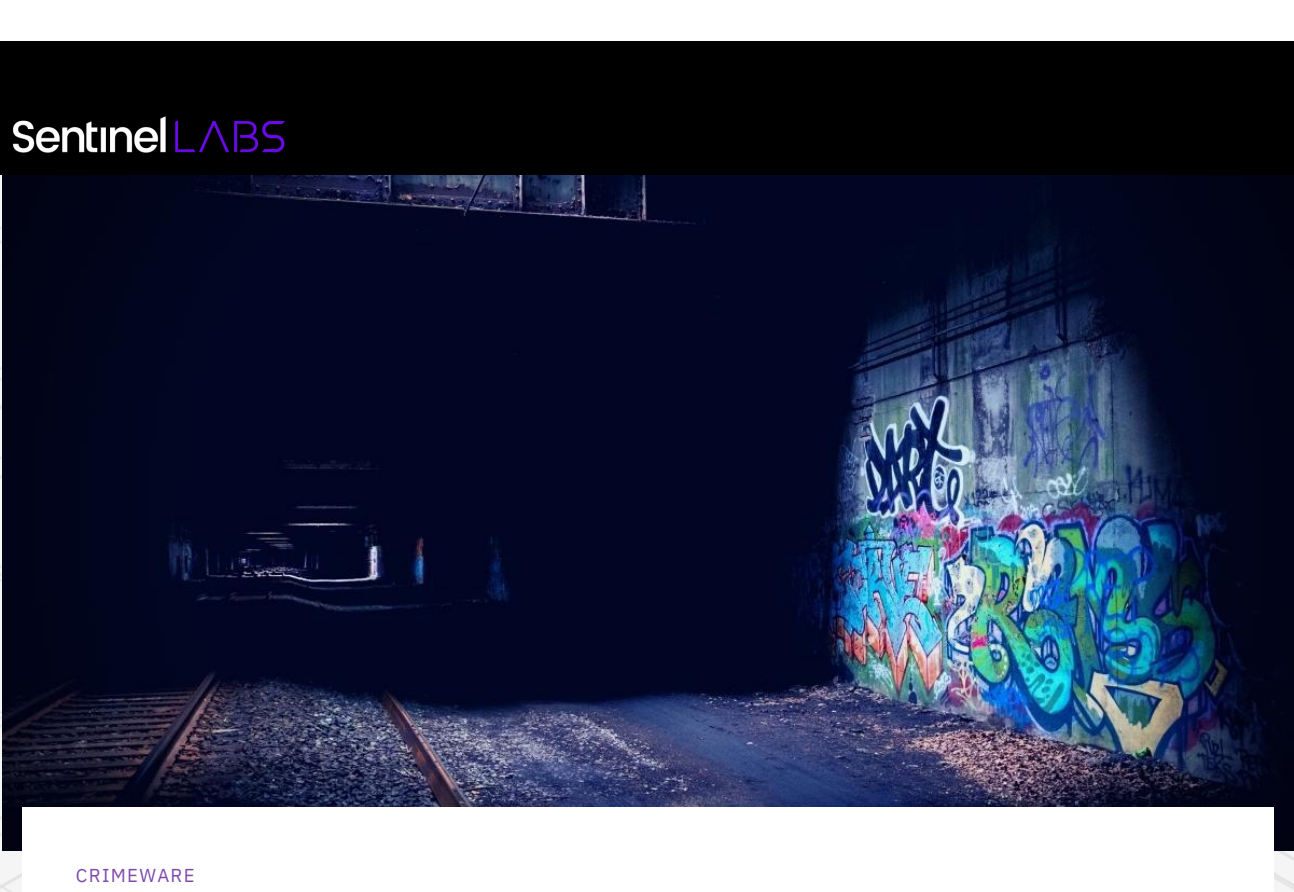


CRIMEWARE

MalVirt | .NET Virtualization Thrives in Malvertising Attacks

ALEKSANDAR MILENKOSKI / FEBRUARY 2, 2023

By Aleksandar Milenkoski and Tom Hegel

Executive Summary

- SentinelLabs observed a cluster of virtualized .NET malware loaders distributed through malvertising attacks.
- The loaders, dubbed MalVirt, use obfuscated virtualization for anti-analysis and evasion along with the Windows Process Explorer driver for terminating processes.
- MalVirt loaders are currently distributing malware of the Formbook family as part of an ongoing campaign.
- To disguise real C2 traffic and evade network detections, the malware beacons to random decoy C2 servers hosted at different hosting providers, including Azure, Tucows, Choopa, and Namecheap.

