Running Head: VOGON\_THUMBDRIVE.IMG FORENSIC REPORT

Vogon\_thumbdrive.img Forensic Report

Rungsree Singholka

CFR105 Midterm Practical

University of Advancing Technology

Vogon\_thumbdrive.img Forensic Report

Mr. Dalil from Saraquoit Corporation requested a 124 MB image be examined due to a tip on a disgruntle employee Mr Steve Vogon. The purpose of this paper is to answer the following questions

* Were there any steps taken or actions performed to conceal the drive’s contents
* What files were found on this disk and how were they recovered
* Full explanation for each file include file type, contents and purpose
* Mr Steve Vogon’s intentions, and did he do it.
* Evidence to support the evidence

MBR

MBR says there is one non-bootable partition of type 0x0c, which is FAT32 LBA. Partition starts at sector 0x01, and total size of 252711 sectors or 129388032 bytes.

However, image size from ls –l returns 129499136 bytes, which means there is 110592 bytes or 216 sectors un-partitioned area at the tail end of the disk. A quick check with the following command shows no hidden data there.

dd if=./Vogon\_thumbdrive.img bs=512 skip=252711 | xxd | more

Reserve Area

At offset

0xb byte/sector: 512 bytes

0xd sector/cluster: 4 sectors

0xe reserve area size: 1 sector

0x10 number of FAT: 2

0x11 max file entry in root directory: 512 (default for FAT16)

0x13 total sectors in file system: 0

0x16 size of FAT: 247 sectors

0x20 total sectors in file system: 252710 sectors

From the required fields above, I really can’t tell if this image is FAT16 or 32. If this was FAT32 offset 0x24 would have returned a reasonable FAT size, and offset 0x11 would have been 0. However, the value was 2267611136 sectors, and that is not possible. In addition offset 0x2c would also returned a reasonable cluster number for root directory, but instead it was 538976288, which is also impossible. So, I conclude that this image is a FAT16. In a FAT16 format the root directory immediately follows the FATs area. So it must be located in MBR + reserve area + FAT + FAT = 1 + 1 + 247 + 247 = 496 sector. And the size is equal to number of entries \* 32 / 512 = 512 \* 32 / 512 = 32 sectors. Following the root directory is cluster 2 or 496 + 32 = 528 sector. A quick look at both FAT shows that there were no clusters allocated, but the beginning 0xfff8 and 0xffff. According to Maverick-os, this is expected with 0xf8 being the media type value, which is consistent with what I found in offset 0x215 of disk image.

dd if=./Vogon\_thumbdrive.img bs=512 skip=2 count=247 | xxd | more

dd if=./Vogon\_thumbdrive.img bs=512 skip=249 count=247 | xxd | more

The root directory was also completely empty with only 0x00

dd if=./Vogon\_thumbdrive.img bs=512 skip=496 count=32 | xxd | more

Since there is no data in both FATs, root directory, and the un-partition area of the disk, I’ll have to turn to file carving. However, before doing that, I’ll throw the image to FTK and Autopsy and see what they say.

FTK

Sector count 252,928 sectors = 129499136 bytes

Partition 1

Start sector 1, size 252,711 sectors

FAT16

Root directory: no entries

FAT: only 0xfff8 and 0xffff.

Unallocated space

Start 252712 sector, and no data found.

Autopsy

One partition with FAT32, Win95

1 audio, 2 zip files.

All 4 html and 98 plain text files most are fragments of Perl scripts.

