Survivalism: Systematic Analysis of Windows Malware Living-Off-The-Land

Frederick Barr-Smith
Oxford University

Xabier Ugarte-Pedrero
Cisco Systems

Mariano Graziano
Cisco Systems

Riccardo Spolaor
Oxford University

Ivan Martinovic
Oxford University

2021 IEEE Symposium on Security and Privacy (SP) | 978-1-7281-8934-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/SP40001.2021.00047

***Abstract*—As malware detection algorithms and methods become more sophisticated, malware authors adopt equally sophisticated evasion mechanisms to defeat them. Anecdotal evidence claims Living-Off-The-Land (LotL) techniques are one of the major evasion techniques used in many malware attacks. These techniques leverage binaries already present in the system to conduct malicious actions. We present the first large-scale systematic investigation of the use of these techniques by malware on Windows systems.**

In this paper, we analyse how common the use of these native system binaries is across several malware datasets, containing a total of 31,805,549 samples. We identify an average 9.41% prevalence. Our results show that the use of LotL techniques is prolific, particularly in Advanced Persistent Threat (APT) malware samples where the prevalence is 26.26%, over twice that of commodity malware.

To illustrate the evasive potential of LotL techniques, we test the usage of LotL techniques against several fully patched Windows systems in a local sandboxed environment and show that there is a generalised detection gap in 10 of the most popular anti-virus products.

I. Introduction

Malware development and detection is a cat and mouse game, in which malware authors are continuously developing new techniques to bypass detection systems. Security products such as anti-virus (AV) implement static and heuristic analysis technologies to detect, classify and prevent malware from effective execution [5]. In the past, many solutions heavily relied on signature-based detection, but unfortunately these approaches have become less effective due to the use of polymorphism and packers. In turn, many products started developing heuristic analysis solutions, consisting of algorithms that allow them to detect malicious behaviour. These algorithms have become a crucial component for AV engines. Over time, these algorithms have increased in sophistication and thus require more innovative evasion techniques.

Malware authors and red teams often research and discover new methods to bypass security solutions. While their underlying goals may be different in nature, both types of attackers typically leverage state-of-the-art evasion techniques to accomplish their goals. From a defender's point of view, it is crucial to understand these attacks and study their trends in order to be able to react in a timely manner. One evasive tactic that has become popular among both red teams and malware authors is the usage of Living-Off-The-Land (LotL) techniques.

LotL techniques refer to the use of binaries that are already present on systems or are easy to install (e.g., signed, legitimate administration tools) to conduct post-exploitation activity. By leveraging these tools, an attacker can achieve registry modification, gain persistence, conduct network or system reconnaissance or perform a proxied execution of other malicious code. They can even be used to reduce the event logs generated by malicious activity without needing additional files to be downloaded onto the system.

Far from being obscure techniques, many of them are publicly documented on the internet [32]. Many open source offensive security tools leverage LotL techniques and are frequently used by the entire spectrum of actors, from legitimate red teams to amateur cybercriminals, organised crime groups or nation state actors. PoshSpy [15], a Russian state-sponsored APT29 attack module, was one of the first detected APT uses of LotL techniques, specifically *Powershell* and *Windows Management Instrumentation*. Iranian threat groups [1], APT33, APT34 and others are also well known for using native Windows binaries and other signed tools, particularly *Powershell* [8]. Table I lists the LotL techniques employed by several state-sponsored actors.

Despite *Living-Off-The-Land* being a relatively well-known term in the information security community, it is sometimes hard to find a precise definition for it. Moreover, to the best of our knowledge no research contains a systematic study of LotL techniques' prevalence in malware samples. Much of the documentation available regarding LotL techniques is in the form of blog posts that document anecdotal findings on certain malware families, or technical descriptions of the techniques used by malicious actors once they obtain remote access to a compromised system.

For instance, Emotet and Trickbot, two of the most common Remote Access Trojans (RAT) at the time of writing, reportedly use chained LotL binaries for stealthiness [58]. As a countermeasure, Microsoft described steps taken to combat LotL techniques used by commodity RATs [37]. The highly evasive RAT Astaroth [43], some of TA505's malware arsenal, Dexphot cryptominer [38] and Nodersok all use multiple LotL binaries simultaneously [42].

In this paper, we analyse the LotL phenomenon as it pertains to LotL binary usage by commodity malware binaries. Our first step is to describe what a LotL binary is and how

DOI 10.1109/SP40001.2021.00047

APT Group	LotL Binaries Used	Purpose Of Execution
APT3	Powershell, Rundll32, Schtasks	Credential Theft, Persistence & Proxied Execution
APT10	Certutil, BitsAdmin, Net, Wmic, PsExec	Data Exfiltration & Lateral Movement
APT29	Powershell, Schtasks, Wmic	Data Exfiltration, Lateral Movement & Persistence
APT33	Powershell, ProcDump, Schtasks, Vbscript	Credential Theft, Data Exfiltration & Lateral Movement
APT34	Certutil, Mshta, Schtasks, Powershell	C+C, Data Exfiltration, Persistence & Proxied Execution
Astaroth	BitsAdmin, Certutil, Regsvr32, Userinit	AV Evasion, C+C, Credential Theft & Proxied Execution
Dexphot	MsiExec, Rundll32, Nslookup, Schtasks	Persistence, Proxied Execution
Gallmaker	Powershell, Winword	C+C, Data Exfiltration & Proxied Execution
Havex	BitsAdmin, Powershell, PsExec	Credential Theft & Lateral Movement
Nodersok	Mshta, Node, Powershell	AV Evasion, Command and Control & Proxied Execution
SoftCell	At, Net, PsExec, Reg, Wmic	Credential Theft, Data Exfiltration, Lateral Movement & Recon
TA505	Msiexec, Net, Rundll32, Powershell	C+C, Data Exfiltration, Proxied Execution & Recon
Turla	Powershell, PsExec, Wmic, Wscript	C+C, Data Exfiltration. & Proxied Execution

TABLE I: APT Groups Using LotL Techniques.

it can be leveraged by malicious software to conduct its nefarious actions. We focus our research on Windows as the dominant operating system in terms of popularity and the most frequently targeted by malware [63]. Many LotL based AV evasions have been documented. As a consequence, the security community has largely assumed that LotL techniques such as proxied execution of malware are in fact effective against security solutions. As a first step, we question this assumption and raise our first research question:

- *Can LotL techniques effectively evade commercial AV?*

To answer this question, we evaluate a representative set of security products and show how some of these techniques, although well-known for attackers and defenders, are still a valid method to bypass security solutions and therefore are an open challenge for the security industry. In fact, LotL binaries are quite often used by system administrators and advanced computer users to perform system administration tasks, making it extraordinarily difficult to distinguish between legitimate and malicious behavior even for a trained analyst. We responsibly disclosed our findings to the affected AV vendors and followed up with many of them as they improved their detection capabilities.

Although existing documentation provides sound evidence of the usage of these techniques, it is still unclear how prevalent the phenomenon is among malware samples. In this way, we raise our second research question:

- *How prevalent is the use of LotL binaries in malware?*

Building on this, we try to shed some light on some of the trends in the current threat landscape, to identify:

- *What purposes do malware binaries use LotL techniques for?*
- *Which malware families and types use LotL binaries most prolifically and how does their usage differ?*

We also investigate the reasons why these techniques are difficult to detect. Some of the AV firms that engaged with our responsible disclosure communications highlighted the difficulty of separating malicious attacks from totally legitimate administration tasks conducted by system administrators. This brings us to another question:

- *What are the overlaps and differences in the behavior of legitimate and malicious binaries with respect to the usage of LotL binaries? How would this affect detection by heuristic AV engines?*

While there are some clear differences in LotL binary usage prevalence between malicious and benign samples, we also noticed certain similarities for categories such as proxied execution.

Finally, we focus our attention on highly evasive and Advanced Persistent Threat (APT) malware to find out that it leverages these techniques twice as much as commodity malware. State-sponsored attack groups, that researchers attribute to China (APT3/APT10) [48], Iran (APT33/APT34) [49] and Russia (APT28/APT29) [50], amongst others, are examples that motivate this study. We list some of these APT groups that conduct attacks using LotL techniques in Table I.

Contributions. To the best of our knowledge, this paper presents the largest scale systematic analysis of the use of LotL techniques by commodity and APT malware to date. We make the following core contributions:

- We assess the evasiveness of LotL techniques by testing a representative set of most popular AV engines against malicious payloads deployed via LotL techniques, showing how the detection complexity of LotL represents a challenge for the industry. Even nine months after disclosure, these techniques are still not successfully detected.
- We conduct a large-scale measurement study across several datasets that are representative of modern commodity malware. We determine the prevalence of this technique and the differences among different malware families and types. We also assess the impact that LotL techniques might have in the industry due to false positive risk.

- We evaluate a dataset of APT malware, which we make public to facilitate further research and ascertain that it executes LotL techniques twice as frequently as commodity malware. We also identify which APT groups use LotL techniques the most.

II. Background & Related Work

In this section, we first define LotL binaries and enumerate some of the purposes for which these binaries are used by malware. Then, we provide an overview of related papers and research on this topic.

A. LotL Binaries

Due to its novelty, there is significant confusion regarding the term *Living-Off-The-Land binary*, or its shorter version *LOLbin*. In recent years, this term has become a common word used to refer to a wide range of binaries used in cyberattacks. Researchers tend to put many different tools under the umbrella of *LOLbins*.

The term 'Living-Off-The-Land' has historically been used to represent the concept of feeding (or living off) what the land can provide, either by farming or hunting. Transposed to malware and intrusion, an attacker might leverage what is already available to them (either already present on the system, or easily installable) to conduct a successful attack and avoid detection.

Some have provided rather restrictive definitions for the term LOLbin. This could be subject to debate, and there is no consensus within the community around this term. For instance, the well-known website LOLBAS project [46] lists a wide range of LotL binaries. Within this list, we are able to ascertain that many of the tools are administrative commands used by system administrators in their daily operation.

In this paper, we define a LotL binary as any binary with a recognised legitimate use, that is leveraged during an attack to directly perform a malicious action; or to assist indirectly, in a sequence of actions that have a final malicious outcome.

Binaries installed by default on Windows systems like `Reg.exe`, `Sc.exe` and `Wmic.exe` are the most frequently executed by malware. Most binaries installed by default are signed by Microsoft Authenticode [35]. An Authenticode signature proves that a binary has not been tampered or modified from its compilation. This may have a high impact in the confidence that security tools have on it; as discussed by Kim et al. [27], these binaries may even be explicitly whitelisted. Malware making use of trusted LotL binaries could thereby be more capable of evading AVs. The use of system binaries on Windows systems can be incorporated as part of the operation of malware, and more importantly, many LotL techniques make use of system binaries for the purpose for which these binaries were intended.

