# Tools and Techniques for Malware Detection and Analysis

Sajedul Talukder
Department of Mathematics and Computer Science
Edinboro University, PA, USA
stalukder@edinboro.edu



***Abstract*—One of the major and serious threats that the Internet faces today is the vast amounts of data and files which need to be evaluated for potential malicious intent. Malicious software, often referred to as a malware that are designed by attackers are polymorphic and metamorphic in nature which have the capability to change their code as they spread. Moreover, the diversity and volume of their variants severely undermine the effectiveness of traditional defenses which typically use signature based techniques and are unable to detect the previously unknown malicious executables. The variants of malware families share typical behavioral patterns reflecting their origin and purpose. The behavioral patterns obtained either statically or dynamically can be exploited to detect and classify unknown malware into their known families using machine learning techniques. This survey paper provides an overview of techniques and tools for detecting and analyzing the malware.**

## I. Introduction

Malware, short for malicious software, is any software used to disrupt computer operation, gather sensitive information, or gain access to private computer systems. Malware is defined by its malicious intent, acting against the requirements of the computer user, and does not include software that causes unintentional harm due to some deficiency. The term badware is sometimes used, and applied to both true (malicious) malware and unintentionally harmful software. These are intended to gain access to computer systems and network resources, disturb computer operations, and gather personal information without taking the consent of systems owner, thus creating a menace to the availability of the internet, integrity of its hosts, and the privacy of its users. Spreading of malware has affected everyday life, from e-governance [1] to social networks [2], from digital automation [3] spreading up to mobile networks [4]. Malware come in wide range of variations like Virus, Worm, Trojan-horse, Rootkit, Backdoor, Botnet, Spyware, Adware etc. These classes of malware are not mutually exclusive meaning thereby that a particular malware may reveal the characteristics of multiple classes at the same time. In order to evade detection, malware authors introduce polymorphism to the malicious components. This means that malicious files belonging to the same malware "family", with the same forms of malicious behavior, are constantly modified and/or obfuscated using various tactics, such that they look like many different files. Malware is one of the most terrible and major security threats facing the Internet today. According to a survey [5], conducted by Symantec in February 2019, 47% of the organizations experienced malware security incidents/network breaches in the past one year, as depicted in figure 1 and 2. The malware are continuously growing in volume (growing threat landscape), variety (innovative malicious methods) and velocity (fluidity of threats). These are evolving, becoming more sophisticated and using new ways to target computers and mobile devices. McAfee [6] catalogs over 100,000 new malware samples every day means about 69 new threats every minute or about one threat per second. With the increase in readily available and sophisticated tools, the new generation cyber threats/attacks are becoming more targeted, persistent and unknown. The advanced malware are targeted, unknown, stealthy, personalized and zero day as compared to the traditional malware which were broad, known, open and one time. Once inside, they hide, replicate and disable host protections. After getting installed, they call their command and control servers for further instructions, which could be to steal data, infect other machines, and allow reconnaissance. Attackers exploit vulnerabilities in web services, browsers and operating systems, or use social engineering techniques to make users run the malicious code in order to spread malware. Malware authors use obfuscation techniques [7] like dead code insertion, register reassignment, subroutine reordering, instruction substitution, code transposition, and code integration to evade detection by traditional defenses like firewalls, antivirus and gateways which typically use signature based techniques and are unable to detect the previously unseen malicious executables. Commercial antivirus vendors are not able to offer immediate protection for zero day malware as they need to analyze these to create their signatures. To overcome the limitation of signature based methods, malware analysis techniques are being followed, which can be either static or dynamic. The malware analysis techniques help the analysts to understand the risks and intentions associated with a malicious code sample. The insight so obtained can be used to react to new trends in malware development or take preventive measures to cope with the threats coming in future. Features derived from analysis of malware can be used to group unknown malware and classify them into their existing families.

This paper presents a review of techniques/approaches and tools for detecting and analyzing the malware executables. There has been some study performed on comparison of static, dynamic, and hybrid analysis for malware detection [8], where as some researchers tried to bridge the static/dynamic gap [9]. Mobile technology in healthcare has also been target of malware [10]. Few recent studies have been done on static and dynamic analysis of Android malware [11], detection using permission [12]–[14], based on system call sequences and LSTM [15].