4 executable and 4 DLLs

In deleted file system, Autopsy says there are files with many different names, but the ones that got my attention are FSEVEN~1 create/modify on 2008-08-08 23:39:02 PT ,VolumeConfig.plist create/modify on 2008-08-08 23:39:02 PT, and Psid.db create on 2008-08-08 23:39:03 PT and modified on 2008-08-08 23:39:34 PT

Content from psid.db

Billing and delivery information.html

upx.exe

libintl3.dll

libiconv2.dll

libeay32.dll

ssleay32.dll

wget.exe

svxfer.exe

upxxfer.exe

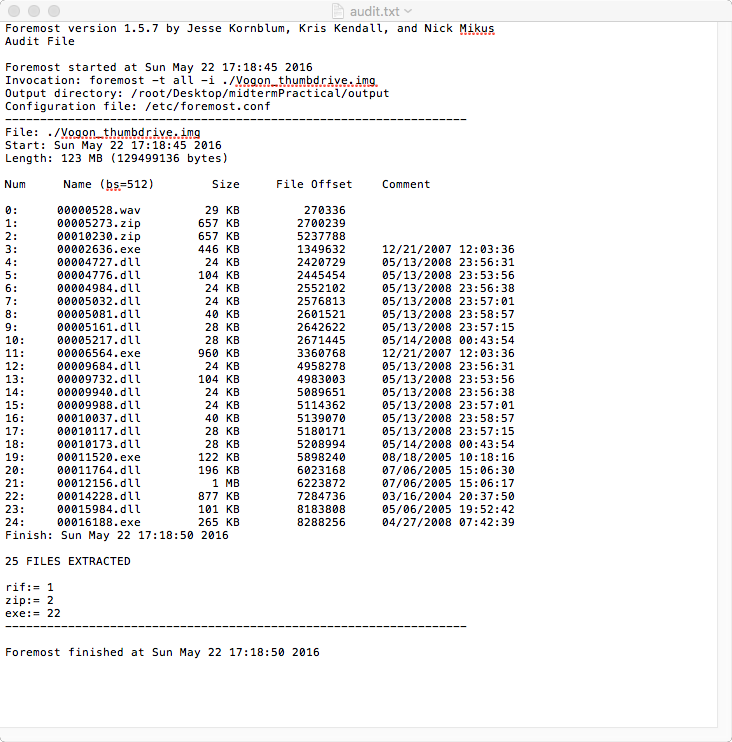
secret

keyfile.wav

The reason why I think these files are interesting is because FSEVEN~1 or fseventsd according hostilefork.com (2009), is a default log file in OSX when a file is create, delete or modified. I tried creating a small disk image with 500 MB with FAT16. The first thing I see in the root directory is .Trash and .fseventsd. If I try to delete them, and unmount, I can still see them in the root directory when view with xxd, but the FAT table will have only ‘f8ff ffff’. For VolumeConfig.plist, Macosxhint (2009), say it’s a file exist from OSX 10.5 and later that is used to tell the SpotLight which file to index or not to. Another thing I noticed from this small experiment is that, although the file system is FAT16, the partition type was FAT32. Steve Vogon’s file system partition type was also a FAT32 only it was 0x0c while mine was 0x0b. They are both FAT32 but one is FAT32 CHS and the other is FAT32 LBA. This is also the “mkdosfs” in the OEM, and according to Die.net, it is a tool in Linux used to create MSDOS file system in Linux. I created a FAT disk image in OSX, copied it over to Kali, mount, and ran mkfs.msdos on the partition. The result is OEM of mkdosfs with a FAT file system. When I run xxd ./image | grep ‘FSEVEN’, I still get the entries for the fseventsd from the previous OSX FAT format. From the file collected and this quick experiment, I’m concluding that the OS used on the thumb drive is an OSX, but later reformatted with mkfs.msdos to hide the data. For the Psid.db, I’ll talk about it after I carved the disk with foremost in Kali. After running the image through foremost with carving all files flag (-t all).

foremost -t all –i ./Vogon\_thumbdrive.img

The program was able to extracted the following files



Zip files

After extracted, the zip file doesn’t seem to work on window, however, it worked find on OSX. The reason is because both zip file 00005273.zip and 00010230.zip are Perl Achieve file. From CPAN (2006), Perl PAR (Perl Archive Toolkit) file is a packed self-contained script/exe with necessary 3rd-party modules that requires only core Perl to run on the target system. So, a simple Perl script can be made into an executable by running the following command that comes with the PAR package. According to CPAN (2006), a Perl PAR can be created with the pp utility, and the results will be like a regular zip file with a PK signature in the beginning. I’ve test this on my computer and it is confirmed.

