为软件产品“探伤”

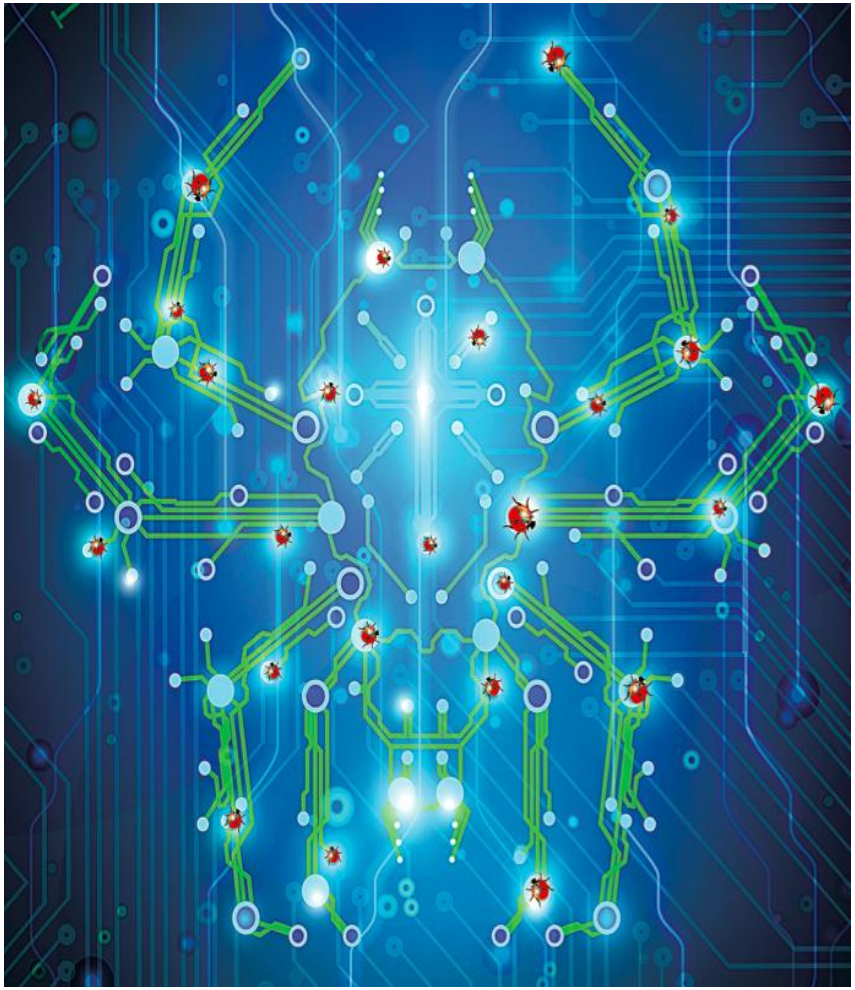
—以攻击者思维分析软件安全问题

苏璞睿
中国科学院软件研究所
可信计算与信息保障实验室

二〇一四年十一月

软件产品的“伤”

- 软件产品“探伤”难度大
  - 隐蔽性
    - 一条错误指令隐藏在G为单位的指令序列中
    - 可能的执行路径更是海量数据
  - 复杂性
    - 软件自身的复杂性
    - 漏洞机理的复杂性
    - 利用模式的复杂性
- 危害严重
  - 首款网络战武器Stuxnet:漏洞是关键
  - 漏洞是APT（Advanced Persistent Threat）攻击成功的主要条件
- 原因复杂
  - 无意造成的错误
  - 有意设计的后门



DARPA启动了多个相关项目

- **Binary Executable Transforms (BET)**
  - 自动分析二进制代码并确定功能组件，自动切片并提取格式化的输入输出，综合利用静态、动态分析手段，提高对二进制代码的分析理解能力；利用形式化方法验证功能组件的相关属性，设计中间语言以支持程序切片等分析工作；研发相关的支撑性工具

