



<https://shorturl.at/bPCzW>

<https://retooling.io/blog>

欢迎来到：
一场意料之外的旅途，深入
Microsoft Defender 的签名世界

你需要:

1. ResourceHacker (~3MB)
2. 基于 msys64 构建的环境 (~350MB)
3. 测试环境

<https://shorturl.at/bPCzW>

我们是谁.... "一群意大利博士" 以及 ...

Silvio



Antonio

→ Co-founder of Retooling LLC → Co-founder of Retooling LLC → Master's degree student in Cybersecurity
@ University of Sapienza

→ Former Senior Cyber Security Architect @ → Former Senior Cyber Security

LEONARDO Spa - Cyber Security Architect @ LEONARDO Spa - Cyber → Bachelor's degree in Information Division
Division Technology @ University of L'Aquila

@DrCh40s → Senior Security Researcher @ EMC/RSA -> → Cyber Threat Analyst / Reverse → Passionate about
malware analysis and
DELL – Center of Excellence

Engineer

→ Malware reverse engineer @ Symantec -
→ PhD System Security @ University of
Security Response

Roma Tre

→ PhD Network Security @ University of
Pisa
→ M.Sc. in Computer Engineering

→ M.Sc. in Computer Science

Davide



Security

@t0nvi



@davidefont96

antonio@retooling.io

silvio@retooling.io **davidefontana96.df@gmail.com**

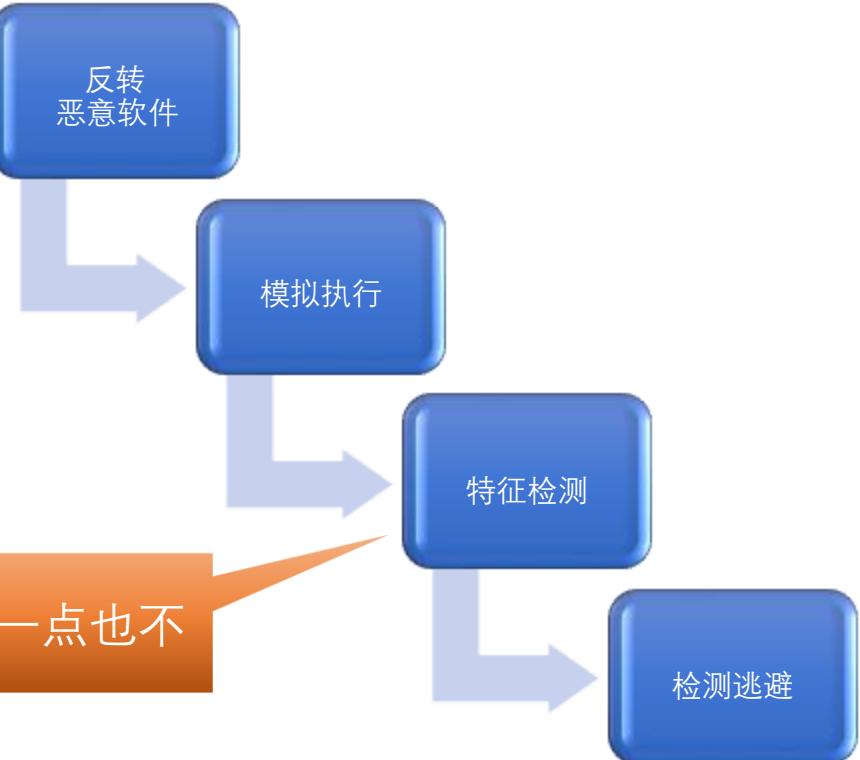
一场意外的旅途：

- 起点： PingPull.exe
- 初始目标：
PingPull.sln
- 终点： Defender.IDB

PingPull 基于 Visual C++ 构建，为攻击者提供了在被入侵主机上执行命令并执行反向 Shell 连接的能力。

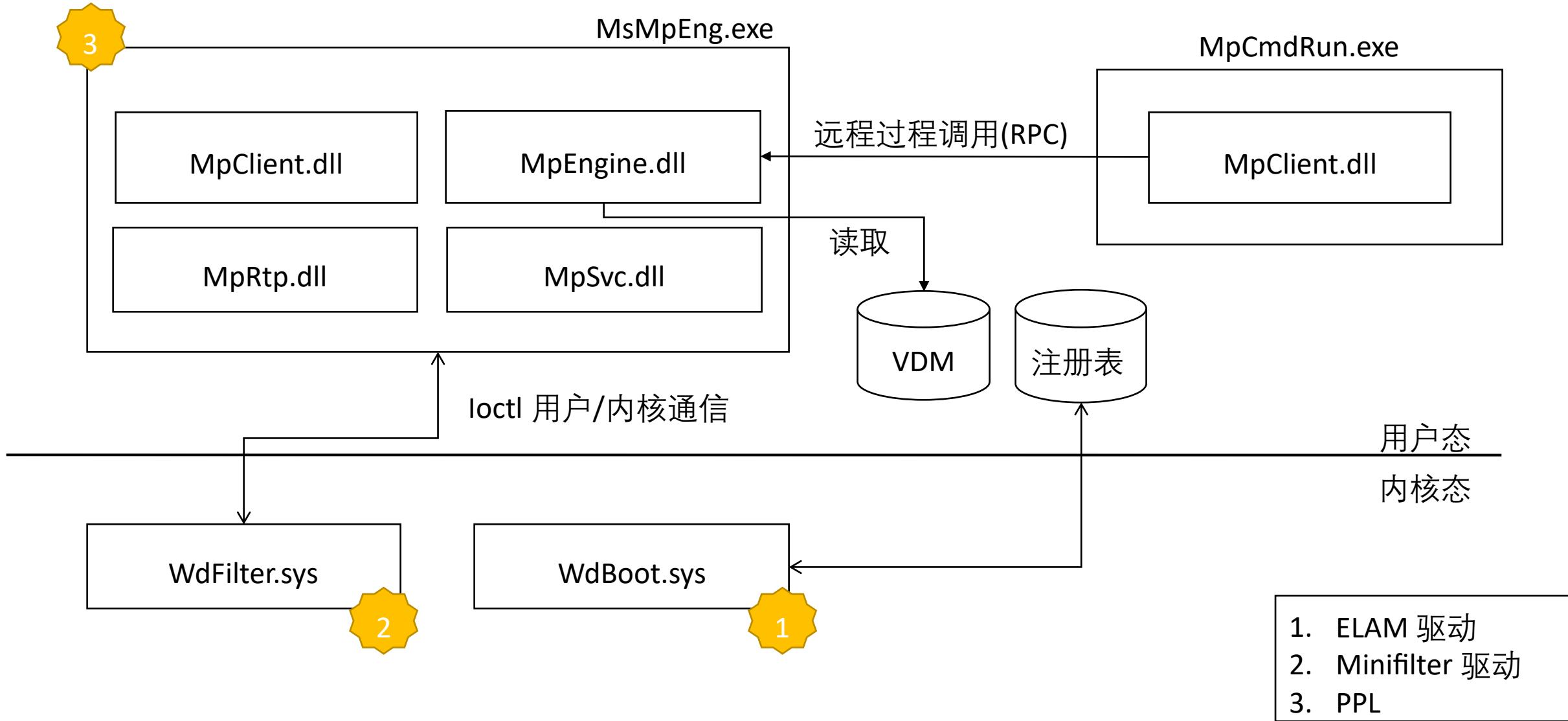
PingPull 有三种变种，它们功能相同，但使用不同协议与其 C2 通信： ICMP、HTTP(S) 和原始 TCP。

——Palo Alto, Unit42



将新威胁整合入 Retooling Revo

Microsoft Defender Antivirus 的框架



Microsoft Defender 的签名文件

- 位于: C:\ProgramData\Microsoft\Windows Defender\Definition Updates\<RandomGUID>\
- 可移植应用程序:
 - mpa{s,v}base.vdm : 每月更新, 包含反恶意软件/反间谍软件签名
 - mpa{s,v}d1ta.vdm: 不定期频繁更新, 包含对上述基础库的反恶意软件/反间谍软件更新
 - 关注 mpavbase.vdm 和 mpasbase.vdm
mpavbase.vdm 和 mpasbase.vdm
 - 均在其资源字节段 (.rsrc) 包含压缩的数据 (签名)

→ 引导时, mpengine 合并 *base.vdm 和 *delta.vdm 文件

```
LoadMoudleHeader
    pCurr = *(unsigned int *)Buffer;
    if ( *(WORD *)Buffer != 'ZM' )
        break;
    QuadPart = v3.QuadPart;
    RsrcOffset.QuadPart = FindResourceOffset(hFile, (_int64)&v17);
    v3 = RsrcOffset;
    if ( RsrcOffset.QuadPart == -1 || WIN32_NATIVE_Seek(hFile, RsrcOffset) )
        return 0xA002i64;
}
*a2 = *(_DWORD *)Buffer;
if ( (_DWORD)pCurr != 'XDMR' ) // vdm header magic
```

The screenshot shows a debugger interface with two main panes. On the left, the code for `LoadMoudleHeader` is displayed. On the right, a memory dump is shown with several resources listed in the left margin:

- RT_RCDATA (1000 : 1033)
- Version Info (1 : 1033)

The memory dump pane shows memory starting at address 00000828. Several bytes are highlighted with red boxes and labeled with orange callouts:

- 偏移量**: Points to the first byte of the memory dump (00).
- 魔法字符串**: Points to the byte sequence 30 01 00 00.
- 校验值?**: Points to the byte sequence DC E6 84 02 68 3C 48 86.
- 压缩数据**: Points to the byte sequence 6C BC 79 3C 97 51 D3 3F.

多种类型的签名

```

switch      (a1)
{
    ...
    case 0x79u:
        return "SIGNATURE_TYPE_VDLL_X86";
    case 0x6Bu:
        return "SIGNATURE_TYPE_WVT_EXCEPTION";
    case 0x6Cu:
        return "SIGNATURE_TYPE_REVOKED_CERTIFICATE";
    case 0x70u:
        return "SIGNATURE_TYPE_TRUSTED_PUBLISHER";
    case 0x71u:
        return "SIGNATURE_TYPE_ASEP_FILEPATH";
    case 0x73u:
        return "SIGNATURE_TYPE_DELTA_BLOB";
    case 0x74u:
        return "SIGNATURE_TYPE_DELTA_BLOB_RECINFO";
    case 0x75u:
        return "SIGNATURE_TYPE_ASEP_FOLDERNAME";
    case 0x77u:
        return "SIGNATURE_TYPE_PATMATCH_V2";
    case 0x78u:
        return "SIGNATURE_TYPE_PEHSTR_EXT";
    ...
}

```

		Magic bytes	Relative offset to buffer to decompress	
00000000	52 4d 44 58 59 b1 57 66 ff ff ff ff ff 02 00 20 00	RMDXY±Wfÿÿÿÿ.. .		
00000010	00 00 00 00 00 00 00 00 00 00 30 01 00 00 d6 b3 01 050 ... Ö³ ..		
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000030	ff ff ff ff 00 00 00 00 00 00 00 00 00 00 1e dd 29 00	ÿÿÿÿ.....Ý).		
00000040	11 cc 06 00 20 c2 0d 00 21 97 00 00 27 47 24 00	.Í... Â...! ... 'G\$.		
00000050	28 0d 86 00 29 42 3e 00 40 02 1e 00 41 40 06 00	(...)B>.@ ... A@..		
00000060	42 80 09 00 43 24 3f 00 44 b3 04 00 49 8d 00 00	B ... C\$?.D³ .. I ...		
00000070	50 27 57 00 55 3c e3 00 58 64 01 00 5c a2 95 04	P'W.U<ä.Xd.. \¢..		
00000080	5d a2 95 04 5f fb 01 00 60 8d 04 00 61 ad 13 00]¢.._û..` ... a ...		
00000090	63 c5 01 00 67 17 6d 14 6c aa 06 00 71 7d 00 00	cÅ..g.m.lª..q}..		
000000a0	78 8a ce 00 7a 02 52 00 7e 0b 49 01 7f b3 01 00	x.Í.z.R.~.I...³ ..		
000000b0	80 d1 27 05 87 b6 09 01 89 7b 01 00 8c 8f 05 00	.Ñ' ... ¶ ... {.....		
000000c0	8d dd 01 00 8e 01 00 00 8f b8 14 00 95 32 01 00	.Ý..... ... 2..		
000000d0	96 66 0d 00 a8 80 06 00 a9 c9 01 00 b3 4c 2b 00	.f... " ... @É..³L+.		
000000e0	b4 02 00 00 ba 22 00 00 bb b6 00 00 bc 53 00 00	' ... °" .. »¶..%S..		
000000f0	bd fa 00 00 be 10 08 00 bf cc 00 00 c5 50 02 00	žú..¾ ... ¿Ì..ÅP..		
00000100	c8 0d 00 00 c9 35 00 00 ce 72 03 00 cf 88 16 00	È ... É5..Ír..Í ...		
00000110	d0 5b 00 00 d3 35 01 00 d4 01 00 00 d6 01 00 00	Ð[..05..Ô ... Ö ...		
00000120	d7 73 03 00 e6 01 00 00 e7 c6 4c 00 ea 01 00 00	xs..æ ... çÆL.ê ...		
00000130	85 c5 e4 02 64 24 ba bf 6c bc 77 3c d7 e1 d7 3f	.Åä.d\$°;l¾w<xáx?		
00000140	7e d9 7b 66 ef bd 89 52 21 ca 2c 8a 88 8c 08 21	~Ù{fiž.R!È,....!		
00000150	a2 90 51 c8 2a 52 46 12 42 84 a2 a2 92 52 d2 30	¢.QÈ*RF.B.¢¢.RÒ0		

