

# QCon 全球软件开发大会 【北京站】2016

使用静态分析技术找到“真正”的代码质量  
缺陷与安全漏洞

韩葆 (Bob Han), Synopsys Software Integrity  
Group, 13311307163, Bao.Han@synopsys.com

# QCon

2016.10.20~22  
上海·宝华万豪酒店

## 全球软件开发大会 2016

### [上海站]



购票热线: 010-64738142  
会务咨询: [qcon@cn.infoq.com](mailto:qcon@cn.infoq.com)  
赞助咨询: [sponsor@cn.infoq.com](mailto:sponsor@cn.infoq.com)  
议题提交: [speakers@cn.infoq.com](mailto:speakers@cn.infoq.com)  
在线咨询 (QQ): 1173834688

团·购·享·受·更·多·优·惠

**7折** 优惠 (截至06月21日)  
现在报名, 立省2040元/张

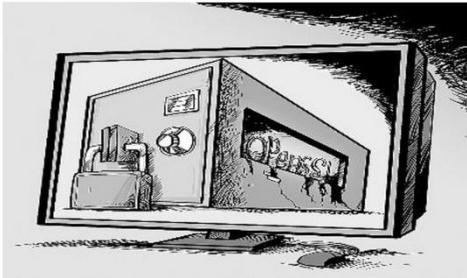
# HeartBleed Bug



新浪科技 > 互联网 > 正文

## 互联网“心脏出血” 网民信息面临泄密风险

2014年04月11日 17:35 法制晚报 我有话说(7人参与) 收藏本文



### 2亿网民面临泄密风险!OpenSSL漏洞危害大

近日，安全协议OpenSSL爆出本年度最严重的安全漏洞“心脏出血”。利用该漏洞，黑客坐在自己家里电脑前，就可以实时获取到很多https开头网址的用户登录账号密码，包括网银、知名购物网站、电子邮件等信息，这相当于信用卡信息被公开的程度，分钟钟损失几十万的节奏呀。据360网站安全检测平台对国内120万家经过授权的网站扫描，其中有11440个网站主机受该漏洞影响。

目前国内大量网站包括阿里巴巴、腾讯等多个大型互联网服务商通过官微宣布，已经修复了该OpenSSL漏洞。而中国金融认证中心则发文表示，网银受到的影响较少，U盾可以放心使用。这个漏洞为什么称为“心脏出血”漏洞呢？它的危害性有多大呢？到底什么是OpenSSL呢？下面为你全面解读。



什么是OpenSSL?什么是“心脏出血”漏洞？  
面对此漏洞，用户应该怎么做？

- 1 手机ROM是什么意思
- 2 recovery刷机教程，recovery模式
- 3 iPhone怎样鉴别翻新机
- 4 GSM是什么
- 5 手机系统哪个好？5大智能手机系统
- 6 Win7系统硬盘分区怎么调整大小
- 7 无线路由器怎么设置
- 8 ISO是什么？相机ISO是什么意思？
- 9 笔记本如何设置WIFI
- 10 iPhone4S储存数据怎么设置

#### 热门文章专区排行

- 1 手机技术专区
- 2 iPhone4S专区
- 3 笔记本专区
- 4 Win7专区
- 5 华为荣耀3C专区
- 6 iPad4专区
- 7 硬盘专区
- 8 移动存储专区
- 9 iPad Air专区



## “网络核弹”当前 全球互联网大佬紧急排险

来源：今日早报 2014年04月10日 10:26:17

“网络核弹”当前，全球互联网大佬紧急排险

专家建议，网站未打补丁前别急着修改密码

本周二，黑客和“白帽”（使用正当手段优化网站的IT人士）们经历了一个不眠之夜。

他们有的在狂欢，逐个进入戒备森严的网站，耐心地收集泄漏数据，拼凑出用户的明文密码；有的在艰苦升级系统，统计漏洞信息，还要准备说服客户的说辞，让他们意识到问题的严重性……

当天，从亚马逊到雅虎，多个国际大牌互联网公司都召开紧急会议，应对最新发现的互联网漏洞“心脏出血（Heart bleed）”。这是发现者们给这个号称本年度最严重安全漏洞起的形象

www.cnet.com/how-to/which-sites-have-patched-the-heartbleed-bug/

Dropcam Pro. Capture high-def video. Store it in the cloud. Watch it anytime. BUY NOW \$199

Heartbleed bug: Check which sites have been patched

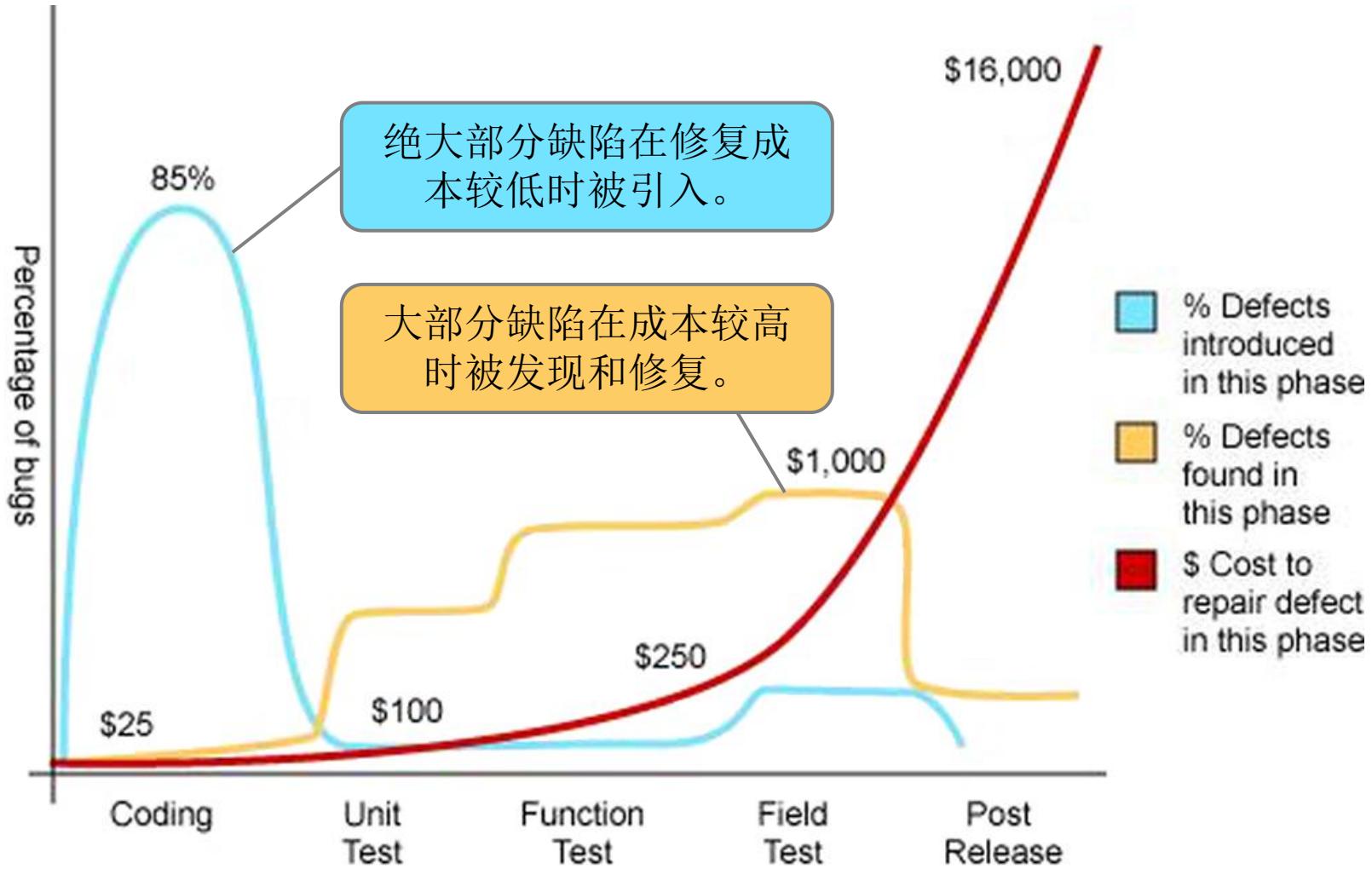
We compiled a list of the top 100 sites across the Web, and checked to see if the Heartbleed bug was patched.

