Cuckoo's Malware Threat Scoring and Classification: Friend or Foe?

Aaron Walker
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, USA
awalker@unr.edu

Muhammad Faisal Amjad
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, USA
mamjad@unr.edu

Shamik Sengupta
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, USA
ssengupta@unr.edu

***Abstract*—Malware threat classification involves understanding the behavior of the malicious software and how it affects a victim host system. Classifying threats allows for measured response appropriate to the risk involved. Malware incident response depends on many automated tools for the classification of threat to help identify the appropriate reaction to a threat alert. Cuckoo Sandbox is one such tool which can be used for automated analysis of malware and one method of threat classification provided is a threat score. A security analyst might submit a suspicious file to Cuckoo for analysis to determine whether or not the file contains malware or performs potentially malicious behavior on a system. Cuckoo is capable of producing a report of this behavior and ranks the severity of the observed actions as a score from one to ten, with ten being the most severe. As such, a malware sample classified as an 8 would likely take priority over a sample classified as a 3. Unfortunately, this scoring classification can be misleading due to the underlying methodology of severity classification. In this paper we demonstrate why the current methodology of threat scoring is flawed and therefore we believe it can be improved with greater emphasis on analyzing the behavior of the malware. This allows for a threat classification rating which scales with the risk involved in the malware behavior.**

***Keywords*—Malware Detection, Malware Behavior Analysis, Dynamic Analysis, Sandbox, Threat Score, Threat Classification**

I. Introduction

Malware, i.e., malicious software, presents a threat to computing environments in the office, at home, and in travel. Malware artists develop software to exploit the flaws in any platform and application which suffers a vulnerability in its defenses, be it through unpatched known attack vectors or zero-day attacks for which there is no current solution. Malware which successfully exploits such a vulnerability produces an information security incident. Security incidents may involve the loss of functionality of a system, compromise of user account credentials, unauthorized access into a system, or any number of conditions which compromise the confidentiality, integrity, or availability of a system or application.

[1]This research is supported by the National Science Foundation (NSF), USA, Award #1528167.

978-1-7281-0554-3/19/$31.00 ©2019 IEEE

While all successful malware compromises require a measure of response in order to restore faith in the proper working order of a system, not all malware attacks are created equal. For large organizations, the number of live information security incidents can be staggering. Prioritization of incidents based upon levels of severity is necessary for the quick elimination of the most severe threats and the continued monitoring and assessment of threats not yet handled. This is especially true for organizations with a relatively small information security incident response team. In situations where high volumes of security incidents may be present, an automated means of prioritization is essential to help incident analysts to quickly triage and respond in an appropriate manner. The use of automated tools, especially those made available through open-source, adds value to the work performed by the security incident response team because in many cases it allows for greater understanding of information security threats, quicker remediation times, and provides more information for after-action analysis and reporting.

The Cuckoo Sandbox [1] provides a means of automated analysis of suspicious files. Through behavioral analysis, hash comparisons, and various integrated tools it is possible to identify malware and discover what indicators of compromise one should look for in their production environment to ascertain the presence of malicious activity on a network or system. It is important to note that as an automated malware analysis system, Cuckoo provides information about the behavior of a suspected file on a system but it is not using these observations to actively classify malware. As an option, Cuckoo can be configured to enable submission of a file to VirusTotal [8] for a comparison to known malicious files and the associated malware families as determined by various anti-virus vendors. Additionally, Cuckoo can be configured to allow custom signatures and/or those from the open source community which define severity scores for particular API calls and malware family attribution [1]. These optional elements are meant to enhance the Cuckoo-generated report on a sample file so as to bring more depth to the analysis. Without these optional features Cuckoo only has the ability to execute potential malware samples in a sandbox, observe the dynamic behavior, and report the actions committed. In this paper we are interested in the optional

element of malware signatures and what value they bring to the automated malware analysis.

Another reason to consider the Cuckoo Sandbox is that it is open source software. It is familiar to the information security community [2] and free to use, which makes it more available to incident response teams who cannot easily justify the expense of similar closed-source products. While the initial setup and configuration of Cuckoo is not particularly user-friendly, the end result is a system which allows for the submission of a file and the automated return of a report describing the observed actions of a suspected malware sample in a sandbox environment. The benefit of Cuckoo is in this automation – no human interaction is required for the behavioral analysis. A report is generated describing the actions which occurred when the suspicious file was opened or executed, and all an analyst needs to do is investigate the report. A detailed list of system operations, file manipulations, attempted communication with external systems, and even screenshots of the activity are generated. All of these artifacts are essential for understanding the nature of the malicious file analyzed.

