

Global Information Assurance Certification Paper

Copyright SANS Institute Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering "Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics at http://www.giac.org/registration/gcfa



Global Information Assurance Certification Paper

Copyright SANS Institute Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering "Advanced Computer Forensic Analysis and Incident Response (Forensics 508) at http://www.giac.org/registration/gcfa

retains full rights Indicators of compromise in memory forensics

GIAC (GCFA) Gold Certification

Author: Chad Robertson, chad@rocacon.com

Advisor: Tim Proffitt

Accepted: TBA

Abstract

Utilizing memory forensics during incident response provides valuable cyber threat intelligence. By both providing mechanisms to verify current compromise using known indicators and to discover additional indicators, memory forensics can be leveraged to identify, track, isolate and remediate more efficiently. .0. 6 2013 SAN

1.0 Introduction

There has been a recent increase in the availability of intelligence related to malware. New IP addresses, hostnames, MD5 hashes, mutex values, and other attacker artifacts are shared often. Historically, there exist many host-based and network-based standard methods to utilize these artifacts, such as intrusion detection/prevention systems, firewalls, anti-virus, and file whitelisting. While these solutions provide various benefits, they can fall short of confirming an infection.

Consider antivirus for example. In an article published June 11th, 2012, MIT Technology Review boldly proclaimed "The Antivirus Era Is Over". Mikko Hypponen, Chief Research Officer of F-Secure, wrote that it was "a spectacular failure of our company, and for the antivirus industry in general" (Hypponen, 2012) that they had possessed samples of Flame, one of the most complex pieces of malware ever discovered, for at least two years without examining it closely and developing detection mechanisms. (Hypponen, 2012) Later, in the same article, Mr. Hypponen states, "The truth is, consumer-grade antivirus products can't protect against targeted malware created by well-resourced nation-states with bulging budgets. They can protect you against run-ofthe-mill malware: banking trojans, keystroke loggers, and e-mail worms. But targeted attacks like these go to great lengths to avoid antivirus products on purpose. And the zero-day exploits used in these attacks are unknown to antivirus companies by definition." (Hypponen, 2012)

SC Magazine's Tom Cross states, "Although basic controls like anti-virus will always have a place in the security arsenal, they are not up to the task of defending networks against sophisticated, targeted attacks." (Cross, 2012) "It is going to take human analysts to recognize the subtle and often unpredictable patterns of evidence that sophisticated attacks leave behind. Therefore, the best strategies are going to focus on arming incident responders with the tools that they need to monitor their environments and actively hunt for active attack activity." (Cross, 2012)

"So what's the next-generation solution? The future of security lies in shifting toward behavior-oriented scanning, says Dennis Pollutro, president and founder of cloud security vendor Taasera. While "there will always be a place for signatures," security products have to begin identifying malware by what it's doing, rather than what it looks like, he says." (Rashid, 2012)

"Several things have to happen before the malware infection results in damage or data theft on the compromised computer, which gives defenders a "couple hundred processes" to monitor for, Pollutro adds. Threat intelligence allows administrators to recognize patterns of behavior, such as creating directories on a file system or communicating with an IP address that had previously been flagged as suspicious." (Rashid, 2012)

Malware can be identified by seeking out common TTP's (tools, techniques, and procedures) used during development and infection. These relationships help analyst gain a better understanding of the adversary and develop attacker profiles. From the profiles we can draw inferences to better adapt and respond to attacks.

Categorization techniques vary among researchers. David French, Senior Malware Researcher at CERT suggests a malware-centric approach. "To express such relationships between files, we use the concept of a "malware family", which is loosely defined as "a set of files related by objective criteria derived from the files themselves." Using this definition, we can apply different criteria to different sets of files to form a family." (French, 2012)

Michael Cloppert, a senior member of Lockheed Martin's Computer Incident Response Team suggests taking a more adversary-centric approach. "The best way to behaviorally describe an adversary is by how he or she does his job...that "job" is compromising data, and therefore we describe our attacker in terms of the anatomy of their attacks." (Cloppert, 2009). "We as analysts seek the most static of all indicators [those closest to the adversary] but often must settle for indicators further from the adversary until those key elements reveal themselves." (Cloppert, 2009)

According to Greg Hoglund, Founder of HBGary, "It makes no sense to separate the human from the malware and TTP's. They are two ends of the same spectrum. This is not a black and white science; it works because humans aren't perfect. It works because humans are creatures of habit and tend to use what they know. They use the same tools every day and don't rewrite their malware every morning. They don't have perfect OPSEC. They put their digital footprints out on the Internet long ago – and it's usually just a few clicks away from discovery. There is a reflection of the threat actor behind every intrusion. To discount this is to discount forensic science." (Hoglund, 2011)

According to The MITRE Corporation, "to be proactive, cyber defenders need to fundamentally change the nature of the game by stopping the adversary's advance, preferably before the exploit stage of the attack [which] requires defenders to evolve from a defensive strategy based primarily on after-the-fact incident investigation and response to one driven by cyber threat intelligence." (The MITRE Corporation, 2012)

Identification of singular characteristics of malware helps to automate future identification tasks. Often malware from the same family share these characteristics, which helps analysts, identify related files. These characteristics can be used by analysts to organize malware within repositories, root out additional infections present on the network, or identify new variants of malware.

Memory forensics allows analysts to "reconstruct the state of the original system, including what applications were running, what files those applications were accessing, which network connections were active, and many other artifacts" (Michael Ligh, 2010). Many free tools are available to analyst for local memory imaging as well as several enterprise solutions for remote imaging. While memory forensics shouldn't be considered a replacement for any other security technology, it is most certainly a valuable tool and should be considered, as is stated in the Malware Analyst's Cookbook, "extremely important to incident response" (Michael Ligh, 2010). This paper discusses analyzing malware. It does not describe designing a safe and effective malware analysis environment. If you are new to malware analysis and wish to perform any of the examples within this paper please spend some time researching how to do so safely.

