Dissecting REMCOS RAT: An in- depth analysis of a widespread 2024 malware, Part Four

Selastic.co/security-labs/dissecting-remcos-rat-part-four





Subscribe



Detections, hunts using ES|QL, and conclusion

In previous articles in this multipart series [1] [2] [3], malware researchers on the Elastic Security Labs team decomposed the REMCOS configuration structure and gave details about its C2 commands. In this final part, you'll learn more about detecting and hunting REMCOS using Elastic technologies.

Detection and Hunt

The following Elastic Defend detections trigger on those techniques:

Persistence (Run key)

Startup Persistence by a Low Reputation Process

Process Injection

Privilege Escalation (UAC Bypass)

UAC Bypass via ICMLuaUtil Elevated COM Interface

Evasion (Disable UAC)

<u>Disabling User Account Control via Registry Modification</u> (REMCOS spawns cmd.exe that uses reg.exe to disable UAC via registry modification)

Command and Control

<u>Connection to Dynamic DNS Provider by an Unsigned Binary</u> (although it's not a requirement but most of the observed samples use dynamic DNS)

File Deletion

Remcos RAT INETCookies File Deletion

Modify Registry

Remcos RAT ExePath Registry Modification

The ExePath registry value used by the REMCOS watchdog process can be used as an indicator of compromise. Below is a KQL query example :

event.category:"registry" and event.action:"modification" and registry.value:"EXEpath" and not process.code_signature.trusted:true

~	elastic	Q Find apps, content, and more.	*/		o 🗞 o			
≡	D Discover 🗸		New	Open Share Alerts I	nspect 🛛 🕄 Save			
log	ıs-* ∨	tegory; "registry" and event.action:"modification" and reg	istry.value:"EXEpath" and not process.code_signature.trust	ted == true	🛱 🗸 🔿 Refresh			
Ð	28 hits				ø (¢			
	Documents Field statistics			i≣ Columns ↓ 1 f	ield sorted 🔤 🕾 🖸			
	\downarrow @timestamp \bigcirc \checkmark	k registry.data.strings	k registry.path V	k process.name	~			
	∠ ⁿ □ Mar 8, 2024 @ 14:56:02.720	‡,òc2%¶ øBgüõuÒ íB õ¥òá£ñ O{Ò ä΋{†? ──ÒÞP û	HKEY_USERS\S-1-5-21-2202811333-351044183- 798744757-500\Software\remcos_ynpcmznxzg\EXEpath	remcos3.exe				
	<pre>eⁿ □ Feb 5, 2024 @ 08:51:24.748</pre>	BKH⁻wA‡à †oT7& X⊗ •I⁻-ðj⊗⊗12	HKEY_USERS\S-1-5-21-2723637291-3094084044- 761125207-1000\Software\remcos_wgwfvnfssp\EXEpath	file.exe				
	√ [^] _ Jan 25, 2024 @ 07:46:14.959 ßKH⊤wAţàĂ toT7£ Xœ •I ⁻ -ðjœe12		HKEY_USERS\S-1-5-21-2723637291-3094084044- 761125207-1000\Software\remcos_wgwfvnfssp\EXEpath	file.exe h				
		cMàDce‰V Kr°øÿÄx>Å6™`'8	HKEY_USERS\S-1-5-21-2723637291-3094084044- 761125207-1000\Software\remcos_eoumsscfcq\EXEpath	file.exe uth				
	v ⁿ □ Jan 18, 2024 @ 09:39:49.431	‡,òc2%¶ øvpô1Gÿê[î°ÜÛŽò rÅ	HKEY_USERS\S-1-5-21-2723637291-3094084044- 761125207-1000\Software\remcos_ynpcmznxzg\EXEpath	file.exe				
		Ð ^'ŸJu9àÂ`ãÑJØFh№ ñ' T°\Ÿ•xR/éò_c+r	HKEY_USERS\S-1-5-21-2723637291-3094084044- 761125207-1000\Software\Windows Audio\EXEpath	Windows Sessions Start.exe				
	e ³ □ Sep 23, 2023 @ 13:27:48.166	iQ †ôÂ<Š œÆ œÓ×7ùPA−b >Zp %pÇ:i!:XÆï;@²′=°[âRÖ è#	HKEY_USERS\S-1-5-21-2723637291-3094084044- 761125207-1000\Software\remcos_hiqrammked\EXEpath	RegAsm.exe				
	√ ⁿ	%ü~él ³-exݱ0Ě″«ô;sB1Û,ö-i÷″+ cĐWNa:)if nìFF()É ⁻ &vã	HKEY_USERS\S-1-5-21-2723637291-3094084044- 761125207-1900\Software\Windows Audio\EXEnath	Windows Session Start.exe				
	Rows per page: 100 🗸				< 1 >			