One element of the Cuckoo report stands out as an element which immediately captures the eye of an analyst, and for good reason. The "Score" section appears at the top of the report and represents the threat severity of a malicious file as a number out of ten. The color of the section containing the score changes with severity, further suggesting the impact of a file as it goes from benign light green with a score of 0/10 to an angry red with a score of 10/10. This score does come with a notice from the developers: "Please notice: The scoring system is currently still in development and should be considered an alpha feature."

While using Cuckoo as an analysis tool in practice, we discovered odd behavior in the reported score for many malware samples. Several executable files were identified as malware with a score higher than the upper threshold of 10. This was confusing and caused us to wonder if there was an error in the scoring mechanism. Instead we discovered that there was no error in the configuration of our system or a bug in the Cuckoo software. Instead, we found that the methodology used by Cuckoo to generate this threat score results in an arbitrary value that is of little help to illustrate the threat severity of malicious code to an incident responder. The arbitrary nature of this score is not immediately apparent to the end user. The otherwise excellent value of the Cuckoo Sandbox in automated behavioral analysis might lead an incident responder to place a similar value in this score, perhaps so far as to reduce the priority of remediation for an incident involving a malware with a score of 2/10 in comparison to a similar incident involving a malware with score of 9/10 or 15/10. This could also lead an incident responder to prioritize an incident involving less actual risk, given the arbitrary nature of the threat score.

Contribution: Our contributions in this paper include the following:

- We provide an analysis of the malware threat scoring mechanism of the Cuckoo sandbox.
- We demonstrate that the current malware threat scoring in Cuckoo sandbox is arbitrary; we propose the need for a new solution.

The rest of the paper is organized as follows. Section II presents an overview of related work. Section III describes the malware behavior analysis methodology we employed including the hardware and software setup, malware dataset, and analysis of the scoring of API calls. Section IV discusses our evaluation of the Cuckoo malware threat scoring, problems identified, and possibilities for future work. Finally, Section V concludes the paper.

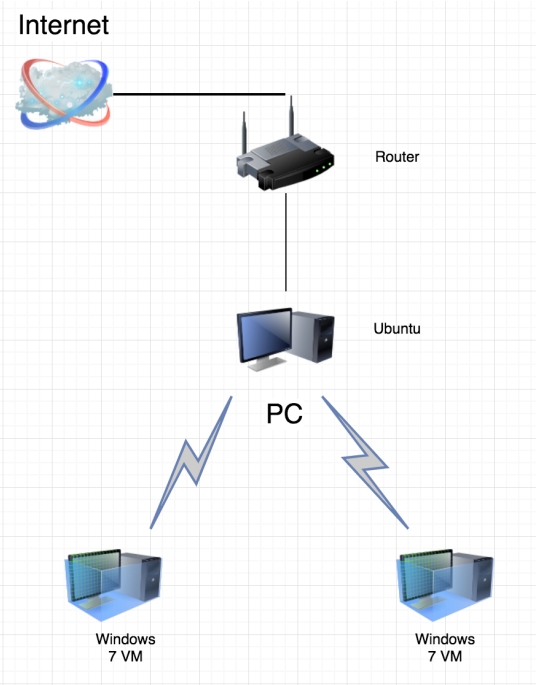

Fig. 1. Setup for Cuckoo Sandbox and the VM Environment.

II. Related Work

Researchers have evaluated the value of threat scoring in terms of privacy risk [3], cyber threat feeds [4], and Common Vulnerability Scoring System metric-based analysis (CVSS) [5], to name a few. This research shows the need for a robust scoring mechanism for the assessment of threats. This concept led us to consider the threat scoring system used by the Cuckoo Sandbox so as to determine what problems exist or improvements could be made to benefit the practice of information security incident response. Researchers have shown how the behavioral analysis reports from Cuckoo can be used to classify malware [7] in conjunction with labels for malware families extracted from VirusTotal [8] reports for each sample. This methodology relies on the presence of key features in the malware which correspond to a signature; however, malware authors can overcome signature-based analysis via mutation engine generating polymorphic code [10]. When malware is detected, another problem lies in automatic classification of malware family. To reduce the number of false positives, it is

shown that using a combination of static and dynamic analysis is more beneficial in identifying and classifying malware than using a single feature such as a signature [9]. Even machine learning techniques can be thwarted by a malware author and lead to misclassification through adversarial examples - this has led to analysis of malware through image analysis performed within generative adversarial networks [11].

