A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne *blog*

EN ⌄

How AppleScript Is Used For Attacking macOS

March 16, 2020

by Phil Stokes

PDF

When we think about security on macOS and the tools used by offensive actors, whether those are real in the wild attacks or red team exercises, we tend to think of things like python scripts, shell

A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne blog   🌐 EN ⌄   ☰

AppleScript in macOS malware

the security field on AppleScript – a built-in macOS

technology – despite the fact it's been around for   Using AppleScript Without Apple Events

as long as Python and predates macOS 10 itself by

8 or 9 years.   Using AppleScript to Avoid Detection

As I'll show in this post, AppleScript is widely used   Conclusion

by offensive actors. This includes its use in adware,

its use for tasks such as persistence, anti-analysis,

browser hijacking, spoofing and more. Worryingly,

given the lack of attention paid to AppleScript in | Search ...   🔍 |

the research community, that is all without even

leveraging some of AppleScript's most powerful or

unique features, some of which we'll cover below   ## Sign Up

(others I've written about before here).   Keep up to date with our weekly digest of articles.

How AppleScript Is Used For Attacking macOS | Business Email | > |
By Phil Stokes

                                                          By clicking Subscribe, I agree to the use of my personal data in accordance with SentinelOne Privacy Notice. SentinelOne will not sell, trade, lease, or rent your personal data to third parties. This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

                                                          ## Recent Posts

                                                          Safely Expanding the Frontiers of AI & LLMs | S Ventures' Investment in Galileo

                                                          October 25, 2024

A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne blog   🌐 EN ⌄   ☰

(Part 1)
October 23, 2024

Python, a cross-platform, beginner-friendly scripting language that has achieved widespread adoption and praise, AppleScript is a language peculiar to macOS; not only can you NOT use it on other Desktop operating systems like Windows and Linux, you can't even use it on Apple's other operating systems like iOS and iPadOS.

Blog Categories

Cloud

Company

Data Platform

Feature Spotlight

For CISO/CIO

From the Front Lines

Identity

Integrations & Partners

macOS

PinnacleOne

The Good, the Bad and the Ugly

As a language, AppleScript has a reputation for being quirky, slow and difficult to develop even simple scripts with. It's quirky because it attempts to use "natural language" but in a grammar that is entirely artificial, often inconsistent and frustratingly unintuitive. It's also incredibly verbose. Compare the code for the simple task of counting the number of items in `/usr/bin` directory. As ever, a shell script will always be the most concise:

```
ls -l /usr/bin | wc -l
```

Python is a little more verbose, but still fairly clean and familiar:

SentinelOne blog

The AppleScript version, however, is something of an entirely different nature.

```
tell application "System Events"
        set theFiles to name of every
        count of theFiles
end tell
```

AppleScript is also slow in execution because, among other things, the underlying technology involves constructing and sending Apple Events through an archaic interface called the Apple Event Manager that was written for Apple's System 7 operating system (released in 1991) and not optimized for performance even back then.

A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne blog

🌐 EN ⌄   ≡
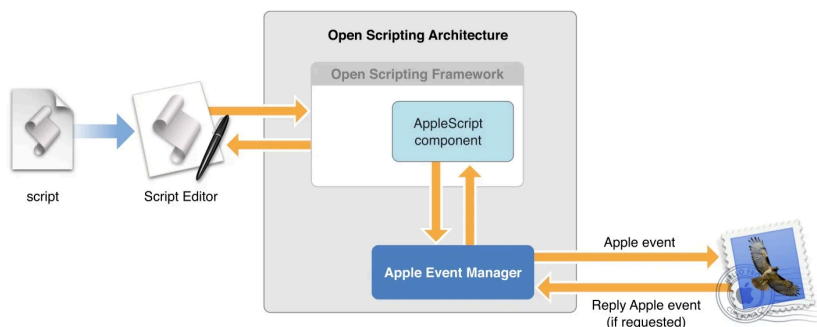
with AppleScript because most people who come to it will attempt to use the free, built-in but notoriously spartan (Apple)Script Editor.app, which lacks almost every and any feature developers normally expect and need. There's no debugger, there's no variable introspection, there's no code snippets or effective code completion, to name just a few missing features.
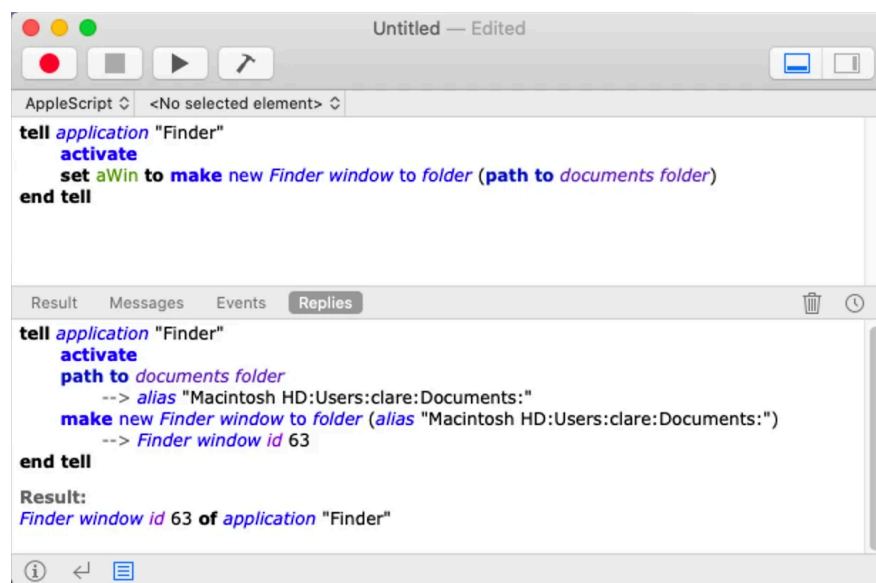
Until recently, the only 3rd party alternative to Script Editor was priced at $200 and had a time-limited, 20-day demo period.

SentinelOne blog   🌐 EN ⌄   ☰

effectively puts AppleScript at the bottom of the list for most people when it comes to choosing a useful or productive programming language. As a result, despite being with us for nearly 30 years, AppleScript is barely used by the majority of Mac admins, Mac developers or Mac end users. Indeed, AppleScript may have a good claim to being the Most Unloved and Unlovable Programming Language Ever.

So Why Do the Bad Guys Love Using AppleScript?

AppleScript was designed for automation and interapplication communication: the goal being to allow ordinary users to chain together repetitive tasks and execute them without further user interaction. For example, you can have Mail.app automatically trigger a script when it receives an email from a certain sender, or with a particular keyword in the subject line or content, extract whatever details you want from the email, and then populate a database in Excel or Numbers

SentinelOne blog   EN ⌄

is set up.

And as it turns out, automating interapplication communication and sidestepping user interaction is a godsend for malware authors. What could be more useful than bending popular applications like email clients, web browsers and the Microsoft Office suite to your will without needing to involve the user (*aka* in this scenario, **the victim**)?

And so, despite its general lack of appeal in almost all possible audiences for a scripting or programming language, there is one audience that does use AppleScript widely, if not particularly artfully or cleverly, and that is threat actors. Let's look at some examples.

A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne blog   🌐 EN ⌄ ☰

macOS malware

A recent browser hijacker targeting Safari installs a hidden LaunchAgent that, via a shell script, loads, compiles and executes AppleScript.

It starts with a shell installer script packaged inside a DMG that's supposed to contain an application called 'PDFConverter4u'. But in a hidden `.assets` folder on the disk image is a first stage shell script:

```
5f198e82c0cf9a9f7d7a8d01273a6ad75a17a95960d8996d
cdd028922b3d97bc
```

This unpacks and executes a second stage shell script:

```
55529224e9f70f5cab007e2ca98f6aec5cf31eb923fdfc09
f60c01cc45c80666
```

that executes another shell script containing the following AppleScript code, launched via `osascript`.

```
cdaa2121d79031cf39159198dfe64d3695a9c99ff7c3478a
0b8953ade9052ecc
```

The purpose of the AppleScript is to replace the user's search query on popular search engines google, bing and yahoo, with one provided by the attacker's shell script. It's a quick and easy way for bad actors to make money out of clicks which negatively impacts the victim's productivity.

While this particular sample comes packed in a separate file, many others write their AppleScript

A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne blog   🌐 EN ⌄ ☰

The following strings extracted from a Bundlore installer show that the code tries to force enable JavaScript execution in Google Chrome, then uses AppleScript to execute it in the active tab of the browser's front window.

Strings from
`41e0d31d52cb93f6a5020a278e8f360a6e134e6cc7092b4a5e575ac8b96a8d74`

The next sample is a variant of a Pirrit malware.

`21331ccee215801ca682f1764f3e37ff806e7510ded5576c0fb4d514b4cf2b7c`

SentinelOne blog   🌐 EN ⌄ ☰

Here's some of the decoded base64 targeting
Firefox and attempting to perform automated
keystrokes to copy the current URL to the user's
clipboard before replacing the URL to that of the
attacker's choice:

SentinelOne blog   EN ⌄

something that many ordinary users and

developers have overlooked about AppleScript, but

which offensive actors have not: you can use

AppleScript to execute any other kind of script,

including python scripts like this one which drops

the Empire exploit kit.

And what's true of AppleScript and Python, is true

of AppleScript and Perl, AppleScript and Bash and

indeed AppleScript and absolutely any command

line tool at all: you can call them all and bring their

functionality into your AppleScript and combine

that with other utilities.

The examples above all use what we might call 'vanilla AppleScript'. That is, the native AppleScript language that's been around since the early days of the platform. But starting in Yosemite, 10.10 and continuing up to and including the most recent version of macOS, AppleScript has been given increasing power through access to Cocoa frameworks, and this opens up the possibility of creating full-blown, powerful programs and applications with nothing other than AppleScript itself. Objective C executed through AppleScript is, speed-wise, more or less on a par with Objective C executed in a [MachO](#) binary.

And interestingly, although we haven't seen threat actors making use of these powerful capabilities so far, there's at least two reasons why we may well do so in the future: first to avoid detection, and second, because of the easy availability of a good development environment.

A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne blog   🌐 EN ⌄   ☰

Using AppleScript to Avoid Detection

Avoiding detection on execution is a primary objective for all malware (even [ransomware](#), which doesn't want to get noticed until after execution). AppleScript offers offensive actors a plethora of ways to execute. In addition to simply executing a `.scrpt` file, you can run AppleScripts from [Mail rules](#), from a shell script, in memory, from the command line, from within a MachO, in a plain text, uncompiled file, from an Automator workflow, from a [Folder Action](#), a Finder Service or from a Calendar event.

SentinelOne_{blog}

🌐 EN ⌄

Because of AppleScript's ability to execute Objective C code without needing a compiled binary, this opens up a number of interesting attack possibilities. It also potentially opens up the ability to bypass detection tools based on Apple's new kextless security framework introduced in [macOS Catalina 10.15](#).

In an excellent post by Cedric Owens called [Taking the macOS Endpoint Security Framework For A Quick Spin](#), Owens sets out to test what can and cannot be detected using three recently developed 3rd-party security tools that leverage the new [Apple Endpoint Security framework](#).

One of the interesting things that Owens found was that if you tried capturing the user's clipboard via `osascript` and vanilla AppleScript, this activity

```
osascript -e "the clipboard"
```

However, when using the native Cocoa API, NSPasteboard, none of the Endpoint Security framework-powered tools Owens tested appeared to capture that activity. But now, of course, we can execute NSPasteboard natively from AppleScript, too!