External signed binaries can also be used, such as `PsExec.exe` or other SysInternals binaries. We also included these in our analysis, but they are less commonly used by malware. Examples of nation state malware using external signed binaries, amongst many, are SoftCell [30] and Havex [13]. These campaigns both used `PsExec.exe` to stealthily run remote commands for the purpose of lateral movement within compromised networks. In some rarer cases, vulnerable (but signed) drivers have been used to escalate privileges on the system. This is a technique used by the RobbinHood ransomware [7] and various APT wiper malware samples targeting Saudi Arabian systems including Dustman [51], Shamoon and Zerocleare [25].

Traceability. Certain LotL binaries may leave more system logs than others, that can be leveraged by security tools or forensic analysts in order to detect a malicious action. For instance, Powershell can be configured to have comprehensive logging. Microsoft also recommend a number of these native binaries to be blocked from executing on systems, unless there is good reason to do otherwise [39].

B. Scope of our Study

In this paper, we focus on Windows malware that executes system binaries for various purposes. These purposes typically involve progression along the intrusion kill chain [24] or avoiding AV detection. All of these techniques are deployed within the user space of the system.

Process hollowing and injection are also not within the scope of our analysis, despite being common techniques deployed by fileless malware. We do not include these process manipulation techniques as they are not LotL techniques according to our earlier definition.

C. Related Work

LotL malware and its occasional aliases, 'advanced volatile threat' or 'fileless' malware are mentioned sparsely in the current academic literature. This is mostly limited to introductory analyses or described as an emerging highly evasive malware variant. Li et al. [31] present an analysis of malicious Powershell scripts that has a subsection specifically describing LotL attacks and fileless attacks as *the trend of cyber attacks in recent years*. A recent paper by Wang et al. [72] on the subject of data provenance analysis identifies Living-Off-The-Land as an emerging prominent evasive malware subtype. Prior work includes introductory analyses [64], however LotL malware has not yet been subject to robust academic analysis. Symantec [73, 66] and Cisco Talos' [65] white papers introduce the subject with an analysis of prevalence across a number of datasets. Currently, no papers conduct a systematic analysis of the use of LotL techniques by Windows malware at scale, comprising multiple datasets.

LotL techniques are mentioned in some papers, emphasising stealthiness and their use by APT malware. Pendergrass et al. [56] state LotL is a malware subtype that is *often undetected*. These descriptions provide evidence of the need to

quantify the evasive capabilities of these techniques against specific AV engines.

In a paper on the malware analysis tool Yara, Cohen [9] describes LotL as a *trend that has been recently observed in the tactics used by elite threat actors*; this claim is reinforced by the results of our analysis. Research by Hassan et al. [21] states that APT malware uses *LotL attack strategies* to enable persistent campaigns and analyses two campaigns, compared to our less granular analysis of 16,232 samples. Their work also leverages MITRE's ATT&CK [45] by which MITRE defines an excellent taxonomy that describes and classifies most well-known attacks. Numerous LotL techniques are indexed within the MITRE ATT&CK framework. MITRE corporation and their Common Vulnerabilities and Exposures (CVE) are an established authority within the security field. Their inclusion and description of many LotL techniques are an indication that this is a subject worthy of further analysis.

Orthogonal to our research is the analysis and deobfuscation of script-based malware. Malware making use of LotL techniques often uses malicious scripts as payloads. Mitigations against malicious scripts invoked by the `Powershell.exe` binary have been tested by the identification of suspicious behavioural patterns by Ugarte et al. [67]. Other papers such as Rubin et al. [61] apply machine learning to the detection of Powershell malware. Curtsinger et al. [11] propose detection mechanisms for malicious Javascript. While these papers propose effective detection methods for narrow subsets of malicious payloads delivered, they do not analyse the wider malware ecosystem and how these payloads are triggered by LotL binaries.

III. Motivation: Anti-virus products vs. Living-off-the-Land Techniques

Security researchers have documented many cases where LotL techniques can be used to successfully evade security products [41]. In many of these cases, these LotL binaries are leveraged to proxy the execution of a malicious payload, having it execute under the context of a legitimate looking process, or spawning a new process as a child of a legitimate system process. In some cases, these payloads are executed as a side effect of the LotL binary invocation, while in others it is just the result of its main documented behaviour. Many AV products reportedly fail to properly detect these techniques.

To answer our first research question, we first analysed whether current AV products flag LotL techniques as indicators of malicious behavior. To this end, we first selected a representative set of 10 end-user AV products and simulated attacks by leveraging a reverse shell executed by common well-known LotL based proxied execution techniques. The AV engines we installed on each host were taken from an industry compiled list of the ten most popular AV engines [54], all of which advertise behavioural analysis capabilities. Moreover, the intention of this study is not to test the detection capabilities of any particular AV product or to compare them against each other, but to determine if a generalised detection gap exists. Although we endeavoured to configure every product to apply its full detection capabilities, these results shall not be used for comparison of AVs as behavior may vary depending on the environment, targeted end users, and configuration settings.

AV	Ftp.exe	Mshta.exe	Wmic.exe	Rundll32.exe	Regsvr32.exe	Bitsadmin.exe
Avast Premium Security						
Bitdefender Internet Security						
Cylance Smart AV						
Eset Internet Security						
Kaspersky AV				✓	✓	
Malwarebytes for Windows Premium						
McAfee Total Protection						
Sophos Home Premium						
Webroot SecureAnywhere AV						
Windows Defender / AMSI					✓	✓

TABLE II: AV detections for our initial experiments on Windows 10.

We performed these experiments within networked Windows 10 virtual machines, with up-to-date local AV products connected to their cloud components. We omit the precise setup details here, but describe them in Appendix C.

We leveraged a reverse shell to assess how vulnerable AV systems are to evasion by malware deploying LotL techniques. We deem a reverse shell, capable of allowing remote execution of commands, to be a sufficient proof of successful code execution. A remote network connection executing commands is identical to many Remote Access Trojans (RAT)'s functionality. We conducted our experiments by running this reverse shell from different LotL binaries to test whether AV products detect these techniques as malicious. We obfuscated reverse-shell payloads when necessary and used various payload types to test AV's capability to detect the delivery mechanism itself, not the specific payload being delivered through static signatures. The LotL binaries that we leveraged for testing and the specific techniques by which we achieved evasion are described in detail in Appendix D.

Table II shows the results for this experiment. We can observe that most popular AV engines allowed us to establish a reverse shell and execute commands, using various LotL techniques that are common in commodity and APT malware.

Responsible Disclosure and Response. We issued a document to all AV vendors concerned, containing in-depth technical results of our evasion tests with above 90 days notice. To assist with remediation, the documents we provided included

AV	Ftp.exe	Mshta.exe	Wmic.exe	Rundll32.exe	Regsvr32.exe	Bitsadmin.exe
Avast Premium Security					✓	
Bitdefender Internet Security					✓	
Cylance Smart AV						
Eset Internet Security	✓			✓	✓	
Kaspersky AV	✓	✓	✓	✓	✓	✓
Malwarebytes for Windows Premium						
McAfee Total Protection						✓
Sophos Home Premium	✓	✓	✓	✓	✓	✓
Webroot SecureAnywhere AV			✓			
Windows Defender / AMSI	✓	✓	✓	✓	✓	✓

TABLE III: AV detections for our repeated experiments (9 months after initial testing).

evidence of how each evasion was conducted. Some vendors patched their detection algorithms and actively engaged with our research; others denied or ignored the responsible disclosure process. Moreover, some AV vendors reported difficulties with detecting some of the documented cases as detection rules are prone to false positives.

Experiment Repetition. We repeated our tests on Windows 10 machines, nine months after our initial evasion testing. This allowed us to test if AV vendors had included new heuristic rules in their products to detect LotL binary usage. The detection capabilities of AVs in this second experiment is shown in Table III. We can observe that a number of vendors updated their detection capability. 25 out of 60 of the exact same payloads were detected.

In some of the tests where our reverse shell was detected, we modified the payload (applied a different obfuscation method or ran a different payload) while maintaining the exact same command line arguments for the LotL binaries, in an attempt to establish whether the detection was implemented to match the delivery mechanism or the payload. By leveraging these obfuscated and modified payloads, we successfully executed a reverse shell in 19 of these 25 blocked instances.

These results show that LotL techniques are still a significant challenge for AV vendors. These tools are commonly leveraged by legitimate users in unpredictable ways, and security companies struggle to deploy effective detection strategies without false positives. The following sections will show how these techniques are a prevalent phenomenon in commodity malware, and how this problem should not be overlooked by the security community.

IV. Measuring LotL prevalence

In this section we measure the prevalence of LotL techniques in malware and try to answer the research questions raised. This measurement effort was conducted over 9 separate sub-datasets. In total, we collected 31,805,549 samples from which we obtained 16,048,202 behavioural reports from VirusTotal (VT).

A. Dataset Composition

To be as comprehensive as possible, we obtained public and private datasets from different sources. We chose varied datasets to represent possible malware distributions. We included public malware datasets leveraged by the research community throughout the years to ensure completeness and reproducibility. To complement these data sources, we created three additional datasets following three different collection strategies. These datasets are listed in Table IV.

Public Datasets. We included several public and well-known malware datasets to enable reproducibility, conducting pre-processing where necessary to isolate PE binaries. We included these larger scale datasets to represent commodity malware used in previous research efforts [2, 55]. We included a collection from the VirusShare [69] corpus of hashes of unpacked binaries, in addition to hashes identified as PE hashes by existing pre-processing of files. We included the Ember dataset [3] of Windows malware PE files, released for training machine learning models. We also leveraged a feed of malware executables, collected and distributed by Georgia Tech [26]. Finally, we also included samples from VX-Underground [70] and Malshare [33], both well known public datasets.

VirusTotal Balanced Dataset. We gathered 237,288 hashes from VT. We attempted to balance the presence of families and variants to avoid over-representing certain prolific families. Malware feeds like VT usually distribute large numbers of variants of the same polymorphic families [68]. The prevalence of a given malware family in such a feed is not necessarily related to its freshness, impact, or prevalence in the wild. For this reason, we used a custom approach to measure AV label similarity. This approach is based on the min-hash algorithm applied to normalised AV labels for samples with at least 20 positives. In order to normalise labels, we leveraged the pre-processing code in the AVClass [62] project. AVClass is a tool that facilitates extraction of the family name that a sample belongs to, according to the labels provided by the different AV vendors.

By leveraging this sample selection method, we avoided the over-representation of prominent polymorphic families (such as worms or file infectors like Virut or Allaple) [68] or identical variants of the same family. The removal of these variants balanced the representation of every family in the dataset.

We collected a total of 1.5 million unique PE executable files from VT in the course of 6 days (December 10th to December 16th, 2019), and finally selected a subset of 237,288 diverse

Type	Dataset Name	No. Of Hashes	No. Of Behavioural Reports	No. Of Crash Reports	No. Of Blank Reports
Public	Ember [3]	1,235,190	612,400	56,339	113,021
	Ember Benign	740,679	158,763	10,364	76,320
	VirusShare [69]	18,176,364	8,639,474	234,416	2,027,913
	Vx Underground [70]	394,383	102,541	6,550	28,952
	Georgia Tech [26]	8,070,223	5,095,615	285,134	281,451
	MalShare [33]	2,903,350	1,277,507	66,319	176,701
Private	APT	16,232	7,668	336	2,081
	Yara	31,840	31,834	436	50
	VirusTotal Balanced	237,288	122,400	10,270	16,513

TABLE IV: Constituent datasets and their behavioural reports.

samples. We make these hashes available to enable further research.