Overview

While investigating recent malvertising (malicious advertising) attacks, SentinelLabs spotted a cluster of virtualized malware loaders that has joined the trend. The loaders are implemented in .NET and use virtualization, based on the KoiVM virtualizing protector of .NET applications, in order to obfuscate their implementation and execution. We refer to these loaders as MalVirt (a recently observed and likely related implementation is referred to as KoiVM Loader). Although popular for hacking tools and cracks, the use of KoiVM virtualization is not often seen as an obfuscation method utilized by cybercrime threat actors.

Among the payloads that MalVirt loaders distribute, we spotted infostealer malware of the Formbook family as part of an ongoing campaign at the time of writing. The distribution of this malware through the MalVirt loaders is characterized by an unusual amount of applied anti-analysis and anti-detection techniques.

The current spikes in threat actors using alternative malware distribution methods to Office macros, such as malvertising, Windows Shortcuts (LNK files), and ISO files, comes as a response to Microsoft blocking by default Office macros in documents from the Internet. Malvertising is a malware delivery method that is currently very popular among threat actors, marked by a significant increase in malicious search engine advertisements in recent weeks.

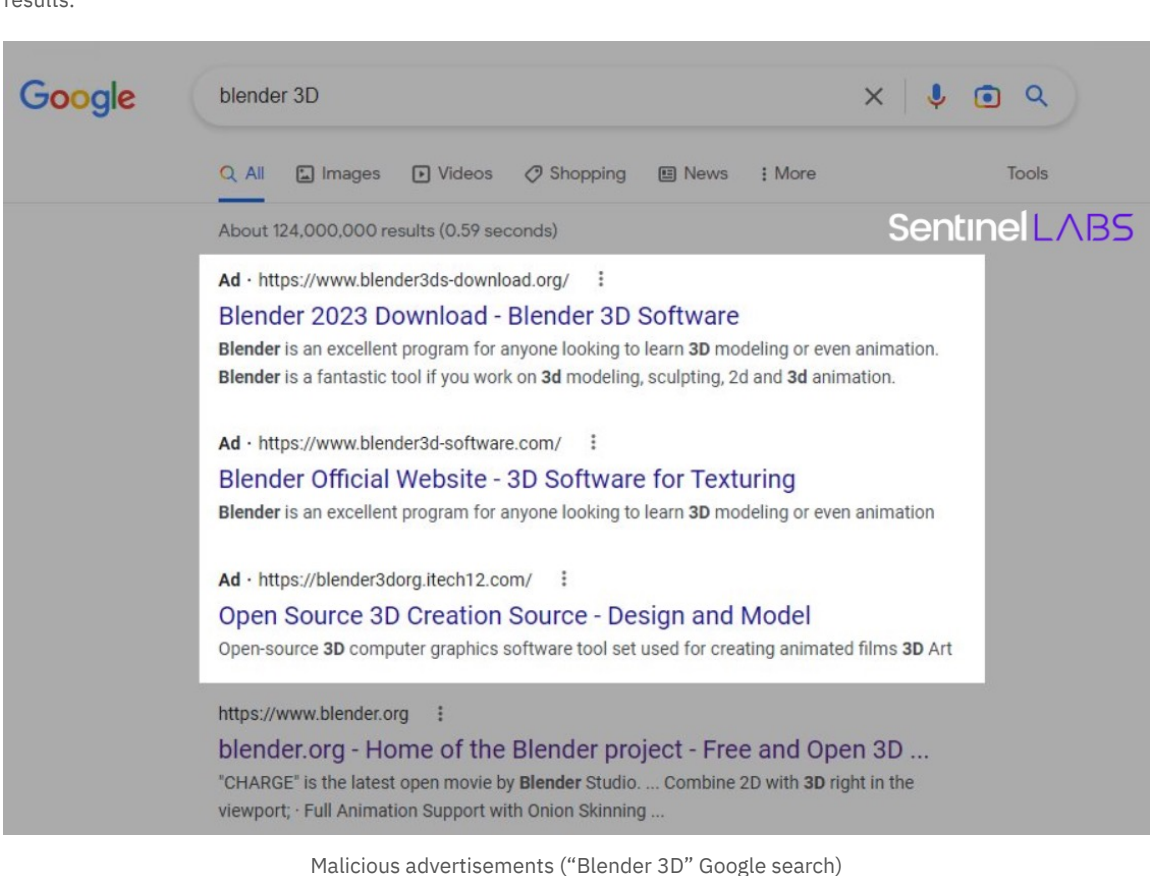
The Formbook family – Formbook and its newer version XLoader – is a feature-rich infostealer malware that implements a wide range of functionalities, such as keylogging, screenshot theft, theft of web and other credentials, and staging of additional malware. For example, one of the hallmarks of XLoader is its intricate disguising of C2 traffic.