Notice that our one line, simple but also detectable `osascript` has turned into about 14 lines of complex-looking AppleScript-ObjC. Few people, certainly not I, would want to try and construct that kind of code in Script Editor.

However, the problem of developing complex AppleScripts is now more or less a thing of the past. The [3rd party alternative](#) mentioned earlier in

SentinelOne blog

🌐 EN ⌄

great deal of boilerplate code like that used in the script above straight into your scripts. And it provides developer-friendly functionality like code completion and API lookups that really take the pain out of developing AppleScript code.

Let's look at another example. A lot of offensive operations want to avoid targets that are running particular software. Little Snitch is a prime example, various VM software is another. We can easily get a list of running apps by name and test for those, again directly by calling into Cocoa APIs, this time via NSWorkspace.

SentinelOne blog

🌐 EN ⌄

In other words, by leveraging AppleScript's hook into Cocoa frameworks, we can execute native code without the overhead of building MachO binaries or MachO apps (although you can do both of those with AppleScript, too!). We can do this filelessly so that we don't get caught by new 'kextless' tools such as those tested by Owens, and we can execute this code in far more ways than any other kind of code available on macOS, whether that's shell scripts, Python scripts or native macOS bundles.

Conclusion

already seen threat actors taking advantage of AppleScript despite those obstacles in the past, it's only reasonable to assume that they may delve deeper into what this unique language has to offer in the future. Thanks to the native hook into Objective C and the powerful Cocoa frameworks, the variety of execution methods and now the availability of an excellent, free-to-use IDE, AppleScript has become a tool that is powerful, versatile and easy-to-develop with.

Attackers will always look to exploit the things defenders ignore, and to say that AppleScript has been ignored by the security community thus far is an understatement. I have elsewhere described AppleScript as "the PowerShell of macOS". Certainly, it's time we stopped thinking of AppleScript as the Most Unlovable Programming Language Ever and recognize that it may actually be the One macOS Programming Language to Rule Them All.

A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne *blog*   EN ⌄

- [Is SearchMine Adware Teeing Up Your Endpoints For Other Threat Actors?](#)
- [macOS MetaStealer | New Family of Obfuscated Go Infostealers Spread in Targeted Attacks](#)
- [AI-Driven Real-Time Malware and Ransomware Detection for NetApp](#)
- [Backdoor Activator Malware Running Rife Through Torrents of macOS Apps](#)
- [Protecting macOS | 7 Strategies for Enterprise Security in 2024](#)
- [macOS Adload | Prolific Adware Pivots Just Days After Apple's XProtect Clampdown](#)

Read More

A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne blog   EN ⌄

Defeat every attack, at every stage of the threat lifecycle with SentinelOne

Book a demo and see the world's most advanced cybersecurity platform in action.

SentinelLabs: Threat Intel & Malware Analysis

We are hunters, reversers, exploit developers, & tinkerers shedding light on the vast world of malware, exploits, APTs, & cybercrime across all platforms.

MITRE Engenuity ATT&CK Evaluation Results

SentinelOne leads in the latest Evaluation with 100% prevention. Leading analytic coverage. Leading visibility. Zero detection delays.

SentinelOne Secure Tomorrow™

Company

Our Customers
Why SentinelOne
Platform
About
Partners
Support

Resources

Blog
Labs
Product Tour
Press
News
FAQ

A Leader in the Gartner® Magic Quadrant™   Read the Report →

SentinelOne blog

EN ˅

Master Subscription Agreement

Investor Relations

Sign Up For Our Newsletter

| Business Email | → |

Language

English ˅

Global Headquarters

444 Castro Street
Suite 400
Mountain View, CA 94041

+1-855-868-3733

sales@sentinelone.com