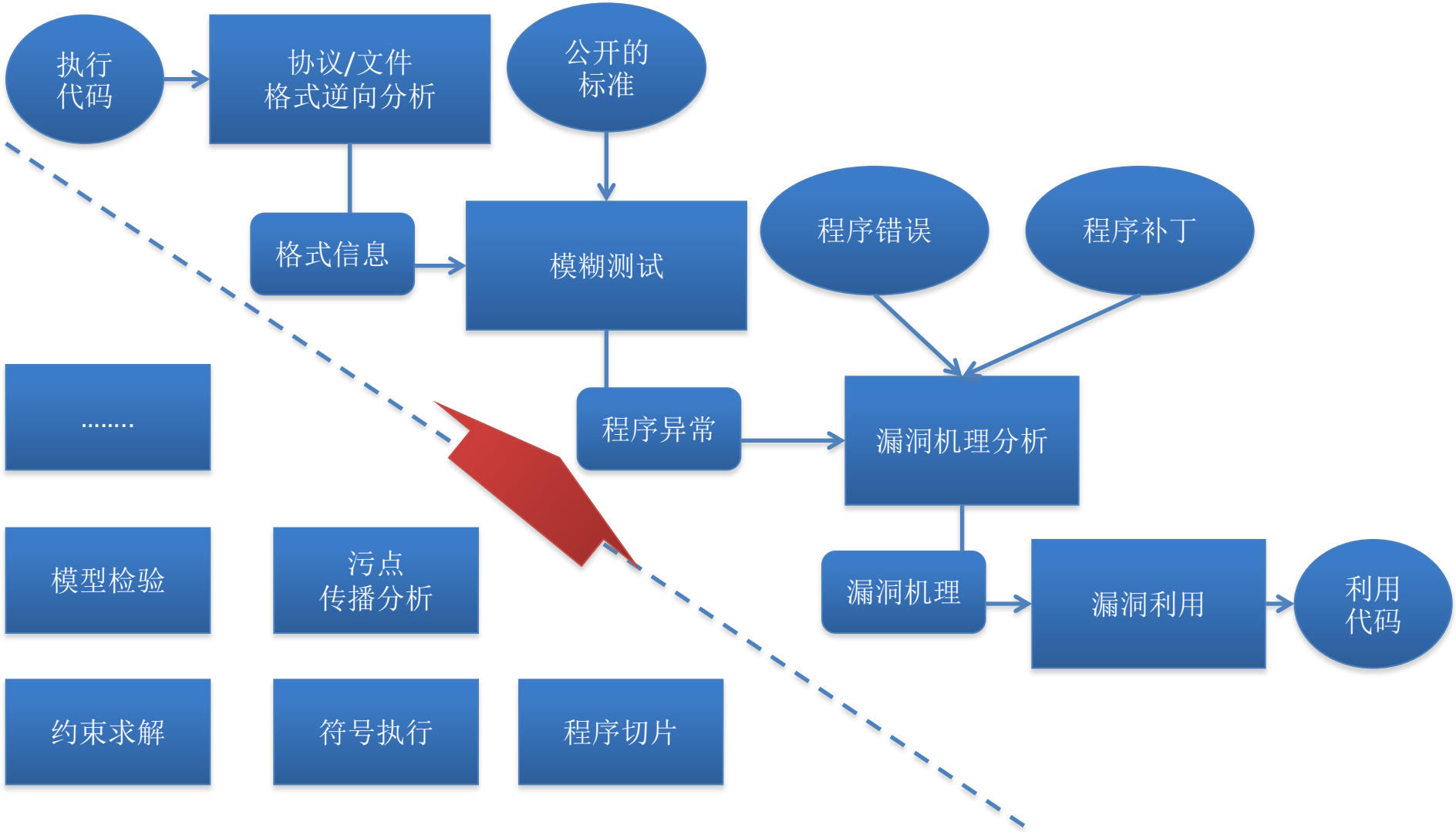
- **VETTING COMMODITY IT SOFTWARE AND FIRMWARE (VET)**
  - 具备对商业IT产品中可能存在的恶意功能的分析和检测能力;具备对IT产品中看似无意造成的漏洞（accidental-seeming）、功能可能存在的负面作用的分析能力；在对手恶意欺骗的情况下的漏洞、恶意行为的分析检测能力

- **CYBER GRAND CHALLENGE (CGC)**
  - 网络安全挑战赛，重点内容包括软件的逆向分析、漏洞挖掘与利用等内容，以期发现系统的安全缺陷，探索安全防御措施

国内相关项目

- NSFC“可信软件基础研究”重大研究计划
  - “航天嵌入式软件可信性保障集成环境和示范验证与应用”
  - “可信嵌入式软件系统试验环境与示范应用”
  - “可信软件理论、方法集成与综合试验平台”
  - “基于认识与理解途径的度量与评估体系及支撑技术研究”
  - “面向软件可信性演进的软件测试技术研究”
  - “航天嵌入式软件可信性构造与验证的关键技术研究”
  - “基于测试的软件可信性增长模型及其评估方法研究”
  - “面向性质的可信软件建模与时序性质验证及支持工具”
  - “基于模型的嵌入式软件测试与验证技术及针对国产列车控制系统的实例研究”
  - “航天嵌入式软件缺陷检测方法研究、系统研发及应用”
  - ……

攻击者的工作模式


执行
代码
协议/文件
格式逆向分析
公开的
标准
格式信息
模糊测试
程序错误
程序补丁
.....
程序异常
漏洞机理分析
模型检验
污点
传播分析
漏洞机理
漏洞利用
利用
代码
约束求解
符号执行
程序切片

相关研究进展

协议逆向分析

- Aligot(CCS'12),Dispatcher(CCS'09),ReFormat(ESORICS'09), Polyglot(CCS'07), Discover(USNIX'07), TUPNI(CCS08), Prospex(S&P'09)

