“Squirrelwaffle” Maldoc Analysis

security-soup.net/squirrelwaffle-maldoc-analysis/

admin September 18, 2021

Summary

Squirrelwaffle is an emerging malware threat noted by several security researchers beginning around September 13th. TheAnalyst, @ffforward noted a new payload delivered on the “TR” botnet.

Brad Duncan at Malware Traffic Analysis also observed that this new loader was being delivered by the same “TR” infrastructure that historically delivered the Qakbot banking trojan. He also noted the name came from a tag in Proofpoint’s ruleset.

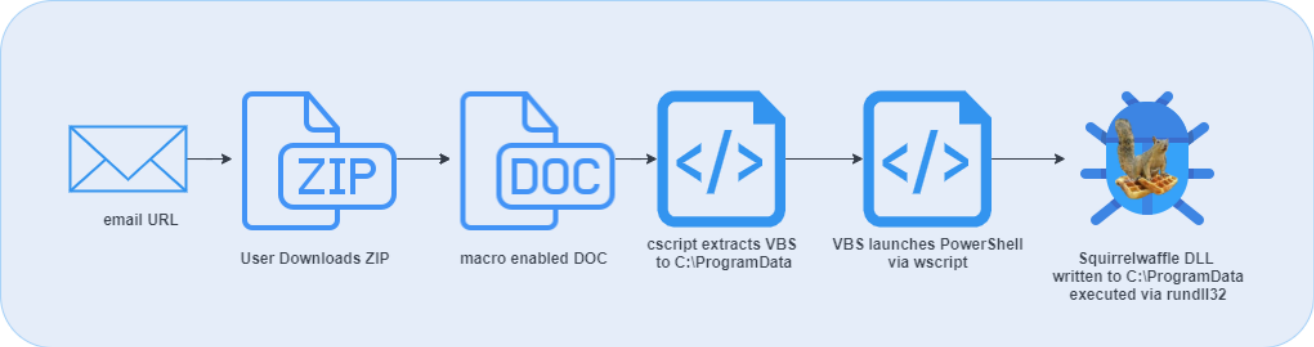
According to Duncan,

> The name “Squirrelwaffle loader” was used in Proofpoint’s Emerging Threats ruleset to identify traffic from this malware.
>
> *https://www.malware-traffic-analysis.net/2021/09/17/index.html*

In this blog, we will take a quick look at a recent Squrrelwaffle maldoc in order to gain some insights into the operators’ TTPs and the malware’s infection chain that I hope will help other researchers and responders in their efforts to identify and combat this new threat.

Delivery and Execution

The recent downloader maldocs appear to be delivered via email campaigns with embedded URLs. Reports also seem to suggest that the campaigns leverage reply chain threadjacking technique that has been commonly deployed in historical Emotet and Qakbot campaigns. If a user clicks on the URL, a ZIP archive containing a Microsoft Word document is served.



Squirrelwaffle Execution Chain

The documents appear to follow the naming convention of “diagram-[0-9]{2,3}/.doc”. These documents are weaponized with macros per usual (more detail on the scheme below). The macro leverages a *cscript* process to extract an embedded VBS script file, writes it to disk, and executes it via a *wscript* process. That VBS script contains an obfuscated *PowerShell* download cradle that attempts to download the Squirrelwaffel payload from a series of five locations.

The loader is written to the C:\ProgramData directory with a naming convention “www[1-5]{1}/.dll”, depending on the C2 from which it is retrieved. The DLL is then executed via a *rundll32* process with an argument to export the “ldr” function. Predictably, a follow on payload has been reported to be CobaltStrike.

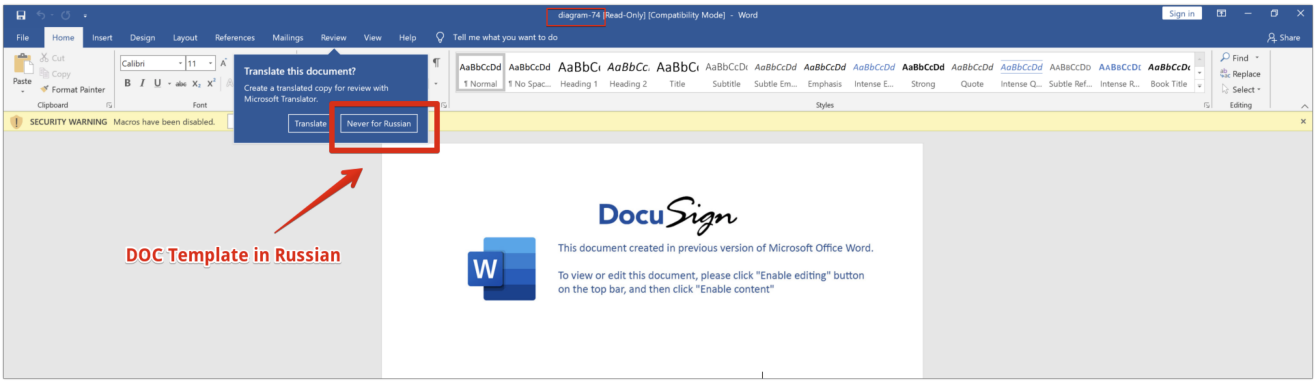
Cobalt Strike Payload you say?

