Mac-A-Mal: An Automated Framework for Mac Malware Hunting

Pham Duy Phuc
pham.duy.phuc@sfylabs.com
SfyLabs

Fabio Massacci
fabio.massacci@unitn.it
University of Trento

Abstract

As Mac systems grow in popularity, so does macOS malware whilst macOS malware analysis is still lagging behind, even when researchers deal with malicious behaviors in the user space. To amend this shortcoming, we have come up with macOS analyzer for malware Mac-A-Mal: A system for behavioral monitoring of components at kernel level which allows analysts to automatically investigate malware on macOS, broadly extending what is available today with Cuckoo sandbox. By leveraging on kernel-level system calls hooking, the framework is able to detect and mitigate malware anti-analysis techniques. In particular, it combines static and dynamic analysis to extract useful information and suspicious behaviors from malware binaries, their monitored behaviors such as network traffic, malware evasion techniques, persistence methods, file operations etc., without being detected by common Mac malware evasion techniques. We have used the framework to evaluate thousands macOS samples to estimate how widespread Mac malware variants and families are today (thanks to VirusTotal). Mac malware in 2017 demonstrates a drastic improvement by using evasion techniques. Overall, we used our systems to classify the dataset and found that 85% of collected samples are adware, 49% of classified variants belong to backdoor/trojan. By hunting Mac samples on VirusTotal, we found a so-far-undiscovered organized adware campaign which leverages several Apple legitimate developer certificates, a few of other undetected keyloggers, and trojan samples participating in APT32 OceanLotus targeting Chinese and Vietnamese organizations, as well as hundreds of malware samples which have otherwise low detection rates.

1 Introduction

Contrary to popular belief, Mac is not immune from malware. According to a cyber-security report from Bit9 and Carbon Black Threat Research team, Mac devices have seen more malware attacks in 2015 than the past five years combined. Mac malware grew 744% with around 460,000 instances detected, says McAfee report in 2016. The trend in Mac malware was not slowing down in 2017, there were nearly 300,000 new instances of macOS malware in the first 3 quarters of 2017, lifting the year-total to over 700,000.[6]

The malware problem has become bigger and stronger over years, while the majority of malware targets at Microsoft Windows operating system, other operating systems have become relevant targets as well. By understanding how they behave, what these threats are, and engaging in ways to detect them, we can contribute to more secure and stable solutions for Mac computing experience. However, research about macOS malware and relevant solutions to automated Mac malware analysis are limited. While many types of state-of-the-art malware on Windows platform took decades to develop from the first known malicious software to happen in the wild, now they start emerging on Apple computers in a shorter time.

Static malware analyses are mostly used to analyze macOS malware. E. Walkup [10] was able to extract Mach-O structure, import libraries (DyLib) and functions feature from a data set consisting of 420 malware samples and 1000 goodware over 20 machine learning models. Furthermore, S. Hsieh et al.[4] presented a study of classifying Mac OS X malware in 2016 with a set of features extracted from Mach-O metadata using tools such as `nm`, `otool`, or `strings` on VirusTotal sample collection of 2015-2016. They also included derivative numerical features created from meta information, which are introduced in learning-based malware classification, e.g. function call distribution, structure complexity, etc. D. Dorsey [3] presented a weighted distance metric solution to generate and compare assembly mnemonics signature of Mach-O binaries using `mpesm`. The study successfully discovered 7 over 18,000 samples to be UPX compressed, and 3 groups signature matched at least 85% of malicious samples. However, no significant signature could identify the difference between malicious samples and 17,000 known benign samples.

Regarding dynamic Mac malware studies, V. Mieghem[8, 9] presented a novel generic behavioral detection method based on system calls names and prevention mechanism for malware on OS X. Sequences of system call traces are analyzed, from which certain malicious system call patterns, interactions with shells and auto-run services appear to be an accurate indication of malware on a system. Three types of user profiles are established to evaluate the detection patterns, resulting in a 100% detection rate and a 0% to 20% false positive rate, depending on the type of user profile. M. Lindorfer et al.[5] have built an OS X honeypot based high-interaction and used it to evaluate more than 6 thousands blacklisted URLs to estimate how widespread malware for Mac OS X is in 2013. Only five websites were found to drop binaries through drive-by downloads, but none of them targeted OS X. Furthermore, they developed a dynamic analysis environment with DTrace and analyzed 148 malicious samples. The result has shown that while some OS X malware families are sophisticated, several fail to perform simple but critical jobs like persistence. A. Case et al. [1, 2] presented malware detection techniques with a specific focus on kernel-mode components for OS X, particularly for rootkit that targets at kernel data and user land malware written in Objective-C for Mac. They utilize Volatility to detect rookit in kernel memory. For Objective-C malicious code, they analyzed important artifacts in the memory and produced output that could easily be used by analysts to isolate and investigate more deeply these behaviors even when the Objective-C runtime maintained state out of the dynamic loader and the code section of executables.