■ Signature type code

■ Signature counter

Compressed data buffer

实验 0: 解压 Windows Defender 的签名文件

1. 打开文件夹 `C:\ProgramData\Microsoft\Windows Defender\Definition_Updates\<Your_GUID_Here>\`
2. 复制 `mpavbase.vdm` 到你的工作目录
3. 剪切出上述的压缩数据，并保存为 `x.gz`

```
import zlib
compressed = open('x.gz', 'rb').read()
decompressed = zlib.decompress(compressed, -zlib.MAX_WBITS)
```

4. 在同目录下运行此 `python3` 脚本

无 gz 文件头

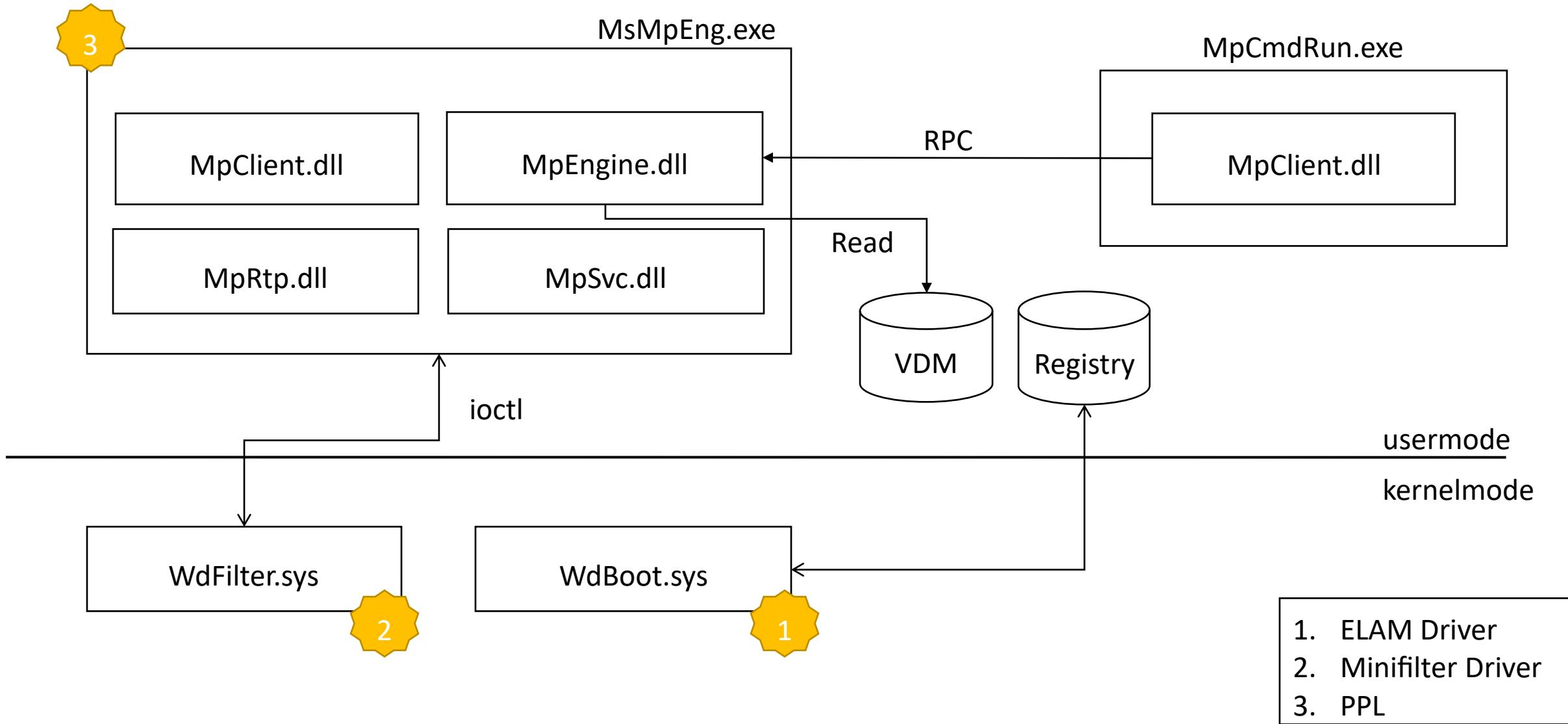
预计从解压的 vdm 文件中输出

- 包含涉及到威胁名称的 ASCII 字符串：
- !Hupigon
- !Plugx.C
- ...
- 各威胁名称间距不一

00000000	5C 1F 00 00 01 10 00 80 00 00 01 00 06 00 09 00	\.....È.....
00000010	84 21 48 75 70 69 67 6F 6E 00 00 ED 62 05 82 42	"!Hupigon..ib.,B
00000020	00 04 00 28 0E 00 00 00 65 6F 27 F2 04 8C 9B 29	...(...eo'ò.Ó»)
00000030	8D EE 00 00 00 28 0E 00 00 00 96 06 31 66 04 53	.i....(....-lf.S
00000040	A8 5A 0F EC 02 00 00 28 0E 00 00 00 96 06 31 66	"Z.i....(....-lf
00000050	04 C5 76 1B A4 E2 02 00 00 28 0E 00 00 00 42 CF	.Av.»å...(....BI
00000060	38 31 04 57 B3 EE AC E7 02 00 00 28 0E 00 00 00	81.W'i-ç....(....

00F681C0	80 5D 04 00 00 77 C0 02 80 5C 1F 00 00 78 C0 02	€]...wÀ.€\...xÀ.
00F681D0	80 00 00 01 00 08 00 09 00 AC 21 50 6C 75 67 78	€.....-!Plugx
00F681E0	2E 43 00 00 01 40 05 82 70 00 04 00 78 76 00 00	.C...@.,p...xv..
00F681F0	03 00 03 00 03 00 00 01 00 3E 03 52 6A 40 03 F0>.Rj@.ö
00F68200	6A 10 56 FF D7 85 C0 74 35 B8 90 01 04 2B C6 83	j.Výx..Àt5,...+Ef
00F68210	E8 05 88 46 01 8B C8 8B D0 C1 E8 18 C1 E9 08 88	è.^F.<È<ÐÁè.Áé.^
00F68220	46 04 C1 EA 10 8D 44 24 08 50 C6 06 E9 88 4E 02	F.Áê..D\$.PÈ.é^N.
00F68230	88 56 03 8B 4C 24 0C 90 00 01 00 17 01 8B F0 83	^V.<L\$.....<8f
00F68240	FE FF 74 3C 6A 00 8D 44 24 0C 50 68 00 00 10 00	být<j..D\$.Ph....
00F68250	57 56 FF 15 01 00 0C 01 50 45 00 00 75 54 56 8B	WVý.....PE..uTV<
00F68260	71 28 57 8B 00 00 5D 04 00 00 78 C0 02 80 5C 2A	q(W<...])...xÀ.€*
00F68270	00 00 79 C0 02 80 00 00 01 00 03 00 14 00 50 57	..yÀ.€.....PW

Microsoft Defender Antivirus 结构 (重放)



阶段 1: 签名数据库预加载

ksignal

```
260     case 0x400Bu:  
261         modprobe_init_status = modprobe_init(rsignal_code, 0x75A100000i64, (void *)a3);  
262         if ( modprobe_init_status )  
263             modprobe_cleanup(0i64);  
264         return modprobe_init_status;
```

modprobe_init_worker

```
195 StringCchPrintfW(&pGkTab->engine_version, 0x40ui64, L"%hs", "1.1.23100.2009");  
351 bDbLoaded = preload_database((wchar_t *)L"mpavdlta.vdm", (_int64)ShaCtx);  
352 if ( bDbLoaded || (bDbLoaded = preload_database((wchar_t *)L"mpavbase.vdm", (_int64)ShaCtx)) != 0 )  
387     result = preload_database((wchar_t *)L"mpasdlta.vdm", (_int64)ShaCtx);  
388     NumberOfBytesWritten = result;  
389     if ( (_DWORD)result )  
390         goto LABEL_49;  
391     result = preload_database((wchar_t *)L"mpasbase.vdm", (_int64)ShaCtx);  
392     NumberOfBytesWritten = result;  
393     if ( (_DWORD)result )  
394         goto LABEL_49;
```

读取文件头并检索一般信息，如签名版本和数字

LoadModuleHeader : 加载数据库头前 16 字节

签名版本 1.401.1166.0

一旦对签名文件的预处理完成，Defender 模块初始化启动……

阶段 2：初始化 Defender 各模块

```
g_pUnimodEntries unimod_entry_t <offset aPrivilegeutils, \
    ; DATA XREF: init_modules(void)+70↑o
    offset ?PrivilegeUtils_init_module@@YA?AW4MP_ERROR@@PEAVAutoInitModules@@@Z, \
    offset ?PrivilegeUtils_cleanup_module@@YAXXZ, 1>
unimod_entry_t <offset aDbvars, \
    ; dbvars_cleanup_module(void) ...
    offset ?dbvars_init_module@@YA?AW4MP_ERROR@@PEAVAutoInitModules@@@Z, \
    offset ?dbvars_cleanup_module@@YAXXZ, 1>
dq offset aDbload ; "dbload"
dq offset ?DbloadInitModule@@YA?AW4MP_ERROR@@PEAVAutoInitModules@@@Z ; DbloadInitModule(AutoInitModules *)
dq offset ?DbloadCleanupModule@@YAXXZ ; DbloadCleanupModule(void)
dq 1
```

3

```
1 init_modules
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
```

循环所有在
g_pUnimodEntries 中的模块
并调用模块专用的初始化函数
pfnInit()

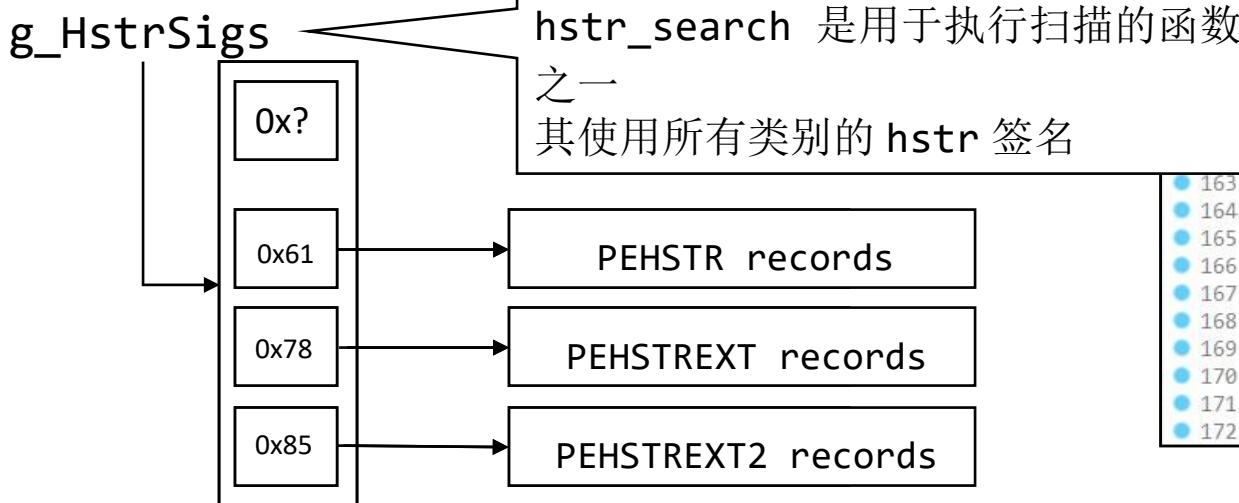
```
68     while ( 1 )
69     {
70         v8 = *((_QWORD *)pCurrAutoinitModule + 5);
71         if ( v8 >= v7 )
72             break;
73         pUnimodEntry = (punimod_entry_t)(*( _QWORD *)pCurrAutoinitModule + 32 * v8);
74         v22 = (_int64 *)pUnimodEntry;
75         v10 = (HANDLE *)WPP_GLOBAL_Control;
76         if ( WPP_GLOBAL_Control != &WPP_GLOBAL_Control && (*(( _BYTE *)WPP_GLOBAL_Control + 28) & 8) != 0 )
77         {
78             WPP_SF_Ps(*(( _QWORD *)WPP_GLOBAL_Control + 2), 17, v8, v8, (_int64)pUnimodEntry->pModuleName);
79             v10 = (HANDLE *)WPP_GLOBAL_Control;
80         }
81         if ( !g_InsideSandbox || LOBYTE(pUnimodEntry->Unk) )
82         {
83             v11 = (( _int64 (__fastcall *)(AutoInitModules *))pUnimodEntry->pfnInit)(pCurrAutoinitModule);
84             v14 = v11,
```