What is clear from this research is that a combination of several techniques of malware analysis is most beneficial. Term Frequency-Inverse Document Frequency [12] metrics are used to effectively extract features of malware given access to logs of both malicious and non-malicious behavior and is robust to polymorphism [13]. Analysis of the semantics of code has been shown to detect malicious software independent from signature-based bytecode analysis [14]. The classification of particular API calls for potential malicious behavior can be used to analyze malware for particular traits [15]. We believe that the combination of several robust techniques to create a malware threat score will increase the confidence in such a metric.

III. Malware Behavior Analysis

In order to evaluate the current scoring mechanism Cuckoo uses to classify severity, we designed an environment to allow for the installation of Cuckoo and the analysis of known malware samples in a virtual machine sandbox per the installation instructions provided by Cuckoo [1]. Our goal was to create an environment which would perhaps be most typical for small to medium-sized security incident response teams and focus on the evaluation of potentially malicious software affecting Windows operating systems. This included the usage of a single Linux host machine to run the Cuckoo application and house Windows 7 virtual machine guest environments. Cuckoo allows for the configuration of Linux virtual machine guest environments as well; however, for this paper we were most interested in the analysis of Windows-based threats.

A. Setup

Cuckoo Sandbox version 2.0.6 was installed on a dedicated Ubuntu Linux host with access to the public internet, as shown in figure 1. Cuckoo was configured per the installation guide found on the Cuckoo website [1], including two 64-bit Windows 7 virtual machines installed on the Ubuntu host. Cuckoo recommends the usage of 64-bit Windows 7 over Windows XP for better results. Cuckoo supports many virtualization software solutions but does assume the usage of VirtualBox by default, so for ease of setup we chose this platform. VirtualBox is a free system virtualization product developed by Oracle and it easily integrates with Cuckoo for administration of the virtual machines.

The Windows 7 virtual machines were configured with essentially none of the built-in security measures in order to make them as vulnerable to malware as possible, including the disabling of User Access Control, Windows Firewall, and automatic updates. Software known to be the target of attack for malware was also installed on the virtual machines, including Adobe PDF reader, Java, and Microsoft Office. This ensures that when a malicious software sample is executed in the environment, the full breadth of the malware's behavior might be observed as it may attempt to access these applications as part of the process to compromise the system. Python has been installed to facilitate the communication between the Windows operating system on the virtual machine and Cuckoo on the host machine. Afterward, the Windows virtual machine resets to the base, non-compromised state via an automated Cuckoo command to VirtualBox. Furthermore, on the host Ubuntu system we ran the "cuckoo community" command to load the signatures provided by contributions from the Cuckoo user community. These signatures are curated by the Cuckoo development team and provide the definitions of malware severity and family attribution described later in this Section.

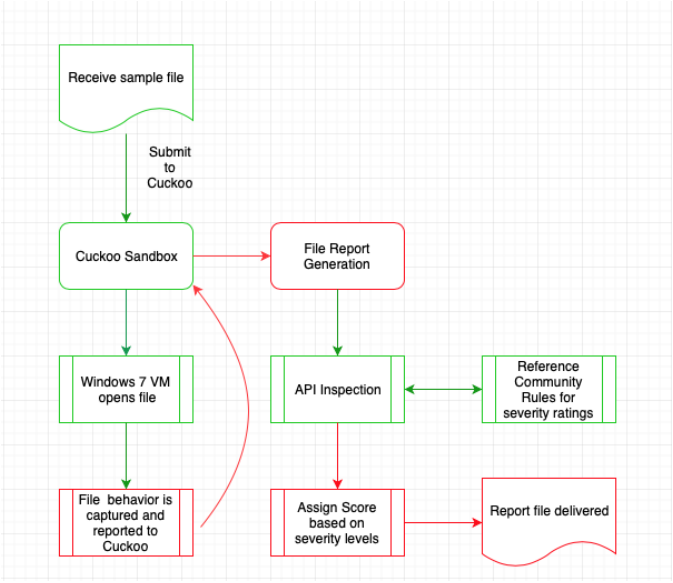

Fig. 2. Flow of the Cuckoo Sandbox malware analysis and report generation.

