Unshelling VShell at Scale

Kazuya Nomura, Rintaro Koike

NTT Security (Japan) KK

$ whoami

Kazuya Nomura

Security Researcher @ NTT Security

CODE BLUE, JSAC, AVAR, HITCON

Music Producer & DJ

Rintaro Koike

Security Researcher @ NTT Security

VB, FIRST, Botconf, CARO, AVAR and others

Founder of nao_sec

NTT Security

Introduction

VShell

- Multi-platform RAT written in Golang
  - Windows, Linux, macOS
- Developed in the Chinese-speaking community
  - Binaries were once publicly available on GitHub
- Often used by China-nexus read teams and threat actors
  - Notably abused by UNC5174

UNC5174

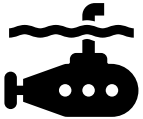
- aka CL-STA-1015
- Originally a cybercrime actor, later compromising edge devices as an IAB providing initial access to the MSS and others
- Targeted a wide range of sectors worldwide
  - Exploited React2Shell in December 2025

Worldwide

Government

Telecom

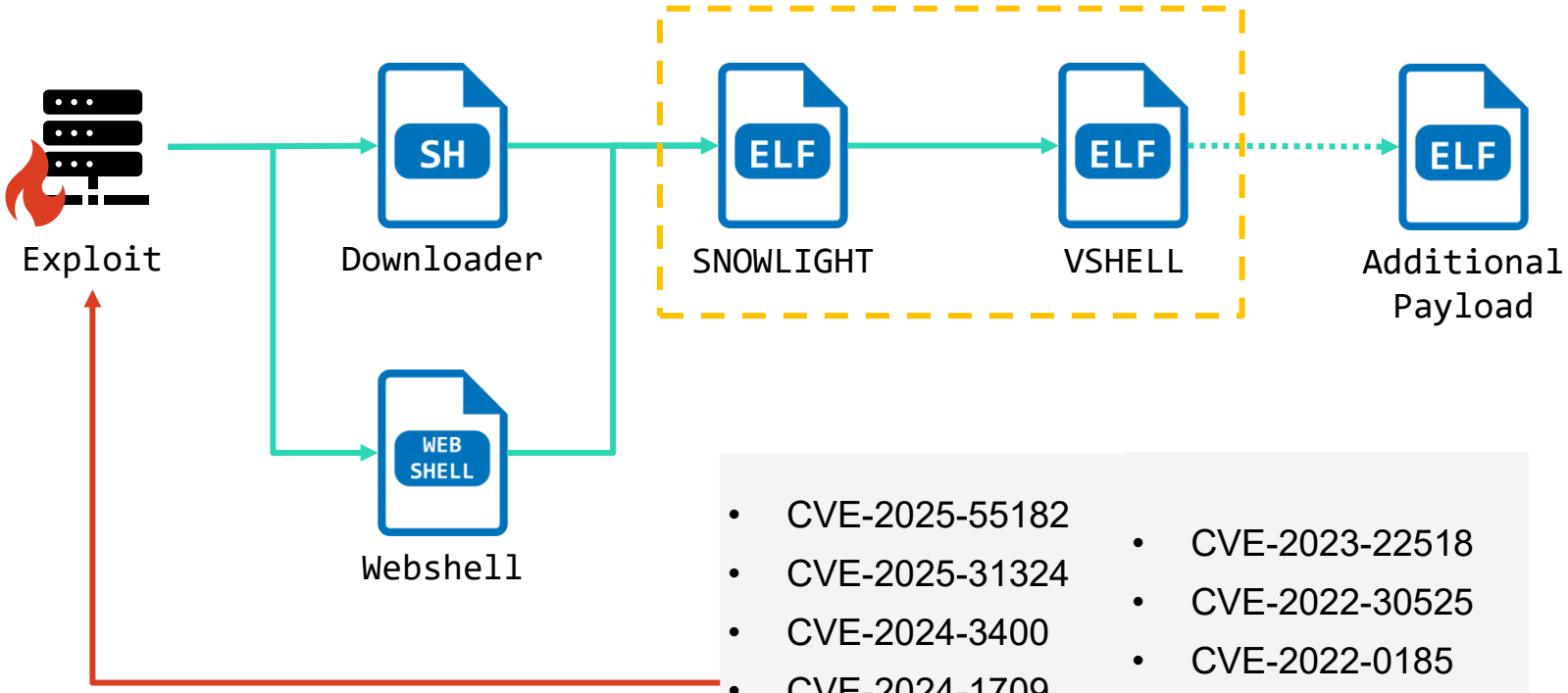
Manufacturing



Defence

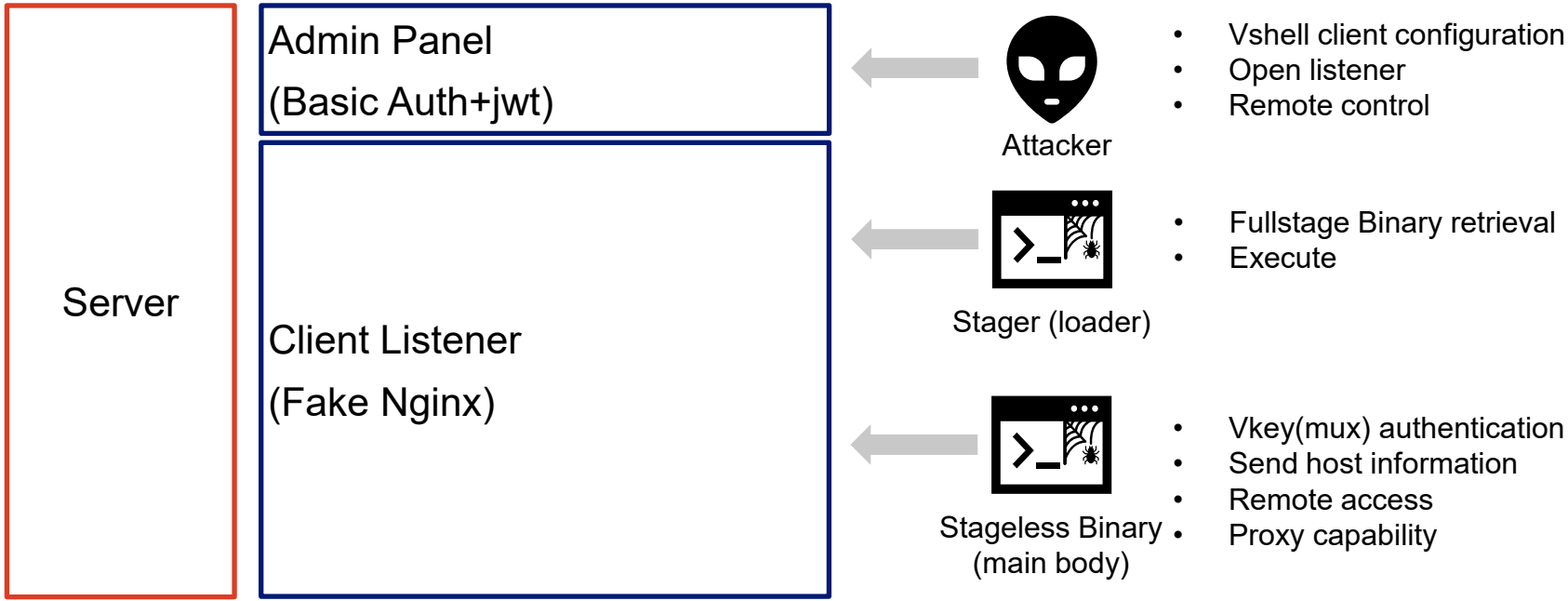
Exploit

UNC5174: Attack Flow

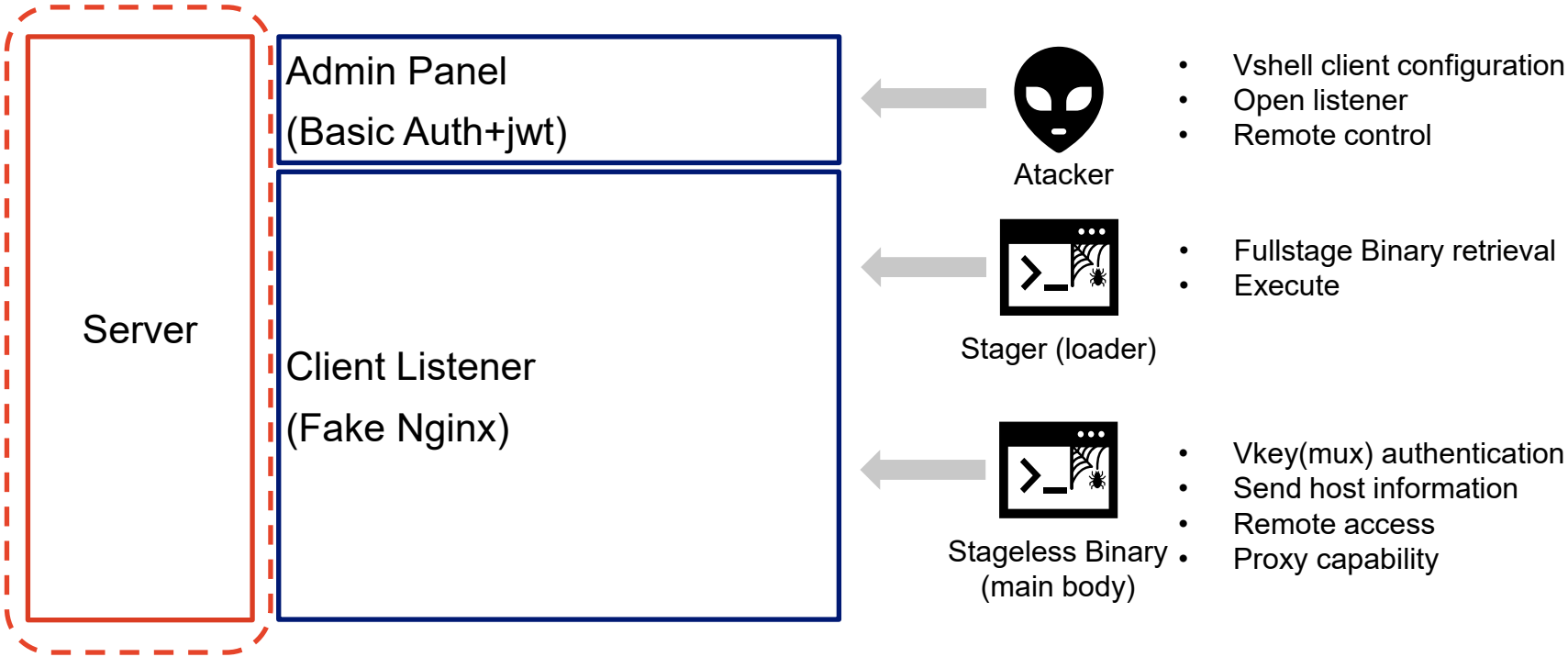


Unsheling VShell

VShell: Infrastructure



VShell: Infrastructure

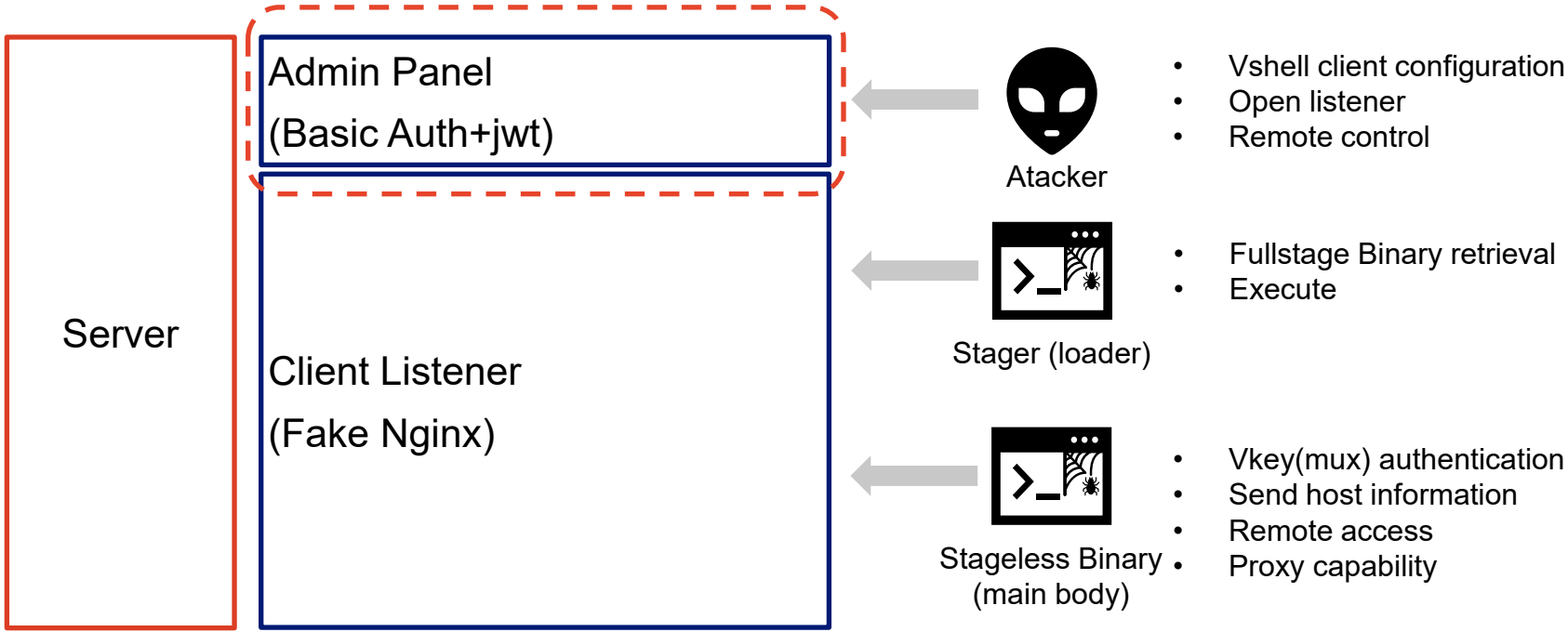


VShell: Server

- A single golang binary
- A configuration file exists
- Default values are as follows; vkey is described later

Item	Value