This malware is sold on the dark web and is traditionally delivered as an attachment to phishing emails. While it is typically used by threat actors with cybercrime motivations, its use has also been recently observed as part of attacks with potentially political motivations – in September 2022, Ukraine's CERT reported a Formbook/XLoader campaign targeting Ukrainian state organizations through war-themed phishing emails. In the case of an intricate loader, this could suggest an attempt to co-opt cybercriminal distribution methods to load more targeted second-stage malware onto specific victims after initial validation.

We focus on the MalVirt loaders and the infostealer malware subsequently distributed by them in order to highlight the effort the threat actors have invested in evading detection and thwarting analysis.

The MalVirt Loaders

We first spotted a MalVirt sample when performing a routine Google search for "Blender 3D" and examining the Ad results.

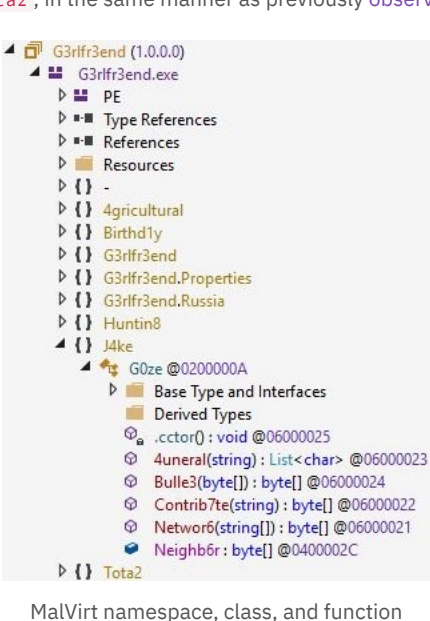


Malicious advertisements ("Blender 3D" Google search)

The MalVirt samples we analyzed have the PDB path

```
C:\Users\Administrator\source\repos\DVS-Calculator-Windows-App-main\Calculator\obj\x86\Debug\Calculat
```

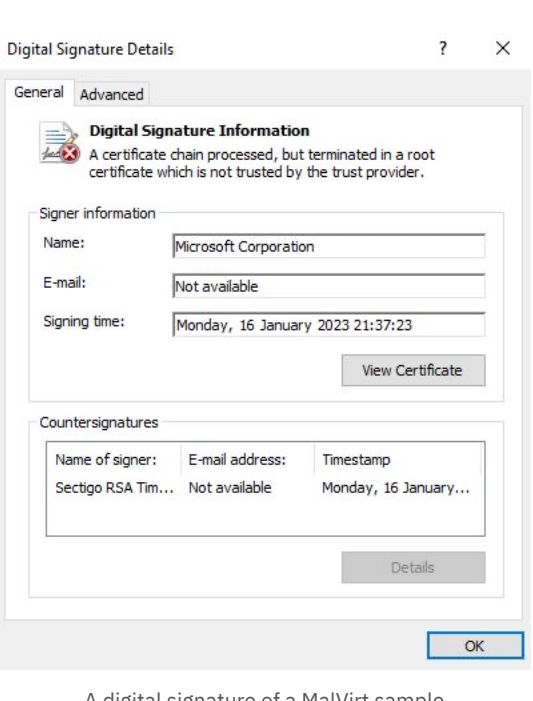
They can be further characterized by obfuscated namespace, class, and function names composed of alphanumeric characters, such as `Birthd1y` or `Tota2`, in the same manner as previously observed Formbook loaders.



MalVirt namespace, class, and function names

The loaders pretend to be digitally signed using signatures and countersignatures from companies such as Microsoft, Acer, DigiCert, Sectigo, and AVG Technologies USA. However, in each case the signatures are invalid, created using invalid certificates or are certificates untrusted by the system (i.e., not stored in the Trusted Root Certification Authorities certificate store). For example, the following certificate appears to be from Microsoft but doesn't pass signature validation.

- Name: Microsoft Corporation
- Thumbprint: 8c2136e83f9526d3c44c0bb0bccc6cf242702b16
- Serial Number: 00b6bce5a3c0e0111b78adf33d9fdc3793



A digital signature of a MalVirt sample

The MalVirt loaders we analyzed, especially those distributing malware of the Formbook family, implement a range of anti-analysis and anti-detection techniques, with some variations across MalVirt samples. For example, some samples patch the `AmsiScanBuffer` function implemented in `amsi.dll` to bypass the Anti Malware Scan Interface (AMSI) that detects malicious PowerShell commands.

