



看雪 2018 安全开发者峰会

Kanxue 2018 Security Developer Summit

2000-2018

看雪2018
安全开发者峰会

自动逆向机器人

弗为@阿里安全

看雪2018
安全开发者峰会



自我介绍

弗为

- 阿里巴巴安全部-猎户座攻防实验室安全专家
- 主要关注：二进制程序分析与漏洞挖掘、软件供应链安全
- 结合工程界传统人工分析经验与学术界方法论成果，多种尝试、工具规模化自动化。



向“逆向分析”致敬



第4章

逆向分析技术

将可执行程序反汇编，通过分析反汇编代码来理解其代码功能，如各接口的数据结构等，然后用高级语言重新描述这段代码，逆向分析原软件的思路。这个过程被称做“逆向工程 (Reverse Engineering)”，或者有时只是简单地称作“逆向 (Reversing)”。这是一个很重要的技能，需要扎实的编程功底和汇编知识。逆向分析的首选工具就是 IDA，其中它的一款插件 Hex-Rays Decompiler 能完成许多代码反编译的工作，逆向时可以作为一款辅助工具参考。

逆向工程可以让人们了解程序的结构以及程序的逻辑，因此利用逆向工程可以深入洞察程序的运行过程。一般所谓的“软件破解”只是逆向工程中非常初级的一部分。本节探讨的代码分析技术是基于 IA-32 处理器体系结构的。

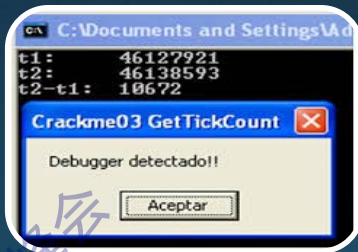
- 我们很多人从这里入门，
- 从此走上了逆向“痛并快乐”的生涯



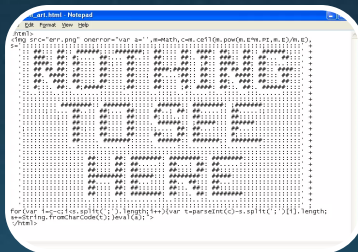
从狭义“逆向”的痛点说起



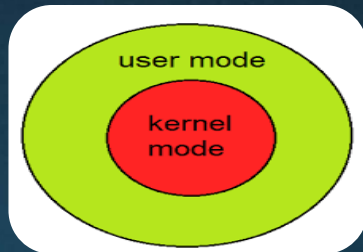
机械重复



反调试



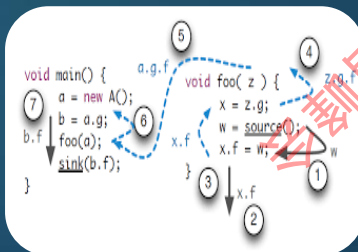
混淆



多体系/架构



不稳定重现



数据流跟踪



理想的基础工具能力



录像机



播放机



显微镜



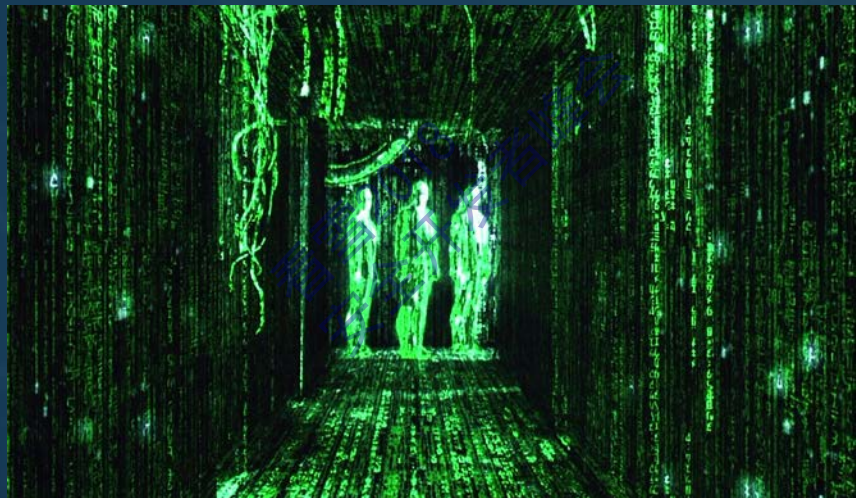
实现准确录放



- 全系统模拟
- 变量控制
 - 初始状态: CPU、内存、外设
 - 运行过程: 中断、IO输入、DMA等
- 确定性与变化
 - 对确定的重放, 可在线或离线分析
 - 可修改上下文