(C2 panel) login username	admin
(C2 panel) login password	qwe123qwe
vkey	qwe123qwe
salt	qwe123qwe
AES key / iv	MD5(qwe123qwe)

VShell: Infrastructure

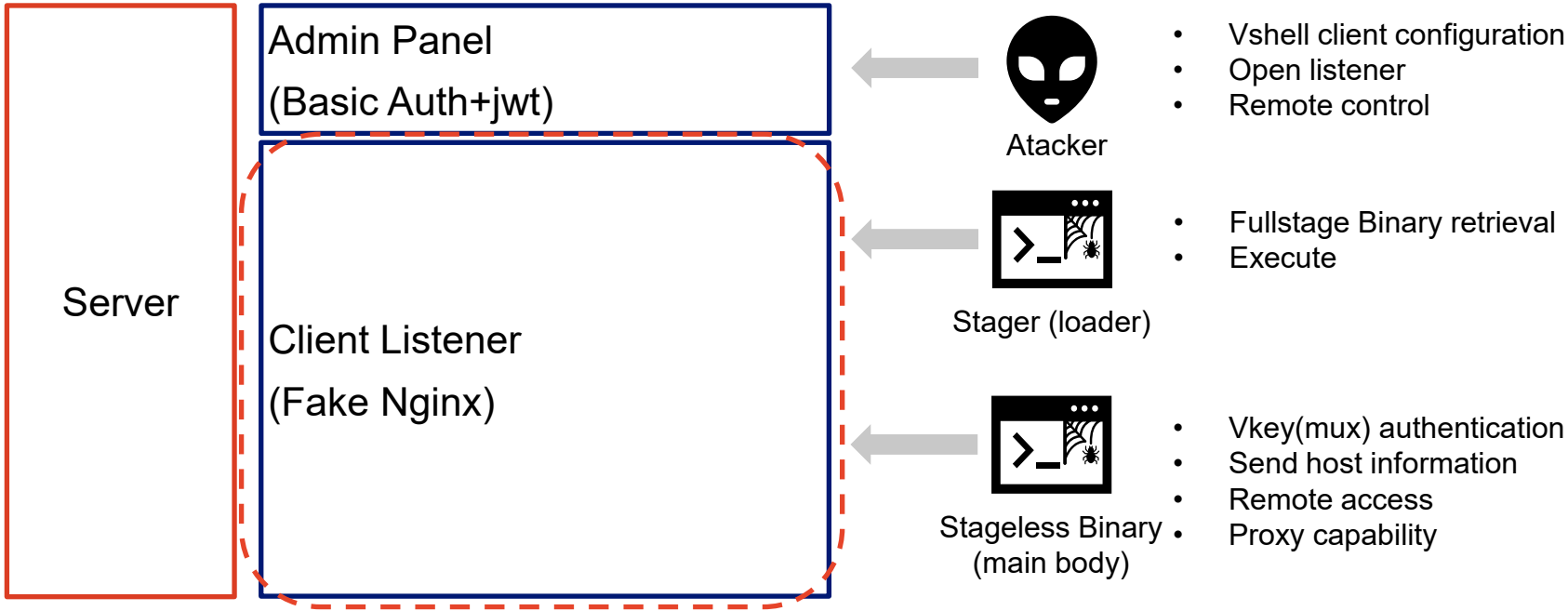


VShell: Server

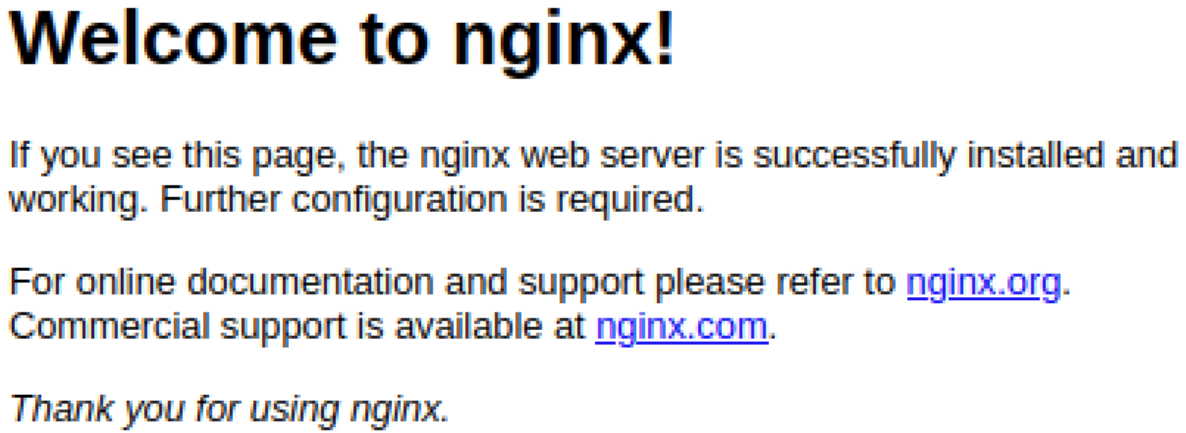
- Statically analyzed and C2 panel front-end extracted
  - Vue / vue-vben-admin based
- Information can be retrieved from the following endpoints after login

Method	Path	Response	Paginated
GET	/api/getUserInfo	user_info	No
GET	/api/getMenuList	menu_list	No
GET	/api/getPermCode	perm_code	No
GET	/api/dashboard/info	dashboard_info	No
POST	/api/client/list	client_list	Yes
POST	/api/listener/list	listener_list	Yes
POST	/api/tunnel/list	tunnel_list	Yes
GET	/api/setting/get	settings	No
GET	/api/runner/list	runner_list	No
GET	/api/file/getdisk	disk_info	No

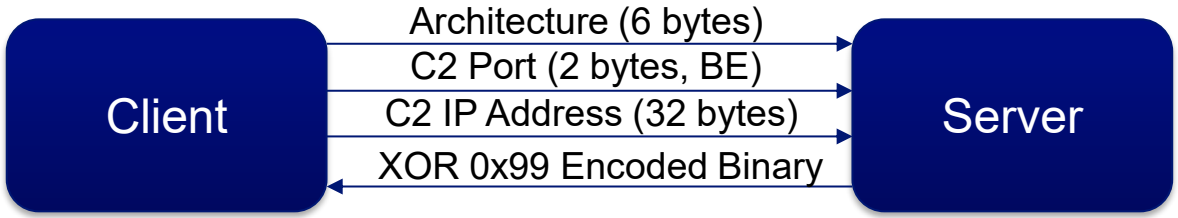
VShell: Infrastructure



Client Listener



Characteristic Response



Request
(l64=Linux 64)

```
l64   ..192.168.10.16.....................9M.............  ..............................
.....................................................!.....!.......................i.
.....i.....i.....1w.....1w.....................y.....y*.....y*.........................|..........
.....................................................!0.............................
```

Response
(0x99 XORed ELF)