Further, in an attempt to evade static detection mechanisms, some strings (such as `amsi.dll` and `AmsiScanBuffer`) are Base-64 encoded and AES-encrypted. The MalVirt loaders decode and decrypt such strings using hardcoded, Base64-encoded AES encryption keys.

```
Aes aes = Aes.Create();
aes.Key = G0ze.Networ6(new string[]
{
    "b2","8MVPVFqNBA3","/ibz2","+L0ZI923jcDo","AVtkXzb912gQY="
});
aes.Mode = 2;
aes.Padding = 2;
byte[] result;
using (ICryptoTransform cryptoTransform = aes.CreateDecryptor())
{
    byte[] array = cryptoTransform.TransformFinalBlock(T2n, 0, T2n.Length);
    result = array;
}
```

String decryption

We also observed MalVirt samples evaluating whether they are executing within a virtual machine or an application sandbox environment. For example, detecting the VirtualBox and VMWare environments involves querying the registry keys `HKEY_LOCAL_MACHINE\SOFTWARE\Oracle\VirtualBox Guest` and `HKEY_LOCAL_MACHINE\SOFTWARE\VMware, Inc.\VMware Tools`, and evaluating the presence of the drivers `vboxmouse.sys`, `vmmouse.sys`, and `vmhgfs.sys` on victim systems.

Detecting the Wine and Sandboxie application sandbox environments involves evaluating the presence of the `wine_get_unix_file_name` function in the `kernel32.dll` Windows library and the `SbieDll.dll` Sandboxie library on victim systems.

```
[...]
[0070.493] RegOpenKeyExW ([...], lpSubKey="SOFTWARE\\Oracle\\VirtualBox Guest Additions", [...]
[0070.510] GetFileAttributesA (lpFileName="C:\\WINDOWS\\system32\\drivers\\VBoxMouse.sys" [...]
[0070.533] RegOpenKeyExW ([...], lpSubKey="SOFTWARE\\VMware, Inc.\\VMware Tools", [...]
[0070.533] GetFileAttributesA (lpFileName="C:\\WINDOWS\\system32\\drivers\\vmmouse.sys" [...]
[0070.533] GetFileAttributesA (lpFileName="C:\\WINDOWS\\system32\\drivers\\vmhgfs.sys"[...]
[0070.555] GetModuleHandleA (lpModuleName="kernel32.dll")[...]
[0070.563] GetProcAddress ([...], lpProcName="wine_get_unix_file_name")[...]
[...]
```

Detection of virtual machine and application sandbox environments

Process Explorer Driver

We observed MalVirt samples deploying and loading the Process Explorer driver, part of the Windows Sysinternals toolset. This includes a sample (SHA-1: 15DB79699DCEF4EB5D731108AAD6F97B2DC0EC9C) that distributes malware of the Formbook family as part of an active campaign at the time of writing. An assembly named `0onfirm`, which this sample reflectively loads, deploys the Process Explorer driver in the `%LOCALAPPDATA%\Temp` directory under the name *Иисус.sys*. The driver has a valid digital signature issued by Microsoft using an expired certificate (validity period between 15 December 2020 and 12 December 2021).

`0onfirm` then deploys the driver by creating a service named `TaskKill`. The assembly creates the `ImagePath`, `Start`, `Type`, and `ErrorControl` registry values at `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TaskKill` to deploy the driver and configure its loading at system start-up. The name `TaskKill` indicates the potential malicious use of `Иисус.sys` – process termination with kernel privileges.

```
[...]
[0080.239] RegOpenKeyExW ([...],
lpSubKey="\\Registry\\Machine\\System\\CurrentControlSet\\Services\\TaskKill", [...]
[0080.243] RegOpenKeyExW ([...],
lpSubKey="System\\CurrentControlSet\\Services\\TaskKill", [...]
[0080.304] RegCreateKeyExW ([...],
lpSubKey="System\\CurrentControlSet\\Services\\TaskKill", [...]
[...]
[0080.311] RegSetValueExW ([...], lpValueName="Type", [...]
[...]
[0080.312] RegSetValueExW ([...], lpValueName="ErrorControl", [...]
[...]
[0080.313] RegSetValueExW ([...], lpValueName="Start", [...]
[...]
[0080.314] RegSetValueExW ([...], lpValueName="ImagePath", Reserved=0x0, dwType=0x1,
lpData="\\??\\C:\\Users\\vnPxSFeoNN\\AppData\\Local\\Temp\\Иисус.sys", cbData=0x6a[...]
[...]
[0080.851] NtLoadDriver (DriverServiceName="\\Registry\\Machine\\System\\CurrentControlSet\\Services\\TaskKill")
[...]
```