We evaluated some of popular dynamic macOS analyzers and we found that Cuckoo sandbox is an open source project providing most robust solution for malware analysis framework. Although it only supports Windows, Linux but partially Android and macOS. It adopts DTrace which is basically a binary instrumentation platform rather than a malware analysis. Apple does not allow DTrace to monitor official Apple binaries and requires traced software to run under root permission, its tracing techniques can be easily defeated by trivial anti-debugging tricks. Besides, Macsandbox[1] is a modified version of Cuckoo sandbox on Mac, using Dtrace and library injection for process tracing. However, running on user space makes it vulnerable to trivial anti-debug and anti-hook techniques. For analysis on kernel-space region, Fireeye Monitor[2] is a closed source software for manual analysis, which performs process execution, file and network logging.

In this white paper, we study an automated solution for Mac malware analysis framework. We show that it is possible to automatically perform behavioral monitor of process execution, file activities, network traffic, with regards of virtualized environment, malware evasion detection and mitigation.

2 Mac-A-Mal: Automated macOS malware analyzer

Mac-A-Mal is a combination of both static and dynamic analysis. Using static analysis to understand in depth suspicious areas of code, and dynamic analysis to unpack packed binary and monitor malware behaviors. It takes actual behavioral data of malware samples executions inside virtual sandboxes simultaneously, in which it can process multiple samples at once. The sandbox is armored with network sniffer, system calls and behavior logging, as well as anti-evasion from kernel-mode to send back report to analysis machine. Researchers thereafter can review reports to spot any suspect activities such as suspicious network activity, invoked anti debug techniques, persistent activities etc. Researchers can also perform common malware detection techniques such as YARA rules and behavioral signature to automatically detect a whole sample family. The overview design of Mac-A-Mal framework is illustrated in Fig. 1. The analysis machine gathers samples and feed them to the monitor machine(s) running on macOS simultaneously. Data sharing features and default Apple protection mechanisms such as XProtect, Gatekeeper, etc. are disabled to capture malware behavior accurately.

The analyzer processes 2 main tasks:

- *Static analysis*: Parsing Macho executable and display: Symbol table, segment, sections, load commands, dylib, entropy etc. Analyzing other common Mac file types such as DMG, PKG etc. as well as their certificate details.
- *Results collection*: Output to web front-end as well as JSON format which can be easily applied with behavioral signature, machine learning or YARA rules.

The monitor processes following tasks:

- *Dynamic analysis*: Processing monitor behavior under kernel-space with Anti anti-vm, anti-debug etc. logging and mitigation. Additionally, common malware analysis techniques are implemented such as network monitor, dropped files collection, etc. We implement an analysis agent on analysis machine under kernel space, which basically performs: Syscall

[1] `https://github.com/sandialabs/mac-sandbox`

[2] `https://www.fireeye.com/blog/threat-research/2017/03/introducing_monitor.html`