This binary is notably **not packed**

Hunting Approach

Under the following conditions, Censys and similar, C2s can be hunted

BASIC Authentication response hash
"sha1:baf371b6fd867efdcec2a3c65248eb21c7e8f902“

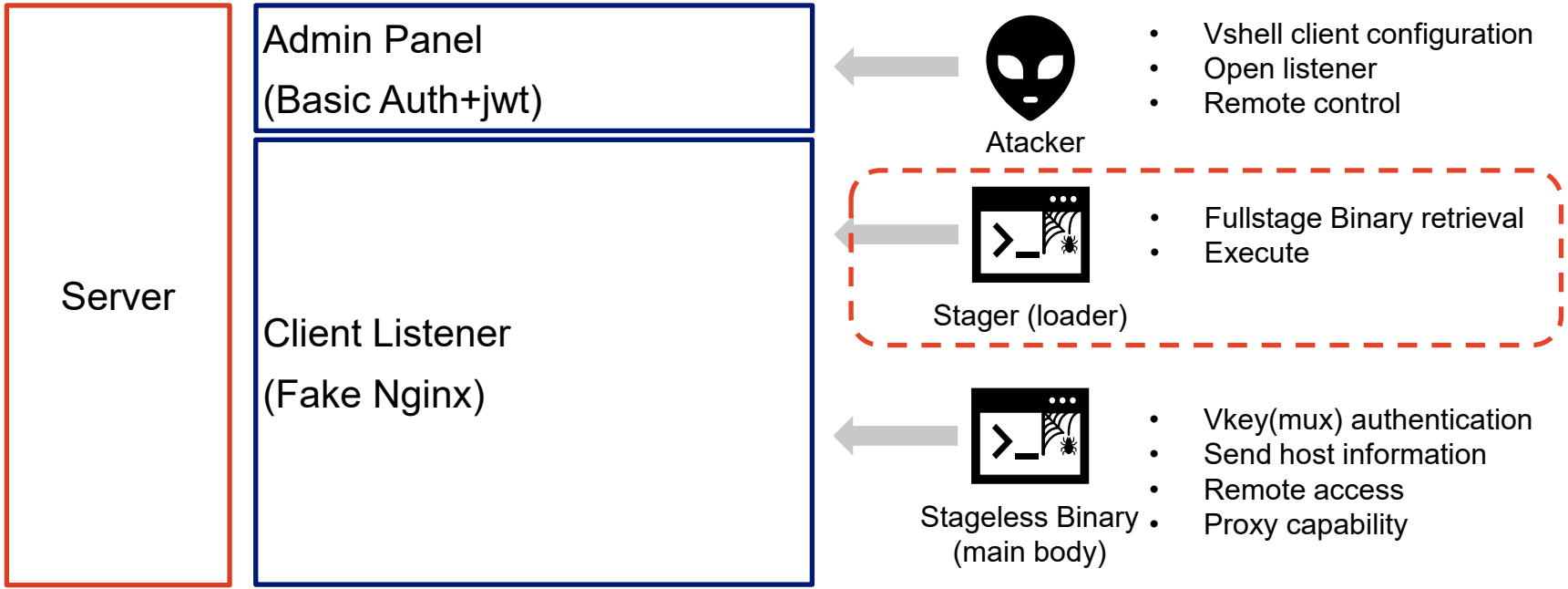
+

nginx response hash "sha1:9025c022f8b57671a0c1dcb0b3a9b1a97cec7be2“

+

When a specific TCP packet is sent, XORed stageless binary is returned

VShell: Infrastructure

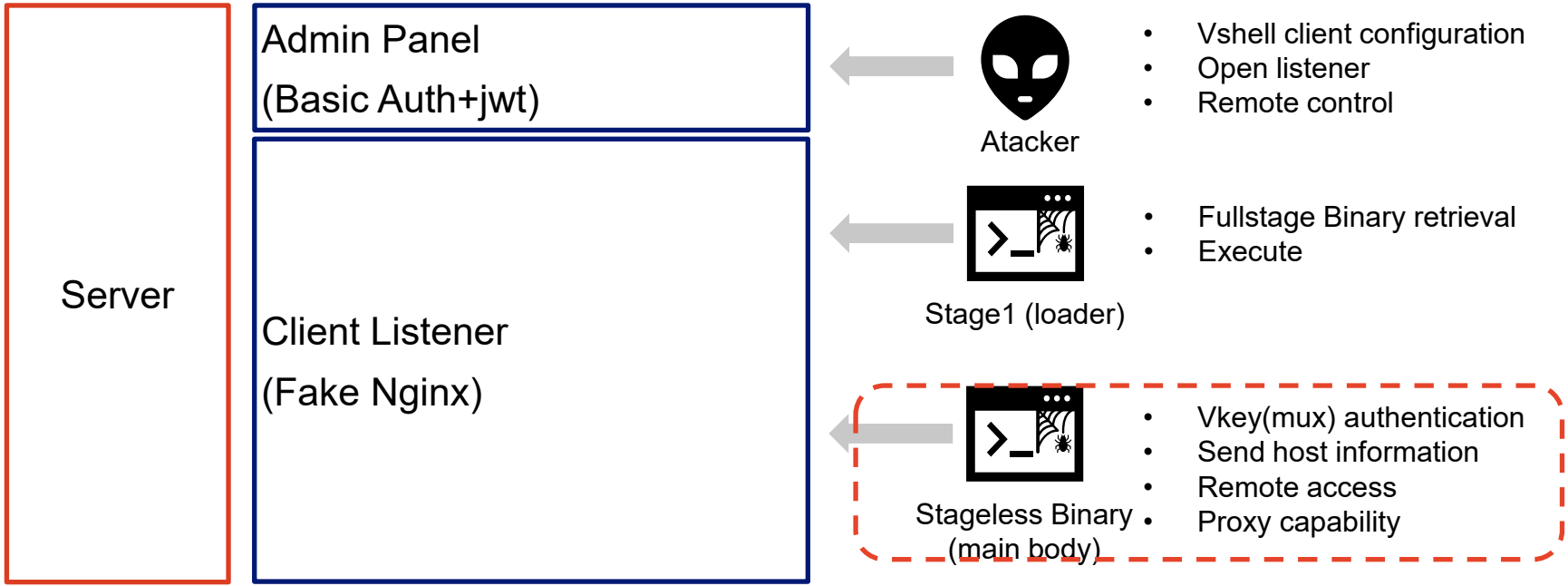


VShell: Stager Client

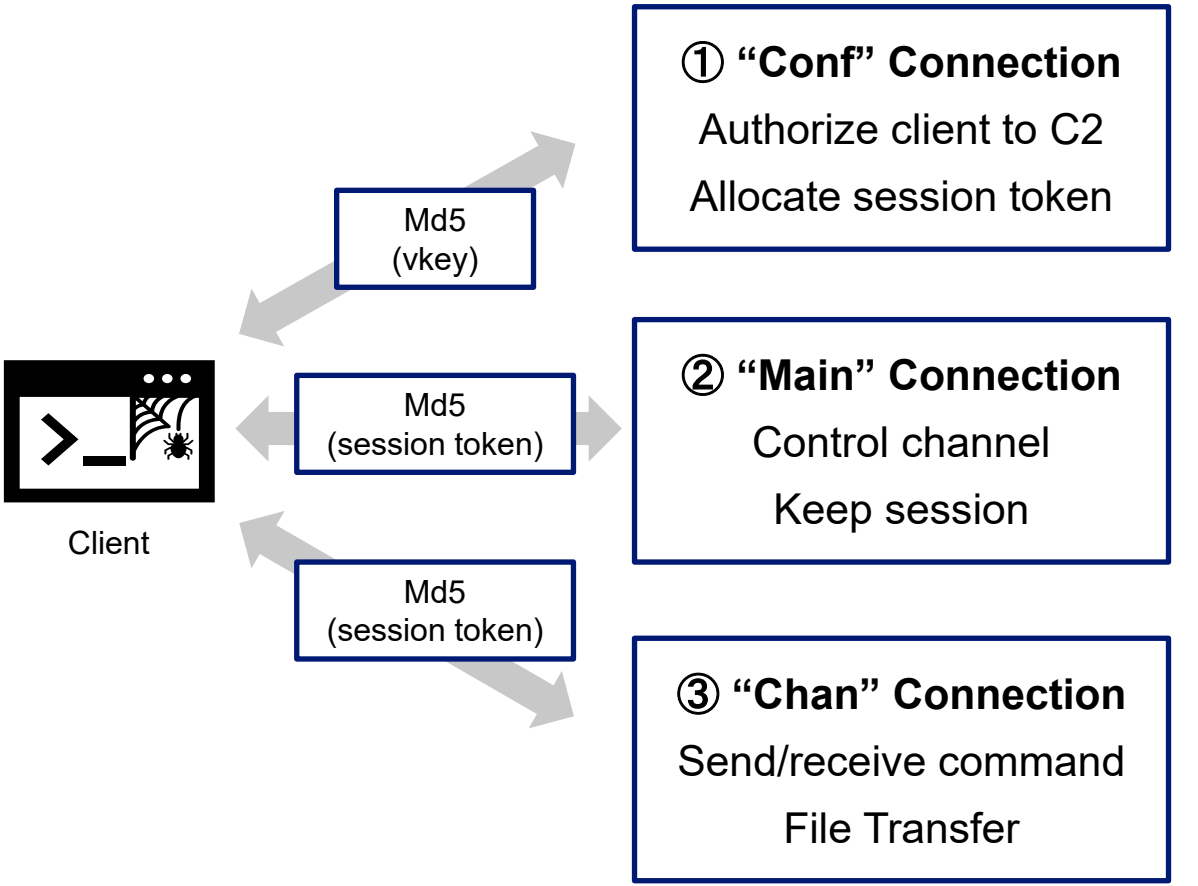
- A Loader download & execute VShell
- Some intelligences tracked it as SNOWLIGHT
  - Started disguised as [kworker/0:2]
  - Check for the existence of /tmp/log_de.log
  - TCP connect to C2
  - Send Architecture (e.g., i64)+port+IP
  - Receive and decode XOR 0x99 ELF (VShell)
  - Fileless execution using memfd_create + fexecve

It may come from
“/slt” (stageless linux tcp?)
“/swt” (stageless windows tcp?)
endpoint.