2.0 Scope and Assumptions

The research discussed herein focuses on the creation of signatures to help speed up memory analysis during incident response. Because they are not meant to be used as host-based indicators deployed uniformly there is a greater tolerance for false positives. If, instead, the task was to develop signatures to deploy to all client machines then there would be a much greater need to tune each rule since, in that case, any false positive could have an immediate impact on the client experience. For this research, the resulting signatures will be run by the analyst, for the analyst, to expedite the identification, containment, and eradication of threats.

We will touch on many topics related to malware analysis and memory forensics. To limit the scope of this research, specific malware analysis techniques will not be discussed. We will rely on various publicly available materials from which to understand our topic. If malware analysis is of interest to the reader, they are encouraged to see the reference section for the articles cited and the appendix for a list of additional resources.

There are many tools mentioned throughout this paper. Detailing each tool is out of scope, but links to the tools mentioned will be included within the appendix for reference.

3.0 Finding Indicators

Indicators can come from a variety of sources. There are numerous blogs dedicated to malware reversing and analysis. "Malwaremustdie", "Joe Security LLC", and "Hooked on Mnemonics worked for me" to name a few (see Appendix 1). There are also many security companies that publish some of their own internal findings publicly. Organizations such as Mandiant, Fireeye, and Dell Secureworks are great sources of detailed reports that include indicators.

Another option is to acquire malware samples for analysis. Enterprising and ambitious analysts could mine the internet for malware themselves. There are a variety of sites (See Appendix 1) that note active malware sources that can be downloaded manually or scrapped via script (such as maltrieve - See Appendix 1) to automate collection. Alternatively, one could setup a honeypot (such as Dionea - See Appendix 1) to glean current samples from the internet by allowing attackers access to seemingly vulnerable services.

Another approach is to turn to the numerous websites that provide users access to malware. Sites such as virusshare.com, malwr.com, malware.lu, offensivecomputing.net, and contagiodump offer access to millions of samples (See Appendix 1).

3.1 YARA

To help aid in scanning memory samples for indicators of compromise, we will use a tool called YARA. "YARA is a tool aimed at helping malware researchers to identify and classify malware samples. With YARA you can create descriptions of malware families based on textual or binary patterns contained on samples of those families. Each description consists of a set of strings and a Boolean expression which determines its logic." (YARA in a nutshell, 2013)

YARA provides a mechanism to match indicators such as strings and hexadecimal within memory samples.

3.2 Indicators Defined

Michael Cloppert wrote a phenomenal blog post in 2009 on the computer forensic blog on SANS called *Security Intelligence: Attacking the Kill Chain*. In it he describes classifying indicators into one of three categories: atomic, computed, and behavioral. Michael's description of each is shown here:

"Atomic indicators are pieces of data that are indicators of adversary activity on their own. Examples include IP addresses, email addresses, a static string in a Covert Command-and-control (C2) channel, or fully-qualified domain names (FQDN's). Atomic indicators can be problematic, as they may or may not exclusively represent activity by an adversary." (Cloppert, 2009)

"Computed indicators are those which are, well, computed. The most common

amongst these indicators are hashes of malicious files, but can also include specific data in decoded custom C2 protocols, etc. Your more complicated IDS signatures may fall into this category." (Cloppert, 2009)

"Behavioral indicators are those which combine other indicators — including other behaviors - to form a profile." (Cloppert, 2009)

As an example of a complex behavioral indicator, consider the following. An adversary is identified that tends to rely on "spear phished" email attachments. The emails tend to come from a specific range of IP space. When the attachment is executed, the dropper malware gains persistence by writing to specific registry keys and then installs a PoisonIvy variant. From there the attacker uses Windows Credential Editor to move laterally within the network. Once interesting data has been identified, Winrar is used to zip up the files and exfiltrate them to a set of IPs. These indicators could have been identified during separate incident response engagements, and may has little significance alone, but when viewed together build behavioral indicators that can be assigned to specific attacker profiles. We will discuss identifying atomic and computed indicators in depth here, and will leave the creation of behavior indicators to the reader.

Indeed, the identification of atomic and computed artifacts is the easiest step in malware analysis. Sources of unique strings, IP addresses, registry keys, file locations, etc. related to malware are numerous. The task of determining the relevance of each accumulated artifact and its contribution to the attacker profile falls upon the analyst team.

3.2.1 Avoiding poor indicators

As we have been discussing, not all indicators are equal. Some artifacts are easily changed by the attacker. For example, consider GhostRat – a popular RAT (remote access tool) used in many recently documented attacks. According to Norton, "the most stable indicator of GhostRat is its network communication. It is well documented and quite distinctive, as it always begins with a "magic word" which in its default configuration is "Gh0st" (Norman, 2012). However, because the source code for Gh0stRAT is freely available, modifying this "magic word" is simple.