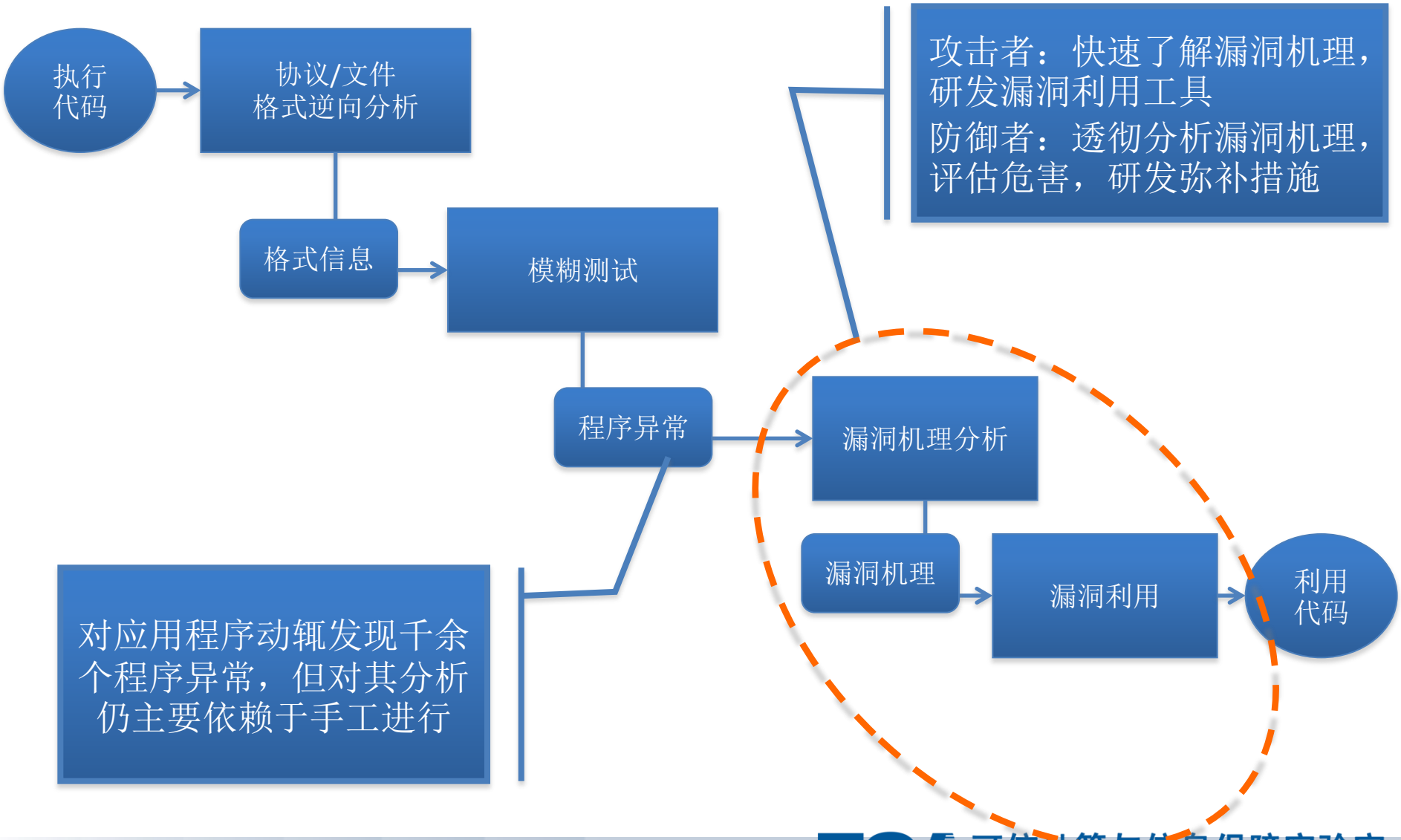
模糊测试

- TaintScop(TISSEC'11), Automated whitebox fuzz testing(NDSS'08),
- Grammar-Based Whitebox Fuzzing (PLDI'08)
- SPIKE, PEACH, SULLEY