2

cksig_init_module

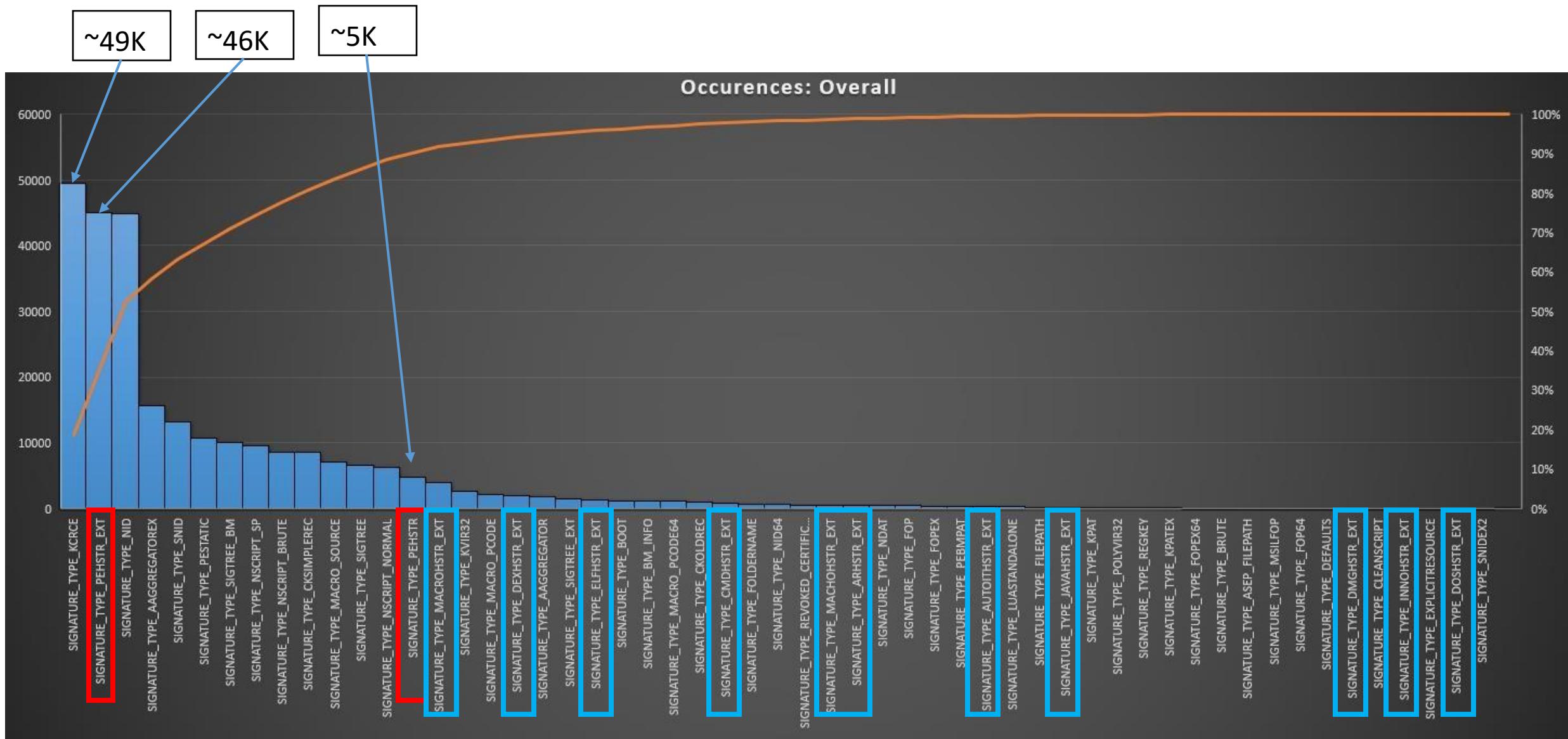
pattsearch_init

- 调用 pattsearch_init 初始化包含签名的数据结构，包括 g_HstrSigs 和 g_DynamicHstrSigs
- 这些符号是指向一个哈希表的指针，包含所有文件类型（原文 HSTR）的指针（例如 ELF PE MACOS 等）
- load_database/load_database_cache 会 DispatchRecords（分配上述记录）到正确的存储栈

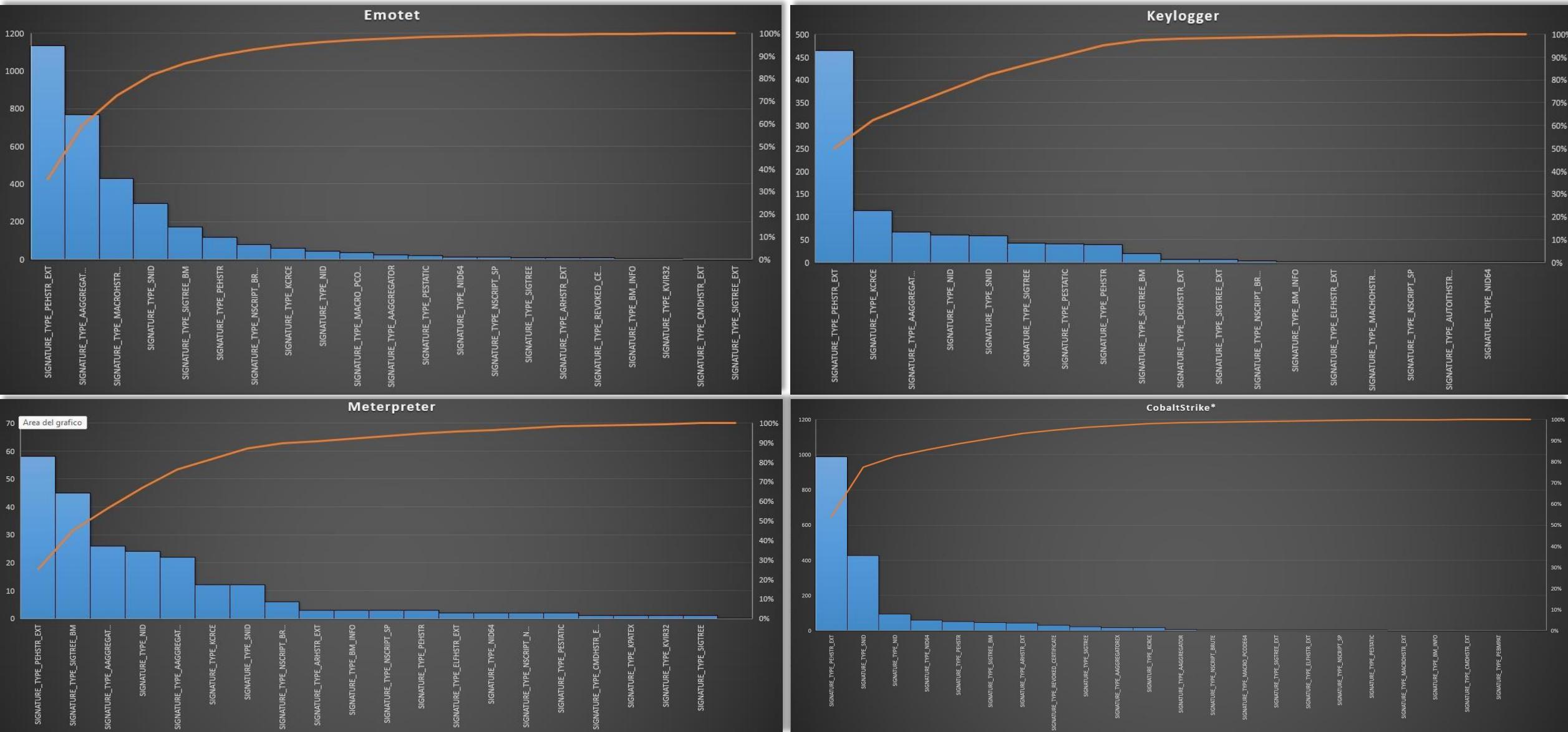


```
137     pehstr_record_cnt = ESTIMATED_RECORDS(0x61);
138     pehstr_ext_record_cnt = ESTIMATED_RECORDS(0x78);
139     pehstr_ext2_record_cnt = ESTIMATED_RECORDS(0x85);
140
141     if ( penstr_ext_record_cnt + penstr_record_cnt < pehstr_record_cnt
142         || (pe_hstr_total_cnt = pehstr_record_cnt + pehstr_ext_record_cnt + pehstr_ext2_record_cnt,
143              (unsigned int)pe_hstr_total_cnt < pehstr_ext_record_cnt + pehstr_record_cnt) )
144     {
145         v0 = 32780;
146         goto out_1;
147     }
148     g_pe_hstr_total_cnt = pehstr_record_cnt + pehstr_ext_record_cnt + pehstr_ext2_record_cnt;
149     g_p_pehstr_total = (_int64)malloc(pe_hstr_total_cnt, 0x14ui64);
150     if ( !g_p_pehstr_total )
151         goto out;
152
153     byte_5b1AF70 = 0;
154     curr_handler.pfn_push = (UINT64)hstr_push;
155     curr_handler.hstr_type = 0x61;
156     curr_handler.pfn_pushend = (_int64 (_fastcall *())hstr_pushend_common;
157     p_gHstrSigs = (char *)&g_HstrSigs;
158     v0 = regctl(&curr_handler, 0x30ui64, 0xC);
159
160     if ( v0 )
161         goto out_1;
162
163     curr_handler.pfn_push = (UINT64)hstr_push_ext;
164     curr_handler.hstr_type = 0x78;
165     curr_handler.pfn_pushend = (_int64 (_fastcall *())hstr_pushend_common;
166     p_gHstrSigs = (char *)&g_HstrSigs;
167     v0 = regctl(&curr_handler, 0x30ui64, 0xC);
168
169     if ( v0 )
170         goto out_1;
171
172     curr_handler.pfn_push = (UINT64)hstr_push_ext2;
173     curr_handler.hstr_type = 0x85;
174     curr_handler.pfn_pushend = (_int64 (_fastcall *())hstr_pushend_common;
175     p_gHstrSigs = (char *)&g_HstrSigs;
176     v0 = regctl(&curr_handler, 0x30ui64, 0xC);
177
178     if ( v0 )
179         goto out_1;
```