REMCOS includes three options for clearing browser data, possibly in an attempt to force victim users to re-enter their web credentials for keylogging:

- enable_browser_cleaning_on_startup_flag
- enable_browser_cleaning_only_for_the_first_run_flag
- browser_cleaning_sleep_time_in_minutes

This results in the deletion of browser cookies and history-related files. The following KQL query can be used to hunt for such behavior by an unsigned process:

event.category:file and event.action:deletion and file.name:container.dat and file.path:*INetCookies* and not process.code_signature.trusted:true

\$	elastic	Q Find ap	os, content, and more.	*/					O 🖋 D					
≡	D Discover 🗸			Ne	ew Open	Share	Alert	s Inspect	Save					
log	s-* ∨	tegory:file and event.action:	deletion and file.name:container.dat	and file.path:*INetCookies* and not		8	*	Last 1 year	ී Refresh					
Ð	87 hits								ø @					
	Documents Field statistics					i≣ Col	umns	1 field sorte	d 🖙 🕾 🖸					
	\downarrow @timestamp \bigcirc \checkmark	k process.name v	k file.name v	k file.path					~					
	∠ ⁿ □ Mar 8, 2024 @ 14:53:26.605	Windriver.exe	container.dat	C:\Users\Administrator\AppData\Local\Micr	osoft\Windo	ows\INet	Cookies\8	SE\container	.dat					
	∠ ² □ Mar 8, 2024 @ 14:53:26.604	Windriver.exe	container.dat	C:\Users\Administrator\AppData\Local\Wicrosoft\Windows\INetCookies\DNTException\container.dat										
	√ ² □ Mar 8, 2024 @ 14:53:26.603	Windriver.exe	container.dat	C:\Users\Administrator\AppData\Local\Micr	osoft\Windo	\Windows\INetCookies\container.dat								
	√ ⁿ □ Feb 16, 2024 @ 08:06:19.555	file.exe	container.dat	C:\Users\user\AppData\Local\Microsoft\Windows\INetCookies\Low\ESE\container.dat										
	√ ² ☐ Feb 16, 2024 @ 08:06:19.465	file.exe	container.dat	C:\Users\user\AppData\Local\Microsoft\Windows\INetCookies\ESE\container.dat										
	√ □ Feb 16, 2024 @ 08:06:19.424	file.exe	container.dat	C:\Users\user\AppData\Local\Microsoft\Windows\INetCookies\DNTException\container.dat										
	√ ² □ Feb 16, 2024 @ 08:06:19.346	file.exe	container.dat	C:\Users\user\AppData\Local\Microsoft\Windows\INetCookies\container.dat										
	√ ² □ Jan 22, 2024 @ 10:36:58.078	Windriver.exe	container.dat	C:\Users\user\AppData\Local\Microsoft\Win	ndows\INetCo	ookies\L	ow\ESE\co	ontainer.dat						
	√ ⁿ □ Jan 22, 2024 @ 10:36:58.071	Windriver.exe	container.dat	C:\Users\user\AppData\Local\Microsoft\Win	ndows\INetCo	ookies\E	SE\contai	ner.dat						
	√ ⁿ ☐ Jan 22, 2024 @ 10:36:58.065	Windriver.exe	container.dat	C:\Users\user\AppData\Local\Microsoft\Win	ndows\INetCo	ookies\D	NTExcepti	on\container	.dat					
	∠ Jan 22, 2024 @ 10:36:58.058	Windriver.exe	container.dat	C:\Users\user\AppData\Local\Microsoft\Win	ndows\INetCo	ookies\c	ontainer	dat						
	Rows per page: 100 v								(1)					