0onfirm deploys and loads *Иисус.sys*

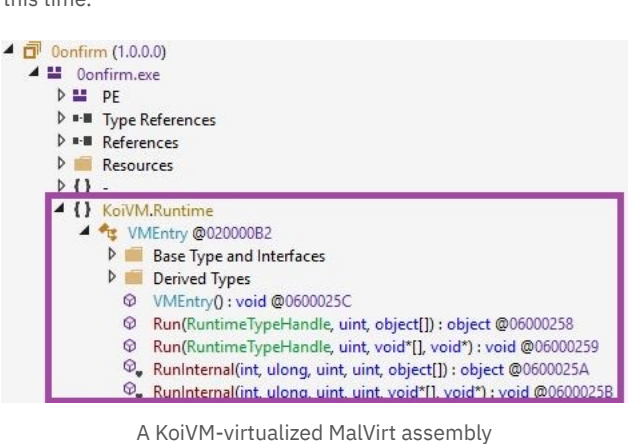
```
<?xml version="1.0"  encoding="ISO-8859-1" ?>
<drivers_list>
    <item>
        <driver_name>Иисус.sys</driver_name>
        <address>FFFFF80E`65EB0000</address>
        <end_address>FFFFF80E`65EBC000</end_address>
        <size>0x0000c000</size>
        <load_count>1</load_count>
        <index>160</index>
        <file_type>Unknown</file_type>
        <description></description>
        <version></version>
        <company></company>
        <product_name></product_name>
        <modified_date>N/A</modified_date>
        <created_date>N/A</created_date>
        <filename>C:\Users\[...]\AppData\Local\Temp\Иисус.sys</filename>
        <file_attributes></file_attributes>
        <service_name>TaskKill</service_name>
        <service_display_name></service_display_name>
        <digital_signature></digital_signature>
    </item>
    [...]
```

Иисус.sys loaded at system start-up (a DriverView output)

Malware in general uses the Process Explorer driver to conduct activities with kernel privileges, such as killing processes of detection mechanisms to evade detection or duplicating process handles for tampering. For example, in late 2022, the use of the `Иисус.sys` driver was observed as part of the deployment (potentially also through a MalVirt loader) of a different payload – Agent Tesla. The open-source tool Backstab also demonstrates the malicious use of the Process Explorer driver.

Obfuscated .NET Virtualization

A hallmark of the MalVirt loaders is the use of .NET virtualization as an anti-analysis and -detection technique. When executed, a MalVirt sample reflectively loads an assembly, such as `0onfirm`, which further orchestrates the staging of the final payload. These assemblies are virtualized using the KoiVM virtualizing protector of .NET applications, modified with additional obfuscation techniques. Code virtualization on its own is among the most advanced methods for obfuscating executables at this time.



A KoiVM-virtualized MalVirt assembly

Virtualization frameworks such as KoiVM obfuscate executables by replacing the original code, such as NET Common Intermediate Language (CIL) instructions, with virtualized code that only the virtualization framework understands. A virtual machine engine executes the virtualized code by translating it into the original code at runtime. When put to malicious use, virtualization makes malware analysis challenging and also represents an attempt to evade static analysis mechanisms.

Tools for the automated de-virtualization of virtualized executables using KoiVM, such as OldRod, can be very effective when facing the standard implementation of KoiVM. OldRod recompiles virtualized code into .NET CIL code in an attempt to recover the original code.

The current standard implementation of KoiVM defines 119 constant variables that the framework uses to virtualize code constructs. These constructs include, for example, flag and instruction opcode definitions. The variables are grouped and ordered according to the constructs they virtualize.

When initialized, KoiVM assigns values to these variables in a designated routine. This is a crucial component of the KoiVM virtualization process. Automated de-virtualization involves detecting this routine by searching for assignment instructions, and using the assigned values to recompile the virtualized code to its native form. However, MalVirt makes automated de-virtualization challenging by using a modified version of the standard KoiVM implementation with obfuscation techniques.

```
static RTMap() {
        const string map = @"
REG_R0                          R0
REG_R1                          R1
REG_R2                          R2
REG_R3                          R3
[...]
FL_OVERFLOW                     OVERFLOW
FL_CARRY                        CARRY
FL_ZERO                         ZERO
[...]
OP_NOP                          NOP
OP_LIND_PTR                     LIND_PTR
OP_LIND_OBJECT                  LIND_OBJECT
[...]
VCALL_EXIT                      EXIT
VCALL_BREAK                     BREAK
[...]
HELPER_INIT                     INIT
ECALL_CALL                      E_CALL
[...]
FLAG_INSTANCE                   INSTANCE
EH_CATCH                        CATCH
[...]
";
```

