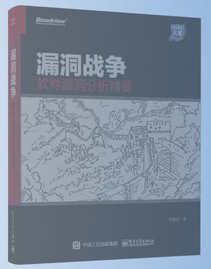
Fuzzing平台建设的研究与设计

riusksk

关于个人

- riusksk，江湖人称“泉哥”、“林大夫”
- 福建中医药大学—中西医骨伤专业
- 腾讯安全平台部研发安全团队
- 《漏洞战争：软件漏洞分析精要》作者



- MSRC (Microsoft Security Response Center) Most Valuable Security Researcher 2019
- Microsoft、Apple、Google、Adobe 四大国际厂商 60+ CVE 漏洞致谢

关于腾讯安全平台部


1
网络安全
宙斯盾
流量分析
DNS劫持监控
2
应用安全
洞犀、金刚
门神
啄木鸟
3
主机安全
洋葱
铁将军
4
数据安全
数据保护
安全质量
5
多维能力
Blade
TSRC
腾讯蓝军

目录

不同工种眼中的Fuzzing

开发人员：Fuzzing是什么东西？

大多缺乏专业安全技能

测试人员：我们也Fuzzing下，看产品功能还能用吗？

偏代码质量测试，技术相通，测试用例差异

安全人员：钱、钱、钱

看到的是价值

Fuzzing平台的价值思考

- 解放双手，减少重复的人力工作
- 让业余选手也能够干专业的事
- 多工种协同，覆盖更多攻击面的测试，提升漏洞发现率
- 在开发测试阶段，在上线前及时发现问题，降低外部攻击风险

Fuzzing平台工作流水线设计

协同Fuzzing

- 对开发、测试进行安全培训，推广libfuzzer/afl等工具，常见漏洞攻防原理等
- 借助开发、测试提供的test harness，快速接入Fuzzing平台进行测试
- 将Fuzzing测试的时间线覆盖到研发的各个阶段，比传统安全测试介入时间点更早
- 与刷CVE、刷SRC、打比赛不同，业务产品覆盖领域广泛且复杂，需要多人多工种协同，才能覆盖住

协同Fuzzing

面向企业产品安全，

Fuzzing的最佳实践途径是协同，

安全工作均如是！

Fuzzing三要素

目标

策略

样本

目标：攻击面分析

- 官方文档，比如MSDN、Apple 开发文档、Acrobat Javascript API 手册等等
- 官方和第三方提供的相关示例及测试开源项目，GitHub一直是最佳的搜索之地
- 业务需求设计文档，高效的攻击面分析捷径
- 业界公开的漏洞信息，比如Project Zero、ZDI
- 厂商的漏洞公告，补丁日是刷洞的最佳时机
- 人工逆向分析
- 源码审计

策略：变异之法

- **基于暴力的变异**：随机数据替换、插入、删除、数值增减、边界值替换、目录变异、数据拷贝覆盖等等

- **基于模板变异**：文件格式、协议格式、语法模板变异等等

- **基于覆盖率引导**：以AFL、libfuzzer、honggfuzz为代表的源码插桩方式、Qemu/Boch/Unicorn虚拟化技术、硬件PT技术、二进制动态插桩等等

变异策略插件式扩展

- 方便适配业务场景作定制化
- 漏洞应急
- 便于实现功能扩展，提升漏洞发现率
- 社区化/企业内多人协同开发

```
------------------------[ riufuzz v1.1 (WINWORD.EXE) ]------------------------
  Iterations : 2
    Run Mode : External (ole_fuzz.py)
    Run Time : 00:00:13
   Input Dir : [114] '/cygdrive/c/Users/Administrator/.../Desktop/公式doc'
     Threads : 1, CPUs: 8, CPU: 49.0%
       Speed : 0/sec (avg: 0)
     Crashes : 0 (unique: 0, blacklist: 0, verified: 0)
    Timeouts : 2 [10 sec]
    Coverage : N/A
---------------------------------[ LOGS ]---------------------------------
['ObjectPool', '_1329999399', '\x03ObjInfo']
[*] arthmetic fuzzing
0x00000001: 0x00 => 0x01
0x00000000: 0x00 => 0x44
0x00000002: 0x03 => 0x02
0x00000002: 0x02 => 0x01
0x00000000: 0x44 => 0x45
0x00000000: 0x45 => 0xCC
0x00000005: 0x00 => 0x01
0x00000001: 0x01 => 0x48
['ObjectPool', '_1329999399', 'Equation Native']
[*] knownints fuzzing
0x0000008F: 0x458f442f4150f410 => 0x0101010101010101
['WordDocument']
[*] arthmetic fuzzing
0x000021C3: 0x62 => 0x20
0x00001CD7: 0x5F => 0x12
0x00001FF3: 0x00 => 0x01
0x00001D34: 0x61 => 0x6D
0x0000124D: 0x00 => 0xFF
0x00000000: 0xEC => 0xF3
All threads done
```

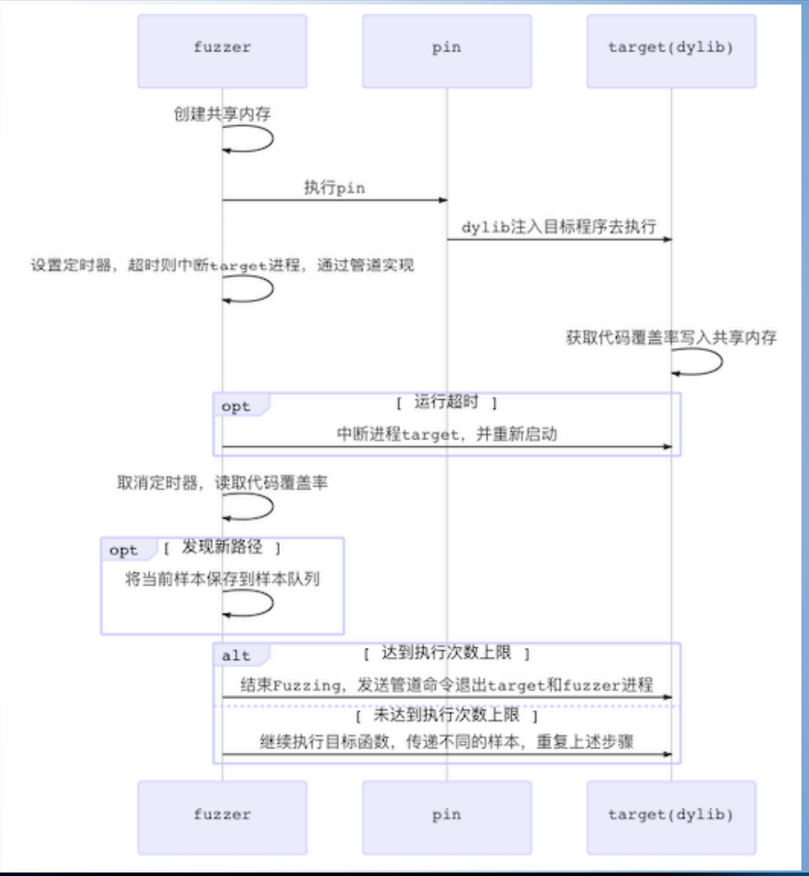
代码覆盖引导Fuzzer现状

Fuzzer	源码插桩	二进制插桩	程序分析	硬件PT	QEMU	Unicorn	Bochs	支持平台
afl	✓				✓(仅支持Linux)			Linux,macOS,Android
afl-go	✓		✓		✓(仅支持Linux)			Linux,macOS
libfuzzer	✓							Linux,macOS
honggfuzz	✓			✓(仅支持Linux)				Windows,Linux,macOS,Android
vuzzer		✓	✓					Linux
pe-afl		✓						Windows
winafl		✓						Windows
bochspwn							✓	Windows,Linux
syzkaller	✓							Linux
PTfuzzer				✓				Linux
afl-unicorn						✓		Linux
kAFL				✓	✓			Linux
TriforceAFL					✓			Linux
uniFuzzer						✓		Linux
fuzzilli	✓							Linux,macOS

缺乏macOS闭源程序的代码覆盖引导能力

macOS平台闭源程序的代码覆盖引导方案

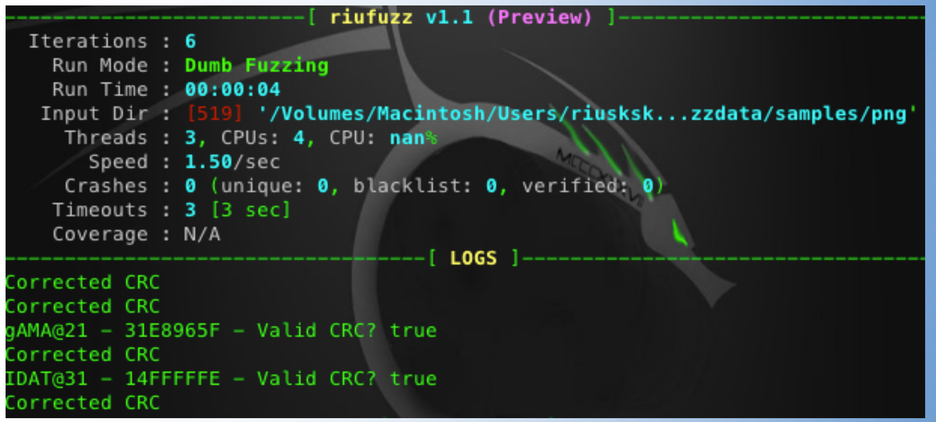
- 通过Pin进行二进制插桩获取代码覆盖率，共享内存记录覆盖率，利用管道通讯控制进程，触发新路径则保存到样本库
- riufuzz：基于honggfuzz二次开发
- DynamoRIO不支持macOS



变异后处理机制

- 变异后的样本可能导致CRC等校验不通过的问题，比如PNG，因此需要在变异后作修复处理；如果是开源项目，也可直接删除校验代码，在构造PoC时再修复

- 对于复合文档中的某特定格式的文件变异后，需要重组打包，比如变异docx中的图片、pdf中的字体图片等等，此过程注意后缀名的变更问题



样本收集

- 爬虫搜索引擎，支持的文件格式有限
- 爬虫Github，文件格式不限，但限制100页，需要变换关键词，也可Google+Github组合搜索
- 关键词变更：字典库、单词库、输入法词库
- 样本生样本：利用复合文档去提取出其它格式文件，比如pdf/docx中提取字体图片
- 通过改源码或Hook技术dump出二进制流样本数据

基于代码覆盖率的样本筛选

- 使用Pin插桩获取样本触发的代码覆盖情况
- 支持跨平台（Windows/Linux/macOS）
- 采用C++与Python开发

```
173  static VOID OnTrace(TRACE trace, VOID* v)
174  {
175      auto& context = *reinterpret_cast<ToolContext*>(v);
176      BBL bbl = TRACE_BblHead(trace);
177      ADDRINT addr = BBL_Address(bbl);
178
179      // Check if the address is inside a white-listed image.
180      if (!context.m_tracing_enabled || !context.m_images->isInterestingAddress(addr))
181          return;
182
183      // For each basic block in the trace.
184      for (; BBL_Valid(bbl); bbl = BBL_Next(bbl)) {
185          addr = BBL_Address(bbl);
186          BBL_InsertCall(bbl, IPOINT_ANYWHERE, (AFUNPTR)OnBasicBlockHit,
187              IARG_FAST_ANALYSIS_CALL,
188              IARG_THREAD_ID,
189              IARG_ADDRINT, addr,
190              IARG_UINT32, BBL_Size(bbl),
191              IARG_PTR, v,
192              IARG_END);
193      }
194  }
```

基于代码覆盖率的样本筛选

```
riusksk@MacBook-Pro ~/Downloads ‹ruby-2.1.4›
$ python corpus_distiller.py
[Usage]: corpus_distiller.py -i <input_dir> -w [white_list_module] -p [pin] -t [dylib] -o [output_dir] -c <cmd>
        <input_dir>
                samples dir, "~/downloads/ttf/"
        [white_list_module]:
                option, "pdf.dylib;test;libc.dylib"
        [pin_path]
                option, pin bin path
                default: /Volumes/Macintosh/Users/riusksk/Reverse-Engineering/pin-3.7-clang-mac/pin
        [dylib]
                option, pin_coverage.dylib path
                default: /Volumes/Macintosh/Users/riusksk/bughunt_tools/riufuzz/tools/pin_coverage.dylib
        [output_dir]
                option, use save generated logs file , dafault: ./logs/
        <cmd>
                "font_fuzzer -i @@ -o test.pdf"

[Example]: python corpus_distiller.py -i ./pdf -c "../PDFdemo @@"
```

基于代码覆盖率的样本筛选

```
######################################################################
#基于BBCount的筛选方案
def select_by_bbcont(sample_list):
    global mini_samples_by_bbcount
    global total_size_by_bbcount
    global BBAddr_Set
    global sample_count

    sample_list.sort(key=get_bbcount, reverse=True) #按bbcount降序
    for sample in sample_list:
        if not BBAddr_Set:
            BBAddr_Set = set(sample.bb_list)
            mini_samples_by_bbcount.append(sample.name)
            total_size_by_bbcount += sample.fsize
        elif BBAddr_Set - set(sample.bb_list):  #存在BBAddr差集
            BBAddr_Set = BBAddr_Set | set(sample.bb_list)   #并集
            mini_samples_by_bbcount.append(sample.name)
            total_size_by_bbcount += sample.fsize

    #print "[*] 基于BBCount的筛选结果："
    #print ''.join("\t{}\n".format(n) for n in mini_samples_by_bbcount)
    print "[*] 基于BBCount筛选前后样本个数：" + str(sample_count) + ' => ' + str(len(mini_samples_by_bbcount))
    print "[*] 基于BBCount筛选前后总样本大小：" + get_byte_unit(total_samples_size) + ' => ' + get_byte_unit(total_size_by_bbcount)
```

样本筛选效果

总体大小：16.17G => 563.8M

文件数量：10074 => 1105

运行时间：3 天 22 小时

Fuzzing效果

2016年底至今，共发现Microsoft、Apple、Google、Adobe 四大国际厂商 60+ CVE漏洞




未来计划

- 优化闭源程序的代码覆盖引导的通用性、速度与性能
- 推广与实践多工种协同Fuzzing的工作
- 建设更加完善的平台管理控制系统，方便多人协同工作


MULWAR