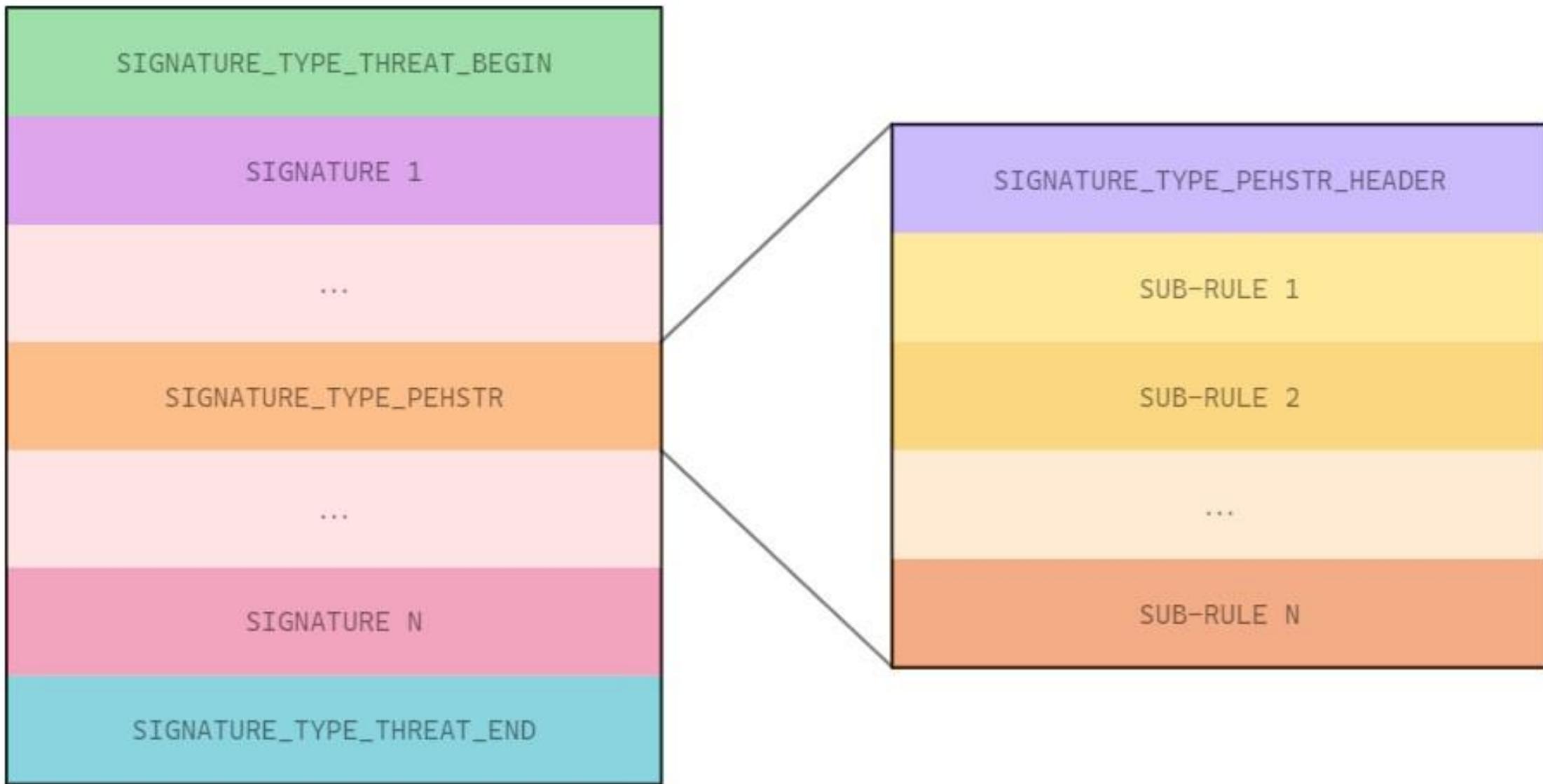
数字和统计：出现次数



对特定威胁进行切分



签名通用结构



开始到结束

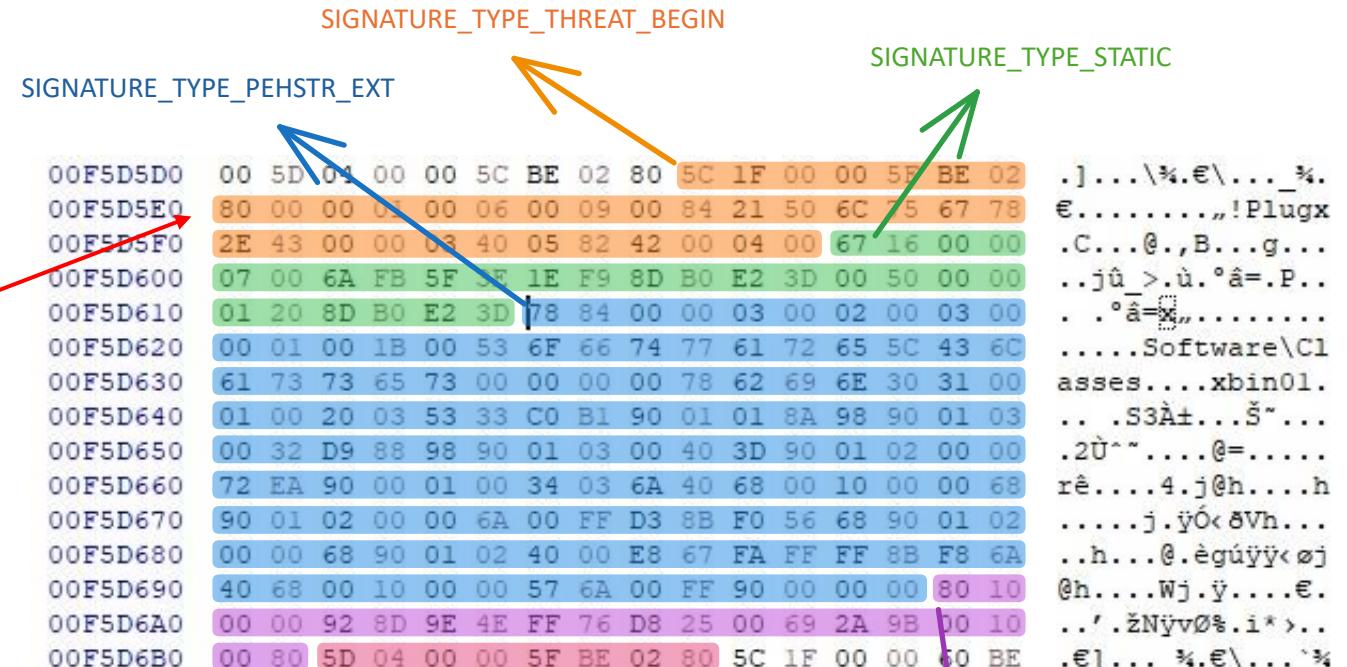
SIGNATURE_TYPE_THREAT_BEGIN 和

SIGNATURE_TYPE_THREAT_END

包含自定义数据

→ 其中之一是四字节的规则 ID (示例: 0x8002be5f)

```
createrecid
81     if ( ThreadId >= 0x80000000 )
82         v12 = AV_AutoGenThreatID++;
83     else
84         v12 = AS_AutoGenThreatID++;
```



```
typedef struct _STRUCT_COMMON_SIGNATURE_TYPE {
    UINT8 ui8SignatureType; // 定义签名类型
    UINT8 ui8SizeLow; // 签名低位字节大小
    UINT16 ui16SizeHigh; // 签名高位字节大小
    BYTE pbRuleContent[]; // 规则内容
};
```

SIGNATURE_TYPE_THREAT_BEGIN

→ 用相对检测名称定义威胁的起点，示例: !Plugx.C

→ 代码标识符: 0x5C

→ 内部包含不同的签名，用于标记威胁

00F5D5D0	00 5D 04 00 00 5C BE 02 80 5C 1F 00 00 5F BE 02	...]...%\.\...\%.
00F5D5E0	80 00 00 01 00 06 00 09 00 84 21 50 6C 75 67 78	e.....!Plugx
00F5D5F0	2E 43 00 00 03 40 05 82 42 00 04 00 67 16 00 00	.C....@.,B....g...
00F5D600	07 00 6A FB 5F 3E 1E F9 8D B0 E2 3D 00 50 00 00	..jù>.ù.ºâ=.P..
00F5D610	01 20 8D B0 E2 3D 78 84 00 00 03 00 02 00 03 00	..ºâ=x.....
00F5D620	00 01 00 1B 00 53 6F 66 74 77 61 72 65 5C 43 6CSoftware\Cl
00F5D630	61 73 73 65 73 00 00 00 00 78 62 69 6E 30 31 00	asses....xbin01.
00F5D640	01 00 20 03 53 33 C0 B1 90 01 01 8A 98 90 01 03	...S3À±...Š~...
00F5D650	00 32 D9 88 98 90 01 03 00 40 3D 90 01 02 00 00	.2Ù^~....@=.....
00F5D660	72 EA 90 00 01 00 34 03 6A 40 68 00 10 00 00 68	rê....4.j@h....h
00F5D670	90 01 02 00 00 6A 00 FF D3 8B F0 56 68 90 01 02j.ýÓ<8Vh...
00F5D680	00 00 68 90 01 02 40 00 E8 67 FA FF FF 8B F8 6A	..h...@.ègúÿÿ<øj
00F5D690	40 68 00 10 00 00 57 6A 00 FF 90 00 00 00 80 10	@h....Wj.ý....€.
00F5D6A0	00 00 92 8D 9E 4E FF 76 D8 25 00 69 2A 9B 00 10	...'.žNývØ%.i*..
00F5D6B0	00 80 5D 04 00 00 5F BE 02 80 5C 1F 00 00 60 BE	.€]...%\.\...\%.

```
typedef struct _STRUCT_SIG_TYPE_THREAT_BEGIN {  
    UINT8 ui8SignatureType;  
    UINT8 ui8SizeLow;  
    UINT16 ui16SizeHigh;  
    UI32SignatureId;  
    unknownBytes1[6];  
    UI8SizeThreatName;  
    unknownBytes2[2];  
    lpszThreatName[ui8SizeThreatName];  
    unknownBytes3[9];  
} STRUCT_SIG_TYPE_THREAT_BEGIN,  
*PSTRUCT_SIG_TYPE_THREAT_BEGIN;
```

SIGNATURE_TYPE_THREAT_END

→

→ 定义威胁的终点

代码标识符: 0x5D

→ pbRuleContent 值和
SIGNATURE_TYPE_THREAT_BEGIN 中使用的
相应 ui32SignatureId 值相同

```
typedef struct _STRUCT_SIG_TYPE_THREAT_END
{
    UINT8    ui8SignatureType;
    UINT8    ui8SizeLow;
    UINT16   ui16SizeHigh;
    BYTE     pbRuleContent[];
} STRUCT_SIG_TYPE_THREAT_END,
* PSTRUCT_SIG_TYPE_THREAT_END;
```

00F5D5D0	00 5D 04 00 00 5C BE 02 80 50 1F 00 00 5F BE 02 .]...\\%4.%\\..._%
00F5D5E0	80 00 00 01 00 06 00 09 00 84 21 50 6C 75 67 78 €.....!Plugx
00F5D5F0	2E 43 00 00 03 40 05 82 42 00 04 00 67 16 00 00 .C...@.,B...g...
00F5D600	07 00 6A FB 5F 3E 1E F9 8D B0 E2 3D 00 50 00 00 ..jü_>.ù.ºâ=.P..
00F5D610	01 20 8D B0 E2 3D 78 84 00 00 03 00 02 00 03 00 ..°â=x.....
00F5D620	00 01 00 1B 00 53 6F 65 74 77 61 72 65 5C 43 6CSoftware\Cl
00F5D630	61 73 73 65 73 00 00 00 00 78 62 69 6E 30 31 00 asses....xbin01.
00F5D640	01 00 20 03 53 35 C0 B1 90 01 01 8A 98 90 01 03 ... S3À±...Š~...
00F5D650	00 32 D9 88 98 90 01 03 00 40 3D 90 01 02 00 00 .2Ù^~....@=.....
00F5D660	72 EA 90 00 01 00 34 03 6A 40 68 00 10 00 00 68 rè....4.j@h....h
00F5D670	90 01 02 00 00 6A 00 FF D3 8B F0 56 68 90 01 02j.ýÓ<ðVh...
00F5D680	00 00 68 90 01 02 40 00 E8 67 FA FF FF 8B F8 6A ...h...@.ègúÿÿ<øj
00F5D690	40 68 00 10 00 00 57 6A 00 FF 90 00 00 00 80 10 @h....Wj.ý....€.
00F5D6A0	00 00 92 8D 9E 4E FF 76 08 25 00 69 2A 9B 00 10 ...'žNývØ%.i*..
00F5D6B0	00 80 5D 04 00 00 5F BE 02 80 5C 1F 00 00 60 BE .€]..._%._%

SIGNATURE_TYPE_PEHSTR vs SIGNATURE_TYPE_PEHSTR_EXT

→ SIGNATURE_TYPE_PEHSTR

用于实行对 PE 文件的字符串匹配

```
typedef struct _STRUCT_COMMON_SIGNATURE_TYPE {  
    UINT8 ui8SignatureType;  
    UINT8 ui8SizeLow;  
    UINT16 ui16SizeHigh;  
    BYTE pbRuleContent[];  
} STRUCT_COMMON_SIGNATURE_TYPE,  
*PSTRUCT_COMMON_SIGNATURE_TYPE;
```

→ 代码标识符: **0x61**

00F5D5D0	00 5D 04 00 00 5C BE 02 80 5C 1F 00 00 5F BE 02 .]....\%.\...%.
00F5D5E0	80 00 00 01 00 06 00 09 00 84 21 50 6C 75 67 78 €.....,.,!Plugx
00F5D5F0	2E 43 00 00 03 40 05 82 42 00 04 00 67 16 00 00 .C...@.,B...g...
00F5D600	07 00 6A FB 5F 3E 1E F9 8D B0 E2 3D 00 50 00 00 ..jû_>.ù.ºâ=.P..
00F5D610	01 20 8D B0 E2 3D 78 B4 00 00 03 00 02 00 03 00 ..°â=%
00F5D620	00 01 00 1B 00 53 6F 66 74 77 61 72 65 5C 43 6CSoftware\Cl
00F5D630	61 73 73 65 73 00 00 00 00 78 62 69 6E 30 31 00 asses....xbin01.
00F5D640	01 00 20 03 53 33 C0 B1 90 01 01 8A 98 90 01 03 ...S3À±...Š~...
00F5D650	00 32 D9 88 98 90 01 03 00 40 3D 90 01 02 00 00 .2Ù~~....@=.....
00F5D660	72 EA 90 00 01 00 34 03 6A 40 68 00 10 00 00 68 rê....4.j@h....h
00F5D670	90 01 02 00 00 6A 00 FF D3 8B F0 56 68 90 01 02j.ýÓ<8Vh...
00F5D680	00 00 68 90 01 02 40 00 E8 67 FA FF FF 8B F8 6A ..h...@.ègúÿÿ<øj
00F5D690	40 68 00 10 00 00 57 6A 00 FF 90 00 00 00 80 10 @h....Wj.ý....€..
00F5D6A0	00 00 92 8D 9E 4E FF 76 D8 25 00 69 2A 9B 00 10 ..'..žNÿvØ%.i*>..
00F5D6B0	00 80 5D 04 00 00 5F BE 02 80 5C 1F 00 00 60 BE .€]....\%.\...%.