by Jason Cipriani / 5h Copy / April 9, 2014 2:54 PM PDT

201 / 8.5K / 2.0K / 860 / 3 / more +

DON'T MISS / Heartbleed bug: What you need to know (FAQ)

# 软件研发测试经济学



Source: *Applied Software Measurement*, Capers Jones, 1996

# 代码静态分析技术



# 代码静态分析

- 定义：在不执行计算机程序的条件下，对源代码进行分析，找出代码缺陷
- 执行方式：一般配合静态程序分析工具进行
- 采用技术：数据流分析、机器学习、语义精简...
- 可检测类型：死锁，空指针，资源泄露，缓冲区溢出，安全漏洞，竞态条件...
- 优点：
  - 能够检测所有的代码级别的可执行路径组合，快速，准确
  - 直接面向源码，分析多种问题
  - 在研发阶段开始找到并修复多种问题，节省大量时间/人力成本
- 注意：静态分析不是万能的，测试是持续的过程，非一劳永逸

# 现存问题

- 编译器警告: 保证类型安全
  - 最初级的静态分析, 检测规则简单
- 中间语言分析: 检测字节码 (Byte Code) 的缺陷, 将其重新映射到真实代码中
  - 在转换与映射过程中易造成精度丢失
- 高误报率: 目前静态分析产品的误报率普遍在30%以上。
- 缺陷种类较少, 找到的问题级别不高: 多数为代码规范或低级缺陷, 非实际Bug
  - 如命名规范、类定义规范, 最佳实践.....
- 易用性较低: 基本上都是一次性的使用工具, 无法与SDLC集成
  - SCM集成: 如SVN, CVS, Perforce, Git
  - Bug Tracking: 如Bugzilla, Jira

# 改进型的静态分析方案

- 基于Meta Compilation的静态分析：
  - 由斯坦福大学教授Dawson Engler提出，在深度理解代码与程序语义的基础上检测缺陷
  - 旨在查找“真正的代码缺陷”
- 实现原理：
  - 使用可扩展的metal语言定义正确性Checker
  - 将程序的源码使用状态机进行抽象描述（State Machine Abstraction）。
  - 使用xgcc系统匹配Checker与抽象状态机状态，找到问题所在的点。
- 可准确检测实际的Bug（内存和指针问题、资源泄露、缓冲区溢出，数组越界，心脏出血漏洞...）
  - 能够检测高达亿行级别的代码库，避免“状态爆炸”
  - 使用模型检验与符号执行技术，误报率降低至15%以下
- 算法已步入实际应用
  - 面向企业的Coverity 软件
  - 面向开源代码的Coverity SCAN

# 源码分析-数据流分析

```
if (x == 0)
```

```
    do_something(x);
```

```
x = 1;
```

- 源码分析可以探知开发者的想法：“`x=1`”需要在调用“`do_something`”后继续执行。
- 提出警告：if循环没有包含所有语句

# 如何进行Java代码静态分析？

Java语言被编译成JVM bytecode - 在运行时被转换成本地可执行代码的分析

## 选项一

- 分析 **byte-code**: 用户编译他们的软件，然后分析编译后的可执行文件与调试信息，分析引擎联系找到的缺陷与源代码位置
- 某些开源工具的实现原理

## 选项二：

- 获取所有的Java编译过程并执行分析
- Bytecode分析工作仍旧存在，但包含更多的内容

# 基本的工作流

- 获取所有编译过程
- 每当 “javac (或其他相关API)” 被调用后，编译获取系统记录所有的编译器选项，操作，源代码与调用的库文件
- 面向源代码和库文件可进行全面编译后分析
- 找到的缺陷将被展示给研发人员修复

# 如何分析缺陷？

- **过程间分析** (**Intra-procedural analyses**) 将考虑每一个合理的可执行路径
  - 快速修剪不可行路径是一件很麻烦的事情！
  - 数学方案
- 获取一系列的函数定义
  - 资源分配
  - 调用....
- **过程间分析**
  - **Bytecode** 分析将创建函数定义