B. Malware Dataset

Known malware samples were acquired from Malpedia [6], a curated online resource of malicious software containing multiple versions of malware samples seen over time. This allows for the observation of evolving behaviors as the methods of exploiting system and application vulnerabilities changes with new generations of malware. Malpedia samples often include references to third party analysis of the malware as well as identified malware family and threat actor affiliation. This information is quite valuable for those desiring to create custom signatures in Cuckoo for malware family attribution. However in this paper we were mainly concerned with the native functionality of Cuckoo scoring using the community-provided signature set.

C. Methodology

Figure 2 describes the process of malware analysis in the Cuckoo Sandbox. When a malware sample is submitted to Cuckoo, it will designate a virtual machine for use in analyzing

the software. The virtual machine will resume from a snapshot from which it was in a known good, non-compromised state and Cuckoo passes the malware sample to the virtual machine for execution and analysis. Once the analysis has been performed, Cuckoo generates a report of the observed activity, including but not limited to changes to the registry, newly spawned processes, file creation and access, virtual memory access, HTTP communication to an external IP, and much more. These behavioral events are captured as a number of Windows API calls.


```
{
    "markcount": 1,
    "families": [],
    "description": "Queries for the computername",
    "severity": 1,
    "marks": [
        {
            "call": {
                "category": "misc",
                "status": 1,
                "stacktrace": [],
                "api": "GetComputerNameW",
                "return_value": 1,
                "arguments": {
                    "computer_name": "IBLIS"
                },
                "time": 1537025743.625125,
                "tid": 2740,
                "flags": {}
            },
            "pid": 2736,
            "type": "call",
            "cid": 330
        }
    ],
    "references": [],
    "name": "antivm_queries_computername"
},
```


Fig. 3. Sample of "GetComputerNameW" API call with a severity score of 1.

These API calls are each designated a severity score, which is determined by a repository of rules defined by the Cuckoo user community. According to the Cuckoo documentation [1], the range of severity scores is from 1-3, though our observation shows scores of 5 as well. While often a description will be included in the signature asserting the malicious nature of how a particular API call is being manipulated by malware, it is not immediately clear how the actual severity value is determined. All of the API calls flagged in these signatures perform common actions in a Windows system. This makes ranking a particular API call as more suspicious than another quite difficult. The community signatures clearly are attempting to note API calls which are used to have a potentially damaging effect, but the ranking from 1-3 or higher appears to be at the discretion of the author of the signature and subject to review by the Cuckoo developers.

Once all of the behavior witnessed by Cuckoo has been checked against these signatures and severity scores have been tallied, a final report is generated and available for inspection by an analyst via local web page or command line delivery. Below is an example of an API call with a severity score of 1. The activity shown here is the usage of the GetComputerNameW Windows function which retrieves the NetBIOS name of the local computer. The signature matched is identified as "antivm_queries_computername" and the function of this call is indeed to query for the name of the computer. In this example, our Windows 7 VM chosen by Cuckoo to analyze the malware sample was named "IBLIS." For the full set of sample malware we analyzed, we found that this particular API call was committed a total of 38,867 times.


```
{
    "markcount": 145,
    "families": [],
    "description": "Executed a process and injected code
       into it, probably while unpacking",
    "severity": 5,
    "marks": [
        {
            "call": {
                "category": "process",
                "status": 1,
                "stacktrace": [],
                "api": "CreateProcessInternalW",
                "return_value": 1,
                "arguments": {
                    "thread_identifier": 2740,
                    "thread_handle": "0x000001cc",
                    "process_identifier": 2736,
                    "current_directory": "",
                    "filepath": "C:\\Users\\bob\\AppData
                                \\Local\\Temp\\this.exe",
                    "track": 1,
                    "command_line": "",
                    "filepath_r": "C:\\Users\\bob\\AppData
                                \\Local\\Temp\\this.exe",
                    "stack_pivoted": 0,
                    "creation_flags": 4,
                    "process_handle": "0x000001d0",
                    "inherit_handles": 0
```


Fig. 4. Sample of "CreateProcessInternalW" API call with a severity score of 5.