VShell: Infrastructure



VShell: Connection Protocol



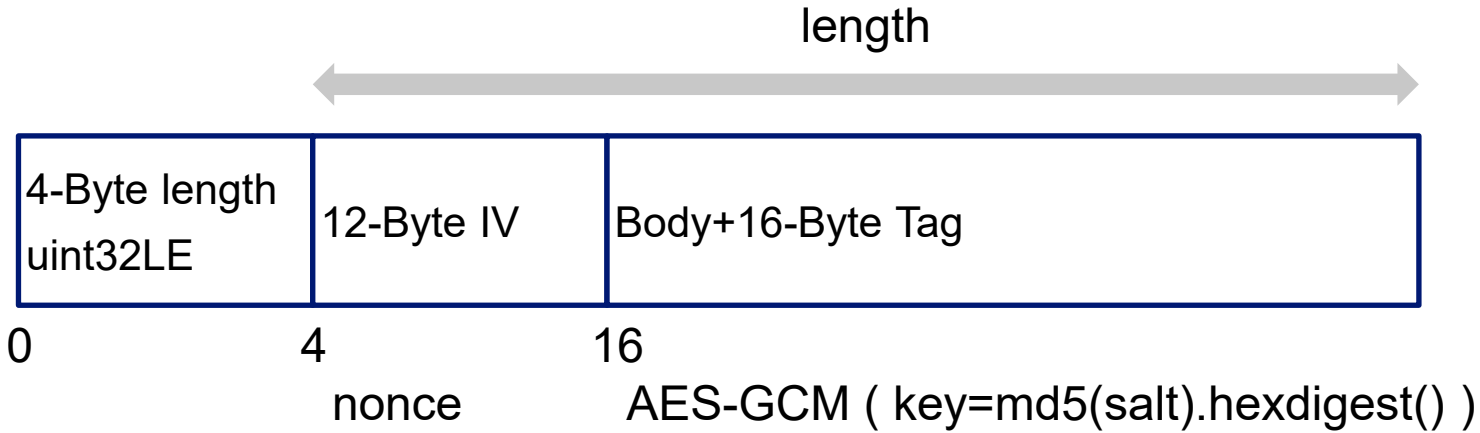
①Client is shown in C2 Panel

②Client is marked as online
(While ping loop is valid)

③C2 can execute command on client

VShell: Packet Structure

All packets has same encrypted structure

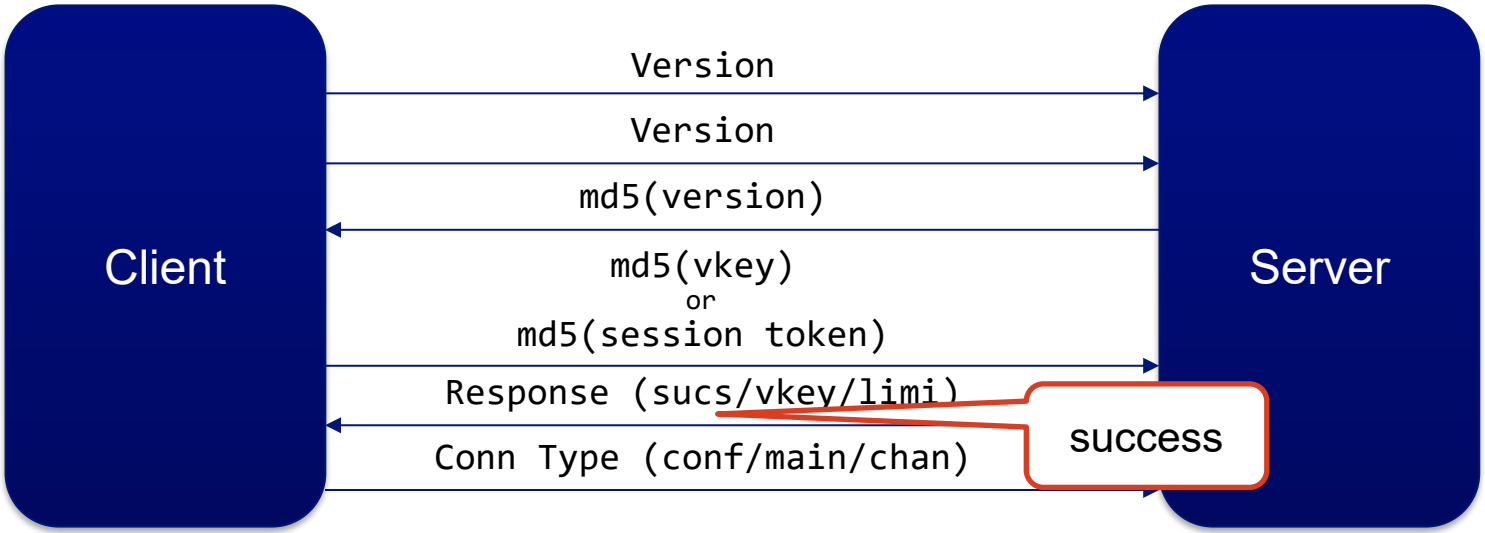


Length mismatch causes **server crash** (Golang specification?)

VShell: Authorization

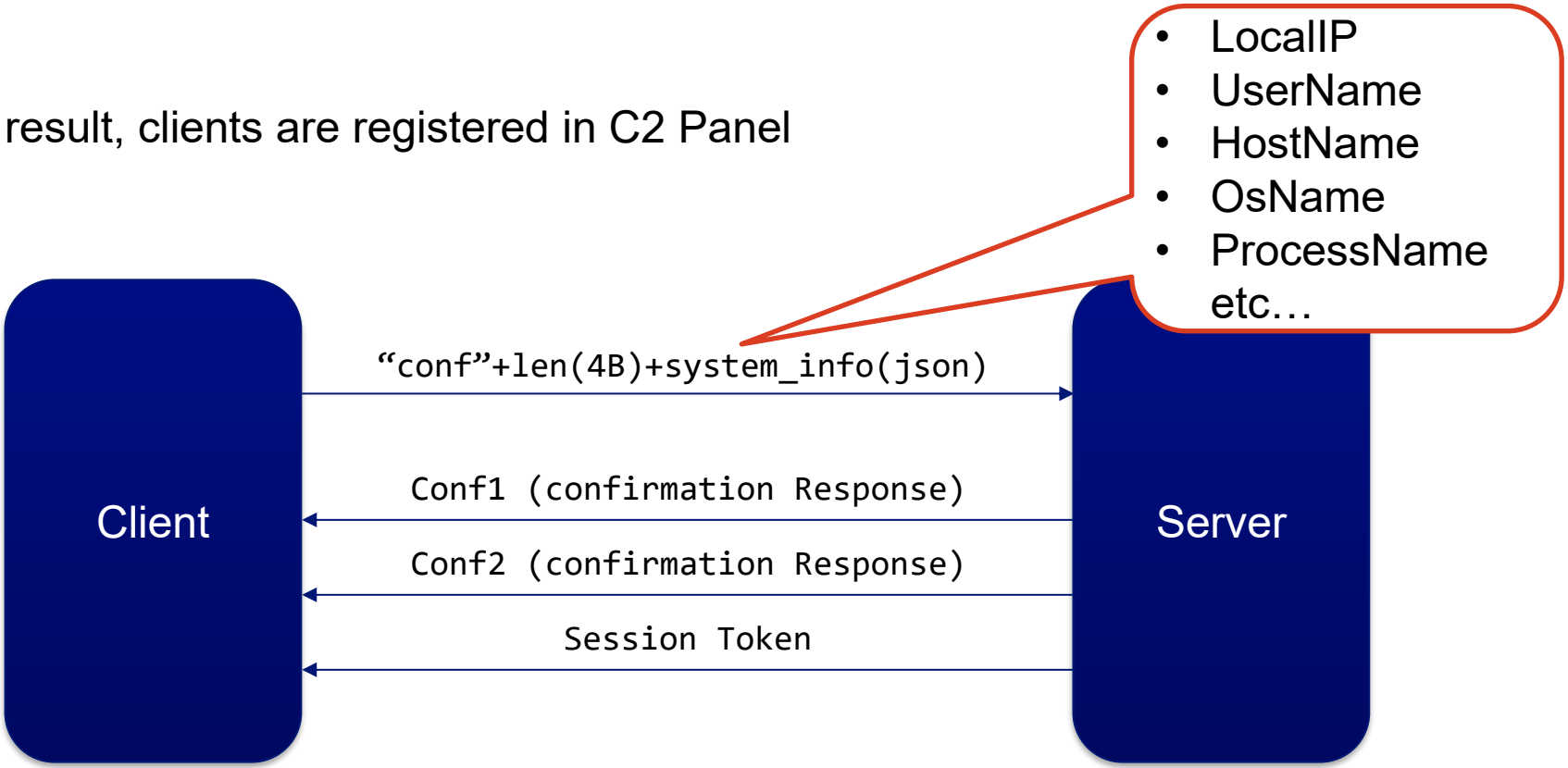
All channel have same authentication method

