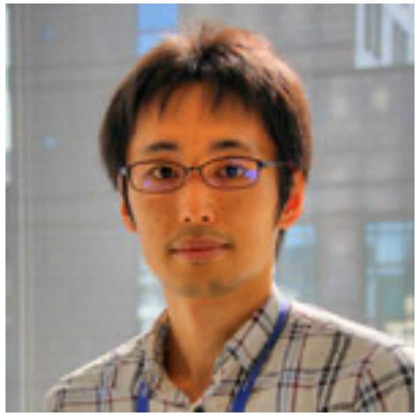
Let's go door with KCP

LAC

Who Are We ?



Yoshihiro Ishikawa

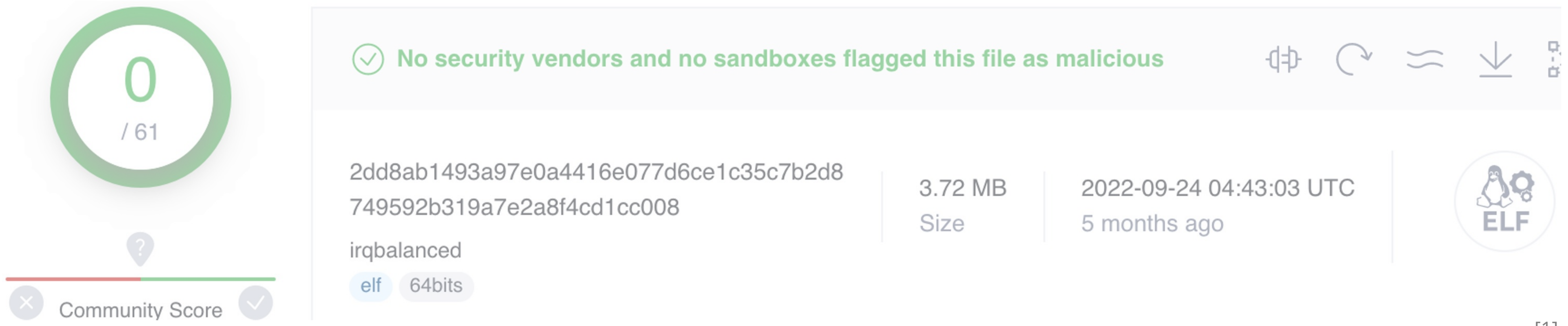
- Organization: LAC Co.,Ltd. (lac.co.jp)
- Department: Cyber Emergency Center
- Job Title: Cyber Threat and Malware Analyst

Takuma Matsumoto

- Organization: LAC Co.,Ltd. (lac.co.jp)
- Department: Cyber Emergency Center
- Job Title: Malware Analyst

Agenda

Introduction


No security vendors and no sandboxes flagged this file as malicious

2dd8ab1493a97e0a4416e077d6ce1c35c7b2d8
749592b319a7e2a8f4cd1cc008
irqbalanced

[1]

- **gokcpdoor** is an interesting malware using **KCP** protocol coded by **golang**
- Increasing use of KCP protocol as **APT malware** communication
- Several incident cases have been confirmed since **April 2022**
- **Not detected** by security software until July 2022

We introduce the analysis result of **gokcpdoor** and related threat to **prevent similar attacks** in the future.

01

A Study of KCP

KCP

- A fast and reliable Automatic repeat-request (ARQ) protocol
- Providing **low-latency communications**
- The code written in C was published by skywind3000 in 2011 [2]
- KCP requires a transmission mode for sending and receiving of the underlying data
- **Most implementations use UDP Protocol**
  - Transmission speed is 30%-40% faster than TCP
  - Bandwidth is increased by 10%-20%

Proxy software

Online game

Streaming service

Layer	Protocol
Application	**KCP**
Transport	UDP, TCP, …
Internet	IP, APR, …
Network Interface	Ethernet, …

KCP Message Segment

- The message segment consists of a 24-byte header and variable length data

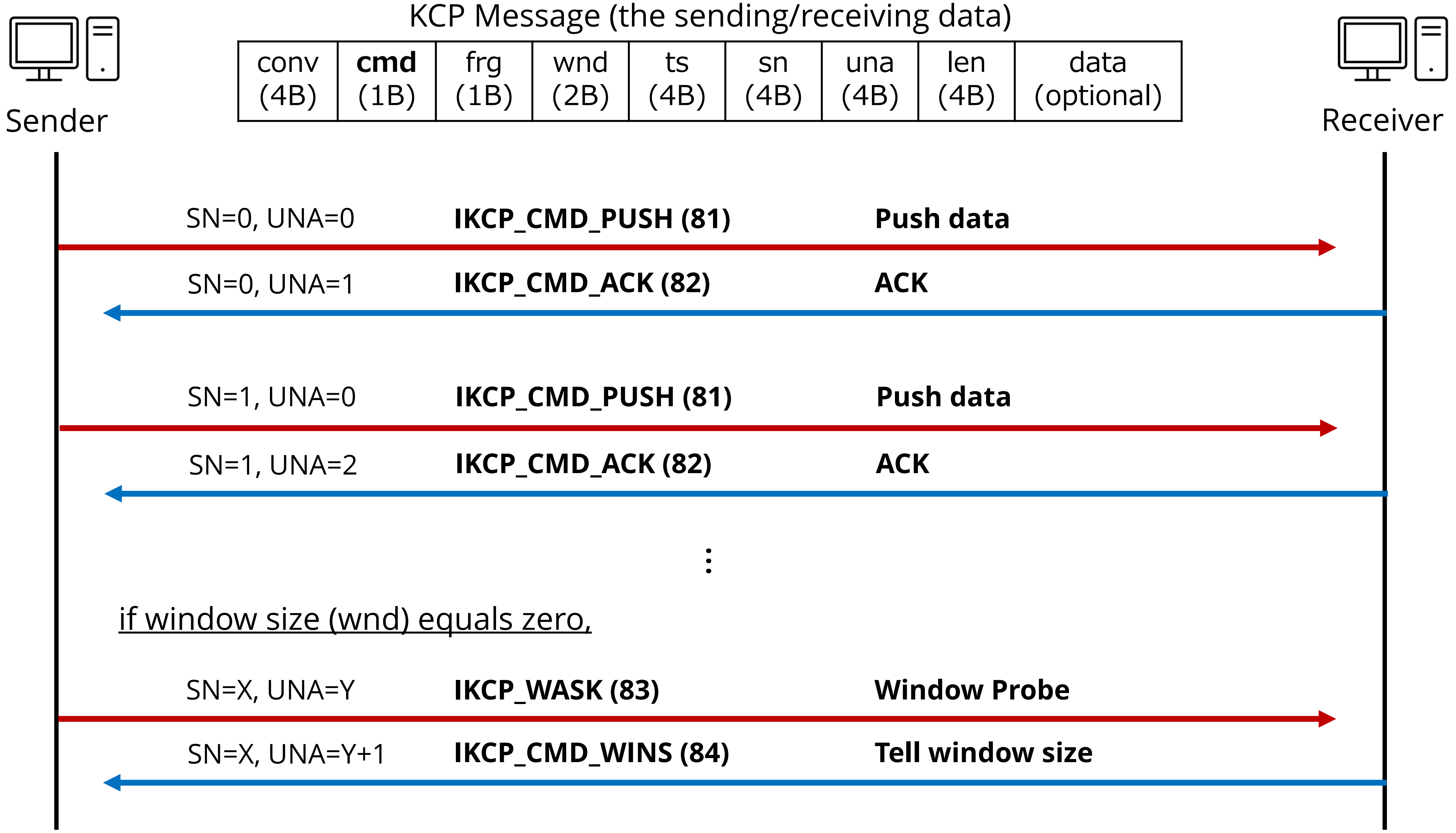
Header (24 bytes)								Body
conv (4B)	**cmd** (1B)	frg (1B)	wnd (2B)	ts (4B)	sn (4B)	una (4B)	len (4B)	**data** (optional)

Field	Size	Description
conv	4 bytes	Session number
cmd	1 byte	Commands: • IKCP_CMD_PUSH 81 (0x51 ‘Q’) : Data message • IKCP_CMD_ACK 82 (0x52 ‘R’) : Acknowledgement message • IKCP_WASK 83 (0x53 ‘S’) : Window probe message • IKCP_CMD_WINS 84 (0x54 ‘T’) : Window receive message
frg	1 byte	Number of fragments
wnd	2 bytes	Window size (Size of the sender’s remaining receive window)
ts	4 bytes	Timestamp
sn	4 bytes	Serial number
una	4 bytes	Number of KCP message segments received
len	4 bytes	Length of the data segment
data	variable	Data segment (Only data messages have this field)

Communication Flow

KCP Message (the sending/receiving data)

conv (4B)	**cmd** (1B)	frg (1B)	wnd (2B)	ts (4B)	sn (4B)	una (4B)	len (4B)	data (optional)

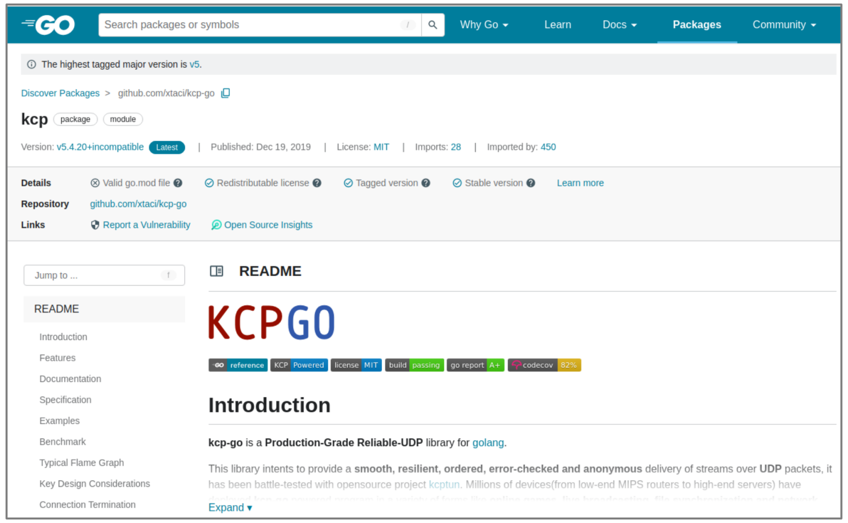


kcp-go

- **A Reliable-UDP library for golang** with extensions based on **KCP** [3]

kcp-go = UDP + KCP + FEC+ Block Encryption

- Supports Forward Error Correction (FEC) with Reed-Solomon Coding
- Supports encryptions of KCP message:
  - AES
  - Blowfish
  - Cast5
  - SM4
  - Salsa20
  - TEA
  - TripleDES
  - Twofish
  - XTEA
  - XOR



The highest tagged major version is v5.

Discover Packages > github.com/xtaci/kcp-go

kcp package module

Version: v5.4.20+incompatible Latest | Published: Dec 19, 2019 | License: MIT | Imports: 28 | Imported by: 450

Details: Valid go.mod file · Redistributable license · Tagged version · Stable version · Learn more

Repository: github.com/xtaci/kcp-go

Links: Report a Vulnerability · Open Source Insights

README

KCPGO

Introduction

kcp-go is a **Production-Grade Reliable-UDP** library for golang.

This library intents to provide a **smooth, resilient, ordered, error-checked and anonymous** delivery of streams over **UDP** packets, it has been battle-tested with opensource project kcptun. Millions of devices(from low-end MIPS routers to high-end servers) have

Expand ▾

Date	Version
Jun 29, 2016	(First commit)
Aug 15, 2016	v1.0.0
Sep 09, 2016	v2.0.2
Mar 07, 2017	v3.0
Sep 14, 2018	v4.1
Dec 28, 2018	v5.0
Aug 11, 2023	v5.6.3 (Newest)

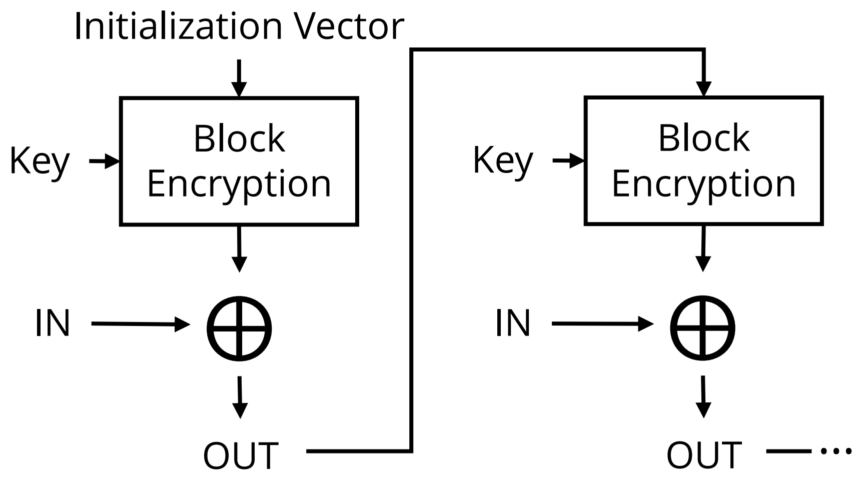
Encryption by kcp-go

- The plaintext consists of a 28-byte header and variable length body

Header (28 bytes)					Body
Nonce (16B)	CRC32 (4B)	FEC SEQID (4B)	FEC TYPE (2B)	SIZE (2B)	**KCP Message** / ParityShard (SIZE – 2B)

- kcp-go adds a **nonce** as part of the header, hence encrypting the same plaintext yields different results each time
- Uses the following settings for encryption:
  - **Cipher Feedback (CFB) mode**
  - Initialization Vector (IV)

```
[]byte{167, 115, 79, 156, 18, 172, 27, 1, 164, 21, 242, 193, 252, 120, 230, 107}
```



CFB mode encryption
(IN = Plaintext, OUT = Cipher)
CFB-mode decryption
(IN = Cipher, OUT = Plaintext)

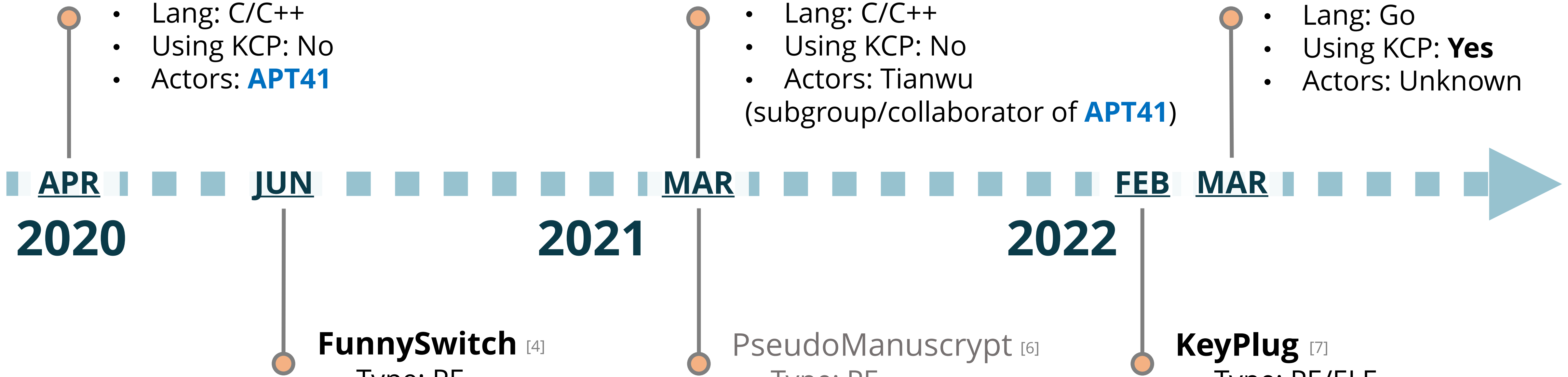
02

APT Malware Using KCP Protocol

Timeline of Malware Family with KCP Protocol

LAC

* Malware activity timeline based on sample compile time



Crosswalk [4]
- Type: PE
- Lang: C/C++
- Using KCP: No
- Actors: **APT41**

FunnySwitch [4]
- Type: PE
- Lang: C#(.Net)
- Using KCP: No
- Actors: **APT41**

Pangolin8RAT [5]
- Type: PE
- Lang: C/C++
- Using KCP: No
- Actors: Tianwu

(subgroup/collaborator of **APT41**)

PseudoManuscrypt [6]
- Type: PE
- Lang: C/C++
- Using KCP: Yes
- Actors: Unknown

* KCP implementation unconfirmed

KeyPlug [7]
- Type: PE/ELF
- Lang: C/C++
- Using KCP: No
- Actors: **APT41**

gokcpdoor
- Type: PE/ELF
- Lang: Go
- Using KCP: **Yes**
- Actors: Unknown

About Crosswalk, KeyPlug and Pangolin8RAT with KCP

kcp/ikcp.c [2]

```
const IUINT32 IKCP_CMD_PUSH = 81;
const IUINT32 IKCP_CMD_ACK  = 82;
const IUINT32 IKCP_CMD_WASK = 83;
const IUINT32 IKCP_CMD_WINS = 84;
```

(redacted)

```
if (cmd != IKCP_CMD_PUSH && cmd != IKCP_CMD_ACK &&
        cmd != IKCP_CMD_WASK && cmd != IKCP_CMD_WINS)
        return -3;
```

(redacted)

```
if (cmd == IKCP_CMD_ACK) {
        if (_itimediff(kcp->current, ts) >= 0) {
                ikcp_update_ack(kcp, _itimediff(kcp->current, ts));
```

(redacted)

```
if (ikcp_canlog(kcp, IKCP_LOG_IN_ACK)) {
        ikcp_log(kcp, IKCP_LOG_IN_ACK,
                "input ack: sn=%lu rtt=%ld rto=%ld", (unsigned long)sn,
                (long)_itimediff(kcp->current, ts),
                (long)kcp->rx_rto);
```

Crosswalk (MD5: a8bb1d69fb8a9d323bbc5d78f0e62850)

```
if ( cmd != 0x51 && cmd != 0x52 && cmd != 0x53 && cmd != 0x54 )
  return -3;
```

(redacted)

```
if ( cmd == 0x52 )
{
  if ( a1[19] - v28 >= 0 )
    ikcp_update_ack(a1);
```

KeyPlug (MD5:070eb0289afef7856f50fa63e7ebde87)

```
if ( cmd == 0x52 )
{
  if ( (int)(*(_DWORD *)(a1 + 76) - v44) >= 0 )
    ikcp_update_ack(a1);
```

(redacted)