pp –p ./sth.pl

To build a Perl script into an EXE Perl Training Australia (2014), says “One of the greatest uses of PAR is being able to create a stand-alone executable, which will work on systems that don't even have Perl installed”

pp -o hello.exe hello.pl           # Windows



Above is the unzipped 5273.zip file’s MENIFEST opened in Firefox. According to CPAN (2006), the MENIFEST file inside the zip (Perl PAR) shows the packed executable dependencies, and can be viewed with a normal web browser. After looking at the Perl scripts in the program, the program name is “exfer.pl”, and the code is malicious in nature. It will send a given file to the receiver disguised as multiple http get request traversing through the follow URLs

#my $proxy\_ip = "219.93.175.67:80";

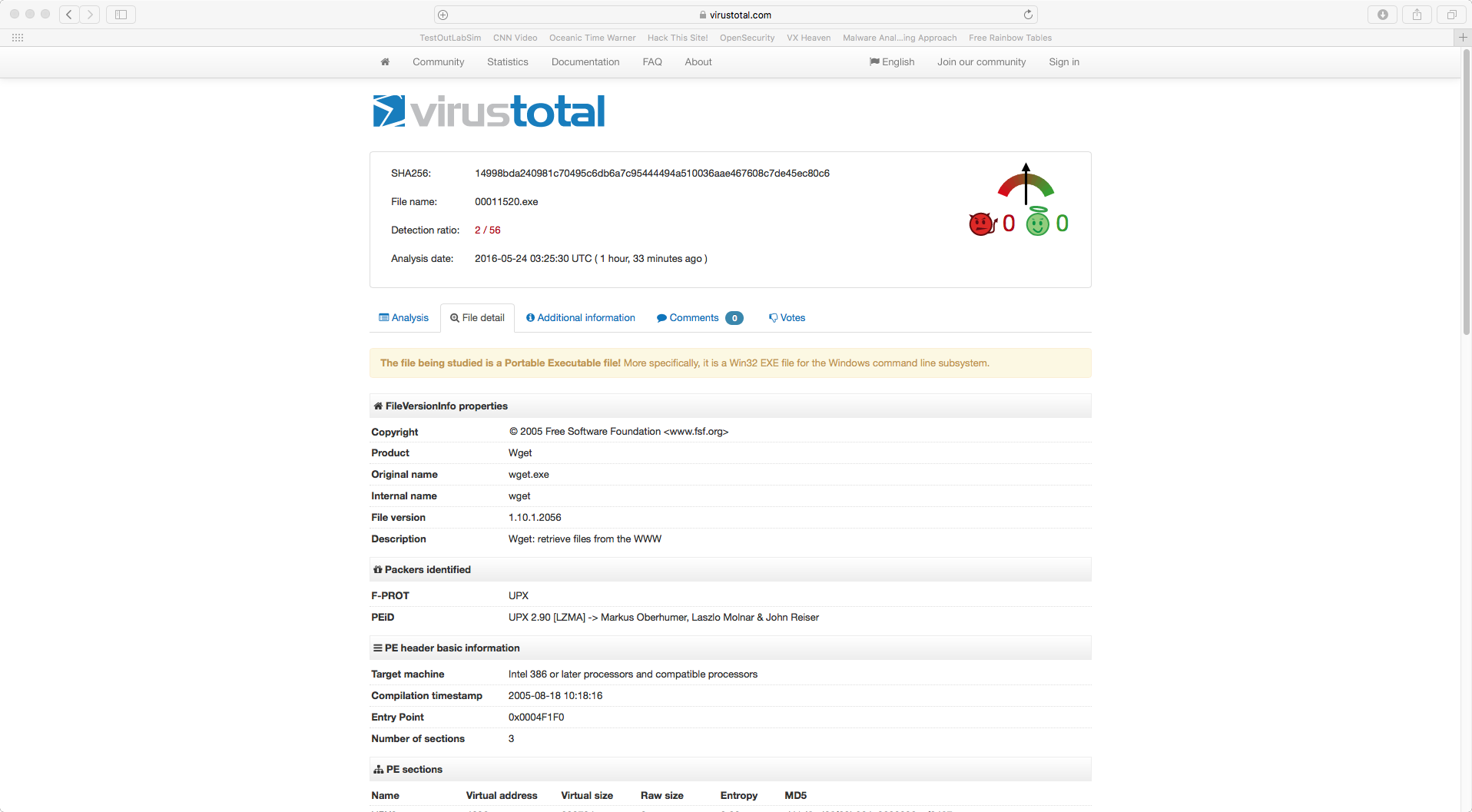
@urls = ( "http://youtube.com/", "http://www.google.com/search?hl=en&q=pig+latin", "http://www.idioma-software.com/pig/pig\_latin.html", "http://www.yahoo.com/", "http://mail.yahoo.com/", "http://www.myspace.com/", "http://vids.myspace.com/index.cfm?fuseaction=vids.individual&VideoID=23886700", "http://youtube.com/", "http://youtube.com/watch?v=ZiRHyzjb5SI", "http://youtube.com/watch?v=1RUFBGDvsy0", "http://www.google.com/search?hl=en&q=juicy+fruit", "http://www.wrigley.com/wrigley/products/pop\_juicy\_fruit.asp", "http://www.amazon.com/Juicy-Fruit-Mtume/dp/B0000025UL", "http://www.facebook.com/", "http://www.live.com/", "http://search.live.com/results.aspx?q=hurricane", "http://www.ebay.com/", "http://books.ebay.com/", "http://photography.ebay.com/", "http://crafts.ebay.com/", "http://en.wikipedia.org/wiki/Main\_Page", "http://en.wikipedia.org/wiki/Lee\_Smith\_\%28baseball\_player\%29", "http://en.wikipedia.org/wiki/Lee\_Smith\_\%28baseball\_player\%29&action=edit", "http://www.msn.com/", "http://www.slate.com/id/2179838/?GT1=10733", "http://mail.live.com/", "http://costarica.en.craigslist.org/rfs/", "http://costarica.en.craigslist.org/apa/");