# 如何分析缺陷？

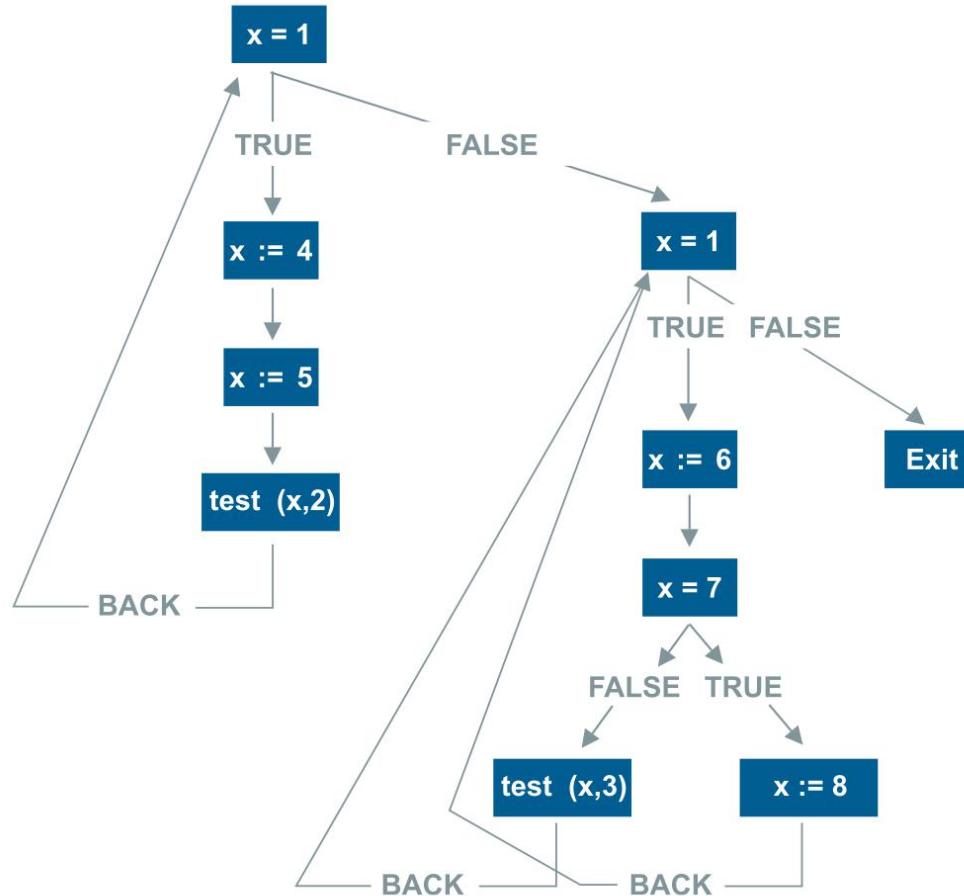
- 数据流分析将跟踪所有应用中的不可信数据
  - “source”
  - “sink”
  - 二者之间必须进行验证
- 某些使用智能静态分析，例如：
  - checked this return value for null 19 times out of 20
  - accessed this field under a lock 19 times out of 20
  - called base.Foo() in 19 overrides of Foo() out of 20

# 找到潜在Bug其实只是难题之一

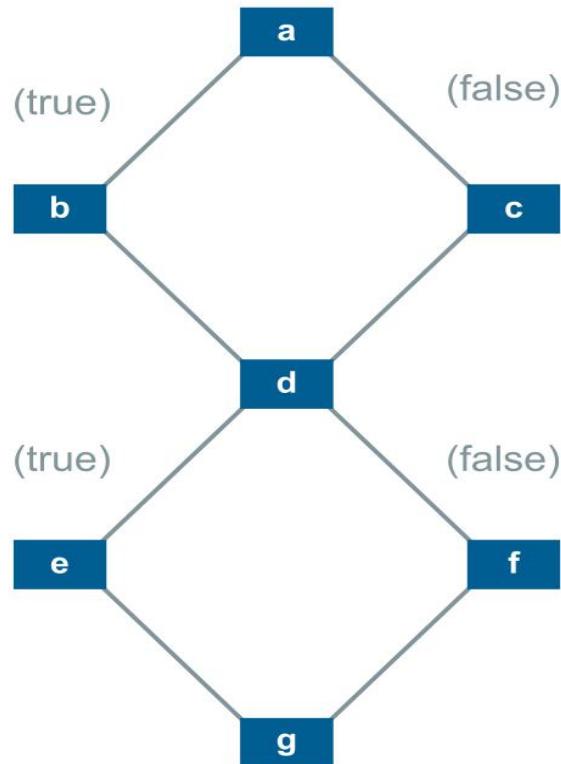
- 消除误报非常难
- 将复杂的缺陷解释出来很难
- 只找潜在的一次性缺陷是很难的

# 难题！

## Control Flow Graph

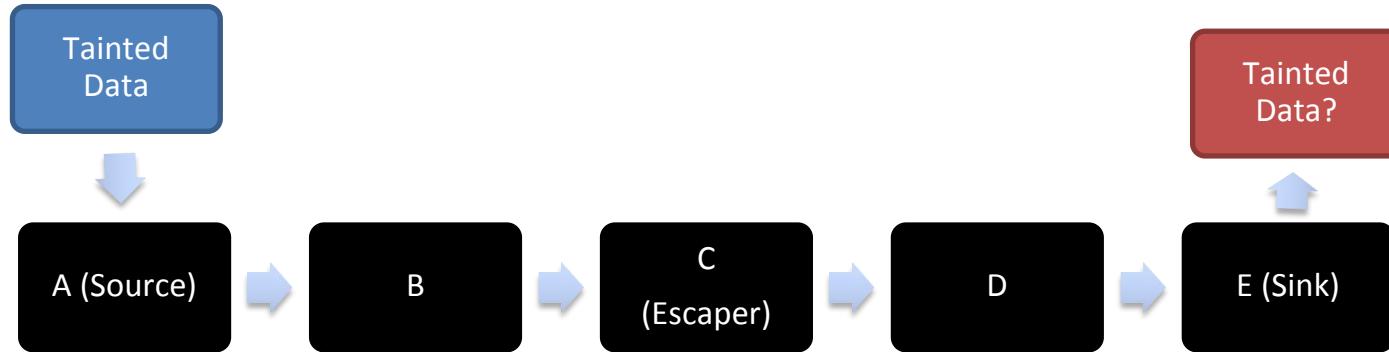


# 如何简化搜索？



# 数据流分析

Advanced data flow



- 动态“fuzzing”
- 基于上下文分析

# Coverity OWASP top 10: JSP&ASP

OWASP 10 - 2013	CWE映射
A1: 注入	77, 78, 88, 89, 90, 564, 917
A2: 失效认证与会话管理	259, 321, 384, 798
A3: 跨站脚本攻击 (XSS)	79, 80, 81, 82, 83, 84, 86, 87
A4: 不安全的直接对象引用	22, 23, 36
A5: 安全配置错误	4, 7, 86, 650
A6: 敏感信息泄露	321
A7: 功能级访问控制缺失	425, 862, 863
A8: 跨站请求伪造	352
A9: 使用含有已知漏洞的组件	NA
A10: 未验证的重定向和转发	938

# Java 缺陷

## Web 应用安全缺陷 (OWASP Top 10)

- 跨站脚本攻击
- SQL 注入
- 命令行注入
- 路径遍历...