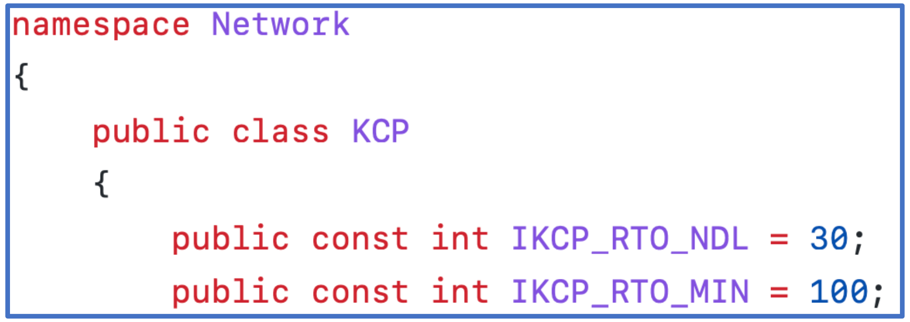
```
if ( ikcp_canlog(a1, 0x20) )
{
  LODWORD(v44) = *(_DWORD *)(a1 + 0x30);
  LODWORD(v43) = *(_DWORD *)(a1 + 0x4C) - v22;
  ikcp_log(a1, 0x10, "input ack: sn=%lu rtt=%ld rto=%ld", v10, v43, v44);
```

Pangolin8RAT (MD5:bf421d42174edb2f31007cbede9cf5b9)

```
if ( (*(_BYTE *)(a1 + 244) & 0x20) != 0 && *(_QWORD *)(a1 + 256) )
{
  LODWORD(v56) = *(_DWORD *)(a1 + 0x30);
  LODWORD(v55) = *(_DWORD *)(a1 + 0x4C) - v57;
  ikcp_log(a1, 0x10, "input ack: sn=%lu rtt=%ld rto=%ld", v59, v55, v56);
```

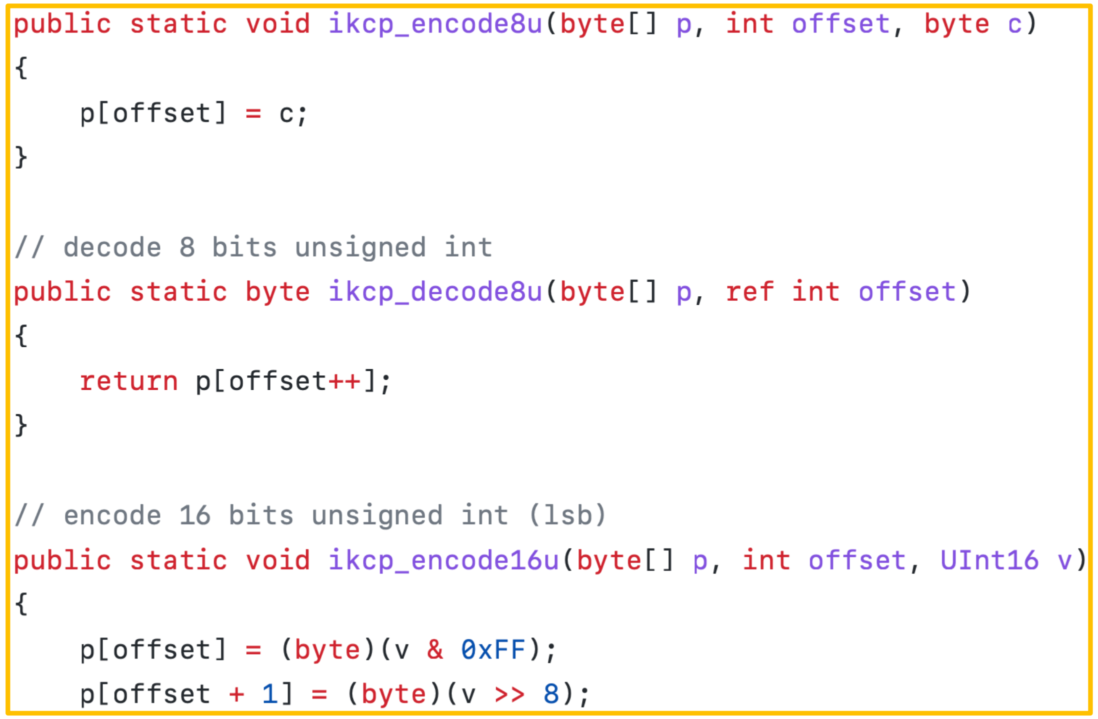
About FunnySwitch with KCP

kcp-dotnet/KCP.cs [8]



```
namespace Network
{
    public class KCP
    {
        public const int IKCP_RTO_NDL = 30;
        public const int IKCP_RTO_MIN = 100;
```

⋮ (redacted)

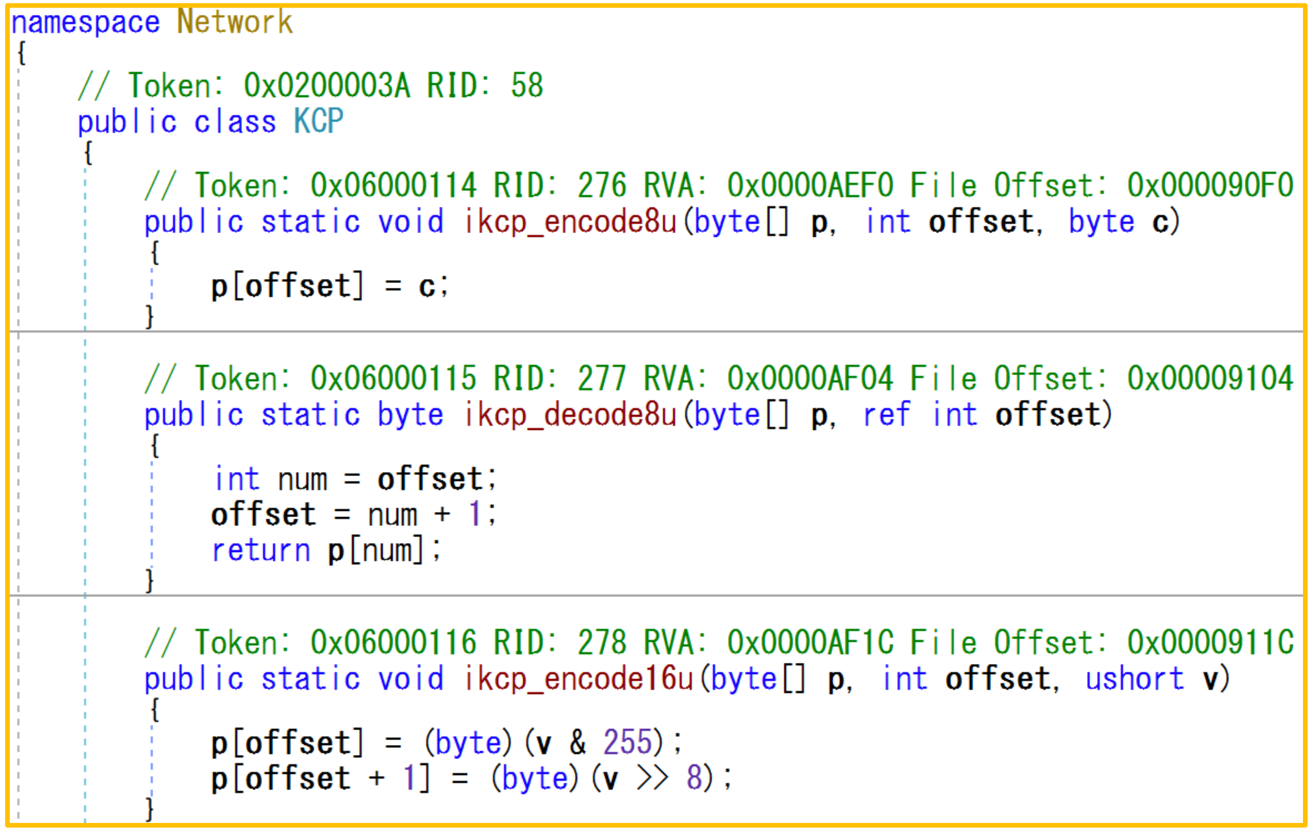


```
public static void ikcp_encode8u(byte[] p, int offset, byte c)
{
    p[offset] = c;
}

// decode 8 bits unsigned int
public static byte ikcp_decode8u(byte[] p, ref int offset)
{
    return p[offset++];
}

// encode 16 bits unsigned int (lsb)
public static void ikcp_encode16u(byte[] p, int offset, UInt16 v)
{
    p[offset] = (byte)(v & 0xFF);
    p[offset + 1] = (byte)(v >> 8);
```

FunnySwitch (MD5: 2b0c692d9eafed5e24f2b52234ea0fa2)

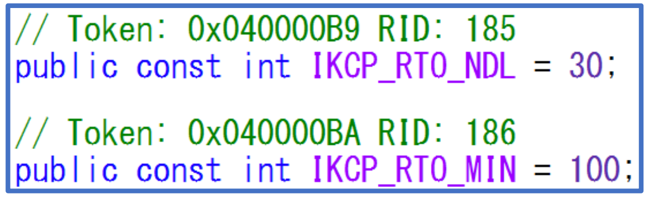


```
namespace Network
{
    // Token: 0x0200003A RID: 58
    public class KCP
    {
        // Token: 0x06000114 RID: 276 RVA: 0x0000AEF0 File Offset: 0x000090F0
        public static void ikcp_encode8u(byte[] p, int offset, byte c)
        {
            p[offset] = c;
        }

        // Token: 0x06000115 RID: 277 RVA: 0x0000AF04 File Offset: 0x00009104
        public static byte ikcp_decode8u(byte[] p, ref int offset)
        {
            int num = offset;
            offset = num + 1;
            return p[num];
        }

        // Token: 0x06000116 RID: 278 RVA: 0x0000AF1C File Offset: 0x0000911C
        public static void ikcp_encode16u(byte[] p, int offset, ushort v)
        {
            p[offset] = (byte)(v & 255);
            p[offset + 1] = (byte)(v >> 8);
        }
```

⋮ (redacted)



```
// Token: 0x040000B9 RID: 185
public const int IKCP_RTO_NDL = 30;

// Token: 0x040000BA RID: 186
public const int IKCP_RTO_MIN = 100;
```

About gokcpdoor with KCP

kcp-go/kcp.go [3]

```
func (kcp *KCP) Input(data []byte, regular, ackNoDelay bool)

if cmd != IKCP_CMD_PUSH && cmd != IKCP_CMD_ACK &&
        cmd != IKCP_CMD_WASK && cmd != IKCP_CMD_WINS {
        return -3
```

⋮ (redacted)

```
if cmd == IKCP_CMD_ACK {
        kcp.parse_ack(sn)
        kcp.parse_fastack(sn, ts)
        flag |= 1
        latest = ts
```

⋮ (redacted)

```
if windowSlides { // if window has slided, flush
        kcp.flush(false)
} else if ackNoDelay && len(kcp.acklist) > 0 { // ack immediately
        kcp.flush(true)
}
return 0
```

gokcpdoor (MD5: a6f4a5ec66b7c5f275e793be02885543)

```
// program/kcp.(*KCP).Input

if ( cmd != 0x51 && cmd != 0x52 && cmd != 0x53 && cmd != 0x54 )
  return -3LL;
```

⋮ (redacted)

```
if ( cmd == 0x52 )
{
  program_kcp__ptr_KCP_parse_ack(v83, v68, v31, a4, a5, 0x52LL
  a2 = v68;
  program_kcp__ptr_KCP_parse_fastack(v83, v68, v66);
  v9 = v76 | 1;
  v11 = v66;
  a1 = v83;
```

⋮ (redacted)

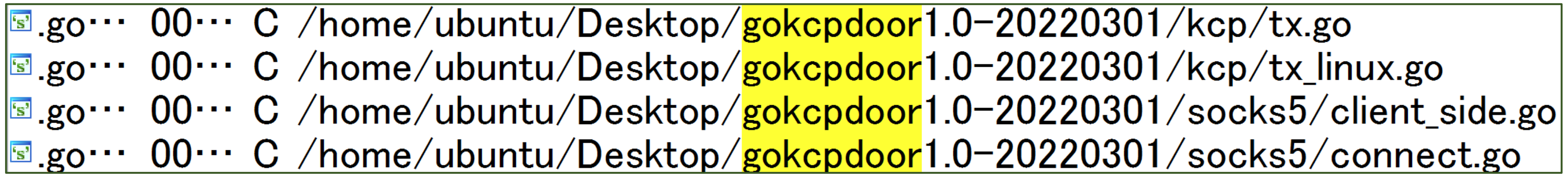
```
if ( windowSlides )
{
  program_kcp__ptr_KCP_flush(v50, 0LL, a3, v45, a5);
}
else if ( (_BYTE)a6 && *(__int64 *)(v50 + 216) > 0 )
{
  program_kcp__ptr_KCP_flush(v50, 1uLL, a3, v45, a5);
}
return 0LL;
```

03

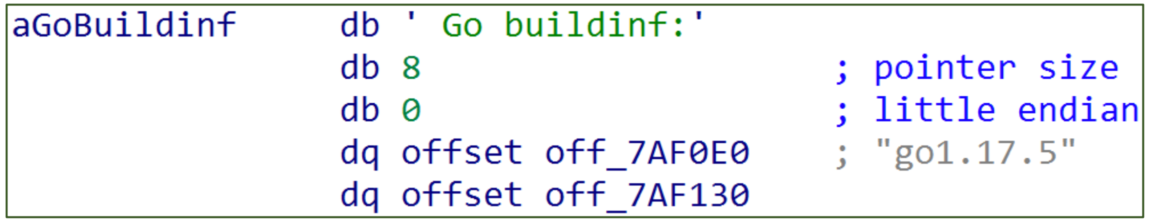
Deep Dive into gokcpdoor

What's about gokcpdoor

- Functions
  - Backdoor commands
  - Open port
  - **KCP** protocol
- Identification
  - Lang: **golang**
  - Type: **Windows** and **Linux** (ELF)
  - Characteristics:
    - Naming is compile path contains **gokcpdoor**
    - Compiled with Ubuntu (build with go1.17.5)
    - Using some **OSS libraries**
  - First seen: April 2022



```
.go··· 00··· C /home/ubuntu/Desktop/gokcpdoor1.0-20220301/kcp/tx.go
.go··· 00··· C /home/ubuntu/Desktop/gokcpdoor1.0-20220301/kcp/tx_linux.go
.go··· 00··· C /home/ubuntu/Desktop/gokcpdoor1.0-20220301/socks5/client_side.go
.go··· 00··· C /home/ubuntu/Desktop/gokcpdoor1.0-20220301/socks5/connect.go
```



```
aGoBuildinf      db ' Go buildinf:'
                 db 8                    ; pointer size
                 db 0                    ; little endian
                 dq offset off_7AF0E0    ; "go1.17.5"
                 dq offset off_7AF130
```

gokcpdoor Has OSS Library Lists (1/2)

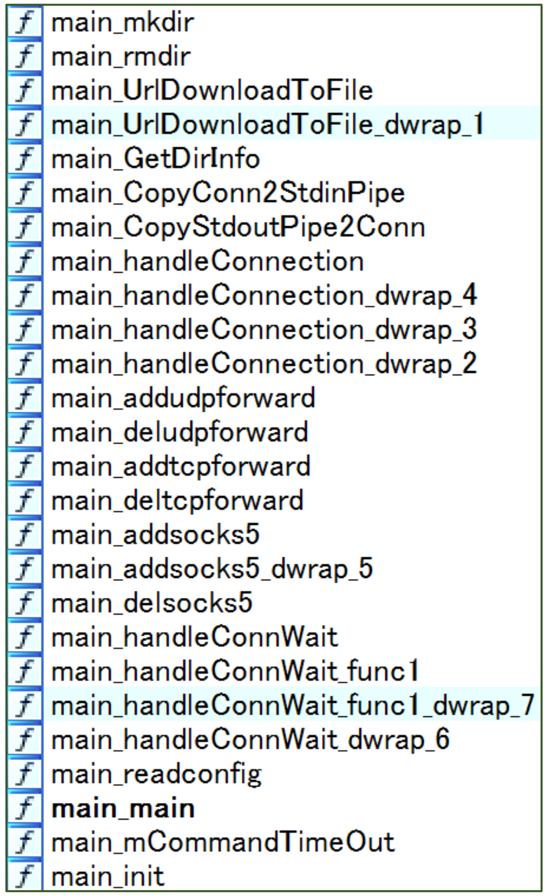
Path	OSS Libraries (GitHub)	Description
/home/ubuntu/go/pkg/mod/github.com/**klauspost/reedsolomon** @v1.9.13/reedsolomon.go	klauspost/Reedsolomon	Provide Reed-Solomon coding
/home/ubuntu/go/pkg/mod/github.com/**klauspost/cpuid**/v2@v2.0 .6/cpuid.go	klauspost/cpuid	Get information about related CPU
/home/ubuntu/go/pkg/mod/github.com/**templexxx/cpu**@v0.0.7/c pu.go	templexxx/cpu	Get information about related CPU
/home/ubuntu/go/pkg/mod/github.com/**templexxx/xorsimd**@v0. 4.1/xor.go	templexxx/xorsimd	Provide XOR code engine
/home/ubuntu/go/pkg/mod/github.com/**pkg/errors**@v0.9.1/errors. go	pkg/errors	Provide simple error handling primitives
/home/ubuntu/go/pkg/mod/github.com/**tjfoc/gmsm**@v1.4.1/sm4/ sm4.go	tjfoc/gmsm	Provide Chinese cryptographic algorithm
/home/ubuntu/go/pkg/mod/github.com/**txthinking/x**@v0.0.0-20210326105829-476fab902fbe/dial.go	txthinking/x	Provide some network utilities function
/home/ubuntu/go/pkg/mod/github.com/**txthinking/runnergroup** @v0.0.0-20210608031112-152c7c4432bf/runnergroup.go	txthinking/runnergroup	End concurrency reliably
/home/ubuntu/go/pkg/mod/github.com/**patrickmn/go-cache**@v2.1.0+incompatible/cache.go	patrickmn/go-cache	Provide in-memory cache function

gokcpdoor Has OSS Library Lists (2/2)

Path	OSS Libraries (GitHub)	Description
/home/ubuntu/Desktop/gokcpdoor1.0-20220301/**kcp/kcp.go**	xtaci/kcp-go	Provides KCP connection
/home/ubuntu/Desktop/gokcpdoor1.0-20220301/**kcp/sess.go**		Provides KCP session implemented by UDP
/home/ubuntu/Desktop/gokcpdoor1.0-20220301/**socks5/client_side.go**	txthinking/socks5	Provides SOCKS5 implemented for client
/home/ubuntu/Desktop/gokcpdoor1.0-20220301/**socks5/udp.go**		Provides UDP support for SOCKS5
/home/ubuntu/Desktop/gokcpdoor1.0-20220301/syscmds/**ps/ps_linux.go** /home/ubuntu/Desktop/gokcpdoor1.0-20220301/syscmds/**ps/ps_windows.go**	BishopFox/Sliver	Provides API for finding and listing processes
/home/ubuntu/Desktop/gokcpdoor1.0-20220301/syscmds/**netstat/netstat_linux.go** /home/ubuntu/Desktop/gokcpdoor1.0-20220301/syscmds/**netstat/netstat_windows.go**		Provides “netstat” command function
/home/ubuntu/Desktop/gokcpdoor1.0-20220301/tcpforward/tcpforward.go	digibib/tcpforward	Provides forward TCP traffic
/home/ubuntu/Desktop/gokcpdoor1.0-20220301/udpforward/udpforward.go	1lann/udp-forward	Provides forward UDP traffic