APT Malware. We gathered a dataset of APT malware following a method similar to the dataset paper *dAPTaset* [59]. We processed HTML pages and pdf files and extracted all malware hashes contained within these files. We collected APT reports from various threat intelligence companies, government agencies and other entities that release threat intelligence data online, mostly from the APTnotes [4] repository. We then processed these reports, creating an aggregated collection of APT malware hashes, which we make public.

Yara Rule Match Malware. We collected an additional dataset by leveraging the live hunting service from VT. To this end, we deployed three Yara [9] rules we wrote to detect the use of LotL binaries based on the behavioural footprint of the samples. We make these rules public in our repository. Livehunt ran the provided Yara rules against all malware binaries uploaded to the VT platform. We used Livehunt to identify new malware hashes uploaded to VT that matched the behavioural characteristics of malware using LotL techniques. We subsequently inserted these malware hashes into our database for augmentation and analysis.

Our Yara rules import an extension of the default Yara libraries that integrates the use of the Cuckoo module [74]. The Cuckoo module in Yara allows rules to match artefacts of the behavioural report. To this end, we leveraged the *file_access* parameter. Whenever a process creates another process, the report shows an entry indicating that the image (PE file) of the created process has been accessed. This artefact allowed us to preselect candidate samples by finding occurrences of the file system locations of LotL binaries being accessed.

B. Analysis Pipeline

Once we had gathered the different datasets composed of Windows Portable Executable (PE) binaries, we analysed the behaviour of the samples. To facilitate this we implemented an analysis pipeline consisting of three phases: data collection (described in the previous section), data augmentation and analysis.

We augmented the data for the collected malware hashes by querying VT's public API and adding the returned additional data to our database. We added this data to provide contextual information about the malware under analysis as well as enabling the filtering of malware by characteristics. This additional data includes detection by different AV products, the *first_seen* date and behavioural reports from dynamic analysis of the binary. Each behavioural analysis report contains a list of shell commands and processes executed or created. We additionally ran AVClass [62] against each sample to add a family classification. Finally, we processed all the behavioural reports to identify the samples that apply any LotL technique among all the candidates. We provide a detailed description of the process followed in Section IV-C.

C. LotL Technique Identification

We processed all the collected behavioural reports by using pattern matching to identify invocations of LotL binaries within the execution of the malware. For this, we parsed the following report indicators:

Shell Commands. Shell commands are commands that a malicious binary executes in the shell of the host operating system. Shell command logs can show executions of a system binary by a reference to its absolute path. The command prompt for Windows also includes a number of aliases such as *reg* for `Reg.exe`, by which system binaries can also be executed.

Processes. Process logs specify system binaries executed by a malware sample. Parameters provided to the executed binary are also contained within process logs in behavioural reports.

In our analysis, we assessed that a sample uses LotL techniques if its behavioural report contained at least one execution of a LotL binary. We recorded the details of every LotL

binary execution with its parameters and inserted them into the database. We then analysed the parameters of these malware samples to determine the most frequent parameter types and execution purposes in every dataset. More specifically, we identify these two separate types of binaries:

Default System Binaries. This set of binaries comprises those native to the system according to the LOLBAS project, together with those present in the System32 folder of a Windows 10 system. To identify the execution of these binaries, we conducted string matching over the process logs.

Installed Signed Binaries. We extended our selection of LotL binaries with those not installed by default, mentioned in the 2017 Symantec LotL report and the SysInternals tool suite [36], in addition to those listed in the LOLBAS project.

Pattern Matching Refinement. We iteratively validated a cross-section of each result set, identified misclassified cases and removed them from our results database, refining our pattern matching approach. We conducted this until all identified LotL commands were properly categorised and mapped to an execution purpose. Additionally, as shown in Table IV, we identified anti-VM malware artefacts or erroneous reports and do not include them within our measurements.

We also found a number of LotL binary executions where no parameters were provided. These occur in the case of either an execution of a binary without parameters, or process hollowing, replacement or injection, all of which are out of scope for our analysis. To justify removal, we manually validated a small number of these samples.

Additionally, we identified several system binaries being executed in the reports that were artefacts of the sandbox and are not directly related to LotL technique usage. One such forensic artefact we removed is the execution of `Explorer.exe`, that is typically a remnant of the execution of malware samples by the sandboxed environment. We removed executions of `Explorer.exe` with the parameter of the absolute address of *C:* and a suffix of the SHA256 hash of the malware sample in question. We did not remove forensically relevant executions of `Explorer.exe` to open webpages or other directories.

We also removed instances of `Verclsid.exe`, after manually verifying a subset of the 64,385 executions in our dataset. We removed these instances as they were simply forensic artefacts showing verification of COM objects before they were instantiated by Windows Explorer. Whilst the malicious usage of this tool to spawn processes [29] is known, we did not observe any occurrences of this behaviour in our dataset. Finally, we found that many legitimate binaries from the Ember benign dataset were spawning `Msiexec.exe`. We manually inspected these cases and determined them to be legitimate installer files packaged with the MSI installer tool.

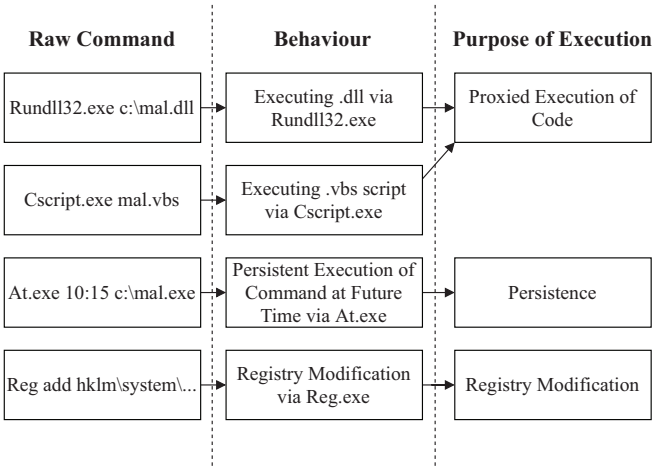

Fig. 1: Examples of raw commands being mapped to behaviours, which are in turn mapped to execution purposes. This illustrates the classification heuristic we applied to identify an execution purpose.

We discarded these cases as they do not represent LotL binary usage.

D. Parameter Analysis to Identify Execution Purpose

To identify the purpose of LotL technique executions, we observed the parameters provided by malware samples. Figure 1 illustrates the mapping of four process executions. This mapping was conducted across all datasets by identifying individual execution purposes, such as executions of `Net.exe` with the *stop* parameter as task stopping. After individual commands had been mapped to an execution purpose, we then selected all matching executions for that binary. We repeated this across all system binary executions until every execution was classified as belonging to a specific execution purpose or removed as a misclassified case (refining our pattern matching rules). Following this approach, we grouped the parameters by purpose into nine separate categories. Three of these categories are related to execution:

- **Proxied Execution.** Malware using a LotL binary to abstract execution of other code *e.g.,* `Mshta.exe` executing *.hta* files and `Rundll32.exe` executing *.dll* files.
- **Persistence.** Malicious code achieves persistence if it configures or modifies the system to execute a command or stored job at a future point in time. This is typically resultant from a system configuration change by a malware sample. *e.g.,* `Sc.exe` with the *create* parameter, `Bitsadmin.exe` with the */create* parameter or `Schtasks.exe`/`At.exe` with a datetime parameter.
- **Delayed Execution.** Using a LotL binary to delay execution *e.g.,* `Ping.exe` executing *-n*, followed with a number, and an `&` to trigger the execution of another binary.

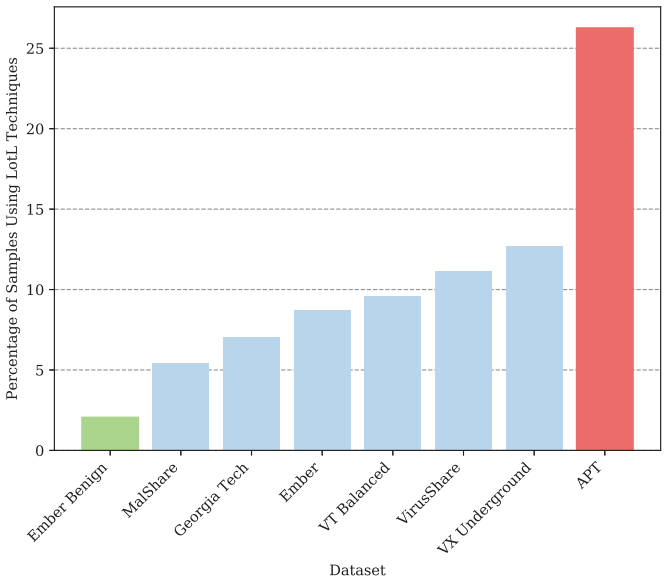

Fig. 2: Relative prevalence of samples using LotL techniques per dataset.

The following three categories, instead, relate to the modification of underlying system components. Malware typically engages in this behaviour in order to enable further propagation or action on objectives on a machine.

- **Firewall Modification.** Modifying firewall configuration *e.g.,* `Netsh.exe` with the *Firewall* parameter and successive commands.
- **Registry Modification.** Modifying registry settings *e.g.,* `Reg.exe` with an *Add* or *Delete* parameter and a registry location.
- **Permissions Modification.** Modifying permissions of a file *e.g.,* `Cacls.exe` with a parameter that includes an absolute file location.

Finally, we distinguish the following three categories that are not related to execution or system modification:

- **File Opening.** Opening a file, via binary execution *e.g.,* `Explorer.exe` followed by a relative or absolute file location.
- **Reconnaissance.** Elicitation of local or remote configuration for lateral movement, leveraging a LotL binary *e.g.,* `Net.exe` with *user localgroup administrators* or execution of `Ipconfig.exe`, outputting to a file.
- **Task Stopping.** Surreptitious stopping of another process or service using a LotL binary *e.g.,* `Taskkill.exe` executed with another process name as parameter. In many cases, the goal of these executions is to disrupt system security services.