REMCOS also employs three main information collection methods. The first one is keylogging via <u>SetWindowsHookEx</u> API. The following <u>ES|QL</u> can be used to hunt for rare or unusual processes performing this behavior:

from logs-endpoint.events.api*

/* keylogging can be done by calling SetwindowsHook to hook keyboard events */

| where event.category == "api" and process.Ext.api.name == "SetWindowsHookEx" and process.Ext.api.parameters.hook_type like "WH_KEYBOARD*"

/* normalize process paths to ease aggregation by process path */

| eval process_path = replace(process.executable, """([0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}|ns[a-z][A-Z0-9]{3,4}\.tmp|DX[A-Z0-9] {3,4}\.tmp|7z[A-Z0-9]{3,5}\.tmp|[0-9\.\-_]{3,})""", "") | eval process_path = replace(process_path, """[cC]:\\[uU][sS][eE][rR][sS]\\[a-zA-Z0-9\.\-_\\$~]+\\""", "C:\\\\users\\\\user\\\\")

/* limit results to those that are unique to a host across the agents fleet */

```
| stats occurrences = count(*), agents = count_distinct(host.id) by process_path
| where occurrences == 1 and agents == 1
```

Below is an example of matches on *iexplore.exe* (injected by REMCOS):

🤣 e	elastic	Q Find apps, content, and more.	*/				Setup g	guides	© (% E
≡	D Discover			New	Open	Share	Alerts	Inspect	٦	Save
ES	qL ∨						✓ Last	30 days	CR	efresh
≡									ہ ^{لا}	Ţ.
1 1 2 3 4 7 5 6 7 8 9 9	<pre>from logs-endpoint.events.api* /* keylogging can be done by calling where event.category == "api" and /* normalize process.path = replace(proces [A-Z0-9]{3,4}\.tmp 7z[A-Z0-9]{3,5}\. eval process.path = replace(proces /* limit results to those that are u stats occurences == l and agents = </pre>	<pre>SetwindowsHook to hook keyboard events process.Ext.api.name == "SetWindowsHookE: ggregation by process path */ s.executable, """([0=0=fA-F](8)-[0=9a-fi tmp1[0=9\.\-_]{3,})""", "") s.path, """[cC]:\\[uU][sS][dE][rR][sS]\\ nique to a host accross the fleet */ s = count_distinct(host.id) by process_p = 1</pre>	<br */ and process.Ext.api.parameters A-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F [a-zA-Z0-9\.\-_\\$~]+***, *C:* ath	.hook_type]{4}-[0-9a- \\users\\\\	like "WH fA-F]{1: user\\\'	H_KEYBOA 2} ns[a- \")	RD*" z][A-Z0-	-9]{3,4}\	.tmp D	×
9 line	es @timestamp found		_			🗣 Sut	bmit feedba	ack Run que	ery ≋ +	Enter
Ð	ESQL is currently in technical preview.	ind more information in the documentation $arnothing$.								×
	3 hits					i≣ Colu	umns 🗘	Sort fields		\$ \$
		# occurences ~	# agents	a ∽ (t) proces	ss_path					~
	~ 🗆	1		1 C:\Progr	am Files	\Internet	Explorer	\iexplore.e	exe	
	en 🗋	1		1 C:\Progr	am Files	(x86)\In	ternet Ex	plorer∖iexp	olore.e	xe