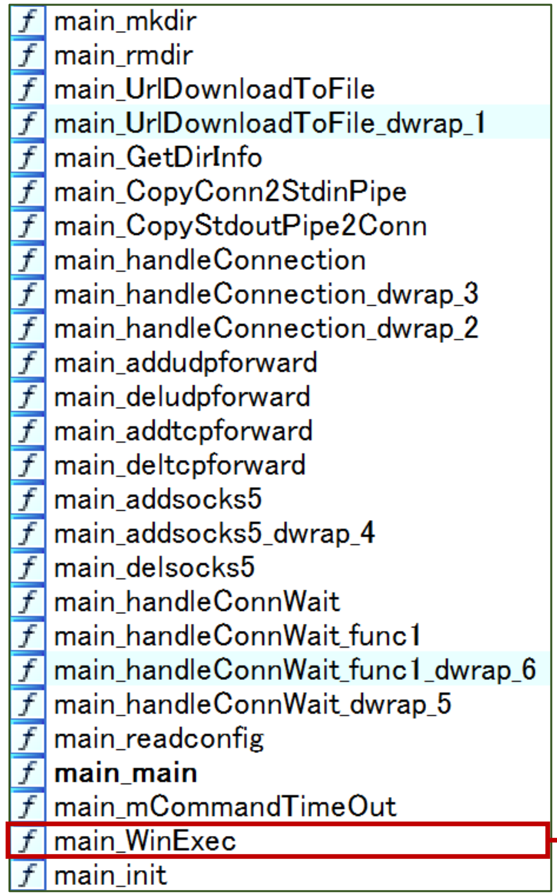
Comparison Linux and Windows gokcpdoor Functions

- Compares **specific functions (main.*)** implemented by **gokcpdoor developers**
- Function is **almost the same** on Linux and Windows, but **Windows** has one **characteristic**



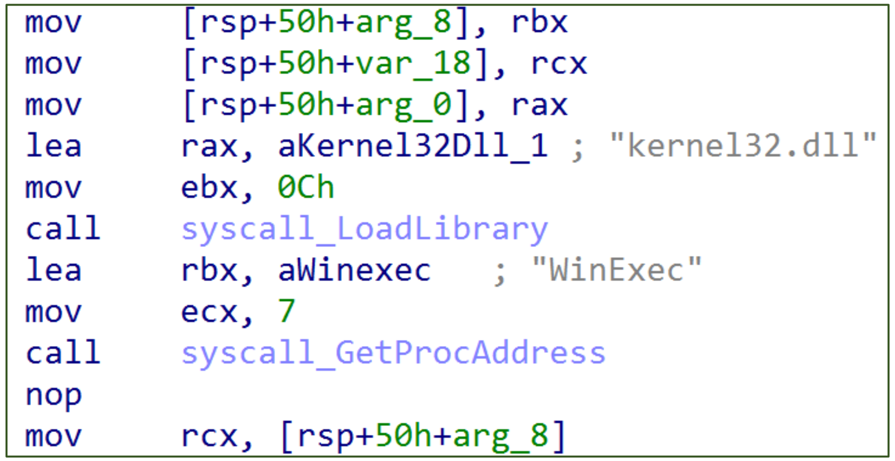
```
main_mkdir
main_rmdir
main_UrlDownloadToFile
main_UrlDownloadToFile_dwrap_1
main_GetDirInfo
main_CopyConn2StdinPipe
main_CopyStdoutPipe2Conn
main_handleConnection
main_handleConnection_dwrap_4
main_handleConnection_dwrap_3
main_handleConnection_dwrap_2
main_addudpforward
main_deludpforward
main_addtcpforward
main_deltcpforward
main_addsocks5
main_addsocks5_dwrap_5
main_delsocks5
main_handleConnWait
main_handleConnWait_func1
main_handleConnWait_func1_dwrap_7
main_handleConnWait_dwrap_6
main_readconfig
main_main
main_mCommandTimeOut
main_init
```

For Linux functions

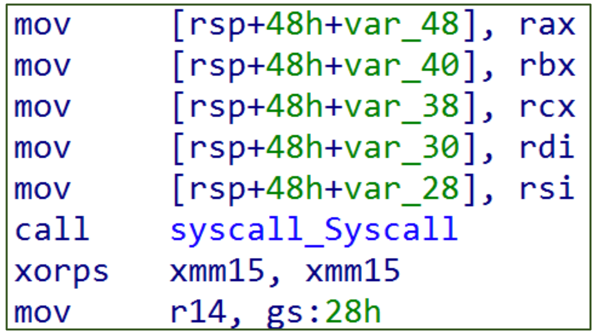


```
main_mkdir
main_rmdir
main_UrlDownloadToFile
main_UrlDownloadToFile_dwrap_1
main_GetDirInfo
main_CopyConn2StdinPipe
main_CopyStdoutPipe2Conn
main_handleConnection
main_handleConnection_dwrap_3
main_handleConnection_dwrap_2
main_addudpforward
main_deludpforward
main_addtcpforward
main_deltcpforward
main_addsocks5
main_addsocks5_dwrap_4
main_delsocks5
main_handleConnWait
main_handleConnWait_func1
main_handleConnWait_func1_dwrap_6
main_handleConnWait_dwrap_5
main_readconfig
main_main
main_mCommandTimeOut
main_WinExec
main_init
```

For Windows functions



```
mov     [rsp+50h+arg_8], rbx
mov     [rsp+50h+var_18], rcx
mov     [rsp+50h+arg_0], rax
lea     rax, aKernel32Dll_1 ; "kernel32.dll"
mov     ebx, 0Ch
call    syscall_LoadLibrary
lea     rbx, aWinexec    ; "WinExec"
mov     ecx, 7
call    syscall_GetProcAddress
nop
mov     rcx, [rsp+50h+arg_8]
```

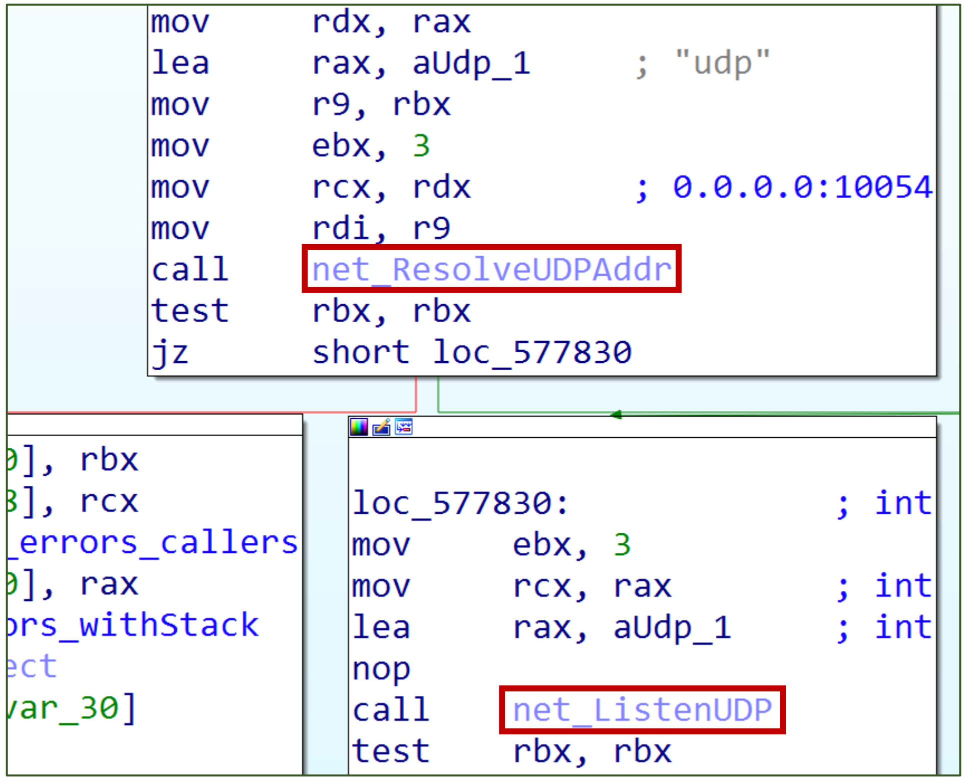


```
mov     [rsp+48h+var_48], rax
mov     [rsp+48h+var_40], rbx
mov     [rsp+48h+var_38], rcx
mov     [rsp+48h+var_30], rdi
mov     [rsp+48h+var_28], rsi
call    syscall_Syscall
xorps   xmm15, xmm15
mov     r14, gs:28h
```

Get address of **WinExec** and call it via Syscall function

Open Port (1/2)

- gokcpdoor starts opening port with **hardcoded port number**
- The port number differs depending on the sample

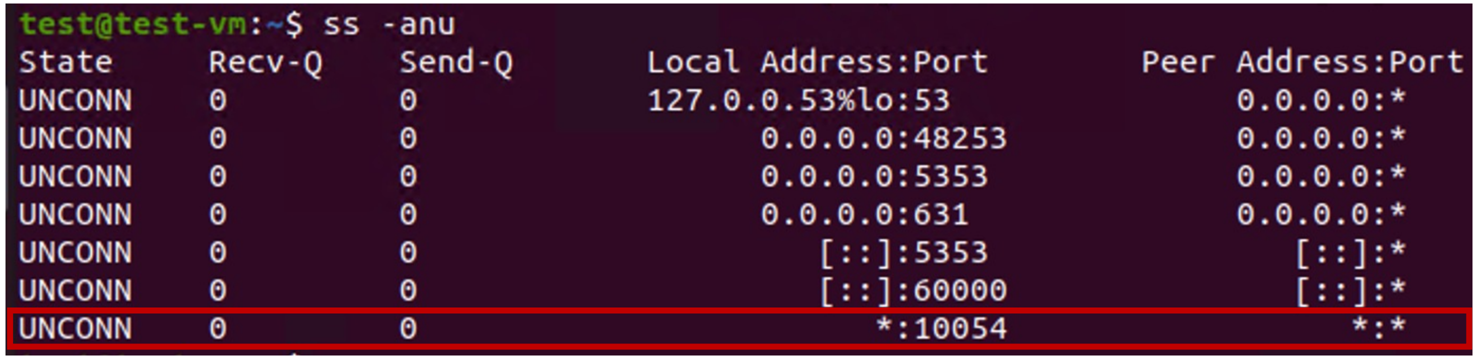


```
mov     rdx, rax
lea     rax, aUdp_1      ; "udp"
mov     r9, rbx
mov     ebx, 3
mov     rcx, rdx         ; 0.0.0.0:10054
mov     rdi, r9
call    net_ResolveUDPAddr
test    rbx, rbx
jz      short loc_577830
```

```
0], rbx
8], rcx
_errors_callers
0], rax
ors_withStack
ect
var_30]
```

```
loc_577830:              ; int
mov     ebx, 3
mov     rcx, rax         ; int
lea     rax, aUdp_1      ; int
nop
call    net_ListenUDP
test    rbx, rbx
```

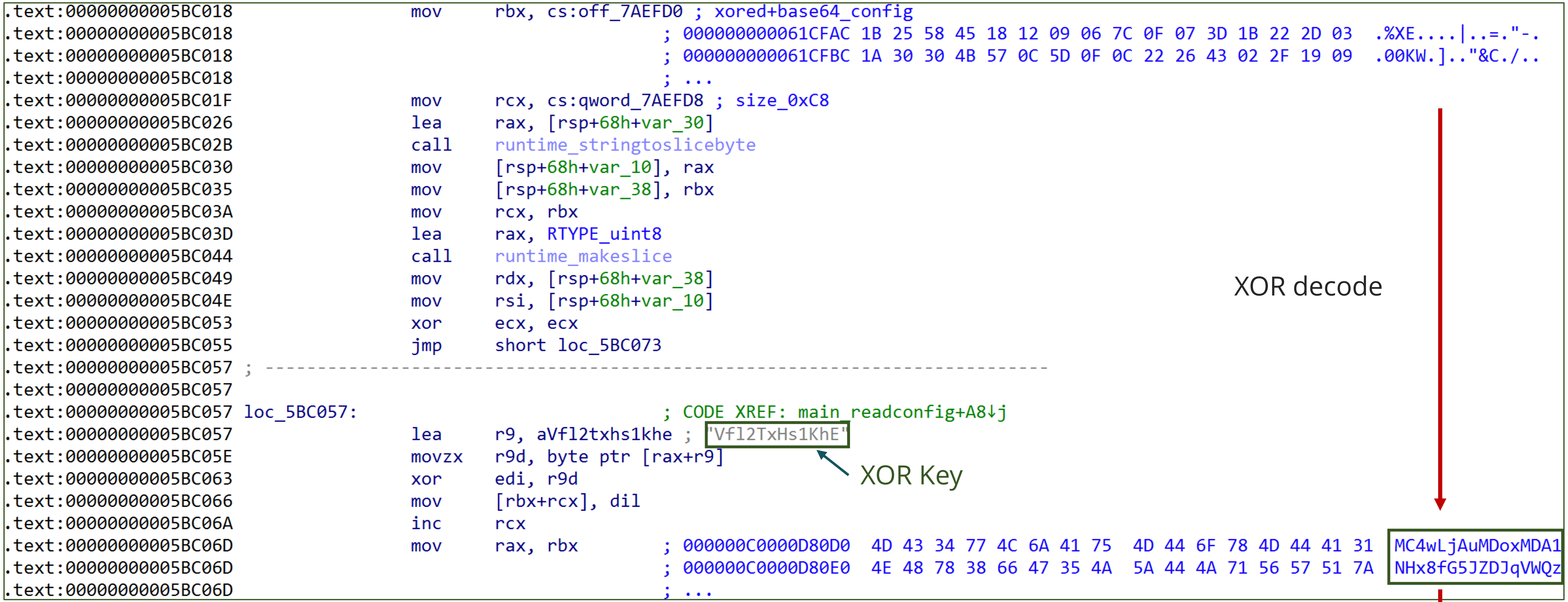
Opening 10054/udp using net package some functions



```
test@test-vm:~$ ss -anu
State    Recv-Q    Send-Q        Local Address:Port        Peer Address:Port
UNCONN   0         0         127.0.0.53%lo:53                   0.0.0.0:*
UNCONN   0         0               0.0.0.0:48253                0.0.0.0:*
UNCONN   0         0               0.0.0.0:5353                 0.0.0.0:*
UNCONN   0         0               0.0.0.0:631                  0.0.0.0:*
UNCONN   0         0                  [::]:5353                    [::]:*
UNCONN   0         0                  [::]:60000                   [::]:*
UNCONN   0         0                     *:10054                      *:*
```

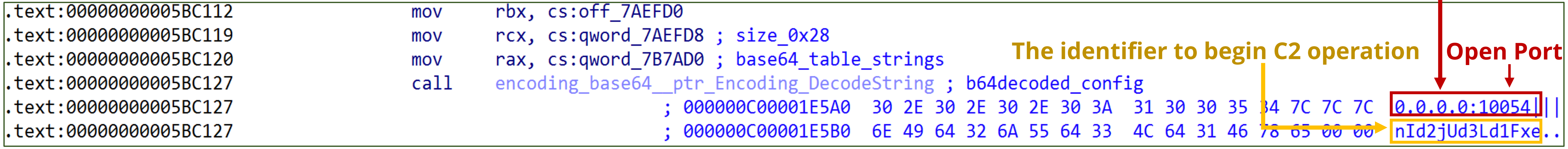
Show open port of UDP using ss command

Open Port (2/2)



```
.text:00000000005BC018                 mov     rbx, cs:off_7AEFD0 ; xored+base64_config
.text:00000000005BC018                                         ; 000000000061CFAC 1B 25 58 45 18 12 09 06 7C 0F 07 3D 1B 22 2D 03  .%XE....|..=."-.
.text:00000000005BC018                                         ; 000000000061CFBC 1A 30 30 4B 57 0C 5D 0F 0C 22 26 43 02 2F 19 09  .00KW.].."&C./..
.text:00000000005BC018                                         ; ...
.text:00000000005BC01F                 mov     rcx, cs:qword_7AEFD8 ; size_0xC8
.text:00000000005BC026                 lea     rax, [rsp+68h+var_30]
.text:00000000005BC02B                 call    runtime_stringtoslicebyte
.text:00000000005BC030                 mov     [rsp+68h+var_10], rax
.text:00000000005BC035                 mov     [rsp+68h+var_38], rbx
.text:00000000005BC03A                 mov     rcx, rbx
.text:00000000005BC03D                 lea     rax, RTYPE_uint8
.text:00000000005BC044                 call    runtime_makeslice
.text:00000000005BC049                 mov     rdx, [rsp+68h+var_38]
.text:00000000005BC04E                 mov     rsi, [rsp+68h+var_10]
.text:00000000005BC053                 xor     ecx, ecx
.text:00000000005BC055                 jmp     short loc_5BC073
.text:00000000005BC057 ; ---------------------------------------------------------------------------
.text:00000000005BC057
.text:00000000005BC057 loc_5BC057:                             ; CODE XREF: main_readconfig+A8↓j
.text:00000000005BC057                 lea     r9, aVfl2txhs1khe ; "Vfl2TxHs1KhE"
.text:00000000005BC05E                 movzx   r9d, byte ptr [rax+r9]
.text:00000000005BC063                 xor     edi, r9d
.text:00000000005BC066                 mov     [rbx+rcx], dil
.text:00000000005BC06A                 inc     rcx
.text:00000000005BC06D                 mov     rax, rbx        ; 000000C0000D80D0  4D 43 34 77 4C 6A 41 75  4D 44 6F 78 4D 44 41 31  MC4wLjAuMDoxMDA1
.text:00000000005BC06D                                         ; 000000C0000D80E0  4E 48 78 38 66 47 35 4A  5A 44 4A 71 56 57 51 7A  NHx8fG5JZDJqVWQz
.text:00000000005BC06D                                         ; ...
```

⋮ (redacted)