System Binary	Frequency of LotL Binaries By Dataset					
	VTB	Ember	GT	MS	VS	VXU
Reg	15.49	7.07	42.16	4.26	11.10	5.88
Nslookup	15.14	4.49	0.58	4.55	0.00	0.70
Regasm	9.93	1.25	0.00	0.44	0.00	1.60
Runas	7.84	0.39	6.51	0.00	0.00	0.00
Schtasks	7.50	3.78	3.66	0.26	0.00	4.27
Sc	5.87	6.08	1.44	1.17	14.08	10.00
Wscript	3.31	1.95	1.59	0.61	2.36	5.50
Rundll32	3.16	4.70	2.63	14.00	5.25	8.62
Regsvr32	2.99	3.62	2.99	8.58	4.47	5.87
Attrib	2.83	4.63	15.59	0.32	1.18	3.63
Net	2.52	8.45	4.14	1.89	9.85	9.48
Ping	2.14	27.19	5.61	1.31	5.06	4.81
Taskkill	1.49	2.39	0.67	4.04	3.40	6.30
Netsh	1.40	3.37	0.62	2.49	5.19	6.54
Timeout	1.36	0.74	0.56	0.33	0.00	1.13
Wmic	1.27	0.62	0.50	36.14	9.63	0.55
Mshta	1.09	4.68	0.76	10.72	0.74	0.60
Cacls	0.89	0.00	0.48	0.23	0.80	3.11
Regedit	0.52	1.55	0.00	0.00	6.97	2.79
Tasklist	0.00	0.00	0.00	0.00	2.60	0.85
Cscript	0.00	0.88	3.96	0.00	1.52	0.00
Explorer	0.69	0.69	0.41	0.00	1.91	6.0
Msiexec	0.55	1.78	1.78	0.57	0.58	0.00
Vssadmin	0.00	0.81	0.00	0.58	0.00	0.00

TABLE V: Most frequently used binaries for each dataset (percentage of the number of distinct sample-binary executions observed). 5 most common highlighted in grey.

V. Measurement Results

We followed the methodology described in Section IV to evaluate the prevalence and nature of LotL techniques, and to answer the research questions raised for this study.

A. Prevalence of LotL Techniques in Commodity Malware

Relative Prevalence Between Datasets. Figure 2 shows the percentage of malware samples that use LotL techniques in every dataset. If we omit Ember's benign dataset and the APT malware dataset, we can observe that between 5.42% and 12.72% of commodity malware samples used a LotL technique at least once. Within our VT balanced dataset, we can observe that 9.6% of samples leveraged this type of technique. These numbers show that LotL techniques are a relatively widespread phenomenon and not an anecdotal occurrence in specific malware samples or families.

Interestingly, APT datasets employed LotL techniques significantly more frequently than commodity malware. 26.26% of the APT dataset made use of LotL techniques, more than twice that of comparable commodity malware datasets. Our findings confirm the assertions of many reports that state that APT campaigns use LotL techniques due to the increased sophistication and evasive requirements of state-sponsored malware authors. Section V-C presents deeper findings about the APT dataset, while the rest of this section focuses on commodity malware. We excluded the Yara dataset from this

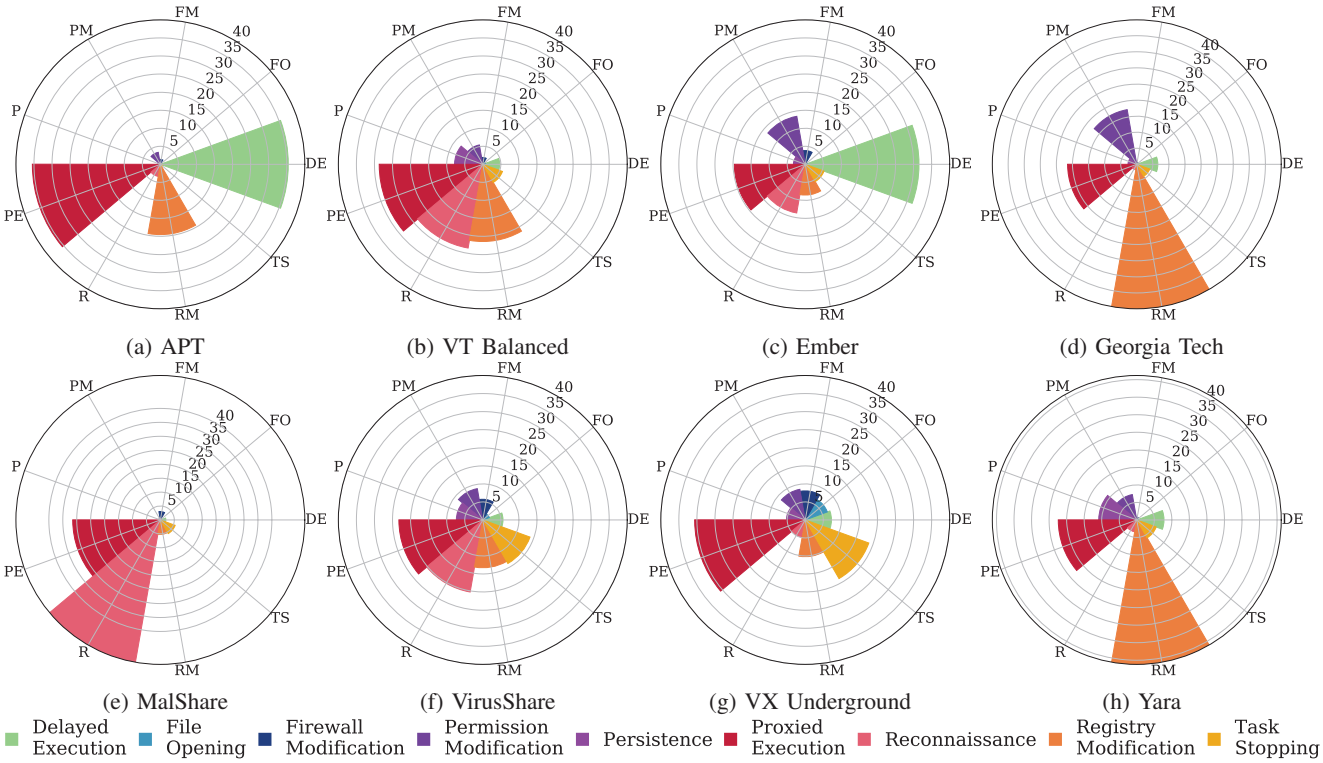

Fig. 3: Comparative execution purposes for each dataset by percentage.

figure given its different nature. In this particular case, 47% of the files retrieved by our Yara rules were later assessed to have used some kind of LotL technique. This result is also interesting, as it means that half of the times that a malicious sample accessed a system binary, it did so to leverage it as a LotL technique, instead of for other purposes.

Another notable finding is that in the Ember benign dataset, 2.05% of the samples show executions of LotL techniques. This number is significantly lower than the 9.41% average prevalence of LotL techniques across regular commodity malware datasets. We provide additional evidence for this observation in Section V-B.

Most Frequently Used LotL Binaries. Table V shows the most commonly executed LotL binaries in our commodity malware datasets. We can observe that commodity malware executes some system binaries more frequently than others. These numbers also show a significant variability depending on the specific dataset distribution, as each of the datasets might have been collected in a different way and at a different time. If we look at the VT balanced dataset, the most commonly used binaries were `Reg.exe`, `Nslookup.exe`, `Regasm.exe`, `Runas.exe`, `Schtasks.exe`, and `Sc.exe`. Some of these binaries were used for system administration tasks such as editing the registry or creating scheduled tasks. Others were used to change or elevate privileges, or to enable network activities. It is notable that certain binaries show different distributions across datasets. This means that although LotL techniques are a widespread approach among malware writers, they are used in an heterogeneous way among malware families or malware types.

Parameters and Execution Purpose. We also identified the execution purpose for the occurrences of the most commonly used LotL binaries by parsing their parameters. Figure 3 shows the resulting distribution of execution purposes. The most frequent purposes are proxied execution, reconnaissance, task stopping and modification of the registry. We can also observe that there is a significant variability between datasets, as expected from the results shown in Table V. Nevertheless, these results show that malware uses LotL binaries not only to surreptitiously execute other code, but also to modify the underlying operating system via registry modification, to enable lateral movement via reconnaissance, or to avoid other software running in the system (e.g., many malware families try to stop security software applications when they start running on a system).

Malware Families. We identified a number of malware families that are notably prolific (i.e., those for which nearly all of their samples used one or more LotL techniques), while others had a very low LotL binary adoption. We limited this classification to families that contained at least 100 samples, to prevent bias caused by potentially under-represented families with a small number of samples.

Figure 4 shows the number of families, as reported by AVClass, for which a given percentage of their samples leveraged

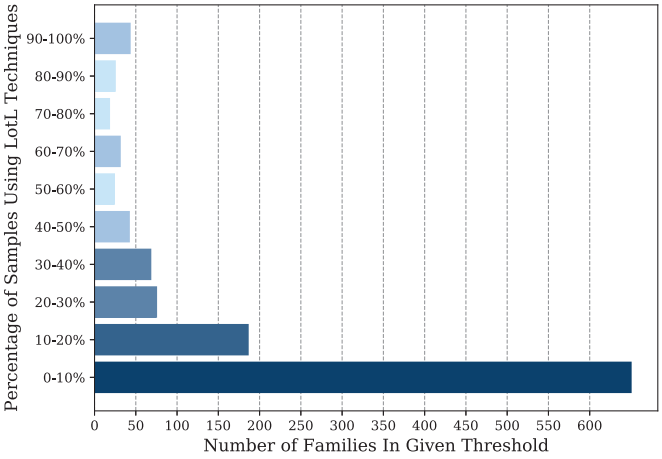

Fig. 4: Distribution of families using LotL techniques.

a LotL technique. We can observe that a high number of families did not use LotL binaries, while a small number of families presented high prevalence of these techniques. This means that while there is a significant adoption of LotL overall, the usage of these techniques is concentrated in a limited number of families.

B. Comparison of Benign and Malicious Samples

We have observed that LotL techniques are a widespread phenomenon both in commodity malware and APT campaigns, and although there might be some differences across malware types or families, it is clear that the security industry should not overlook this threat. With this in mind, we raise another question: *Why do some security products have difficulties detecting these techniques?* At the beginning of this paper we confirm that many AV products cannot effectively detect the usage of these binaries, and some responses to our responsible disclosure messages suggest that detection mechanisms could be very prone to false positives. In this section, we compare the usage of LotL binaries in legitimate software. To this end, we leveraged the benign software dataset from Ember described in more detail in Section B, and our VT balanced dataset.

Table VI shows the most commonly executed binaries in the benign dataset. `Regsvr32.exe`, `Sc.exe` and `Rundll32.exe` are among the most commonly executed binaries, all of which are used for the purpose of executing code. We can notice that the most commonly executed binaries for malware (see Table V) and benign software differ significantly.

Figure 5 shows samples that leverage a LotL technique, grouped by execution purpose. We observe that while some execution purposes are more prevalent in malicious binaries than in benign binaries (e.g. reconnaissance, persistence and registry modification), there is an opposite trend for others. For instance, we can notice that the most frequent purpose in both malware and benign software is proxied execution. This finding correlates with our observations in Section III, where

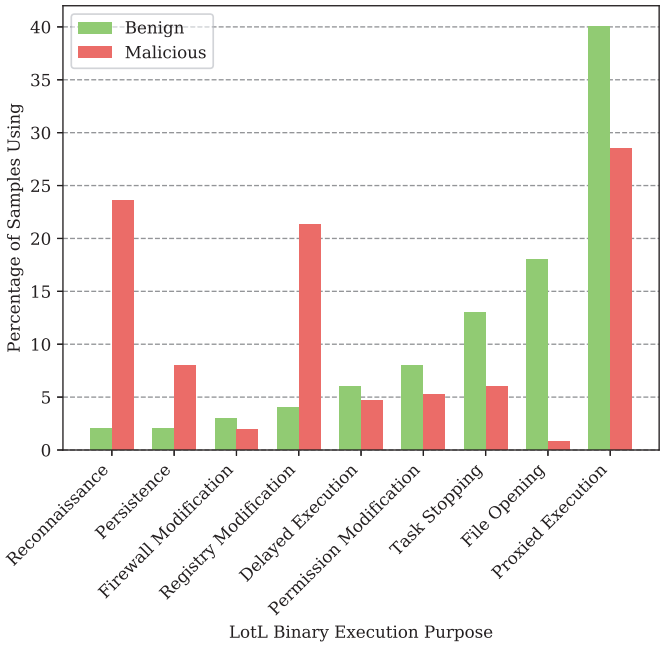

Fig. 5: Execution purpose of benign and malicious samples from Ember and Ember Benign datasets.

