**4.3 特征库子系统**

为检测与防护系统提供静态特征库与动态行为特征库，特征库结构设计合理、特征描述信息全面，建设的主要目标为：

**1. 收集、整理、分析已知木马程序（含特种木马），从中提取特征码，生成特征码库，用于防护系统软件静态检测木马程序之依据。**

所谓的特征代码是指：防护系统捕获到一个程序后, 将会提取该程序中比较关键的一段代码作为辨认这个程序的特征代码，将它和已知的特征序列做比对。

独立设计并实现原型特种木马，在虚拟机环境中，特种木马能够完成整个生命周期的行为动作，提取原型特种木马的特征码，并存储一个到两个可实际运行的摆渡木马用于特种木马防护技术研究。

收集灰鸽子等多种已知木马，运用修改代码、Hash文件校验、PE文件格式解析、二进制特征代码提取等多种方式，从已知木马中提取特征码。

选择符合涉密信息系统环境特点，并可能进入涉密信息系统的未知的特种木马程序，重点研究并提取其特征码。

特征码提取的基本方式如下：

第一阶段：自动参数中，生成文件间隔秒数设为4，最小替换字节数设为100 字节。主要用于大体定位内存特征码；

第二阶段：自动参数中，生成文件间隔秒数设为4，最小替换字节数设为4 字节。主要用于准确定位内存特征码。

目前，常见的特征码扫描方式包括文件查杀、内存查杀和行为查杀。同时，针对上述查杀方式，特种木马程序往往选择通过加壳、加花指令、修改壳和修改特征码的方式，试图躲避防护系统查杀。为此，在对木马特征进行提取和分析的过程中，应采取多种方案并行处理的方式，准确找寻木马程序的特征代码部分，并存入特征码库中，算法高效、可扩展性强。

为了提高静态扫描的准确性，确保涉密信息系统安全性，特征码库的容量应保持在300条以上。

**2. 建立基于特种木马的整个生命活动周期的行为特征库，用于检测与防护系统软件动态防护与检测木马程序之依据。**

无论木马如何隐蔽自己，在其植入阶段、安装阶段、运行阶段和通信阶段都会表现出一定的行为特征。基于整个生命周期的行为特征库容量将达到500条以上，且均采取生命周期内的组合规则存储。

1. 植入阶段的行为特征分析

在本阶段，攻击者通过各种方式将木马植入到目标机器中，一些特征如下：

1）利用操作系统的漏洞进行攻击植入，造成缓冲区溢出。

* 堆栈型(Stack) 缓冲区溢出；
* 格式化字符串(Format String) 漏洞；
* 堆(Heap) 和静态数据(BSS) 的缓冲区溢出；

2）利用应用软件的漏洞进行植入

3）利用交互脚本植入：如通过Script、ActiveX 及XML、ASP、CGI 交互脚本植入。

4）通过伪装文件植入：将木马的被控制端以文件的方式发送给目标用户，一旦用户点击该文件，被控制端程序将自动被植入到目标机器。

5）利用Java Applet 植入：先利用HTML 把木马被控制端下载到目标机器的缓存中，然后再通过修改注册表，指向此木马被控制端。

1. 安装阶段的行为特征分析

和其他阶段相比，木马被控制端程序在安装阶段存在显著的区别于一般合法程序的行为特征。表现出的具体行为特征如下：

1）自动压缩或者解压缩文件：利用Winrar，Winzip等压缩文件，木马被控制端制作成自解压的执行程序，自动创建或解压缩 RAR、ZIP 文件。

2）文件增大：木马被控制端安装时捆绑在记事本、写字板、注册表等常用程序上，导致这些感染程序文件增大。

3）隐藏伪装安装木马控制端过程中产生的部分文件。

* + 将文件属性设置为系统隐蔽；
  + 将文件属性设置为只读；
  + 将文件图标更换进行伪装，冒充系统文件或者图片、文本等其他非可执行的文件；
  + 将文件的名字进行伪装，冒充系统文件或常用的应用程序文件名；
  + 对文件安装的目录进行伪装。如 windows或者windows\system32 目录下或C:\Documents and Settings\用户名\Local Settings\Temp 目录下，并且修改自身的生成日期；
  + 将文件自我删除：木马被控制端安装之后，安装文件将自动被删除，很好地隐藏了木马被控制端文件及来源；
  + 利用 Windows NTFS 分区数据流达到文件隐藏。根据一个子文件系统允许额外的数据链接到一个特别的文件中的原理，将文件隐藏在系统中任何文件中通过修改内核资料达到隐藏文件的目的。