The Maldoc

The Word document is weaponized with VBA macros and leverages a series of scripts to kick off the execution chain and download the Squirrelwaffle payload. If you are following along at home, the document can be found here on VirusTotal:

- filename: inquiry diagram-74.doc
- SHA256: 195EBA46828B9DFDE47FFECDF61D9672DB1A8BF13CD9FF03B71074DB458B6CDF

The document uses a common DocuSign style template, presumably to enhance the perception of security and build a sense of trust with he user. However, the DOC also appears to be composed in Russian, which would hopefully be a red flag for the end user.

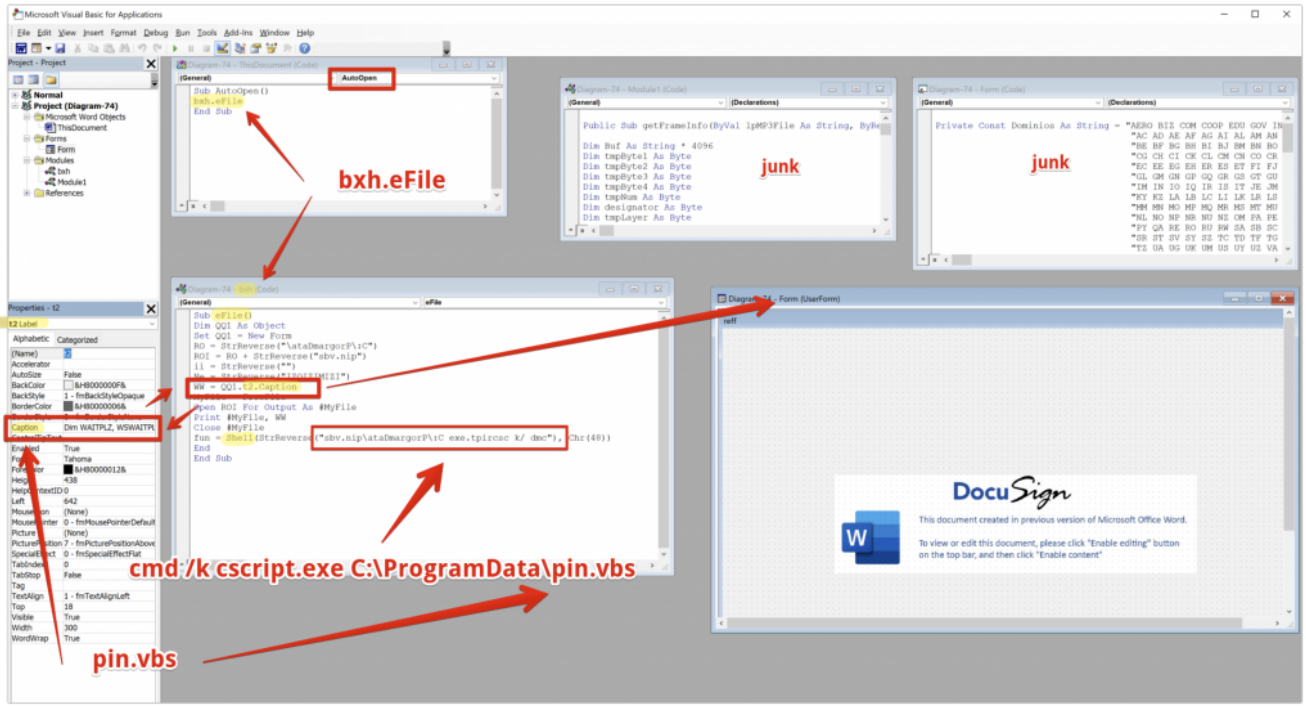


DocuSign Template

Stepping into the VBA editor in Word, we can clearly see multiple modules that contain the VBA code that will kick off the execution chain. The VBA itself is lightly obfuscated, with some variable assignments and string reversals, but on the whole, it is not difficult to identify the code's purposes.