Figure 5 presents an example of a CreateProcessInternalW API call which has been classified by the community rules with a severity score of 5. This particular call demonstrates the malware executing a process and injecting code into it. In this example, a process identified as 2736 is being loaded with the code within the file "this.exe" from within the logged in user's temporary files within an AppData subdirectory. It is interesting to note that the description of the behavior matched in the signature involves the possibility of code unpacking, which refers to a compressed executable file which must "unpack" in memory in order to execute. This procedure of unpacking is common in malware samples so it is likely that this is why the severity score for this signature was rated as a five. However, there is no clear indication that this is the case, which further leads to confusion regarding the assignment of this value. We discovered that this particular API was called 32,880 times across our sample set. The severity score of each identified signature-matched activity is evaluated and a final "Score" is produced in the report. Not all of the signatures single out specific API calls, as it has been seen that certain signatures refer to specific known malicious sites, known malicious malware file names, etc.

D. Discussion

Cuckoo reports were generated for 7,401 known malware samples retrieved from Malpedia. Analysis of these reports show that there were 138,523,300 API calls in total with just 264 unique API calls in all. Given that the Cuckoo analysis assigns severity scores to particular API calls as a result of a signature match, we found it interesting to compare

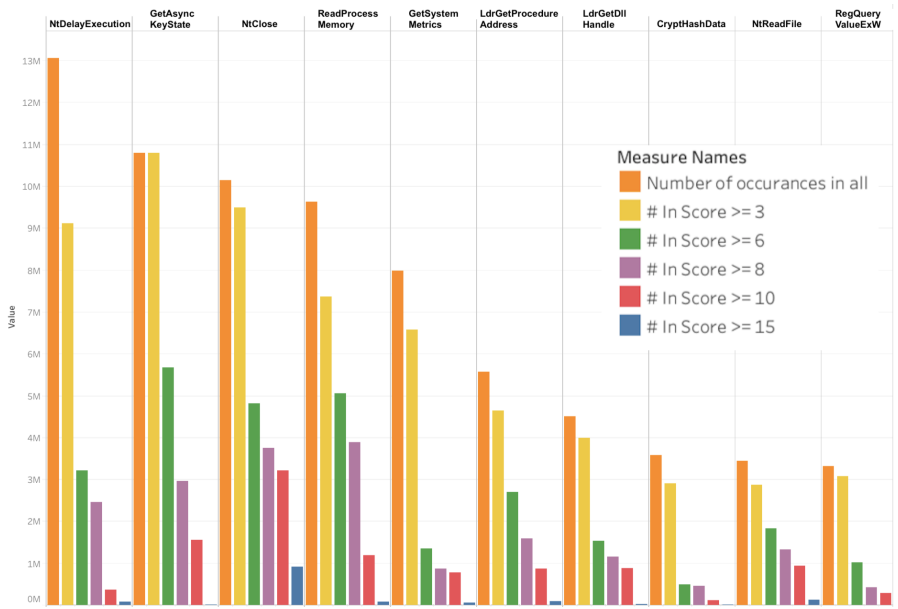

Fig. 5. Sample frequency of API calls by number of samples in Cuckoo report score range.

the frequency of API calls to the overall report threat score reported by Cuckoo. The severity ratings for these API calls are determined by Cuckoo community signatures, but not all of the APIs discovered in a Cuckoo analysis match these signatures. Thus we find that a high number of API calls will not necessarily translate to a high report threat score. As seen in Figure 6, our sample file named 891 had a reported 1,326,166 individual API calls (174 unique) and was assigned a threat score of 16.8/10 by Cuckoo. In contrast, our most threatening file with a score of 23.6/10 committed only 34,480 API calls (190 unique). What we discovered is that the majority of these individual function calls do not result in a severity score because there is no associated signature to assign the value. In fact, a single API function call may result in different severity rating assignments based upon different signature matched. For example, one malware sample was observed calling the "NtAllocateVirtualMemory" function with this behavior of possibly allowing code injection for another process and was assigned a severity score of 3. In the same malware sample this API function call was made again with the behavior matching a rule describing the possibility of the injection of code into an executed process while unpacking, resulting in a severity classification of 5. This creates a challenge when attempting to determine the relative threat of a particular API call observed in a malware sample. We believe a great deal of confusion regarding consistent threat classification exists given the disparity between severity scores for the same API calls amongst the signatures.

Table I describes this relationship for the top ten occurring API calls.

