Mandiant Memoryze with Audit Viewer

By Russ McRee – ISSA member, Puget Sound (Seattle), USA chapter

Prerequisites

Python 2.5 or 2.6 and the wxPython library for Audit Viewer

Audit Viewer for Memoryze XML results

Windows 2k, 2k3, or XP (Vista and Windows 7 support pending)

Similar Projects

F-Response (commercial)

RAPIER (free)

MEMORYZE™

In the endless effort to improve malware analysis methodology it's not very often that a new tool comes along that sparks everyone's interest, but Mandiant's Memoryze is just such a tool.

You may recall we discussed Mandiant's Red Curtain in December 2007's *toolsmith*.[1]

As per our exploration of Red Curtain, Mandiant kindly provided useful feedback regarding Memoryze, including plans for further development and successful use of the offering to date.

Memoryze™ is the memory forensic engine from Mandiant Intelligent Response™, released as a free stand-alone tool. Memoryze extends the capabilities of an analyst working on multiple systems or memory images. Imagine using Memoryze to conduct deep-memory forensics on thousands of machines at a time resulting in easily searchable results useful for discovering malicious activity lurking across your enterprise.

Future enhancements to Memoryze will include import and export enumeration for each process it finds in memory. Memoryze will also be improved to analyze the data it collects to find malware. While Memoryze currently enumerates all processes and DLLs within a given process; it offers no intelligence, inherent to Memoryze, to identify malware other than that of the analyst. Peter Silberman's work, as offered at OpenRCE.org,[2] and recently presented at Hack in the Box,[3] describes where he could identify injected shellcode with the help of Memoryze. According to Mandiant, they intend to tune algorithms useful in such detection, better quantify the false-positive rate, and then incorporate these algorithms into Memoryze.

Memoryze offers additional functionality that has not been discussed much to date. Memoryze can utilize XPath filters and apply them to the data it collects. This feature allows users to create their own *evidence of compromise* filter and supply it to Memoryze as part of a script. Using the filter, Memoryze will only report processes that match the criteria, thus limited data analysis overload during large enterprise searches.

The Memoryze engine has been used as part of Mandiant's Intelligent Response product across thousands of systems at various Fortune 500 companies. Mandiant consultants have used Memoryze functionality to investigate advanced, persistent attackers who employ techniques that are often difficult to find using traditional forensic techniques. Seeking malware and other attack signatures via their memory characteristics as an executing process has allowed investigators to find malicious behavior in many circumstances where traditional methods would have failed (for example, identifying process-injected malware that had no disk/file footprint).

Mandiant has shared Memoryze with several malware reverse engineering labs who, in turn, have been impressed with Memoryze's ability to capture most, if not all, of a packed binary from memory in its unpacked form. Memoryze has proven especially effective against Themidia, which some malware has used as a protection mechanism.

While Memoryze currently only runs on Windows 2000 SP4, Server 2003 SP2 and Windows XP, in 2009 Mandiant will be releasing Vista support. Windows 7 is also on the road map, and support for it will follow Microsoft's release plans.

Using Memoryze and Audit Viewer

Installation

Installation is a point and click process. Mandiant tools typically install to C:\Program Files\Mandiant which is logical enough. When you unzip the Audit Viewer package, move it to the Mandiant directory. In a perfect world you then have Red Curtain, Memoryze, and Audit Viewer all in one directory with a shortcut to it on your malware analysis virtual

1 http://holisticinfosec.org/toolsmith/docs/december2007.pdf.

2 http://www.openrce.org/articles/full_view/32.

3 https://conference.hackinthebox.org/hitbsecconf2008kl/materials/D1T1%20-%20Peter%20Silberman%20-%20Full%20Process%20Reconstitution%20from%20Memory.pdf.