Many studies use static analysis for malware detection using exact decompilation [16], similarity testing framework [17], based on register contents [18], using two dimensional binary program features [19], subroutine based detection [20], statistics of assembly instructions [21], file relation graphs [22], de-anonymizing programmers via code stylometry [23], based upon a wavelet package technique [24], analysis and comparison of disassemblers for opcode [25]

The studies that use dynamic analysis perform synthesis the semantics of obfuscated code [7], multi-hypothesis testing [26], analyzing quantitative data flow graph metrics [27], using simplified data dependent api call graph [28], downloader graph analytics [29], access behavior [30], [31], APIs in initial behavior [32], log based crowdsourcing analysis [33]

There have been many studies on the detection and analysis of malware using machine learning that study fine-grained features [34], deep learning [35], [36], dynamic features [37], static features [38], concept drift [39], predicting signatures [40], hybrid framework [41], malware metadata [42], reverse engineering of large datasets of binaries [43].

**Our Contributions**. This paper presents the following contributions:

- **Techniques and tools for detecting and analyzing the malware**. This paper provides the first comprehensive survey on techniques and tools for detecting and analyzing the malware. There have been a numerous survey in the area of malware detection specific to machine learning, android and a few survey on static and dynamic analysis. However, none of the work addresses the techniques and available tools.
- **State-of-the-art Survey**. This paper reveals that the most existing surveys in this area are either outdated [44] or fail to provide a holistic view of the problem, since they usually focus on a specific subset of the standard [45].
- **List of comprehensive tools**. This paper presents a novel overview about the list of comprehensive tools available for malware detection, memory forensics, packet analysis, scanners/sandboxes, reverse engineering, debugging, and website analysis. Further, it differentiates the malware analysis tools based on specific domain and approach.
- **Guide for malware analysts**. Finally, it is realized that the contribution claimed in this paper will help, guide and assist researchers and malware analysts on getting appropriate tools for their domain specific analysis.

The rest of the paper is organized as follows. Section II describes the different types of malware. Section III describes the ways of malware analysis. Section IV discusses the malware analysis tools. Finally, Section V concludes the paper with a highlight on the scope of future work.

## II. Different Types of Malware

With so many different types of malware and the vast range of malicious software programs within each type its important that every malware item can be unambiguously classified and easily distinguished from other malicious programs. The term malware includes viruses, worms, Trojan Horses, rootkits, spyware, keyloggers and more. To get an overview of the

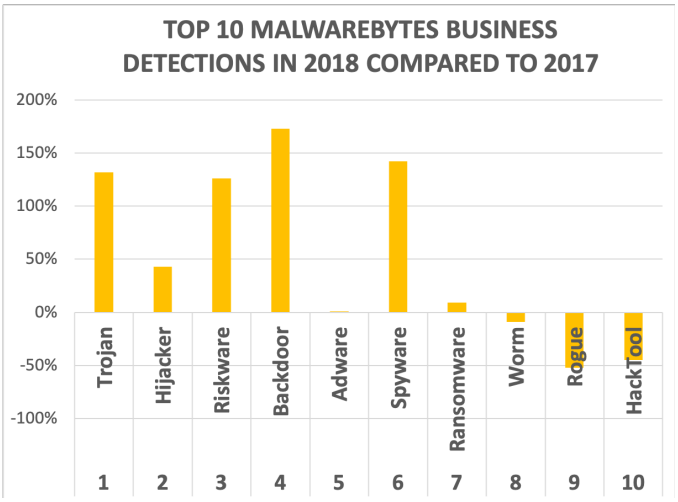



Fig. 1. Top 10 malwarebytes business detections in 2018 compared to 2017 [46]

difference between all these types of threats and the way they work, it makes sense to divide them into groups:

### A. Viruses and worms: the contagious threat

Viruses and worms both are designed to spread without the user's knowledge. A computer virus is a small program written to alter the way a computer operates, without the permission or knowledge of the user. A virus must meet two criteria:

1) It must execute itself. It will often place its own code in the path of execution of another program.
2) It must replicate itself. For example, it may replace other executable files with a copy of the virus infected file.