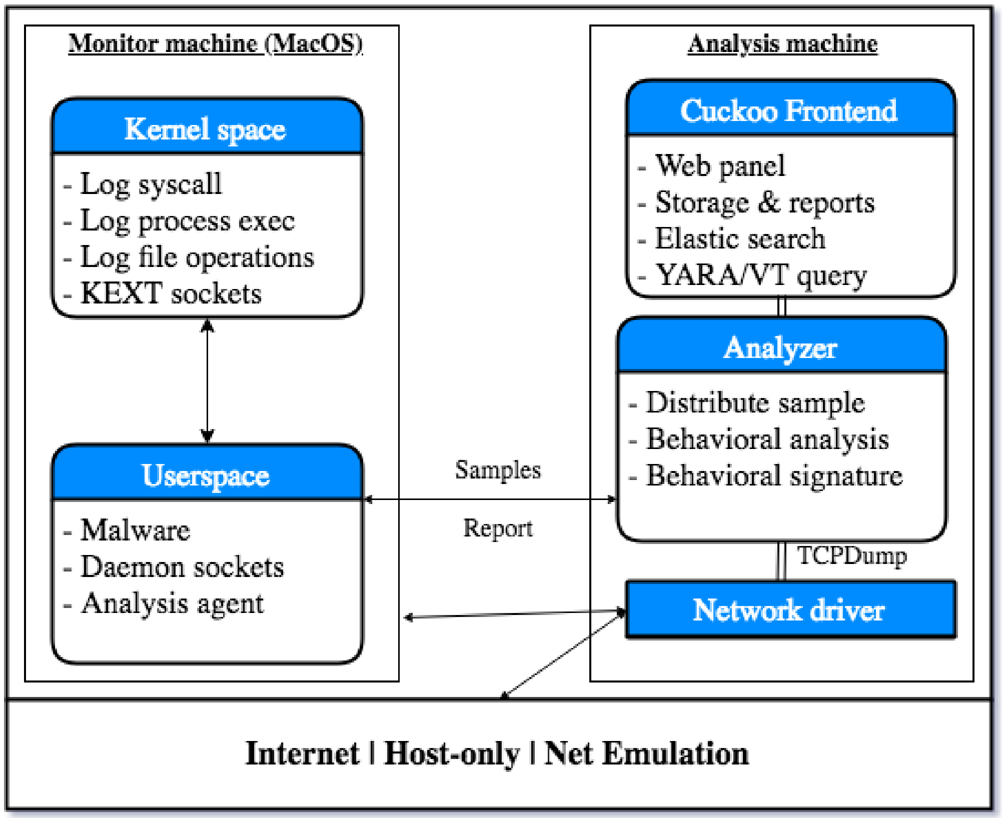


Figure 1: Mac-A-Mal framework design

table discovery technique to unslide KASLR to obtain kernel base image address, then walks through the kernel base image to locate load command segment, and finally performs system table lookups. These proposed techniques are compatible to latest version of macOS High Sierra. Once the sysent table is located, we patch its entries to our defined callback functions. In particular, when a malware invokes system calls, it queries the address of the call in the XNU kernel syscall table, which now is patched to our callback function address. The callback functions will perform execution logging, malware evasion detection and mitigation, as well as post-processing for the calls (forward or drop).

- *Anti evasion*: We develop kernel system calls logging in which existing solutions failed, e.g. by hooking into `posix_spawn()`, `execve()` and copying double pointers between user space and kernel space. Fig. 2 shows how we detect and mitigate evasion techniques by hooking into `ptrace()`, `csrctl()`, `ioctl()` and `sysctl()`.

- *Execute samples via open default handler*: By hooking in kernel space, we can execute samples using default Mac handler - `/usr/bin/open` as well as XPCProxy, in which Dtrace and MacSandbox failed to trace. It means any types of files can be opened with capability of process forks tracing (e.g. .app;.dmg;.doc;.zip etc.) without using other app-opener.

- *No human interaction*: During the analysis, the sandbox implements libraries for taking screenshots of the analysis screen. These screen shots are helpful for analysts to review automated analysis and recognize some cases that needs human interaction, for instance mouse click to confirm installation or password typing for Authorization solver. We use Quartz library to detect the login window then answer password using `KeyboardEvents`. Screenshot images are analyzed to seek for confirmation buttons. If detected, it performs mouse clicks by using Quartz `MouseEvents`. If packages are stored within an Apple disk image, `hdiutil` will be used to attach the disk, and execute all application inside its root folder.