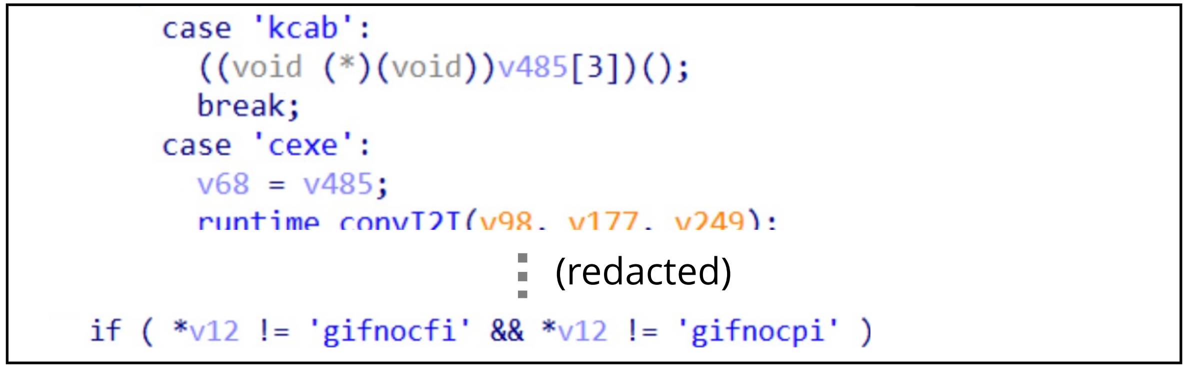


```
.text:00000000005BC112                 mov     rbx, cs:off_7AEFD0
.text:00000000005BC119                 mov     rcx, cs:qword_7AEFD8 ; size_0x28
.text:00000000005BC120                 mov     rax, cs:qword_7B7AD0 ; base64_table_strings
.text:00000000005BC127                 call    encoding_base64__ptr_Encoding_DecodeString ; b64decoded_config
.text:00000000005BC127                                         ; 000000C00001E5A0  30 2E 30 2E 30 2E 30 3A  31 30 30 35 34 7C 7C 7C  0.0.0.0:10054|||
.text:00000000005BC127                                         ; 000000C00001E5B0  6E 49 64 32 6A 55 64 33  4C 64 31 46 78 65 00 00  nId2jUd3Ld1Fxe..
```

C2 Commands

Command	Description
exec	Execute a program
shell	Start reverse shell session
wget	Download a file from URL on infected host
upload	Upload a file from C2 server to infected host
download	Download a file from infected host to C2 server
dir / ls	List the contents of the specified directory
mkdir	Create a directory
rm	Remove the specified directory or file
cd	Change current directory
pwd	Get current directory path
whoami / id	Get username by executing “whoami” or “id” command
getos	Get OS information by executing “wmic os get name” or “uname –a” command
ps	List all running processes
ifconfig / ipconfig	List all network interfaces
netstat	Get network statistics about all active connections

Command		Description
portforward	list	List all port forwarding settings
	add	Add port forwarding setting which TCP or UDP can be selected
	del	Delete port forwarding setting
socks5	list	List all SOCKS5 settings
	add	Add SOCKS5 setting
	del	Delete SOCKS5 setting
charset		Change character set (gokcpdoor only supports UTF-8)
back		End C2 command operation
exitprocess		Terminate own process



```
    case 'kcab':
      ((void (*)(void))v485[3])();
      break;
    case 'cexe':
      v68 = v485;
      runtime_convI2I(v98, v177, v249);
```

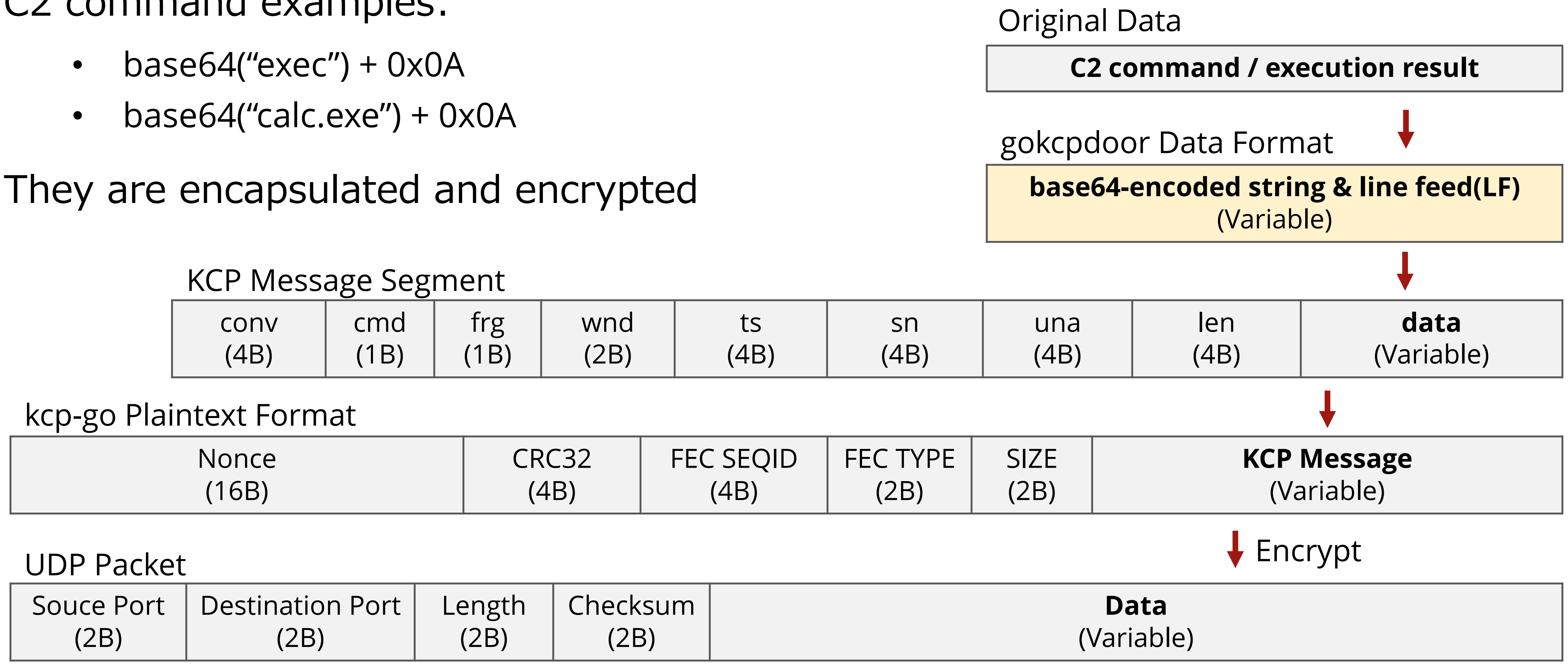
⋮ (redacted)

```
if ( *v12 != 'gifnocfi' && *v12 != 'gifnocpi' )
```

gokcpdoor implemented C2 commands example

Communication Data Format

- The format is **base64-encoded string** and **a newline code**
- C2 command examples:
  - base64("exec") + 0x0A
  - base64("calc.exe") + 0x0A
- They are encapsulated and encrypted



Original Data

C2 command / execution result

gokcpdoor Data Format

base64-encoded string & line feed(LF) (Variable)

KCP Message Segment

conv	cmd	frg	wnd	ts	sn	una	len	**data**
(4B)	(1B)	(1B)	(2B)	(4B)	(4B)	(4B)	(4B)	(Variable)

kcp-go Plaintext Format

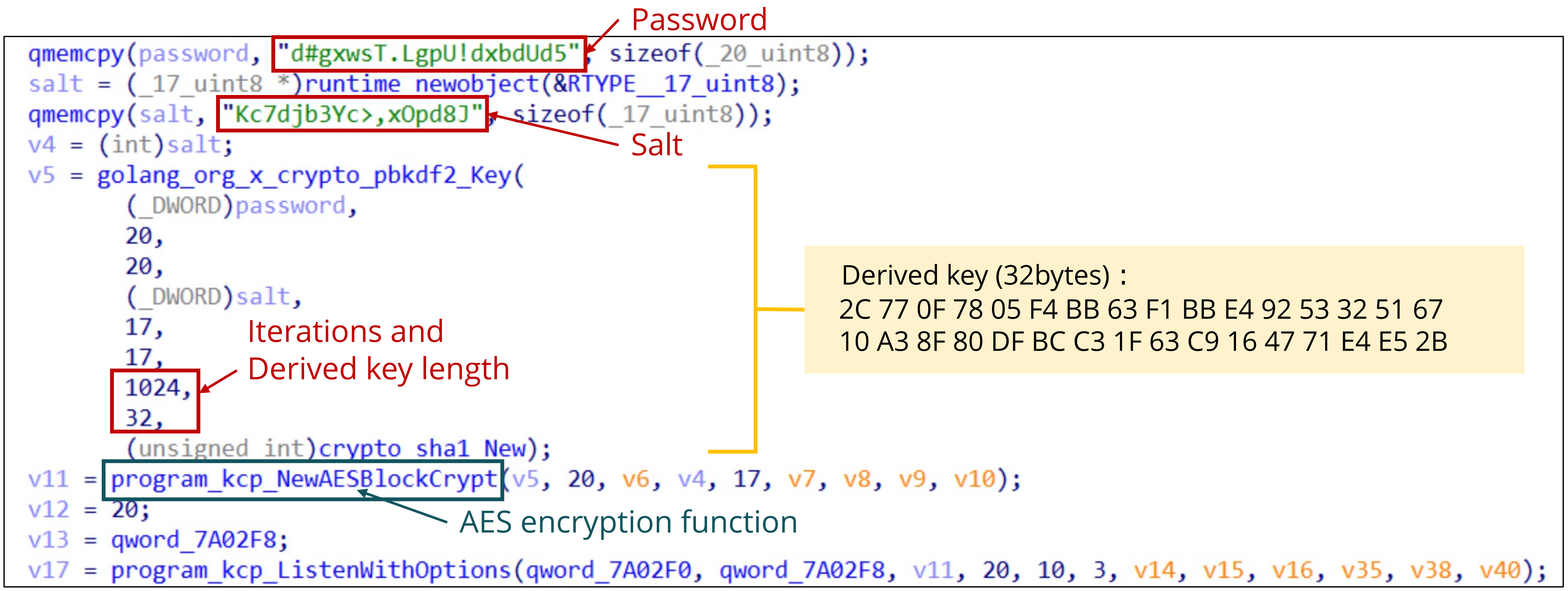
Nonce	CRC32	FEC SEQID	FEC TYPE	SIZE	**KCP Message**
(16B)	(4B)	(4B)	(2B)	(2B)	(Variable)

UDP Packet

Souce Port	Destination Port	Length	Checksum	**Data**
(2B)	(2B)	(2B)	(2B)	(Variable)

Encryption Methods by gokcpdoor

- gokcpdoor uses PBKDF2 with HMAC-SHA-1 and AES-256



```
qmemcpy(password, "d#gxwsT.LgpU!dxbdUd5", sizeof(_20_uint8));
salt = (_17_uint8 *)runtime_newobject(&RTYPE__17_uint8);
qmemcpy(salt, "Kc7djb3Yc>,xOpd8J", sizeof(_17_uint8));
v4 = (int)salt;
v5 = golang_org_x_crypto_pbkdf2_Key(
       (_DWORD)password,
       20,
       20,
       (_DWORD)salt,
       17,
       17,
       1024,
       32,
       (unsigned int)crypto_sha1_New);
v11 = program_kcp_NewAESBlockCrypt(v5, 20, v6, v4, 17, v7, v8, v9, v10);
v12 = 20;
v13 = qword_7A02F8;
v17 = program_kcp_ListenWithOptions(qword_7A02F0, qword_7A02F8, v11, 20, 10, 3, v14, v15, v16, v35, v38, v40);
```

Derived key (32bytes) :
2C 77 0F 78 05 F4 BB 63 F1 BB E4 92 53 32 51 67
10 A3 8F 80 DF BC C3 1F 63 C9 16 47 71 E4 E5 2B

gokcpdoor vs gokcprat

- **gokcprat** is similar to gokcpdoor in terms of used libraries, codes and strings contained
- However, there is a difference between **Backdoor** and **RAT**

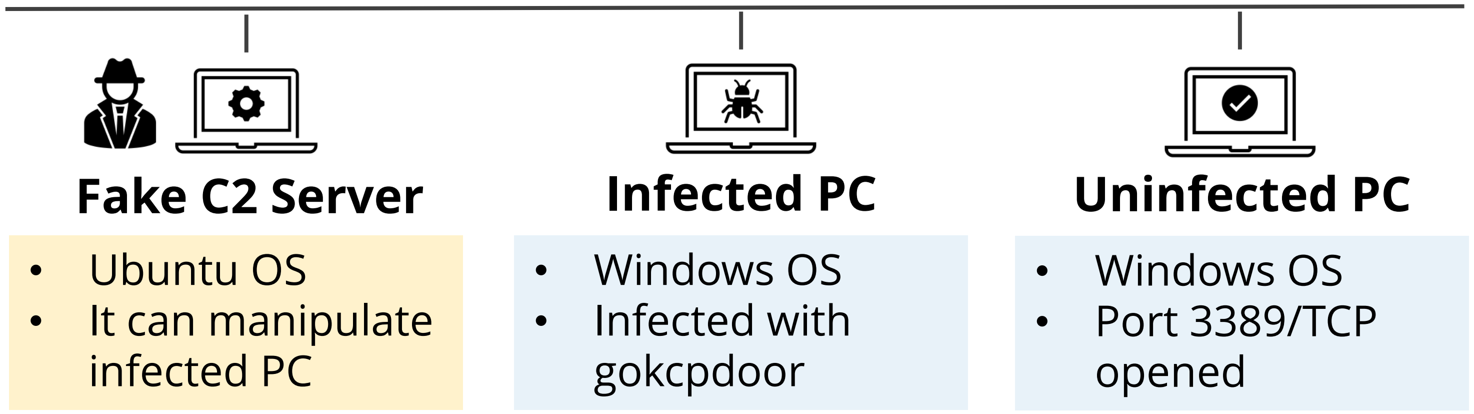
Functions	gokcpdoor	gokcprat
Build version	go1.17.5	go1.16.6
Support OS	Linux, Windows	Linux
Malware Types	Backdoor	RAT
Traffic Mode	Listener	Reverser
Encryption Methods	PBKDF2 and AES-256	PBKDF2 and AES-256
Bot commands	20 various commands	15 various commands
C2 Communication Protocols	KCP	KCP
VT First Submissions	7.12.2022	11.18.2021

04

C2 Traffic Emulation and Demonstration

Demo

- The closed environment for demonstration



Fake C2 Server

- Ubuntu OS
- It can manipulate infected PC

Infected PC

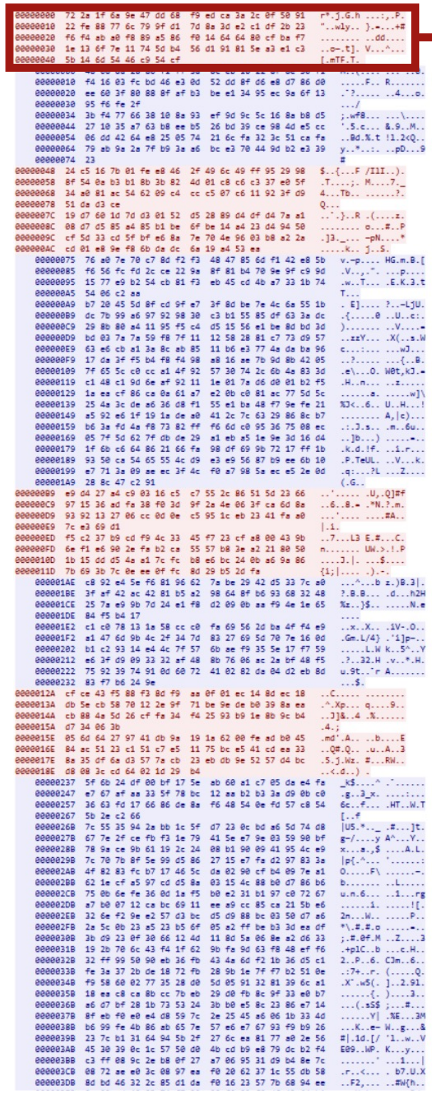
- Windows OS
- Infected with gokcpdoor

Uninfected PC

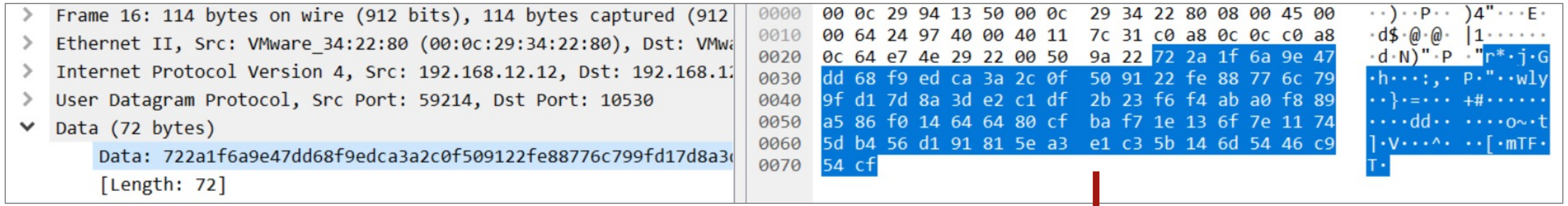
- Windows OS
- Port 3389/TCP opened

- Operations to try
  1. Get device information from infected PC
  2. Upload and download file
  3. Execute some files
  4. Set up port forwarding and connect to uninfected PC via infected PC

