macOS MS Office Sandbox Brain Dump

By Cedric Owens

Published: 2021-05-22 · Archived: 2026-04-05 17:04:07 UTC

Press enter or click to view image in full size

This blog will take a look at some observations regarding what is still possible from the MS Office Sandbox on macOS. This is a combination of insight from others as well as some tests that I have attempted. Hopefully this will help readers better understand what is possible via remote sandboxed access gained through an MS Office macro.

Binaries

What are some macOS binaries that can be executed from sandboxed MS Office macros? Below are some:

- ***osascript***: You can prompt the user for credentials (ex: **osascript -e 'set popup to display dialog \"Keychain Access wants to use the login keychain\" & return & return & \"Please enter the keychain password\" & return default answer \"\" with icon file \"System:Library:CoreServices:CoreTypes.bundle:Contents:Resources:FileVaultIcon.icns\" with title \"Authentication Needed\" with hidden answer'**) , capture clipboard contents (ex: **osascript -e 'return (the clipboard)'**), grab system information (ex: **osascript -e 'return (system info)'**), launch JXA payloads (ex: **osascript file.js**), etc.
- ***curl***: you can make web requests (GET, POST, etc.) and download files
- ***launchctl***: In my testing I have been able to call launchctl, but have not been able to successfully load any plists. Some commands that did work when testing: "launchctl managername", "launchctl managerpid",

"launchctl hostinfo", "launchctl asuser <uid> <bin_path> <args>" (Note: while this works, the sandbox still prevents execution of binaries outside of the sandbox)

- ***ifconfig***: Can display local IP config info (ex: **ifconfig -a**)
- ***dscl:*** can run some dscl commands (ex: "**dscl . list /Users**", "**dscl /Local/Default read Computers/localhost IPAddress**", "**dscl . ls /**", "**dscl "/Active Directory/[domain]/All Domains" ls /users**" (if your macOS host is joined directly to AD)
- ***screencapture***: Can grab screenshots (ex: "**screencapture -x -t jpg out.jpg**"): In this example command, you can dump a screen shot to the sandbox's default directory (ex: /Users/[username]/Library/Containers/com.microsoft.Word/Data)
- ***nc***: Can use netcat to connect to ports on other hosts (ex: "**nc -v 192.168.1.1 22**")
- ***ssh:*** can run the ssh client but I have not been able to successfully complete a login to another host (I used this command: "s***sh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no [username]@[host]***")
- ***python:*** Can run the default python installation on macOS (up to this point 2.7 has been default though I expect that to change in the near future, as scripting runtimes such as python, ruby, and perl are expected to be removed from base macOS installs)
- ***env***: Can run the env command to list info

Disk Access

Get Cedric Owens's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

Adam Chester wrote an excellent blog a while back, where he read the entitlements of macOS Office products and used that information to determine where files could be written to disk from a sandboxed MS Office macro. His blog can be found here:

In a nutshell, he found that files could be written to disk when the files were pre-pended with "~$" (ex: "~$test.zip"). This entitlement still is in place on current versions of MS Office on macOS (as of the date of this blog post), and in testing I am still able to drop files to disk as long as I use this naming convention.

One exception is if you want to drop files to the local sandbox directory (/Users/[user]/Library/Containers/com.microsoft.Word/Data). In that case the prepending "~$" to the beginning of the filename is not needed and you can drop files as normal to this directory. One example of this is the MS Office Macro Payload Generator I wrote a while back for the Mythic C2 framework. Link:

The example above shows how you do not need the leading "~$" to drop or execute files local to the path above.

One other exception I have noticed in my testing: **You can write files to /Users/<user>/.Trash without needing to prepend "~$" to the filename**. **You can also cd into /Users/<user>/.Trash from the MS Office Sandbox. However, you cannot perform an ls to list files there or execute any files in that directory. Additionally in my testing I could only cat files from /Users/<user>/.Trash if I wrote them there.**

Sandbox Escape

published a technique for escaping the macOS MS Office Sandbox:

In a nutshell, this technique creates a .zshenv file (which runs a payload), zips it into a .zip file with "~$" prepended to the filename, uploads it to the user's home directory, and creates a Login Item for that newly added .zip file containing the .zshenv file. When the system reboots the .zip file in the user's home dir is extracted due to the Login Item entry, which then drops the .zshenv file in the user's home directory. Then, each time the user opens a new Terminal, .zshenv executes when then executes a command and control payload.

Summary

I hope you found the information above useful, especially from a defensive perspective in terms of understanding what is still possible even from sandboxed MS Office macros on macOS. A high fidelity detection that will find MS Office macro executions in general on macOS is searching for:

[Any MS Office Product] → /bin/sh

Any MS Office product that spawns a shell environment (/bin/sh, /bin/zsh, /bin/bash, etc.) should be considered malicious. I recommend adding detection content (and if possible prevention content) for this activity, given how high fidelity it is.

Source: https://cedowens.medium.com/macos-ms-office-sandbox-brain-dump-4509b5fed49a