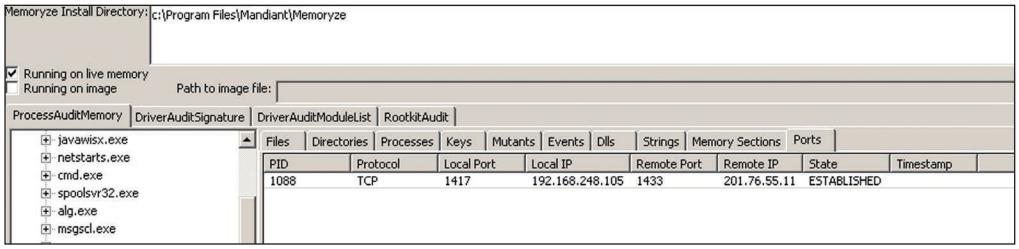

Figure 1 – Memoryze Process.bat shows a SQL connection

machine. Be sure to read both *MemoryzeUserGuide.pdf* and *AuditViewerUserGuide.pdf* included in the install directories; they both bring you up to speed quickly.

Getting down to business

Memoryze can be run on a live system or can be used to analyze memory images taken from other machines. It's ideal for malware analysis; if you know what you're looking for, the malicious tidbits will jump right out at you. For this discussion, I infected my malware analysis VM with Win32/Malushka.B (MD5: 152d44ebce29afaf6bcca9c1d4927514), as it offers a number of elements useful in amplifying Memoryze's capabilities. From here out though, I treated the process as if I had no idea what the root cause consisted of.

I started analysis with one of the more simple approaches to sniffing out evil with Memoryze; I executed `process.bat –ports true` after infecting my VM. Memoryze includes a number of batch scripts that pass the appropriate parameters for the task at hand. Examples include:

- **Process.bat** to enumerate everything about a process including handles, virtual memory, network ports, and strings
- **HookDetection.bat** to look for hooks within the operating system
- **DriverSearch.bat** to find drivers

Once the audit driven by Process.bat completed, I opened Audit Viewer and navigated to the appropriate output directory. Keep in mind that Audit Viewer is Python dependent; you'll need Python 2.5 or 2.6 and the wxPython library before you can run AuditViewer.py. Of immediate note were a number of processes found in the ProcessAuditMemory view that were certainly out of place on a healthy system. I'm quite certain that javawins.exe, javawisx.exe, netstarts.exe, msgscl.exe, msgsdb.exe, and spoolsvr32.exe have no business on a clean Windows XP instance. I am also quite certain that said Windows XP instance should not be making a SQL connection to a server in Brazil. Hmm…smells a bit like a Trojan.Banker variant (See Figure 1).

As mentioned earlier, Mandiant's Peter Silberman offers a useful discussion of Memoryze on OpenRCE.org; it's a great starting point, and as such, doesn't require me to reinvent the wheel here. The most important thing to keep in mind about Memoryze use is that it centers around XML documents that script its actions, and output is written to XML that is then made human readable by Audit Viewer.

Peter offers what he calls AllAudits.Batch.XML, a custom audit configuration that is very useful because it runs a full process audit including ports, handles, sections, and PIDs, as well as a driver signature scan and rootkit detection. Where I differ slightly from Peter's perspective is with regard to enabling strings analysis. He's correct about the amount of data it generates (it's quite a bit), but if your VMWare server has enough horsepower and your malware analysis VM can churn the data in relatively short order, I suggest trying AllAudits.Batch.xml with `<param name="strings">` set to `true`. If Audit Viewer grinds too slowly parsing the output simply change it to `false`, and it won't run strings against all processes. I suggest a 2GB RAM minimum on your host operating system and a guest VM memory configuration of 1GB RAM.

I created a copy of AllAudits.Batch.xml available for download here: http://holisticinfosec.org/toolsmith/files/memoryze/AllAudits.Batch.xml. Go to *Page*, then *Save As* in Internet Explorer or *File*, then *Save Page As* in Firefox, to save AllAudits.Batch.xml on your local system. Again, thanks to Peter Silberman, this is entirely his work. To execute Memoryze using the audit file, execute

```
Memoryze.exe -o -script AllAudits.Batch.xml -
  encoding none
```

The `-o` flag ensures that the output is written to `C:\Program Files\Mandiant\Audits\<computername><DateTime>`, assuming you accepted defaults during install. You can then navigate to this directory via Audit Viewer and open your just completed audit.