- *Hardening the virtual analysis machine*: We modify virtual machine footprints in machine configuration as well as manipulate its hardware information in kernel level. Besides, default Apple security features

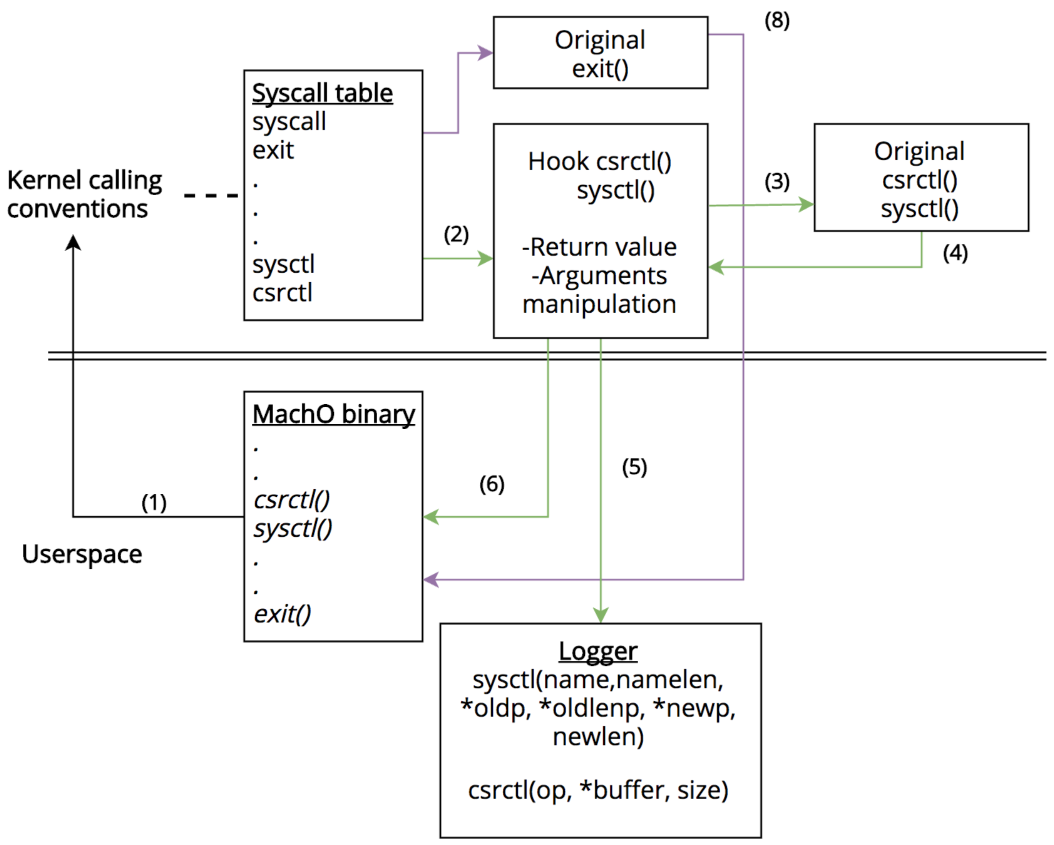


Figure 2: Mac-a-Mal anti environment evasion module using kernel hooking and memory patching. (1) A system call is invoked in the user space, (2) Hook the system call, (3) Return the control to the original system call, (4) Returns value to the user space and manipulate system call arguments if anti-evasion attempts are detected, (5) The return value is logged for further analysis using Logger component, (6) Return values from the kernel space to the analyzer in userspace, (7) the `exit()` system call is not hooked and forwarded back to the user space.

such as XProtect, GateKeeper, etc. are deactivated in order to execute any arbitrary malicious samples.

3 Case studies of undetected Mac malware discovery

These case studies dive deeper into the illustration drawn in Section 2 by explaining detailed analysis on some major malware campaigns which are found by Mac-A-Mal.

- *OSX/Mughthesec*[7]

  The adware campaign was first founded in the wild in August 2017. It pretended to install legitimate Adobe flash and silently install Potential Unwanted Applications (PUA) such as Booking, Advanced Mac Cleaner, SafeFinder Safari extension, AdBlock, etc. Reports show that the campaign dropped some malicious binaries related to AMCleaner adware campaign. It was likely an affiliation advertising campaign, in which adware authors spent some money for buying legitimate Apple developer certificates. By using a combination of behavioral rules (antivm), static rules (suspicious legitimate Apple developer certificate), and suspect network activities based on results from Mac-A-Mal, we discovered 71 signed archives. Attackers had used at least 10 Apple legitimate developer certificates, and only 2 of them were revoked at the time of discovery. Moreover, some of dropped MachO executables were not signed, which means it could transform from adware to backdoor silently.