ES|QL hunt for rare processes calling SetWindowsHoook to hook keyboard events

The second method takes multiple screenshots and saves them as jpg files with a specific naming pattern starting with time_year-month-day_hour-min-sec.jpb (e.g. time_20240308_171037.jpg). The following <u>ES|QL</u> hunt can be used to identify suspicious processes with similar behavior :

from logs-endpoint.events.file*
/* remcos screenshots naming pattern */
| where event.category == "file" and host.os.family == "windows" and event.action ==
"creation" and file.extension == "jpg" and file.name rlike
"""time_202\d{5}_\d{6}.jpg"""
| stats occurrences = count(*), agents = count_distinct(host.id) by process.name,
process.entity_id
 /* number of screenshots i more than 5 by same process.pid and this behavior is
limited to a unique host/process */

| where occurrences >= 5 and agents == 1

The following image shows both REMCOS and the injected iexplore.exe instance (further investigation can be done by pivoting by the <u>process.entity_id</u>):

*	elastic	Q Find apps, content,	and more.		×/			Setup guides	٥	۵.	•
≡	Discover				New	Open	Share	Alerts Inspe	et 📕	🗟 Sav	е
ES	siqu v							 Last 30 days 		3 Refre	sh
=	5									,≓ [<u>i</u>
1 2 3 4 5 6	<pre>from logs-endpoint.events.file* /* keylogging can be done by calling where event.category == "file" and file.extension == "jpg" and file.r stats occurrences = count(*), agent where occurences >= 5 and agents =</pre>	SetwindowsHook to host.os.family == ' ame rlike """time_20 s = count_distinct(N = 1	nook keyboard ev "windows" and ev 02\d{5}_\d{6}.jp nost.id) by proc	ents */ ent.action == "creation" and g""" ess.name, process.entity_id							_
7 lin	nes @timestamp found 🔥 1 warning						🗣 Sut	bmit feedback Ru	n query	* + Ent	er
Ð	ESQL is currently in technical preview.	ind more information in t	the documentation	= 3.							×
	6 hits										¢
							i≣ Colu	umns 🗘 Sort f	elds	∞	
	# occurences ~	/ agents 🗸	t process.na v	t process.entity_id							~
2 5 1 iexplore.exe YWExM2Y1N2UtNGQwhS800x281LWEyYTYthjJiMjA1YWhjZmY4LTExhTQ0LTE3MDk5MTM50TYuMzA50DQyMT											
		1	remcos.exe	YWExM2Y1N2UtNGQwNS00NzBlLWEyYTYtNjJiMj	A1YWNjZmY4LTI2NjQ1	tMTcw0Tkx	MjUwNS4yN	DcxNTgwMA==			
	ى 🖸	1	iexplore.exe	YWExM2Y1N2UtNGQwNS00NzBlLWEyYTYtNjJiMj	A1YWNjZmY4LTE0MzA4	4LTE3MDk5	MTMyNTIuN	DQ3OTA×MTAw			

ES|QL hunt for rare processes creating JPG files similar to REMCOS behavior

The third collection method is an audio recording saved as WAV files. The following <u>ES|QL</u> hunt can be used to find rare processes dropping WAV files:

```
from logs-endpoint.events.file*
| where event.category == "file" and host.os.family == "windows" and event.action ==
"creation" and file.extension == "wav"
/* normalize process paths to ease aggregation by process path */
| eval process_path = replace(process.executable, """([0-9a-fA-F]{8}-[0-9a-fA-F]{4}-
[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}|ns[a-z][A-Z0-9]{3,4}\.tmp|DX[A-Z0-9]
{3,4}\.tmp|7z[A-Z0-9]{3,5}\.tmp|[0-9\.\-\_]{3,})""", "")
| eval process_path = replace(process_path, """[cC]:\\[uU][sS][eE][rR][sS]\\[a-zA-Z0-
9\.\-\_$~]+\\""", "C:\\\\users\\\\user\\\\")
| stats wav_files_count = count(*), agents = count_distinct(host.id) by process_path
/* limit results to unique process observed in 1 agent and number of dropped wav
files is less than 20 */
```