Viruses can infect desktop computers and network servers alike. Some viruses are programmed to damage the computer by damaging programs, deleting files, or reformatting the hard disk. Others are not designed to do any damage, but simply to replicate themselves and make their presence known by presenting text, video, and audio messages. Even these benign viruses can create problems for the computer user. They typically take up computer memory used by legitimate programs. As a result, they often cause erratic behavior and can result in system crashes. In addition, many viruses are bug-ridden, and these bugs may lead to system crashes and data loss.

Computer worms, on the other hand, spread across the internet by replicating itself on computers via their network. Both viruses and worms can carry a so-called "payload", malicious code designed to do damage. Worms are programs that replicate themselves from system to system without the use of a host file. This is in contrast to viruses, which requires the spreading of an infected host file. Although worms generally exist inside of other files, often Word or Excel documents, there is a difference between how worms and viruses use the host file. Usually the worm will release a document that already has the "worm" macro inside the document. The entire document will travel from computer to computer, so the entire document should be considered the worm. PrettyPark is a particularly prevalent example.

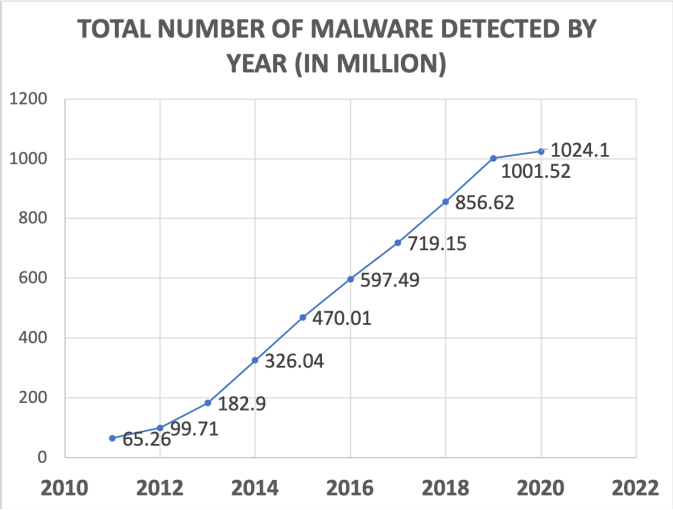



Fig. 2. Total number of malware detected by year (in million) [47]

### B. Trojans and Rootkits: the masked threat

Trojans and rootkits are grouped together as they both seek to conceal attacks on computers. Trojan Horses are malignant pieces of software pretending to be benign applications. Users therefore download them thinking they will get a useful piece of software and instead end up with a malware infected computer. A Trojan horse, or Trojan, in computing is a generally non-self-replicating type of malware program containing malicious code that, when executed, carries out actions determined by the nature of the Trojan, typically causing loss or theft of data, and possible system harm. The term is derived from the story of the wooden horse used to trick defenders of Troy into taking concealed warriors into their city in ancient Anatolia, because computer Trojans often employ a form of social engineering.

Rootkits are different. They are a masking technique for malware, but do not contain damaging software. Rootkit techniques were invented by virus writers to conceal malware, so it could go unnoticed by antivirus detection and removal programs. A rootkit is a stealthy type of software, typically malicious, designed to hide the existence of certain processes or programs from normal methods of detection and enable continued privileged access to a computer. The term rootkit is a concatenation of "root" (the traditional name of the privileged account on Unix operating systems) and the word "kit" (which refers to the software components that implement the tool).

### C. Spyware and keyloggers: the financial threat

Spyware and keyloggers are malware used in malicious attacks like identity theft, phishing and social engineering, threats designed to steal money from unknowing computer users, businesses and banks. Spyware is a type of malicious software (also called "malware") that scammers try to install on your computer. As the name suggests, spyware programs allow people to spy on what you are doing on your computer: the websites you visit, the files you use and the details you store on your PC.