The proxy server may be Mr Vogon‘s accomplice, and a quick WHOIS look up shows it’s from Telekom Malaysia. When a file is given to the program, it is chopped up, encoded with bas64, added to the cookie field in a http get requests, and sent via Wget. Since the script first sets up a proxy, all http requests with the file chunk can be sent to and reassembled at the proxy. In short, this program can be used to stealthily upload files through http tunneling. The developer doesn’t seem to be Steve Vogon because there was a lot of mocking of him in the code.

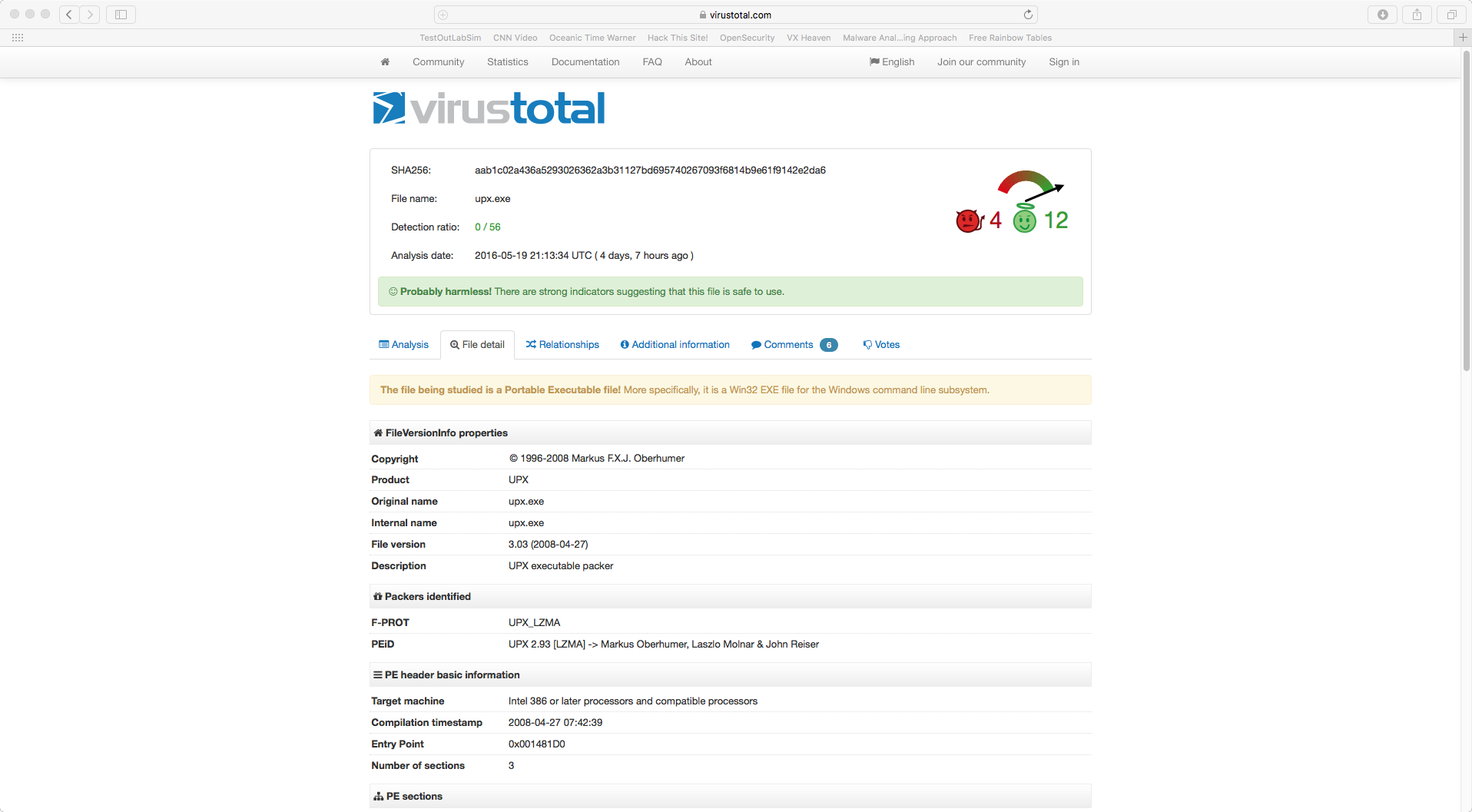
Wave file

The 00000528.wav is a one second audio file that says “Key file”, and end. Running exiftool on this file return no useful information other than it was encoded with Microsoft PCM.