| where agents == 1 and wav_files_count <= 10

😓 (elastic	 Find apps, content, a 	and more.		*/				Setup	guides	٥	St (
≡	D Discover					New	Open	Share	Alerts	Inspect	8	Save	
ES	qL ∨								 Last 	30 days	G	Refresh	
≡											7		
1 2 3 4 5 6 7	<pre>1 from logs-endpoint.events.file* 2 where event.category == "file" and host.os.family == "windows" and event.action == "creation" and file.extension == "wav" 3 /* normalize process path to ease aggregation by process path */ 4 eval process_path = replace(process.executable, """([0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12} ns[a-z][A-Z0-9]{3,4}\.tmp DX [A-Z0-9]{3,4}\.tmp DX [A-</pre>												
10 lir	nes @timestamp found <u>A</u> 1 warning							🗣 Sut	bmit feedba	ack Run q	uery 🛪 🕯	+ Enter	
Ð		ind more information in t	the documentation 🖄.	=									×
	2 hits												¢
								i≣ Colu	umns 🗍	Sort field	s	≌≂ (
		# agents ~	t process_path										~
	ج 🗋 4	1	C:\ProgramData\Remcos\	remcos.exe									
	√ □ 1	1	C:\Program Files (x86)	\Internet Explorer\ie:	plore.exe								

ES|QL hunt for rare processes creating WAV files

The following <u>ES|QL</u> hunt can also look for processes that drop both JPG and WAV files using the same process.pid :

```
from logs-endpoint.events.file*
| where event.category == "file" and host.os.family == "windows" and event.action ==
"creation" and file.extension in ("wav", "jpg") and
/* excluding privileged processes and limiting the hunt to unsigned
process or signed by untrusted certificate or signed by Microsoft */
not user.id in ("S-1-5-18", "S-1-5-19", "S-1-5-20") and
(process.code_signature.trusted == false or process.code_signature.exists == false or
starts_with(process.code_signature.subject_name, "Microsoft"))
| eval wav_pids = case(file.extension == "wav", process.entity_id, null), jpg_pids =
case(file.extension == "jpg", process.entity_id, null), others = case(file.extension
!= "wav" and file.extension != "jpg", process.entity_id, null)
/* number of jpg and wav files created by unique process identifier */
stats count_wav_files = count(wav_pids), count_jpg_files = count(jpg_pids),
other_files = count(others) by process.entity_id, process.name
/* limit results to same process dropping both file extensions */
where count_jpg_files >= 1 and count_wav_files >= 1
```

Examples of matches on both REMCOS and the injected *iexplore.exe* process:

😽 elas	C Find apps, content, a	nd more.	×/				Setup g	guides	٥	& E		
	Discover			New	Open	Share	Alerts	Inspect	•	Save		
ES QL	/						✓ Last	30 days	D	Update		
⇒									أحر	r T		
1 from 2 wh 3 ev 4 /* 5 st 6 /* 7 wh	<pre>logs-endpoint.events.file* re event.category == "file" and host.os.family == " ul wav.pids = case(file.extension == "wav", process.unumber of jpg and wav files created by unique process its count_wav_files = count(wav_pids), count_jpg_file imit results to same process dropping both file exter re count_jpg_files >= 1 and count_wav_files >= 1</pre>	<pre>windows" and event.action entity_id, null), jpg_pids s identifier */ es = count(jpg_pids) by p ensions */</pre>	<pre>== "creation" and file.extens s = case(file.extension == "jp rocess.entity_id, process.name</pre>	ion in (" g", proce	wav", " ss.enti	jpg") ty_id, r	null)			_		
8 lines (timestamp found 🔥 1 warning					🗣 Su	bmit feedba	ack Run qu	iery 🕷 -	+ Enter		
A	SQL is currently in technical preview. Find more information in th	e documentation ⊠.								×		
4 h	ts					i≣ Cole	umns 🗘	Sort field	5	\$ \$		
		∉ count_jpg_files ∨	t process.entity_id	~	t proces	ss.name				~		
<i>رچ</i> ا] 1	7	YWExM2Y1N2UtNGQwNS00NzB1LWEyYTYtN WNjZmY4LTE0MzA4LTE3MDk5MTMyNTIuND Aw	jJiMjA1Y Q3OTAxMT	iexplore	.exe	e					
♪ 1 8 YWEXM2Y1N2UtNGQwNS00NzBllWEyYTYtNjJIMjATY remcos.exe WNjZmY4LTI2NjQtMTcw0TkxMjUwNS4yNDcxNTgwMA												
21] 1	8	YWExM2Y1N2UtNGQwNS00NzB1LWEyYTYtN	jJiMjA1Y	remcos.e	xe						