AES256 (key=md5(salt).hexdigest()) encryption



1. “conf” channel

As a result, clients are registered in C2 Panel



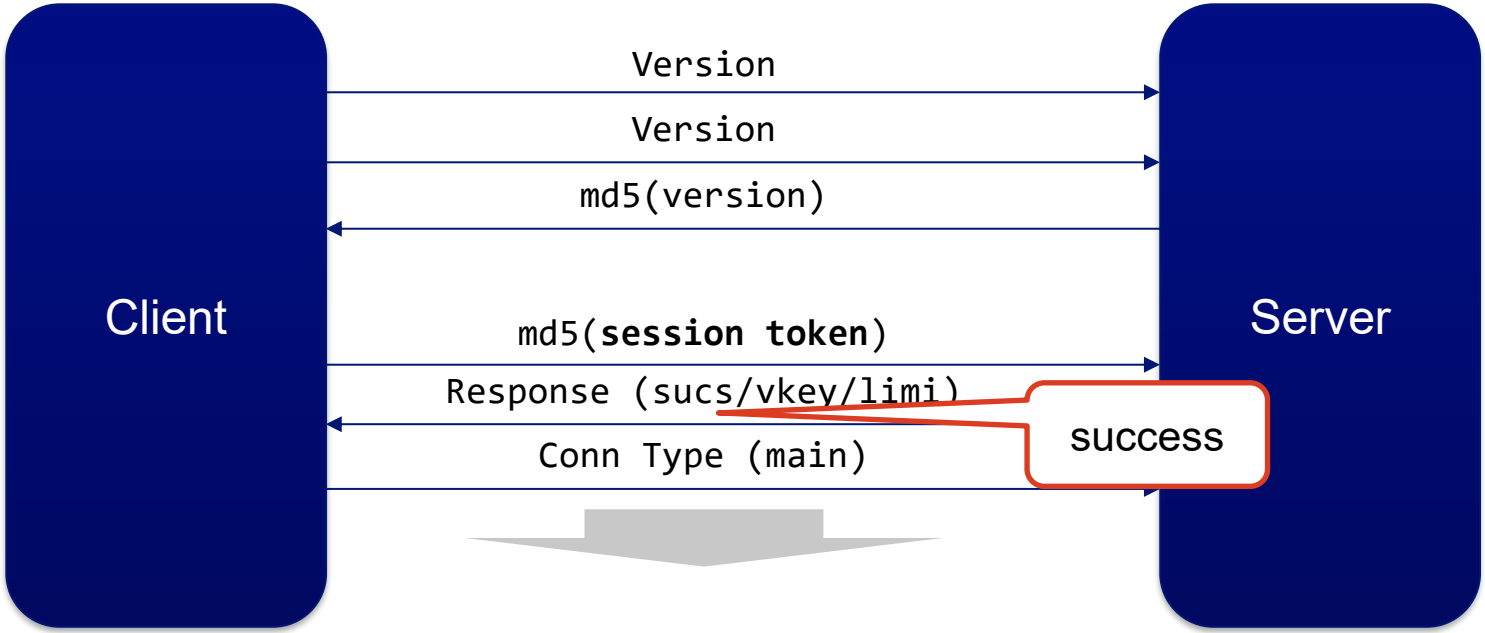
VShell: Registration

- JSON: send client information
  - Conn Type=Conf
- Once accepted by the server,
  the client is registered on the server
- When accepted, a session token
  (used for C2 communication)
  is issued by C2


```
{
  "Id": 0,
  "IsConnect": false,
  "VerifyKey": "",
  "Tp": "tcp",
  "Addr": "",
  "Remark": "",
  "Status": false,
  "LocalIP": "192.168.10.16",
  "UserName": "www",
  "HostName": "www-server",
  "Location": "",
  "OsName": "linux_amd64",
  "ProcessName": "[kworker/0:2]",
  "PingCheckTime": 0,
  "NoStore": false,
  "NoDisplay": false,
  "MaxConn": 0,
  "NowConn": 0
}
```

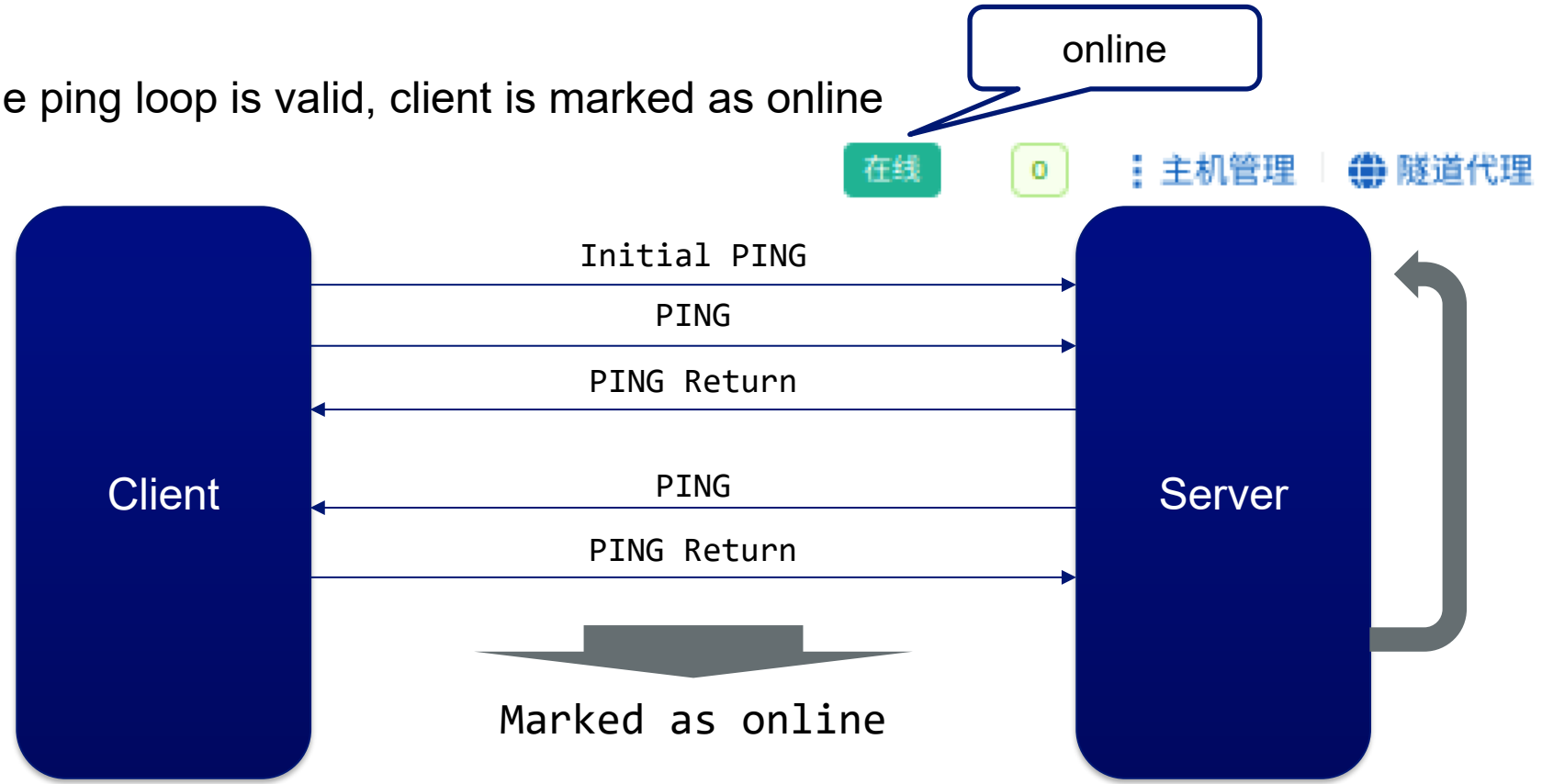
2. “main” channel

Connection is established and kept (and seems almost not in used)



3. “chan” channel (ping loop)

While ping loop is valid, client is marked as online



MUX Protocol

IN “chan” channel, MUX Protocol is used to communication

Flag	Name	Meaning
0x00	PING	Liveness check request (with timestamp)
0x01	NEW_CONN	Establish a new logical connection
0x03	DATA	Actual data (command JSON / response)
0x05	WINDOW	Flow control (offset + size)
0x06	OPEN	Notification that the connection is opened
0x07	CLOSE	Close the connection
0x08	PING_RETURN	Response to PING

Header structure will be different for each flag

VShell: Protocol — Three Header Types

- VShell's mux protocol uses three header formats, distinguished by total packet length (5 / 7 / 13 bytes).
- The receiver branches on the packet length to decide how to parse each header.

flag	Name	Total length	Structure	Body packet
`0x00`	**PING**	7 byte + body	`flag + conn_id + length`	Yes (timestamp)
`0x01`	**NEW_CONN**	5 byte	`flag + conn_id`	No
`0x03`	**DATA**	7 byte + body	`flag + conn_id + length`	Yes (JSON / data)
`0x05`	**WINDOW**	13 byte	`flag + conn_id + offset + size`	No
`0x06`	**OPEN**	5 byte	`flag + conn_id`	No
`0x07`	**CLOSE**	5 byte	`flag + conn_id`	No
`0x08`	**PING_RETURN**	7 byte + body	`flag + conn_id + length`	Yes (timestamp)

5-byte Header — Control Signals

- NEW_CONN (0x01) / OPEN (0x06) / CLOSE (0x07)
  — control signals only; they carry **no data**.

flag	conn_id
1 byte	4 bytes — uint32, little-endian

- Flag values: **NEW_CONN = 0x01** **OPEN = 0x06** **CLOSE = 0x07**

Example (OPEN, conn=1):

06 01 00 00 00

7-byte Header + Separate Body

- PING (0x00) / PING_RETURN (0x08) / DATA (0x03)
  — The header and body are **two independent encrypted packets**.

flag	conn_id	length
1 byte	4 byte	2 byte – uint16

length = byte length of the following body packet (sent as a separate packet)

PING / PING_RETURN (0x00 / 0x08)	DATA (0x03)
conn_id = 0xFFFFFFFF (main, fixed) body = nanosecond timestamp **Header:** `00 ff ff ff ff 1e 00` **Body:** `"2026-06-05T06:10:07.343766748Z"` (30B) *PING_RETURN is identical but starts with 0x08*	conn_id = logical connection ID body = command JSON / response data **Header** (conn=3, body=178B): `03 03 00 00 00 b2 00` **Body:** `b1 00 00 00 7b 22 …` *({"ConnType":"v", ...})*

13-byte Header — WINDOW (0x05)

- WINDOW (0x05) carries flow-control values and has **no body packet**.