7hero, Adobe, B1X6Z, BEiLa, BeiJi, ByShe, FKJP3, FLYNN, FWAPR, FWKJG, GWRAT, GhOst, GOLDt, HEART, HTTPS, HXWAN, Heart, IM007, ITore, KOBBX, KrisR, LUCKK, LURKO, LYRAT, Level, Lover, Lyyyy, MYFYB, MoZhe, MyRat, OXXMM, PCRat, QWPOT, Spidern, Tyjhu, URATU, WOLFKO, Wangz, Winds, World, X6RAT, XDAPR, Xjjhj, agOft, attac, cblst, https, whmhl, xhjyk

Figure 1

Several known values of Gh0stRAT's "Magic word" -- Source: http://download01.norman.no/documents/ThemanyfacesofGh0stRat.pdf

4.0 Writing indicator-based YARA rules

Consider the differences between host-based signatures, such as those used by anti-virus, and incident response rules used for YARA. Because many anti-virus platforms have real-time functionality, they must constantly monitor the file system and memory for both static and heuristic based signatures. Thus, system performance is impacted by the simple addition of anti-virus, regardless of the quality of its signatures. If the signatures written for the anti-virus platform are vague, the resulting additional overhead on the system impacts the user experience at best by simply slowing down processing and, at worse, incorrectly identifying files as malicious, quarantining them, and denying the user access until a patch can be deployed.

YARA signatures, specifically those focused on memory forensics, involve only a snapshot of memory, frozen in time and independent of the host machine, thus there is no real-time performance impact to consider. Also, because incident response engagements typically involve a small number of systems, performing matches for less precise characteristics is tolerated and, in some cases, preferred.

To that end, as we create YARA signatures we will test them against a corpus of malware to assess the false positive rate. We will ask ourselves if, in some cases, defining signatures based upon characteristics that might be considered poor form for anti-virus platforms is more tolerated for memory analysis.

Chad Robertson, chad@rocacon.com

4.2 Python and IDA

To help speed up the process of creating YARA rules based on disassembled malware, we turn to IDA Pro and Python. Within idc.py, the Python plugin for IDA, the o_* constants allow for the identification of variable memory addresses within malware. These addresses require the use of wildcards within YARA. A very useful script to automate the substitution of the variable memory addresses with '??' was written by Case Barnes of Accuvant (Barnes, 2012). His script relies on the following operands, o_mem, o_phase, o_displ, o_far, o_near, which are defined within idc.py:

o_void o_reg o_mem o_phrase o_displ o_imm o_far o_near o_idpspec1 o_idpspec2 o_idpspec2 o_idpspec3 o_idpspec5 o_last	<pre>idaapi.o_void idaapi.o_reg idaapi.o_mem idaapi.o_phrase idaapi.o_displ idaapi.o_idspl idaapi.o_far idaapi.o_idpspec0 idaapi.o_idpspec1 idaapi.o_idpspec2 idaapi.o_idpspec3 idaapi.o_idpspec4 idaapi.o_idpspec5 idaapi.o_last</pre>	****	No Operand General Register (al,ax,es,ds) Direct Memory Reference (DATA) Memory Ref [Base Reg + Index Reg] Memory Reg [Base Reg + Index Reg + Immediate Value Immediate Far Address (CODE) Immediate Near Address (CODE) IDP specific type IDP specific type	reg addr phrase Displacement] value addr addr	phrase+addr
			Figure 2		

Operands within idc.py – Source: Author created

5.0 Demonstrations

To test our ability to identify indicators in one piece of malware, and then use them to create signatures to identify the same or similar malware across multiple memory dumps, we will turn to three pieces of malware; Stabuniq, X0rb0t, and a sample from APT1. Each sample will be assessed for indicators, and then YARA will be used to scan across several memory images to verify the effectiveness of the rule.

5.1 Stabuniq

Stabuniq is not considered APT, but due to its availability and interesting characteristics it is an excellent example for our purpose.

I will refer to indicators identified within two public analysis posts:

5

- 1. http://quequero.org/2013/01/stabuniq-financial-infostealer-trojan-analysis/
- http://www.symantec.com/connect/blogs/trojanstabuniq-found-financialinstitution-servers

According to Symantec, Stabuniq was first spotted in 2011 on servers belonging to financial institutions, banking firms, and credit unions. (Gutierrez, 2012) Symantec published an analysis in December 2012. During that same month, Mila Parkor of Contagio Malware Dump posted links to samples of the malware (see Appendix 1).

The initial executable contains a second executable that is decoded and dropped at runtime. Because our approach cares only about what eventually ends up in memory, the initial binary is of little interest. Therefore, to identify pertinent indicators, we must review the final executable dropped on the system.

After the malware has been run and the subsequent memory image acquired, similar artifacts to those noted within the online analysis can be seen using Volatility. To follow along, we must identify the injected process PID:

Volatile Sy	stems Volatility	Framework 2	.2							
Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start		E>
0x819cc9c8	System	4	Θ	55	233		Θ			
0x8177b020	smss.exe	368	4	3	21		Θ	2013-02-14	21:34:56	
0x81898410	csrss.exe	616	368	10	356	Θ	Θ	2013-02-14	21:34:56	
0x81883b88	winlogon.exe	640	368	24	535	Θ	Θ	2013-02-14	21:34:56	
0x81810c08	services.exe	684	640	22	273	Θ	Θ	2013-02-14	21:34:56	
0x81786da0	lsass.exe	696	640	23	355	Θ	Θ	2013-02-14	21:34:56	
0x817bbda0	VBoxService.exe	852	684	8	107	Θ	Θ	2013-02-14	21:34:56	
0x817742c0	svchost.exe	896	684	18	194	Θ	Θ	2013-02-14	21:34:56	
0x81772da0	svchost.exe	984	684	12	217	Θ	Θ	2013-02-14	21:34:56	
0x817a1778	svchost.exe	1104	684	66	1130	Θ	Θ	2013-02-14	21:34:56	
0x819dd020	svchost.exe	1144	684	5	59	Θ	Θ	2013-02-14	21:34:56	
0x817e3458	svchost.exe	1188	684	13	207	Θ	Θ	2013-02-14	21:34:57	
0x81612da0	spoolsv.exe	1588	684	13	129	Θ	Θ	2013-02-14	21:34:57	
0x8160bda0	explorer.exe	1632	1552	17	408	Θ	Θ	2013-02-14	21:34:57	
0x815ffa10	VBoxTray.exe	1704	1632	7	54	Θ	Θ	2013-02-14	21:34:57	
0x81599698	alg.exe	1404	684	7	103	Θ	Θ	2013-02-14	21:35:08	
0x81752c10	IEXPLORE.EXE	1856	1776	5	123	0	Θ	2013-02-14	21:35:09	
0x815f7020	IEXPLORE.EXE	1868	1856	2	30	Θ	Θ	2013-02-14	21:35:09	
0x8176cda0	wscntfy.exe	352	1104	1	27	Θ	Θ	2013-02-14	21:35:09	
0x815e6020	DumpIt.exe	1836	1632	1	24	Θ	Θ	2013-02-14	21:35:28	