Figure 5 details the relationship between these highest occurring API function calls and the final threat score Cuckoo assigned to malware samples containing these calls. It is

TABLE I
TOP TEN OCCURRING API FUNCTION CALLS

Function	Type	Occurances
NtDelayExecution	Windows Kernel	13,052,984
GetAsyncKeyState	Windows Control	10,786,787
NtClose	File System	10,142,832
ReadProcessMemory	Process Control	9,626,589
GetSystemMetrics	Windows Conrol	7,974,563
LdrGetProcedureAddress	Windows Kernel	5,563,886
LdrGetDllHandle	Windows Kernel	4,497,628
CryptHashData	Security & Identity	3,568,810
NtReadFile	File System	3,425,244
RegQueryValueExW	Windows Registry	3,303,343

interesting to note the changes in frequencies of these API calls between the different threat values. For instance, "GetAsyncKeyState" is called a total of 10,786,459 times in the malware samples Cuckoo rated a threat score greater than or equal to three and just 1,543,317 for malware samples greater than or equal to ten. Of all of our tested malware samples, the average score was 2.6 with the majority of samples with a score in the 0 – 2.12 range. Of particular interest is that we found that 257 samples were rated above the suggested maximum "10" rating. These ranged from 10.2 – 23.6. This finding is of interest because nearly 3.5% of the samples analyzed were classified a threat rating above the maximum value. Why was this the case and what does this mean about the severity of these samples in comparison to those with a threat rating below 10? The answer to our question is that the arbitrary nature of the threat score denies the opportunity for the 0 – 10 rating to provide an evaluation of the threat of a given malware sample in relation to others. Essentially, what we can say about any suspected malware sample with a score of 0 is that no behaviors matched a signature rule. A score of 0.2, or 1/5, implies that exactly

one signature was matched for an observed behavior. Since any higher score is similarly only representative of the number of potentially malicious actions and not truly representative of the impact of the risks associated with those actions, we feel that this is an insufficient metric for assigning priority of malware remediation tasks for an incident response team.

IV. Evaluation

A. Results

Given that in this sample set we observed malware threat severity scores from 0 – 23.6, we decided to understand the methodology behind the value of the threat score. This required the analysis of the reports for each malware sample in our set which was evaluated by Cuckoo Sandbox. These reports are easily parseable as they are in JSON format. We found that the bulk of the Cuckoo report involves the observed behavior of the malware in terms of Windows API calls. Many of these API calls were designated a "severity" score within this report and it is here that we see what Cuckoo uses to determine the final report threat score value. When creating the report for a sample, captured behaviors are compared to any signature rules enabled as part of the Cuckoo configuration. The final threat score assigned by Cuckoo to the analyzed malware is the result of adding the scores of all the signatures which match an observed behavior of the malware and then dividing the sum by 5. Therefore, if S_o^n is the threat rating of a signature which matches the n-th observed behavior of the malware then the malware's final threat score S_f is given by:

$$S_f = \frac{\sum_{n=1}^{k} S_o^n}{5} \quad (1)$$

where k is number of observed malware behaviors that matched a cuckoo signature. For example, in the highest rated malware sample in our testing set Cuckoo observed six behaviors which matched signatures with a severity rating of 1, sixteen behaviors which matched signatures with a severity rating of 2, twenty-five behaviors which matched signatures with a score of 3, and one behavior which matched a signature with a severity rating of 5. The sum of these severity ratings is 118, which when divided by five results in the final threat score of 23.6 assigned by Cuckoo. We observed this formula again when analyzing a malware sample with 1,326,166 observed API function calls, with only 36 of these API calls resulting in a signature match – eight with a severity of 1, ten with a severity of 2, seventeen with a severity of 3, and finally one behavior matching a signature with a severity score of 5. The sum of these severity ratings is 84, which when divided by five results in the final threat score of 16.8 delivered in the Cuckoo report. The number five is the consistent denominator across all malware samples regardless of the number of APIs called or signatures matched. The developers of the Cuckoo Sandbox acknowledge the use of this number five [16] and state that this means of generating a final threat score is in need of improvement [17].

There is no obvious benefit in the use of the number five to divide the total severity score to achieve the final threat value for a malware sample. For all 7,401 malware samples, a sum was tallied for every severity score in a sample report and divided by five to achieve the same final threat score reported by Cuckoo. The consistent use of the number five across all analyzed malware sample suggests that this number is not related to any behavior or attribute of the analyzed malware samples but is instead an arbitrarily chosen value, similar to the severity values assigned to the Cuckoo community signatures. This adds to the confusion regarding the true value of the threat reported for a malware sample by Cuckoo.