→ SIGNATURE_TYPE_PEHSTR_EXT

用于匹配 PE 文件的字节

→ 代码标识符: **0x78**

PEHSTR 和 PEHSTR_EXT 公共标头

- ui8ThresholdRequiredLow:
此阈值需要源自 Windows Defender (低位) 的检测
- ui8ThresholdRequiredHigh:
此阈值需要源自 Windows Defender (高位) 的检测
- ui8SubRulesNumberLow:
在此特定签名中找到的子规则数字, 用于识别威胁 (低位)
- ui8SubRulesNumberHigh:
在此特定签名中找到的子规则数字, 用于识别威胁 (高位)
- pbRuleData[]:
包含所有子规则, 用于字节匹配检测

00F5D5D0	00 5D 04 00 00 5C BE 02 80 5C 1F 00 00 5F BE 02	.1...\\%.%\\...%
00F5D5E0	80 00 00 01 00 06 00 09 00 84 21 50 6C 75 67 78	€.....!Plugx
00F5D5F0	2E 43 00 00 03 40 05 82 42 00 04 00 67 16 00 00	.C...@..B...g...
00F5D600	07 00 6A FB 5F 3E 1E F9 8D B0 E2 3D 00 50 00 00	..jü_>ù.ºâ=.P..
00F5D610	01 20 8D B0 E2 3D 78 84 00 00 03 00 02 00 03 00	..ºâ=x.....
00F5D620	00 01 00 1B 00 53 6F 66 74 77 61 72 65 5C 43 6CSoftware\Cl
00F5D630	61 73 73 65 73 00 00 00 00 78 62 69 6E 30 31 00	asses....xbin01.
00F5D640	01 00 20 03 53 33 C0 B1 90 01 01 8A 98 90 01 03	...S3À±...š...
00F5D650	00 32 D9 88 98 90 01 03 00 40 3D 90 01 02 00 00	.2Ù~....@=.....
00F5D660	72 EA 90 00 01 00 34 03 6A 40 68 00 10 00 00 68	rê....4.j@h....h
00F5D670	90 01 02 00 00 6A 00 FF D3 8B F0 56 68 90 01 02j.ýó<8Vh...
00F5D680	00 00 68 90 01 02 40 00 E8 67 FA FF FF 8B F8 6A	..h...@.ègúÿÿ<øj
00F5D690	40 68 00 10 00 00 57 6A 00 FF 90 00 00 00 80 10	@h....Wj.ý....é.
00F5D6A0	00 00 92 8D 9E 4E FF 76 D8 25 00 69 2A 9B 00 10	...žNývØ%.i*...%
00F5D6B0	00 80 5D 04 00 00 5F BE 02 80 5C 1F 00 00 5F BE 02	SIGNATURE_TYPE_PEHSTR_EXT

```
typedef struct _STRUCT_PEHSTR_HEADER {  
    ui16Unknown;  
    ui8ThresholdRequiredLow;  
    ui8ThresholdRequiredHigh;  
    ui8SubRulesNumberLow;  
    ui8SubRulesNumberHigh;  
    bEmpty;  
    pbRuleData[];  
} STRUCT_PEHSTR_HEADER, * PSTRUCT_PEHSTR_HEADER;
```

- 所有类型的签名共享同种结构
- 主要区别在于子规则格式略有不同
- SIGNATURE_TYPE_PEHSTR
用于检测“可读字符串”
- SIGNATURE_TYPE_PEHSTR_EXT
可用于检测操作码并拥有额外特殊功能

PEHSTR 和 PEHSTR_EXT 子规则结构

- ui8SubRuleWeightLow
表示子规则在检测过程中的权重（低位）
- ui8SubRuleWeightHigh
表示子规则在检测过程中的权重（高位）
- ui8SubRuleSize
指定用于匹配给定 PE 的字节字符串的大小
- ui8CodeUnknown 未知区域
- pbSubRuleBytesToMatch[]:
为取得检测而必须找到的字节

```
typedef struct _STRUCT_RULE_PEHSTR_EXT {
    UINT8 ui8SubRuleWeightLow;
    UINT8 ui8SubRuleWeightHigh;
    UINT8 ui8SubRuleSize;
    ui8CodeUnknown; // _EXT only
    pbSubRuleBytesToMatch[];
} STRUCT_RULE_PEHSTR_EXT,
*PSTRUCT_RULE_PEHSTR_EXT;
```

00F5D610	01	20	8D	B0	E2	3D	78	84	00	00	03	00	00	02	00	03	00	.. .°â=x.....
00F5D620	00	01	00	1B	00	53	6F	66	74	77	61	72	65	5C	43	6C	 Software\Cl
00F5D630	61	73	73	65	73	00	00	00	00	78	62	69	6E	30	31	00		asses....xbin01.
00F5D640	01	00	20	03	53	33	C0	B1	90	01	01	8A	98	90	01	03		... S3À±...Š~...
00F5D650	00	32	D9	88	98	90	01	03	00	40	3D	90	01	02	00	00		.2Ù^~....@=.....
00F5D660	72	EA	90	00	01	00	34	03	6A	40	68	00	10	00	00	68		rê....4.j@h....h
00F5D670	90	01	02	00	00	6A	00	FF	D3	8B	F0	56	68	90	01	02	j.ýÓ<ðVh...
00F5D680	00	00	68	90	01	02	40	00	E8	67	FA	FF	FF	8B	F8	6A		...h....@.ègúÿÿ<øj
00F5D690	40	68	00	10	00	00	57	6A	00	FF	90	00	00	00	80	10		@h....Wj.ý....€.
00F5D6A0	00	00	92	8D	9E	4E	FF	76	D8	25	00	69	2A	9B	00	10		...’žNývØ%..i*>..
00F5D6B0	00	80	5D	04	00	00	5F	BE	02	80	5C	1F	00	00	60	BE		.€]....¾.€\....¾

三个子规则示例

实验 1: SIGNATURE_TYPE_PEHSTR

- 用十六进制编辑器打开解压后的 mpavbase.vdm 并找到所有属于威胁 !Darby.A 的 SIGNATURE_TYPE_PEHSTR (0x61)
- 标记每个签名的所有区域（提示：将十六进制转存中的相关字节截图，用 Windows 画图截取相关字节并高亮显示）
- 识别子规则
- 识别阈值
- 识别每个子规则的权重



解决 SIGNATURE_TYPE_PEHSTR: 真实样本

→ 图中示例显示了一个源自威胁 !Darby.A 的 SIGNATURE_TYPE_PEHSTR

→ _STRUCT_PEHSTR_HEADER:

→ ui16Counter1: 用 青色 标记

→ ui16ThresholdRequired: 用 紫色 标记

→ ui16SubRulesNumber: 用 棕色 标记

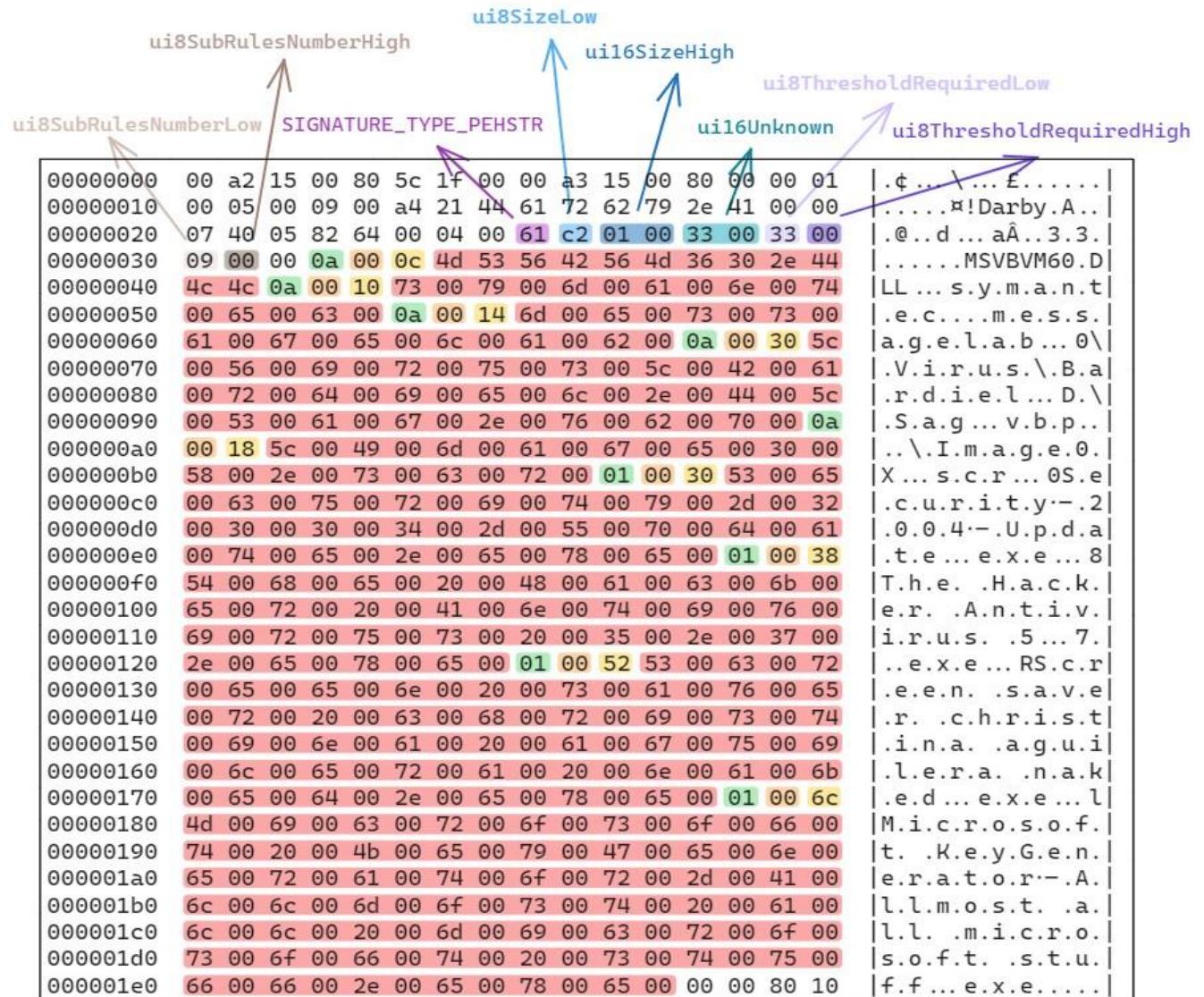
→ _STRUCT_RULE_PEHSTR:

→ ui16SubRuleWeight: 用 绿色 标记

→ ui8UnknownCode: 用 橙色 标记

→ ui8SubRuleSize: 用 黄色 标记

→ pbSubRuleBytesToMatch[]: 用 红色 标记



SIGNATURE_TYPE_PEHSTR: 匹配 !Darby.A 签名

- 此签名包含等于 0x33 的 ui16ThresholdRequired

必须达到阈值才能取得检测

- 本样本匹配下列子规则
- Sub-rule 1: 权重 0x0A.

Sub-rule 2: 权重 0x0A.

Sub-rule 3: 权重 0x0A.

Sub-rule 4: 权重 0x0A.