Figure 3

Injected processes -- Source: Author created

The malware uses StabililtyMutexString as its mutex. We can confirm that by using the Volatility mutantscan plugin.

Volatile S	ystems	s Vola	atility	Framework 2	2.2		
Offset(P)	#Ptr	#Hnd	Signal	Thread	CID	Name	
0x017abd68	2	1	1	0×00000000		WindowsUpdateTracingMutex	
0x017b0908	1	1	1	0×00000000			
0x017d8578	3	2	1	0×00000000		StabilityMutexString	
0x017db1d8	1	1	1	0×00000000			
0x017de240	1	1	1	0×00000000			
0x017e8480	1	1	1	0×00000000			
0x017e8b00	1	1	1	0×00000000			
0x017f7dd8	1	1	1	0×00000000			
0x01802718	1	1	1	0×00000000			
					Figure 4	× 0	

Output from Volatility's mutantscan plugin -- Source: author created

To create a YARA rule to identify this sample by this mutex, we must consider how the data looks within memory. One way to see is to use volshell to display the contents of memory: 10.1

Volatile Systems Volatility Framework 2.2																										
Current context: process System, pid=4, ppid=0 DTB=0x311000																										
Welcome to	<pre>/elcome to volshell! Current memory image is:</pre>																									
file:///opt	ile:///opt/cuckoo/shares/Windows-XP/WINDOWSXP-stabuniq.raw																									
o get help, type 'hh()'																										
->> cc(pid1856)																										
Traceback (most recent call last):																										
File " <co< td=""><td>nsol</td><td>le>'</td><td>٦, "</td><td>line</td><td>e 1,</td><td>, ir</td><td>ז> ר</td><td>nodu</td><td>ule></td><td>></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></co<>	nsol	le>'	٦, "	line	e 1,	, ir	ז> ר	nodu	ule>	>																
NameError:	name	∋ 'k	pid	1856	5' i	is r	not	det	fine	∋d																
>>> cc(pid=	1856	5)																								
Current con	text	t: p	orod	ces	s IB	EXPL	OR	E.E>	ΚE,	pi	d=18	356,	, pp	pid=	=177	76 DTE	3=0	x6	b۷	10	26	60				
>>> db(0x00)	1518	3b6))																							
0x001518b6	53	00	74	00	61	00	62	00	69	00	6c	00	69	00	74	00	S.	t.	a.	b	.i		ι.	i.	.t	
0x001518c6	79	00	4d	00	75	00	74	00	65	00	78	00	53	00	74	00	у.	М.	u.	t	.е	÷.	х.	S	.t	
0x001518d6	72	00	69	00	6e	00	67	00	00	00	00	00	00	00	00	00	r.	i.	n.	g						
0x001518e6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00										
0x001518f6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00										
0x00151906	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00										
0x00151916	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00										
0x00151926	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			• •							
>>>																										

Figure 5

The mutex as stored in process memory -- Source: author created

As you can see above, the string is encoded with two bytes per character. The 'wide' YARA modifier must be used:



Figure 6

YARA rule to match Stabuniq's mutex - Source: author created

The signature is checked and verified to match:

Stabuniq_strings [] /usr/share/memory_dumps/public/WINDOWSXP-stabuniq.raw Figure 7

Results shows match - Source: author created

Because the mutex name is likely easy to change, we return to the quequero.org analysis for a preferred indicator. The analysis also mentions the segment of code used to re-injecting Internet Explorer. Perhaps this function would be more difficult for the malware author to change and thus make a better signature.

.text:0040802B	call	dword ptr [eax+0ACh]:
.text:00408031	mex	[ebp+var_8], <u>eax</u>
.text:00408034	CMP	[ebp+var_8], 0
.text:00408038	inz	short loc_408071
.text:0040803A	max	<pre>[ebp+var_4], offset injectedThread</pre>
.text:00408068	mer	<pre>ecx, [ebp+arg_0]</pre>
.text:0040806B	push	ecx
.text:0040806C	call	createRemoteIEThread
.text:00408071		
.text:00408071 loc 408071:		; CODE XREF:
threadReinjectIE+30 [†] j		
.text:00408071	push	1000
.text:00408076	max	<pre>edx, [ebp+arg_0]</pre>
.text:00408079	call	dword ptr [edx+0A8h] Sleep
.text:0040807F	imp	short loc_40800E

Figure 8

Interesting code block from Stabuniq - Source: http://quequero.org/2013/01/stabuniqfinancial-infostealer-trojan-analysis/



Using the Python script mentioned above, we use this bit of code to create a YARA rule:

Figure 9



While process could certainly be done manually, the script automatically inserts wildcards where needed to adjust for memory addressing.