KoiVM constant variables

The designated KoiVM routine is obfuscated such that it conducts arithmetic operations instead of concise assignments. This is to confuse devirtualization frameworks, such as OldRod, attempting to detect the routine and extract the variable values crucial for accurate de-virtualization.

```
static SQLSTRING939AF857()
{
    int num = 1853837200;
    for (;;)
    {
        num &= 332;
        if ((num ^ 373) == 125)
        {
            null.RBTREEERROR85526C2 = (byte)(392 ^ num);
            goto IL_110;
        }
        num += 363;
        goto IL_2C;
        for (;;)
        {
            IL_56B:
            num += 144;
            if (num + 119 == 790)
            {
                goto Block_38;
            }
            if ((num ^ 106) == 386)
            {
                break;
            }
            null.SQLCONNECTIONPOOLKEYD0E1EDC8 = (byte)(-461 + num);
        }
        null.FUNCTIONNODEF7685B89 = (byte)(-456 + num);
        [...]
```

Obfuscated value assignments

To defeat this obfuscation technique, the values that the modified implementation of KoiVM assigns to the constant variables can be extracted from the memory of the virtualized MalVirt assembly while it executes. The routine can then be patched such that it assigns the appropriate value to each constant variable using concise assignments. This helps a de-virtualization framework to detect the routine and extract the values.

```
:007> !DumpClass /d 00007ffdd154ba70
Class Name:      SQLSTRING939AF857
mdToken:         00000000020000af
File:            0onfirm, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
Parent Class:    00007ffe2f402f68
Module:          00007ffdd13b87e0
Method Table:    00007ffdd1559e90
[...]
NumStaticFields:      77
              MT    Field   Offset                 Type VT     Attr            Value Name

00007ffe2f427d20  40000cc      114          System.Byte  1   static                8 STATUSBARPANELBORDERSTYLEBABF39CA
00007ffe2f427d20  40000cd      115          System.Byte  1   static               96 BORDERTYPE8DE55EA1
00007ffe2f427d20  40000ce      116          System.Byte  1   static              134 DATASYSDESCRIPTIONATTRIBUTE1281E78C
00007ffe2f427d20  40000cf      117          System.Byte  1   static                2 USERSTRINGBUILDER8BF57D7B
00007ffe2f427d20  40000d0      118          System.Byte  1   static               83 ERRARGNOREF68E8549E
[...]
```

Values of constant variables in the memory of a virtualized MalVirt assembly

```
static SQLSTRING939AF857()
{
    SQLSTRING939AF857.STATUSBARPANELBORDERSTYLEBABF39CA = 8;
    SQLSTRING939AF857.BORDERTYPE8DE55EA1 = 96;
    SQLSTRING939AF857.DATASYSDESCRIPTIONATTRIBUTE1281E78C = 134;
    SQLSTRING939AF857.USERSTRINGBUILDER8BF57D7B = 2;
    SQLSTRING939AF857.ERRARGNOREF68E8549E = 83;
    SQLSTRING939AF857.HORIZONTALALIGN65603365 = 4;
    SQLSTRING939AF857.DATARELATIONBBA8CD80 = 152;
    SQLSTRING939AF857.EXPRBINOP176262DF = 106;
    SQLSTRING939AF857.VISUALSTYLESYSTEMPROPERTY4799D2B2 = 15;
    SQLSTRING939AF857.SQLPROCEDUREATTRIBUTEBC939269 = 193;
    SQLSTRING939AF857.FOREIGNKEYCONSTRAINTENUMERATORE88438B4 = 76;
    SQLSTRING939AF857.SQLCOLUMNENCRYPTIONKEYSTOREPROVIDER43518C83 = 251;
    SQLSTRING939AF857.TYPEBITSCHEMAIMPORTEREXTENSIONDE5A4806 = 161;
    SQLSTRING939AF857.ARRAYTYPEBY08CBG884CFFAD = 65;
    SQLSTRING939AF857.LIKENODECE1BDD28 = 2;
    SQLSTRING939AF857.EVENTSYMBOL66DF1E67 = 108;
    SQLSTRING939AF857.SQLCONNECTIONSTRINGBUILDER53EF889A = 1;
    [...]
```

Patched value assignment routine

However, the modified implementation of KoiVM used by MalVirt adds yet another layer of obfuscation – it distorts the original order of the constant variables defined by the standard KoiVM implementation. This confuses de-virtualization frameworks and may lead to incorrect de-virtualization.