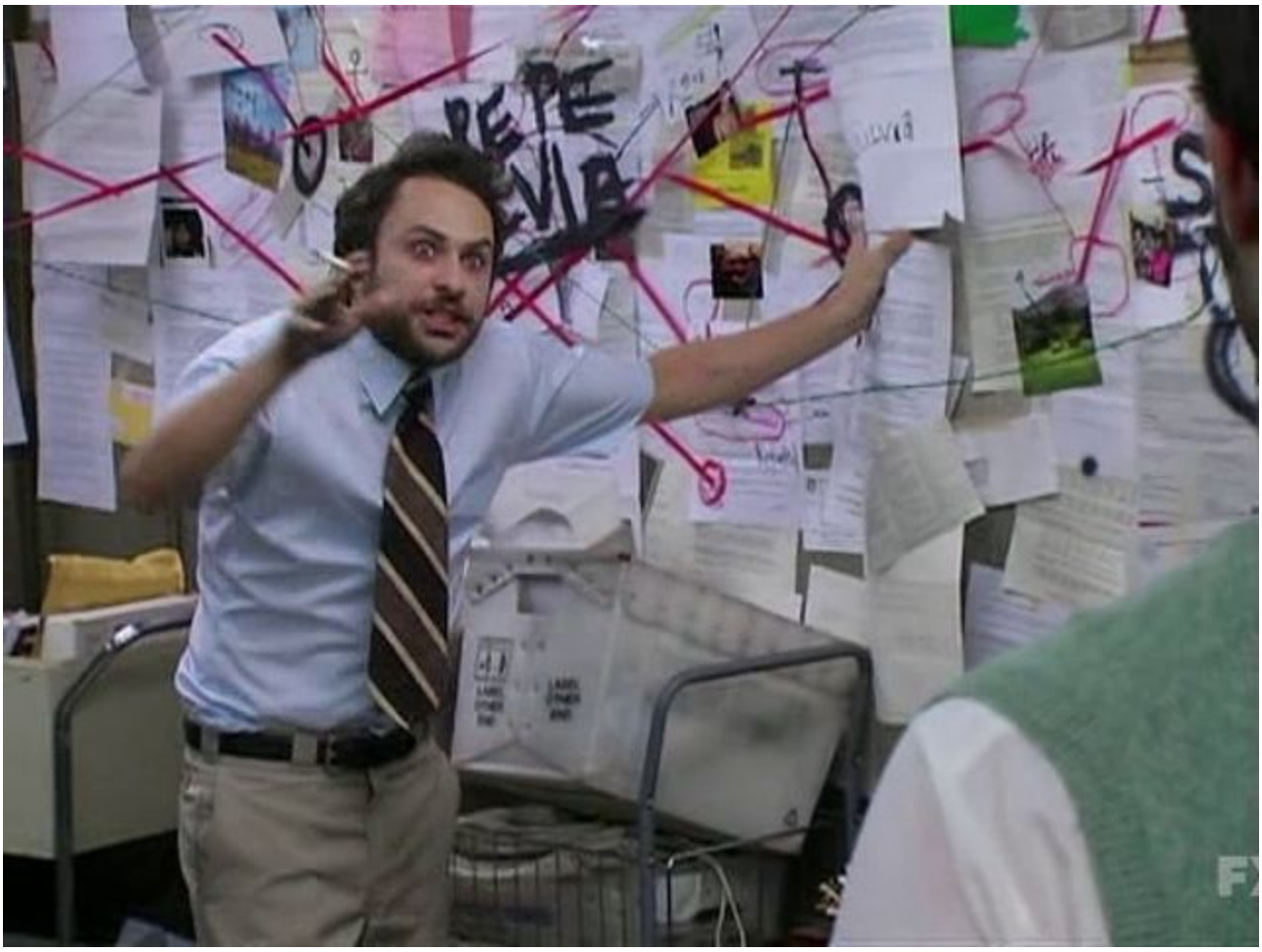
The smoking gun with these malicious macros is always the "Sub AutoOpen()" function, which is the part of the code executed immediately when the user enables the macro content. In this case, this function serves as a pointer to another function called "eFile" in the "bxh" module.

There is also a UserForm object that has the VBS file (pin.vbs) hidden and embedded as the caption of the DocuSign image presented in the main DOC file. This UserForm labeled "t2" has a Caption field where the VBS file is hidden. The VBS file is then extracted from the Caption field of the Label and subsequently written to disk in C:\ProgramData\pin.vbs.



VBA Modules and UserForm object

Apologies, I realized too late that my love for the red arrow knows no bounds.



Me marking up a VBA project

The VBS file is written with a Loop function that cycles through five different URLs that are each hosting Squirrelwaffle payloads. It attempts to download these and writes them to disk in C:\ProgramData and executes them via rundll32. The command line is a variation of the below:

> cmd/c rundll32.exe C:\ProgramData\ww1.dll,ldr

The script itself is obfuscated fairly simply with just some split variable assignments designed to break up the strings such as:

- IEX (alias for the Invoke-Expression cmdlet)
- (New-Object Net.WebClient).Download
- and powershell

pin.vbs

Many static detections are based on these particular strings so this indicates at least some minimal attempt at evasion. However the the URLs hosting the payloads and the commands designed to execute the DLL are clearly visible, and could easily be isolated here by running strings or grep from the command line without ever opening the DOC itself. The threat actors are likely to change this up in the future in order to frustrate automated analysis efforts.

Conclusion

Anyways, that's it, my take on a quick analysis of a recent "Squirrelwaffle" maldoc. It will be interesting to track these campaigns if they become more prevalent in the future. It is always worthwhile to take a look at adversary TTPs, and I hope this information will be helpful to investigators that may be looking to extract IOCs and better understand an emerging threat. As more information becomes available I will need to take a deeper dive into the payload's capabilities, but will need to save that post for another time.