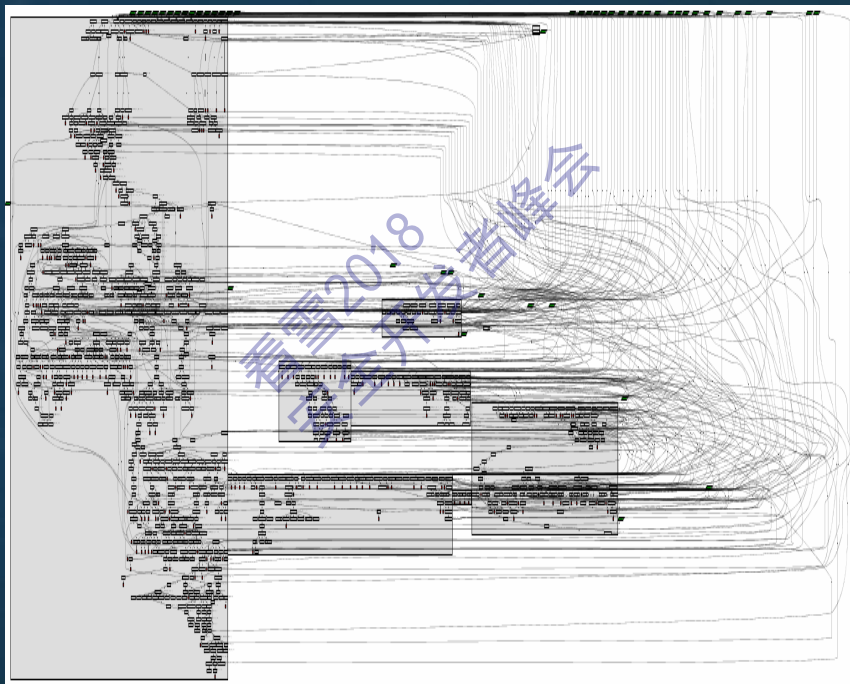
全系统与细粒度分析



- 细粒度：处理器视角
 - 每条指令，每瞬时内存上下文
- 架构与系统支持
 - 虚拟机自省：进程/线程，文件系统，网络，IPC，内存管理
- 上帝模式
 - 对抗反调试反虚拟机
 - 形成“关联搜索”



符号化污点数据流分析



- 符号执行+污点分析 扬长避短
 - 标定-跟踪-表示
 - 以符号运算表示指令语义
- 还原程序、函数的数据本质：变换



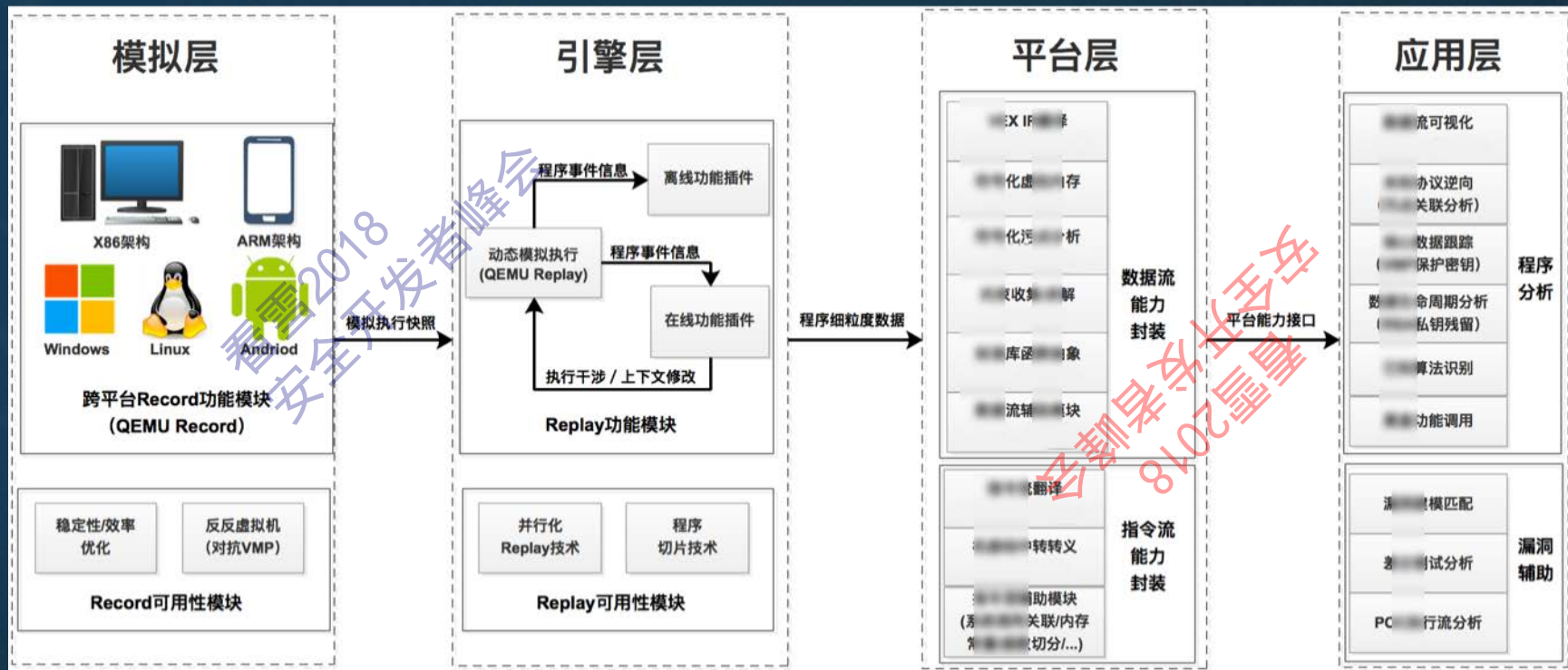
性能



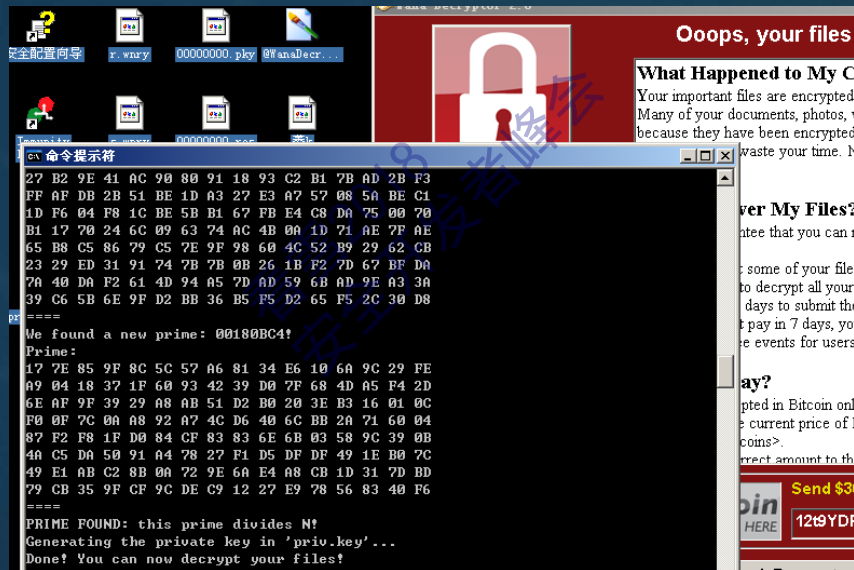
- 运算架构
 - 流水线-并行-归并
- 录制性能损失 < 30x
 - 纯CPU密集型约30x



自动逆向机器人TimePlayer



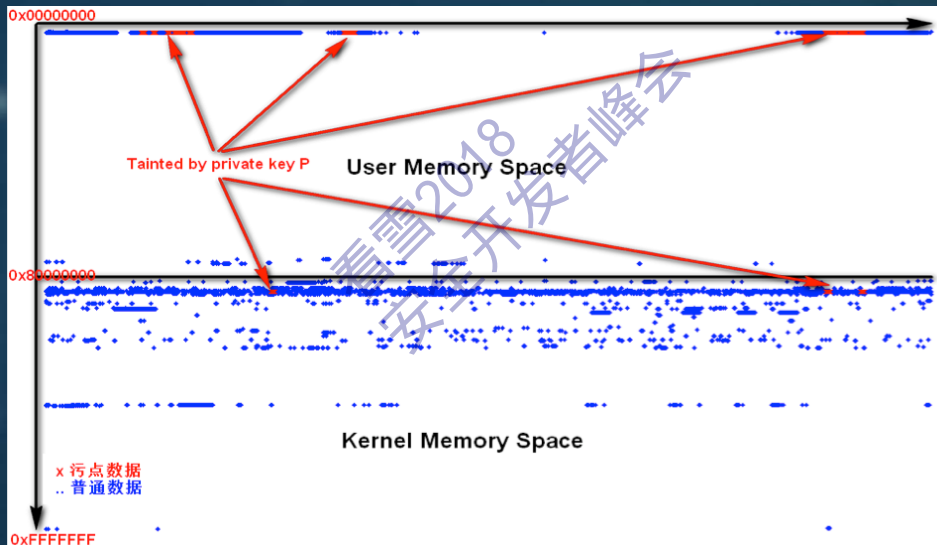
应用1: WannaCry勒索软件RSA私钥恢复



- 寻找甜点：残留私钥？
 - 未得到有目的性擦除的内存
- 如果有，有哪些地方、在什么时机？
- 密码学组件，是否可能“故意”残留了敏感数据？



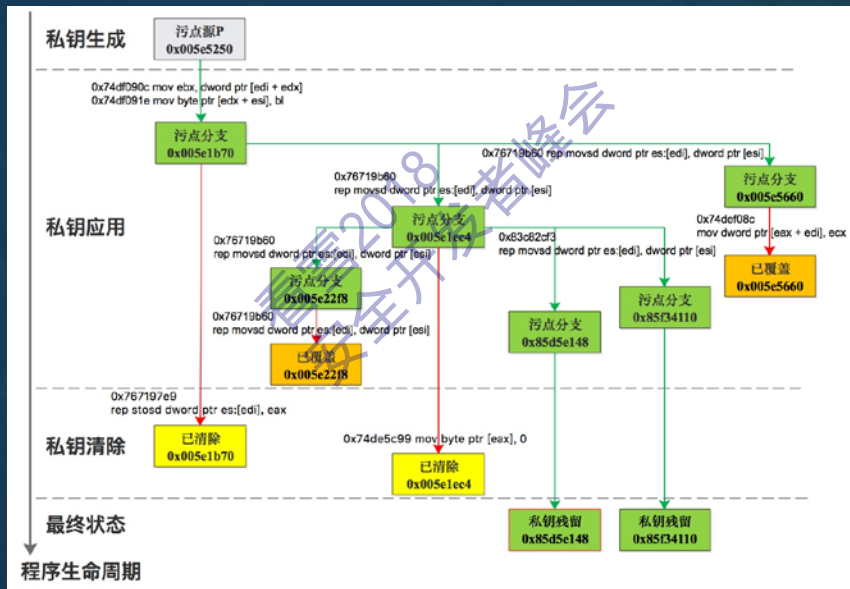
应用1: WannaCry勒索软件RSA私钥恢复



- 针对正常的加解密全过程，应用符号化污点分析，分析全内存空间的数据痕迹



应用1: WannaCry勒索软件RSA私钥恢复



私钥参数P的救赎

- 结合符号化污点分析结果和程序运行周期，整理私钥P局部传播图
- 从用户态到内核态



应用1: WannaCry勒索软件RSA私钥恢复

```
Icount 2318937773, Address 0x85f34110, Data:
c7ddf28624148121 c08550ce21ce56dc
6ba9742f501a0268 3028d523d07cb6bc
6468ab162c611bdd 6905d9602ace3215
e2240f9cd8bbfac5 d7634676b8b0e84c
13069c99547a7f2e cc84e346bfe0d692
cdcd1637ed1cab1 5c4394e4adf666d7
7755dad17cb0b793 dc90abd16ceecd1
6bd4804bea228e12 8c88312548a328fa

Icount 2318937773, Address 0x85d5e148, Data:
c7ddf28624148121 c08550ce21ce56dc
6ba9742f501a0268 3028d523d07cb6bc
6468ab162c611bdd 6905d9602ace3215
e2240f9cd8bbfac5 d7634676b8b0e84c
13069c99547a7f2e cc84e346bfe0d692
cdcd1637ed1cab1 5c4394e4adf666d7
7755dad17cb0b793 dc90abd16ceecd1
6bd4804bea228e12 8c88312548a328fa

prime1(p)(len:128):
c7 dd f2 86 24 14 81 21 c0 85 50 ce 21 ce 56 dc
6b a9 74 2f 50 1a 02 68 30 28 d5 23 d0 7c b6 bc
64 68 ab 16 2c 61 1b dd 69 05 d9 60 2a ce 32 15
e2 24 0f 9c d8 bb fa c5 d7 63 46 76 b8 b0 e8 4c
13 06 9c 99 54 7a 7f 2e cc 84 e3 46 bf e0 d6 92
cd cd 16 37 ed 1c ab c1 5c 43 94 e4 ad f6 66 d7
77 55 da d1 7c b0 b7 93 dc 90 ab d1 6c e0 ec d1
6b d4 80 4b ea 22 8e 12 8c 88 31 25 48 a3 28 fa

prime2(q)(len:128):
ed 61 68 c3 f8 da d4 ef ea 3b 85 5e 4a d2 56 ba
d2 6d db fa 41 79 0a 17 8e fb f6 71 79 6d 84 9b
70 cc 11 de 3b a9 1f 90 0b 9f 84 43 c7 0b cb f8
21 ec 8c b3 7a 90 78 03 00 ca 3f 13 f8 b7 46 cf
```

- 在程序调用CryptDestroyKey和CryptReleaseContext之后提取地址0x85d5e148和0x85f34110的内存数据，与P的值做比较



应用2：自动化私有网络协议逆向

```
p frame 1: 5498 bytes on wire (43984 bits), 5498 bytes captured (43984 bits) on interface 0
HDNet file
# PCAP file format
# Header
Magic number: d4c3b2a1 (little-endian)
Version major: 2
Version minor: 4
This dump is: 0
Signatures: 0
Snapshot length: 262144
Link type: ETHERNET (1)
# Packet 1
Timestamp: Nov 28, 1999 10:12:35.391343000 中国标准时间
Included length: 93
Origin length: 93
# Data
# Ethernet II, Src: Lite-Onu_3b:b9:fa:00:00:00, Dst: 192.168.0.2
# Destination: 192.168.0.2 (08:00:0c:3b:b9:fa:00:00)
# Source: Lite-Onu_3b:b9:fa:00:00:00 (08:00:0c:3b:b9:fa:00:00)
Type: IPv4 (0x0800)
# Internet Protocol Version 4, Src: 192.168.0.3, Dst: 192.168.0.2
0100 ..... = version: 4
..... = header length: 20 bytes (5)
# Differentiated Services Field: dsw0 (DSCP: unknown)
Total length: 79
Identification: 0x443e (17982)
Flags: 0x0000, don't fragment
Time to live: 64
Protocol: TCP (6)
Header checksum: 0x9097 (validation disabled)
[header checksum status: confirmed]
Source: 192.168.0.3
Destination: 192.168.0.2
[Source (OSPF): unknown]
[Destination (OSPF): unknown]
# Transmission Control Protocol, Src Port: 1558, Dst Port: 80
Sequence number: 1 (relative sequence number)
[Next sequence number: 28 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
1800 ..... = header length: 32 bytes (8)
# Flags: 0x0000 (RST, ACK)
Window size value: 32128
[Calculated window size: 32128]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x8667 [unverified]
[Checksum status: Unverified]
Urgent pointer: 0
# Options: (12 bytes), No-Operation (NOP), No-Operation (NOP)
# [SQACK analysis]
TCP payload (27 bytes)
# Telnet
# DO Suppress Go Ahead
Command: DO (253)
Subcommand: Suppress Go Ahead
# Will Terminal type
# Will negotiate about window size
# Will Terminal Speed
# Will X-Window Protocol
```

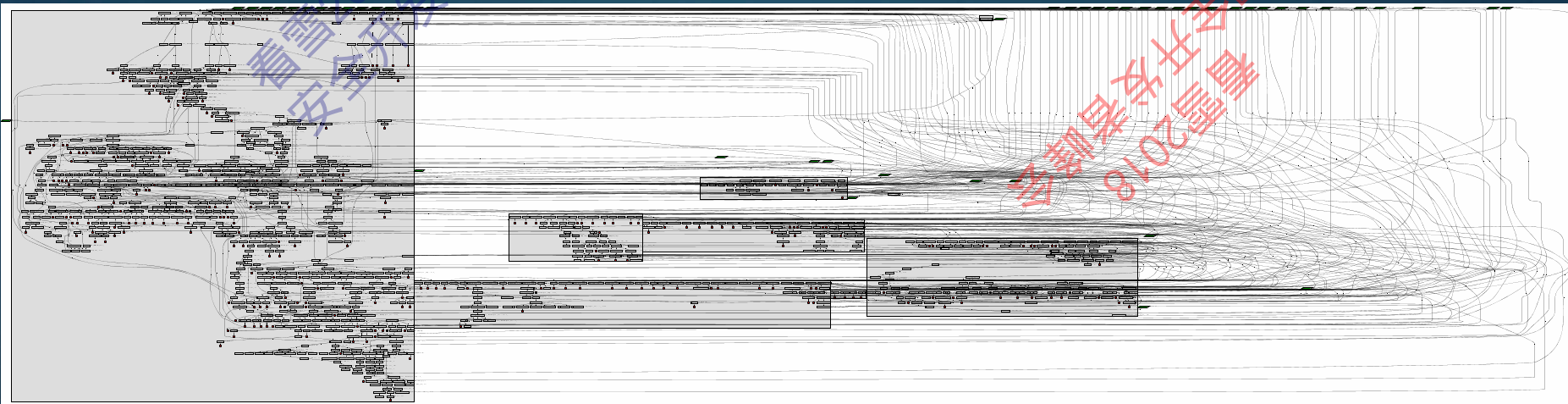
- 拿得到程序与数据，却对数据格式一头雾水？
— 以往依据不断的调试、猜测从中获取有用信息
- 掌握了程序对数据的处理过程，就能归纳推测出程序处理过程的“语义”，进而对输入数据的“语法”做推断。



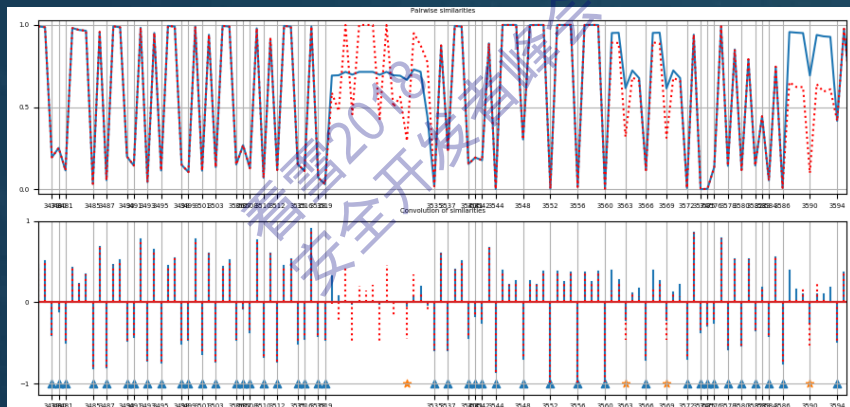
应用2：自动化私有网络协议逆向



- 字节粒度/带符号信息的准确描述，保证尽量消除了过传播与欠传播，并保证全量覆盖。
- 对每个字节做污点数据流图，可获得主观的结构认知和相似印象。



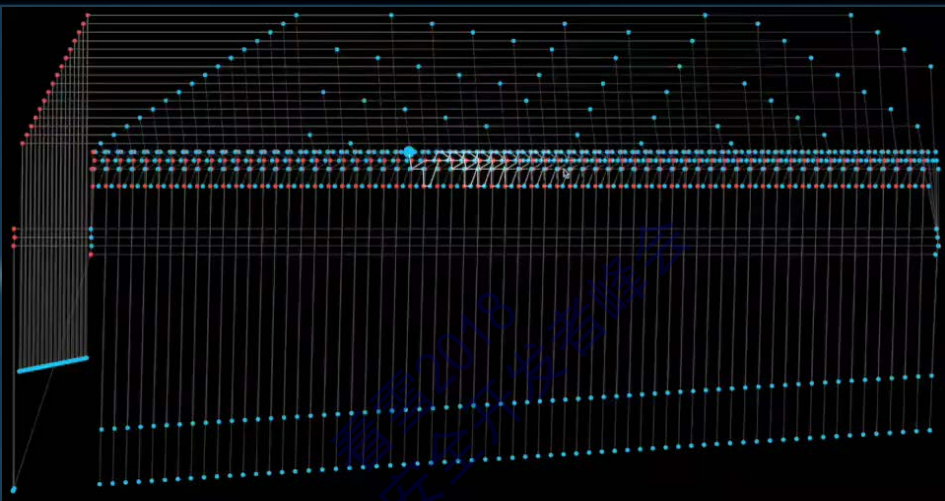
应用2：自动化私有网络协议逆向



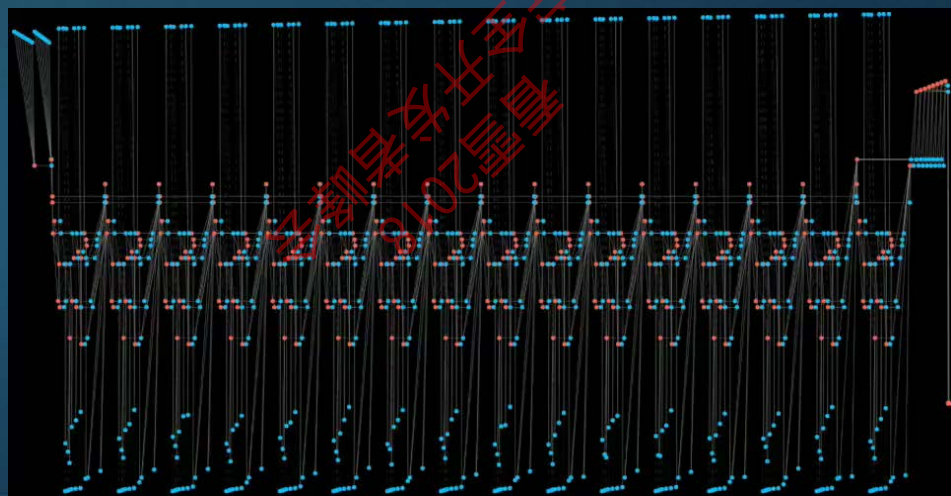
- 字段边界判定
- 字段聚类
 - 考察时间代码相关性，暴露字段相似度，聚类得到上层协议结构；
- 典型模式学习匹配，获取特殊属性字段推断
 - 加密、压缩、序列化数据块
 - 校验和、哈希字段
 - 长度、偏移量、间隔符、终止符字段等



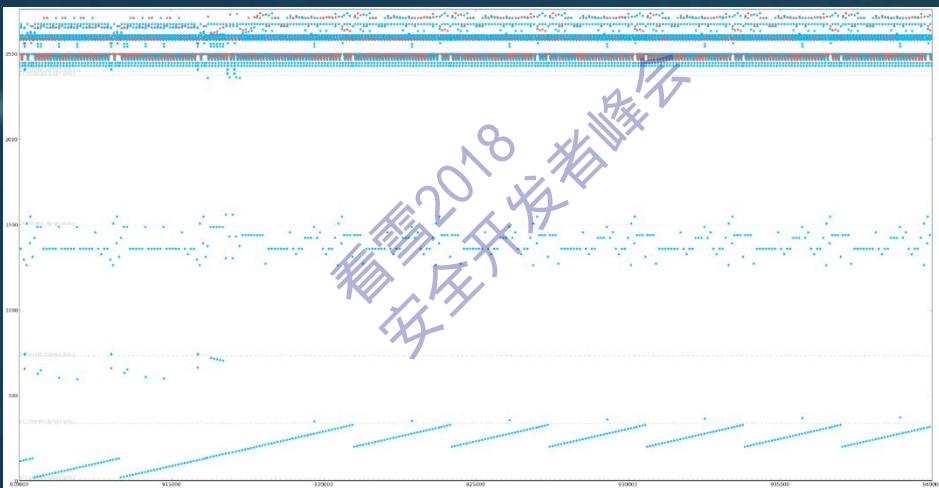
应用3：虚拟机壳等混淆下的算法识别



- 任何函数、过程、算法，从数据维度总有特定的输入输出模式



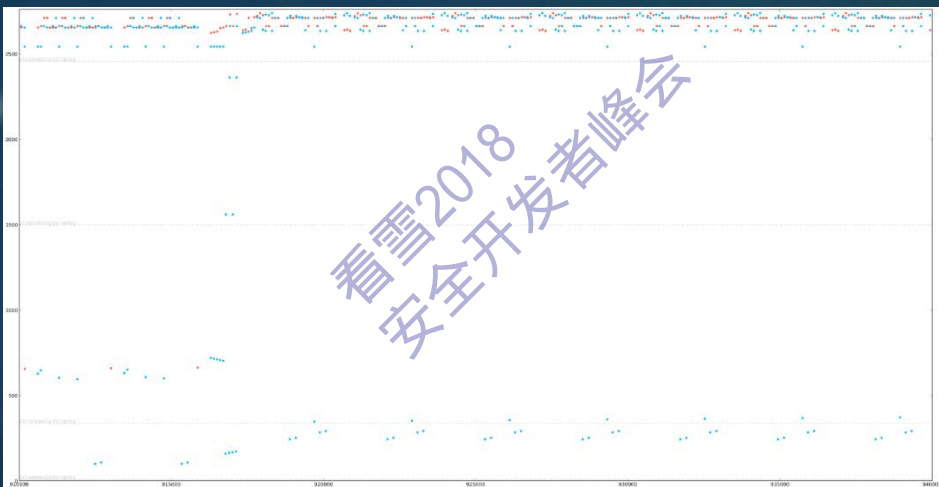
应用3：虚拟机壳等混淆下的算法识别



- 通过逆向污染分析，筛选出算法相关的指令，排除掉混淆部分的无关指令，指令数量大幅缩减，再通过数据流可视化，勾勒出数据计算之间的关系，方便进行抽丝剥茧的分析。



应用3：虚拟机壳等混淆下的算法识别



- 通过逆向污染分析，筛选出算法相关的指令，排除掉混淆部分的无关指令，指令数量大幅缩减，再通过数据流可视化，勾勒出数据计算之间的关系，方便进行抽丝剥茧的分析。



应用3：虚拟机壳等混淆下的算法识别



- 通过逆向污染分析，筛选出算法相关的指令，排除掉混淆部分的无关指令，指令数量大幅缩减，再通过数据流可视化，勾勒出数据计算之间的关系，方便进行抽丝剥茧的分析。



总结&思考

- 重新定义“逆向工程” 😞
 - 逆向的不只是代码碎片，更是全局、逻辑、功能
 - 为工具开发者打call，自动化、规模化是大势所趋
- 二进制的黑盒性质，代码混淆的障眼法，总有失效的一天
 - 人力+经验vs固定模式的机器语言混淆，在两方面都将改变
 - 将出现的密码学强度可证明的混淆模型
- 业界圈子需要吸纳学术成果，理论创新需要工程实践改造