What we found is that Cuckoo is designed to produce a final threat score which is the sum of the severity ratings for each observed API call, divided by five. What is not immediately obvious is what this means for the relationship between different malware samples analyzed in Cuckoo. The denominator value of five used in the calculation of the final threat score is not associated with the number of API function calls or other behaviors observed by Cuckoo in its behavioral analysis, nor is it linked to the number of matched signatures. Therefore, we cannot take this final threat score as an accurate measure to compare malware samples for the relative priority by which they should be addressed to mitigate the risk which they represent.

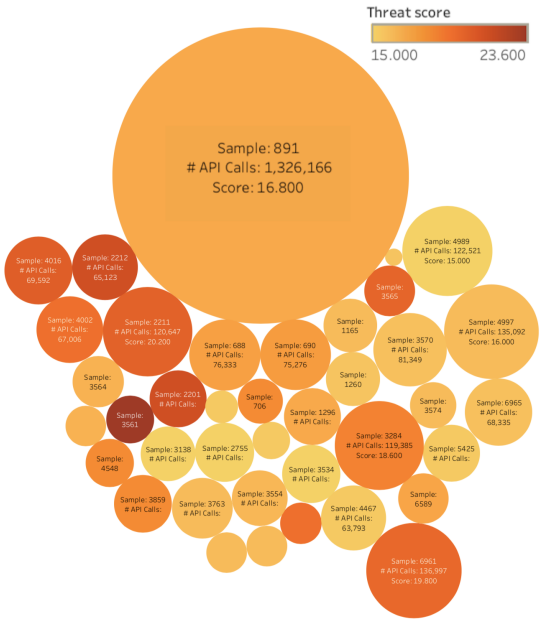

Fig. 6. Distribution of malware samples with score of 15 and greater, sized by the number of API calls.

B. Discussion and Recommendations

The arbitrary nature of the Cuckoo scoring methodology casts confusion upon the threat level of a given malware sample. For the incident responder, it might seem appropriate to immediately act to contain an incident involving a malware with a score of 18.8 instead of an incident involving a malware with a score of 6.4. However, if it is determined that the malware with the larger score is simply repeatedly attempting to reach out to a dead host on the internet (and of course

failing) while the malware with the lower score is actively injecting malicious code into legitimate Windows services, this casts confusion upon the meaning of the severity score. Should such a score be evaluated in conjunction with the behavior to determine actual severity? One could make a case for ignoring the scoring mechanism altogether and focus on the relevant indicators of compromise.

Yet the purpose of a threat score is to provide a quick, immediate index value to support effective triage. Even though the current methodology used in the Cuckoo Sandbox is arbitrary, one could argue that it was included to provide a general guide to help support a quick assessment. What is necessary is a more reliable metric. Currently the Cuckoo report threat score is dependent upon community-provided signatures. The severity score assigned to these signatures affects the final score Cuckoo evaluates, again via an arbitrary method. Of note is the fact that if the community signatures are not loaded into Cuckoo as described in Section III and no custom signatures are created by the user, each file analyzed by Cuckoo will have a threat score of zero due to not having any API severity values to evaluate.

This issue is not limited to the problem of scoring. Malware family classification is also dependent on these community signatures. Thus when Cuckoo analyzes a malware sample and reports that it belongs to a particular malware family, this is determined by the logic provided in a signature which checks for the presence of a particular mutex or specific filename in a particular Windows folder, for example. This is not a particularly robust method as malware behavior will change over time with new generations and variants.

We believe that in order to take the confusion out of the malware threat scoring, a more thorough analysis of malicious API calls is necessary. Statistical analysis of these function calls shows that a large amount of activity witnessed by Cuckoo remains unevaluated. Instead of relying on signatures, the breadth of the API calls can be evaluated in conjunction with the context of the behavior. For example, an API call made to reach an external host on the internet which is known to be a recent command and control server used by a large threat actor should carry a heavier severity score than a similar API call reaching out to Google. This requires both statistical analysis of each malware sample as well as reliable, current threat intelligence. In this manner, threat scoring and malware family attribution will be dynamic, robust, and provide greater confidence in threat prioritization for the information security incident response team.

V. Conclusion and Future Research

In this paper we have identified the arbitrary nature of the Cuckoo Sandbox malware Score value and the need for a more robust methodology to replace it. This would add value to the Cuckoo application as well as to any incident response methodology which makes use of malware analysis. A system of evaluating function calls in relation to successful actions performed on a system along with a comparison to CVSS scores and other threat intelligence sources will add a great deal more confidence in such a malware threat scoring system. Future research will consider the usage of Windows API calls in known malware samples in relation to known non-malicious software so as to develop a means of improved scoring and malware family identification.