Key-loggers are a particular type of spyware. Key-loggers secretly record what keys you press on your keyboard and sends this data back to the scammer over the internet. Scammers use these programs to steal passwords such as online banking passwords. They may also use spyware to steal other personal information from you such as documents that you have stored on your computer. Scammers use a wide range of tricks to get their spyware and key-loggers loaded on to your computer. This usually involves tricking you into clicking on a link in a spam email they have sent, or visiting a website that they have set up solely to infect people's computers. Other sources of spyware and key-loggers are free games or music that you can download from the internet. When they are delivered in this way, they are sometimes called "Trojans"–a file that claims to be for some harmless purpose so it can get under your guard, but in fact contains a nasty surprise.

## III. Malware Analysis

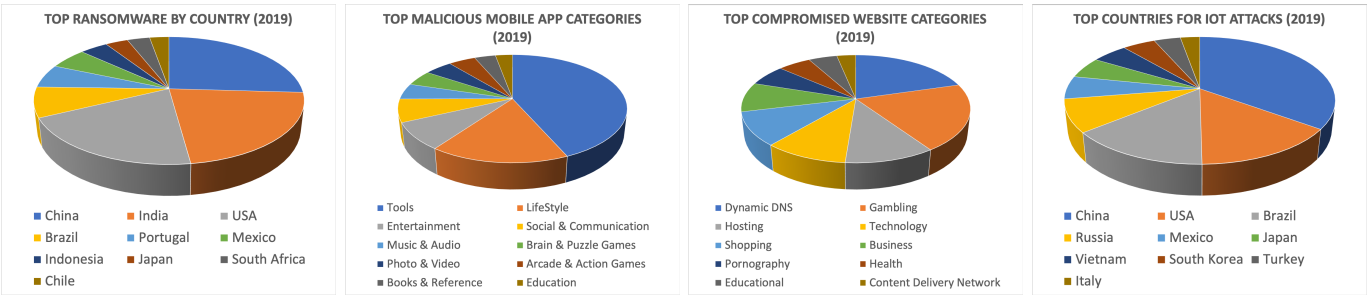
Before creating the signatures for newly arrived malware, these are required to be analyzed so as to understand the associated risks and intentions. The malicious program and its capabilities can be observed either by examining its code or by executing it in a safe environment.

### A. Static analysis

Analyzing malicious software without executing it is called static analysis. The detection patterns used in static analysis include string signature, byte-sequence n-grams, syntactic library call, control flow graph and opcode (operational code) frequency distribution etc. The executable has to be unpacked and decrypted before doing static analysis. The disassembler/debugger and memory dumper tools can be used to reverse com pile windows executables. Disassemble/Debugger tools like IDA Pro and OllyDbg displays the malware's code as Intel X86 assembly instructions, which provide a lot of insight into what the malware is doing and provide patterns to identify the attackers. Memory dumper tools like LordPE [9] and OllyDump [10] are used to obtain protected code located in the system's memory and dump it to a file. This is a useful technique to analyze packed executables which are difficult to disassemble. Binary obfuscation techniques, which transform the malware binaries into self-compressed and uniquely structured binary files, are designed to resist reverse engineering and thus make the static analysis very expensive and unreliable. Moreover, when utilizing binary executables (obtained by compiling source code) for static analysis, the information like size of data structures or variables gets lost thereby complicating the malware code analysis [11]. The evolving evasion techniques being used by malware writers to thwart static analysis led to the development of dynamic analysis. Moser et al. [12], explored the drawbacks of static analysis methodology. In their work, they introduced a scheme based on code obfuscation revealing the fact that the static analysis alone is not enough to detect or classify malware. Further, they proposed that dynamic analysis is a necessary complement to static analysis as it is less vulnerable to code obfuscation conversion.

### B. Dynamic analysis

Analyzing the behavior of a malicious code (interaction with the system) while it is being executed in a controlled environment (virtual machine, simulator, emulator, sandbox




(a) (b) (c) (d)