Captured UDP Traffic and Decryption



Captured UDP traffic

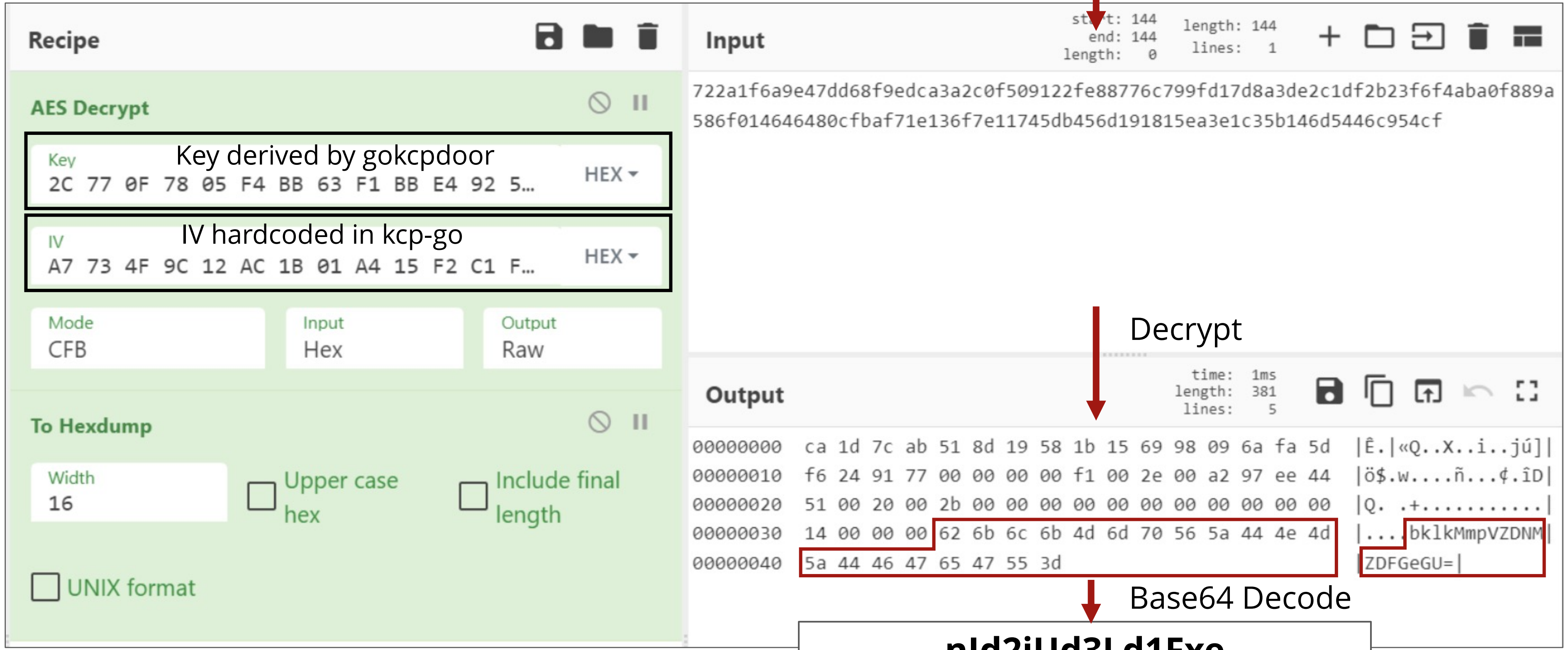


```
> Frame 16: 114 bytes on wire (912 bits), 114 bytes captured (912
> Ethernet II, Src: VMware_34:22:80 (00:0c:29:34:22:80), Dst: VMwa
> Internet Protocol Version 4, Src: 192.168.12.12, Dst: 192.168.12
> User Datagram Protocol, Src Port: 59214, Dst Port: 10530
v Data (72 bytes)
    Data: 722a1f6a9e47dd68f9edca3a2c0f509122fe88776c799fd17d8a3d
    [Length: 72]
```

```
0000  00 0c 29 94 13 50 00 0c  29 34 22 80 08 00 45 00   ··)··P·· )4"···E·
0010  00 64 24 97 40 00 40 11  7c 31 c0 a8 0c 0c c0 a8   ·d$·@·@· |1······
0020  0c 64 e7 4e 29 22 00 50  9a 22 72 2a 1f 6a 9e 47   ·d·N)"·P ·"r*·j·G
0030  dd 68 f9 ed ca 3a 2c 0f  50 91 22 fe 88 77 6c 79   ·h···:,· P·"··wly
0040  9f d1 7d 8a 3d e2 c1 df  2b 23 f6 f4 ab a0 f8 89   ··}·=··· +#······
0050  a5 86 f0 14 64 64 80 cf  ba f7 1e 13 6f 7e 11 74   ····dd·· ····o~·t
0060  5d b4 56 d1 91 81 5e a3  e1 c3 5b 14 6d 54 46 c9   ]·V···^· ··[·mTF·
0070  54 cf                                              T·
```

The first UDP packet captured by Wireshark [9]

Extract UDP Data



Recipe

AES Decrypt

Key derived by gokcpdoor

Key: 2C 77 0F 78 05 F4 BB 63 F1 BB E4 92 5… HEX

IV hardcoded in kcp-go

IV: A7 73 4F 9C 12 AC 1B 01 A4 15 F2 C1 F… HEX

Mode: CFB | Input: Hex | Output: Raw

To Hexdump

Width: 16 | ☐ Upper case hex | ☐ Include final length

☐ UNIX format

Input (start: 144, end: 144, length: 0; length: 144, lines: 1)

```
722a1f6a9e47dd68f9edca3a2c0f509122fe88776c799fd17d8a3de2c1df2b23f6f4aba0f889a
586f014646480cfbaf71e136f7e11745db456d191815ea3e1c35b146d5446c954cf
```

Decrypt

Output (time: 1ms, length: 381, lines: 5)

```
00000000  ca 1d 7c ab 51 8d 19 58 1b 15 69 98 09 6a fa 5d  |Ê.|«Q..X..i..jú]|
00000010  f6 24 91 77 00 00 00 00 f1 00 2e 00 a2 97 ee 44  |ö$.w....ñ...¢.îD|
00000020  51 00 20 00 2b 00 00 00 00 00 00 00 00 00 00 00  |Q. .+...........|
00000030  14 00 00 00 62 6b 6c 6b 4d 6d 70 56 5a 44 4e 4d  |....bklkMmpVZDNM|
00000040  5a 44 46 47 65 47 55 3d                          |ZDFGeGU=|
```

Base64 Decode

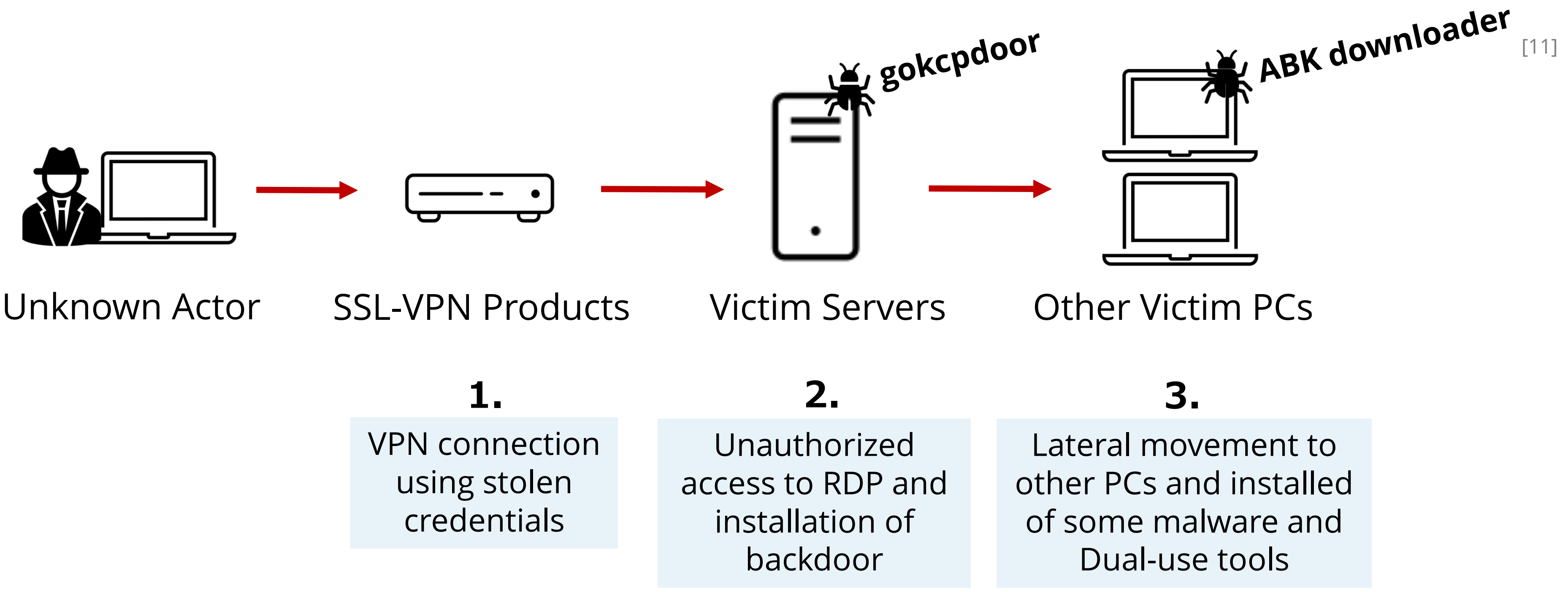
nId2jUd3Ld1Fxe
(The identifier to begin C2 operation)

The UDP data decrypted by CyberChef [10]

05

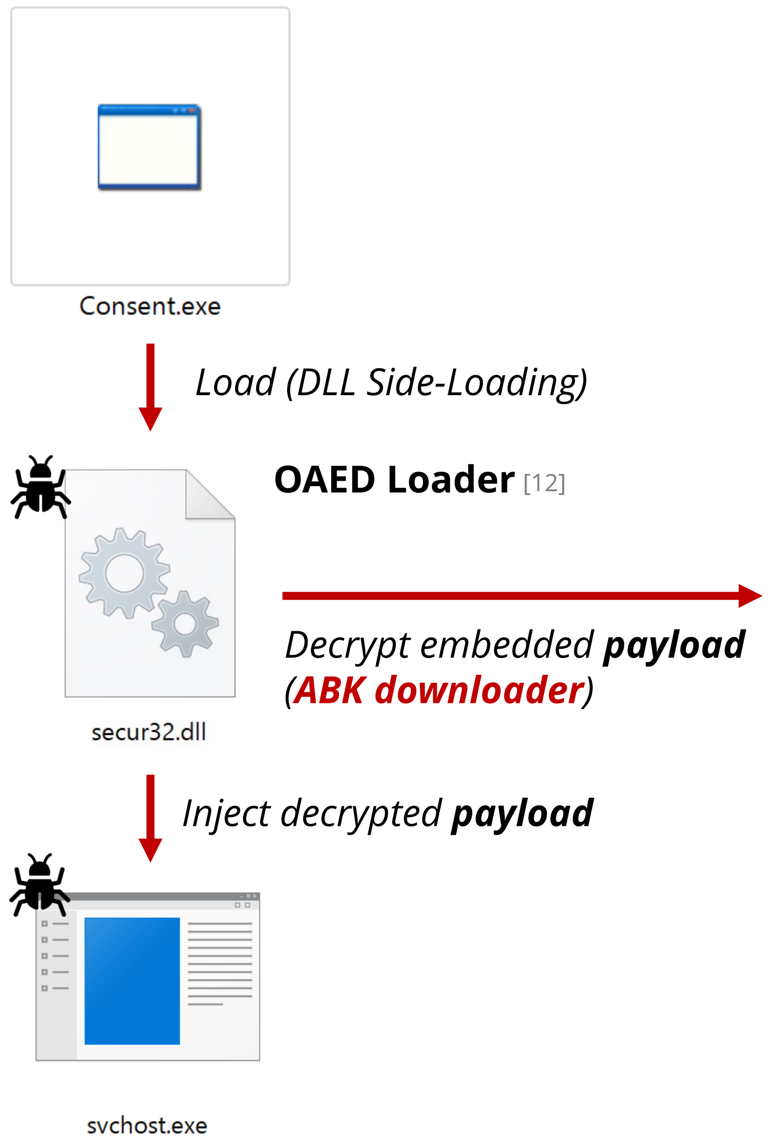
Attribution

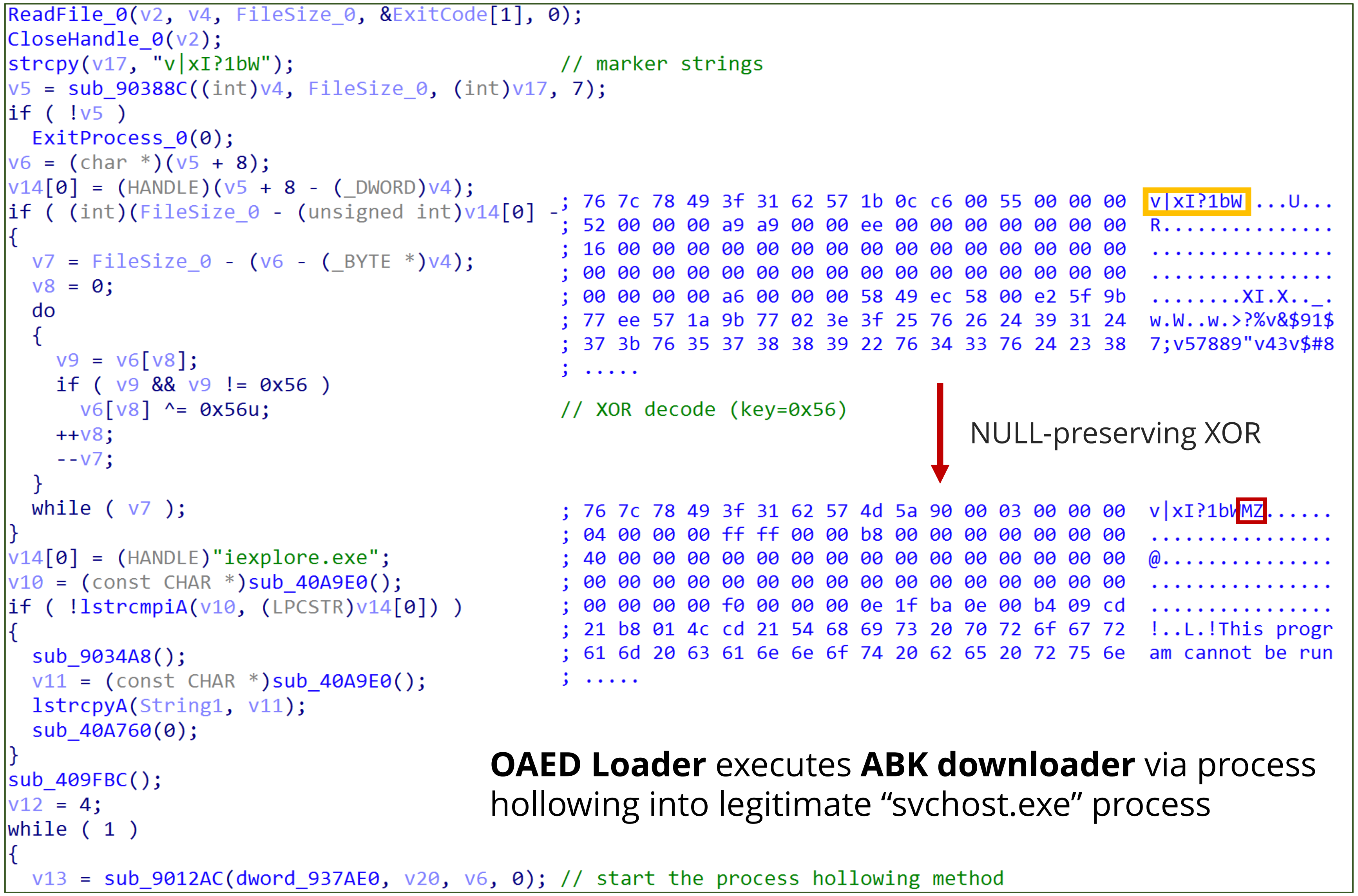
Infection Chain for gokcpdoor - One Case


gokcpdoor
ABK downloader
[11]
Unknown Actor
SSL-VPN Products
Victim Servers
Other Victim PCs
1.
VPN connection using stolen credentials
2.
Unauthorized access to RDP and installation of backdoor
3.
Lateral movement to other PCs and installed of some malware and Dual-use tools

- We confirmed gokcpdoor from March 2021 to around March 2022
- In this case, we have confirmed **ABK downloader** used by Chinese APT actors **Tick** together with **gokcpdoor**

ABK Downloader Infection Process





```
ReadFile_0(v2, v4, FileSize_0, &ExitCode[1], 0);
CloseHandle_0(v2);
strcpy(v17, "v|xI?1bW");                              // marker strings
v5 = sub_90388C((int)v4, FileSize_0, (int)v17, 7);
if ( !v5 )
  ExitProcess_0(0);
v6 = (char *)(v5 + 8);
v14[0] = (HANDLE)(v5 + 8 - (_DWORD)v4);
if ( (int)(FileSize_0 - (unsigned int)v14[0] -
{
  v7 = FileSize_0 - (v6 - (_BYTE *)v4);
  v8 = 0;
  do
  {
    v9 = v6[v8];
    if ( v9 && v9 != 0x56 )
      v6[v8] ^= 0x56u;
    ++v8;
    --v7;
  }
  while ( v7 );
}
v14[0] = (HANDLE)"iexplore.exe";
v10 = (const CHAR *)sub_40A9E0();
if ( !lstrcmpiA(v10, (LPCSTR)v14[0]) )
{
  sub_9034A8();
  v11 = (const CHAR *)sub_40A9E0();
  lstrcpyA(String1, v11);
  sub_40A760(0);
}
sub_409FBC();
v12 = 4;
while ( 1 )
{
  v13 = sub_9012AC(dword_937AE0, v20, v6, 0); // start the process hollowing method
```

```
; 76 7c 78 49 3f 31 62 57 1b 0c c6 00 55 00 00 00  v|xI?1bW...U...
; 52 00 00 00 a9 a9 00 00 ee 00 00 00 00 00 00 00  R...............
; 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
; 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
; 00 00 00 00 a6 00 00 00 58 49 ec 58 00 e2 5f 9b  ........XI.X.._.
; 77 ee 57 1a 9b 77 02 3e 3f 25 76 26 24 39 31 24  w.W..w.>?%v&$91$
; 37 3b 76 35 37 38 38 39 22 76 34 33 76 24 23 38  7;v57889"v43v$#8
; .....

// XOR decode (key=0x56)
```

NULL-preserving XOR

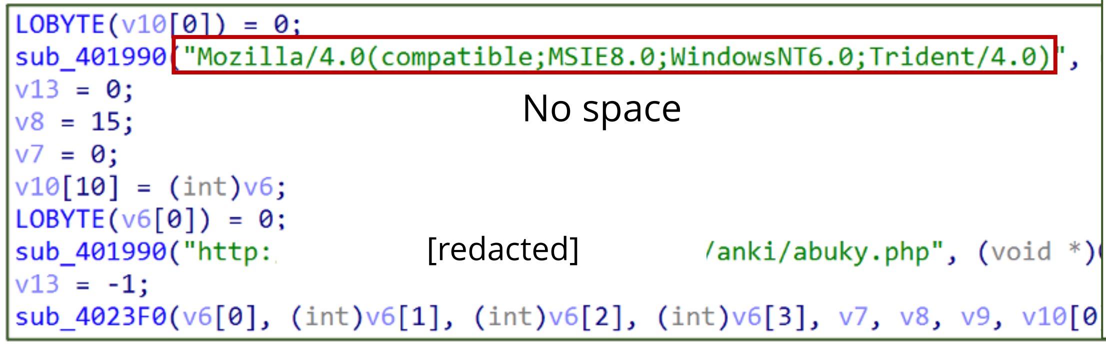
```
; 76 7c 78 49 3f 31 62 57 4d 5a 90 00 03 00 00 00  v|xI?1bWMZ......
; 04 00 00 00 ff ff 00 00 b8 00 00 00 00 00 00 00  ................
; 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  @...............
; 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
; 00 00 00 00 f0 00 00 00 0e 1f ba 0e 00 b4 09 cd  ................
; 21 b8 01 4c cd 21 54 68 69 73 20 70 72 6f 67 72  !..L.!This progr
; 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 6e  am cannot be run
; .....
```