© 2013

To validate the rule, I will use various publicly available memory images, including all ins full rights. those linked to from the Volatility website. I will also include various known-infected memory images acquired during research. These memory images are shown below:



Figure 10

13 SANS Sample memory images - Source: author created

The results are shown here:



Stabuniq YARA signature matched - Source: author created

As you can see, the rule matched the two Stabuniq-infected memory dumps, but none of the other dumps in the folder. To further test the resilience of the rule, we can extract the malware from memory by using Volatility's malfind command:

Process:	IEXPLORE.EXE Pid	: 1856 Address: 0x160000	
Vad Tag:	VadS Protection:	PAGE_EXECUTE_READWRITE	1.64
Flags: Co	ommitCharge: 7, M	emCommit: 1, PrivateMemory: 1, Protection: 6	
0×0016000	00 55 8b ec 83 e	c 14 53 56 57 e8 00 00 00 00 5b 81 USVW[.	
0x0016001	10 eb 6e 2b 40 0	0 89 5d fc 8b 45 08 8b 4d fc 89 88 .n+@lEM	
0x0016002	20 14 02 00 00 c	7 45 f4 00 00 00 00 8b 55 f4 6b d2EU.k.	
0×0016003	30 28 8b 45 08 3	3 c9 66 8b 8c 10 32 12 00 00 85 c9 (.E.3.f2	
0×160000	55	PIISH FRP	
0x160001	8hec	MOV FRP FSP	
0x160001	83ec14	SUB FSP 0×14	
0x160006	53	PUSH FRX	
0x160007	56	PUSH EST	
0x160008	57	PUSH EDT	
0x160009	e800000000	CALL 0x16000e	
0x16000e	5b	POP EBX	
0x16000f	81eb6e2b4000	SUB_EBX. 0x402b6e	
0x160015	895dfc	MOV [EBP-0x4], EBX	
0x160018	8b4508	MOV EAX. [EBP+0x8]	
0x16001b	8b4dfc	MOV ECX, [EBP-0x4]	
0x16001e	898814020000	MOV [EAX+0x214], ECX	
0x160024	c745f400000000	MOV DWORD [EBP-0xc], 0x0	
0x16002b	8b55f4	MOV EDX, [EBP-0xc]	
0x16002e	6bd228	IMUL EDX, EDX, 0x28	
0x160031	8b4508	MOV EAX, [EBP+0x8]	
0x160034	33c9	XOR ECX, ECX	
0x160036	668b8c1032120000	MOV CX, [EAX+EDX+0x1232]	
0x16003e	85c9	TEST ECX, ECX	

Figure 12

IEXPLORE.EXE injected process #1 - Source: author created

Process	TEV		DE	EVE	= р;	d.	186	8 /	dd	roce	6	0∨16	5000	00										
Vad Tagi			Drot			.u. 	TOO				DE/			=										
Flage Co	vac mmi	13 r i+Ch		10.	7	и. г Мот		∟⁄ mit		יר <u>-</u> ו נ	_n_r	10WF			,. ·	1 1	Prot	oct	tion	. 6				
rtays. ct			เลางู	je.	Ϊ,	men			•••	L, I	1 7 1	ace	rici		/	L ,	100		. 1011	. 0				
0×0016000	90	55	8h	ec	83	ec	14	53	56	57	e8	00	00	00	00	5h	81	1	1	9	VW.		г.	
0x0016001	10	eb	6e	2b	40	00	89	5d	fc	8b	45	08	8b	4d	fc	89	88			11	F	м		
0x0016002	20	14	02	00	00	c7	45	f4	00	00	00	00	8b	55	f4	6b	d2			E.			. k .	
0×0016003	30	28	8b	45	08	33	c9	66	8b	8c	10	32	12	00	00	85	c9	((.E.	3.f		2		
0×160000	55					F	USH	I EE	βP															
0x160001	8be	ec				M	10V	EBF	Р, E	ESP														
0x160003	83e	ec14	ļ.			S	UB	ESF	P, (9x14	1													
0x160006	53					F	PUSH	I EE	ЗX															
0×160007	56					F	PUSH	I ES	δI															
0×160008	57					F	PUSH	I EC	Ι															
0×160009	e80	0000	0000	90		C	ALL	. 0>	(160	900e	11													
0x16000e	5b					F	'0P	EB>	(
0x16000f	81e	eb6e	2b4	1000	9	S	UΒ	EΒ>	(, (9x40	92b6	5e												
0x160015	895	ōdfo				M	10V	[EE	3P - (9x4]	, E	EBX												
0×160018	8b4	1508	3			M	10V	EA>	ζ,	EBF	2+0>	(8]												
0x16001b	8b4	ldfo				Μ	10 V	EC>	(,	LEBF	°-0>	(4]												
0x16001e	898	3814	1020	0000	9	M	10V	[EA	X+(9x21	[4],	E	CX											
0x160024	c74	15†4	1000	0000	900	M	10V	DWC	RD	LEE	3P - 6)xc], (0x0										
0x16002b	805	05†4	ł			ľ	10 V	ED)	,	[EBH	<0 - ^ر	(C]												
0x16002e	000	1228	5			1		EL)х,	ED/	(, (9X28	5											
0X160031	804	1508	5			P N		EAX	,		2+0>	(8]												
0x160034	330	.9 	103	0010	0000		UK	EC)	, t		ED		.105	21										
0x160036	000	0000	.103		2000	יין יט יפ ד		CX,			-=U/ /	(+0)	(12:	DZ]										
0x10003e	000	-9					ESI	EC	. ^ .	EU														

Figure 13

IEXPLORE.EXE injected process #2 - Source: author created

Chad Robertson, chad@rocacon.com

If we now run the same scan against the dumped binaries we see the same results:

Stabuniq [] /usr/share/malware//process.0x815f7020.0x160000.dmp Stabuniq [] /usr/share/malware//process.0x81752c10.0x160000.dmp *Figure 14*

Stabuniq YARA rule matched - Source: author created

As a final verification, we can use Virus Share's malware corpus. Scanning 131073 samples returns zero results.