System Binary	Samples	Percentage
Explorer	230	12.62%
Regsvr32	190	10.43%
Sc	148	8.12%
Rundll32	128	7.03%
Taskkill	99	5.43%
Ping	71	3.90%
Net	66	3.62%
Mstsc	58	3.18%
Attrib	58	3.18%
Regedit	40	2.20%

TABLE VI: Top 10 most frequently used binaries in benign dataset (number of samples and percentage of the number of distinct sample-binary executions observed).

we tested several AV solutions to find out if they detected the proxied execution of a reverse shell. The results show that these behavioural patterns pose a significant challenge for creators of heuristic AV algorithms. This is also reflected by the response of some vendors to our responsible disclosure process, where one of the claims suggested that the false positive risk was significantly high. The finding also simultaneously illustrates that not all LotL binaries or purposes are equally prevalent in benign and malicious software, and therefore AV vendors still have the opportunity to create heuristic rules to identify the types of techniques that are not as prevalent in benign software.

C. Prevalence of LotL techniques in APT Malware

In the previous section, we observed how LotL techniques are significantly more frequent in APT malware than commodity malware (see Figure 2). For this reason, in this section we

System Binary	Samples	Percentage
Ping	493	25.96%
Rundll32	228	12.01%
Reg	212	11.16%
Wscript	194	10.22%
Xcopy	122	6.42%
Net	77	4.05%
Tasklist	48	2.53%
Ipconfig	47	2.47%
Expand	44	2.32%
Systeminfo	41	2.16%

TABLE VII: Top 10 most frequently used binaries in APT dataset (number of samples and percentage of the number of distinct sample-binary executions observed).

APT Campaign	Percentage	LotL Binaries
Havex	100.00%	Rundll32
Hurricane Panda	100.00%	Msiexec
El Machete	100.00%	Schtasks
Regin	100.00%	Dllhost
Lazarus	100.00%	Net, Netstat, Ping, Reg
Keyboy	100.00%	Net, Rundll32
Keyboy	100.00%	Rundll32
Black Vine	94.39%	Net, Ping, Reg, Regsvr32, Rundll32
Roaming Tiger	89.47%	Expand, Powershell
WIRTE Group	80.00%	Rundll32, Schtasks, Wmic
APT28 Zebrocy	77.78%	Reg, Tasklist
Magic Hound	75.00%	Attrib, Taskkill, Wscript
Hangover	71.86%	Attrib, Cscript, Findstr, Net, Reg, Wscript
APT28 Zebrocy	66.67%	Mshta, Wscript
Lotus Blossom	66.67%	Net, Rundll32
APT27	63.64%	Msiexec
Subaat	60.42%	Attrib, Ping, Reg, Regasm
Dimnie	54.36%	Ping, Rundll2

TABLE VIII: APT campaigns (from threat intelligence campaign reports) with over 50% of samples using LotL binaries.

focus on APT malware and measure several aspects. First, we show the particular APT campaigns with the highest percentage of samples that use LotL binaries, and we group them based on threat intelligence reports. We also enumerate the LotL binaries that APT malware leverages most often.

Table VII lists the most common LotL binaries that we observed in the APT dataset. We can observe that that APT malware uses many of the same binaries as commodity malware, with `Ping.exe` as the most frequent one. This binary is often used for subtly delaying execution, with '*localhost*' as the most frequent destination. `Rundll32.exe` and `Wscript.exe`, tools for proxying the execution of malicious code, are used by many samples; substantially more than they are within commodity malware datasets. The reconnaissance tools `Ipconfig.exe`, `Net.exe` and `Tasklist.exe` are also used by a number of different samples. `Reg.exe` is commonly executed by APT malware, but more infrequently than in commodity malware. We observe that APT malware mainly uses LotL techniques for delayed and proxied execution, as well as reconnaissance. This may in part reflect that APT malware aims to enable stealthy lateral movement and exfiltrate data.

APT Campaigns. Table VIII displays some of the APT campaigns we identified as using LotL techniques most prolifically, based on threat intelligence reports where behavioural data was available. It is notable that several APT groups, such as APT28 and Keyboy, have multiple campaigns and samples. This indicates that APT groups that use LotL techniques use them across multiple campaigns.

VI. Case Studies

In this section we investigate and describe two recent ransomware families from our dataset: Gandcrab and Cerber. Ransomware is one of the most prevalent threats nowadays due to how profitable it is for malicious actors [23]. We chose these families because they heavily rely on LotL techniques of different types and execution purposes and illustrate how a malware sample can chain different LotL techniques to achieve its goals. We executed samples from these families to observe and document their behaviour at different phases of the intrusion killchain.

We also describe two APT malware groups, Turla and GreyEnergy. APT group Turla shows an evolution in the adoption of LotL techniques over time, while GreyEnergy [53] was involved in highly disruptive attacks to the Ukranian power grid. We manually analysed the use of LotL binaries in these samples.

a) Gandcrab: Gandcrab is a ransomware family that operated from late 2018 to early 2019, in several different variants. We executed Gandcrab ransomware samples in a local sandboxed environment and recorded the executed commands. We observed the following LotL binaries being used by Gandcrab samples:

```
C:\Windows\System32\cmd.exe /c vssadmin delete
    shadows /all /quiet
C:\Windows\System32\cmd.exe /c wmic shadowcopy
    delete
```

Many other ransomware families, such as *Petya* use the LotL binary `Vssadmin.exe`, with the parameters *delete shadows /all /quiet*, to silently delete backups. Functionally *shadowcopy delete* provided as parameters to `Wmic.exe` accomplishes the same effect.

We observed other samples in the family using the `Nslookup.exe` binary, passing a parameter of '*gandcrab.bit*' or a similar domain, with no further functionality. This subroutine is repetitively executed in attempts to contact command and control servers for Gandcrab [28] that were no longer online and responding to the query.

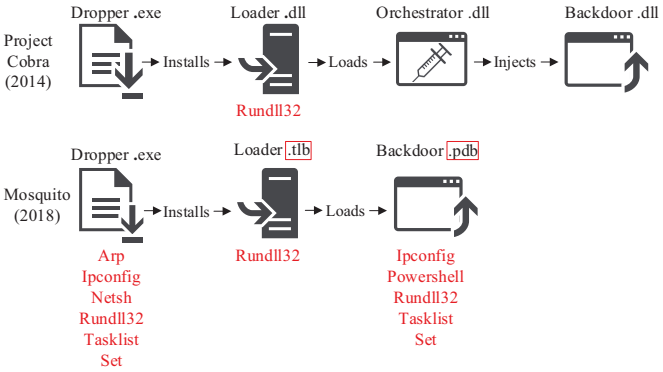

Fig. 6: Increasing adoption of LotL techniques by APT group Turla. Shown in two successive modular highly evasive trojans, Cobra (2014) [16] and Mosquito (2018) [14].

b) Cerber: Cerber, like Gandcrab, is a notably successful ransomware that has several variants. Apart from proxied execution via LotL binaries, we observed other evasive techniques in use by Cerber, such as process injection.

Unlike other ransomware families, Cerber does not delete shadow copies. However, we noticed an especially complex proxied execution chain. We observed that `Wscript.exe` and `Rundll32.exe` execute a malicious *.vbs* and *.dll* file, sequentially:

```
C:\Windows\System32\WScript.exe C:\Users\admin
    \enYXval36C\38oDr5.vbs
C:\Windows\System32\Rundll32.exe 8ivq.dll
    arzy949
```

We also observed that Cerber delays the execution via `Ping.exe` and modifies the registry to achieve persistence. The addition of a malicious *.vbs* file to the `HKEY_CURRENT_USER\Software\Microsoft\Windows` registry key means this malicious script is run every time a user logs on.

```
C:\Windows\System32\cmd.exe /c ping 127.0.0.1
    && reg add HKEY_CURRENT_USER\Software\
    Microsoft\Windows\CurrentVersion\RunOnce /
    v enYXval36C /t REG_SZ /d C:\
    enYXval36CenYXval36C\enYXval36C.vbs /f
```

We also note that some Cerber samples invoke `Mshta.exe` to open the ransom note.

```
C:\Windows\System32\mshta.exe ransom.hta
```

c) Turla and GreyEnergy: We executed the GreyEnergy [53] dropper malware locally (used for attack campaigns on the Ukranian power grid) and recorded the following shell commands executed using Process Monitor [40]:

```
C:\Windows\SysWOW64\Rundll32.exe {64F97CDC...
    FAB40CA\}.db #1 #1}
C:\Windows\System32\cmd.exe /c (ping localhost
    >> nul & del $path\grey3.exe >> nul)
```

The execution of `Rundll32.exe` by GreyEnergy with the parameter of a *.db* file causes the proxied execution of a malicious *.dll* file, given a *.db* suffix to appear more innocuous. This technique is also used by the Mosquito malware visible in Figure 6, however Mosquito masks malicious *.dll*s with the use of *.tlb* and *.pdb* extensions to the files. Figure 6 also illustrates how the Turla APT group has adopted more LotL techniques in its recent malware campaigns, and exemplifies how some malicious actors are evolving to conceal their actions.

`Rundll32.exe` execution by GreyEnergy is also made persistent by having its location stored in a startup folder that executes every time the system starts. The execution purpose of `Ping.exe` by the second command of the GreyEnergy dropper is to delay the deletion of the initial dropper file. The dropper deletes itself to remove forensic evidence.

VII. Main Takeaways and Discussion

Our results confirmed that LotL techniques are not a negligible phenomenon. While several technical articles cover this topic, our systematic measurement went one step further in facilitating the understanding of the adoption of this technique by commodity malware. In this paper, we examined several different malware datasets and conducted an AV evaluation experiment, wherein we confirmed low detection rates for several documented LotL techniques. Given these results, we can distill the following conclusions and takeaways:

- Almost every popular AV product we tested had difficulties detecting malicious usage of LotL binaries. Even after responsibly disclosing these issues to each vendor, only a fraction of them implemented successful detection mechanisms. Some of these vendors implemented detection mechanisms for the specific malicious payloads we implemented, but not the delivery mechanism itself. Others reported that it is challenging to implement such countermeasures due to a prohibitive false positive risk.