软件漏洞分析与利用

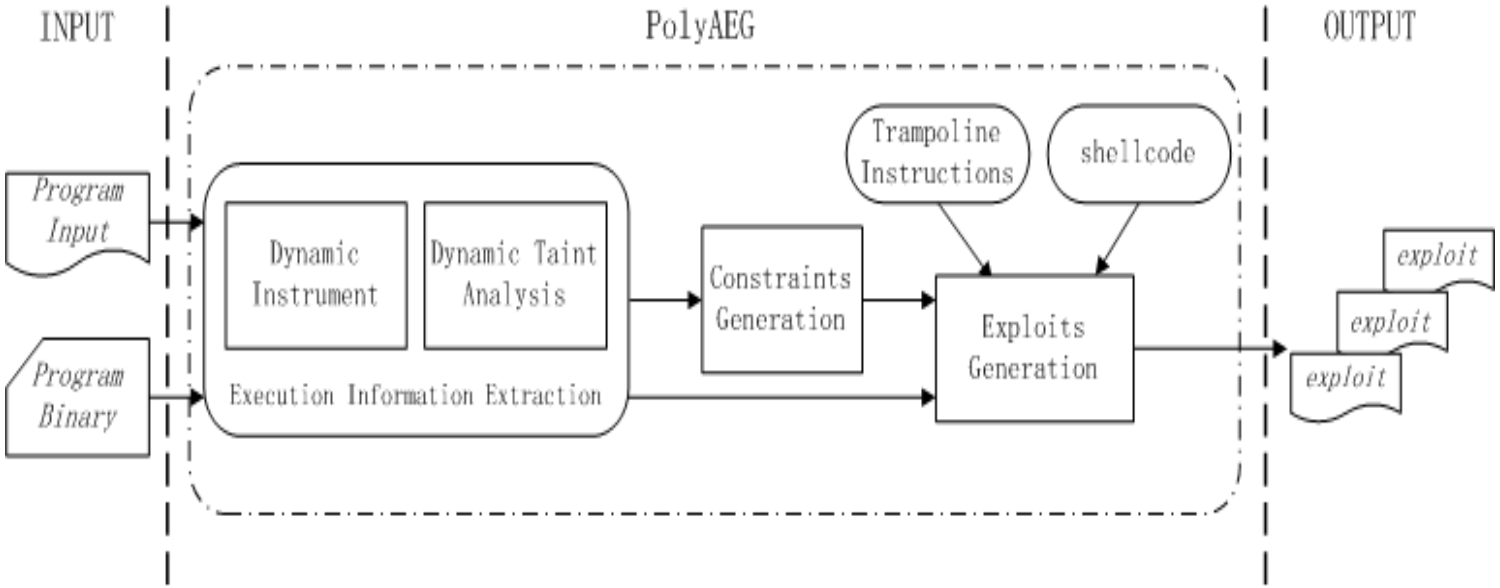
- Differential Slicing(S&P'11),Comparative Causality(ICSE'13),DTA++(NDSS'11)
- Automating Information Flow Analysis of Low Level Code(CCS'14)
- AEG on Source Code(NDSS'11),APEG(S&P'08)

存在的问题及挑战


执行
代码
协议/文件
格式逆向分析
格式信息
模糊测试
程序异常
攻击者：快速了解漏洞机理，研发漏洞利用工具
防御者：透彻分析漏洞机理，评估危害，研发弥补措施
漏洞机理分析
漏洞机理
漏洞利用
利用
代码
对应用程序动辄发现千余个程序异常，但对其分析仍主要依赖于手工进行

软件漏洞利用自动生成

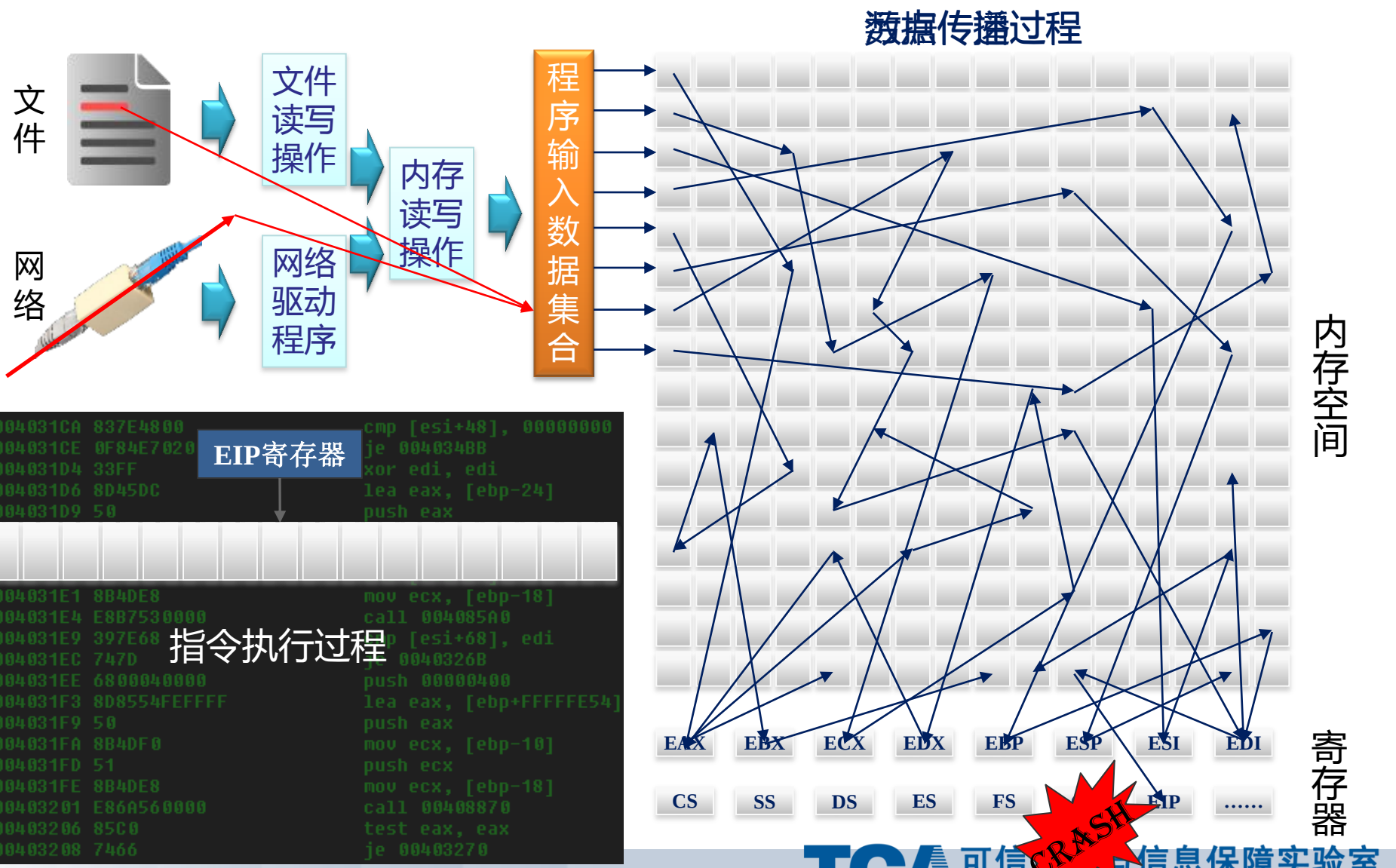
- PolyAEG（Automatic Exploit Generation ）系统
- 从攻击者的角度寻找漏洞利用的途径，并根据潜在的攻击途径，构造不同的利用样本，以剖析漏洞利用的机理，为漏洞防御提供支撑