flag	conn_id	offset	size
0x05	4B	4B — uint32	4B — uint32

offset = cumulative received ACK · size = receive-buffer free space

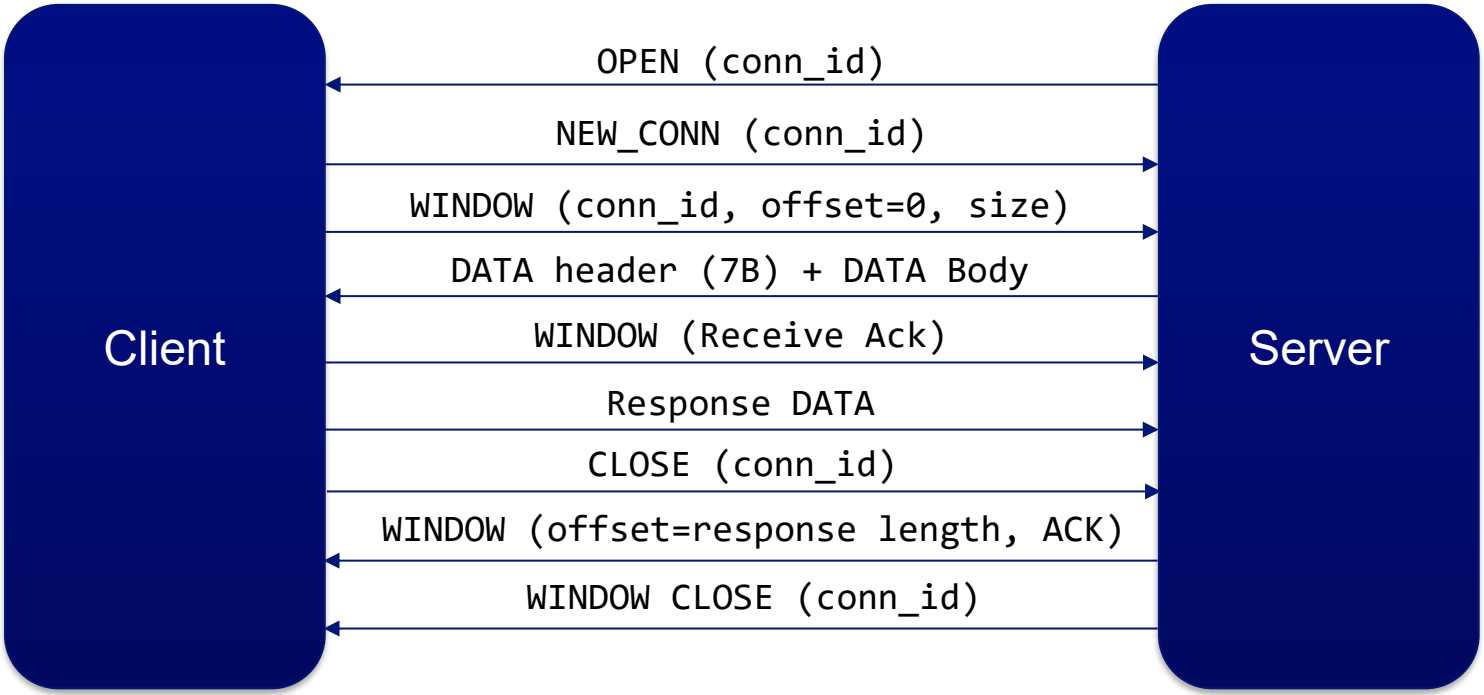
Example (conn=3, offset=750, size=735300): `05 03 00 00 00  ee 02 00 00  c4 36 0b 00`

Receiver decision logic (branch on total length):

```
if   len(pkt)==5:  flag = pkt[0]                          # NEW_CONN / OPEN / CLOSE
elif len(pkt)==7:  flag, conn_id, length = header(pkt) # PING / DATA / PING_RETURN
elif len(pkt)==13: flag, conn_id, offset, size = ...   # WINDOW
```

3. “chan” channel

1 command = 1 connection, MUX Protocol



Additional Features

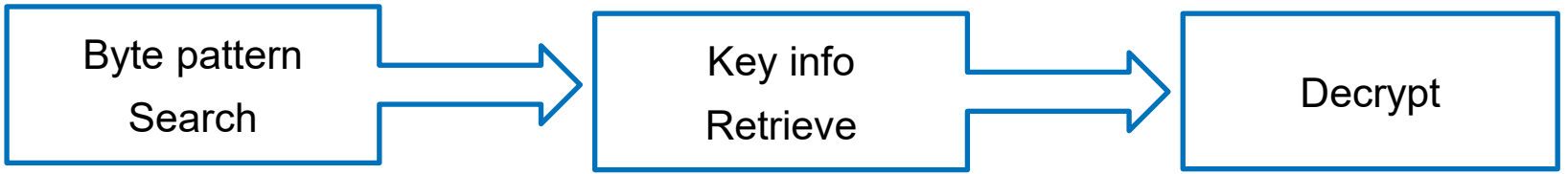
- Terminates when the following files exist
  - $TMPDIR/log_de.log` or `/tmp/log_de.log
  - $TMPDIR/log_de-0.log` or `/tmp/log_de-0.log
  - /home/vbccsb
- Vkey authentication failure causes termination
- Terminates if something resembling container detection is present

NTT Security

Observation & Result

Config Extraction

- Config is AES128-CBC encryption
- Key : MD5(<password>).digest()

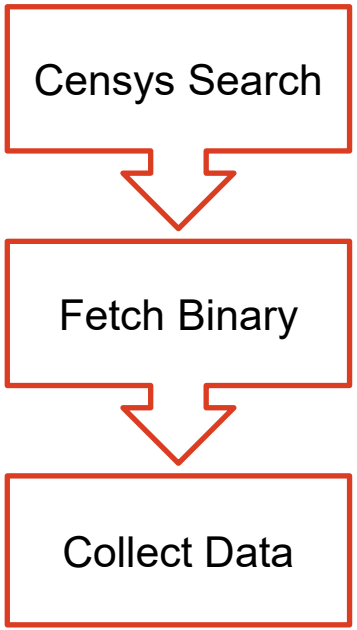



```
mov ecx, 0xa00
#within next 10 bytes
rep movsq
#within previous 50 bytes
lea rsi, [rip+offset]
```

First 16byte are the MD5-hashed password
Remainder is AES-encrypted config

```
"config": {
  "server": "0.0.0.0:443",
  "type": "tcp",
  "vkey": "qwe888qwe",
  "proxy": "",
  "salt": "qwe888qwe",
  "l": false,
  "e": false,
  "d": 30,
  "h": 10
},
```

Hunting Approach



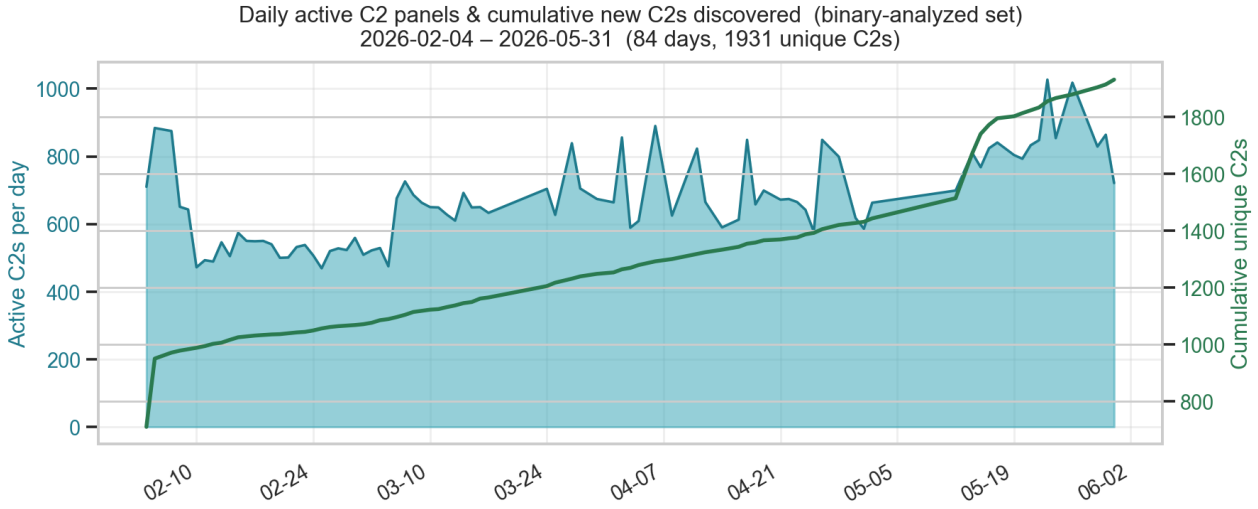
- Censys search:
  sha1:baf371b6fd867efdcec2a3c65248eb21c7e8f902
  sha1:9025c022f8b57671a0c1dcb0b3a9b1a97cec7be2

- TCP request sent to fetch binary
- Config retrieval and analysis

- If accessable, retrieve data from each endpoint

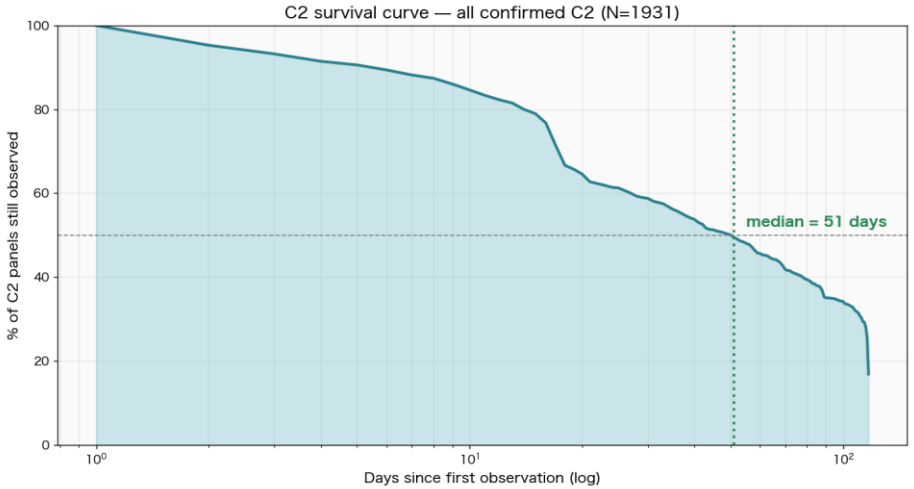
Hunting Approach

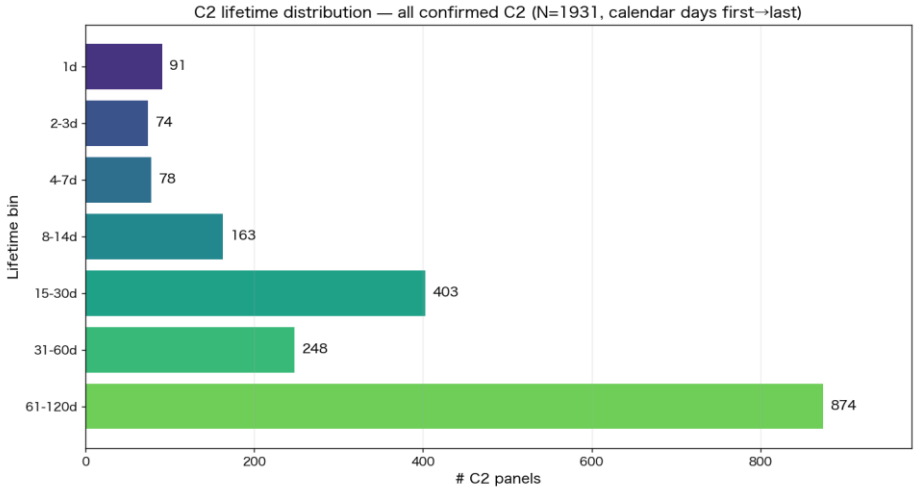
- About 500 C2 observed daily
  - Feb 4 to May 31
- The attackers' OPSEC Fail allows data from about 30 C2s to be retrieved daily