5.2 X0rb0t

Next, we take a look at one of the public analysis posted by Malware.lu called X0rb0t. The analysis can be found here:

1. http://code.google.com/p/malware-lu/wiki/en_x0rb0t_analysis

This malware sample uses XOR for encoding. Let's focus on that function to create a YARA rule to test.

The function is shown below:

Γ		.text:00406CDE	cmp	[ebp+var_C], 7
		.text:00406CE2	jnb	short loc_406CFF
		.text:00406CE4	mov	eax, [ebp+Str]
	•	.text:00406CE7	add	eax, [ebp+var_8]
		.text:00406CEA	mov	ecx, [ebp+var_C]
	•	.text:00406CED	mov	dl, [eax]
		.text:00406CEF	xor	dl, byte_40EE60[ecx]
	•	.text:00406CF5	mov	eax, [ebp+Str]
		.text:00406CF8	add	eax, [ebp+var_8]
	•	.text:00406CFB	mov	[eax], dl
	<u> </u>	.text:00406CFD	jmp	short loc_406CD5



Related xor function - Source: author created

and the resulting YARA rule:

```
DISASSEMBLY:
        short loc_406CFF
jnb
-
10072
         eax, [ebp+Str]
add
         eax, [ebp+var 8]
         ecx, [ebp+var_C]
mov
mov
         dl, [eax]
        dl, byte_40EE60[ecx]
xor
        eax, [ebp+Str]
mov
add
         eax, [ebp+var_8]
        [eax], dl
short loc_406CD5
mov
jmp
BYTES:
73 1b 8b 45 fc 03 45 f8 8b 4d f4 8a 10 32 91 60 ee 40 00 8b 45 fc 03 45 f8 88 10 eb d6
BYTES PER LINE:
73 lb
8b 45 fc
03 45 f8
8b 4d f4
8a 10
32 91 60 ee 40 00
8b 45 fc
03 45 f8
88 10 |
eb d6
WILD-CARDED MEMORY BYTES:
73 ?? 8b ?? ?? 03 ?? ?? 8b ?? ?? 8a ?? 32 ?? ?? ?? ?? ?? 8b ?? ?? 03 ?? ?? 88 ?? eb ??
YARA SIGNATURE:
rule Sig
            strings:
                         $hex string = { 73 ?? 8b ?? ?? 03 ?? ?? 8b ?? ?? 8a ?? 32 ?? ?? ?? ?? ?? 8b ?? ?? 03 ?? ?? 88
            condition:
                         $hex_string
```

Figure 16

Output of Python script in IDA - Source: author created

When tested against the same set of memory dumps as was used for Stabuniq, with the addition of a memory dump from an x0rb0t infection, the Yara rule matches only the known x0rb0t-infected dump:

X0rb0t [] /usr/share/memory dumps/public//WINDOWSXP-x0rb0t.raw Figure 17

X0rbot YARA rule matched - Source: author created

When run against a corpus of 131073 files, the rule matches 8 times. Taking a closer look at one of the matches (503783a11080777a35b6349099fb3c3d) revels that the same function is utilized within both samples.

Below, on the right is the segment from x0rb0t, and the left the same code segment from 503783a11080777a35b6349099fb3c3d:

	tout - 88481120	0.00	Cohnauge Cl. A		text:00406CDF	cmp	[ebn+uar C1 7
	.LEXL.0040112D	Cill	[eup+var_c], o	_		Cinb	[cob.ogi_o]; /
	.text:0040112F	jnb	short loc 40114C	- 64	.text:00406CE2	jnb	short loc_406CFF
Į	text:00401131	mov	eax, [ebp+var 4]	- 8	.text:00406CE4	mov	eax, [ebp+Str]
Į	text:00401134	add	eax, [ebp+var 8]	- 8	.text:00406CE7	add	eax, [ebp+var_8]
Į	text:00401137	mov	ecx, [ebp+var C]	- 8	.text:00406CEA	MOV	ecx, [ebp+var_C]
Į	text:0040113A	mov	d1, [eax]	- 8	.text:00406CED	mov	dl, [eax]
Į	.text:0040113C	xor	dl, byte 4030B8[ecx]	- 8	.text:00406CEF	xor	dl, byte_40EE60[ecx]
Į	text:00401142	mov	eax, [ebp+var 4]	- 8	.text:00406CF5	mov	eax, [ebp+Str]
Į	.text:00401145	add	eax, [ebp+var 8]	- 8	.text:00406CF8	add	eax, [ebp+var_8]
Į	text:00401148	mov	[eax], dl	- 8	.text:00406CFB	mov	[eax], dl
Į	.text:0040114A	imp	short loc 401122	- 8	.text:00406CFD	jmp	short loc_406CD5
1		2.1	_	Fi	gure 18		

Output of Python script in IDA - Source: author created

As expected, the same instructions are present in both binaries.

5.3 APT1

Soon after Mandiant released their report on APT1 (See appendix 1), Alienvault released several YARA signatures to identify malware seemingly associated with that group. Both the YARA signatures and the malware samples are easily found online (See Appendix 1). To test the effectiveness of utilizing the YARA rules to identify APT1 activity within memory, a sample from the set is selected (8934aeed5d213fe29e858eee616a6ec7), executed, and the memory from the compromised host dumped to disk.