4） 木马被控制端在目标机器中设置自启动。

在Autostart文件所在目录下建立木马被控制端的快捷方式 , 并写入程序中的启动项目组。对木马自动启动行为的监控与控制，监控对象包括：登录时运行的程序，资源管理器插件，IE浏览器插件，系统服务，驱动程序，计划任务

* Autostart文件在Windows 的中文版本 / 英文版本和法文版本中分别对应目录：C:\windows\start menu\programs\startup

C:\windows\Menu Darrer\Programmes\Darrage

在这两个目录下建立木马被控制端 .exe 快捷方式，则系统启动时会运行木马被控制端 .exe。

* 设置Win.ini文件与System.ini文件设置自启动

在Win.ini 文件中设置[windows]节为load=木马被控制端 .exe run=木马被控制端.exe，则系统下次启动时，可启动木马被控制端 .exe。

在System.ini文件中设置[boot]Shell=Explorer.exe木马被控制端 .exe，则以后系统启动时启动木马被控制端 .exe。[386Enh]字段内的“driver=路径程序名”及[mic]、 [drivers]、 [drivers32]字段，也被有些木马被控制端修改后用于自启动。

* 修改Autoexec.bat( 在引导系统时执行)、winstart.bat(在启动GUI图形界面环境时执行)或Dos start.bat(在进入MS-DOS方式时执行或 Config.sys文件主要是设置一些启动参数和加载驱动程序)等批处理文件下加载启动木马被控制端。
* 利用Windows 根目录下的 wininit.ini 文件，对木马被控制端进行自启动或执行修改文件等操作。
* 修改“组策略”设置自启动。
* 利用 Schedule 服务和 AT 命令设置自启动。
* 利用 WEB 服务实现木马被控制端的启动。

如果目标机器提供WEB服务，首先将木马被控制端拷贝到目标机器的WWW可执行目录，如cgi-bin或scripts，然后在浏览器地址栏输入url如：

http://x.x.x.x/scripts/ 木马被控制端.exe或http://x.x.x.x/cgi-gin/ 木马被控制端.exe，则目标机器就会启动木马被控制端 .exe。

* + 利用 IE 开始页启动木马被控制端。

修改设置注册表中IE启动时首先打开的页面[HKEY\_CURRENT\_USER\Software\Microsoft\Internet Explorer\Main] 中“start page=”，将设定的网址前加入木马被控制端 .exe，则打开 IE 首先启动木马被控制端 .exe。

* + 利用Appname.local文件设置启动。

利用Appname.local文件进行的 DLL 转移就可以顺利替换任何应用程序启动时加载的默认 DLL。

* + 利用系统自动运行的程序启动。

在系统运行过程中，许多程序是自动运行的，比如:磁盘空间不足时，系统自动运行“磁盘清理”程序(cleanmgr.exe)；启动资源管理器失败时，双击桌面将自动运行“任务管理器”程序 (Taskman.exe)；格式化磁盘完成后，系统将提示使用“磁盘扫描”程序 (Scandskw.exe)；点击帮助或按Fl时，系统将运行winhelp.exe或h.exe 打开帮助文件等。木马被控制端通过覆盖目标机器的系统自动运行文件，实现自启动。

* + 利用 autorun.inf 文件自动运行功能。
  + 利用注册表实现相关程序的自动启动。

a 修改 Run 项

b 修改RunOnce项

c 修改RunServices项

d 修改RunServicesOnce项

e 修改Winlogon项

f 修改 Load 项

* 利用 MIME 漏洞。

5） 修改文件关联启动。

* 修改txtfile( 文本文件的关联 )
* 修改regfile( 注册表文件关联 )
* 修改exefile（exe 文件的关联 )
* 创建一个新文件类型，并修改之进行关联启动