OAED Loader executes **ABK downloader** via process hollowing into legitimate “svchost.exe” process

ABK Downloader Functions

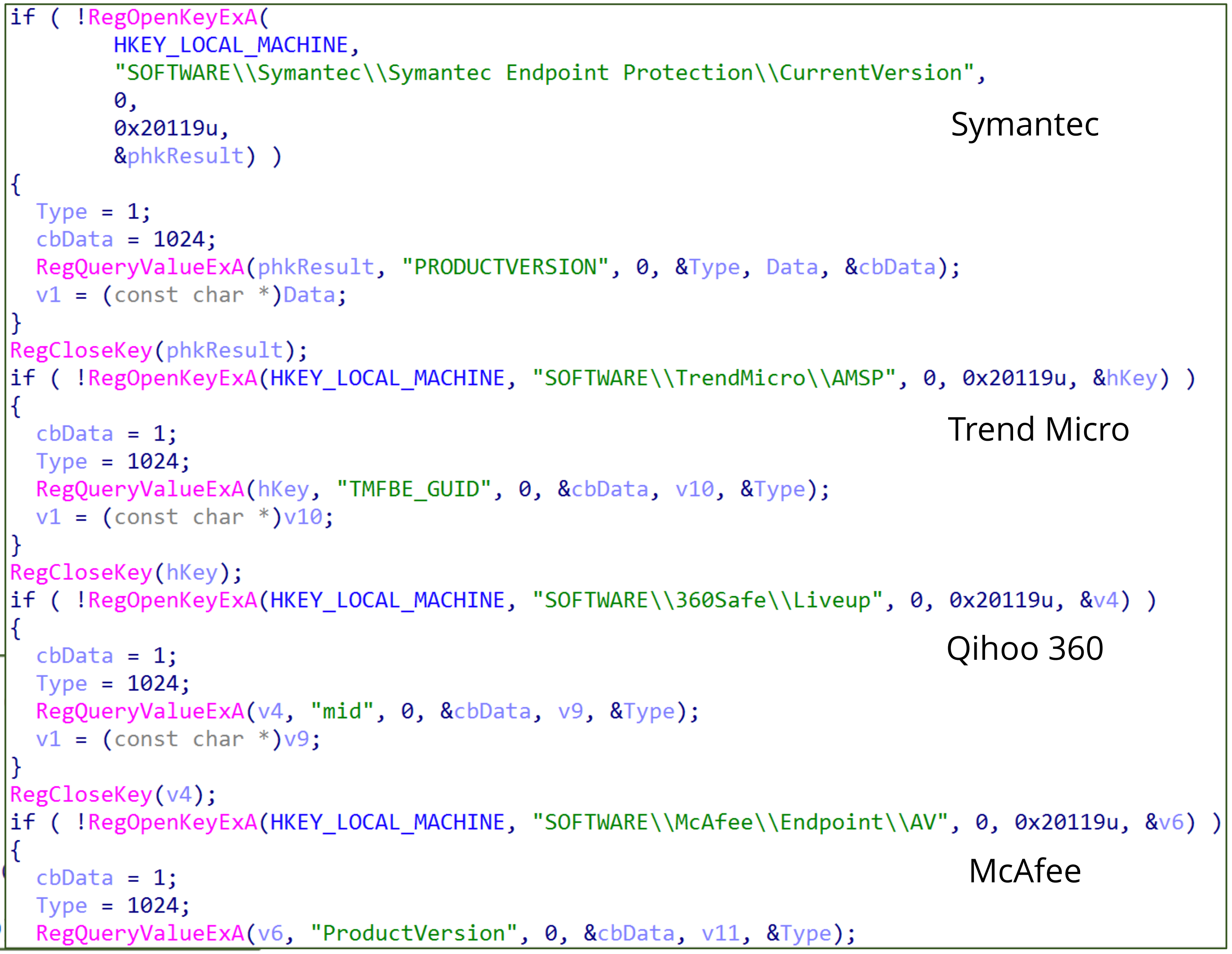
It mainly has four features:

- Detects some **security products**
- Collects MAC address, System information and AV information and send to C2 servers using **no space User-Agent**
- Executes only during **working hours (8:00-18:00)** using GetLocalTime API
- Uses legitimate websites for C2 servers and downloads next malware



```
LOBYTE(v10[0]) = 0;
sub_401990("Mozilla/4.0(compatible;MSIE8.0;WindowsNT6.0;Trident/4.0)",
v13 = 0;
v8 = 15;
v7 = 0;
v10[10] = (int)v6;
LOBYTE(v6[0]) = 0;
sub_401990("http:[redacted]/anki/abuky.php", (void *)(
v13 = -1;
sub_4023F0(v6[0], (int)v6[1], (int)v6[2], (int)v6[3], v7, v8, v9, v10[0
```

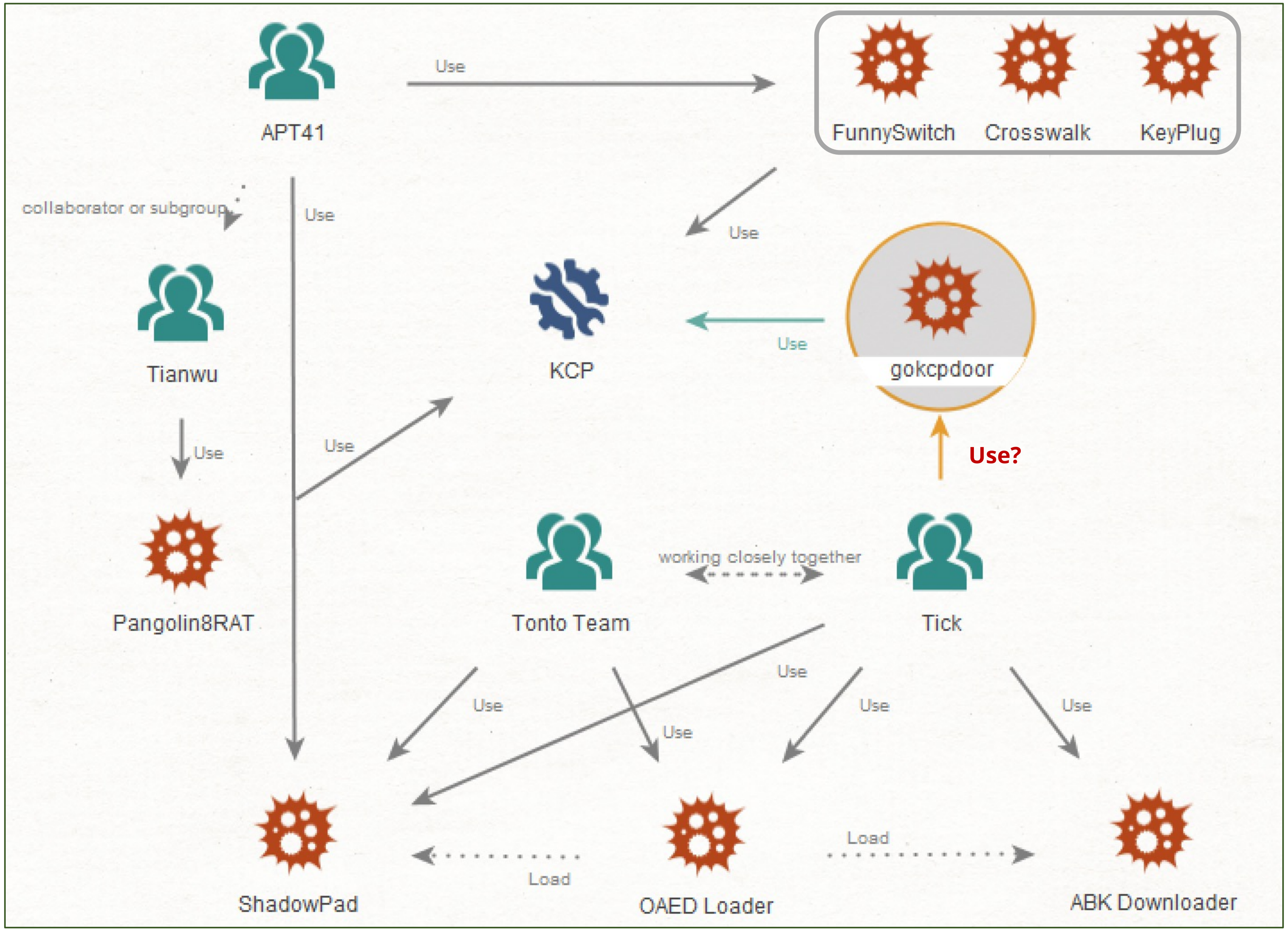
Specific User-Agent and C2 server



```
if ( !RegOpenKeyExA(
        HKEY_LOCAL_MACHINE,
        "SOFTWARE\\Symantec\\Symantec Endpoint Protection\\CurrentVersion",
        0,
        0x20119u,
        &phkResult) )
{
  Type = 1;
  cbData = 1024;
  RegQueryValueExA(phkResult, "PRODUCTVERSION", 0, &Type, Data, &cbData);
  v1 = (const char *)Data;
}
RegCloseKey(phkResult);
if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\TrendMicro\\AMSP", 0, 0x20119u, &hKey) )
{
  cbData = 1;
  Type = 1024;
  RegQueryValueExA(hKey, "TMFBE_GUID", 0, &cbData, v10, &Type);
  v1 = (const char *)v10;
}
RegCloseKey(hKey);
if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\360Safe\\Liveup", 0, 0x20119u, &v4) )
{
  cbData = 1;
  Type = 1024;
  RegQueryValueExA(v4, "mid", 0, &cbData, v9, &Type);
  v1 = (const char *)v9;
}
RegCloseKey(v4);
if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\McAfee\\Endpoint\\AV", 0, 0x20119u, &v6) )
{
  cbData = 1;
  Type = 1024;
  RegQueryValueExA(v6, "ProductVersion", 0, &cbData, v11, &Type);
```

Detect of specific antivirus products

Relationship Between APT Actors and Malware



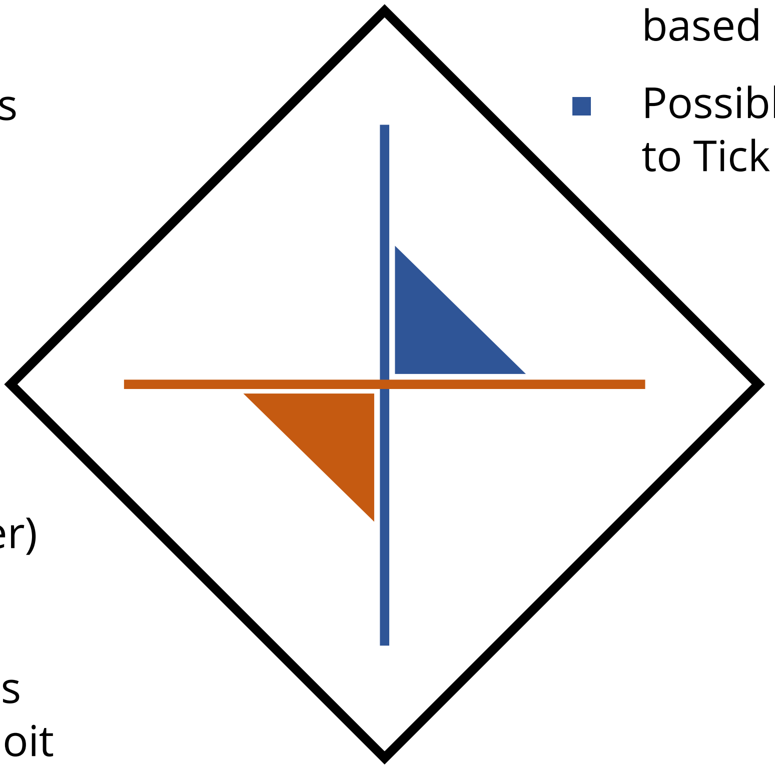
Diamond Model of gokcpdoor Campaign

Technical Axis

- A part of custom malware overlap with known Tick tools
- Intrusion via SSL-VPN products

Social-Political Axis

- National security interests and concerns



ADVERSARY

- Believed to be based in China
- Possible relationship to Tick and APT41

CAPABILITY

- Custom malware (gokcpdoor, ABK downloader, OAED Loader)
- Using KCP Protocol
- Intrusion via SSL-VPN products with stolen certificates or exploit

INFRASTRUCTURE

- Heavy usage of IP address not domain
- Multiple VPS and Hosting Service Companies in East Asia

VICTIM

- Targeting Japanese companies
- Manufacturing, Academic institutions

06

Countermeasures of Threat

Prevention and Detection (1/2)

- For KCP traffic
  - IDS/IPS/FW products
    - **Checks** for **UDP traffic** (e.g., KCP traffic has detected as “**unknown-udp**” in Palo Alto Networks Products)
    - However, it may be difficult to identify only KCP traffic in normal traffic
  - Splunk [13]
    - Using **Splunk Stream** (details will be introduced in the next topic and Appendix B)
  - Wireshark [9]
    - Using **KCP dissector** to analyze suspicious UDP traffic (details will be introduced in the Appendix C)

Prevention and Detection (2/2)

- For gokcpdoor
  - Yara (for Linux and Windows) [14]
    - **Detected** by Yara rule (details will be introduced in the Appendix D)
  - Autoruns (for Windows) [15]
    - Checking suspicious **AutoStart Extensibility Points (ASEPs)**
  - Sysmon (for Linux and Windows) [16] [17]
    - **Create Process** and **Network Connect** events are **recorded** (details on later slide)
  - EDR products
    - Shell commands execution can be traced by **process tree** (details on later slide)

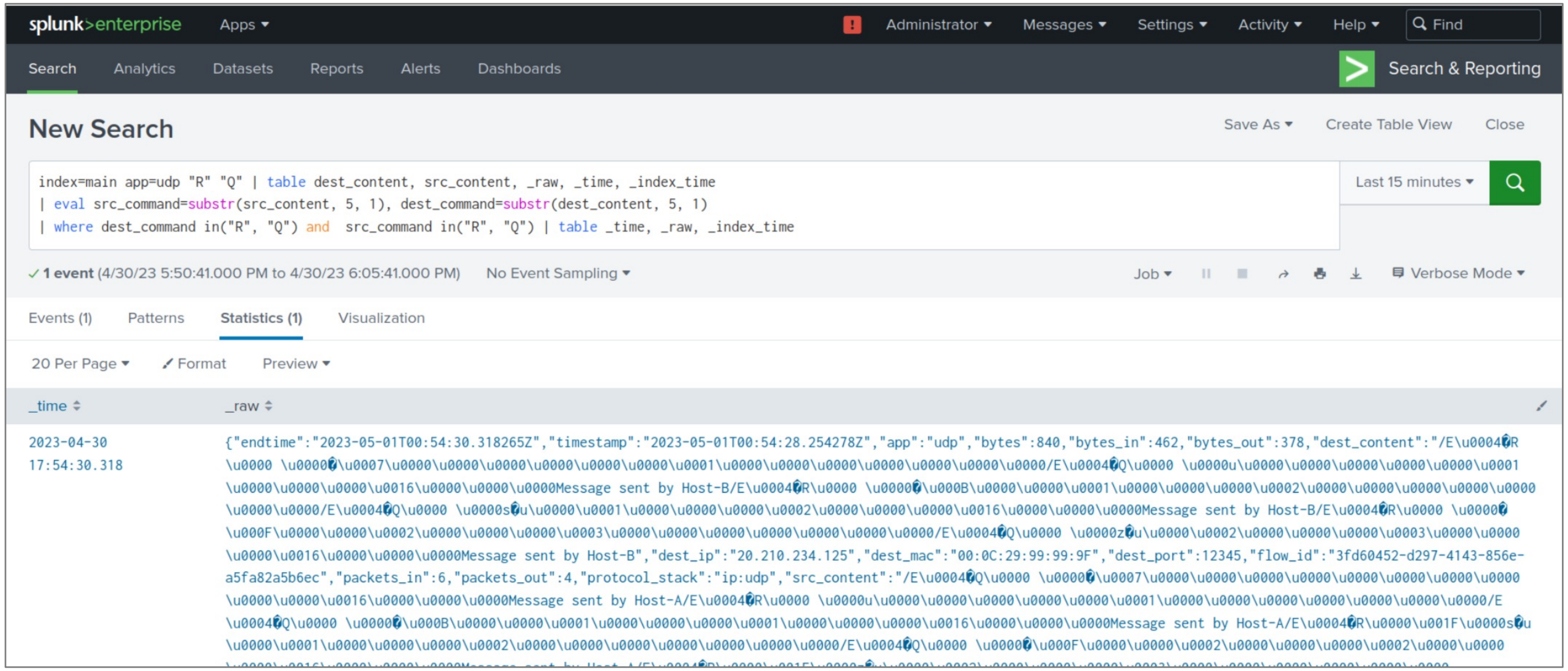
Searching for KCP Traffic on Splunk

- We can search KCP traffic using by Splunk and Splunk Stream

Experimental Search Query

＊ Please note that this search query is expected to be delayed due to log volume.

```
index=main app=udp "R" "Q" | table dest_content, src_content, _raw, _time
| eval src_command=substr(src_content, 5, 1), dest_command=substr(dest_content, 5, 1)
| where dest_command in("R", "Q") and  src_command in("R", "Q") | table _time, _raw
```



New Search

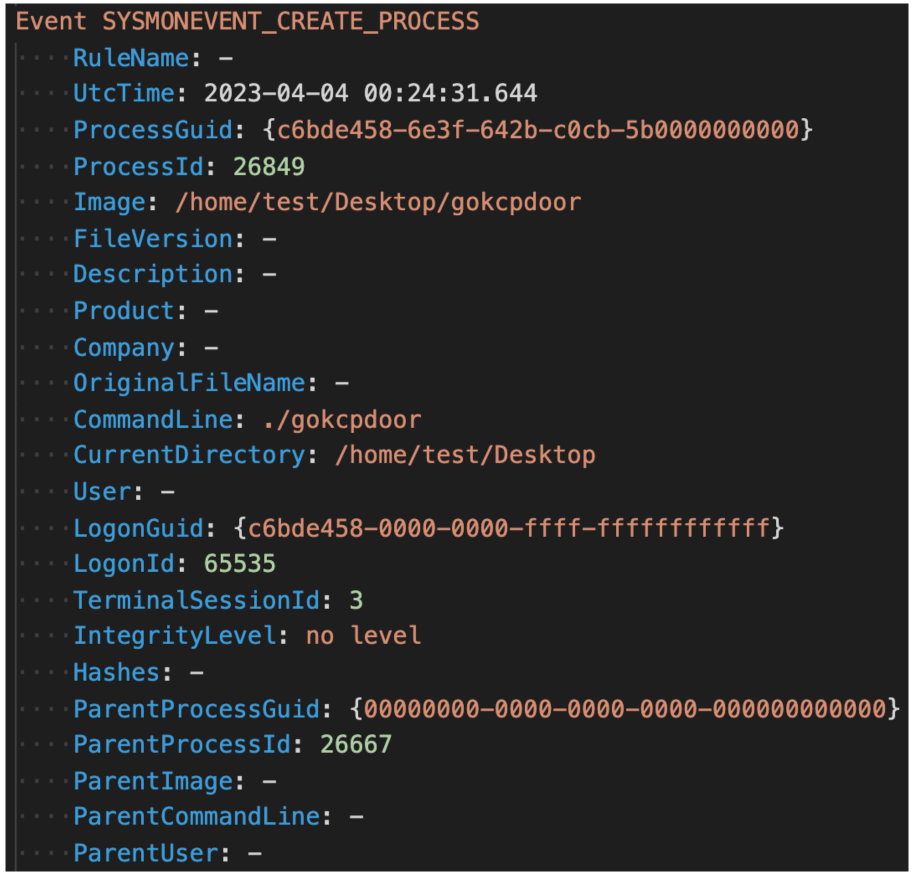
```
index=main app=udp "R" "Q" | table dest_content, src_content, _raw, _time, _index_time
| eval src_command=substr(src_content, 5, 1), dest_command=substr(dest_content, 5, 1)
| where dest_command in("R", "Q") and  src_command in("R", "Q") | table _time, _raw, _index_time
```