To start, we test the Alienvault YARA rule against the malicious binary to verify the rule works:



Figure 19

The known malicious bilary matches the rule – Source: Author created

The Alienvault rule matches the malware as expected. Now, the malicious binary is executed on a test system, and the memory from the test system acquired. The same YARA rule is then used to scan the memory dump:

																.vol.py -f /opt/cuckoo/shares/Windows-XP/WINDOW
XP-20130228	3-192	408	.raw	p	rof:	ile=	=Wir	NXPS	5P2>	x86	ya	raso	an	- y		/Downloads/yara-1.6/apt-test.yar
Volatile Sy	ster	is V	olat	ilit	y F	rame	ewor	rk 2	2.2							
Rule: Eclip	seSu	InCl	oudR	AT												
Owner: Proc	ess	021	b4ce	5c4d	9eb4	4 P:	id 7	748								
0x00422244	2f	75	63 5	f 73	65	72	76	65	72	2f	64	61	74	61	2f	f /uc_server/data/
0x00422254	66	6f	72 7	5 6d	2e	61	73	70	00	00	00	50	4f	53	54	forum.aspPOST
0x00422264	00	00	00 00	0 6d	75	6c	74	69	70	61	72	74	2f	66	6f	fmultipart/fo
0x00422274	72	6d	2d 64	4 61	74	61	3b	20	62	6f	75	6e	64	61	72	2 rm-data;.boundar
Rule: Eclip	seSu	InCl	oudR	AT												
Owner: Proc	ess	021	b4ce	5c4d	9eb،	4 P:	id 7	748								
0x0042289b	58	69	61 6	f 4d	45	5c	53	75	6e	43	6c	6f	75	64	2d	d XiaoME\SunCloud-
0x004228ab	43	6f	64 6	5 5c	45	63	6c	69	70	73	65	5f	41	5c	52	2 Code\Eclipse_A\R
0x004228bb	65	6c	65 6	1 73	65	5c	45	63	6c	69	70	73	65	5f	43	3 elease\Eclipse_C
0x004228cb	6c	69	65 6	e 74	5f	42	2e	70	64	62	00	00	00	00	00	Ð lient_B.pdb
Rule: Eclip	seSu	InCl	oudR	AT												
Owner: Proc	ess	021	b4ce	5c4d	9eb	4 P:	id 7	748								
0x004228a2	53	75	6e 4	3 6c	6f	75	64	2d	43	6f	64	65	5c	45	63	B SunCloud-Code\Ec
0x004228b2	6c	69	70 7	3 65	5f	41	5c	52	65	6c	65	61	73	65	5c	c lipse_A\Release\
0x004228c2	45	63	6c 6	9 70	73	65	5f	43	6c	69	65	6e	74	5f	42	2 Eclipse_Client_B
0x004228d2	2e	70	64 6	2 00	00	00	00	00	00	00	00	00	00	00	00	0.pdb
Rule: Eclip	seSu	InCl	oudR	AT												
Owner: Proc	ess	021	b4ce	5c4d	9eb،	4 P:	id 7	748								
0x004228b0	45	63	6c 6	9 70	73	65	5f	41	5c	52	65	6c	65	61	73	B Eclipse_A\Releas
0x004228c0	65	5c	45 6	3 6c	69	70	73	65	5f	43	6c	69	65	6e	74	4 e\Eclipse_Client
0x004228d0	5f	42	2e 7	0 64	62	00	00	00	00	00	00	00	00	00	00	
0x004228e0	00	00	00 00	0 14	aa	42	00	b8	2a	42	00	00	00	00	00	DB*B
Rule: Eclip	seSu	InCl	oudR	AT												
Owner: Proc	ess	021	b4ce	5c4d	9eb،	4 P:	id 7	748								
0x004228c2	45	63	6c 6	9 70	73	65	5f	43	6c	69	65	6e	74	5f	42	2 Eclipse_Client_B
0x004228d2	2e	70	64 6	2 00	00	00	00	00	00	00	00	00	00	00	00	.pdb
0x004228e2	00	00	14 a	a 42	00	b8	2a	42	00	00	00	00	00	00	00	DB*B
0x004228f2	00	00	00 00	0 00	00	14	50	42	00	00	29	42	00	00	00	DPB)B

Figure 20

The Alienvault YARA rule matches the memory dump – Source: Author created

With very little effort we were able to take publically available indicators contained within YARA rules and make them actionable against the test machine. The same techniques can be utilized to identify compromised systems across production environments during incident response.

Conclusion

Incident response is evolving. Malware authors are constantly coming up with new ways to compromise systems. They have the ability to adapt quickly and often circumvent our slower moving, often signature-based, archaic security solutions. As such, we must begin to consider how we can achieve similar agility in the way we respond. Perhaps the best way to meet today's threats is to stop relying solely on these legacy tools and begin to focus more on the analyst, our human capital, to piece together malware artifacts.

The research conducted during the writing of this paper set out to identify how memory forensics can be leveraged to aid incident response. We began by discussing the limitations of standard security platforms. Next, we discussed how behavioral indicators can be created based on the tools, techniques, and procedures utilized by attackers. Then, we discussed how we might find data to be used as indicators, how YARA can be used to aid us, and how we might label and group our identified artifacts. Finally, we took a quick look at two pieces of malware and how we can use readily available analysis data to identify them using YARA.

With the techniques discussed here, an analyst can begin to accumulate and utilize indicators from both publicly disclosed sources and private research. These indicators can then grouped together to form behavioral indicators to move our overall intelligence closer to the adversary. Volatility provides analysts an unprecedented capability to analyze memory images and acquire intelligence. Finally, YARA can be leveraged to identify those indicators across binary data and memory images, allowing for rapid response capabilities.