Sub-rule 5: 权重 0x0A.

Sub-rule 6: 权重 0x01.

总计: 0x33

	Sub-rule 1: weight 0x0A	Sub-rule 2: weight 0x0A	Sub-rule 3: weight 0x0A	Sub-rule 4: weight 0x0A	Sub-rule 5: weight 0x0A	Sub-rule 6: weight 0x01
00000000	00 00 00 00 40 00 00 40 2e 72 65 6c 6f 63 00 00				@...@.reloc..
00000010	30 00 00 00 00 50 00 00 00 02 00 00 00 00 2a 00 00					0.....`.....*
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 42				@...B
00000030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				
00000040	4d 53 56 42 56 4d 36 30 2e 44 4c 4c 00 00 00 00 00					MSVBVM60.DLL...
00000050	73 00 79 00 6d 00 61 00 6e 00 74 00 65 00 63 00					s.y.m.a.n.t.e.c.
00000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				
00000070	6d 00 65 00 73 00 73 00 61 00 67 00 65 00 6c 00					m.e.s.s.a.g.e.l.
00000080	61 00 62 00 00 00 00 00 00 00 00 00 00 00 00 00 00					a.b.....
00000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				
000000a0	5c 00 56 00 69 00 72 00 75 00 73 00 5c 00 42 00					\.V.i.r.u.s.\.B.
000000b0	61 00 72 00 64 00 69 00 65 00 6c 00 2e 00 44 00					a.r.d.i.e.l...D.
000000c0	5c 00 53 00 61 00 67 00 2e 00 76 00 62 00 70 00					\.S.a.g...v.b.p.
000000d0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				
000000e0	5c 00 49 00 6d 00 61 00 67 00 65 00 30 00 58 00					\.I.m.a.g.e.0.X.
000000f0	2e 00 73 00 63 00 72 00 00 00 00 00 00 00 00 00 00					..s.c.r.....
00000100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				
00000110	53 00 65 00 63 00 75 00 72 00 69 00 74 00 79 00					S.e.c.u.r.i.t.y.
00000120	2d 00 32 00 30 00 30 00 34 00 2d 00 55 00 70 00					-.2.0.0.4--U.p.
00000130	64 00 61 00 74 00 65 00 2e 00 65 00 78 00 65 00					d.a.t.e...e.x.e.
00000140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				

```
PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -File <filepath> -  
DisableRemediation -Trace -Level 0x10
```

```
PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -File  
'C:\Users\user\test-DarbyA.exe' -DisableRemediation -Trace -Level 0x10  
Scan starting ...  
Scan finished.  
Scanning C:\Users\user\test-DarbyA.exe found 1 threats.
```

```
<=====LIST OF DETECTED THREATS=====>  
----- Threat information -----  
Threat : Worm:Win32/Darby.A  
Resources : 1 total  
file : C:\Users\user\test-DarbyA.exe
```

Sub-rule 4:
weight 0xA

: 03 00 00@..@.reloc..
✓ 2a 00 00	0.....*..
) 00 00 42@..B
) 00 00 00
) 00 00 00	MSVBVM60.DLL....
✓ 00 63 00	s.y.m.a.n.t.e.c.
) 00 00 00
✓ 00 6c 00	m.e.s.s.a.g.e.l.
) 00 00 00	a.b.....
) 00 00 00
✓ 00 42 00	\.V.i.r.u.s.\.B.
✓ 00 44 00	a.r.d.i.e.l...D.
000000c0 5c 00 53 00 61 00 67 00 2e 00 76 00 62 00 70 00	\.S.a.g...v.b.p.
000000d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000e0 5c 00 49 00 6d 00 61 00 67 00 65 00 30 00 58 00	\.I.m.a.g.e.0.X.
000000f0 2e 00 73 00 63 00 72 00 00 00 00 00 00 00 00 00	..s.c.r.....
00000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000110 53 00 65 00 63 00 75 00 72 00 69 00 74 00 79 00	S.e.c.u.r.i.t.y.
00000120 2d 00 32 00 30 00 30 00 34 00 2d 00 55 00 70 00	-.2.0.4-.U.p.
00000130 64 00 61 00 74 00 65 00 2e 00 65 00 78 00 65 00	d.a.t.e...e.x.e.
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Expected detection

Sub-rule 5:
weight 0xA

Sub-rule 6:
weight 0x01

使用 Windows Defender 提供的 MpCmdRun.exe 扫描你的文件

实验 2: 移除 Darby 的签名检测

- 添加一个文件夹到 Defender 的排除项

```
PS> Add-MpPreference -ExclusionPath 'C:\YOUR_PATH_HERE'
```

- 将包含 Darby 的压缩文件复制到排除的文件夹并解压 (密码: infected)

- 用十六进制编辑器打开二进制文件

- 找到哪些字节命中了签名并修改，用以规避检测

至少需要修改多少字节才能绕过检测？

- 当同一子规则出现多次，对总权重有何影响？

假设字符串 S_1 (拥有权重 w_1) 在二进制中出现了 2 次，这个二进制是否会获得 $2 \times w_1$ 的权重？



为 EXT 加权

- SIGNATURE_TYPE_PEHSTR_EXT 中的子规则里存在多种形式可被用于检测操作码和其他类型

→ 用于匹配特定的字节序列

- 已确定的通配符：

90 01 XX

90 02 XX

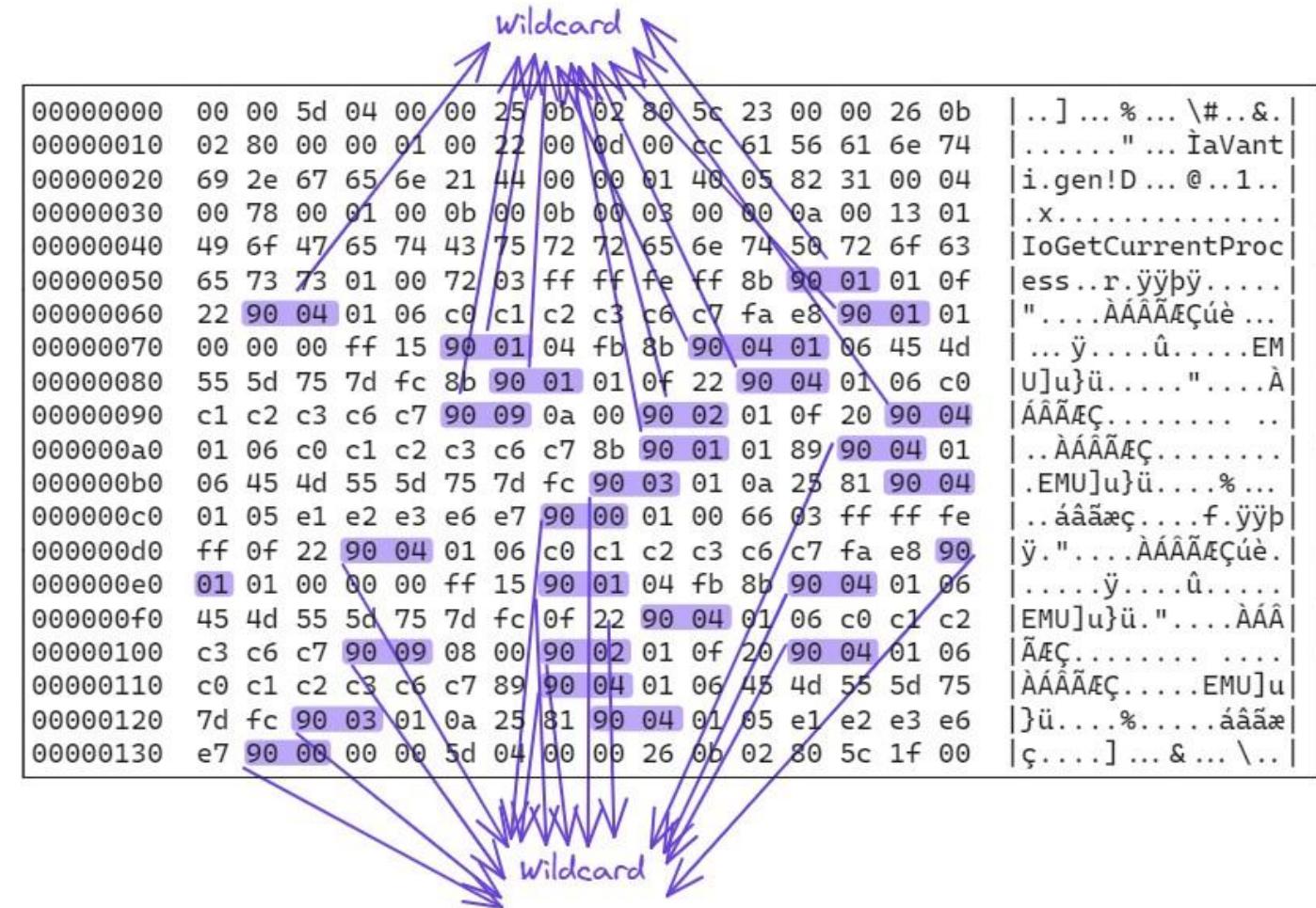
90 03 XX YY

90 04 XX YY

90 05 XX YY

- 未确认的通配符：

90 06 -> 90 20

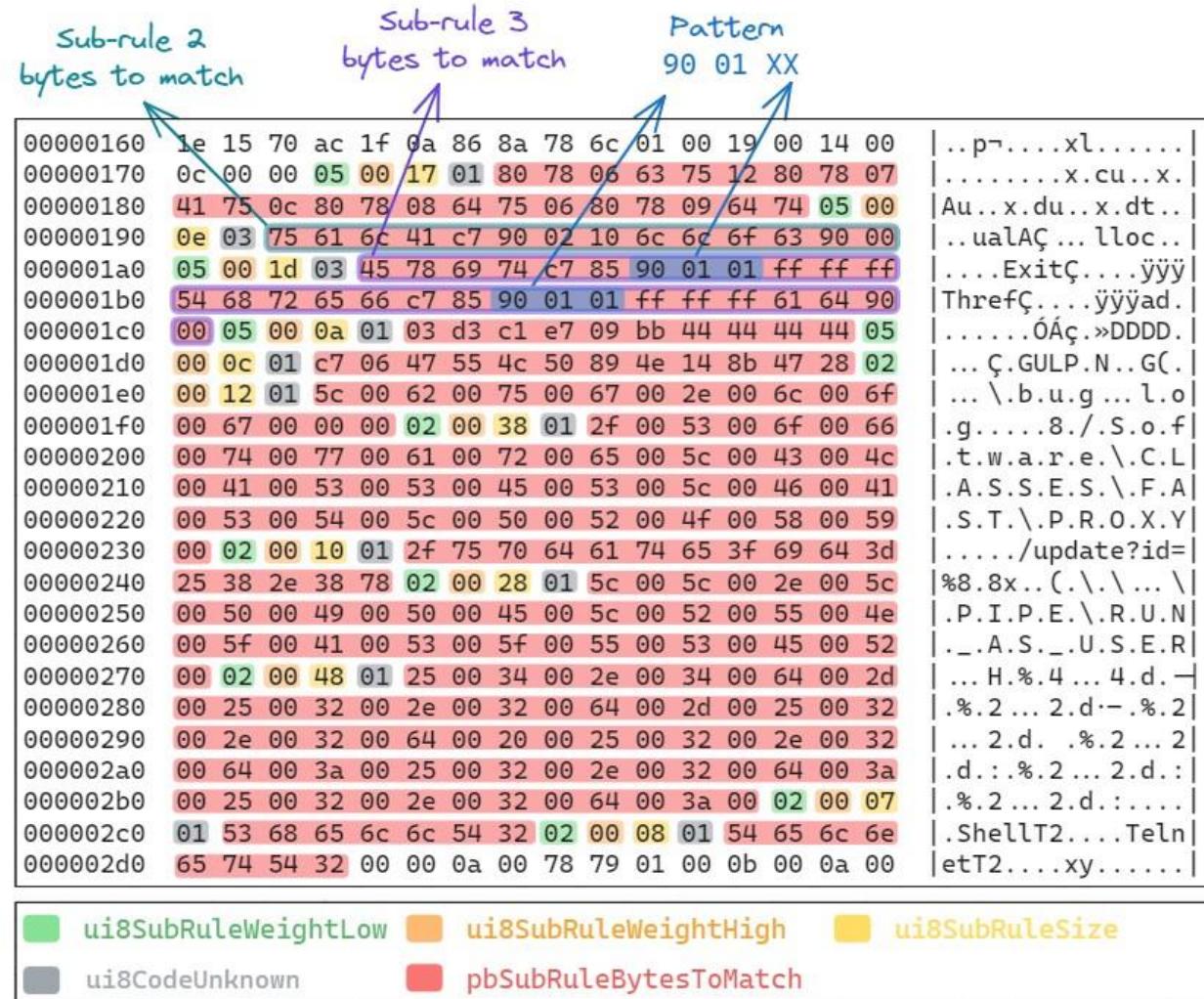


模块 90 01 XX

模式 90 01 XX:

- ✈ 用于 SIGNATURE_TYPE_PEHSTR_EXT 的子规则
- ✈ 匹配一个顺序特定，长度由 XX 定义的字节段，
顺序必定紧接 XX 后出现
- ✈ 示例用蓝色标记

```
PlugxA-Sub-Rule3-Example{
    strings:
        $sub_rule_3_hex = {
            45 78 69 74 C7 85 ?? FF FF FF 54 68
            72 65 66 C7 85 ?? 04 FF FF FF 61 64
        }
    condition:
        $sub_rule_3_hex }
```



90 01 XX

模块 90 01 XX 检测：

- 使用 MpCmdRun.exe
- 取代 90 01 01 模块的是
(用蓝色标记):

0x00

0x04

在红色子规则 2 中

在绿色子规则 3 中

- 预计检测为: Plugx.A

		Sub-rule 2	Sub-rule 3
00000000	04 00 8d 8d 3c ff ff ff 51 56 c7 85 3c ff ff ff	<ÿÿÿQVÇ.<ÿÿÿ
00000010	56 69 72 74 b3 6c c7 85 40 ff ff ff 75 61 6c 41		Virt'lç.@ÿÿÿualA
00000020	c7 85 44 ff ff ff 6c 6c 6f 63 c6 85 48 ff ff ff		Ç.DÿÿÿlllocÆ.Hÿÿ
00000030	00 ff d7 89 45 f8 85 c0 75 0e 5f 5b b8 04 00 00		.ÿx.Eø.Àu._[,...
00000040	00 5e 8b e5 5d c2 04 00 8d 95 1c ff ff ff 52 56		.^.å]Â....ÿÿÿRV
00000050	c7 85 1c ff ff ff 56 69 72 74 66 c7 85 20 ff ff		Ç..ÿÿÿVirtfÇ.ÿÿ
00000060	ff 75 61 88 9d 22 ff ff ff c7 85 23 ff ff ff 46		ÿua.."ÿÿÿÇ.#ÿÿÿF
00000070	72 65 65 c6 85 27 ff ff ff 00 ff d7 89 45 a8 85		reeÆ.'ÿÿÿ.ÿx.E".
00000080	c0 75 0e 5f 5b b8 05 00 00 00 5e 8b e5 5d c2 04		Àu._[,...^.å]Â.
00000090	00 8d 85 fc fe ff ff 50 56 c7 85 fc fe ff ff 45		...üþÿÿPVÇ.üþÿÿE
000000a0	78 69 74 c7 85 00 ff ff ff 54 68 72 65 66 c7 85		xitÇ..ÿÿÿThrefç.
000000b0	04 ff ff ff 61 64 c6 85 06 ff ff ff 00 ff d7 85		.ÿÿÿadÆ..ÿÿÿ.ÿx.
000000c0	c0 75 0e 5f 5b b8 06 00 00 5e 8b e5 5d c2 04		Àu._[,...^.å]Â.

Replaced bytes for pattern

```
PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -File 'C:\Users\user\deeac56026f3804968348c8afa5b7aba10900aeabee05751c0fcac2b88cff71e' -DisableRemediation -Trace -Level 0x10
Scan starting ...
Scan finished.
Scanning C:\Users\user\deeac56026f3804968348c8afa5b7aba10900aeabee05751c0fcac2b88cff71e found 1 threats.
```

```
<=====LIST OF DETECTED THREATS=====>
----- Threat information -----
Threat : Backdoor:Win32/Plugx.A
Resources : 1 total
file : C:\Users\user\deeac56026f3804968348c8afa5b7aba10900aeabee05751c0fcac2b88cff71e
-----
```

Expected detection

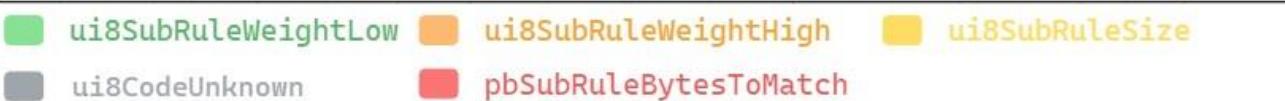
90 02 XX

模块 90 02 XX:

用作占位符，**最多**可匹配 XX 个特定字节

示例模块标记为 青色

Pattern 90 02 XX		
00000160	1e 15 70 ac 1f 0a 86 8a 78 6c 01 00 19 00 14 00	..p....xl.....
00000170	0c 00 00 05 00 17 01 80 78 06 63 75 12 80 78 07x.cu..x..
00000180	41 75 0c 80 78 08 64 75 06 80 78 09 64 74 05 00	Au..x.du..x.dt..
00000190	0e 03 75 61 6c 41 c7 90 02 10 6c 6c 6f 63 90 00	..ualAC...llloc..
000001a0	05 00 1d 03 45 78 69 74 c7 85 90 01 01 ff ff ff ffExitÇ....ÿÿÿ
000001b0	54 68 72 65 66 c7 85 90 01 01 ff ff ff 61 64 90	ThrefÇ....ÿÿad..
000001c0	00 05 00 0a 01 03 d3 c1 e7 09 bb 44 44 44 44 05ÓÁç..»DDDD..
000001d0	00 0c 01 c7 06 47 55 4c 50 89 4e 14 8b 47 28 02	...Ç.GULP.N..G(..
000001e0	00 12 01 5c 00 62 00 75 00 67 00 2e 00 6c 00 6f	...\b.u.g...l.o
000001f0	00 67 00 00 00 02 00 38 01 2f 00 53 00 6f 00 66	.g.....8./S.o.f
00000200	00 74 00 77 00 61 00 72 00 65 00 5c 00 43 00 4c	.t.w.a.r.e.\.C.L
00000210	00 41 00 53 00 53 00 45 00 53 00 5c 00 46 00 41	.A.S.S.E.S.\.F.A
00000220	00 53 00 54 00 5c 00 50 00 52 00 4f 00 58 00 59	.S.T.\.P.R.O.X.Y
00000230	00 02 00 10 01 2f 75 70 64 61 74 65 3f 69 64 3d/update?id=
00000240	25 38 2e 38 78 02 00 28 01 5c 00 5c 00 2e 00 5c	%8.8x..(\.\.\.\.\
00000250	00 50 00 49 00 50 00 45 00 5c 00 52 00 55 00 4e	.P.I.P.E.\.R.U.N
00000260	00 5f 00 41 00 53 00 5f 00 55 00 53 00 45 00 52	._.A.S._.U.S.E.R
00000270	00 02 00 48 01 25 00 34 00 2e 00 34 00 64 00 2d	...H.%4...4.d.-
00000280	00 25 00 32 00 2e 00 32 00 64 00 2d 00 25 00 32	.%.2...2.d.-.%.2
00000290	00 2e 00 32 00 64 00 20 00 25 00 32 00 2e 00 32	...2.d. .%.2...2
000002a0	00 64 00 3a 00 25 00 32 00 2e 00 32 00 64 00 3a	.d...%.2...2.d.:
000002b0	00 25 00 32 00 2e 00 32 00 64 00 3a 00 02 00 07	.%.2...2.d.:....
000002c0	01 53 68 65 6c 6c 54 32 02 00 08 01 54 65 6c 6e	.ShellT2....Teln
000002d0	65 74 54 32 00 00 0a 00 78 79 01 00 0b 00 0a 00	etT2....xy.....



```
rule PlugxA-Sub-Rule2-Example { strings:  
    $sub_rule_2_hex = {75 61 6C 41 C7 [0-16] 6C 6C 6F 63 }  
condition:  
    $sub_rule_2_hex }
```

90 02 XX

→ 字节位于模块 90 02 10

标记为 紫色

→ 整个子规则 2 标记为 红色

	Replaced bytes for pattern 90 02	Sub-rule 2	
00000000	04 00 8d 8d 3c ff ff ff 51 56 c7 85 3c ff ff ff	<ÿÿÿQVÇ.<ÿÿÿ
00000010	56 69 72 74 b3 6c c7 85 40 ff ff ff 75 61 6c 41		Virt³lç.@ÿÿüuaLA
00000020	c7 85 44 ff ff ff 6c 6c 6f 63 c6 85 48 ff ff ff		Ç.DÿÿÜllocÆ.Hÿÿÿ
00000030	00 ff d7 89 45 f8 85 c0 75 0e 5f 5b b8 04 00 00		.ÿx.Eø.Àu._[,...
00000040	00 5e 8b e5 5d c2 04 00 8d 95 1c ff ff ff 52 56		.^å]Â....ÿÿRV
00000050	c7 85 1c ff ff ff 56 69 72 74 66 c7 85 20 ff ff		Ç..ÿÿÿVirtfÇ.ÿÿ
00000060	ff 75 61 88 9d 22 ff ff c7 85 23 ff ff ff 46		ÿua.."ÿÿç.#ÿÿF
00000070	72 65 65 c6 85 27 ff ff ff 00 ff d7 89 45 a8 85		reeÆ.'ÿÿ.ÿx.E".
00000080	c0 75 0e 5f 5b b8 05 00 00 00 5e 8b e5 5d c2 04		Àu._[,...^å]Â.
00000090	00 8d 85 fc fe ff ff 50 56 c7 85 fc fe ff ff 45		...üþÿPVÇ.üþÿE
000000a0	78 69 74 c7 85 00 ff ff ff 54 68 72 65 66 c7 85		xitç..ÿÿÿThrefç.
000000b0	04 ff ff ff 61 64 c6 85 06 ff ff ff 00 ff d7 85		.ÿÿÿadÆ..ÿÿ.ÿx.
000000c0	c0 75 0e 5f 5b b8 06 00 00 00 5e 8b e5 5d c2 04		Àu._[,...^å]Â.

```
PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -File 'C:\Users\user\deeac56026f3804968348c8afa5b7aba10900aeabee05751c0fcac2b88cff71e' -DisableRemediation -Trace -Level 0x10
Scan starting ...
Scan finished.
Scanning C:\Users\user\deeac56026f3804968348c8afa5b7aba10900aeabee05751c0fcac2b88cff71e found 1 threats.

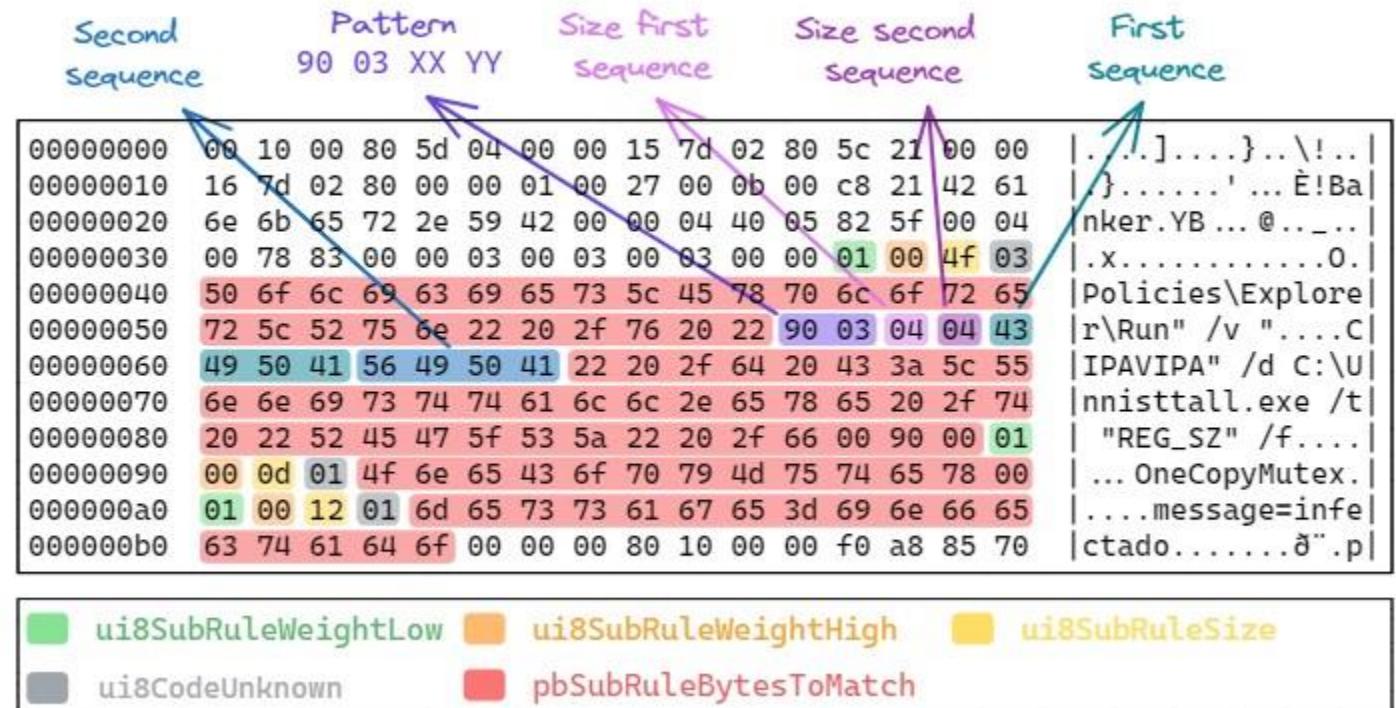
<=====LIST OF DETECTED THREATS=====>
----- Threat information -----
Threat : Backdoor:Win32/Plugx.A
Resources : 1 total
file : C:\Users\user\deeac56026f3804968348c8afa5b7aba10900aeabee05751c0fcac2b88cff71e
```

Expected
detection

90 03 XX YY

模块 90 03 XX YY:

- XX : 首个序列 (序列 A) 长度 , 标记为 粉色
- YY : 次个序列 (序列 B) , 标记为 葡萄紫
- 匹配的样本中序列 A 或 B 或许会出现



```

rule
BankerYB_Sub_Rule1_Example{   strings:
    $sub_rule_1_hex = { 50 6f 6c 69 63 69 65 73 5c 45 78 70 6c 6f 72 65 72 5c 52 75 6e 22 20 2f 76 20 22
                        (43 49 50 41|56 49 50 41) 22 20 2f 64 20 43 3a 5c 55 6e 6e 69 73 74 74 61 6c 6c 2e
                        65 78 65 20 2f 74 20 22 52 45 47 5f 53 5a 22 20 2f 66 00 90
    00 }   condition:   $sub_rule_1_hex
}

```

90 04 XX YY

- XX : 预计字节长度
- YY : 类正则表达式模块在图中标记为紫色
- 90 04 XX YY 后字节以正则表达式方式描述模块本身
- 此示例为 30 2d 39, 用蓝色标记是 0-9

```
rule Pattern-90-04-example
{
    strings:
```

```
        $example1_90_04 =
{ 68 74 74 70 3a 2f 2f 61 72 70 2e 31 38 31 38 [30-39]
- [30
[39] 2e 63 6e 2f 61 72 70 2e 68 74 6d 90 00 }
```

```
        $example2_90_04 =
{ 5c 48 61 70 70 79 [30-39] -39] 68 79 74 2e 65 78 65 90
00 } [30
```

```
    condition:
        $ example1_90_04 and
$ example2_90_04 }
```

example1	Pattern 90 04 XX YY	Size of the regex-like pattern	Regex-like pattern
00000000	32 00 23 02 68 74 74 70 3a 2f 2f 61 72 70 2e 31	2.#.http://arp.1	
00000010	38 31 38 90 04 02 03 30 2d 39 2e 63 6e 2f 61 72	818....0-9.cn/ar	
00000020	70 2e 68 74 6d 90 00 00 00 00 00 00 00 00 00 00	p.htm.....	
	ui8SubRuleWeightLow ui8SubRuleWeightHigh ui8SubRuleSize	ui8CodeUnknown pbSubRuleBytesToMatch	

example2	Pattern 90 04 XX YY	Size of the regex-like pattern	Regex-like pattern
00000000	01 00 1d 03 5c 48 61 70 70 79 90 04 02 0a 30 31\Happy....01	
00000010	32 33 34 35 36 37 38 39 68 79 74 2e 65 78 65 90	23456789hyt.exe.	
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
	ui8SubRuleWeightLow ui8SubRuleWeightHigh ui8SubRuleSize	ui8CodeUnknown pbSubRuleBytesToMatch	

90 04 XX YY

→ 取代 90

04 02 03 30 2D 39 (示例 1) 的字段为:

→ 0x30

→ 0x39

→ 标记为青色

→ 红色字段命中子规则

Replaced bytes for pattern 90 04 02 03 30 2d 39

00000000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 64 72 69 76driv
00000010	65 72 73 5c 73 79 73 74 65 6d 2e 65 78 65 00 00	ers\system.exe..
00000020	00 00 00 00 00 68 74 74 70 3a 2f 2f 61 72 70 2ehttp://arp.
00000030	31 38 31 38 30 39 2e 63 6e 2f 61 72 70 2e 68 74	181809.cn/arp.ht
00000040	6d 90 00 00 00 00 00 00 00 00 00 72 6f 76 65 72 00	m.....rover.
00000050	00 00 00 00 00 00 77 70 63 61 70 2e 64 6c 6c 00wpcap.dll.
00000060	00 00 00 00 00 6d 79 65 78 65 00 00 00 00 00 00myexe.....
00000070	00 00 64 72 69 76 65 72 73 5c 6e 70 66 2e 73 79	..drivers\npf.sy
00000080	73 00 00 00 00 00 00 00 50 61 63 6b 65 74 2e 64	s.....Packet.d
00000090	6c 6c 00 00 00 00 00 00 00 57 61 6e 50 61 63 6b	ll.....WanPack
000000a0	65 74 2e 64 6c 6c 00 00 00 00 00 00 00 5f 64 65	et.dll....._de
000000b0	6c 65 74 65 6d 65 2e 62 61 74 00 00 00 00 00 00	leteme.bat.....
000000c0	00 3a 74 72 79 00 00 00 00 00 00 00 69 66 20 20	:try.....if
000000d0	20 65 78 69 73 74 00 00 00 00 00 00 00 00 00 00	exist.....

example1	Pattern 90 04 XX YY	Size of the regex-like pattern	Regex-like pattern
00000000	32 00 23 02 68 74 74 70 3a 2f 2f 61 72 70 2e 31	2.#.http://arp.1	
00000010	38 31 38 90 04 02 03 30 2d 39 2e 63 6e 2f 61 72	818....0-9.cn/ar	
00000020	70 2e 68 74 6d 90 00 00 00 00 00 00 00 00 00 00	p.htm.....	
	ui8SubRuleWeightLow ui8SubRuleWeightHigh ui8SubRuleSize		
	ui8CodeUnknown pbSubRuleBytesToMatch		

```
PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -File 'C:\Users\user\threat-small.exe' -DisableRemediation -Trace -Level 0x10
Scan starting ...
Scan finished.
Scanning C:\Users\user\threat-small.exe found 1 threats.

<=====LIST OF DETECTED THREATS=====>
----- Threat information -----
Threat : Backdoor:Win32/Small
Resources : 1 total
file : C:\Users\user\threat-small.exe
----- Expected detection -----
```

90 05 XX YY

- XX : 预计字段**最大**长度
- YY : 图中正则匹配式长度标记为**葡萄紫**
- 和 04 模块不同，此参数不区分大小写
- 90 05 XX YY 后的字段用正则匹配式描述模块本身

	Pattern 90 05 XX YY	Pattern length	Regex-like pattern
00000000	dc e6 2b 01 00 80 5d 04 00 00 e6 ad 01 80 5c 23		Üa+ ...] ... æ ... \#
00000010	00 00 e7 ad 01 80 00 00 01 00 04 00 0d 00 88 21		...ç.....!
00000020	53 74 72 61 74 69 6f 6e 2e 43 43 00 00 01 40 05		Stration.CC ... @.
00000030	82 5c 00 04 00 78 5f 00 00 1e 00 1e 00 03 00 00		.\\ ... x.....
00000040	0a 00 18 00 47 45 54 20 2f 64 66 72 67 33 32 2e	GET /dfrg32.
00000050	65 78 65 20 48 54 54 50 2f 31 2e 31 0a 00 1f 02		exe HTTP/1.1....
00000060	68 74 74 70 3a 2f 2f 90 05 40 03 61 2d 7a 2e 63		http:// ..@.a-z.c
00000070	6f 6d 2f 64 66 72 67 33 32 2e 65 78 65 90 00 0a		om/dfrg32.exe ...
00000080	00 13 02 48 6f 73 74 3a 20 90 05 40 03 61 2d 7a		... Host: ..@.a-z
00000090	2e 63 6f 6d 90 00 00 00 5d 04 00 00 e7 ad 01 80		.com....] ... ç ...

```

rule Pattern-90-05-
example{ strings:
    "$example_90_05 ="
    "http:// [a-zA-Z]{0,64} \\.com/dfrg32\\.exe"
    condition:
        $example_90_05
}

```

实验 3: 匹配检测

→ 打开 msys64 文件夹并运行 msys64.exe

→ 切换至实验根目录

```
cd /c/<your_path>/lab3_stration/Exercise
```

→ 分析 Stration.CC PEHSTR 签名

了解权重和通配符

→ 修改提供的 StrationCC.c 文件使其在编译后同 Stration.CC 检测结果匹配

→ 用 Build.sh 脚本进行编译

解决方案: 模块: 90 05 XX YY

- 子规则 2 中取代了 90 05 40 03 61 2D 7A 模块的字段标记为蓝色
- 子规则 2 于绿色中修复的字段
- 预计检测:

	sub-rule 1 fixed bytes	Pattern 1 matching sub-rule 2: 0x40 bytes	Pattern 2 matching sub-rule 3: 0x40 bytes
00000000	00 00 00 00 40 00 00 40	2e 72 65 6c 6f 63 00 00@...@.reloc..
00000010	30 00 00 00 00 60 00 00	00 02 00 00 00 2a 00 00	0.....`.....*..
00000020	00 00 00 00 00 00 00 00	00 00 00 00 00 40 00 00@...B
00000030	00 00 00 00 00 00 00 00	00 00 00 00 00 47 45 54GET
00000040	2f 64 66 72 67 33 32 2e	65 78 65 20 48 54 54 50	/dfrg32.exe HTTP
00000050	2f 31 2e 31 00 00 00 00	00 00 00 68 74 74 70 3a	/1.1.....http:
00000060	2f 2f 41 41 41 41 41 41	41 41 41 41 41 41 41 41	//AAAAAAAAAAAAAA
00000070	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAA
00000080	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAA
00000090	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAA
000000a0	41 41 2e 63 6f 6d 2f 64	66 72 67 33 32 2e 65 78	AA.com/dfrg32.ex
000000b0	65 90 00 00 00 00 00 00	00 00 00 48 6f 73 74 3a	e.....Host:
000000c0	41 41 41 41 41 41 41 41	20 41 41 41 41 41 41 41	AAAAAAAAAAAAAAA
000000d0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAA
000000e0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAA
000000f0	41 41 41 41 41 41 41 41	41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAA
00000100	2e 63 6f 6d 90 00 00 00	00 00 00 00 00 00 00 00	+com.....

sub-rule 2 fixed bytes sub-rule 3 fixed bytes

Stration.CC

```
PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -File 'C:\Users\user\threat-strationcc.exe' -DisableRemediation -Trace -Level 0x10
Scan starting ...
Scan finished.
Scanning C:\Users\user\threat-strationcc.exe found 1 threats.

<=====LIST OF DETECTED THREATS=====>
----- Threat information -----
Threat : TrojanDownloader:Win32/Stration.CC
Resources : 1 total
file : C:\Users\user\threat-strationcc.exe
----- Expected detection -----
```

最终实验

- 目标: 实现一个可运作的命中了 Defender Backdoor:Win64/Havoc.A!MTB 签名检测的样本
打开解压的签名数据库并找到签名
 - 明确签名类型
 - 明确签名字节的意义
- 2. 反编译在 `lab4_havoc\Exercise\sample.zip` 提供的样本（这是个真样本，小心，密码：`infected`）
 - 找到并分析命中检测的功能
- 3. 修改 `lab4_havoc\Exercise\havoc_emu_asm.S` 以实现与提供的 `XorAlgorithm` 样本相同的功能
- 4. 使用 `build.sh` 脚本构建



<https://retooling.io/blog>

That's All Folks

silvio@retooling.io
antonio@retooling.io

参考

- <https://www.safebreach.com/blog/defender-pretender-when-windowsdefender-updates-become-a-security-risk/>
- <https://gist.githubusercontent.com/mattifestation/3af5a472e11b7e135273e71cb5f>
- <https://raw.githubusercontent.com/15be4f2ae75b2d62465cf9faef72a2f61147a393/ExpandDefenderSig.p1>
- <https://learn.microsoft.com/en-us/defender-endpoint/command-linearguments-microsoft-defender-antivirus>