- To correct this detection gap, we worked with AV vendors directly to implement detection capabilities. For instance, Kaspersky implemented detections resultant from our responsible disclosure process, such as `Trojan.Win32.Lolbas.btad.exec`, `Trojan-Spy.Win32.Agent.ftps` and others. We also release Yara rules to aid in the detection of LotL techniques. Furthermore, our study shows that there are differences in execution purposes between benign and malicious samples, providing a vector for development of detection algorithms. In fact, recent papers [71, 21] explore this promising line of research to overcome the limitation of existing security products.

- In some descriptions, LotL techniques exclusively refer to the subset of approaches that can achieve AV evasion. Nevertheless, in this paper we adopted a broader scope, and observed that within a balanced dataset of malware, 9.6% of malware used a native system binary to perform a malicious action.

- With regards to execution purpose, we observe that LotL binaries were not only leveraged for proxied execution or evasion, but also to implement common malicious routines such as delaying execution, modifying system configuration, ensuring persistence or stopping security services.
- There was a large variability of prevalence of LotL techniques among different families as reported by the AVClass tool. Nevertheless, we can observe that a majority of families showed a low prevalence, while a minority of families used these techniques with a much higher prevalence. This means that these techniques are common enough to be adopted by certain malware authors, but are not the only (or most prevalent) way of implementing malicious functionality in malware.
- Legimate software used LotL binaries less than malware, and although these binaries were used for different purposes, the prevalence is significant enough to make the accurate detection of malicious usage a challenge for security vendors. Conversely, APT malware leveraged LotL binaries at twice the rate of commodity malware.

Given this evidence, we can conclude that LotL techniques have a significant adoption in current, state-of-the-art malware and that they represent a challenge for the security industry from a detection perspective. With this paper, we tried to shed light onto this phenomenon and raise the awareness within the research community about this open problem.

LOLBAS Project Contributions. As an examplar of some of the techniques in use by modern malware, we identified and documented two new Windows system binaries and added them to the LOLBAS project repository [46].

`At.exe` executes scheduled tasks at a future time, which is used to stealthily maintain persistence on infected systems. An example of this is SoftCell [30], a threat group that uses `At.exe` for the purpose of persistence. Many other threat groups use `Schtasks.exe` to execute scheduled tasks, for persistence.

`Netsh.exe` is a Windows binary that is typically used to alter firewall configurations. This is the only purpose we observe it being used for by malware. However, we document a use case, where `Netsh.exe` can be used to load malicious *.dll*s.

VIII. Limitations & Future Work

Intended or Unexpected Functionality. Our measurement results do not differentiate between the standard functionality of binaries versus non-standard uses that leverage side effects to achieve some result. Some of these cases are documented in the LOLBAS project [46]. An example of intended or standard usage is `Netsh.exe` being used to modify firewall rules, while an unintended usage would be to use `Netsh.exe` to run *.dll*s.

We do measure whether parameters from executed samples match these usage patterns. Malware binaries in our dataset do employ these techniques, but make up less than 2% of total measured LotL binary usage. For this reason, we do not include a comparison of intent in our measurement results.

Anti-VM Malware. Due to data originating from dynamic analysis sandboxes hosted in the cloud, anti-VM evasion techniques used by malware may affect data quality. We mitigate this through the exclusion of malware samples that have minimal execution or crash during execution in the sandbox. Malware has been making use of environment detection and similar methodologies to halt execution [44] for a number of years. This may affect data quality, as evasive malware making use of anti-VM techniques does not demonstrate its actual behaviour. As such, we may underestimate the number of samples that potentially use LotL binaries. Nevertheless, our numbers are a lower-bound of the prevalence of this technique and prove that it is a significant phenomenon that should not be ignored.

Human Operators. Many threat actors that use LotL techniques are human operators executing them from remote shells, offensive security tools, Powershell, Visual Basic or Batch scripts [30]. While there is anecdotal evidence of cases like these, it is harder to generalise and to conduct a measurement study over a representative dataset as we did. Instead, we narrowed the scope of our paper and focused on the usage of LotL binaries by malware. We show that this phenomenon is not negligible and that malware authors also leverage these techniques in their binaries and not solely in post-exploitation scripts.

Linux LotL. A future line of research is to explore the use of these techniques on Linux systems (titled GTFObins [19]). In a similar manner to LotL techniques on Windows, these binaries can be used to achieve malicious functionality. Whilst Linux malware are not as numerous as their Windows counterparts, it is a subject worthy of analysis due to the rise of IoT botnets running lightweight Linux systems [12].

Detection. Another future direction for research in this field is the deployment of detection technologies that attempt to accurately capture the identified patterns of use for LotL techniques such as Endpoint Detection and Response (EDR) systems. Future research should leverage recent work [71, 21] on data provenance analysis for process execution chains to enable modelling of legitimate process relationships and identification of suspicious behavioural patterns.

Availability

Research artefacts are available at https://livingoffthe.land and https://github.com/ssloxford/livingoffthe land.

Acknowledgements

Thanks to Paul Pearce for shepherding this paper. Thanks also to Bernardo Quintero and Emiliano Martinez of VirusTotal and

Sunil Potti of Google Cloud Security for access to VirusTotal process execution data.

REFERENCES

[1] "Threat Brief: Iranian-Linked Cyber Operations," 2020. [Online]. Available: https://unit42.paloaltonetworks.com/threat-brief-iranian-linked-cyber-operations/

[2] H. Aghakhani, F. Gritti, F. Mecca, M. Lindorfer, S. Ortolani, D. Balzarotti, G. Vigna, and C. Kruegel, "When Malware is Packin' Heat; Limits of Machine Learning Classifiers Based on Static Analysis Features." Network and Distributed System Security Symposium, 2020.

[3] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," 2018. [Online]. Available: http://arxiv.org/abs/1804.04637

[4] Aptnotes, "aptnotes." [Online]. Available: https://github.com/aptnotes/data

[5] U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel, "A View on Current Malware Behaviors," 2009. [Online]. Available: http://static.usenix.org/events/leet09/tech/full{_}papers/bayer/bayer.pdf

[6] D. Bohannon, "Invoke-Obfuscation," 2019. [Online]. Available: https://github.com/danielbohannon/Invoke-Obfuscation

[7] A. Brandt and M. Loman, "Living Off Another Land: Ransomware Borrows Vulnerable Driver to Remove Security Software," 2020. [Online]. Available: https://news.sophos.com/en-us/2020/02/06/living-off-another-land-ransomware-borrows-vulnerable-driver-to-remove-security-software/

[8] ClearSky Cyber Security, "MuddyWater Operations in Lebanon and Oman Using an Israeli compromised domain for a two-stage campaign," pp. 1–16, 2018. [Online]. Available: https://www.clearskysec.com/wp-content/uploads/2018/11/MuddyWater-Operations-in-Lebanon-and-Oman.pdf

[9] M. Cohen, "Scanning memory with Yara," *Digital Investigation*, vol. 20, pp. 34–43, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2017.02.005

[10] Context Information Security, "AMSI Bypass," 2019. [Online]. Available: https://www.contextis.com/en/blog/amsi-bypass

[11] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert, "ZOZZLE: Fast and Precise in-Browser JavaScript Malware Detection," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. USA: USENIX Association, 2011, p. 3.

[12] F. Dang, Z. Li, Y. Liu, E. Zhai, Q. A. Chen, T. Xu, Y. Chen, and J. Yang, "Understanding fileless attacks on linux-based IoT devices with HoneyCloud," *MobiSys 2019 - Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 482–493, 2019.

[13] Department of Homeland Security, "Russian Government Cyber Activity Targeting Energy and Other Critical Infrastructure Sectors (TA18-074A)." [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA18-074A

[14] ESET Research, "Diplomats in Eastern Europe Bitten by a Turla Mosquito," no. January, 2018. [Online]. Available: https://www.welivesecurity.com/wp-content/uploads/2018/01/ESET{_}Turla{_}Mosquito.pdf

[15] FireEye Threat Research, "Dissecting One of APT29's Fileless WMI and PowerShell Backdoors (POSHSPY)," 2017. [Online]. Available: https://www.fireeye.com/blog/threat-research/2017/03/dissecting_one_ofap.html

[16] G Data, "Analysis of Project Cobra," 2015. [Online]. Available: https://www.gdatasoftware.com/blog/2015/01/23926-analysis-of-project-cobra

[17] Github, "Koadic." [Online]. Available: https://github.com/zerosum0x0/koadic

[18] ——, "Powercat," 2019. [Online]. Available: https://github.com/besimorhino/powercat

[19] GTFOBins, "GTFOBins," 2020. [Online]. Available: https://gtfobins.github.io/

[20] Halil Dalabasmaz, "SpookFlare," 2020. [Online]. Available: https://github.com/hlldz/SpookFlare

[21] W. U. Hassan, A. Bates, and D. Marino, "Tactical Provenance Analysis for Endpoint Detection and Response Systems," *IEEE Symposium on Security and Privacy*, 2020. [Online]. Available: https://whassan3.web.engr.illinois.edu/papers/rapsheet-oakland20.pdf

[22] Hood3dRob1n, "JSRat-Py." [Online]. Available: https://github.com/Hood3dRob1n/JSRat-Py

[23] D. Y. Huang, M. M. Aliapoulios, V. G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, A. C. Snoeren, and D. McCoy, "Tracking Ransomware End-to-end," *Proceedings - IEEE Symposium on Security and Privacy*, vol. 2018-May, no. 1, pp. 618–631, 2018.

[24] E. Hutchins, M. Cloppert, and R. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *6th International Conference on Information Warfare and Security, ICIW 2011*, no. July 2005, pp. 113–125, 2011. [Online]. Available: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

[25] IBM X-Force Incident Response, "New Destructive Wiper " ZeroCleare " Targets Energy Sector in the Middle East," no. January, 2020. [Online]. Available: https://www.ibm.com/downloads/cas/OAJ4VZNJ

[26] Impact Cyber Trust, "GT Malware Netflow Daily Feed," 2020. [Online]. Available: https://impactcybertrust.org/dataset{_}view?idDataset=1143

[27] D. Kim, B. J. Kwon, and T. Dumitras, "Certified Malware: Measuring breaches of trust in the windows code-signing PKI," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1435–1448, 2017.

[28] Y. Lemmou and E. M. Souidi, "Inside GandCrab Ransomware," in *Cryptology and Network Security*, J. Camenisch and P. Papadimitratos, Eds. Cham: Springer International Publishing, 2018, pp. 154–174.

[29] M. Levan and K. Haag, "Old Phishing Attacks Deploy a New Methodology: Verclsid.exe," 2017. [Online]. Available: https://redcanary.com/blog/verclsid-exe-threat-detection/

[30] M. Levi, A. Serper, and A. Dahan, "Operation Soft Cell: A Worldwide Campaign Against Telecommunications Providers," *Virus Bulletin*, no. October, pp. 1–13, 2019. [Online]. Available: https://www.virusbulletin.com/uploads/pdf/magazine/2019/VB2019-Serper-Levi.pdf

[31] Z. Li, Y. Chen, Q. A. Chen, T. Zhu, C. Xiong, and H. Yang, "Effective and light-weight deobfuscation and semantic-aware attack detection for powershell scripts," *Proceedings of the ACM Conference on Computer and Communications Security*, p. 3, 2019.