Restoring the original order can be a very challenging and time-consuming task. This involves the manual inference of the constructs that each of the 119 variables is used for based on code analysis. Alternatively, one could develop logic to automate this activity, which may prove to be an equally challenging endeavor.

Infostealer Campaign

The infostealer malware samples that the MalVirt loaders distribute are part of an on-going campaign at the time of writing. A campaign is marked by an identifier that is present in HTTP POST and GET requests issued by the malware.

```
POST    http://www.h3lpr3.store/gwmr/

GET     http://www.h3lpr3.store/gwmr/
        ?x7Jt=6fiF78Gp_W&nAZuXsMV=2XCUzVoE8KeJrD7KOVDUzg1w
        3gbP63JD5nt7gMlzUniOKk6M9LcXCUYGAkjZlKgteSl0inhUsK
        S+Nj9B+GzES3B6TdA3Bh5e1WSCyXsHIHrZia938w==
```

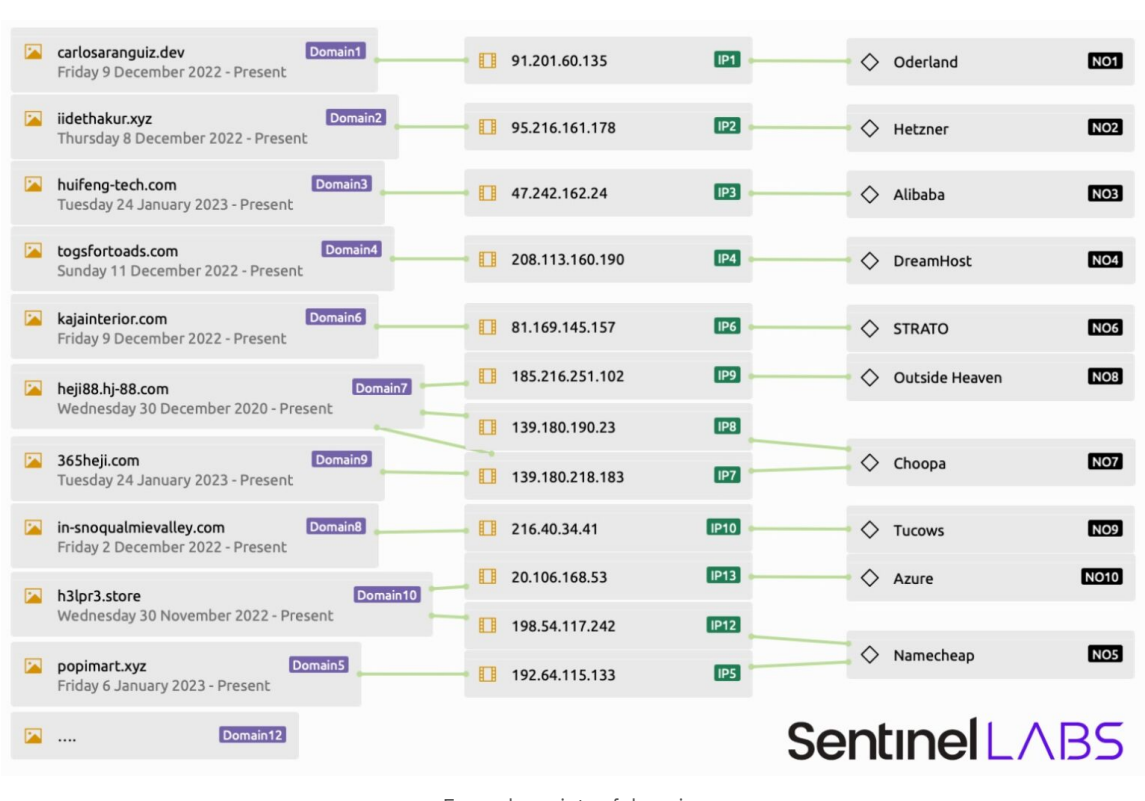
The *gwmr* campaign identifier

Formbook and XLoader have traditionally been distributed via phishing emails and malspam via Macro-enabled Office documents. Our observation of malware of the Formbook family being distributed through MalVirt loaders suggests that it is likely that Formbook and/or XLoader are being (or will be) distributed via malvertising as well. This follows the trend of crimeware actors in their quick shift into Google malvertising.

In addition to the MalVirt loaders, Formbook and XLoader themselves implement considerable protection against analysis and detection, both at executable- and network-level.