C2 TTL

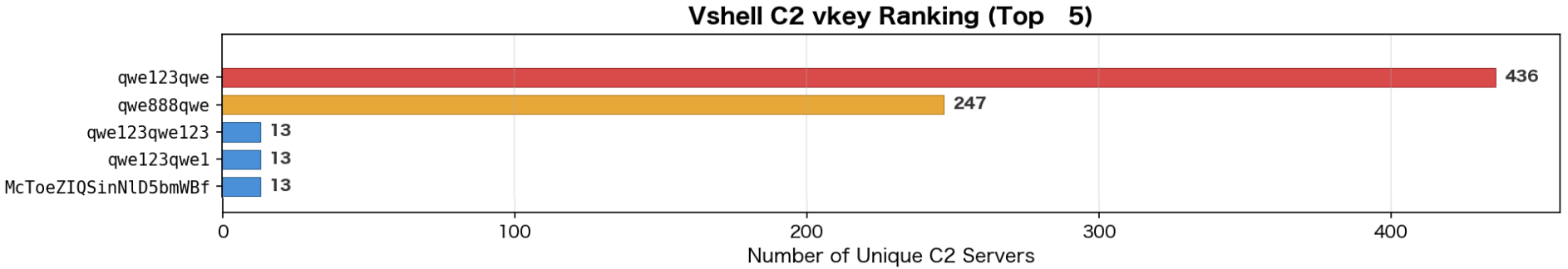
- C2 tends to stay online for relatively long periods
  - Median about 50 days





Continuous IOC collection & monitoring are important

Vkey



The default vkey is the most common

Vkey

- Combining by port and similar fields forms cluster-like groups
- vkey=qwe888qwe cluster
  - All ports are 60578
  - C2 subnet
    - 208.87.203.0~ 208.87.205.0 has over 100 hosts
  - AS
    - SonderCloud
    - Cogent Communications LLC
  - Several hundred C2 belong to this cluster
- Suggests a large-scale IAB that automates its operations

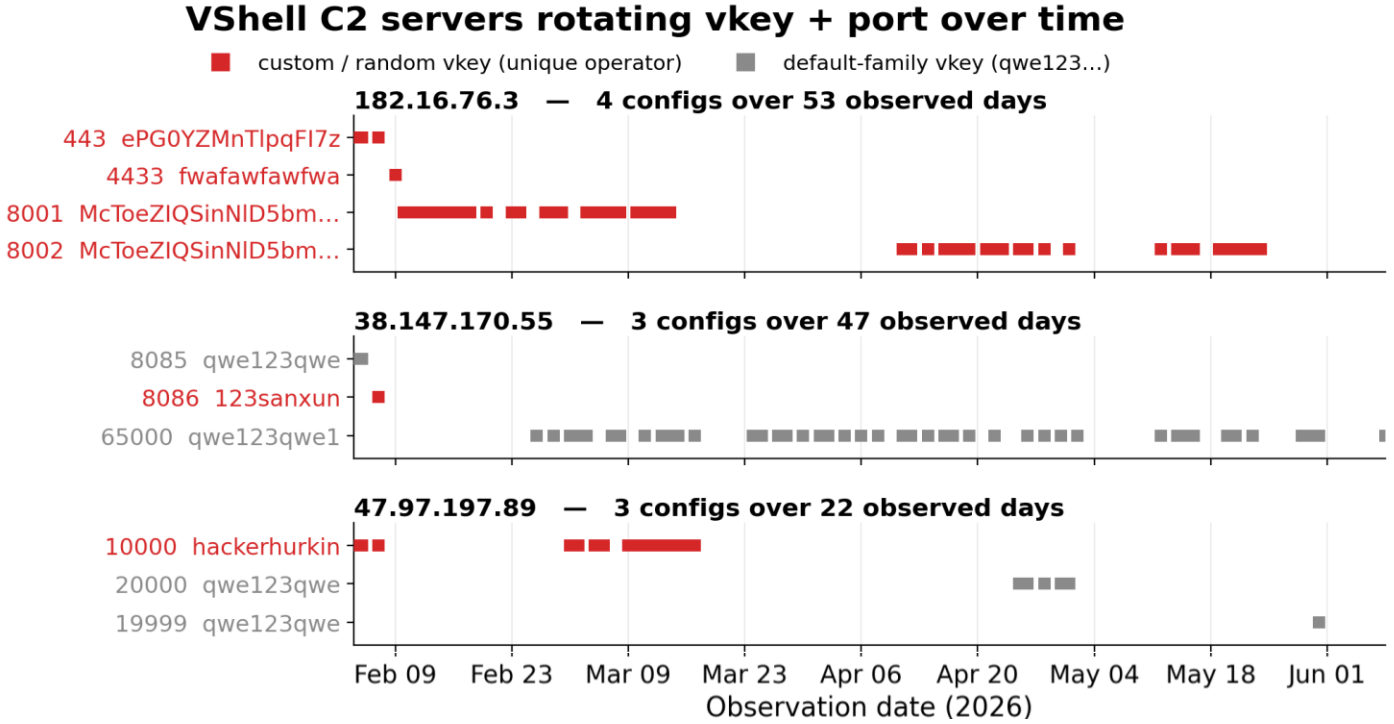
Vkey

- Vkey characteristics (string length, hash-like strings, keywords, etc.) may allow attacker identification
- Other interesting Vkey

vkey	Count	Description
taiwanshizhongguode	1	Taiwan belongs to China (in Chinese)
sb360sbqax	1	Insult toward 360 antivirus and similar (in Chinese)
Admin@114514	1	Japanese internet meme 114514
woyaozhuanmi	1	Want to earn money(in Chinese)
qwe123qwe@brgov	1	Targeting Brazilian government?
Maga@Trump	1	Trump?

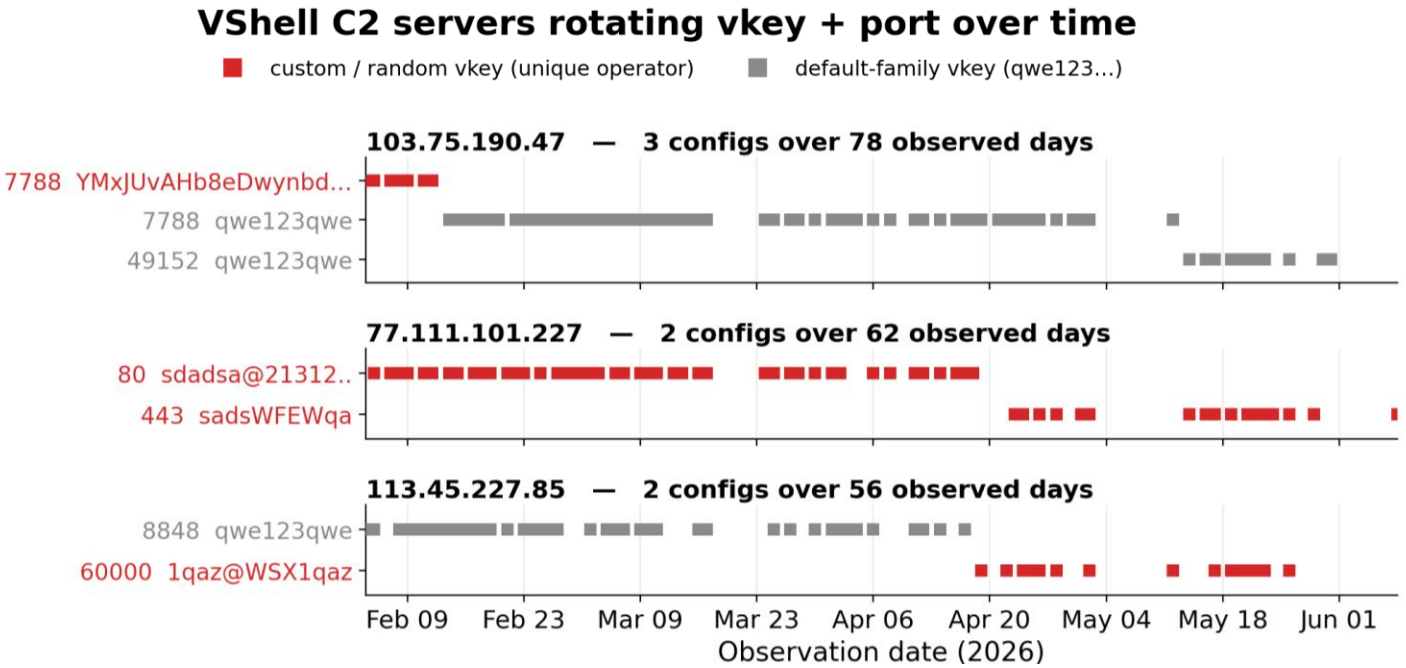
Vkey

- Vkey and ports are rotated on multiple C2s
  - Rotating these settings may extend server lifetime



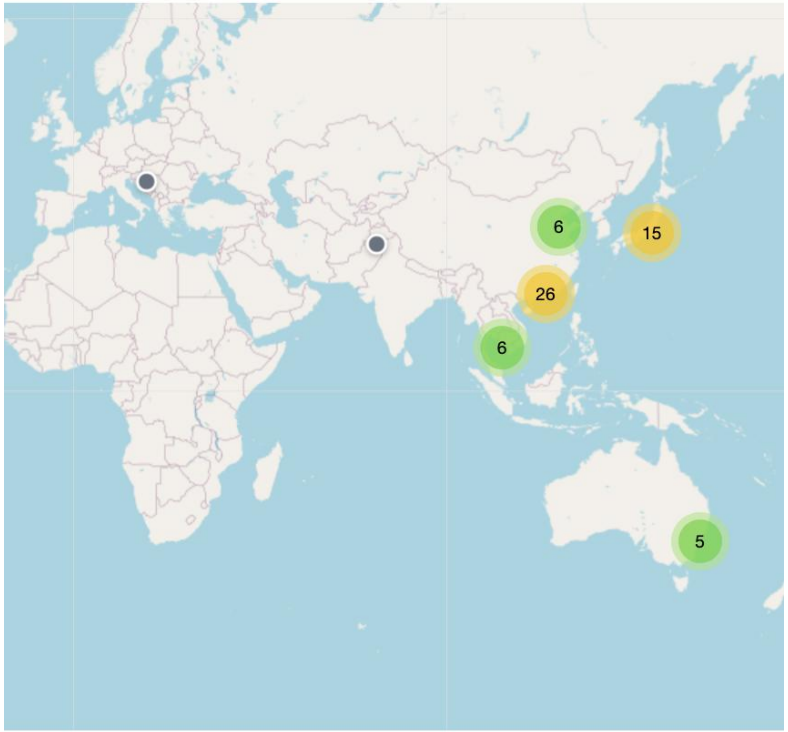
Vkey

- Vkey and ports are rotated on multiple C2s
  - Rotating these settings may extend server lifetime



C2 Crawling: Sample-1

- Listener
  - Resolved from this domain, IP contains attacker group info
  - UNC2727?, APT43 info also present...
- Victims
  - Hong Kong, Australia, Japan, China
- Malware executed from a Youdao Translate decoy?



C2 Crawling: Sample-1

Runner
(files the attacker drops in)

Bold = attacker-specific

Thin = bundled by default

File	Description
1.bat	**Launcher. Starts Youdao Translate (YoudaoDict.exe) while running D:/PATH/1.exe**
vb2.vbs	**Runs 1.bat in a hidden window**
3.vbs	**Creates a desktop shortcut. Icon=favicon.ico, target=vb2.vbs**
favicon.ico	**Legitimate icon of the Youdao Translate app**
favicon__3_.ico	**MS Edge icon**
PATH.zip	**Package of the 5 files above**
1.exe	**x64 shellcode -> PE embedded at offset 0xe55. Process injection?**
output_64.exe	**Same structure as 1.exe. Has AMSI bypass capability**
mimikatz.x64.exe	mimikatz
AddUser.dll	Creates an administrator-privileged user via this DLL
Project1.exe	**Winos 4.0**
fscan.x64.elf	Internal network scanner
gost.x64.exe	GOST Windows version SOCKS5/HTTP/TCP/UDPproxy
gost.x64.so	GOST Linux version

C2 Crawling: Sample-1

— list of process names
the malware masquerades as

Process Name	Count
OSikIO.exe	48
r9Lad6G28TEr2.exe	34
r9Lad.exe	19
Win32_Process.exe	17
Qbrowser.exe	16
ExecQuery.exe	14
r9.exe	13
OS.exe	13
1.exe	8
Configs.db.exe	7
r9Lad6G28TEr2 (1).exe	5
init.exe	2
tcp_windows_amd64.exe	1
diag.exe	1
CktvcX.exe	1

VShell C2 panel (downloaded from here)
Default file names
-> possibly attacker testing

C2 Crawling

NTT Security

Test sample

- Confirmed infections on multiple servers by clients that appear to be test samples
  - C2 panel allows downloading client; naming {protocol}_{architecture}
- The following possibilities are considered
  - Infection testing by the attacker themselves
  - VShell verification, Red Team
- May allow identification of unknown attacker infrastructure

Count	ProcessName	Note
36	tcp_linux_amd64	Linux default
13	tcp_windows_amd64.exe	Windows default
3	tcp_windows_amd64 (1).exe	Browser re-download copy
3	tcp_windows_i386.exe	32-bit Windows
2	tcp_windows_amd64(1).exe	Same as above (no space)
2	tcp_darwin_arm64	**macOS ARM**
2	tcp_windows_amd64 (9).exe	Re-download copy
1	tcp_linux_arm64	**Linux ARM**
1	ws_windows_amd64.exe	WebSocket build
1	tcp_windows_amd64 (6).exe	Re-download copy
1	tcp_windows_amd64 (4).exe	Re-download copy

C2 Crawling: Sample-2

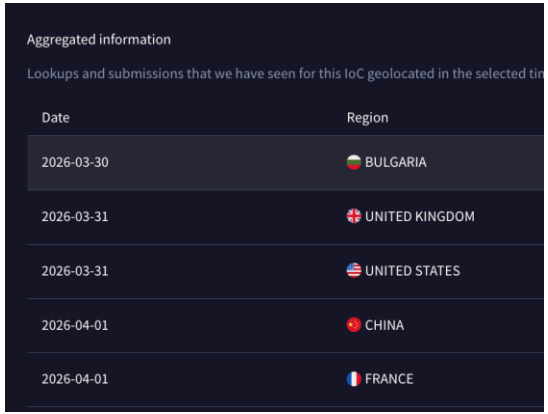
- 2024 SilverFox intelligence reported by Tencent
- On online sample-sharing sites — Multiple submission history
- The malware used has shifted over time

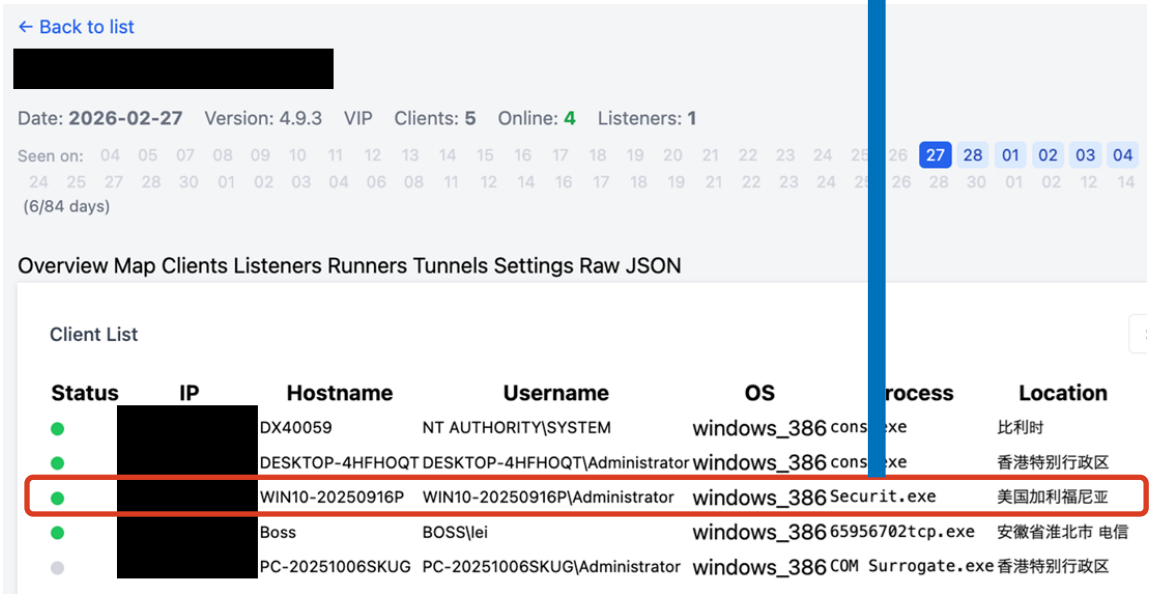
Date	Family	Sample name
2024-04-02	**Gh0st RAT**	Captured by shadowserver
2024-04-10	**Gh0st RAT**	OrionNavigator.exe
2024-04-11		npp.exe (Notepad++ impersonation)
2024-10-14		wannacry_webshell
2025-07-10	**ValleyRAT**	ChromeSetup.exe
2025-07-14	**ValleyRAT**	valleyrat.dll
2025-07-14	**ValleyRAT**	MuMu.exe
2025-08-04		xrkbdsuaX64.msi
2025-08-06		AweSun_win_online_setup_x64.msi
2026-03-31	**VShell**	**securit.exe**
2026-04-22	**FatalRAT**	k4axo5.exe

C2 Crawling: Sample-2

- Same sample appeared on the panel
- Multiple submission history

Date	Detection name	Family	Sample name
2026-03-31	trojan.ggcode	**VShell**	**securit.exe**





C2 Crawling: Sample-2

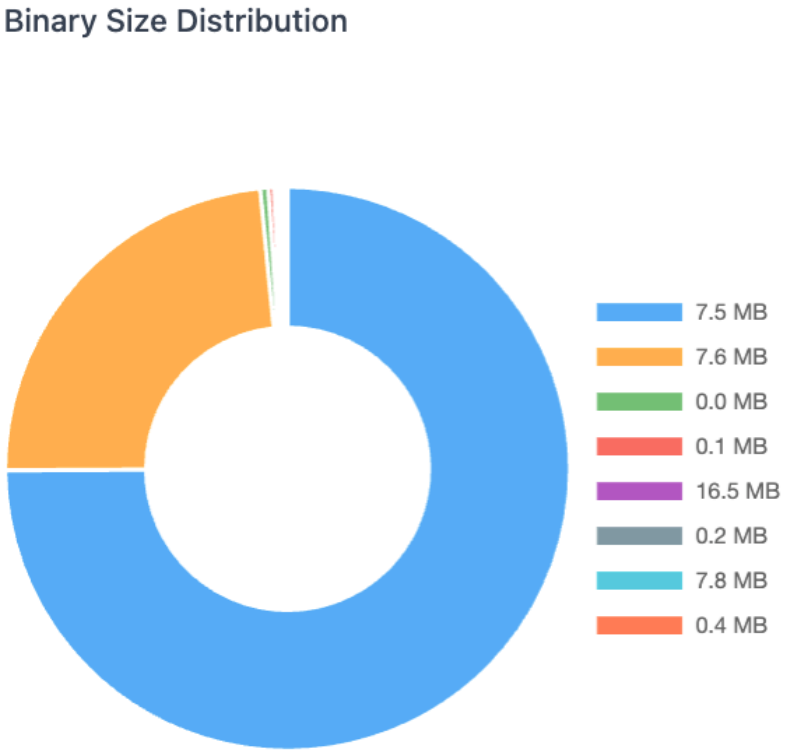
- Test sample

DESKTOP-M8NKG72 DESKTOP-M8NKG72\admin windows_386 tcp_windows_i386.exe 越南河内 Henan, Vietnam

DESKTOP-M8NKG72 DESKTOP-M8NKG72\admin windows_386 tcp_windows_i386.exe 越南 Vietnam

VShell Variants?

- From some C2 servers,
  binaries of differing sizes are dropped
- Download failure.
  Caused by an unconfigured server?
  Config-less incomplete
  binaries, etc.



VShell Variants?

- 10MB sample exists
- A variant, or a fork rebuilt from older source code

	Normal sample	10MB sample
Size	approx. 7.5MB	10MB
Go Version	1.20	**1.24**
Garble(obfuscation)	0.12 and earlier	**0.13 and later**
Version string	4.9.3	**0.1.1**

VShell Variants?

- Config key redundancy suggests an older version was rebuilt
- Variants are very likely to appear in the future

Normal sample	10MB sample
server	Server
type	Type
vkey	Vkey
proxy	Proxy
salt	**EncryptSalt**
L	**Listen**
E	**Ebpf**
D	**DisconnectTime**
H	**HearbeatTime**

Summary

- VShell C2 hunted; hundreds of samples retrieved daily & analyzed
- Revealed detailed connection protocols between C2 and client
- Config-based attacker attribution was shown to be possible
- Revealed the hosts targeted and tools abused by attackers
- After C2 is configured, attackers are likely to run infection tests in their own environment
- VShell source code leaked, showing variants and new versions may appear


NTT Security