需要解决的三个关键问题

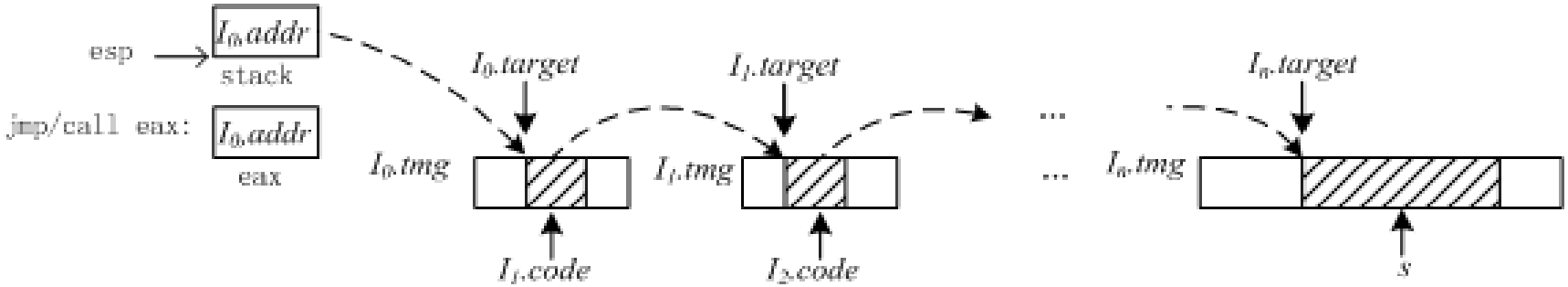
- 关联内存区域与输入数据
  - 在程序错误发生时，哪些数据区域可用于构造攻击代码
  - 构造攻击代码的区域与哪些输入对应
- 构造所有可能的攻击路径
  - 确定所有可能的劫持点
  - 构造不同的跳转链
- 漏洞利用的有效性评估
  - 输入的数据应保证攻击代码成功执行
  - 正确触发异常、准确跳转、正确执行Shellcode