## 资源泄露

- 数据库连接资源泄露
- 资源泄露
- **Socket & Stream** 泄露

## 并发数据访问异常

- 变量非原子更新
- 双重检查锁定
- 数据竞态条件
- **Volatile**非原子更新
- **Servlet** 属性无效锁定
- 单例模式竞态条件

## 程序假死

- 线程死锁
- 死锁

## 空指针引用

- **Null**检查后引用空指针
- 直接引用返回的空指针
- **Null**检查前引用空指针

## API 使用错误

- 无效迭代器使用
- 不可修改的集合错误
- 已释放资源调用

## 性能缺陷

- 低效率方法使用
- 在循环中连接字符串
- 冗余同步

## 逻辑错误

- 不可达代码
- 未使用变量
- 常量表达式
- 非本地资源不当使用
- 整数溢出
- 不当分号

# Java 缺陷

## 类层次结构不一致

- 调用 `super.clone()` 或 `super.finalize()` 失败
- 父函数调用丢失
- 构造函数中使用虚函数

## 控制流缺陷

- 在 `Finally` 模块中返回
- `Switch` 语句中 `break` 丢失

## 错误处理缺陷

- 未验证的返回值

## 数据库操作

- 不正确的实体哈希
- `Load` 函数返回值错误验证
- 不完全持续周期
- `get()` 不当使用

## 代码可维护性缺陷

- 调用已过期方法
- 显式垃圾收集
- 非静态方法中设置静态变量
- 复制/粘贴错误
- 不可达代码

## 可疑代码

- 参数次序错误
- 格式错误

# C# 缺陷 Powered by Eric Lippert

## 资源泄露

- 数据库连接资源泄露
- 资源泄露
- `Socket & Stream` 泄露

## API 使用错误

- 已释放资源调用

## 并发数据访问异常

- 变量非原子更新
- 数据竞态条件

## 性能缺陷

- 低效率方法使用
- 在循环中连接字符串
- 冗余同步

## 程序假死

- 线程死锁
- 死循环

## 可疑代码

- 复制/粘贴错误
- 参数次序错误
- 格式错误

## 类层次结构不一致

- 调用 `base.close()` 或 `base.dispose()` 失败
- 父函数调用丢失

## 控制流缺陷

- 可疑的额外分号
- 不一致比较
- 不兼容的类型比较

## 空指针引用

- `Null` 检查后引用空指针
- 直接引用返回的空指针
- `Null` 检查前引用空指针

## 算术错误

- 错误移位操作
- 不正确的表达式
- 表达式计算过程中溢出

# 检测实例-SQL Injection

webgoat

Issues: By Snapshot | Outstanding Security Risks | Filters: Issue Kind, Classification

CID	Type	Impact	Status	First Detected	Owner	Classification	Severity	Action	Component	Category	File
11593	Cross-site scripting	High	Triaged	09/01/12	jon	Bug	Unspecified	Fix Requested	Other	High impact security	/data00/src/webgoat/we...
11970	SQL injection	High	New	07/23/14	admin	Unclassified	Unspecified	Undecided	Other	High impact security	/Demo/projects/webgo...
11929	Regular expression injection	Low	New	07/23/14	admin	Unclassified	Unspecified	Undecided	Other	Low impact security	/Demo/projects/webgo...
11917	OS Command Injection	High	New	07/23/14	admin	Unclassified	Unspecified	Undecided	Other	High impact security	/Demo/projects/webgo...
11914	Filesystem path or filename manipulation	High	New	07/23/14	admin	Unclassified	Unspecified	Undecided	Other	High impact security	/Demo/projects/webgo...
11873	Cross-site request forgery	High	New	07/23/14	admin	Unclassified	Unspecified	Undecided	Other	High impact security	/Demo/projects/webgo...
12097	Cross-site scripting	High	New	07/23/14	Unassigned	Unclassified	Unspecified	Undecided	Other	High impact security	/Demo/projects/webgo...

1 of 199 issues selected

Page 1 of 1

ThreadSafetyProblem.java

```
59. tainted_path_call: org.owasp.webgoat.session.ParameterParser.getRawParameter(java.lang.String, java.lang.String) returns the tainted data.
60.                                         currentUser = s.getParser().getRawParameter(USER_NAME, "");
61.                                         originalUser = currentUser;
62.
63.                                         // Store the user name
64.                                         String user1 = new String(currentUser);
65.
66.                                         Element b = ECSFactory.makeButton("Submit");
67.                                         ec.addElement(b);
68.                                         ec.addElement(new P());
69.
70.                                         if (!"".equals(currentUser))
71.                                         {
72.                                             Thread.sleep(1500);
73.
74.                                             // Get the users info from the DB
75.
76.                                         String query = "SELECT * FROM user_system_data WHERE user_name = '" + currentUser + "'";
77.                                         Statement statement = connection.createStatement(resultSet.TYPE_SCROLL_INSENSITIVE,
78.                                         resultSet.CONCUR_READ_ONLY);
79.
80.                                         ResultSet results = statement.executeQuery(query);
81.
82.                                         while (results.next())
83.                                         {
84.                                             String user2 = results.getString("user_name");
85.                                             String user3 = results.getString("user_password");
86.
87.                                             if (user1.equals(user2) && user3.equals("password"))
88.                                             {
89.                                                 // User is authenticated
90.                                                 // ...
91.                                             }
92.                                         }
93.
94.                                         // ...
95.
96.
97.
```

◆ CID 11970 (#1 of 1): SQL injection (SQLI)

7. **sql\_taint**: Insecure concatenation of a SQL statement. The value `org.owasp.webgoat.lessons.ThreadSafetyProblem.currentUser` is tainted.

📍 Remediation for SQL injection in JDBC: Specific advice for SQL string

- Refactor the JDBC code to use the PreparedStatement API versus Statement.
- Add a positional parameter to the SQL statement using "?".
- Bind the tainted value to the parameter using the setString method: `PreparedStatement.setString(1, org.owasp.webgoat.lessons.ThreadSafetyProblem.currentUser)`.

More Information

8. **sql\_sink**: Passing the tainted value `query` to the SQL API `java.sql.Statement.executeQuery(java.lang.String)` may allow an attacker to inject SQL.