Fig. 3. (a) Top ransomware by country (2019) (b) Top malicious mobile app categories (2019) (c) Top compromised website categories (2019) (d) Top countries for IoT attacks (2019) [5]

etc.) is called dynamic analysis. Before executing the malware sample, the appropriate monitoring tools like Process Monitor [13] and Capture BAT [14] (for file system and registry monitoring), Process Explorer [15] and Process Hackerreplace [16] (for process monitoring), Wireshark [17] (for network monitoring) and Regshot [18] (for system change detection) are installed and activated. Various techniques that can be applied to perform dynamic analysis include function call monitoring, function parameter analysis, information flow tracking, instruction traces and autostart extensibility points etc. [11]. Dynamic analysis is more effective as compared to static analysis and does not require the executable to be disassembled. It discloses the malware's natural behavior which is more resilient to static analysis. However, it is time intensive and resource consuming, thus elevating the scalability issues. The virtual environment in which malware are executed is different from the real one and the malware may perform in different ways resulting in artificial behavior rather than the exact one. In addition to this, sometimes the malware behavior is triggered only under certain conditions (on specific system date or via a specific command) and can't be detected in virtual environment. Several online automated tools exist for dynamic analysis of malware, e.g. Norman Sandbox [19], CWSandbox [20], Anubis [21] and TTAnalyzer [22], Ether [23] and ThreatExpert [24]. The analysis reports generated by these tools give in-depth understanding of the malware behavior and valuable insight into the actions performed by them. The analysis system is required to have an appropriate representation for malware, which are then used for classification either based on similarity measure or feature vectors. However a large number of new malware samples arriving at anti-virus vendors every day requires an automated approach so as to limit the number of samples that require close human analysis. Several Artificial Intelligence techniques, particularly machine-learning based techniques have been used in the literature for automated malware analysis and classification.

## IV. Malware Analysis Tools

Analysts use tools for analyzing malware to protect and predict future attacks, and share knowledge among themselves. open source tools are often the first choice to carry out such actions. It's no secret that distributing malware is a big business and the fast-growing malware epidemic will only grow in ability and efficiency in the years to come. Using open source malware analysis tools, researchers will check, identify and log different variants of malicious triggers when analyzing the life-cycle of attack. As malware trading forums are proliferating on the dark web, the crypters, botnets and zero-days needed to carry out powerful attacks have become easier than ever to get. With the growth of complexity of malware variants, the job of understanding and benchmarking the specific type have become harder. It's the job of security researchers and analysts to find out the right tool to analyze each specific type of attack. We now present some open source malware analysis tools that can help the researchers and security engineers.

### A. Open Source Malware Analysis Tools

**Google Rapid Response (GRR).** The GRR platform is an incident response system developed by security researchers at Google, identifying common malware footprints workstations focused on remote live forensics. This consists of an application that is installed on the target system to communicate with the agent and a server infrastructure. GRR is a python client (agent) that is installed on target systems, and python server infrastructure that can manage and talk to clients. Once both the server side and the agent are deployed they can become GRR clients and start receiving messages from the servers. Then the incident response staff on the host computer will perform various technical operations, such as reviewing the memory, looking for different settings and handling software choices. GRR has been designed to run on a scale so analysts can easily capture and process data from large numbers of computers. GRR's goal is to support forensics and investigations in a simple, flexible way that allows investigators to rapidly triage incidents and conduct remote analysis.

**REMnux.** REMnux is a free Linux toolkit designed to assist malware analysts with malware reverse engineering. This seeks to make this easy for forensic investigators and accident witnesses to continue using the variety of free-to-use software that can analyze ransomware, although it may be difficult to locate or set up. This Linux toolkit has been developed as a one-stop shop for researchers searching for examples of reverse engineering malware. REMnux is focused on Ubuntu and integrates several resources into one for quickly analyzing malware based on Windows and Linux. The cornerstone of the project is the Ubuntu based REMnux Linux system. This lightweight distro provides various resources to detect Windows and Linux ransomware, review browser-based vulnerabilities such as obfuscated JavaScript, investigate unusual text files, and uninstall other harmful objects. The distro can be used by investigators to intercept suspicious network traffic in

an isolated laboratory. It helps researchers investigate browser-based malware, perform forensics on memory, analyze multiple samples of malware, extract and decode suspicious items, etc.