References

[1] Cuckoo Foundation. 'Automated Malware Analysis - Cuckoo Sandbox', 2014. [Online]. Available: http://www.cuckoosandbox.org/. [Accessed: 10- Nov- 2018].

[2] Malwarebytes, Inc. 'Automating Malware Analysis with Cuckoo Sandbox', 2016. [Online]. Available: https://blog.malwarebytes.com/threat-analysis/2014/04/automating-malware-analysis-with-cuckoo-sandbox/. [Accessed: 10- Nov - 2018].

[3] Best, Daniel M., Jaspreet Bhatia, Elena S. Peterson, and Travis D. Breaux. 'Improved cyber threat indicator sharing by scoring privacy risk.' In Technologies for Homeland Security (HST), 2017 IEEE International Symposium on, pp. 1-5. IEEE, 2017.

[4] Meier, Roland, Cornelia Scherrer, David Gugelmann, Vincent Lenders, and Laurent Vanbever. 'FeedRank: A tamper-resistant method for the ranking of cyber threat intelligence feeds.' In 2018 10th International Conference on Cyber Conflict (CyCon), pp. 321-344. IEEE, 2018.

[5] Kebande, Victor R., Ivans Kigwana, H. S. Venter, Nickson M. Karie, and Ruth D. Wario. 'CVSS Metric-Based Analysis, Classification and Assessment of Computer Network Threats and Vulnerabilities.' In 2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), pp. 1-10. IEEE, 2018.

[6] Plohmann, D., M. Clauß, S. Enders, and E. Padilla. 'Malpedia: a collaborative effort to inventorize the malware landscape.' Proceedings of the Botconf, 2017.

[7] Hansen, Steven Strandlund, Thor Mark Tampus Larsen, Matija Stevanovic, and Jens Myrup Pedersen. 'An approach for detection and family classification of malware based on behavioral analysis.' In Computing, Networking and Communications (ICNC), 2016 International Conference on, pp. 1-5. IEEE, 2016.

[8] VirusTotal. 'Virustotal : Free online virus, malware and url scanner', 2004. [Online]. Available: https://www.virustotal.com/en/documentation/. [Accessed: 10- Nov- 2018].

[9] Ma, Xinjian, Qi Biao, Wu Yang, and Jianguo Jiang. 'Using multi-features to reduce false positive in malware classification.' In Information Technology, Networking, Electronic and Automation Control Conference, IEEE, pp. 361-365. IEEE, 2016.

[10] Selamat, Nur Syuhada, Fakariah Hani Mohd Ali, and Noor Ashitah Abu Othman. 'Polymorphic Malware Detection.' In IT Convergence and Security (ICITCS), 2016 6th International Conference on, pp. 1-5. IEEE, 2016.

[11] Kargaard, Joakim, Tom Drange, Ah-Lian Kor, Hissam Twafik, and Emlyn Butterfield. 'Defending IT systems against intelligent malware.' In 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), pp. 411-417. IEEE, 2018.

[12] Salton, Gerard, and Christopher Buckley. 'Term-weighting approaches in automatic text retrieval.' Information processing and management 24, no. 5 (1988): 513-523.

[13] Chen, Qian, and Robert A. Bridges. 'Automated Behavioral Analysis of Malware A Case Study of WannaCry Ransomware.' arXiv preprint arXiv:1709.08753 (2017).

[14] Lakhotia, Aran, and Paul Black. 'Mining malware secrets.' In Malicious and Unwanted Software (MALWARE), 2017 12th International Conference on, pp. 11-18. IEEE, 2017.

[15] Zhao, Chunlei, Wenbai Zheng, Liangyi Gong, Mengzhe Zhang, and Chundong Wang. 'Quick and Accurate Android Malware Detection Based on Sensitive APIs.' In 2018 IEEE International Conference on Smart Internet of Things (SmartIoT), pp. 143-148. IEEE, 2018.

[16] Cuckoo Foundation. 'Cuckoo GitHub Repository', 2017. [Online]. Available: https://github.com/cuckoosandbox/cuckoo/issues/2019#issuecomment-352305821. [Accessed 10- Nov- 2018].

[17] Cuckoo Foundation. 'Cuckoo GitHub Repository', 2016. [Online]. Available: https://github.com/cuckoosandbox/cuckoo/issues/732#issuecomment-174168657. [Accessed 10- Nov- 2018].