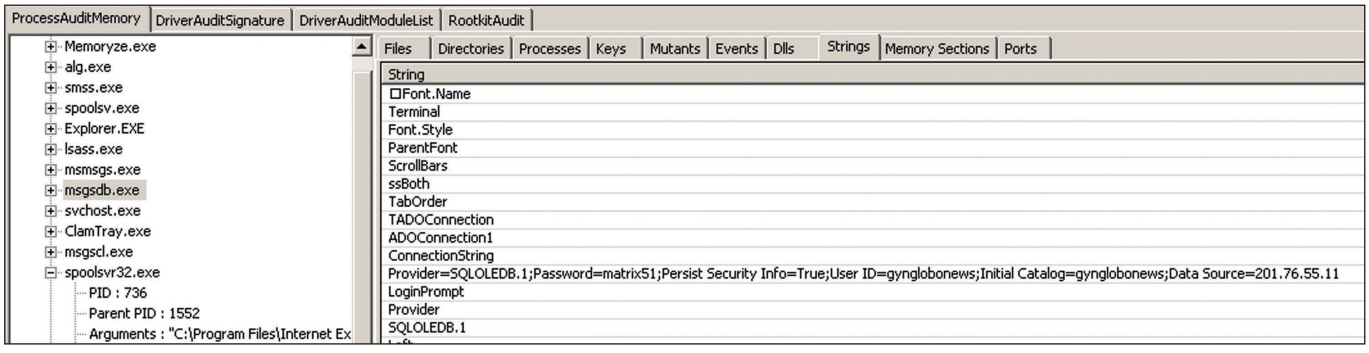

Figure 2 – Memoryze Strings results reveals all

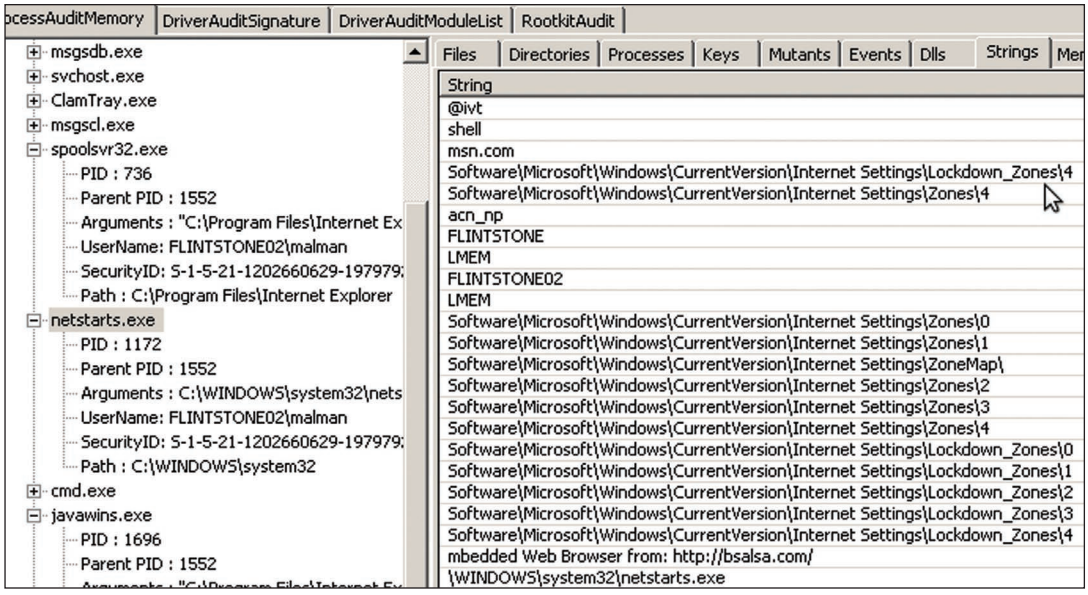

Figure 3 – Memoryze unveils more evil

Here's where the real fun begins. Now that I'd audited far more than just ports used, I learned a great deal more about some of the passengers the infection brought along for the ride. A closer look at the strings output for msgsdb.exe revealed an awful lot about the badguy's backend; specifically, the entire connection string inclusive of the username, password, and data source (Figure 2).

Strings analysis of netstarts.exe also revealed a reference to bsalsa.com, where malware authors grab useful embedded web browser components (figure 3).

With that in mind I decided to "Acquire Process" from netstarts.exe by right-clicking on it. One very cool tidbit I discovered when doing this is that it will grab the .exe and drop it in your output directory. So all 896kb of netstarts.exe immediately yielded an Md5 hash of 2aea435bc3d16922ef3c4bf3e55c4307 (determined with an unrelated tool) which didn't appear in any online search results. I submitted the Memoryze-extracted binary to Virus Total and guess what? Remember when I said the initial analysis of the infection had the stink of Trojan.Banker? (Figure 4)

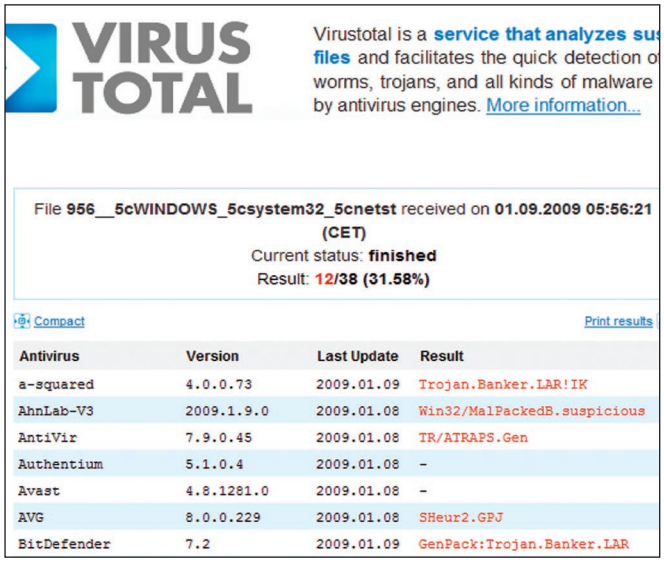

Figure 4 – VirusTotal confirms suspicions

This nasty little bugger intends to steal my banking passwords! Too bad Memoryze helped me uncover the password to his database. I might suggest enhancing password complexity. Matrix51? Please.

In addition to the too few elements I've discussed, Memoryze will cough up Files, Directories, Processes, Keys, Mutants, Events, Dlls, and Memory Sections, as driver and rootkit details if relevant.

Check out Events for sure. Most of the malicious binaries dropped on my VM showed an associated event of crypt32LogoffEvent, behavior very common to malware.

Memoryze resources

Be sure to review Peter Silberman's Hack In The Box[4] presentation, as well as the OpenRCE article. Mandiant's user guides, as well as their *Memoryze - Use Cases and Examples.*[5]

In conclusion

I haven't been this excited about a new tool in awhile. I'll be preaching about Memoryze to everyone who cares to hear my sermon. Add it to your toolkits with every confidence that it will be of significant and immediate use to you. Enjoy, but be careful out there. ;-)

Cheers…until next month.

Acknowledgments

—Anne Mroczynski of Mandiant for coordinating feedback from project developers.

—Peter Silberman of Mandiant for allowing me to make use of his AllAudits.Batch.xml file, as well as all the useful reference material in his OpenRCE article.

About the Author

Russ McRee, GCIH, GCFA, CISSP, is a security analyst working in the Seattle area. As an advocate of a holistic approach to information security, Russ' website is holisticinfosec.org. Contact him at russ@holisticinfosec.org.

4 Ibid.

5 http://www.mandiant.com/software/usememoryze.htm.