- *APT32-OceanLotus*

  APT32 was an APT campaign targeted Chinese and Vietnamese infrastructure. It was first discovered by Qihoo 360, and followed up by AlienVault, FireEye and Palo Alto Networks. By study the first generation of Mac OceanLotus samples through our framework, we generated some similar behavioral signatures amongst the family. In March 2017, we found second generation of Mac APT32 which has 0 detection rate over more than fifty Anti-virus vendors by hunting those behaviors on VirusTotal Intelligence service. Our finding shows that the campaign was improved from the first variant by eliminating all Bash command executions. Unlike common Mac malware, this variant tried to execute under user privilege instead of higher privilege that may easily convey malicious behaviors.

4 Discussion

We have processed more than 2000 malicious samples and 85% of the collected samples are adware, which are dominated by OSX/Pirrit and OSX/MacKeeper. That statistic result is well confirmed according to reports from other Anti-virus vendors. Once installed, macOS adware usually persistent deep inside the victim system and starts hjacking browser. Their purpose is to make advertising revenue for attackers by installing PUA or redirecting victims to unwanted websites. They are widespread because technically they are not viruses but can potentially perform malicious activities.

After analyzing the samples set, results retrieved from Mac-A-Mal were later post-processed through a classifier based on numerous rules: code signing authority, persistence indicators, processes creation and shell execution, backdoor port listening, network activities, Tor network indicator, anti-analysis techniques, file creation and modification, browsers change and super-user permission authentication, etc. We also compared them to VirusTotal detection results and other Anti-virus labeled variant name to verify the accuracy. We observed a total of 86 different Mac malware families until 2017, and 49% of them belongs to backdoor/trojan variant.

We study the evolution of macOS malware by using heat map analysis of most Mac malware variants in Appendix Fig. 3 and Fig. 4. We evaluate that macOS malware in 2017 demonstrates a drastic improvement by using anti-analysis techniques. The behaviors can be grouped into following categories: *(i)* persistence, file write, browser modification; *(ii)* root request, bash execution; *(iii)* anti analysis; *(iv)* network, Tor, open port. 5 variants are found applying various evasion techniques including `ptrace()`, `sysctl()`, `sleep()`. Some variants tried to discover running security tools on the system or terminate analysis software (e.g. DTrace, lldb, etc.). Besides, some samples never made any persistence attempt such as XAgent which is an upgraded version of Komplex in 2016 also known for participating in APT28 targeting individuals in the aerospace industry running macOS. It can be seen that Mac malware in 2017 carried out more frequent browser modification activities. It means that macOS malware is becoming more grey, and it has been boosted by numerous adware to redirect victims to fraud web traffic or unwanted advertising. In addition, a large number of increased super-user request behaviors shows that Mac malware is trying to increment phising user passwords in order to grant root access of the system and perform more advanced malicious activities.

5 Future Work

We would like to later apply more robust and advanced techniques for better features extraction from the analysis, and machine learning for a larger scale of Mac samples.

References

[1] Case, A., and Richard, G. G. Advancing mac os x rootkit detection. *Digital Investigation 14* (2015), 25–33.

[2] Case, A., and Richard, G. G. Detecting objective-c malware through memory forensics. *Digital Investigation 18* (2016), 3–10.

[3] DORSEY, D. Analyzing entrypoint instruction differences in mach-o files with mpesm. `https://www.carbonblack.com/2016/03/01/analyzing-entrypoint-instruction-differences-in-mach-o-files-with-mpesm/`, 2006. Accessed: 30-July-2017.

[4] Hsieh, S., Wu, P., and Liu, H. Automatic classifying of mac os x samples. Virus Bulletin, 2016. Accessed: 20-July-2017.

[5] Lindorfer, M., Miller, B., Neugschwandtner, M., and Platzer, C. Take a bite-finding the worm in the apple. In *Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on* (2013), IEEE, pp. 1–5.

[6] Minihane, N., Moreno, F., Peterson, E., Samani, R., Schmugar, C., Sommer, D., and Sun, B. Mcafee labs threats report, December 2017.

[7] Phuc, P. D. What is safefinder/operatormac campaign?, 07/2017 (accessed 30-August-2017).

[8] Van Mieghem, V. Detecting malicious behaviour using system calls. TU Delft Repositories, 2016.

[9] Van Mieghem, V. Behavioural detection and prevention of malware on os x. Virus Bulletin, 2016 (accessed 20-July-2017).