Result Set `resultSet.CONCUR_READ_ONLY`;

# Copy-paste error in real-world code

```
if (returns!=null) {  
    r.retvals = ScopeParser.parseTypedArgList(returns,  
    returns.getText(), g.tool.errMgr);  
    r.retvals.type = AttributeDict.DictType.RET;  
    r.retvals.ast = returns;  
}  
  
if (locals!=null) {  
    r.locals = ScopeParser.parseTypedArgList(locals,  
    locals.getText(), g.tool.errMgr);  
    r.locals.type = AttributeDict.DictType.LOCAL;  
    r.locals.ast = returns;  
}
```

# C语言静态分析

- Checker描述(metal 语言)

```
#include "extend-lang.hpp"
START_EXTEND_CHECKER( basic_leak, int_store );
ANALYZE_TREE()
{
    // Do not unmangle
    // We want the actual C functions; with unmangling we could get
    // e.g. Foo::malloc
    CallSite malloc( "malloc", /*unmangle*/false );
    CallSite free( "free", /*unmangle*/false );
    LocalVar lv;
    AnySubpart lv_sp( lv );
    if( MATCH( lv_sp = malloc( _ ) ) ) {
        SET_STATE( lv_sp, 1 );
        ADD_EVENT( lv_sp, "alloc", "Allocation assigned to " << lv_sp );
    }
    else if( MATCH( free( lv_sp ) ) ) {
        CLEAR_STATE( lv_sp );
    }
}
ANALYZE_END_OF_PATH()
{
    const ASTNode *t;
    int v;
    FOREACH_IN_STORE( t, v ) {
        COMMIT_ERROR( t, "leaked", t << " was not freed before the end of path" );
    }
}
END_EXTEND_CHECKER();
MAKE_MAIN( basic_leak )
```

## 检测代码：

```
1  #include<stdlib.h>
2  test()
3  {
```

**(1) Event alloc:** Allocation assigned to "p"  
Also see events: [\[leaked\]](#)

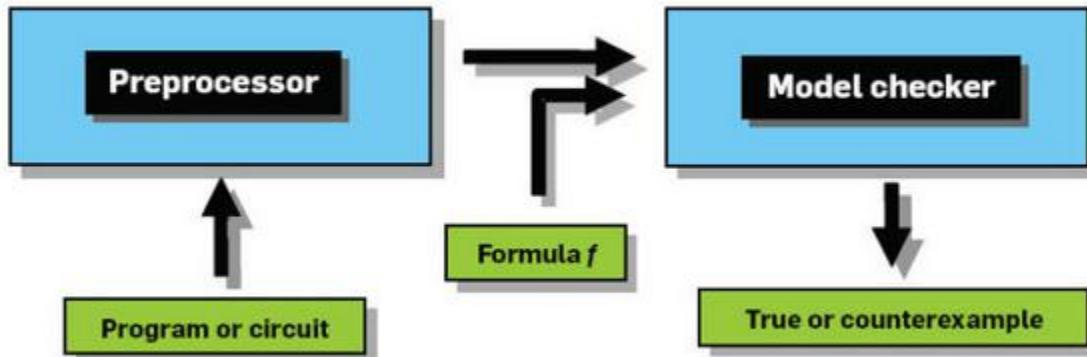
```
4  void *p = malloc(10);
5  /* ... */
6  //free(p);
```

**(2) Event leaked:** "p" was not freed before the end of path  
Also see events: [\[alloc\]](#)

```
7  }
```

# XGCC系统

- 符号执行
  - 不执行程序，用符号值表示程序变量的值，模拟程序执行
  - 可以分析代码的所有/部分语义信息
  - 避免状态爆炸
- 模型检验



# C/C++ 缺陷-Part 1

## 资源泄露

- 内存泄露
- Microsoft COM 内存泄露
- Object资源泄露
- 不当`delete`

## 未初始化变量

- 返回语句丢失
- 未初始化的指针/标量/数组 读写
- 类或结构体中未初始化的数据成员

## 并发缺陷

- 死锁
- 竞态条件 (Race conditions)
- 阻塞调用误用

## 算术错误

- 负变量不当使用
- 异常符号扩展
- 整数溢出
- 除零异常

## 内存崩溃

- 内存访问溢出
- 字符串长度计算错误
- 缓冲区溢出
- 写指针溢出
- 负数组索引写入
- 内存错误分配
- 错误的内存释放

## 非法内存访问

- 不正确的`delete`操作
- 溢出指针读取
- 越界读取
- 返回指针至本地变量
- 负数组索引读取
- 已释放指针读/写
- 不兼容的指针转换

## 控制流缺陷

- 逻辑/结构死代码
- `Switch`语句中`break`遗失
- 非本地资源不当使用

# C/C++ 缺陷-Part 2

## 程序假死

- 死循环
- 双重锁或解锁丢失
- 负循环边界值
- 线程死锁
- 持锁过程中调用sleep()

## 空指针引用

- Null检查后引用空指针
- 直接引用返回的空指针
- Null检查前引用空指针

## 错误处理缺陷

- 未验证的返回值
- 未获取异常
- 负变量不当使用

## 代码维护性缺陷

- 多返回语句
- 无效变量

## 异常代码

- 复制/粘贴错误

## 不安全的数据处理

- 不可信的循环数据源
- 使用非可信数据源读写数组/指针
- 使用非可信数据源格式化字符串

## 性能缺陷

- 值传递大参数
- 使用大堆栈

## 安全措施违反

- 缓冲区溢出
- 固定长度缓冲区写入
- 非安全函数调用
- 非安全临时文件使用
- 检查/使用时间不一致
- 用户空间指针不当使用

## API错误使用

- 非安全chroot调用
- 错误的迭代器使用
- printf() 参数不匹配

# 安全检查

- 定义Tainted Data

```
unsigned int packet_get_int() {
    return __coverity_tainted_data_return__();
}

void custom_read(int fd, void *buf) {
    __coverity_tainted_data_argument__(buf);
}
```

- HeartBleed

```
static unsigned int * __COV_TUI(unsigned int *i) {
    __coverity_tainted_data_argument__(i);
    return i;
}
#define TUI(i) *(__COV_TUI(&i))

#ifndef n2s(c,s) n2s(c,TUI(s))
```

# 检测实例-HeartBleed Bug

OpenSSL ▾ Config

CID	Type	Impact	Status	Count	First Detected	Owner	Classification	Severity	Ac
1201706	Untrusted value as argument	Medium	New	1	04/12/14	Unassigned	Unclassified	Unspecified	U

All 1 issue selected

Page 1 of 1

```
◆ ☰ 📁 /ssl/d1_both.c
1445
1446 #ifndef OPENSSL_NO_HEARTBEATS
1447 int
1448 dtls1_process_heartbeat(SSL *s)
1449 {
1450     unsigned char *p = &s->s3->rrec.data[0], *pl;
1451     unsigned short hbttype;
1452     unsigned int payload;
1453     unsigned int padding = 16; /* Use minimum padding */
1454
1455     /* Read type and payload Length first */
1456     hbttype = *p++;
1457
1458     n2s(p, payload);
1459     pl = p;
1460
1461     if (s->msg_callback)
1462         s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
1463                         &s->s3->rrec.data[0], s->s3->rrec.length,
1464                         s, s->msg_callback_arg);
1465
1466     if (hbttype == TLS1_HB_REQUEST)
1467     {
1468         unsigned char *buffer, *bp;
```

1. **byte\_swapping**: Performing a byte swapping operation on `p` implies that it came from an external source, and is therefore tainted.