[32] LOLBAS-Project, "LOLBAS-Project." [Online]. Available: https://lolbas-project.github.io/

[33] MalShare, "MalShare," 2020. [Online]. Available: https://malshare.com/

[34] Microsoft, "AllowInsecureGuestAuth." [Online]. Available: https://docs.microsoft.com/en-us/windows-hardware/customize/desktop/unattend/microsoft-windows-workstationservice-allowinsecureguestauth

[35] ——, "Windows Authenticode Portable Executable Signature Format," Microsoft, Tech. Rep., 2008. [Online]. Available: http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/Authenticode{_}PE.docx

[36] ——, "Sysinternals Utilities Index," 2017. [Online]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/

[37] ——, "Out Of Sight But Not Invisible: Defeating Fileless Malware with Behavior Monitoring, AMSI, and Next-Gen AV," Microsoft, Tech. Rep., 2018. [Online]. Available: https://www.microsoft.com/security/blog/2018/09/27/out-of-sight-but-not-invisible-defeating-fileless-malware-with-behavior-monitoring-amsi-and-next-gen-av/

[38] ——, "Insights from one year of tracking a polymorphic threat," 2019. [Online]. Available: https://www.microsoft.com/security/blog/2019/11/26/insights-from-one-year-of-tracking-a-polymorphic-threat/

[39] ——, "Microsoft Recommended Block Rules," 2019. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/microsoft-recommended-block-rules

[40] ——, "Process Monitor," 2019. [Online]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/procmon

[41] Microsoft Defender ATP Research Team, "Latest Astaroth Living-Off-The-Land Attacks Are Even More Invisible But Not Less Observable," p. 2020. [Online]. Available: https://www.microsoft.com/security/blog/2020/03/23/latest-astaroth-living-off-the-land-attacks-are-even-more-invisible-but-not-less-observable/

[42] ——, "Bring your own LOLBin: Multi-stage, fileless Nodersok campaign delivers rare Node.js-based malware," 2019. [Online]. Available: https://www.microsoft.com/security/blog/2019/09/26/bring-your-own-lolbin-multi-stage-fileless-nodersok-campaign-delivers-rare-node-js-based-malware/

[43] ——, "Dismantling a Fileless Campaign: Microsoft Defender ATP's Antivirus Exposes Astaroth Attack," 2019. [Online]. Available: https://www.microsoft.com/security/blog/2019/07/08/dismantling-a-fileless-campaign-microsoft-defender-atp-next-gen-protection-exposes-astaroth-attack/

[44] N. Miramirkhani, M. P. Appini, N. Nikiforakis, and M. Polychronakis, "Spotless Sandboxes: Evading Malware Analysis Systems Using Wear-and-Tear Artifacts," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 1009–1024, 2017.

[45] MITRE, "MITRE ATT&CK." [Online]. Available: https://attack.mitre.org/

[46] O. Moe, "Living Off The Land Binaries and Scripts." [Online]. Available: https://lolbas-project.github.io/

[47] ——, "Discovering The Anti-Virus Signature and Bypassing It," 2020. [Online]. Available: https://www.trustedsec.com/blog/discovering-the-anti-virus-signature-and-bypassing-it/

[48] National Cyber Security Centre, "Advisory: Turla Group Exploits Iranian APT to Expand Coverage of Victims." [Online]. Available: https://www.ncsc.gov.uk/news/turla-group-exploits-iran-apt-to-expand-coverage-of-victims

[49] ——, "APT10 Continuing to Target UK Organisations." [Online]. Available: https://www.ncsc.gov.uk/news/apt10-continuing-target-uk-organisations

[50] ——, "Reckless Campaign of Cyber Attacks by Russian Military Intelligence Service Exposed." [Online]. Available: https://www.ncsc.gov.uk/news/reckless-campaign-cyber-attacks-russian-military-intelligence-service-exposed

[51] National Cybersecurity Authority, "Destructive Attack " DUSTMAN " Technical Report," 2019.

[52] R. Nolen, S. Miller, and R. Valdez, "Threat Advisory: "Squiblydoo" Continues Trend of Attackers Using Native OS Tools to "Live off the Land"." [Online]. Available: https://www.carbonblack.com/2016/04/28/threat-advisory-squiblydoo-continues-trend-of-attackers-using-native-os-tools-to-live-off-the-land/

[53] Nozomi Networks, "GreyEnergy : Dissecting the Malware from Maldoc to Backdoor Comprehensive Reverse Engineering Analysis," no. February, 2019. [Online]. Available: https://www.nozominetworks.com/downloads/US/Nozomi-Networks-GreyEnergy-Dissecting-the-Malware.pdf

[54] Opswat Metadefender-Cloud, "Windows Anti-malware Market Share Report," 2019. [Online]. Available: https://metadefender.opswat.com/reports/anti-malware-market-share?date=2019-09-30

[55] S. Pastrana and G. Suarez-Tangil, "A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth," *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, no. ii, pp. 73–86, 2019. [Online]. Available: https://arxiv.org/pdf/1901.00846.pdf

[56] J. A. Pendergrass, N. Hull, J. Clemens, S. Helble, M. Thober, K. McGill, M. Gregory, and P. Loscocco, "Runtime Detection Of Userspace Implants," 2019.

[57] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "Tesseract: Eliminating experimental bias in malware classification across space and time," *Proceedings of the 28th USENIX Security Symposium*, pp. 729–746, 2019. [Online]. Available: https://www.usenix.org/system/files/sec19-pendlebury.pdf

[58] N. Pinkas, L. Rochberger, and M. Zatz, "Triple Threat: Emotet Deploys Trickbot To Steal Data and Spread Ryuk," 2019. [Online]. Available: https://www.cybereason.com/blog/triple-threat-emotet-deploys-trickbot-to-steal-data-spread-ryuk-ransomware

[59] M. Rezaeirad, B. Farinholt, H. Dharmdasani, P. Pearce, K. Levchenko, and D. McCoy, "Schrödinger's RAT: Profiling the stakeholders in the remote access Trojan ecosystem," *Proceedings of the 27th USENIX Security Symposium*, pp. 1043–1060, 2018. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-rezaeirad.pdf

[60] C. Rossow, C. J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos, and M. Van Steen, "Prudent practices for designing malware experiments: Status quo and outlook," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 65–79, 2012. [Online]. Available: https://oaklandsok.github.io/papers/rossow2012.pdf

[61] A. Rubin, S. Kels, and D. Hendler, "AMSI-Based Detection of Malicious PowerShell Code Using Contextual Embeddings," 2019. [Online]. Available: http://arxiv.org/abs/1905.09538

[62] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "AVclass: A Tool for Massive Malware Labeling," in *Research in Attacks, Intrusions, and Defenses*, F. Monrose, M. Dacier, G. Blanc, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2016, pp. 230–253.

[63] StatCounter, "Desktop Operating System Market Share Worldwide." [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide

[64] Sudhakar and S. Kumar, "An emerging threat Fileless malware: a survey and research challenges," *Cybersecurity*, vol. 3, no. 1, pp. 1–12, 2020. [Online]. Available: https://link.springer.com/content/pdf/10.1186/s42400-019-0043-x.pdf

[65] V. Svajcer, "Hunting for LoLBins," 2019. [Online]. Available: https://blog.talosintelligence.com/2019/11/hunting-for-lolbins.html

[66] Symantec, "Living off the Land: Turning Your Infrastructure Against You," 2019. [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/white-papers/living-off-the-land-turning-your-infrastructure-against-you-en.pdf

[67] D. Ugarte, D. Maiorca, F. Cara, and G. Giacinto, "PowerDrive: Accurate de-obfuscation and analysis of powershell malware," *16th Conference on Detection of Intrusions and Malware & Vulnerability Assessment*, vol. 11543 LNCS, pp. 240–259, 2019. [Online]. Available: http://pralab.diee.unica.it/sites/default/files/dimva19-paper76-final.pdf

[68] X. Ugarte-Pedrero, M. Graziano, and D. Balzarotti, "A Close Look at a Daily Dataset of Malware Samples," *ACM Transactions on Privacy and Security*, vol. 22, no. 1, pp. 1–30, 2019. [Online]. Available: http://s3.eurecom.fr/docs/tops19{_}dailymalware.pdf

[69] VirusShare, "VirusShare.com." [Online]. Available: https://virusshare.com/hashes.4n6

[70] VX Underground, "VX Underground Samples," 2020. [Online]. Available: https://vx-underground.org/samples.html

[71] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, J. Rhee, Z. Chen, W. Cheng, C. A. Gunter, and H. Chen, "You Are What You Do : Hunting Stealthy Malware via Data Provenance Analysis," *Network and Distributed System Security Symposium*, no. February, 2020. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2020/02/24167.pdf

[72] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, J. Rhee, Z. Chen, W. Cheng, C. A. Gunter, and H. Chen, "You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis," *Network and Distributed System Security Symposium*, no. February, 2020. [Online]. Available: https://dx.doi.org/10.14722/ndss.2020.24167

[73] C. Wueest and H. Anand, "Living off the land and fileless attack techniques," Symantec, Tech. Rep., 2017. [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-living-off-the-land-and-fileless-attack-techniques-en.pdf

[74] Yara, "Yara: Cuckoo Module," 2019. [Online]. Available: https://yara.readthedocs.io/en/v3.5.0/modules/cuckoo.html

APPENDIX

A. Dataset Characterisation

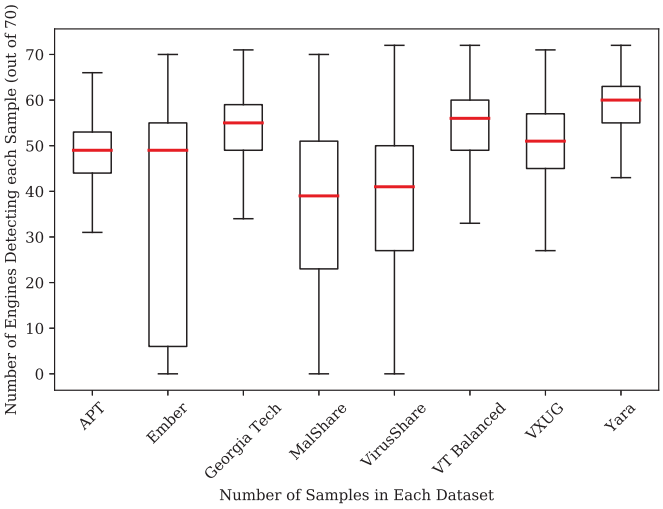

Fig. 7: Number of Positive Detections Distribution

To expand upon the nature of the datasets that we analysed, we illustrate their characteristics and identify the differences between them.

Number of Positives. Within VirusTotal, every sample is scanned by a number of different AV engines. We define the 'Number of Positives' variable as the number of engines that identify a sample as malware. Figure 7 plots the number of positive matches found by various AV engines to characterise the dataset.