Formbook and XLoader disguise real C2 traffic among smokescreen HTTP requests with encoded and encrypted content to multiple domains, randomly selected from an embedded list. Only one of the domains is the real C2 server and the rest are decoys. A sample we analyzed issued HTTP GET and/or POST requests with encoded and encrypted HTTP data to 17 domains (16 endpoints) listed in the IOC table below. Previous research provides detailed information on how XLoader in particular implements this technique.

The technique of camouflaging the true C2 domain through beaconing to multiple domains remains consistent with the previously noted research. The malware beacons to domains containing legitimate and/or unused registered domains. As shown in the following image, as a snapshot of some domains the malware contacts, there is a wide variety of domain times, hosting providers, and age between their relevant registration date.



Example variety of domains

The domains are hosted by a range of providers including Choopa, Namecheap, and multiple others. The random approach to domain selection is beyond the scope of this report; however, it remains a highly effective way of concealing true C2s. XLoader's recent infrastructure concealing techniques in particular should serve as an example of how crimeware can be even more technically sophisticated than many of today's APTs.

At an executable-level, among other anti-analysis techniques, the malware detects the presence of user- and kernel-land debuggers using the NtQueryInformationProcess and NtQuerySystemInformation functions by specifying the `ProcessDebugPort` (0x7) and `SystemKernelDebuggerInformation` (0x23) information classes. Previous research provides a detailed overview of the implemented anti-analysis and -detection techniques.

```
[0114.247] NtQuerySystemInformation (in: SystemInformationClass=0x23, [...]
[...]
[0114.269] NtQueryInformationProcess (in: ProcessHandle=0xffffffff, ProcessInformationClass=0x7, [...]
[...]
```

Debugger detection

Conclusions

As a response to Microsoft blocking Office macros by default in documents from the Internet, threat actors have turned to alternative malware distribution methods – most recently, malvertising. The MalVirt loaders we observed demonstrate just how much effort threat actors are investing in evading detection and thwarting analysis.

Malware of the Formbook family is a highly capable infostealer that is deployed through the application of a significant amount of anti-analysis and anti-detection techniques by the MalVirt loaders. Traditionally distributed as an attachment to phishing emails, we assess that threat actors distributing this malware are likely joining the malvertising trend.

Given the massive size of the audience threat actors can reach through malvertising, we expect malware to continue being distributed using this method.

Indicators Of Compromise

Type	Value	Note
SHA1	15DB79699DCEF4EB5D731108AAD6F97B2DC0EC9C	MalVirt loader sample
SHA1	655D0B6F6570B5E07834AA2DD8211845B4B59200	0onfirm .NET assembly
SHA1	BC47E15537FA7C32DFEFD23168D7E1741F8477ED	Process Explorer driver
SHA1	51582417D24EA3FEEBF441B8047E61CBE1BA2BF4	Infostealer malware payload
Domain	www.togsfortoads[.]com	Contacted domain as part of C2 disguise traffic
Domain	www.popimart[.]xyz	Contacted domain as part of C2 disguise traffic
Domain	www.kajainterior[.]com	Contacted domain as part of C2 disguise traffic
Domain	www.heji88.hj-88[.]com	Contacted domain as part of C2 disguise traffic
Domain	www.headzees[.]com	Contacted domain as part of C2 disguise traffic
Domain	www.in-snoqualmievalley[.]com	Contacted domain as part of C2 disguise traffic
Domain	www.365heji[.]com	Contacted domain as part of C2 disguise traffic
Domain	www.h3lpr3[.]store	Contacted domain as part of C2 disguise traffic
Domain	www.graciesvoice[.]info	Contacted domain as part of C2 disguise traffic
Domain	www.femfirst.co[.]uk	Contacted domain as part of C2 disguise traffic
Domain	www.cistonewhobeliev[.]xyz	Contacted domain as part of C2 disguise traffic
Domain	www.allspaceinfo[.]com	Contacted domain as part of C2 disguise traffic
Domain	www.baldur-power[.]com	Contacted domain as part of C2 disguise traffic
Domain	www.ohotechnologies[.]com	Contacted domain as part of C2 disguise traffic
Domain	www.carlosaranguiz[.]dev	Contacted domain as part of C2 disguise traffic
Domain	www.iidethakur[.]xyz	Contacted domain as part of C2 disguise traffic
Domain	www.huifeng-tech[.]com	Contacted domain as part of C2 disguise traffic

MALVERTISING

ALEKSANDAR MILENKOSKI

Aleksandar Milenkoski is a Senior Threat Researcher at SentinelLabs, with expertise in reverse engineering, malware research, and threat actor analysis. Aleksandar has a PhD in system security and is the author of numerous research papers, book chapters, blog posts, and conference talks. His research has won awards from SPEC, the Bavarian Foundation for Science, and the University of Würzburg.