2. **var\_assign\_var**: Assigning: `payload = ((unsigned int)p[0] << 8) | (unsigned int)p[1]`. Both are now tainted.

3. Condition `s->msg_callback`, taking true branch

4. Condition `hbttype == 1`, taking true branch

件开发大会

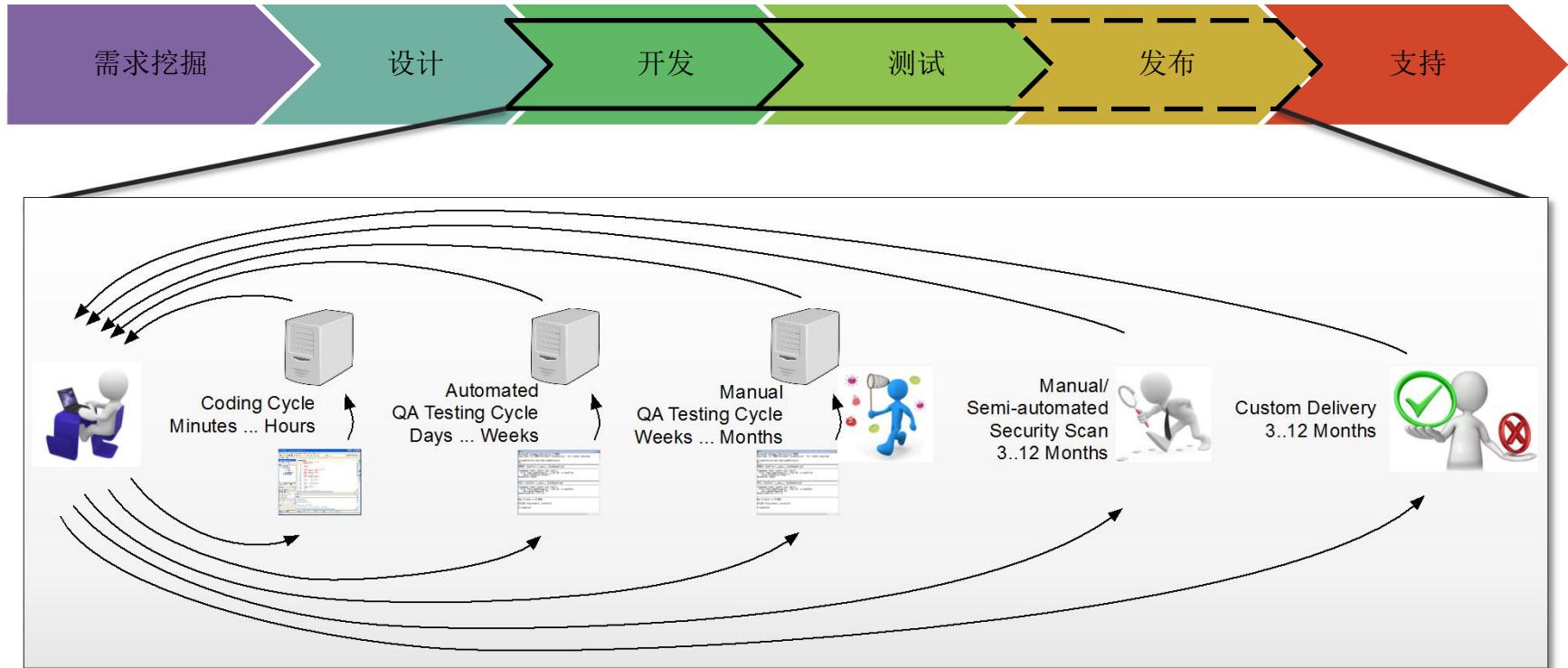
# Jenkins检测对比

类型	Coverity	FindBugs	Shared Defects
未处理的缺陷 (Null引用)	79	7	5
资源泄露	86	12	13
并发问题	22	10	9
重要的缺陷	187	29	27
代码规范, 最佳实践等	9	598	1
Bug 总数	196	627	28

# Freeradius缺陷检测对比

类型	Coverity	Clang	Shared Defects
内存问题	11	5	0
资源泄露	9	3	0
控制流缺陷, 并发访问等问题	83	30	1
重要的缺陷	103	38	1
代码规范, 最佳实践等	9	59	2
Total Bugs	121	97	3

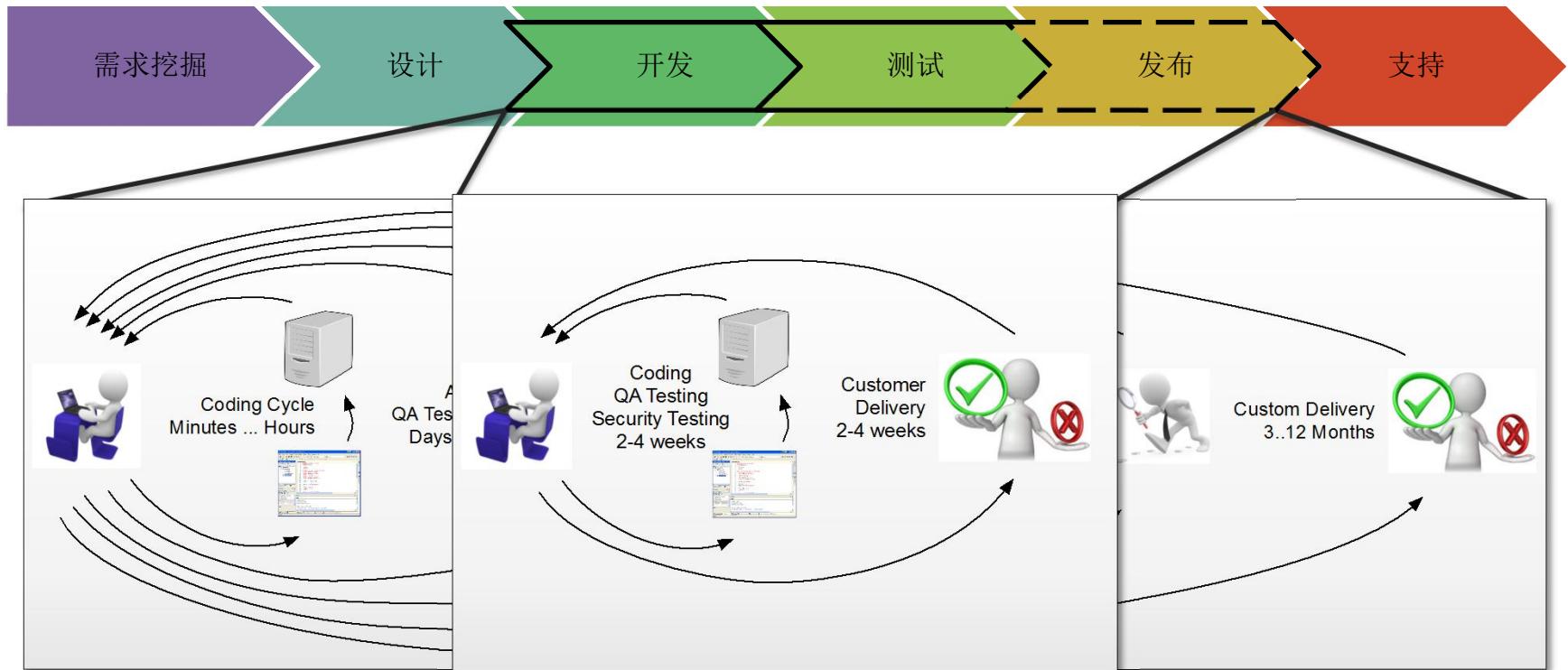
# “瀑布式开发流程”



经典的瀑布式开发流程：

- 代码开发与测试之间存在延迟
- QA经常由其他部门负责，在编码阶段完成后才能开始
- 安全保护基本是在“事发之后”才由单独的安全部门提供

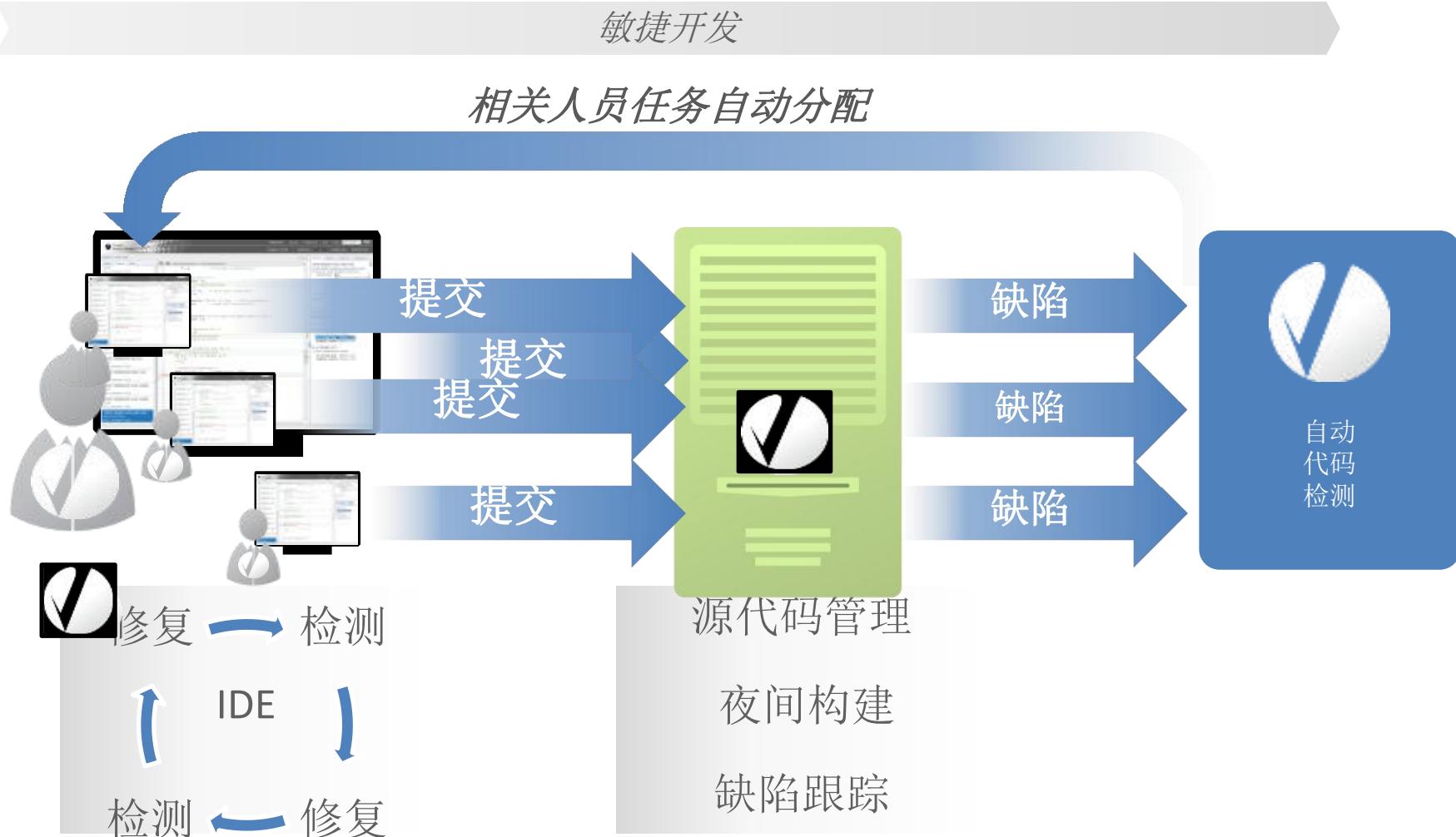
# “敏捷开发流程”



理想的敏捷开发流程：

- 将整个实施和验证流程缩短成2-4周的“冲刺”(sprints)
- 开发好的功能只有在经过全面验证之后才被“接受”
- 为取得成功, QA及安全测试都必须是深入的自动化测试

# 软件生命周期集成



# 示例： C/C++

- 这片代码来自PuTTY
  - PuTTY是面向Win32和Unix平台以及xterm终端仿真程序而免费实施的Telnet与SSH

```
2005年6月20日星期六上午10:25
2 static char *foo(char *s, int len)
{
    char *p = malloc((len + 1) * sizeof(char));
    memcpy(p, s, len);
    p[len] = '\0';
    return p;
}
```

```
char *p = snewn(len + 1, char);
memcpy(p, s, len);
p[len] = '';
return p;
}
```

# CVE-2005-0467

**CVE LIST**   **COMPATIBLE PRODUCTS**   **NEWS — FEBRUARY 27, 2012**   **SEARCH**

**Common Vulnerabilities and Exposures**  
*The Standard for Information Security Vulnerability Names*

**TOTAL CVEs: 49296**

[HOME](#) > [CVE](#) > [CVE-2005-0467 \(UNDER REVIEW\)](#)