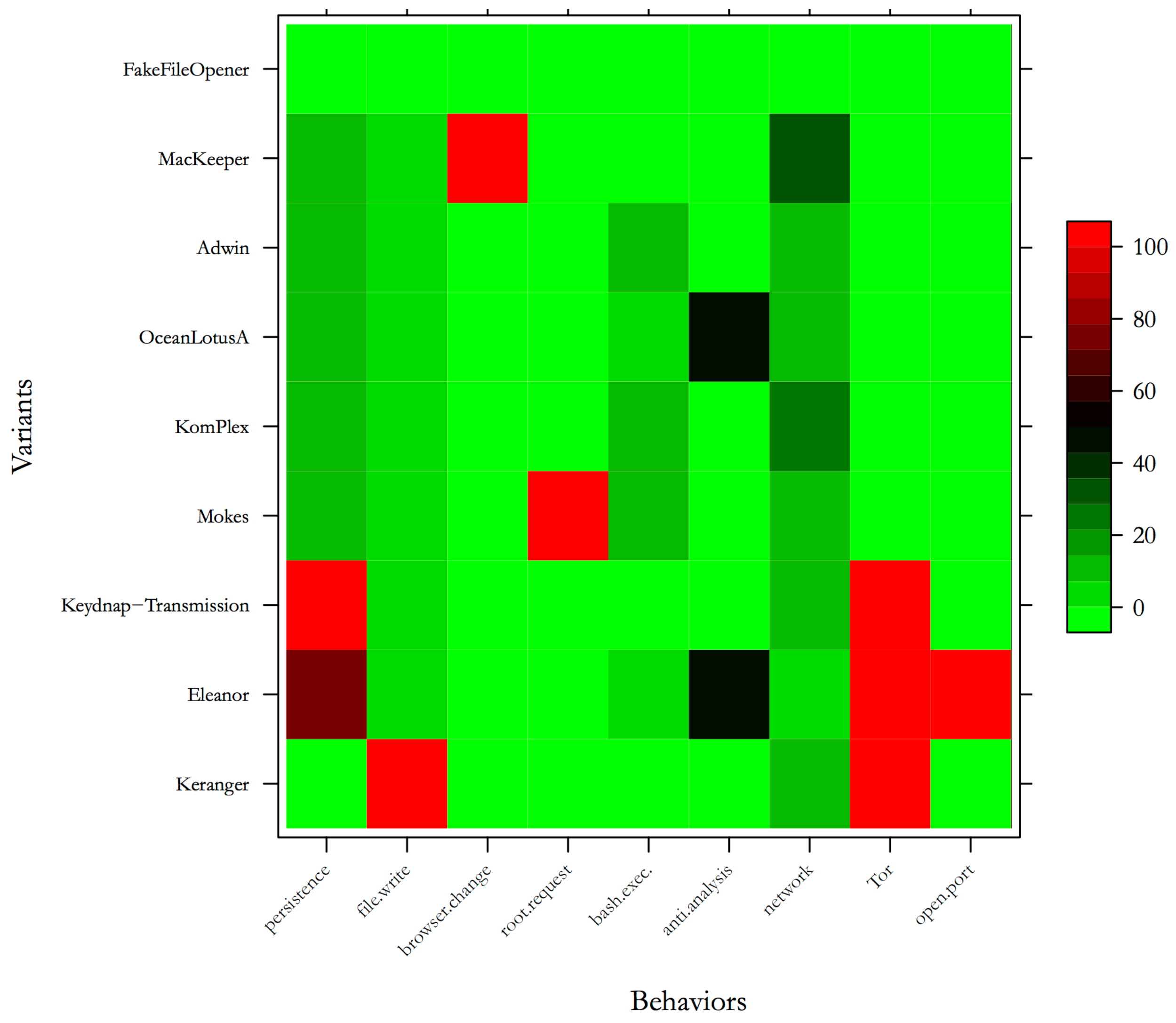
[10] Walkup, E. Mac malware detection via static file structure analysis. University of Stanford, CS 229: Machine Learning Final Projects, 2014.

A Publication of this Work

Source code of the framework is all available for download on Github: `https://github.com/phdphuc/mac-a-mal`.