✓ 1 event (4/30/23 5:50:41.000 PM to 4/30/23 6:05:41.000 PM) No Event Sampling

_time	_raw
2023-04-30 17:54:30.318	{"endtime":"2023-05-01T00:54:30.318265Z","timestamp":"2023-05-01T00:54:28.254278Z","app":"udp","bytes":840,"bytes_in":462,"bytes_out":378,"dest_content":"/E\u0004�R \u0000 \u0000�\u0007\u0000\u0000\u0000\u0000\u0000\u0000\u0001\u0000\u0000\u0000\u0000\u0000\u0000\u0000/E\u0004�Q\u0000 \u0000u\u0000\u0000\u0000\u0000\u0000\u0000\u0001 \u0000\u0000\u0000\u0016\u0000\u0000\u0000Message sent by Host-B/E\u0004�R\u0000 \u0000�\u000B\u0000\u0000\u0001\u0000\u0000\u0000\u0002\u0000\u0000\u0000\u0000\u0000 \u0000\u0000/E\u0004�Q\u0000 \u0000s�u\u0000\u0001\u0000\u0000\u0000\u0002\u0000\u0000\u0000\u0016\u0000\u0000\u0000Message sent by Host-B/E\u0004�R\u0000 \u0000� \u000F\u0000\u0000\u0002\u0000\u0000\u0000\u0003\u0000\u0000\u0000\u0000\u0000\u0000\u0000/E\u0004�Q\u0000 \u0000z�u\u0000\u0002\u0000\u0000\u0000\u0003\u0000\u0000 \u0000\u0016\u0000\u0000\u0000Message sent by Host-B","dest_ip":"20.210.234.125","dest_mac":"00:0C:29:99:99:9F","dest_port":12345,"flow_id":"3fd60452-d297-4143-856e-a5fa82a5b6ec","packets_in":6,"packets_out":4,"protocol_stack":"ip:udp","src_content":"/E\u0004�Q\u0000 \u0000�\u0007\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000 \u0000\u0000\u0016\u0000\u0000\u0000Message sent by Host-A/E\u0004�R\u0000 \u0000u\u0000\u0000\u0000\u0000\u0000\u0001\u0000\u0000\u0000\u0000\u0000\u0000\u0000/E \u0004�Q\u0000 \u0000�\u000B\u0000\u0000\u0001\u0000\u0000\u0000\u0001\u0000\u0000\u0000\u0016\u0000\u0000\u0000Message sent by Host-A/E\u0004�R\u0000\u001F\u0000s�u \u0000\u0001\u0000\u0000\u0000\u0002\u0000\u0000\u0000\u0000\u0000\u0000\u0000/E\u0004�Q\u0000 \u0000�\u000F\u0000\u0000\u0002\u0000\u0000\u0000\u0002\u0000\u0000

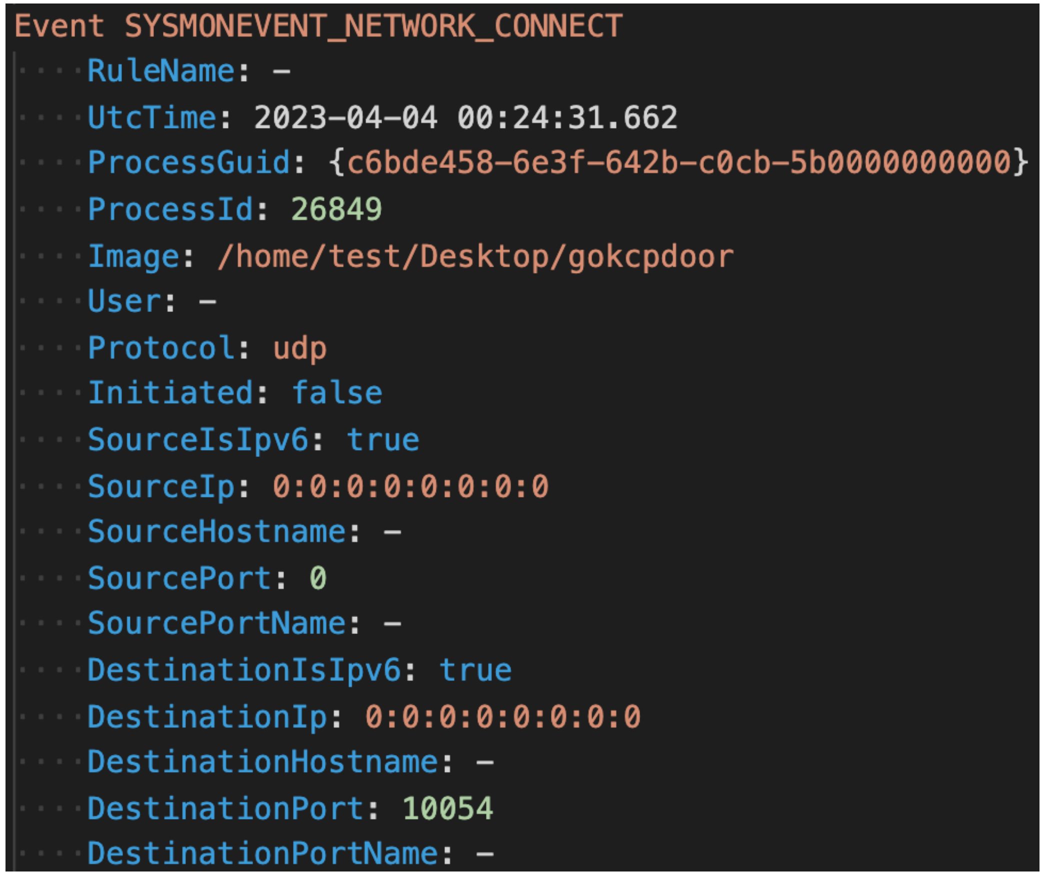
Detected with Sysmon for gokcpdoor Linux

- The following logs can be obtained by using "**Sysmon For Linux"** and "sysmonLogView" tools



```
Event SYSMONEVENT_CREATE_PROCESS
    RuleName: -
    UtcTime: 2023-04-04 00:24:31.644
    ProcessGuid: {c6bde458-6e3f-642b-c0cb-5b0000000000}
    ProcessId: 26849
    Image: /home/test/Desktop/gokcpdoor
    FileVersion: -
    Description: -
    Product: -
    Company: -
    OriginalFileName: -
    CommandLine: ./gokcpdoor
    CurrentDirectory: /home/test/Desktop
    User: -
    LogonGuid: {c6bde458-0000-0000-ffff-ffffffffffff}
    LogonId: 65535
    TerminalSessionId: 3
    IntegrityLevel: no level
    Hashes: -
    ParentProcessGuid: {00000000-0000-0000-0000-000000000000}
    ParentProcessId: 26667
    ParentImage: -
    ParentCommandLine: -
    ParentUser: -
```

Create Process (Event ID 1)

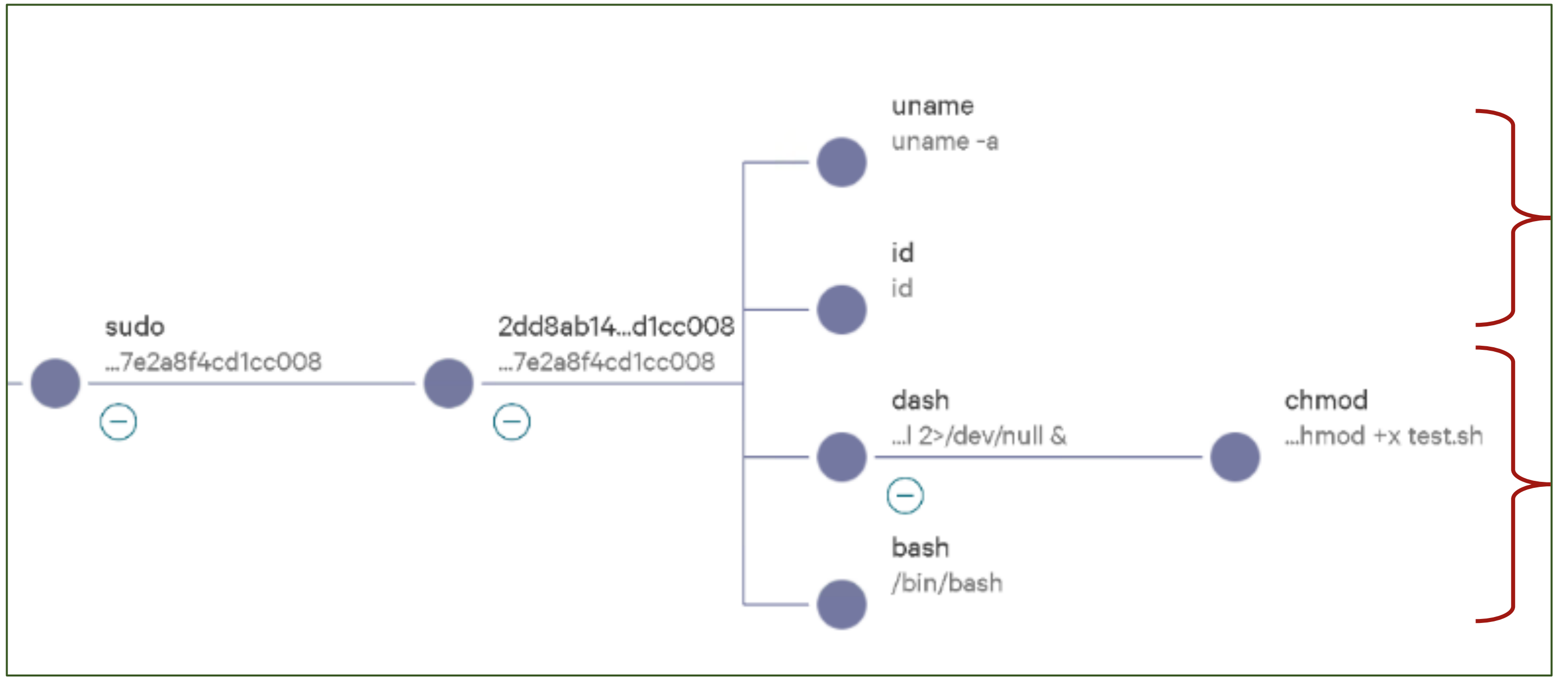


```
Event SYSMONEVENT_NETWORK_CONNECT
    RuleName: -
    UtcTime: 2023-04-04 00:24:31.662
    ProcessGuid: {c6bde458-6e3f-642b-c0cb-5b0000000000}
    ProcessId: 26849
    Image: /home/test/Desktop/gokcpdoor
    User: -
    Protocol: udp
    Initiated: false
    SourceIsIpv6: true
    SourceIp: 0:0:0:0:0:0:0:0
    SourceHostname: -
    SourcePort: 0
    SourcePortName: -
    DestinationIsIpv6: true
    DestinationIp: 0:0:0:0:0:0:0:0
    DestinationHostname: -
    DestinationPort: 10054
    DestinationPortName: -
```

Network Connect (Event ID 3)

EDR Tracing for gokcpdoor Linux

- EDR traces suspicious behavior gokcpdoor process



CrowdStrike Falcon Graphs Process Tree

The behavior caused by "getos" and "whoami" C2 commands

The behavior caused by "upload" and "shell" C2 commands
*We uploaded "test.sh" and executed it

Conclusion

- **gokcpdoor** is a backdoor malware coded on golang using **KCP protocol** for C2 communication
- Sharing about **detection and prevention** methods to protect similar attacks
- **Hunting threats** by using Yara, Sysmon, EDR products, Splunk SPL query and checking ASEPs
- We have introduced a possible relationship Chinese APT actors **Tick** or APT41 about gokcpdoor, but attribution is more getting hard
- Information sharing will help in future threat prevention

Appendix A – References

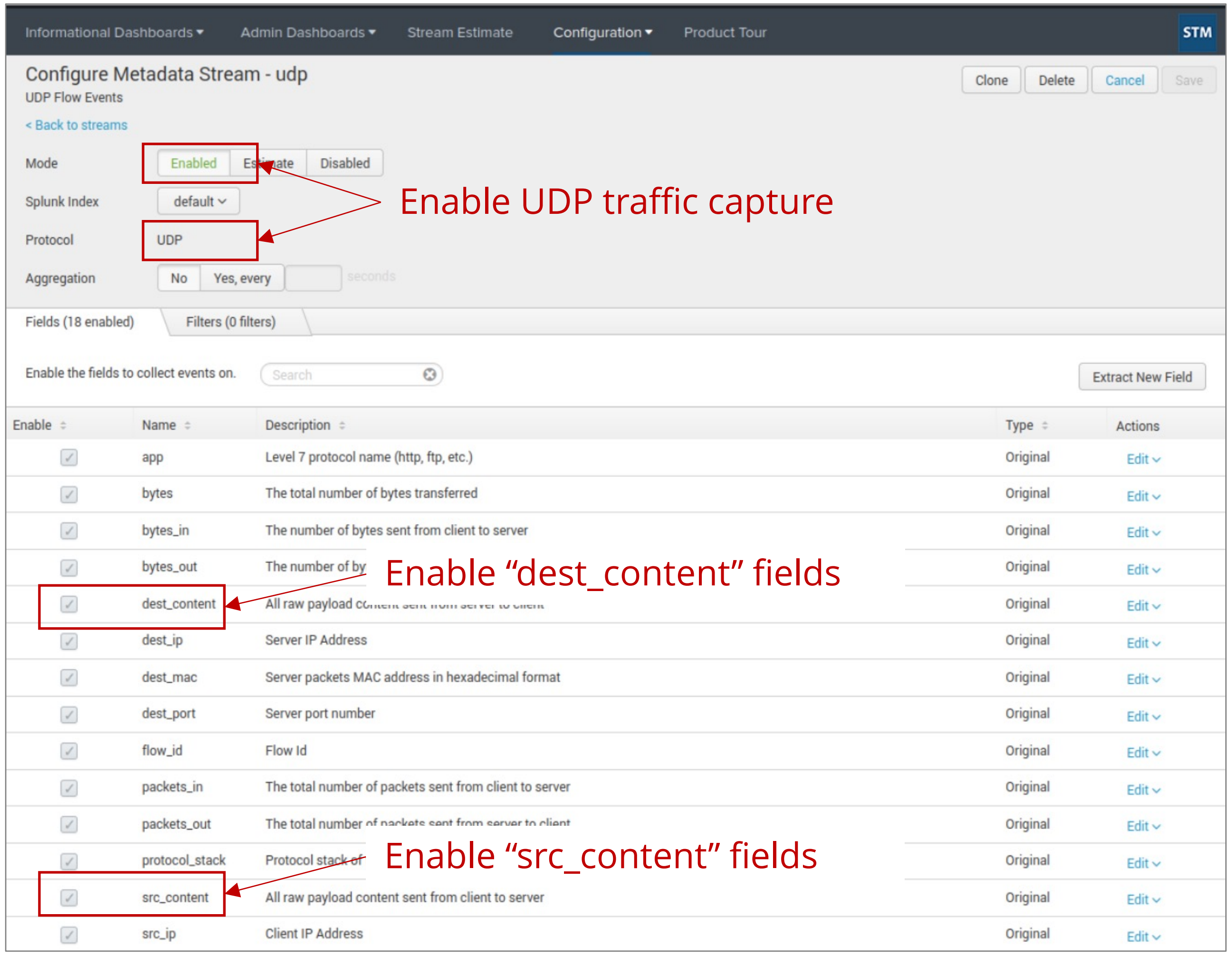
1. https://www.virustotal.com/gui/file/2dd8ab1493a97e0a4416e077d6ce1c35c7b2d8749592b319a7e2a8f4cd1cc008/detection/f-2dd8ab1493a97e0a4416e077d6ce1c35c7b2d8749592b319a7e2a8f4cd1cc008-1657604975
2. https://github.com/skywind3000/kcp
3. https://pkg.go.dev/github.com/xtaci/kcp-go
4. https://www.ptsecurity.com/ww-en/analytics/pt-esc-threat-intelligence/higaisa-or-winnti-apt-41-backdoors-old-and-new/
5. https://i.blackhat.com/Asia-22/Thursday-Materials/AS-22-LeonSilvia-NextGenPlugXShadowPad.pdf
6. https://ics-cert.kaspersky.com/publications/reports/2021/12/16/pseudomanuscrypt-a-mass-scale-spyware-attack-campaign/
7. https://www.mandiant.com/resources/blog/apt41-us-state-governments
8. https://github.com/qchencc/kcp-dotnet/blob/master/Source/Network/KCP.cs
9. https://www.wireshark.org/
10. https://gchq.github.io/CyberChef/
11. https://nao-sec.org/2020/01/an-overhead-view-of-the-royal-road.html
12. https://www.macnica.co.jp/business/security/manufacturers/files/mpressioncss_ta_report_2020_5_en.pdf
13. https://docs.splunk.com/Splexicon:SPL
14. https://virustotal.github.io/yara/
15. https://learn.microsoft.com/en-us/sysinternals/downloads/autoruns
16. https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon
17. https://github.com/Sysinternals/SysmonForLinux

Appendix A – References for OSS Libraries Using KCP

Category	Repository Name	URL
Libraries / Frameworks providing KCP communication functionality	kcp-go	https://github.com/xtaci/kcp-go
	gouxp	https://github.com/shaoyuan1943/gouxp
	HP-Socket	https://github.com/ldcsaa/HP-Socket
	asio_kcp	https://github.com/libinzhangyuan/asio_kcp
	yasio	https://github.com/yasio/yasio
	kcp-cpp	https://github.com/Unit-X/kcp-cpp
	xkcptun	https://github.com/liudf0716/xkcptun
	KCP	https://github.com/KumoKyaku/KCP
	kcp-dotnet	https://github.com/qchencc/kcp-dotnet
	java-kcp	https://github.com/l42111996/java-Kcp
	node-kcp	https://github.com/leenjewel/node-kcp
	kcp-rs	https://github.com/en/kcp-rs
Network tunnel tool / Proxy software	frp	https://github.com/fatedier/frp
	kcptun	https://github.com/xtaci/kcptun
	gost	https://github.com/ginuerzh/gost
	v2ray	https://github.com/v2fly/v2ray-core
	dog-tunnel	https://github.com/vzex/dog-tunnel
Framework for game	ET	https://github.com/egametang/ET

Appendix B – Splunk Steam Settings (1/2)

- In order to search KCP traffic, we must enable UDP traffic capture and content recording in Splunk Stream



Configure Metadata Stream - udp

UDP Flow Events

< Back to streams

Mode	[✓ Enabled] Estimate Disabled
Splunk Index	default
Protocol	UDP
Aggregation	No Yes, every ___ seconds

Enable UDP traffic capture

Fields (18 enabled) Filters (0 filters)

Enable the fields to collect events on.

