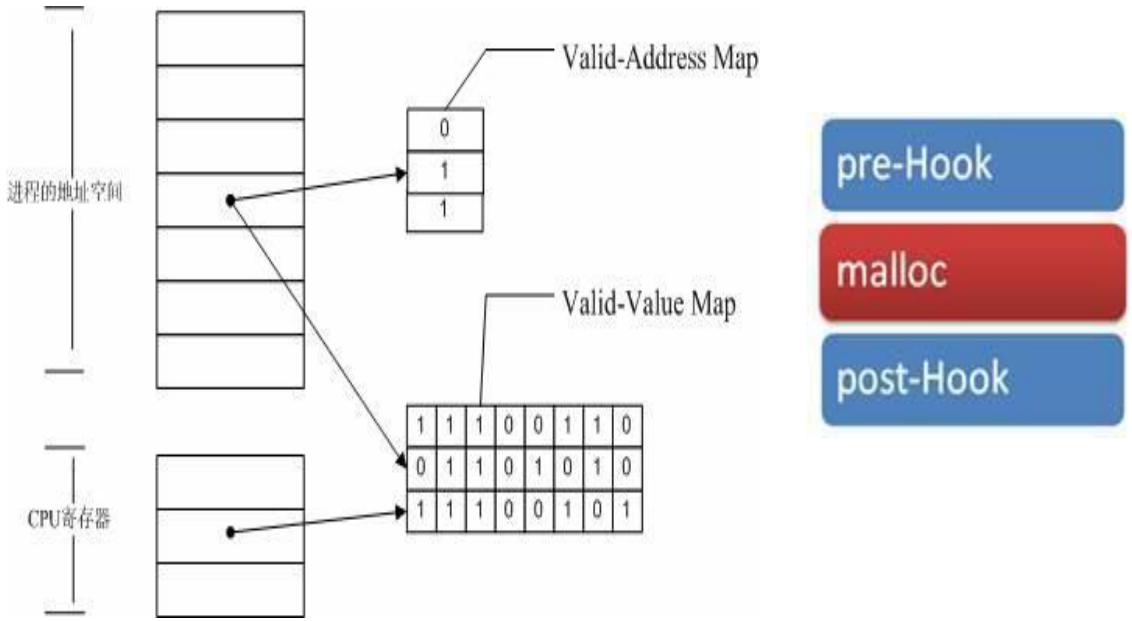
Bad Practice
: Violations of recommended and essential coding practice. Examples include hash code and equals problems, cloneable idiom, dropped exceptions, serializable problems, and misuse of finalize. We strive to make this analysis accurate, although some groups may not care about some of the bad practices.

Dodgy
: Code that is confusing, anomalous, or written in a way that leads itself to errors. Examples include dead local stores, switch fall through, unconfirmed casts, and redundant null check of value known to be null. More false positives accepted. In previous versions of FindBugs, this category was known as Style.

动态分析

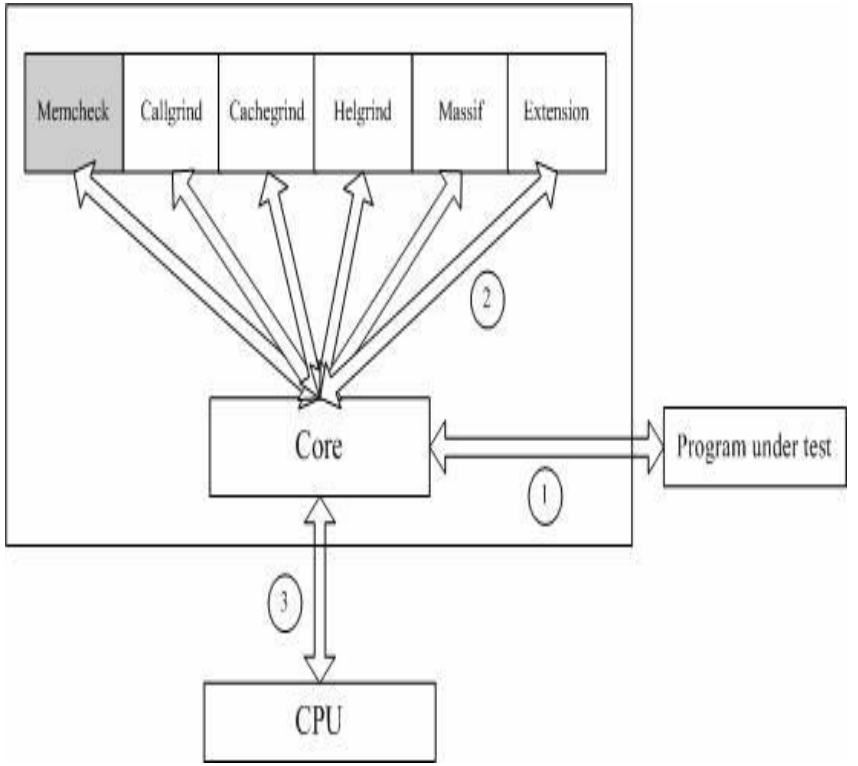
- 动态分析：收集程序多次执行的运行过程的状态信息，结合输入和输出，检测程序存在的缺陷或漏洞。
- **缘起**：静态分析的结果不准确，存在误报较多。

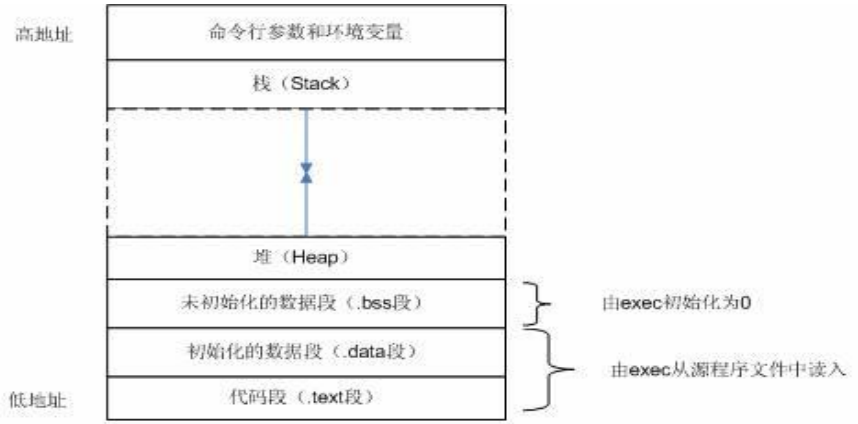
- 分析粒度：
  - 指令
  - 分支
  - 函数
  - 系统调用
- 分析方法：
  - 动态切片
  - 污点传播



动态分析

- MEMcheck：Valgrind工具集下的内存检查工具
  - 使用未初始化内存
  - 对释放后内存的读/写
  - 对已分配内存块尾部的读/写
  - 内存泄露
  - 不匹配的使用malloc/new/new[] 和 free/delete/delete[]
  - 重复释放内存





动态分析

- BitBlaze： 由三个部分组成：静态分析组件（Vine），动态分析组件（TEMU），结合动态和静态分析进行具体和符号化分析的组件（Rudder）。
  - Vine：将汇编语言翻译成一种中间语言（IL），并提供一系列在IL上进行静态分析的核心工具（包括控制流，数据流，优化，符号化执行等）
  - TEMU：进行动态分析，以支持系统全局的细粒度监控和动态二进制插桩。它提供了一系列工具用于提取操作系统级语义，用户自定义的动态污点分析，以及一个用于用户自定义活动的插件接口。
  - Rudder：利用Vine和TEMU提供的核心功能在二进制代码级进行具体及符号化混合的执行。对于一条指定的程序执行路径，它能给出满足要求的符号化输入参数。

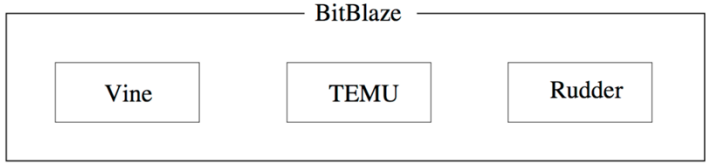


Fig. 1. The BitBlaze Binary Analysis Platform Overview

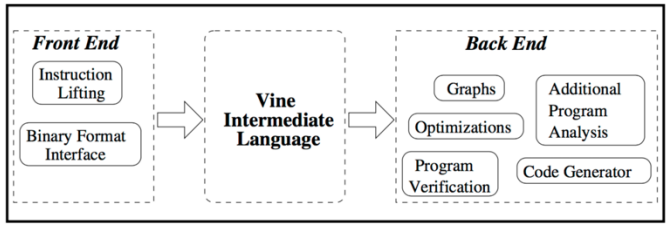


Fig. 2. Vine Overview

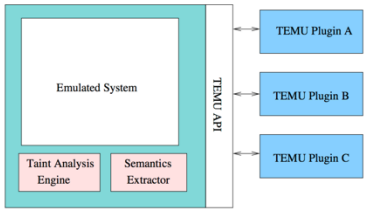


Fig. 5. TEMU Overview

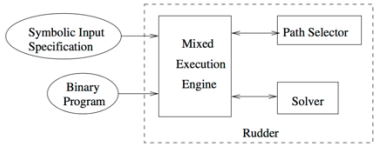
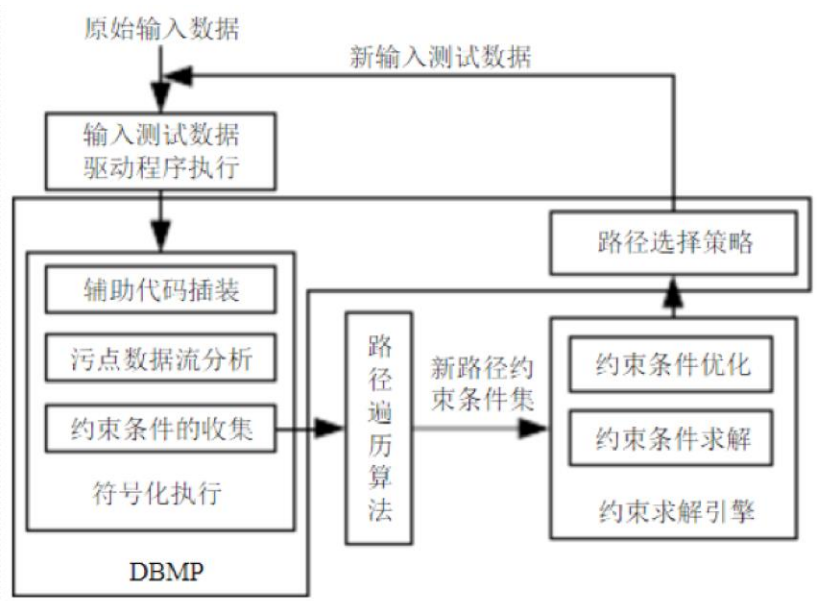


Fig. 6. Rudder Overview

http://bitblaze.cs.berkeley.edu/papers/bitblaze_iciss08.pdf
BitBlaze: A New Approach to Computer Security via Binary Analysis

Fuzzing 模糊测试

- Fuzzing是一种基于错误注入的随机测试技术，向目标程序输入随机或者半随机（semi-valid）的测试集，分析程序的执行结果及检测程序状态，发现目标程序中隐藏的各种安全漏洞
  - Barton Miller于1989年在威斯康星大学提出
  - Fuzzer能很好的检测出可能导致程序崩溃的问题，如缓冲区溢出、跨站点脚本，拒绝服务攻击、格式漏洞和SQL injection等



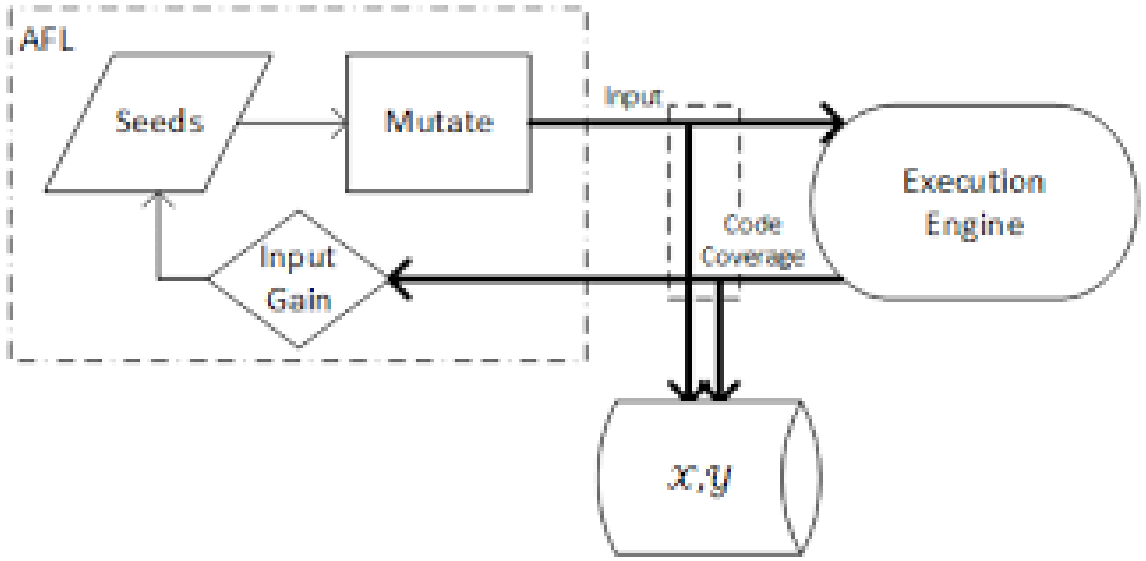
Fuzzing模糊测试

- Fuzzing 命令参数
  - OS命令
  - 系统调用
  - 内核调用
  - COMRaider
  - SyscallFuzz
  - Socket Fuzzer
- Fuzzing文件格式
  - FileFuzzer
  - zzuf
- Fuzzing 网络协议
  - Spike、ProtoFuzz
  - Ircfuzz、Dhcpfuzz
  - Infigo、FTPstress

- 其他：
  - IOCTL Fuzzer： Windows内核Fuzzer；
  - Android kernel Fuzzer v 0.2；
  - Browser Fuzzer: 浏览器Fuzzer；
  - JSFuzzer/Spanner：Javascript Fuzzer；

经典模糊测试工具：AFL Fuzzer

Code-coverage-based fuzzer: 以代码覆盖率作为反馈来引导用户输入改变的模糊测试工具

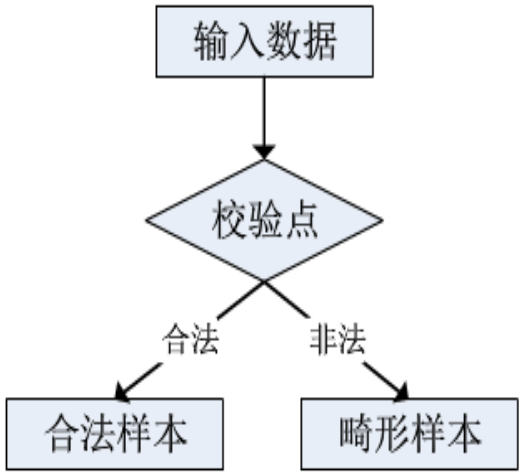


[Fuzzing: a survey]

Fuzzing模糊测试

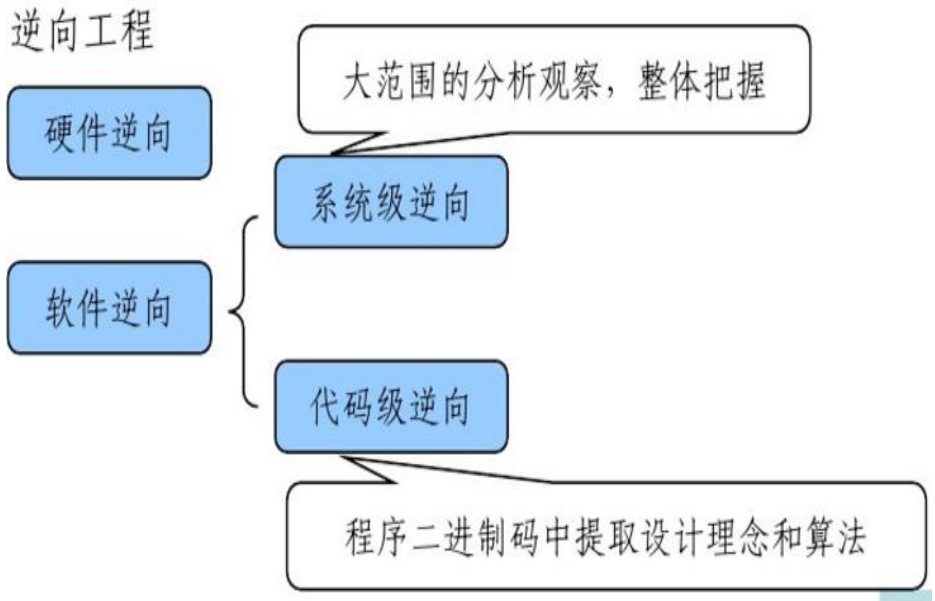
- 难点
  - 输入值在特定小范围引发的异常
  - 几个输入同时作用下引发的异常
  - 需满足特定数据/文件格式后才引发的异常

- SmartFuzzer
  - 输入的格式分割
  - 输入字段的约束
  - 字段之间的约束



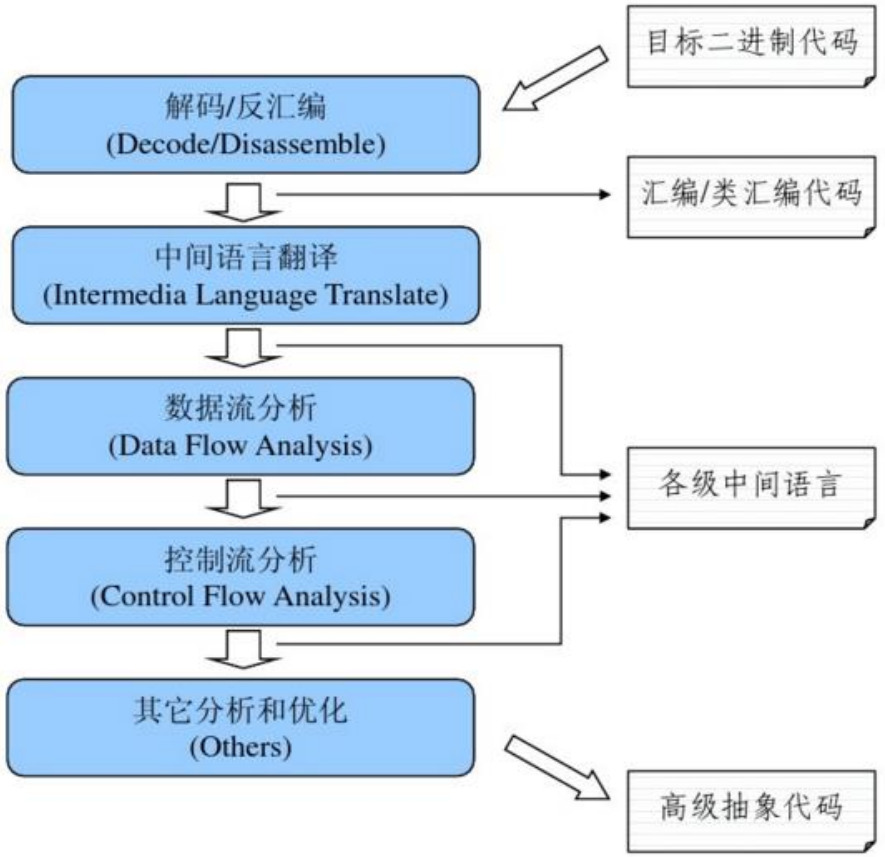
面向二进制程序的逆向分析

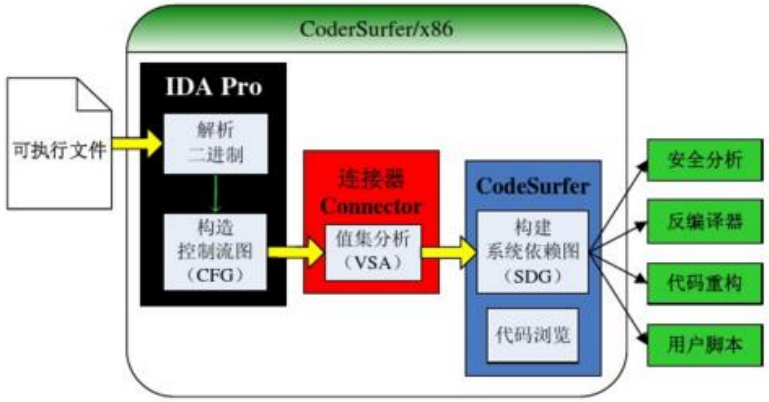
- **程序理解**
  - （密码）算法的理解学习
  - 代码检查
  - 代码比较
  - 查找恶意代码
  - 查找软件Bug
  - 查找软件漏洞
- **代码优化**
  - 平台间的移植
  - 修补Bug
  - 增加新的特性
  - 代码恢复优化



面向二进制程序的逆向分析

- 流程逆向
  - 关键函数（加密、认证）
  - 函数的关联
- 数据格式逆向
  - 格式的分割
  - 类型的推理
  - 字段值溯源
- IDA(Hex-rays)，OD，GDB，WinDBG





基于补丁分析的逆向分析

- 补丁
  - Bug
  - 漏洞
  - 功能优化
- 补丁分析
  - 漏洞机理
  - 功能优化
- 分析方法
  - 基于指令相似性的图形化比较
  - 结构化二进制比较
- **BinDiff（谷歌开源二进制文件对比工具）**
  - 能够针对x86、MIPS、ARM/AArch64、PowerPC等架构进行二进制文件的对比。

思考题

- 1.如何动态检测Shellcode？
- 2.利用栈空间实现一个简单的JOP和COP。
- 3.如何对抗ROP攻击？
- 4.Fuzzing的难点在哪里？如何提高Fuzzing的效率？
- 5.试结合整数溢出分析污点数据跟踪为什么容易漏报？
- 6.讨论补丁设计的安全规范。

参考文献

- 李承远。逆向工程核心原理. 人民邮电出版社,2014
- 钱松林。C++反汇编与逆向分析技术揭秘。机械工业出版社，2011.
- 工具推荐：三款自动化代码审计工具
- 程序动态切片技术研究
  - https://www.cnblogs.com/maifengqiang/archive/2013/05/21/3090739.html
- 谷歌开源二进制文件对比工具 BinDiff
  - https://www.aliyun.com/jiaocheng/164308.html?spm=5176.100033.2.5.jhW7pl
- BitBlaze: A New Approach to Computer Security via Binary Analysis
  - http://bitblaze.cs.berkeley.edu/papers/bitblaze_iciss08.pdf

本章艰巨地的暂告一段落...

- 下一章继续...