6）系统突然弹出一个警告框或者是对话框，或者让用户回答一些奇怪的问题。

7）硬盘空间减少。

8）频繁读写硬盘驱动器。

9）修改系统时间。

10）重命名安全软件的 DLL。

11）破坏系统安全模式。

13）关闭窗口、服务。

14）强制安装软件。

1. 木马运行阶段的行为特征分析（文件访问、回联）

木马被控制端被成功安装到目标机器以后，进入运行阶段，为了防止被用户发现，完成了隐藏进/线程及免杀等动作，表现出的行为特征具体如下：

1）伪装系统进程名。

2）修改进程管理程序：如把要隐藏进程的pid设为0( 空转进程 )，实现进程隐藏等。

3）将指定进程注册为系统服务。启动服务控制分发器、注册服务控制处理器、设置服务状态信息。

4）利用 API HOOK 技术。

* 用户级 API HOOK
* 内核级 API HOOK

5）基于特洛伊 DLL 技术的进程隐藏。

* 利用远程线程技术将DLL木马插入到系统进程等已有进程中运行。
* 木马DLL替换系统DLL，截获进程对此系统DLL中所有函数的调用。

6）利用驱动的进程隐藏。替换掉系统驱动程序或其他必备程序，通过改写vxd、sys 文件建立隐蔽共享、

7）利用进程列表欺骗，在内存中以一个新的单独进程运行。

8）关闭特定进程。一般关闭的常常是杀毒软件和防火墙。

9）调用cmd进程。

1. 通信阶段的行为特征分析

木马被控制端与控制端通常需要建立通信通道进行信息交流，在网络通信阶段，其行为特征表现如下：

1）利用 1024 以上的高端口进行通信。

2）利用端口复用技术或端口寄生技术进行通信。

3）利用反向连接技术进行通信。主要利用防火墙的漏洞、反弹端口、创建套接字连接客户端。

4）利用潜伏技术，使用 ICMP 协议进行通信。

5）利用IRC及ICQ等方式与控制端进行通信。

6）通过发送 UDP 以及 FTP 的方式进行通信。

7）在通信过程中，产生异常的通信流量变化。

8）建立隐蔽通道。

* + 利用 Option 域和传输数据时通常很少用到的域。
  + 利用传输数据时必须强制填充的域。

9）木马被控制端处于监听状态，等待其他进程通信。

10）利用数据缓冲区暂存待发送的数据。

11）利用进程通信发许多 SYN 包。

基于上述木马生命周期的不同阶段，系统将收集和观测上述各种木马攻击行为，建立有效规则和监控策略，建立特种木马动态行为特征库，实现对木马，特别是未知木马的动态查杀。

3.3 特征库子系统

3.3.1 特征库组成

特征码格式主要有基于木马病毒样本的MD5特征码、基于PE文件段的MD5特征码、基于木马病毒的十六进制特征码等。

设计基于数字签名的样本特征库，通过加密的样本特征库文件在扫描引擎加载的时候会进行解码生成cld特征文件。其中，main.cld是引擎使用的主要特征码库。文件首行用冒号隔开的八个字段是对main.cld文件的解析。

VDB:11 Mar 2012 10-34 -0400:54:1044387:60:X:X:sven:1318343660。

各字段含义分别为：

VDB：库名，特征库文件的标志性字符（name of the file）

11 Mar 2012 10-33-0400：CVD文件创建时间（the creation time）

54：库版本号（the version number）

1044387：病毒特征码数目（the number of signatures）

60：需要的引擎版本号（the functionality level）

X：内部库MD5校验和（the MD5 checksum）

X：特征库库数字签名（the digital signature）

Sven：创建者名（the builder name）

1318343660：创建时间（秒）（time in seconds）

main.cld中包含main.db、main.fp、main.hdb、main.info、main.mdb、main.ndb、main.zmd七个库。对这些库的说明如下：

(1) main.db库

main.db库描述了一般文件病毒的特征码，是基于本体的十六进制特征码。基于本体的十六进制特征码就是在恶意软件中提取一段连续的字符串，然后转换为十六进制的字符串。十六进制特征码的最大特点就是支持通配符。特征码在数据库中的存储格式为：MalwareName=HexSignature，使用Boyer Moore算法进行匹配扫描。特征库以“=”符号为分隔符，前面为病毒名，后面为病毒特征码。main.db库的内容经函数cli\_loaddb解析后，存于结构体cli\_bm\_patt的virname和pattern成员中。每一个病毒对应一个cli\_bm\_patt结构实例，这些结构实例组成链表存入在结构cl\_node中。

main.db库部分内容列出如下：

\_0017\_0001\_001=b3005a8b4e27b440cd21e8c2045a59b440cd21e81902b440cd21b8004233c999cd218bd6b90300

\_0017\_0001\_002=8bfd83c72d515757e8b3005a8b4e27b440cd21e8c2045a59b440cd21e81902b440cd21b8004233

\_0017\_0001\_003=ee50f7d8250f008bc85803c150b440cd21582d0300c604e98944018bd6b985092bd1050301