[About CVE](#)  
Terminology  
Documents  
FAQs

**CVE List**  
About CVE Identifiers  
Search CVE  
Search NVD  
Updates & RSS Feeds  
Request a CVE-ID

**CVE In Use**  
CVE Adoption  
CVE-Compatible Products  
NVD for CVE Fix Information  
More...

**News & Events**  
Calendar  
Free Newsletter

**Community**  
CVE Editorial Board  
Sponsor

**Contact Us**  
Search the Site

[Printer-Friendly View](#)

<b>CVE-ID</b>	
<b>CVE-2005-0467</b> (under review)	<a href="#">Learn more at National Vulnerability Database (NVD)</a> • Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings
<b>Description</b>	
Multiple integer overflows in the (1) sftp_pkt_getstring and (2) fpx_readdir_recv functions in the PSFTP and PSCP clients for PuTTY 0.56, and possibly earlier versions, allow remote malicious web sites to execute arbitrary code via SFTP responses that corrupt the heap after insufficient memory has been allocated.	
<b>References</b>	
<b>Note:</b> <a href="#">References</a> are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.	
<ul style="list-style-type: none"><li>• IDEFENSE:20050221 Multiple PuTTY SFTP Client Packet Parsing Integer Overflow Vulnerabilities</li><li>• <a href="http://www.idefense.com/application/poi/display?id=201&amp;type=vulnerabilities">URL: http://www.idefense.com/application/poi/display?id=201&amp;type=vulnerabilities</a></li><li>• CONFIRM:<a href="http://www.chiark.greenend.org.uk/~sqtatham/putty/wishlist/vuln-sftp-string.html">http://www.chiark.greenend.org.uk/~sqtatham/putty/wishlist/vuln-sftp-string.html</a></li><li>• CONFIRM:<a href="http://www.chiark.greenend.org.uk/~sqtatham/putty/wishlist/vuln-sftp-readdir.html">http://www.chiark.greenend.org.uk/~sqtatham/putty/wishlist/vuln-sftp-readdir.html</a></li><li>• CONFIRM:<a href="http://www-1.ibm.com/support/docview.wss?uid=ssq1S1002414">http://www-1.ibm.com/support/docview.wss?uid=ssq1S1002414</a></li><li>• CONFIRM:<a href="http://www-1.ibm.com/support/docview.wss?uid=ssq1S1002416">http://www-1.ibm.com/support/docview.wss?uid=ssq1S1002416</a></li><li>• GENTOO:GLSA-200502-28</li><li>• <a href="http://www.gentoo.org/security/en/glsa/glsa-200502-28.xml">URL: http://www.gentoo.org/security/en/glsa/glsa-200502-28.xml</a></li><li>• SECUNIA:14333</li><li>• <a href="http://secunia.com/advisories/14333">URL: http://secunia.com/advisories/14333</a></li><li>• SECUNIA:17214</li><li>• <a href="http://secunia.com/advisories/17214">URL: http://secunia.com/advisories/17214</a></li><li>• XF:putty-sftpktgetstring-bo(19403)</li><li>• <a href="http://xforce.iss.net/xforce/xfdb/19403">URL: http://xforce.iss.net/xforce/xfdb/19403</a></li></ul>	

**CVE List**  
About CVE Identifiers  
Editorial Policies  
Data Sources  
Reference Key/Maps  
Search Tips  
Updates & RSS Feeds  
Obtain a CVE Identifier

**ITEMS OF INTEREST**  
Terminology  
NVD

```
static void sftp_pkt_getstring(struct sftp_packet *pkt,
                               char **p, int *length)
{
    *p = NULL;
    if (pkt->length - pkt->savedpos < 4)
        return;
    /* length value is taken from user-supplied data */
    *length = GET_32BIT(pkt->data + pkt->savedpos);
    pkt->savedpos += 4;
    /* this check will be passed if length < 0 */
    if (pkt->length - pkt->savedpos < *length)
        return;
    *p = pkt->data + pkt->savedpos;
    pkt->savedpos += *length;
}
```

/\* sftp\_pkt\_getstring call with controlled len value \*/

```
sftp_pkt_getstring(pktin, &hstring, &len);
```

```
...
handle = snew(struct fxp_handle);
/* heap corruption will occur if len == -1 */
handle->hstring = mkstr(hstring, len);
handle->hlen = len;
sftp_pkt_free(pktin);
return handle;
...
```

```
static char *mkstr(char *s, int len)
{
    char *p = snewn(len + 1, char);
    memcpy(p, s, len);
    p[len] = '\0';
    return p;
```

# 如何处理?

The screenshot illustrates the Coverity IDE interface, specifically the 'putty' project view. The interface is divided into several sections:

- Left Sidebar:** Lists various project metrics and filters, including '安全及其他问题的统一视图' (Unified View of Security and Other Issues), '对缺陷进行清晰描述' (Clearly Describe Defects), '清晰描述显示缺陷的控制流' (Clearly Describe the Control Flow of Defects), and '面向宏和进程间缺陷的联防 (inline) 选项' (Countermeasures for Defects between Macros and Processes).
- Top Bar:** Includes the Coverity logo, project name 'putty', and navigation links for Configuration, Help, Admin User, and Enter CID(s).
- Table View:** A table showing 176 issues selected, with columns for CID, Type, Impact, Status, First Detected, Owner, and Classification. The first four rows are highlighted in red, corresponding to the 'CID 12602 (#1 of 1): Untrusted value as argument (TAINTED\_SCALAR)' entry.
- Code Editor:** Displays the source code for `/scratch/build/putty-0.56/psftp.c`. The code shows a sequence of function calls and conditionals. A red arrow points from the '清晰描述显示缺陷的控制流' box to the code editor, highlighting the flow of tainted data.
- Details Panel:** Shows a detailed view of CID 12602, including the error message 'CID 12602 (#1 of 1): Untrusted value as argument (TAINTED\_SCALAR)' and '7. tainted\_data: Passing tainted variable "pktin" to a tainted sink. [hide details]'. It also lists other related errors like '2. tainted\_data\_transitive' and '2. parm\_assign'.
- Triage Panel:** A right-hand panel for managing the defect. It includes fields for Classification (Unclassified), Severity (Unspecified), Action (Undecided), Ext. Reference (Type attribute text), and Owner (Unassigned). A text area for comments and 'Apply' buttons are also present.
- Bottom Navigation:** Includes links for 'Projects and Streams', 'Detection History', 'Triage History', and a 'Occurrences' section which lists the contributing events to the defect.

# Q/A



10号展位😊



# THANKS!