References

- Barnes, C. (2012). Retrieved from Accuvant: http://blog.accuvantlabs.com/blog/case-b/makingida-1-part-one-%E2%80%93-yara-signature-creation-1
- Cloppert, M. (2009). Security Intelligence: Attacking the Kill Chain. Retrieved from SANS.org: http://computer-forensics.sans.org/blog/2009/10/14/security-intelligence-attackingthe-kill-chain/
- Cross, T. (2012, November 20). *Is the era of anti-virus over?* Retrieved from SC Magazine: http://www.scmagazine.com/is-the-era-of-anti-virus-over/article/269210/
- French, D. (2012). Writing Effective YARA Signatures to Identify Malware. Retrieved from Software Engineering Institute - Carnegie Mellon: http://blog.sei.cmu.edu/post.cfm/writing-effective-yara-signatures-to-identify-malware
- Gutierrez, F. (2012, Dec). *Trojan.Stabuniq Found on Financial Institution Servers*. Retrieved from http://www.symantec.com: http://www.symantec.com/connect/blogs/trojanstabuniqfound-financial-institution-servers
- Hoglund, G. (2011). *Is APT really about the person and not the malware?* Retrieved from Fast Horizon: http://fasthorizon.blogspot.com/2011/04/is-apt-really-about-person-andnot.html
- Honig, M. S. (2012). Practical Malware Analysis. No Starch Press.
- Hypponen, M. (2012, June 2). Why antivirus companies like mine failed to catch Flame and Stuxnet. Retrieved from Arstechnica: http://arstechnica.com/security/2012/06/whyantivirus-companies-like-mine-failed-to-catch-flame-and-stuxnet/
- Michael Ligh, S. A. (2010). *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code.* Wiley.
- Norman. (2012, August). *The many faces of Gh0st Rat.* Retrieved from norman.com: download01.norman.no/documents/ThemanyfacesofGh0stRat.pdf
- Rashid, F. Y. (2012, Nov). *How To Detect Zero-Day Malware And Limit Its Impact*. Retrieved from Dark Reading: http://www.darkreading.com/advancedthreats/167901091/security/attacks-breaches/240062798/how-to-detect-zero-daymalware-and-limit-its-impact.html
- The MITRE Corporation. (2012). Standardizing Cyber Threat Intelligence Information with the Structured Threat INformation eXpression (STIX(tm)).
- YARA in a nutshell. (2013). Retrieved from yara-project: http://code.google.com/p/yara-project/

rights.

Appendix A

Blogs, companies, and tools mentioned within this paper:

http://malwaremustdie.blogspot.jp/ - Malware Must Die!

http://joe4security.blogspot.ch - Joe Security LLC

http://hooked-on-mnemonics.blogspot.com - Hooked on Mnemonics worked for me

http://www.mandiant.com/ - Mandiant

http://www.fireeye.com/ - FireEye

http://www.secureworks.com/cyber-threat-intelligence/blog - Dell Secureworks

https://github.com/technoskald/maltrieve - Maltrieve

http://dionaea.carnivore.it – Malware honeypot

http://virusshare.com/ - Virus Share

http://malwr.com/ - Malwr

http://offensivecomputing.net/ - Offensive Computing / Open Malware

http://contagiodump.blogspot.com/ - Contagio Malware Dump

http://code.google.com/p/yara-project/ - YARA Project

https://www.mandiant.com/blog/mandiant-exposes-apt1-chinas-cyber-espionage-unitsreleases-3000-indicators/ - Mandiant's APT1 Report

http://www.threatexpert.com/ Threat Expert automated threat analysis

Upcoming Training

Click Here to {Get CERTIFIED!}



SANS Security West 2014	San Diego, CA	May 08, 2014 - May 17, 2014	Live Event
SANS Malaysia @MCMC 2014	Cyberjaya, Malaysia	May 12, 2014 - May 24, 2014	Live Event
Digital Forensics & Incident Response Summit	Austin, TX	Jun 03, 2014 - Jun 10, 2014	Live Event
SANSFIRE 2014	Baltimore, MD	Jun 21, 2014 - Jun 30, 2014	Live Event
SANS Canberra 2014	Canberra, Australia	Jun 30, 2014 - Jul 12, 2014	Live Event
SANS London Summer 2014	London, United Kingdom	Jul 14, 2014 - Jul 21, 2014	Live Event
SANS vLive - FOR508: Advanced Computer Forensic Analysis and Incident Response	FOR508 - 201407,	Jul 21, 2014 - Aug 27, 2014	vLive
Mentor Session - FOR 508	Saint Louis, MO	Aug 06, 2014 - Oct 08, 2014	Mentor
SANS Virginia Beach 2014	Virginia Beach, VA	Aug 18, 2014 - Aug 29, 2014	Live Event
SANS Chicago 2014	Chicago, IL	Aug 24, 2014 - Aug 29, 2014	Live Event
SANS Bangalore 2014	Bangalore, India	Sep 15, 2014 - Sep 27, 2014	Live Event
SANS DFIR Prague 2014	Prague, Czech Republic	Sep 29, 2014 - Oct 11, 2014	Live Event
SOS: SANS October Singapore 2014	Singapore, Singapore	Oct 07, 2014 - Oct 18, 2014	Live Event
SANS vLive - FOR508: Advanced Computer Forensic Analysis and Incident Response	FOR508 - 201410,	Oct 14, 2014 - Nov 20, 2014	vLive
Community SANS Paris @ HSC - FOR508 (in French)	Paris, France	Nov 03, 2014 - Nov 07, 2014	Community SANS
SANS London 2014	London, United Kingdom	Nov 15, 2014 - Nov 24, 2014	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced