Case Study: Analysis and Mitigation of a Novel Sandbox-Evasion Technique

Ziya Alper Genç
SnT/University of Luxembourg
ziya.genc@uni.lu

Gabriele Lenzini
SnT/University of Luxembourg
gabriele.lenzini@uni.lu

Daniele Sgandurra
Royal Holloway, University of London
daniele.sgandurra@rhul.ac.uk

ABSTRACT

Malware is one of the most popular cyber-attack methods in the digital world. According to the independent test company AV-TEST, 350,000 new malware samples are created every day. To analyze all samples by hand to discover whether they are malware does not scale, so antivirus companies automate the process e.g., using sandboxes where samples can be run, observed, and classified. Malware authors are aware of this fact, and try to evade detection. In this paper we describe one of such evasion technique: unprecedented, we discovered it while analyzing a ransomware sample. Analyzed in a Cuckoo Sandbox, the sample was able to avoid triggering malware indicators, thus scoring significantly below the minimum severity level. Here, we discuss what strategy the sample follows to evade the analysis, proposing practical defense methods to nullify, in our turn, the sample's furtive strategy.

CCS CONCEPTS

• **Security and privacy** → **Malware and its mitigation**.

KEYWORDS

malware, evasion, stateless, detection, ransomware

1 INTRODUCTION

More than ever there is a need of automated systems to perform malware analysis as new malware samples are created at such a pace that security analysts are unable to manually analyse them. For instance, in 2018 more than 856,000,000 samples were found on-the-wild, with an average of 350,000 new malicious program released each day in the latest months of the year [1].

Analysing new samples of malware is a complex task. In fact, on one hand, security researchers are aimed at developing malware analysis systems, such as "sandboxes", i.e. confined and protected systems (typically based on virtualisation) which are used to execute suspected untrusted programs to analyse their behaviours. On the other hand, cyber-criminals creating malware and associated infrastructure want a return on their investments in their effort, and therefore are becoming more adept at developing threats that can evade increasingly sophisticated sandboxing environments [3]. In particular, in recent years, several ransomware families have started incorporating advanced evasion techniques [10]. For example, the first version of the WannaCry ransomware included a basic anti-sandboxing mechanism, based on DNS query, to bypass analysis [8]. Evasive techniques employed by malware are based upon the fact that the execution environment of a process may change slightly depending on whether it is run inside a native or a virtualised/sandboxed environment, e.g. due to the presence of memory and execution artifacts introduced by the sandboxing environment (environment-based evasion) or due to the different execution time of the process inside a sandbox (behavioural-based evasion) [9]. Malware may also specifically look for signs of emulation or virtualisation, e.g. specific drivers created inside a virtual machine, or the presence of specific applications, such as standard add-ons used by sandboxing solutions, or the lack of standard application and files (e.g., an empty virtual machine) to detect sandboxes. Finally, certain malware will check for user input before performing any action, and the lack of user interaction will sometime cause the malware to infer it is being run in a sandboxed environment. In all these cases, the goal of any evasive malware is to detect whether it is being run in a sandbox and, in such a case, stop performing any malicious action to avoid being analyzed further by sandboxing systems – therefore making it hard for analysts to extract the features and describe the behaviour of malware for future detection.

Other works have addressed the problem of the detection of evasive malware. For instance, [2] proposes a reliable and efficient approach to detect malware with split personality – behaving differently according to the running environment, in an attempt to evade analysis. Similarly, [7] describes BareCloud, an automated evasive malware detection system which is based on bare-metal dynamic malware analysis to compare the traces of analysis on different environments and compare them with the bare-metal one – a discrepancy meaning the malware is employing evasive techniques. Finally, [6] presents MalGene, an automated technique for extracting analysis evasion signature by leverages bioinformatics algorithms to locate evasive behavior in system call sequences.

In this paper, we describe a technique to bypass sandboxes that we have found being used by an active ransomware that we inspected. We also describe the process that has allowed us to observe the malicious activity in this specific sample. To the best of our knowledge, this malicious technique has never been described before: it is stateless, *i.e.,* the attack comprises multiple phases but the malware does not store any data on the target machine, and does not depend on any logical condition on the victim system to be triggered. The technique enables the ransomware to employ a stealthy attack strategy which would evade detection by sandboxes. After presenting our findings, we discuss the feasibility and impact of cyber attacks that use this technique, and propose possible mitigation strategies.

2 METHODOLOGY

The detection methodology we follow in this paper is composed of the following steps: firstly, we run a malware sample multiple times in a sandbox and collect the traces. Afterwards, we check if the sample has performed no suspicious activity in the first run, but has acted maliciously in the subsequent runs. If this is the case, it means the malware is empowered with some evasive functionalities. In the

following, we explain the details of (i) the steps we have followed when building the test environment, (ii) how we have gathered the data set and (iii) how we have conducted the experiments.

The analyses are performed using Cuckoo Sandbox, an open source automated malware analysis system [4]. Our test environment consists of 20 Virtual Machines (VMs) running atop Kernel-based Virtual Machine (KVM)[1], each has 2 CPU cores clocked at 2.60 GHz and 2GB RAM. On each VM, we performed a clean installation of Windows 7 32-bit Operating System (OS) with SP1. No guest agent is installed on VMs. Moreover, we performed additional steps to ensure the virtualization artifacts are removed to make the detection of the sandbox harder [5]. For instance, default BIOS[2] of VMs is replaced with a customized version which contains a real-world vendor name. Likewise, the hypervisor flag of the virtual CPUs is disabled to prevent guest OS from being aware of the virtualization. Next, we created user profiles on VMs and installed popular applications such as web browser (along with top-rated extensions), multimedia player, archive utility and office software. Furthermore, we artificially populated usage history in these profiles to reflect an authentic user. Finally, we took the snapshot of VMs and configured the Cuckoo accordingly.

Ransomware samples are obtained from the malware corpus provided by VirusTotal [13]. To filter cryptographic ransomware, *i.e.,* the programs that encrypts victim's data, we performed a query using ransomware-related keywords in the anti-virus scan results, such as *ransom*, *crypt*, and *lock*. After the search is complete, we had a set of 112 potential ransomware samples.

Once the data set is ready, we submitted the ransomware samples to Cuckoo Sandbox to study their behaviours. The label of VM used for each analysis task is noted to use in the second run. After all samples are analyzed, the samples that did not show any malicious activity are re-submitted selecting the same VM. Finally, we compared the reports generated by Cuckoo Sandbox to determine the samples that did not perform encryption in the first execution, but encrypted the victim's files in the second run.

3 RESULTS

Out of 112 malicious samples, one ransomware sample showed no malicious activity in the first execution, but encrypted user's files in the second run. SHA256 digest of the sample is `bcbc1aee86f5e1fd c2ba6fcb2e29933933b132a4c3d0f2eb0f73061702041243`. At the time of this writing (August 2019), 58 out of 66 anti-virus engines at VirusTotal identified this sample as a malicious program. Malware labeling tool AVclass [12] identified the sample as a `TeslaCrypt` variant.

3.1 Behavioral Analysis Reports

Now, we compare the behavioral analysis reports of two executions of the ransomware sample, generated by Cuckoo Sandbox.

3.1.1 First Execution. In the first execution of the sample, we observed no write operation on user files. Although no persistent change were made to the files, the sample

- called `GetComputerName` Application Programming Interface (API) to retrieve the NetBIOS name of host;
- called `GetVolumeInformation` API to collect various information about the virtual volumes and the hard disk;
- read the `InstallDate` key[3] in the registry which stores the installation date of the OS; and
- read `MachineGuid` key[4] from the registry, which is created during the installation of Windows OS.

Up to this point, the collected pieces of information would give the attacker the ability to identify the victim's computer with a high accuracy, *i.e.,* generate the fingerprint of the machine.

Immediately after taking the fingerprint of the environment, the ransomware sample established several network connections. Table 1 represents the IP addresses and the domain names that the sample connected using the HTTP protocol only. Using the threat intelligence services, we were able to confirm that the URL of each connection is related to a well-known, malicious activity pattern previously reported by anti-malware community.

Table 1: HTTP connections of the analyzed sample.

IP Address	Domain Name
`204.11.56.48`	`imagescroll.com`
`85.128.188.138`	`stacon.eu`
`109.73.238.245`	`surrogacyandadoption.com`
`69.89.31.77`	`biocarbon.com.ec`

Despite the suspicious activity summarized above, the sample did not perform any encryption, and shortly after the execution, it terminated itself, leaving no artifacts on the analysis environment. As a result, the malice score assigned to this sample by Cuckoo Sandbox is 2.4.

3.1.2 Second Execution. In the second run, the sample conducted the same reconnaissance steps taken in the first run. However, this time, after fingerprinting and connecting to remote addresses, and instead of ceasing, the sample

- silently deleted Volume Shadow Copy Service (VSS) copies (backup copies or snapshots of files or volumes, created by Windows OS) – this operation is typically encountered in ransomware attacks to prevent recovery of files;
- dropped an executable file – this is another typical malware action to bypass signature based detection;
- configured registry in order to automatically run itself at each login; and
- created encrypted files, rename them by appending the extension `.mp3`, and deleted the original files.

Cuckoo developers state that there is no upper limit for malice score, but currently the security threshold is set to 10. After the second run, the analyzed sample scored 25, and hence labeled by Cuckoo as "very suspicious".

[1] Kernel-based Virtual Machine, https://www.linux-kvm.org.
[2] SeaBIOS, https://seabios.org/SeaBIOS.

[3] `HKML\SOFTWARE\Microsoft\Windows NT\CurrentVersion\InstallDate`
[4] `HKML\\SOFTWARE\Microsoft\Cryptography\MachineGuid`

3.2 Reconstructing the Attack Scheme

The second behavioral analysis report displays a detailed picture of the attack. However, one piece of the puzzle is still missing: how does the ransomware sample decides to commence the attack?

Since the analyses are performed on freshly restored snapshot of VMs, the sample cannot store any state on the VM. Namely, all data generated during an analysis will be lost once the analysis ends. Furthermore, the ransomware is analysis-aware and collects various information that can be used to fingerprint the environment. Moreover, the sample connects to several remote machines before starting the attack, if it attacks.

Algorithm 1 Server-Cooperated Attack Strategy.

```
1: function ATTACK
2:     machineID ← GenerateFingerprint()
3:     SendToServer(machineID)
4:     response ← GetResponseFromCC()
5:     if response == CEASE then ExitProcess()
6:     else EncryptFiles()                    ▷ response is ATTACK
7:     return Success
```

A full disclosure of the sample's logic is possible only if we reverse-engineered the sample's executable. We did not, but we can formulate an hypothesis on the sample's possible functionality from its behavioral analysis. For the sake the goal of our discussion, this suffices. Surely, once infected the victim machine, the sample collects information likely to identify the running environment and generate a fingerprint of the machine. And it is almost certain that the information is sent to a Command and Control (C&C) server. Here, we speculate. The C&C may use the fingerprint to identify the victim machine and to decide whether the sample is executed for the first time on it. In this case, we keep on speculating, the C&C server returns a CEASE message to tell the ransomware to stop all operations and remain silent. On the contrary, when the victim machine is known, the C&C sends an ATTACK message to trigger the ransomware. If we are right, the attack occurs like as in Alg. 1 and its information flow as in Fig. 1.

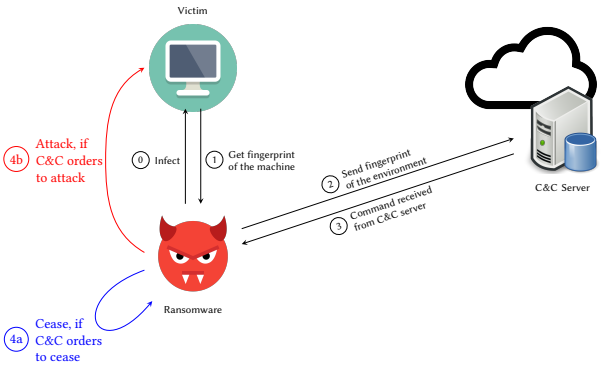

Figure 1: Reconstructed attack diagram.

4 DISCUSSION AND LIMITATIONS

In this section, we discuss the security impact of the described evasion technique and propose potential mitigation strategies.

Evading Sandbox Detection. The evasion technique analyzed in this paper allows the malware to stay under the radar, at least in the first run. This might allow the malware to bypass a defense system, for example, a system that whitelists applications that do not show malicious activities when run in sandbox environment. It should be noted that, though, the ability to evade sandbox detection comes with a price: malicious functionality of the malware gets activated on if the user runs the malware executable as many times as configured by the malware authors, *i.e.,* typically more than once, which is not guaranteed to happen.

Bypassing Malware Appliances. In [14], authors report that three popular malware appliances from well-known vendors analyze the binaries in an isolated environment, *i.e.,* run the executables in a machine disconnected from the network. In this setting, the malware would not be able to connect the C&C server and therefore would not receive an ATTACK command. Consequently, the binary would not exhibit any malicious behaviour. In this case, it is likely that the appliance is left with only signature based detection and the static analysis options which both can be evaded via obfuscation. In the end, the malicious binary would be not detected by these malware appliances. We note that, this result is not exclusive to the attack technique described here, rather, it is the limitation of the isolated analysis strategy followed by these appliances.

Mitigation Strategy. As a countermeasure to this attack, we propose the following strategy: to reveal the malicious behaviours, the analyzed samples should be executed multiple times in each analysis session. This way, the malware will run at the same environment for more than once, which will increase the chance of showing its real behaviour. Of course, the malware authors may set a higher threshold to prevent detection, but this would also delay the attack which is against the goals of cyber-criminals. Still, a high threshold would prevent detection by the sandbox. However, even if the detection fails after N execution, *i.e.,* the malware do not show a malicious activity, and the malware passes sandbox and reaches the actual user environment, the user –at least– would be secure for the first N execution. That said, our analysis assumes that the C&C uses a basic counter and a threshold to decide attack, but it may also utilize a smarter decision algorithm. For example, C&C might ignore subsequent fingerprint messages if the time frame between two executions are too narrow. Finding the best move of C&C in this game is therefore an open problem.

Defense in Depth. Alternatively, we propose a defense-in-depth strategy to be used in user environment as an auxiliary mitigation for this attack. To recall, the analyzed technique aims to bypass sandbox detection by acting benign in the first run. That is, if the each run of the malware executable occurs on an environment different from previous ones, the executable would not cause any damage. Using this idea, one can make the actual user environment look unique to unknown/untrusted executables for each run. This way, the executable –assuming that it is a malware– would report to its C&C server a unique fingerprint at each execution. Once received

a previously unseen fingerprint, C&C server would not find any match in its database, and send a `CEASE` message to the malware. It should be noted that realization of this defense method is feasible. For instance, it can be achieved by hooking certain APIs that can potentially be used to fingerprint the environment and randomize their return values. However, fingerprinting is a practice also used by software companies, *e.g.,* for activating proprietary programs. Therefore, benign applications should be whitelisted when applying this defense method to prevent undesired interference.

Network Dependency. The investigated technique involves communication between a C&C server, to receive ultimate decision to attack. This obviously requires the remote server to be available when the malware program is executed each time. If the C&C server becomes unreachable, there would be no location to keep the state of the attack, and the strategy cannot work: the malware would not perform its nefarious actions. At first, this might look like a shortcoming of the technique, as blocking malicious IP addresses is an efficient and effective practice to distort malware communications. However, malware authors has been encountering this limitation for a long time, and they already employ workarounds. For example, as analyzed in [11], authors of Cerber ransomware uses messages encoded in Bitcoin transactions to coordinate the C&C servers. Being decentralized, impeding communication with blockchain network is considered difficult. Malware authors can enhance the examined evasion technique by integrating the blockchain-based coordination of C&C servers. We therefore argue that the said limitation do not decrease the significance of the threat.

5 CONCLUSION

The arms-race between malware authors and anti-malware developers has been occurring in the digital world since its early days. In the last two decades, however, malware developers are living their golden age. Powerful techniques such as metamorphic obfuscation have largely limited the effectiveness of signature-based defense systems. Consequently, protection systems started to move toward dynamic analysis to detect freshly generated malware. Sandboxes, isolated execution and monitoring environments, have been widely used for this purpose. Consequently, the sandbox environments have been among the top items of the target list of cyber-criminals, and numerous advanced evasion techniques have been seen in real-world malware attacks.

In this paper, we described a not-yet-discussed evasion technique that we found being followed by a still active `TeslaCrypt` ransomware sample. Using it, the sample manages to look benign and therefore is capable to avoid being classified as a malware, when analyzed in Cuckoo sandbox. We suggested two improvements, one tightening current behavioral analysis methods, the other working to mitigate the severity of an attack by that sample. Namely, we discussed (i) a smart execution strategy to increase the detection chance of the malware in the sandbox; and (ii) a complementary defense method to be employed in the actual user environment, preferably integrated into already existing protection systems. We have also compared pros and cons of the two defense approaches and their potential side effects to the user, but an experimental evaluation of the effectiveness and usability of these methods is left as a future work.

ACKNOWLEDGMENTS

This work was partially funded by European Union's Horizon 2020 research and innovation programme under grant agreement No 779391 (FutureTPM) and by Luxembourg National Research Fund (FNR) under the project PoC18/13234766-NoCry PoC.

REFERENCES

[1] AV-TEST. 2019. Malware Statistics & Trends Report. https://www.av-test.org/en/statistics/malware/.
[2] Davide Balzarotti, Marco Cova, Christoph Karlberger, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. 2010. Efficient detection of split personalities in malware. In *17th Annual Network and Distributed System Security Symposium (NDSS 2010)*. The Internet Society, San Diego, US, 16.
[3] Cisco. 2018. Cisco 2018 Annual Cybersecurity Report. https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf.
[4] Cuckoo Sandbox. 2019. Cuckoo Sandbox – Automated Malware Analysis. https://cuckoosandbox.org/.
[5] Olivier Ferrand. 2015. How to detect the cuckoo sandbox and to strengthen it? *Journal of Computer Virology and Hacking Techniques* 11, 1 (2015), 51–58.
[6] Dhilung Kirat and Giovanni Vigna. 2015. MalGene: Automatic Extraction of Malware Analysis Evasion Signature. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. ACM, New York, NY, USA, 769–780.
[7] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. 2014. BareCloud: Bare-metal Analysis-based Evasive Malware Detection. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 287–301. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/kirat
[8] MalwareTech. 2017. How to Accidentally Stop a Global Cyber Attacks. https://www.malwaretech.com/2017/05/how-to-accidentally-stop-a-global-cyber-attacks.html.
[9] Jonathan A. P. Marpaung, Mangal Sain, and Hoon-Jae Lee. 2012. Survey on malware evasion techniques: State of the art and challenges. In *14th International Conference on Advanced Communication Technology (ICACT)*. IEEE, Piscataway, New Jersey, US, 744–749.
[10] Minerva Labs. 2017. 2017 – The Year Malware Went More Evasive. https://l.minerva-labs.com/hubfs/Minerva%20Infographic_Final.pdf.
[11] Stijn Pletinckx, Cyril Trap, and Christian Doerr. 2018. Malware Coordination using the Blockchain: An Analysis of the Cerber Ransomware. In *2018 IEEE Conference on Communications and Network Security, CNS 2018, Beijing, China, May 30 - June 1, 2018*. IEEE, Piscataway, New Jersey, US, 1–9.
[12] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. 2016. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer International Publishing, Cham, 230–253.
[13] VirusTotal. 2019. VirusTotal Threat Intelligence. https://www.virustotal.com/.
[14] Akira Yokoyama, Kou Ishii, Rui Tanabe, Yinmin Papa, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, Daisuke Inoue, Michael Brengel, Michael Backes, and Christian Rossow. 2016. SandPrint: Fingerprinting Malware Sandboxes to Provide Intelligence for Sandbox Evasion. In *Research in Attacks, Intrusions, and Defenses*. Springer International Publishing, Cham, 165–187.