ES|QL hunts for unique processes dropping image and audio files

Pivoting by <u>process.entity_id</u> to further investigate suspicious processes, installers, browsers, and decompression utilities are often the most observed false positives.

YARA rule

The REMCOS version 4.9.3 is detected statically using the following <u>YARA rule</u> produced by Elastic Security Labs

Malware and MITRE ATT&CK

Elastic uses the <u>MITRE ATT&CK</u> framework to document common tactics, techniques, and procedures that advanced persistent threats use against enterprise networks.

Tactics

Tactics represent the *why* of a technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action.

- Execution
- Persistence
- <u>Privilege Escalation</u>
- Defense Evasion
- <u>Credential Access</u>
- <u>Discovery</u>
- <u>Command and Control</u>

Techniques

Techniques represent how an adversary achieves a tactical goal by performing an action.

- <u>Windows Command Shell</u>
- Visual Basic
- Registry Run Keys / Startup Folder
- <u>Process Injection</u>
- <u>Credentials from Web Browsers</u>
- Encrypted Channel
- System Binary Proxy Execution: CMSTP
- <u>Bypass User Account Control</u>

Conclusion

As the REMCOS continues to rapidly evolve, our in-depth analysis of version 4.9.3 offers critical insights that can significantly aid the malware research community in comprehending and combatting this pervasive threat.

By uncovering its features and capabilities in this series, we provide essential information that enhances understanding and strengthens defenses against this malicious software.

We've also shown that our Elastic Defend product can detect and stop the REMCOS threat. As this article demonstrates, our new query language, ES|QL, makes hunting for threats simple and effective.

Elastic Security Labs remains committed to this endeavor as part of our open-source philosophy, which is dedicated to sharing knowledge and collaborating with the broader cybersecurity community. Moving forward, we will persist in analyzing similar malware families, contributing valuable insights to bolster collective defense against emerging cyber threats.

Sample hashes and C2s

(Analysis reference)

0af76f2897158bf752b5ee258053215a6de198e8910458c02282c2d4d284add5

remchukwugixiemu4.duckdns[.]org:57844

remchukwugixiemu4.duckdns[.]org:57846

remchukwugix231fgh.duckdns[.]org:57844

remchukwugix231fgh.duckdns[.]org:57846

3e32447 ea3b5 f07 c7 f6a180269 f5443378 acb32 c5d0 e0 bf01 a 5e39264 f691587

122.176.133[.]66:2404

122.176.133[.]66:2667

8c9202885700b55d73f2a76fbf96c1b8590d28b061efbadf9826cdd0e51b9f26

43.230.202[.]33:7056

95dfdb588c7018babd55642c48f6bed1c281cecccbd522dd40b8bea663686f30

107.175.229[.]139:8087

517f65402d3cf185037b858a5cfe274ca30090550caa39e7a3b75be24e18e179

money001.duckdns[.]org:9596

b1a149e11e9c85dd70056d62b98b369f0776e11b1983aed28c78c7d5189cfdbf

104.250.180[.]178:7902

ba6ee802d60277f655b3c8d0215a2abd73d901a34e3c97741bc377199e3a8670

- 185.70.104[.]90:2404
- 185.70.104[.]90:8080
- 185.70.104[.]90:465
- 185.70.104[.]90:80
- 77.105.132[.]70:80
- 77.105.132[.]70:8080
- 77.105.132[.]70:2404
- 77.105.132[.]70:465

Research references