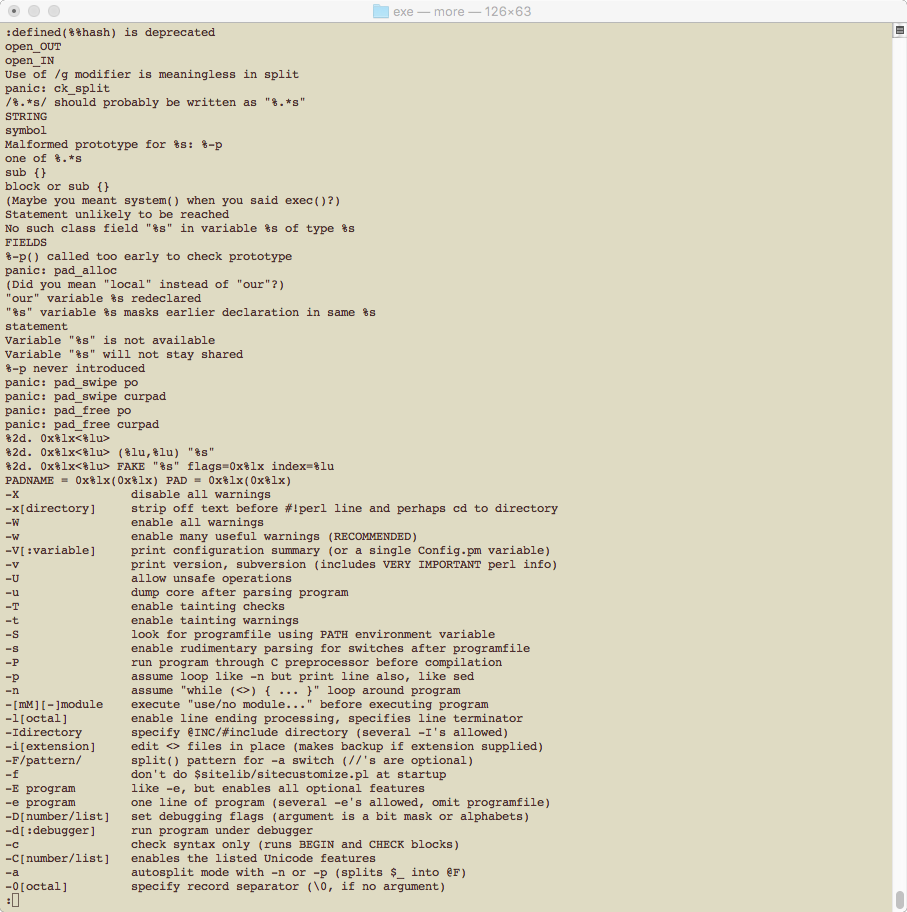
EXE



The 00011520.exe is a Wget executable packed with UPX confirmed by Virustotal.com



The 00016188.exe is UPX executable, packed with UPX, and confirmed by Virustotal.com



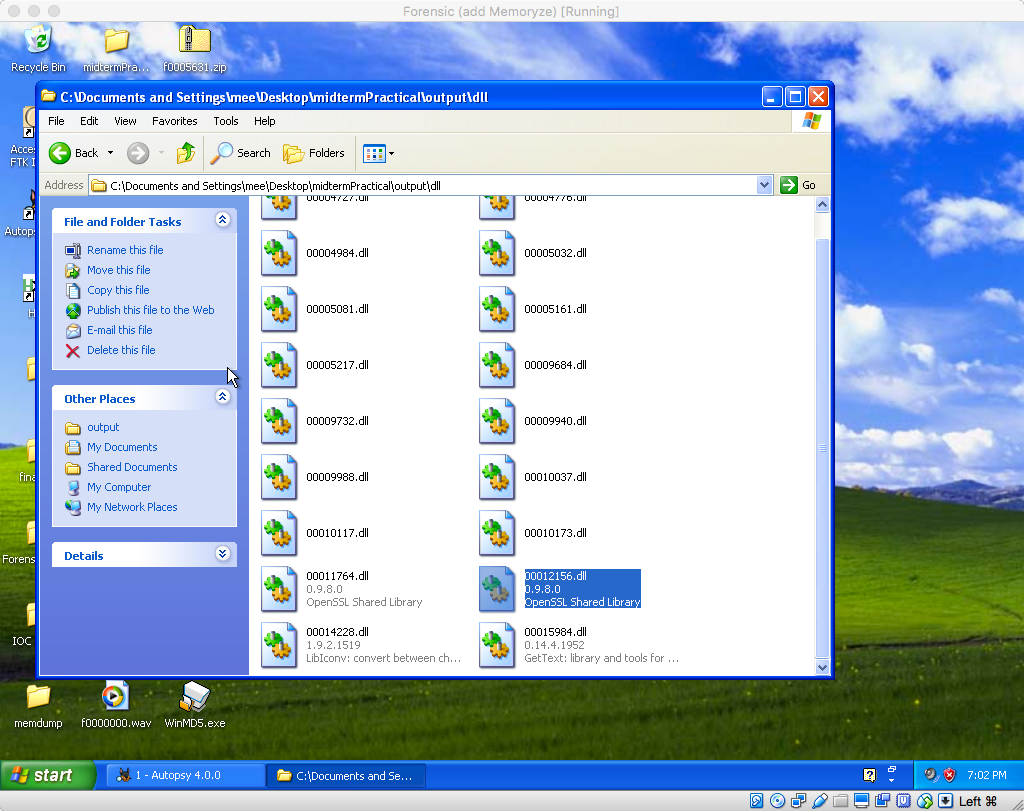
The executable 00006564.exe has a Perl interpreter inside it. Recall that a script packed executable are stand alone script program that is executable by itself due to having interpreter and dependencies included in its package. When run Strings on the executable, I found a help menu typical to Linux command line applications that matches exactly as in Perl interpreter (perl --help) The 00002636.exe is also a 32 bit executable but was packed with UPX. After unpack with the UPX application, I ran it through the Strings command and found the same signature as the 00006564.exe. Just to be 100% sure they are both Perl PAR EXE, a quick search for signature ‘PK\003\004’ according to CPAN (2006) on an unpack version of 00002636.exe and 00006564.exe returned positive results

xxd ./ 00006564.exe | grep ‘PK\003\004’

xxd ./ unpack2636.exe | grep ‘PK\003\004’

If I look back at the file names in psid.db earlier, many of the file names matches the EXE I have just examines. The svxfer.exe and upxxfer.exe is definitely the Perl EXE with one being packed with UPX and one not. The zip files may be original copies of the Perl PAR program. Ups.exe, wget.exe, keyfile.wav are obvious.

DLLs



In windows, some of the DLLs the OS recognizes are such as 0012156.dll and 0001174.dll: “ssleay32.dll”, 00014228.dll: “libiconv2.dll”, and 00015984.dll: “GetText Library”

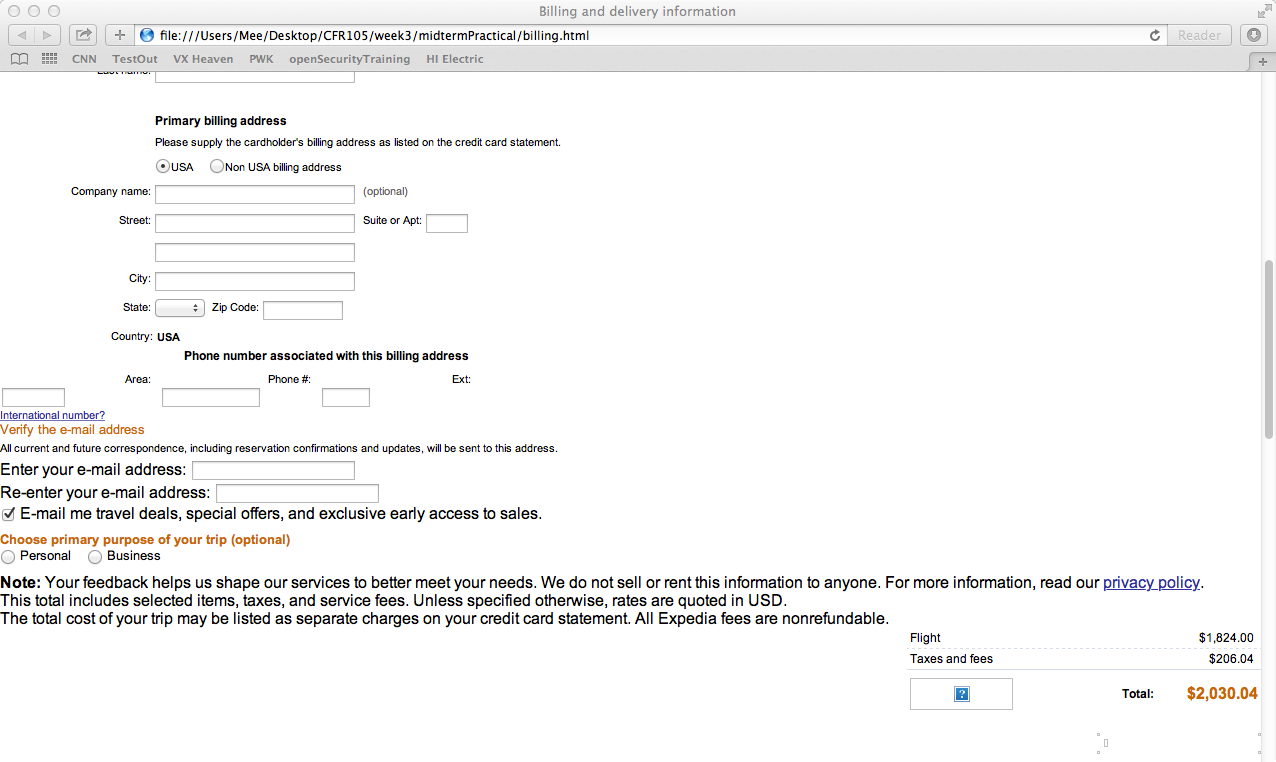
HTML

There was no finding of a html file, however since I was able to find most files in the psid.db, I was sure that the Billing and delivery information.html file does exist. So I tried searching in the image again by first searching for the file name in the html content. Then move to locate the offset in the image, determine the file size, and extract.

strings ./Vogon\_thumbdrive.img | grep 'Billing and delivery information'

xxd ./Vogon\_thumbdrive.img | grep '<TITLE>'

dd if=./Vogon\_thumbdrive.img bs=512 skip=16726 count=222 | strings > billing.html



It seems that the 'Billing and delivery information.html file is a receipt for a plane ticket to somewhere quite far with a price tag of around $2,000. The top hotel near destination link says Benalmadena Hotels. A quick Google Map shows the destination as Spain. So, Mr Vogon is definitely planning for a trip. The last file is secret, however, I could not find or recover it.

Answers for the investigation

1. From initial analysis, there are strong evidences of intentions to conceal the content of the disk. The two FAT tables and root directory entries were wiped clean, however, the 3 bytes signature at the beginning of both FAT were in tack. From my experiments, Mr. Vogon must have tried to hide data by formatting the partition with mkfs.msdos command.
2. The fseventsd and VolumeConfig.plist was common on OSX, which means Mr. Vogon must be using an OSX on the drive, before reformatting it. The psid.db shows some of the original file names that were on the drive. The rest of the important files have been discussed earlier.
3. I believe Mr. Vogon had help from a third party and was trying to move some data out of the company using very stealthy http tunneling technique.
4. From the image, I could not verify if Mr Vogon did ran the malicious executable, however a review of firewall logs or packet captures for destination proxy of 219.93.175.67 or destination for the previous listed URLs can verify if he did ran the executable. Additionally, I recommend a full forensic exam on his workstation to find any trace of the executable, and deny any leave or vacation request from Mr. Vogon during this period.

Mr Vogon definitely had the intention and capability to steal his company’s information. He stored malicious executable on the thumb drive along with all the necessary programs. He also took the time, precaution and effort to hide the presents of everything on the disk. Although I was not able to prove that he did ran the malicious program, I’m sure there is enough evidence to prove that he had violated the company’s acceptable computer use and in the process or have committed a crime.

References

Perl Training Australia (2014), Building executables with PAR

http://perltraining.com.au/tips/2008-05-23.html

Macosxhine (2009), Index only certain directories in Spotlight

http://hints.macworld.com/article.php?story=20090914124459567

hostilefork.com (2009), .Trashes, .fseventsd, and .Spotlight-V100

http://blog.hostilefork.com/trashes-fseventsd-and-spotlight-v100/

CPAN (2006), PAR::Tutorial

http://search.cpan.org/dist/PAR/lib/PAR/Tutorial.pod

Carrier (2005), File System Forensic Analysis

Maverick-os, FAT16 File System

http://www.maverick-os.dk/FileSystemFormats/FAT16\_FileSystem.html