关联内存数据与输入——动态污点分析

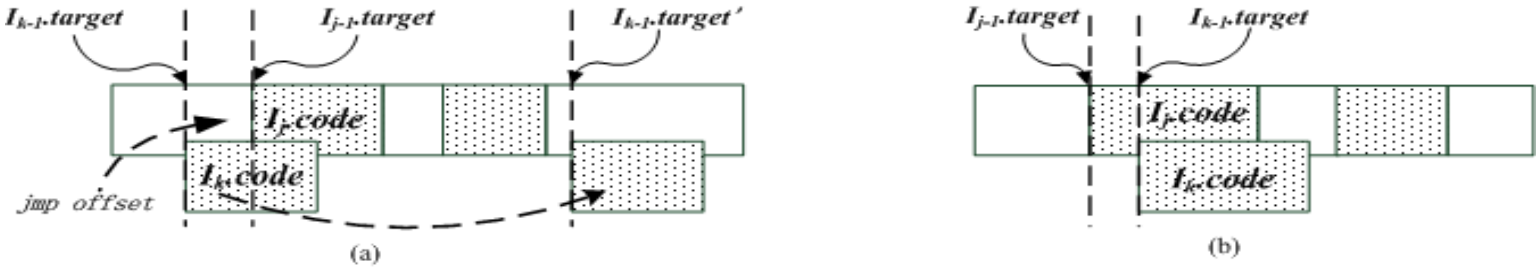


构造不同的攻击路径

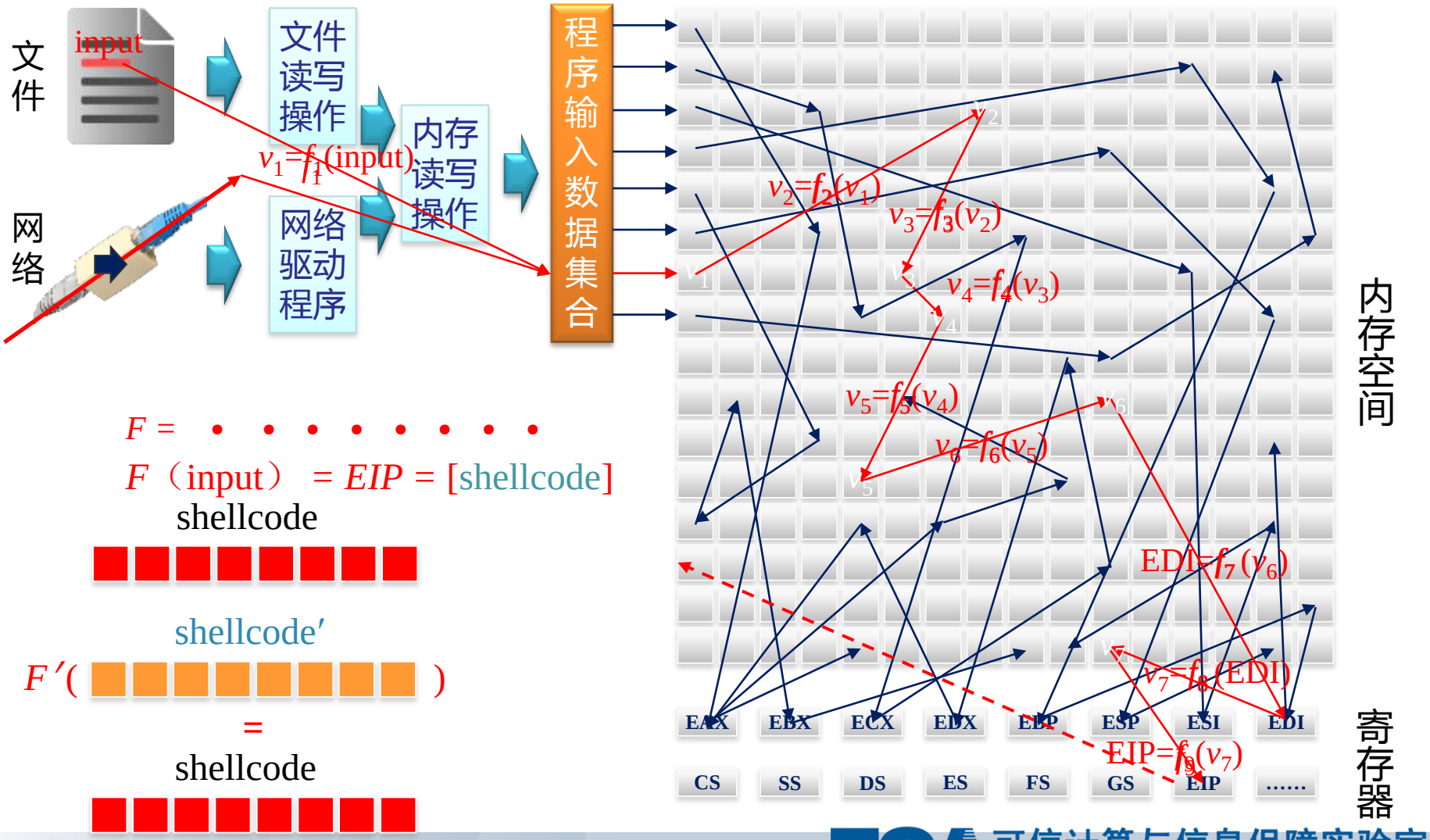
- 跳转链的构建



- 地址冲突的解决方案



确保攻击路径有效


文件
input
网络
文件读写操作
网络驱动程序
内存读写操作
程序输入数据集合
$v_1=f_1(input)$
$v_2=f_2(v_1)$
$v_3=f_3(v_2)$
$v_4=f_4(v_3)$
$v_5=f_5(v_4)$
$v_6=f_6(v_5)$
$EDI=f_7(v_6)$
$v_7=f_8(EDI)$
$EIP=f_9(v_7)$
内存空间
寄存器
EAX EBX ECX EDX EBP ESP ESI EDI
CS SS DS ES FS GS EIP .....
$F=$ • • • • • • • •
F（input） $= EIP =$ [shellcode]
shellcode
shellcode′
$F'($ $)$
=
shellcode

确保攻击路径有效

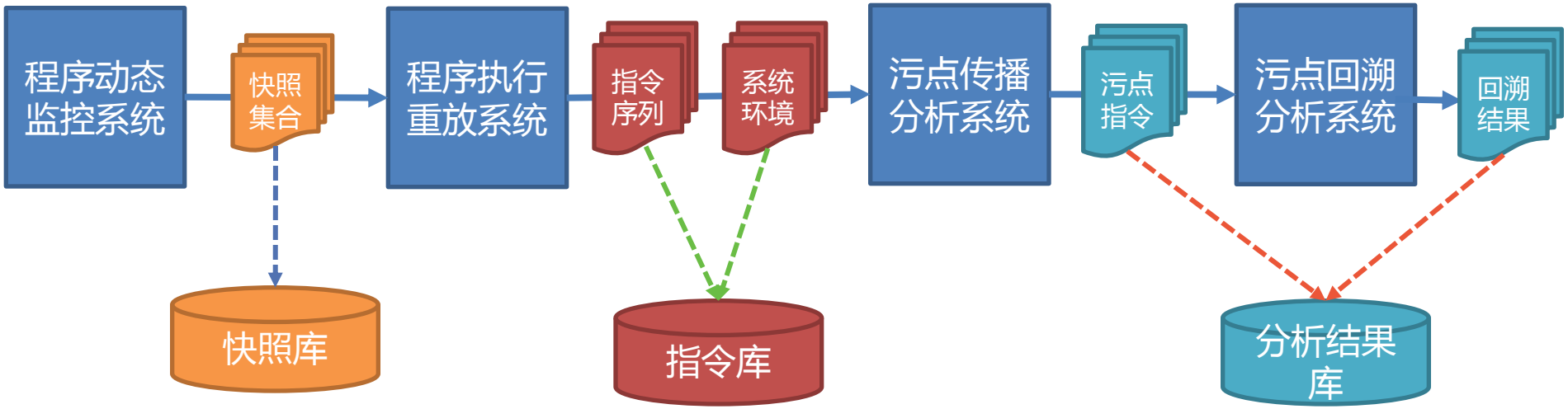
- 约束条件的生成
  - 目的
    - 确保填写的内容仍可成功触发程序异常.
  - 思路
    - 在程序执行路径中，每一个受污点源影响的分支（Tainted Branch），将污点数据符号化，抽取约束表达式
    - 在每一个受污点源影响的分支，回溯分析路径，进行实例化符号执行，产生路径约束表达式

系统实现与实验

- 代码量： 30,000 lines of C/C++
- 基础工具
  - 动态污点传播分析：AOTA系统(Application Oriented Tainting Analysis)
  - 约束求解：Z3 SMT solver

系统实现与实验

- AOTA动态污点传播系统
  - 系统基于Qemu扩展实现，
  - 支持执行过程重放，动态监控过程与污点分析分离，保证了分析效率
  - 支持回溯分析能力，可选择性的支持控制依赖、指针依赖等隐式污点传播分析功能



系统实现与实验

- 实验样本–应用程序及漏洞

软件名称	CVE编号	漏洞类型	异常文件大小
IrfanView v3.99	CVE-2007-2363	return address	2648
CoolPlayer v2.19.2	CVE-2009-1437	return address	601
MP3 CD Ripper v2.6	CVE-2011-5165	return address	4432
AutoPlay v1.33	CVE-2009-0243	return address &function pointer	701
WAV Converter v1.5	CVE-2010-2348	function pointer	8208
FloatFtp v1.00	CNNVD-201302-349	return address	981
Aviosoft DVD Player	CVE-2011-4496	return address	1472
Internet Download Manager V6.12		return address	2340

系统实现与实验

- 实验样本—ShellCode

序号	类型	功能	长度
1	CMD	启动远程Shell	21
2	MSG	弹出MSG窗口	45
3	ADD	添加本地账户	233
4	DWN	下载可执行代码	297
5	STC	启动一个计算器	226
6	REC	反向链接端口	366
7	STN	启动记事本程序	86
8	BLP	绑定侦听端口	338

系统实现与实验

- 漏洞利用生成结果统计

软件名称	ShellCode	劫持点	利用数量
IrfanView v3.99 CVE-2007-2363	DWN	3	4724
CoolPlayer v2.19.2 CVE-2009-1437	CMD	29	3750
MP3 CD Ripper v2.6 CVE-2011-5165	ADD	1	3399
AutoPlay v1.33 CVE-2009-0243	MSG	3	282
WAV Converter v1.5 CVE-2010-2348	STC	4	180
FloatFtp v1.00 CNNVD-201302-349	BLP	4	367
Aviosoft DVD Player CVE-2011-4496	REC	1	126
Internet Download Manager V6.12	STN	3	112

系统实现与实验

- 性能评估

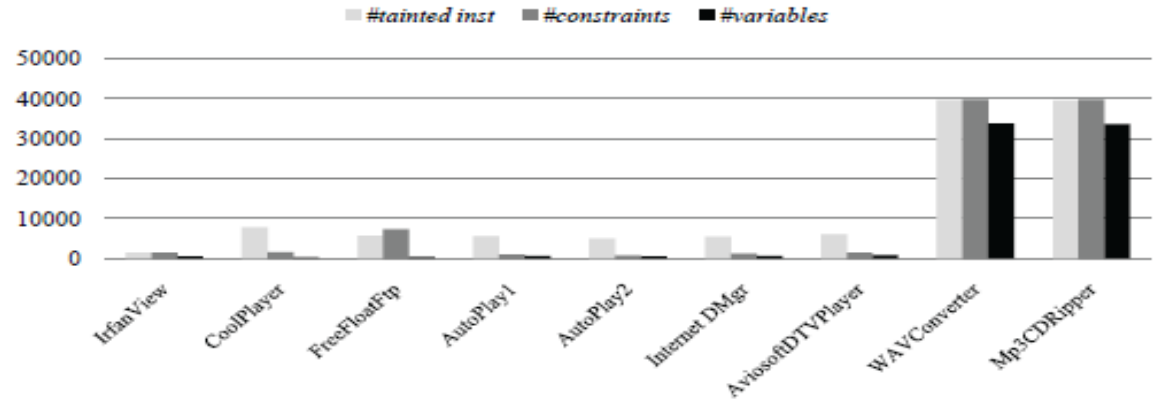


Fig. 8. The statistics about tainted instructions, constraint formulas and symbolic variables for all programs.

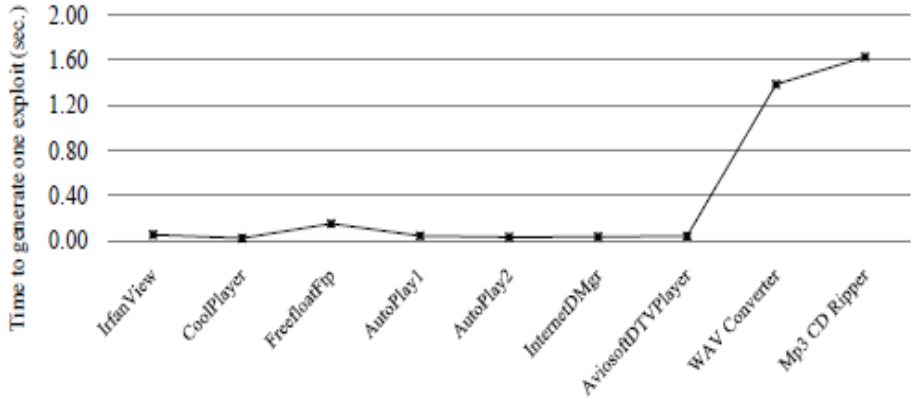


Fig. 9. The time overhead on one exploit generation.

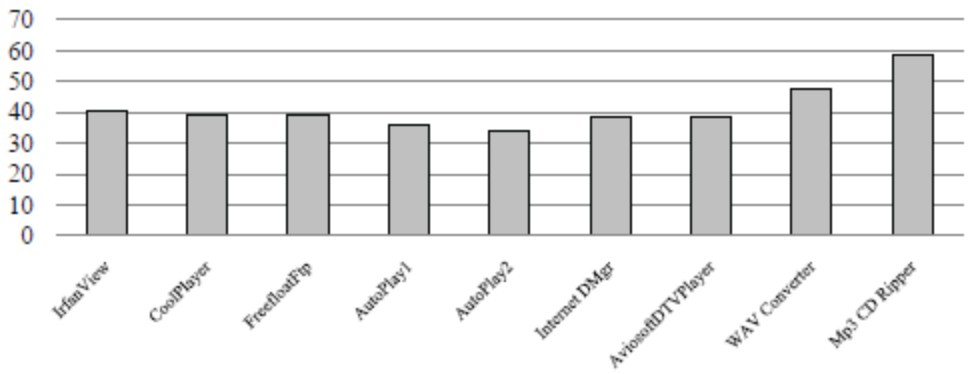


Fig. 10. The memory overhead on polymorphic exploit generation.

相关工作进展

相关工作	方法概述	依赖条件	对比分析
APEG(S&P'08) by CMU	通过对比补丁，构造利用代码	漏洞程序和发布的补丁	仅触发漏洞，无法实现完整利用
AEG(NDSS'11) by CMU	利用符号执行方法提取漏洞信息，构造约束，生成利用	漏洞程序源代码	可生成多种不同利用模式
AXGEN（MSc Thesis'02）by Oxford	通过数据流分析方法构造跳板指令和shellcode与程序输入之间的关系，利用求解器求解构造利用代码	漏洞程序及异常数据	跳板指令选取和定位策略较为简单，对大多数漏洞程序难以有效生成利用
MAYHEM(S&P'12) by CMU	利用混合符号执行（hybrid symbolic execution)和基于索引化的内存模型（index-based memory modeling)生成验证性的利用代码	漏洞程序异常数据	将漏洞发掘与利用完整整合，利用代码仅属于验证性质，无法实际利用
PolyAEG by ISCAS	利用动态污点分析方法获取跳板指令和shellcode可以存放的位置，利用实例化符号执行来获取路径约束，构造多种不同的跳转链，实现利用生成的多样性	二进制程序及异常数据	可生成可利用的样本；能够生成多样的攻击代码；不依赖于源程序

几个问题的探讨

- 隐式污点传播问题—“一步之遥”？
  - 控制依赖、指针依赖….
  - IE、PDF、Office等
- DEP（Data Execution Prevention）问题
  - ROP(Return-oriented programming)….
- 其他漏洞利用模式
  - 逻辑漏洞、信息泄露漏洞……

谢　谢！

苏璞睿 研究员/博导
中国科学院软件研究所可信计算与信息保障实验室副主任
Email:supurui@tca.iscas.ac.cn
电话：010-62661723
http://tca.iscas.ac.cn