First Seen Date. The *first_seen* field is a value within VirusTotal reports which contains the date that this malware was first uploaded to the VT platform. We plot the *first_seen* date for each dataset in Figure 8 and Figure 9, for which the sample range is mostly within the period from 2015 to 2020.

B. Dataset Quality

Rossow et al. [60] describe flaws and limitations in existing malware analysis papers and prudent steps to take in order to avoid biases. We adhere to the three critical assessment criteria (correctness, transparency and realism) as far as is possible in our paper.

Correctness. We removed goodware as recommended by Pendlebury et al. [57], to ensure reliability. We accomplished this by separating elements of the Ember dataset that are labelled as known goodware. We did not remove samples with zero detections, as this may be evasive malware and malware uses LotL techniques partially for the purpose of achieving evasiveness.

We removed hashes from our dataset if there were duplicates in sub-datasets or if there was incomplete data available for a particular hash. We also removed several datasets, reducing our initial count of over 45,000,000 malware hashes and associated reports to a distilled selection of datasets.

Transparency. We describe all datasets and their provenance and characteristics as thoroughly as possible in Appendix A. The virtual environment in which these samples were detonated is VT's fork of Cuckoo sandbox and similar derivatives.

Realism. The aim of dataset selection is to analyse samples that are a representative cross-section of modern malware. We included several well-known public malware datasets as a realistic and typical sample. We also included and focused our analysis on several datasets that we collected, to show the malware landscape from different representative perspectives.

C. Windows 10 System Preparation for AV Experimental Setup

We took two specific steps to facilitate the testing of evasive the potential of malware on Windows 10 systems.

Bypassing AMSI. Windows underlying Anti-Malware Scan Interface (AMSI) is a part of the Windows operating system. Before installing an AV on each virtual system, we disabled AMSI. While various bypasses that have been historically available for Windows 10 systems have been patched afterwards, an AMSI bypass exploit is available at the time of writing [10].

The purpose of using this approach to disable AMSI is to isolate the detection and prevention capabilities of various engines against LotL binaries.

Enabling Unauthenticated Share Access. By default in Windows 10, guest access to shares is disabled. To perform these bypasses this feature was enabled [34]. We accomplished this by running a command as a user and editing group policy

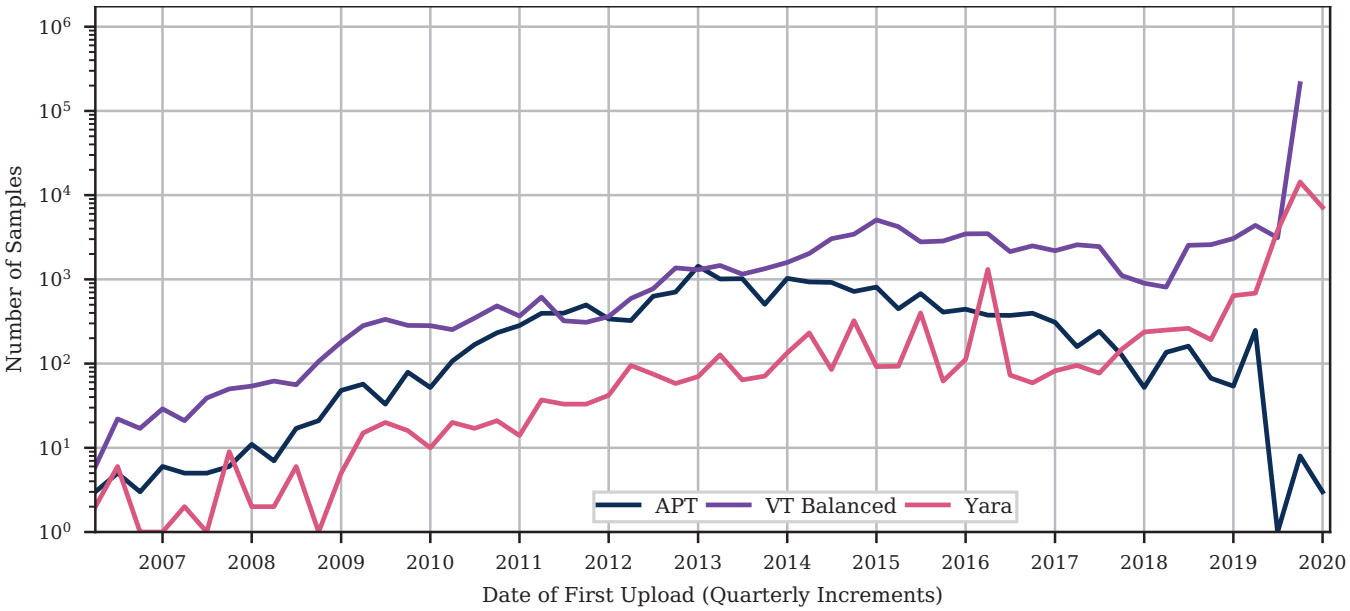

Fig. 8: First Seen Distribution for Private Datasets

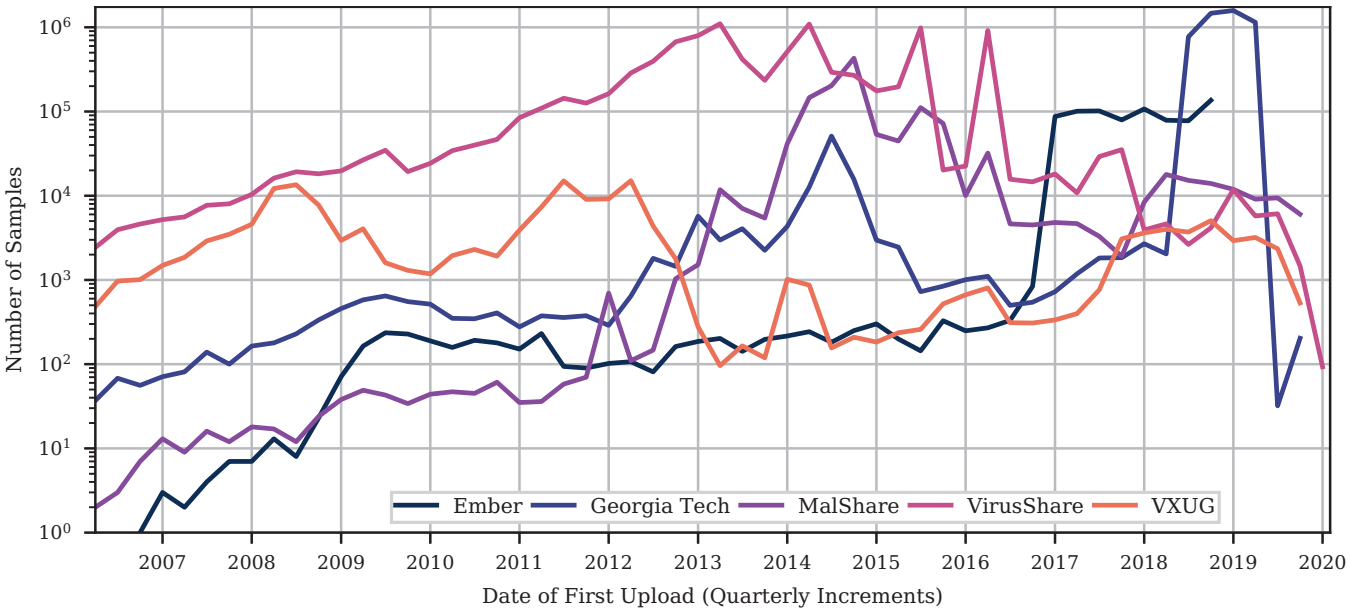

Fig. 9: First Seen Distribution for Public Datasets

to enable insecure guest logons to an SMB share. Whilst unauthenticated access to an SMB share is not necessary for all tests, it was necessary for those that involve staging from an SMB server. Alternatively, SMB Server login details can be cached to access them in a similar manner without the editing of registry keys.

D. LotL binary based Bypass Techniques

Ftp.exe. `Ftp.exe` can be used to execute commands from a stored text file containing instructions (*-s:file.txt* parameter). When the file is executed a simple Powershell reverse shell can be spawned.

For some of the AV engines that are more resilient to evasion techniques, the tool Powercat [18] was invoked to achieve a UDP reverse shell. Additionally for 2 engines, the Invoke-Obfuscation [6] toolkit was invoked to obfuscate the contents of Powercat for evasion.

Mshta.exe. `Mshta.exe` allows the execution of scripts contained within *.hta* files. We found it sufficient to use Mshta to load the *.hta* file from a network share, which would

call a Powershell script from VBScript and then execute a reverse shell to evade AV. Many AV programs were able to identify this payload file as malicious, but when obfuscation was applied (Spookflare [20]), detection was avoided.

Wmic.exe. `Wmic.exe` is a native binary on Windows systems that can, amongst other malicious uses, execute other binaries. For most of the AVs, evasion is achieved by invoking a simple Powershell reverse shell from `Wmic.exe`. For other AVs, evasion was achieved by executing a shell over UDP using Powercat, instead of TCP. Further obfuscation was necessary to bypass AMSI and Windows Defender, for which Invoke-Obfuscation [6] was also used.

Rundll32.exe. We use Koadic [17], a tool for post-exploitation, for testing the viability and detectability of `Rundll32.exe`. Koadic provides several command-line parameters to provide to the `Rundll32.exe` binary. Koadic's evasive exploitation leverages the fact that `Rundll32.exe`, when provided with the '*-javascript*' parameter, allows the execution of Javascript code. Koadic is sufficient to achieve a bypass for most AV engines, for others JS-Rat-Py [22], a similar remote access tool, suffices.

Some AV engines were able to identify the execution of Javascript by `Rundll32.exe` as malicious, spawning a network connection. To bypass these, we generated a malicious *.dll* with MSFvenom, or in particularly evasive cases, we generated and compiled a reverse shell *.dll* from C++ code.

Regsvr32.exe. `Regsvr32.exe` binary can be used to execute remote *.sct* files in the *squiblydoo* [52] attack. This attack enables the execution of remotely hosted *.sct* files, containing Javascript code that can execute arbitrary commands such as creating ActiveXObjects. For our repeated experiments, as the squibblydoo attack has been patched by Microsoft, we used a bypass that still allows this attack to be conducted, described by Oddvar Moe [47].

For a number of the bypasses, Koadic was used. This tool allows a reverse shell to be executed in the form of JavaScript code. Where Koadic was detected, JS-Rat-Py's Regsvr32 component was not. For others, we used custom code within an *.sct* file.

Bitsadmin.exe. `Bitsadmin.exe` was used to create Background Intelligent Transfer Service jobs. It is typically used for the transfer of files between different machines. However it has undocumented functionality allowing the execution of binaries or arbitrary commands. It can execute malicious files and bypass application whitelisting.

`Bitsadmin.exe` was used to create a Powershell reverse shell, by creating a job and resuming it. This Powershell reverse shell was spawned by either calling Powershell directly, or performing this via several layers of obfuscation using an *.hta* file. This *.hta* file calls Powershell via a Visual Basic script within that *.hta* file called by `Mshta.exe`.