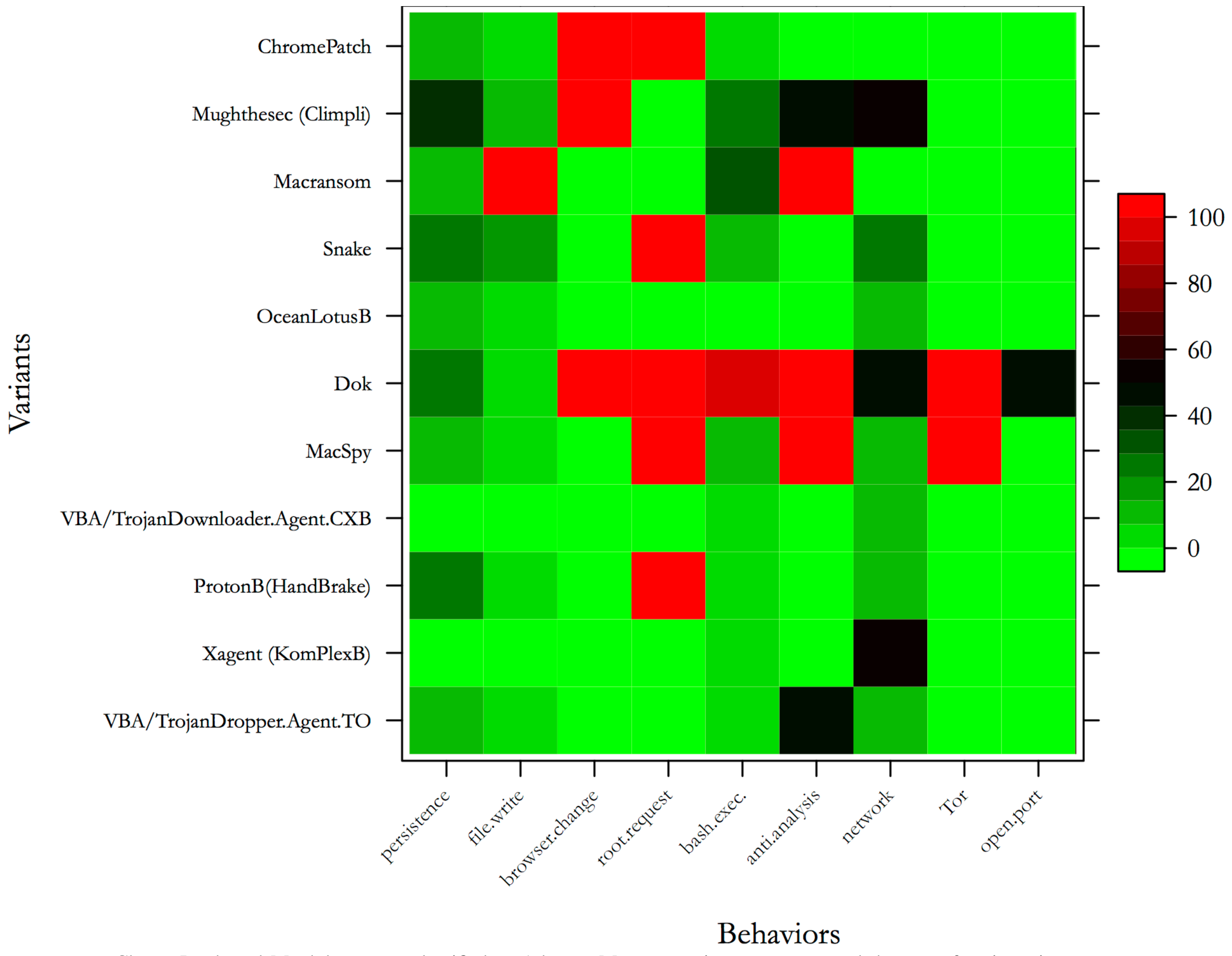
The academic paper of this work has been composed in a paper ”MAC-A-MAL: macOS Malware Analysis Framework Resistant to anti evasion techniques” which was submitted to the International Symposium on Engineering Secure Software and Systems ESSoS 2018.

B Appendix



FakeFileOpener and MacKeeper are classified as Adware, Keranger is ransomware and the rest of variants is trojan/backdoor.

Figure 3: Behavior heatmap of macOS malware in 2016



ChromePatch and Mughthesec are classified as Adware, Macransom is ransomware and the rest of variants is trojan/backdoor.

Figure 4: Behavior heatmap of macOS malware in 2017