Enable	Name	Description	Type	Actions
✓	app	Level 7 protocol name (http, ftp, etc.)	Original	Edit
✓	bytes	The total number of bytes transferred	Original	Edit
✓	bytes_in	The number of bytes sent from client to server	Original	Edit
✓	bytes_out	The number of by	Original	Edit
✓	dest_content	All raw payload content sent from server to client	Original	Edit
✓	dest_ip	Server IP Address	Original	Edit
✓	dest_mac	Server packets MAC address in hexadecimal format	Original	Edit
✓	dest_port	Server port number	Original	Edit
✓	flow_id	Flow Id	Original	Edit
✓	packets_in	The total number of packets sent from client to server	Original	Edit
✓	packets_out	The total number of packets sent from server to client	Original	Edit
✓	protocol_stack	Protocol stack of	Original	Edit
✓	src_content	All raw payload content sent from client to server	Original	Edit
✓	src_ip	Client IP Address	Original	Edit

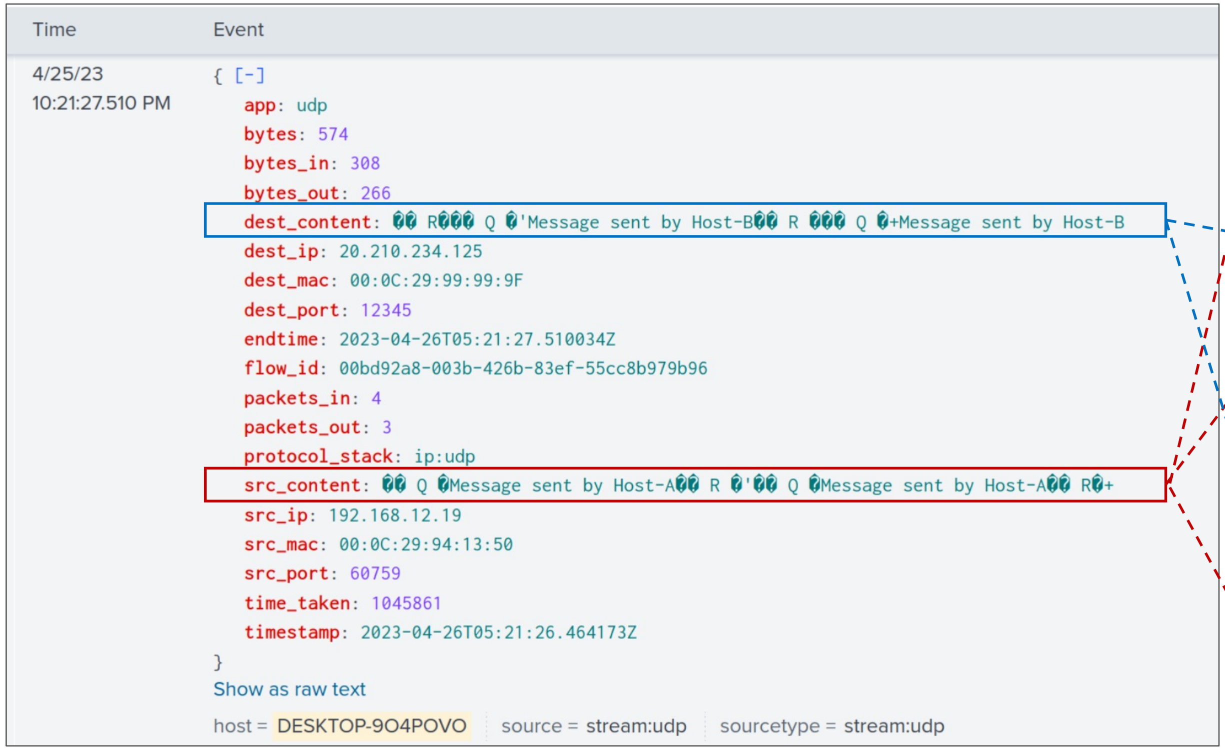
Enable "dest_content" fields

Enable "src_content" fields

* We recommend estimate the amount of log before setting these up in production.

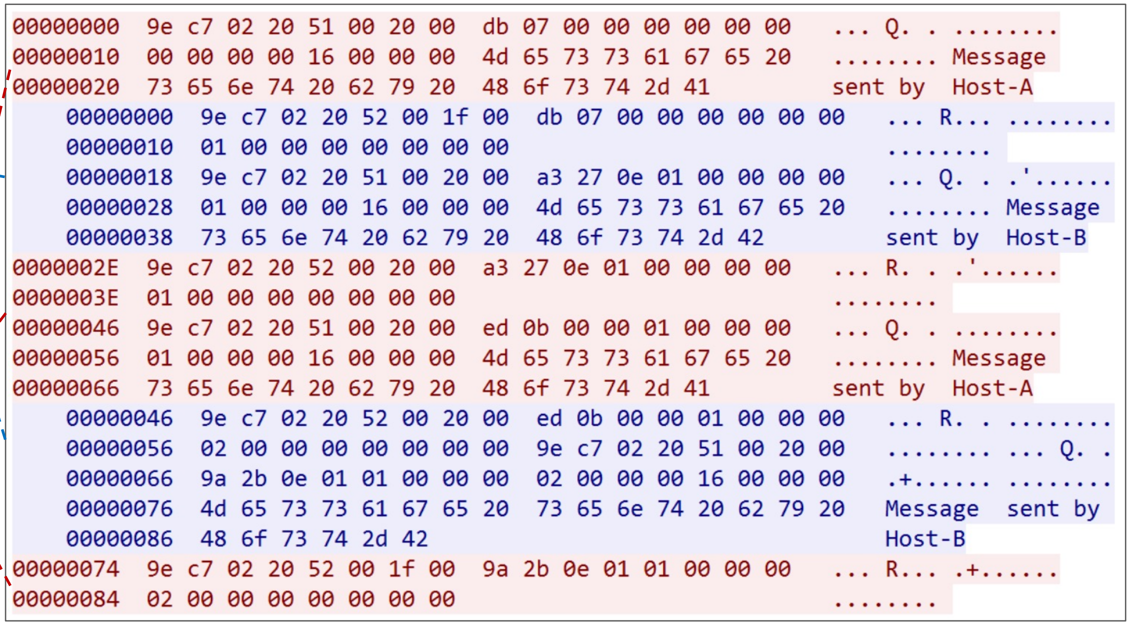
Appendix B – Splunk Steam Settings (2/2)

- After completing the settings, we will see the following logs:



```
Time                 Event
4/25/23              { [-]
10:21:27.510 PM         app: udp
                        bytes: 574
                        bytes_in: 308
                        bytes_out: 266
                        dest_content: �� R��� Q �'Message sent by Host-B�� R ��� Q �+Message sent by Host-B
                        dest_ip: 20.210.234.125
                        dest_mac: 00:0C:29:99:99:9F
                        dest_port: 12345
                        endtime: 2023-04-26T05:21:27.510034Z
                        flow_id: 00bd92a8-003b-426b-83ef-55cc8b979b96
                        packets_in: 4
                        packets_out: 3
                        protocol_stack: ip:udp
                        src_content: �� Q �Message sent by Host-A�� R �'�� Q �Message sent by Host-A�� R�+
                        src_ip: 192.168.12.19
                        src_mac: 00:0C:29:94:13:50
                        src_port: 60759
                        time_taken: 1045861
                        timestamp: 2023-04-26T05:21:26.464173Z
                     }
                     Show as raw text
                     host = DESKTOP-9O4POVO    source = stream:udp    sourcetype = stream:udp
```

Example of log recorded by Splunk Stream



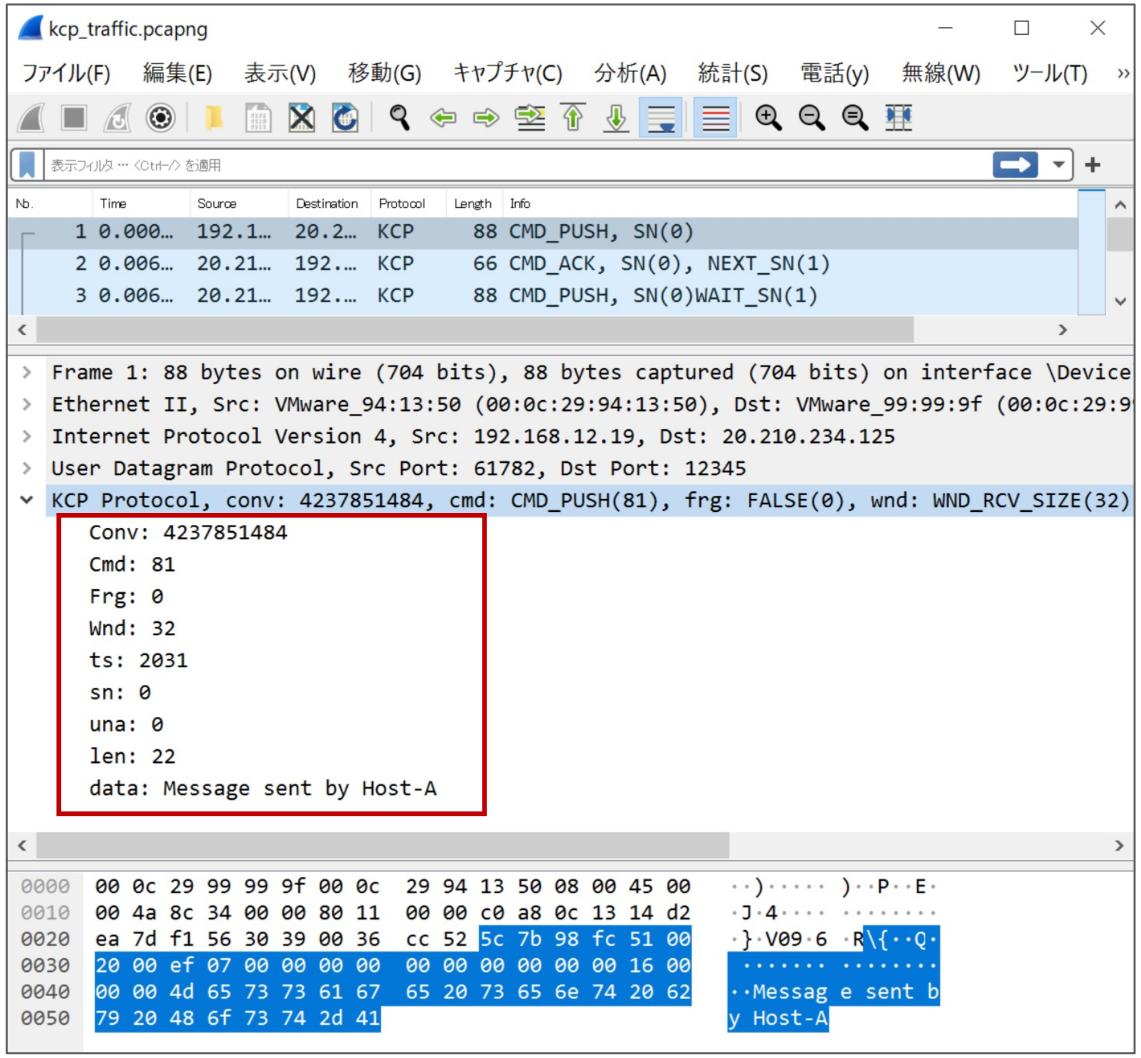
```
00000000  9e c7 02 20 51 00 20 00  db 07 00 00 00 00 00 00   ... Q. . ........
00000010  00 00 00 00 16 00 00 00  4d 65 73 73 61 67 65 20   ........ Message
00000020  73 65 6e 74 20 62 79 20  48 6f 73 74 2d 41         sent by  Host-A
    00000000  9e c7 02 20 52 00 1f 00  db 07 00 00 00 00 00 00   ... R... ........
    00000010  01 00 00 00 00 00 00 00                            ........
    00000018  9e c7 02 20 51 00 20 00  a3 27 0e 01 00 00 00 00   ... Q. . .'......
    00000028  01 00 00 00 16 00 00 00  4d 65 73 73 61 67 65 20   ........ Message
    00000038  73 65 6e 74 20 62 79 20  48 6f 73 74 2d 42         sent by  Host-B
0000002E  9e c7 02 20 52 00 20 00  a3 27 0e 01 00 00 00 00   ... R. . .'......
0000003E  01 00 00 00 00 00 00 00                            ........
00000046  9e c7 02 20 51 00 20 00  ed 0b 00 00 01 00 00 00   ... Q. . ........
00000056  01 00 00 00 16 00 00 00  4d 65 73 73 61 67 65 20   ........ Message
00000066  73 65 6e 74 20 62 79 20  48 6f 73 74 2d 41         sent by  Host-A
    00000046  9e c7 02 20 52 00 20 00  ed 0b 00 00 01 00 00 00   ... R. . ........
    00000056  02 00 00 00 00 00 00 00  9e c7 02 20 51 00 20 00   ........ ... Q. .
    00000066  9a 2b 0e 01 01 00 00 00  02 00 00 00 16 00 00 00   .+...... ........
    00000076  4d 65 73 73 61 67 65 20  73 65 6e 74 20 62 79 20   Message  sent by
    00000086  48 6f 73 74 2d 42                                  Host-B
00000074  9e c7 02 20 52 00 1f 00  9a 2b 0e 01 01 00 00 00   ... R... .+......
00000084  02 00 00 00 00 00 00 00                            ........
```

Example of UDP traffic

Appendix C – KCP Dissector for Wireshark

- Reference to KCP dissector
  - https://github.com/cfadmin-cn/kcp_dissector
  - https://github.com/chosen0ne/kcp-dissector-plugin
  - https://github.com/xtaci/kcp-go/tree/master/wireshark
  - https://github.com/yinkaisheng/kcp_rtp_dissector/

* We recommend deliberate testing and tuning prior to implementation in any production system



kcp_traffic.pcapng
ファイル(F) 編集(E) 表示(V) 移動(G) キャプチャ(C) 分析(A) 統計(S) 電話(y) 無線(W) ツール(T)
表示フィルタ … <Ctrl-/> を適用

```
No.  Time      Source   Destination  Protocol  Length  Info
  1  0.000…    192.1…   20.2…        KCP       88      CMD_PUSH, SN(0)
  2  0.006…    20.21…   192.…        KCP       66      CMD_ACK, SN(0), NEXT_SN(1)
  3  0.006…    20.21…   192.…        KCP       88      CMD_PUSH, SN(0)WAIT_SN(1)
```

```
> Frame 1: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface \Device
> Ethernet II, Src: VMware_94:13:50 (00:0c:29:94:13:50), Dst: VMware_99:99:9f (00:0c:29:9
> Internet Protocol Version 4, Src: 192.168.12.19, Dst: 20.210.234.125
> User Datagram Protocol, Src Port: 61782, Dst Port: 12345
v KCP Protocol, conv: 4237851484, cmd: CMD_PUSH(81), frg: FALSE(0), wnd: WND_RCV_SIZE(32)
     Conv: 4237851484
     Cmd: 81
     Frg: 0
     Wnd: 32
     ts: 2031
     sn: 0
     una: 0
     len: 22
     data: Message sent by Host-A
```

```
0000  00 0c 29 99 99 9f 00 0c  29 94 13 50 08 00 45 00   ··)·····)··P··E·
0010  00 4a 8c 34 00 00 80 11  00 00 c0 a8 0c 13 14 d2   ·J·4···· ········
0020  ea 7d f1 56 30 39 00 36  cc 52 5c 7b 98 fc 51 00   ·}·V09·6 ·R\{··Q·
0030  20 00 ef 07 00 00 00 00  00 00 00 00 00 00 16 00   ········ ········
0040  00 00 4d 65 73 73 61 67  65 20 73 65 6e 74 20 62   ··Messag e sent b
0050  79 20 48 6f 73 74 2d 41                            y Host-A
```

Example of using cfadmin-cn/kcp_dissector

Appendix D – Yara Rules (Samples)

```
rule gokcpdoor {

meta:
    description = "Detects gokcpdoor malware"
    author = "LAC Co., Ltd."

     strings:
        $str1 = "gokcpdoor" ascii
        $str2 = "exec_lin.go" ascii
        $str3 = "exec_win.go" ascii
        $str4 = "syscmds/ps_linux.go" ascii
        $str5 = "syscmds/ps_windows.go" ascii
        $str6 = "target.go" ascii

     condition:
       (4 of ($str*)) and filesize > 2MB
}
```

* We recommend deliberate testing and tuning prior to implementation in any production system

Appendix E – Indicators of Compromise

Indicator	Type	Context
86f02e9f344a8e8009e59ecae934a780	MD5	ABK Downloader
d85c9b3d49b1af482c384a4253c16e28ae65a0f5	SHA1	
61eb25a6e6457087232de7ce7cd7b6cd9926e10674487c9e55b9a3fa54748b4c	SHA256	
Mozilla/4.0(compatible;MSIE8.0;WindowsNT6.0;Trident/4.0)	User-Agent	
a6f4a5ec66b7c5f275e793be02885543	MD5	gokcpdoor for Linux
bdb3db1013b16cb64b3f8156eae621054fa334bf	SHA1	
2dd8ab1493a97e0a4416e077d6ce1c35c7b2d8749592b319a7e2a8f4cd1cc008	SHA256	
fa4a45c531a19744e91bbfb9da1b29c0	MD5	gokcprat for Linux
f43e693cf6d459506249d1801742a190c3a4b483	SHA1	
993dbbce860539e1b7a1f91ebacc0ee7f1cd1a6cc37a9c7a2a2647fc64382f56	SHA256	
103.97.179[.]182	C2	

Thank you!

Any Question?