\_0017\_0001\_004=40cd21e8c2045a59b440cd21e81902b440cd21b8004233c999cd218bd6b90300b440cd218b4c19

(2) main.fp

特征库main.fp是白名单数据库。白名单数据库使用MD5值作为特征码。白名单数据库格式相对简单。数据用于填充结构cli\_md5\_node，函数cli\_loadhdb解析库中的每一行，每行以"："分隔字段，格式为“MD5：文件大小：输出内容”。每个字段填入结构cli\_md5\_node中各成员，每行填充一个结构实例，多个实例组成链表。

库main.fp数据部分数据如下：

d1ef8a0e477570ad39f4667129400b05:1598056:Submission21770 332e5c92be38ce0f195019258c8376dc:1640013:Submission22475 71d934fdf522c4227485716b0413c7be:55296:Submission23647 810439699f2dc802ff8c530f59f23b7c:876032:Submission24405 848b157359e022907645c222b3daf72d:1942528 :Submission 32849

(3) main.hdb

特征库main.hdb存储MD5特征码。对文件整体取MD5值作为特征码，但是这种方法就仅仅针对静态病毒。特征库main.hdb的解析与库main.fp类似，都填充结构cli\_md5\_node，解析函数都是函数cli\_loadhdb，区别仅在于结构cli\_md5\_node填充0。

库main.hdb数据部分数据如下：

aaab755d9baf21baf05de8f32af2c996:57856:Trojan.BO.A-Cli d3edf9b7d99205afda64b3a7c1a63264:307200:Trojan.Boid.20-cli-1 de59dc8df6021d19246f9b74dd1d68bc:32768:Trojan.Boid.20-cli-2 d9c8d35d577b7bc2cdbe26282383400a:36864:Trojan.Boid.20-edit 565ce39278f60226fbbe920f79e77eb2:17408:Trojan.Ceptio.10-cli 01868bc1780b71996e90dafd180dae1b:13312 :Trojan.Ceptio.10-edit

(4) main.info

特征库main.info是对main.cvd解密解压缩后的各个库信息的说明。第一个字段是库文件名称，第二个字段是库文件大小，第三个文件是库文件的校验和。

部分数据如下：

main.db:4730592:aeecda2137b3ac4ba12af48a41ef02b4f834e1a4eb09c89c1347db432ef4e2c5

main.hdb:2556807:7f41b3d9f1f5ede969afb0bc64129cb6d9a75a695df446db20cf393ddac7e82f

main.mdb:52979900:8664fe1d62f2612e34e3882c87967cee9d853b0b233772a49dc5d63844dc5f62

main.ndb:17132295:9ca324b8079d9246e35c630834bdd97a8f67ca2b4955907e1aef6eb7eba9a9ad

main.zmd:217:c2311e27fcebaf2a67ffb734209be6e00fd8ce7130babb200612ad886134170d

main.fp:28420:205b24492c79657bbbf63021e3866f8fbe873029653d3b9bb89a579d98ad7e5e

(5) main.ndb

特征库main.ndb是十六进制扩展特征码。扩展特征码格式允许在特征码上添加额外信息，如目标文件类型，病毒偏移或引擎版本号等。扩展特征码可以让病毒检测工作更可靠，特征码在数据库文件中的存储格式为：“MalwareName:TargetType:Offset:HexSignature”，其中，TargetType是指文件类型，包括可执行文件类型、邮件类型、图片类型等等。Offset是一个“\*”或特殊标志类型，“\*”表示任意偏移、n表示绝对偏移、EOF-n表示距离文件结尾n个字节。

表3-1 TargetType主要类型表

|  |  |
| --- | --- |
| TargetType | 说明 |
| 0 | 任意文件 |
| 1 | PE文件，包括32位和64位 |
| 2 | 在OLE2容器中的文件 |
| 3 | HTML文件 |
| 4 | 邮件 |
| 5 | 图片 |
| 6 | ELF文件 |
| 7 | ASCⅡ文本文件 |
| 8 | 未使用 |
| 9 | Mach-O files，可执行文件，目标代码，动态库，内核转储的文件 |

特征库main.ndb用于填充结构cli\_ac\_patt，函数cli\_loadndb解析库中每一行，每行以“：”分隔字段，每个字段填入结构cli\_ac\_patt中各成员，每行填充一个结构实例，多个实例组成链表。

库main.ndb部分数据列出如下：

HTML.Phishing.Bank-1:3:\*:3c6d6170206e616d653d22{-36}223e3c6172656120636f6f7264733d22302c20302c20{3-12}222073686170653d22726563742220687265663d22{-160}3c2f6d61703e3c696d67207372633d226369643a

Exploit.HTML.MHTRedir.1n:3:\*:6d732d6974733a6d68746d6c3a66696c653a2f2f633a5c\*21687474703a2f2f

Exploit.HTML.MHTRedir.2n:3:\*:646174613d226d732d6974733a6d68746d6c3a66696c653a2f2f(63|64)3a5c

Exploit.HTML.MHTRedir.3n:3:\*:7372633d226d732d6974733a6d68746d6c3a66696c653a2f2f633a5c

Exploit.HTML.DragDrop:3:\*:6265686176696f723a75726c282364656661756c7423616e63686f72636c69636b293b\*666f6c6465723d227368656c6c3a

(6) main.mdb

特征库main.mdb是基于PE文件段的MD5特征码。引擎支持对PE文件的指定段获取MD5值。特征码在数据库文件中存储的格式为：PESectionSize:MD5:MalwareName。

库main.mdb部分数据列出如下：

168:a105e2cc8148158cd048360eb847c7d0:Trojan.Downloader-1421

7168:c61ef67b5e7eef19ef732f55116742f6:Trojan.Downloader-1422

7168:851b6320148122104f50445ea2684c9f:Trojan.Downloader-1423

（7）main.zmd

特征库main.zmd说明了病毒压缩情况。库main.zmd数据用于填充结构cli\_ac\_patt ，函数cli\_loadmd解析库中的每一行，若行开头为“#”符号，表示注释行不需要解析，注释行说明了各个字段名。每行以“：”分隔字段，每个字段填入结构cli\_meta\_node 中各成员，字段为“\*”符号表示为NULL。每行填充一个结构实例，多个实例组成链表。

库main.zmd部分数据列出如下：

Trojan.Dumador-31-zippwd-1:1:\*:21008:20598:ba9f27fb:8:1:1

Worm.Kimazo.A-zippwd:1:\*:75776:43733:7b3fcf13:\*:1:1

Worm.Banwarum.B-zippwd:1:\*:50176:43762:808ad272:\*:1:1

3.3.2 特征库加载

加载之前，需要获得有关特征库的配置选项，决定是否加载该类型特征库。调用cli\_cache\_init函数为引擎的树结构分配空间。检查当前对象属性，分别调用cli\_load或cli\_loaddbdir函数分别加载各自对应的数据文件。

cli\_loaddbdir函数用来加载目录中的所有特征库。引擎中常用特征库后缀对应加载函数和成员的对应关系如表3-2所示。

表3-2 常用特征库加载函数对应表

|  |  |
| --- | --- |
| 库后缀 | 加载函数 加载引擎成员 |
| “.db” | cli\_loaddb engine->root[0] |
| “.cvd .cld” | cli\_cvdload |
| ”.hdb .hdu .mdb  .mdu .fp” | cli\_loadmd5 engine->md5\_hdb  engine->md5\_mdb  engine->md5\_fp |
| “.ndb .ndu .sdb” | cli\_loadndb engine->root[target] |
| “.ldb .ldu” | cli\_loadldb engine->root[tdb.target[0]] |
| “.cbc” | cli\_loadcbc engine->root[tdb.target[0]] |
| “.zmd .rmd” | cli\_loadmd engine->cdb |
| “.cfg” | cli\_dconf\_load engine->dconf |

加载病毒库时，首先调用cli\_cvdload函数，流程如图3-13所示。

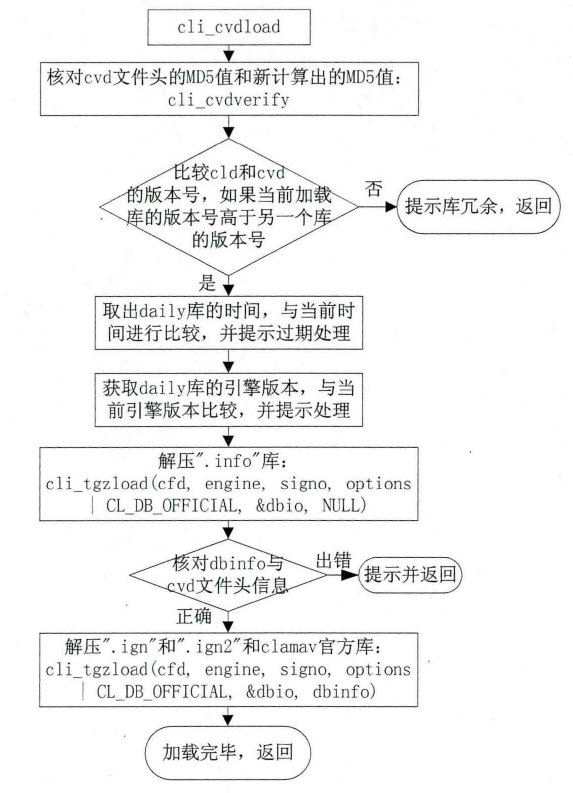


图3-13 特征库加载流程图

3.3.3 自定义特征库

本系统在特征库目录下建立了一个基于文件体的MD5特征库。通过自定义数据库编写新的特征码，可以扫描到默认特征码数据库中尚未检测到的模式。编写的特征码主要包括：

（1）MD5特征码

适用于静态恶意代码、病毒木马检测。对可疑文件提取MD5校验和并把它写入自定义的病毒库.hdb，在相应的路径下就会创建一个名为.hdb的库，打开此库，内容为：

48c4533230e1ae1c118c741c0db19dfb:17387:test.exe,内容符合病毒库中.hdb文件格式。使用引擎扫描文件，扫描结果：test.exe:test.exe FOUND

----------- SCAN SUMMARY -----------

Know viruses: 1

Scanned directories: 0

Scanned files: 1

Infected files: 1

Data scanned: 0.02 MB

Time: 0.024 sec (0 m 0 s)

（2）基于PE段的MD5特征码

创建一个PE文件中特定部分的MD5特征码。这种特征码存储在以.mdb为扩展名的文件中。存取特征码格式为：PESectionSize:MD5:MalwareName。

方法是准确提取目标PE部分到单独的文件，然后运行选项sigtool --mdb.

（3）基于本体的特征码

以十六进制格式存储，将一个恶意文件的一个片段转换成十六进制串，还可以使用各种通配符扩展。可使用sigtool工具将文本转换成十六进制码。在使用sigtool转换特征码时需要添加--hex-dump标志。Sigtool运行时需要通过标准输入设备提供所需的文本，并且将十六进制格式的结果输出到标准输出设备。输入文本时一个常见的错误是在输入文本时没有删除文本后面的换行符，这个换行符通常是由于按回车键而添加到该文本中。

下面的例子展示了如何使用sigtool生成"Hello world"字符串的十六进制码的输出，需要注意的是，必须删除字符串尾部的0a才能与原始模式匹配。

Sigtool --hex-dump

Hello world

68656c6c6f20776f726c640a

为了将这个十六进制转换成可用的特征码，还需要按照特征码语法规则对其进行格式化，这种可扩展的特征码包含由冒号分隔的下列四个字段：

MalwareName:TargetType:Offset:HexSignature

假设要检测包含"Hello world"字符串的任意文件，则应该创建如下特征码：

TestHelloWorld:0:\*:68656c6c6f20776f726c64

可以通过使用通配符来创建更复杂的特征码。比如，如果需要检测"hello"和"world"，但这两个字符串之间并一定仅有一个空格，就可以对该特征码作如下处理，即用一个通配符(??)来匹配0到FF之间的任意字节值。

TestHelloWorldAnySeparator:0:\*:68656c6c6f??776f726c64

也可以指定"hello world"字符串仅在文件中的固定偏移量出现：

TestHelloWorldOffset45:0:45:68656c6c6f20776f726c64

同样也可以指定偏移量的范围，下面的特征码仅会触发引擎检测文件中偏移量在200~250之间的"hello world"。

TestHelloWorldBetween200And250:0:200,50:68656c6c6f20776f726c64

最后，也可以指定"hello"和"world"按照顺序出现在文件中的任意偏移量。

TestHelloWorldAnyDistance:0:\*:68656c6c6f\*776f726c64

为了使用这些特征码，必须将其放置在以.ndb为扩展名的文件中。

使用引擎进行测试：

Clamscan -d clam\_helloworld.ndb test.txt

Test.txt: TestHelloWorldAnyDistance.UNOFFICIAL FOUND

----------- SCAN SUMMARY -----------

Know viruses: 5

Scanned directories: 0

Scanned files: 1

Infected files: 1

Data scanned: 0.01 MB

Time: 0.015 sec (0 m 0 s)

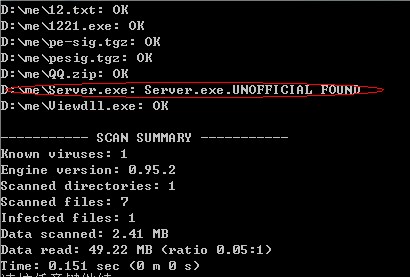
任何时候都可以创建并使用不在项目特征码数据库中的新特征码，新特征码通常以.UNOFFICIAL扩展名显示。查杀引擎会对所有新加入且不在默认项目特征码集中的特征码添加.UNOFFICIAL扩展名。

**实例：灰鸽子自定义特征提取**

MD5为ClamAV创建特征码最简单的方法是使用MD5校验和，但是这种方法只适用于静态的恶意软件。此方法使用了ClamAV自带工具sigtool.exe完成。例如对灰鸽子木马Server.exe文件取MD5校验和并把它写入自定义的病毒库MySigFile.hdb的方法如下：

在命令行执行C:\sigtool>sigtool --md5 Server.exe >> C:\MySigFile.hdb 在相应的路径下就会创建一个名为MySigFile.hdb的库，打开此库其内容为：

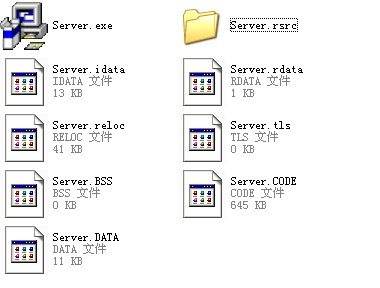
a766f67e11841ce2a73c7fde7e3b6053:761344:Server.exe,内容符合ClamAV病毒库中.hdb文件格式。使用ClamAV引擎扫描D:\Program Files\ClamWin\bin>clamscan -d MySigFile.hdb Server.exe 扫描结果：



2. 基于PE段的MD5

可以创建一个PE文件中特定部分的MD5特征码。这种特征码应该存储在以.mdb为扩展名的文件中。存取特征码格式为：PESectionSize:MD5:MalwareName。

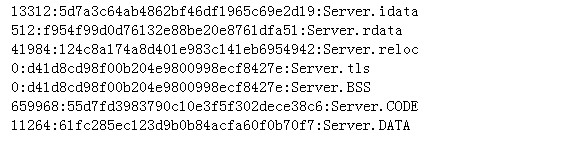
此种生成MD5特征码最简单的方法是准确提取目标PE部分到单独的文件，然后运行选项sigtool --mdb.利用相应软件能准确提取目标PE部分到单独的文件，可以利用7-zip解压缩软件或peditor软件将灰鸽子木马Server.exe每个PE节段分离出来，如下图所示：



然后对分理出的每个单独的文件提取MD5值。对单个文件提取MD5值并写入特征库中方法如下图：



提取出的每个节段的特征码写入MySigFile.mdb文件中



逐条特征测试扫描结果：

R0D[9`FIZDSPS0K6XTM(Q8T

NL[PQ34]1W~AWICZ@ZR~HI9

9X]76WHFHSS7PCHAJ@]L9}Q

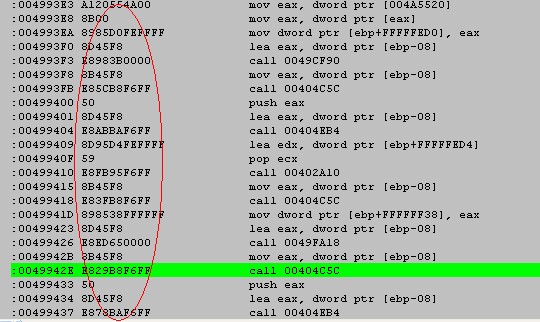
}9@{}1RJQS6YHUC4R_7$YHU

M979@${F2PEF4VCO01@U[76

3. 基于本体的特征码

ClamAV以十六进制格式存储所有的基于本体的特征码。它是指将一个恶意文件的一个片段转换成十六进制串，还可以使用各种通配符扩展。

以灰鸽子木马Server.exe为例，要想找出其特征，需要将其反汇编，根据汇编指令找出木马特征可能存在位置，然会提取出其对应十六进制串。



然后将十六进制串按照一定的格式写入相应的特征库：

R9}CW`TG@5HUW62S]L1RZLB

扫描结果：60S{Y0YESRQQG9(P{O4NSBC

3.3.4 行为规则库

行为规则库使用xml文件存储规则，并对xml文件进行加密，最终保存为dat格式，防止规则泄露，规则库内格式如下：

<rules>

<rule> <path>\*\Software\Classes\\*\shellex\ContextMenuHandlers</path>

<weight>2</weight>

<description>内容菜单</description>

</rule>

…

…

<rule>

<path>\*\SOFTWARE\Microsoft\InternetExplorer\AdvancedOptions\BROWSE\IEONDESKTOP</path>

<weight>6</weight>

<description>IE首选项</description>

</rule>

</rules>

对规则库进行加密后为乱码，程序读取规则库，首先对规则库进行解密。加密算法如下：

void encfile(wchar\_t \*in\_filename,wchar\_t \*pwd,wchar\_t \*out\_filename)

{

FILE \*fp1,\*fp2;

register wchar\_t ch;

int j=0;

int j0=0;

fp1=fopen(changtoduan(in\_filename),"rb");/\*以二进制只读方式打开要加密的文件\*/

fp2=fopen(changtoduan(out\_filename),"wb");

ch=fgetc(fp1);

while(!feof(fp1)){

fputc(ch^pwd[0],fp2);/\*异或后写入fp2文件\*/

ch=fgetc(fp1);

}

fclose(fp1);/\*关闭源文件\*/

fclose(fp2);/\*关闭目标文件\*/

}

规则库建立后，要完成对规则库的读取，添加和删除规则。

（1）读取规则库操作如下：

while(FirstRule){

//获得第一个rule的path节点和weight节点和description节点 TiXmlElement \*PathElement = FirstRule->FirstChildElement();

TiXmlElement \*WeightElement = PathElement->NextSiblingElement();

TiXmlElement \*DescriptionElement = WeightElement->NextSiblingElement();

TiXmlAttribute \*IDAttribute = FirstRule->FirstAttribute();

//插入一行，且每次插入内容都是现在界面最上方

m\_rulelist.InsertItem(0,\_T(" "));

……

在系统前端界面加入读取的数据

……

FirstRule = FirstRule->NextSiblingElement();

}

（2）删除某一条规则方法如下：

TiXmlDocument \*rulesDocument = new TiXmlDocument("c:/tmprules.abc");

rulesDocument->LoadFile();

//获得根元素，即rules。

TiXmlElement \*RootElement = rulesDocument->RootElement();

TiXmlNode \*RootElementNode = RootElement->ToElement();

//获得第一个rule节点。

TiXmlElement \*FirstRule = RootElement->FirstChildElement();

TCHAR PathElementItemText[200];

for (int i=0;i<m\_rulelist.GetItemCount();i++)//循环遍历所选复选框

{

if (m\_rulelist.GetCheckState(i))

{

while(FirstRule)

{

//获得第一个rule的path节点和weight节点和description节点

TiXmlNode \*FirstRuleNode = FirstRule->ToElement();

TiXmlElement \*PathElement = FirstRule->FirstChildElement();

//规则库里面的规则path

const char\* PathElementText = PathElement->FirstChild()->Value();

m\_rulelist.GetItemText(i,1,PathElementItemText,200);//前台list中显示的规则path if(lstrcmp(PathElementItemText,duantochang(PathElementText)))

{

FirstRule = FirstRule->NextSiblingElement();

}

else

{

RootElement->RemoveChild(FirstRuleNode);

rulesDocument->SaveFile("c:/tmprules.abc");

encfile(out\_filename,pwd,out\_filename2);//加密

m\_rulelist.DeleteItem(i);

i--;//针对同时点选多个复选框，删除一个后要让i的值减，继续操作

FirstRule = RootElement->FirstChildElement();//此时FirstRule实际上已经不是第一个节点了，是中间的某一个，在跳出循环之前要重新置回头节点

break;

}

}

}

}

（3）添加规则操作如下：

void InsertButtonRule()

{

/\*读取规则库，使规则库处于打开状态，进而完成添加规则动作\*/

//为节点添加内容（具体行为规则）

RulePath->LinkEndChild(RulePathText);

RuleWeight->LinkEndChild(RuleWeightText);

RuleDescription->LinkEndChild(RuleDescriptionText);

//按规则库的具体格式将规则插入规则库内 RuleElement->LinkEndChild(RulePath);

RuleElement->LinkEndChild(RuleWeight);

RuleElement->LinkEndChild(RuleDescription);

rulesDocument->SaveFile("c:/tmprules.abc");

encfile(out\_filename,pwd,out\_filename2);//加密

}