**Cuckoo Sandbox.** Created by a team of volunteers during the Google Summer of Code initiative back in 2010, it is an open source framework that automates malicious file analysis for Windows, OS X, Linux and Android and offers comprehensive and practical input on how each presented file operates in isolated environments. And since it is open source software, developers are constantly writing plugins that provide enhanced features. Cuckoo is used by malware detection and security firms to help ease the strain of manually wading through troves of potentially malicious data. The modular design allows the recording and analysis phases simple to configure. In recent years, it has, understandably, become one of the most commonly used open source tools. In 2012, Cuckoo published Malwr, a sandbox-as - a-service that allows users to use their collected data through an easy-to-use GUI. The goal was to act as an option for users who can't handle Cuckoo properly but still want to exploit their intellect.

**Zeek.** The Zeek Network Security Monitor (formerly Bro) is a versatile network dependent analytics system that transforms network traffic into events to cause scripts. It is comparable to an IDS (intrusion detection system) in that it gives users a bird's-eye view of their network activity, using both signature-based (looks for rules or trends of documented malicious traffic) and anomaly-based monitoring (looks for unusual activity). Nevertheless, its features go far beyond those of conventional IDS that can be used to conduct investigations in forensics, network monitoring and interface research. Although focussing on tracking network security, Zeek also provides a comprehensive forum for more general analysis of network traffic. Well grounded in more than 20 years of research, Zeek has since its inception succeeded in bridging the traditional gap between academia and operations.

**Yara Rules.** Another open source malware identification tool that can identify samples of malware based on textual or binary trends once they are tested in Cuckoo. Investigators use Yara to compose pattern-based definitions of the malware families. YARA stands for "Yet Another Recursive Acronym" as the descriptions are called rules. This helps researchers to identify and categorize apparently similar malware types and can be adapted for use inside Cuckoo. IBM calls Yara the "pattern matching Swiss army knife" of the malware researcher and can be used on both Windows and Linux computers. Yara's creators released a new service still in alpha called YaraRules Analyzer, that lets users analyze files in the cloud using full rulesets. This ensures that users are always analyzing samples against the most recent ruleset version and frees them from needing to install Yara locally. Yara rules have been added to many Endpoint Detection and Response framework to help them identify the malware samples they encounter, classify them and share their findings with clients and the community later.

Table I and II list a number of tools available for malware detection, memory forensics, packet analysis, scanners/sandboxes, reverse engineering, debugging, and website analysis.

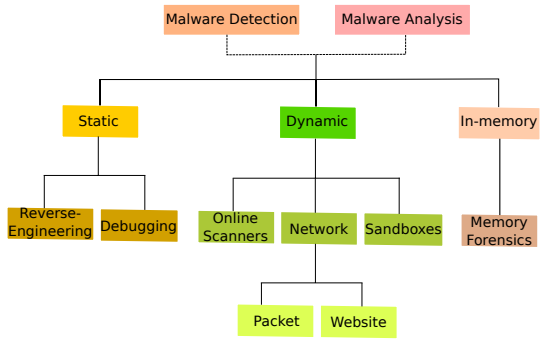



Fig. 4. Malware Detection and Analysis Tools

### *B. Mobile Malware Analysis Tools*

**APKTool.** A tool for reverse engineering 3rd party, closed, binary Android apps. By making several changes, it can decode resources to almost original form and restore them. It also makes it easier to deal with a device owing to the project such as file creation and completion of some repetitive tasks such as creating the apk etc. We can decode APK resources to almost original form with the help of Apktool; we can modify the source code on the fly and rebuild the decoded resources back into APK. Its project-like structure makes working with them easy. Apktool can decode APK resources (resources.arsc, classes.dex and XMLs), rebuild decoded resources back to binary APK, organize and handle APKs that depend on framework resources along with automating the repetitive tasks.

**Smali.** Smali / baksmali is a dex format assembler / disassembler used by Dalvik, a Java VM implementation for Android. The syntax is loosely based on the syntax of Jasmin / dedexer, and follows the complete dex format features (annotations, debug data, line details, etc.). In addition, code created by the baksmali is often considered to be written in the Smali language. Baksmali is a dex Bytecode software disassembler. The terms "Smali" and "Baksmali" are just the corresponding Icelandic versions of "assembler" and "disassembler." It wasn't an easy task to debug smali code before, but recently a wonderful plugin was launched for Intellij IDEA / Android Studio - Smalidea.

**Dex2Jar.** Dex2Jar is a free tool for dealing with the files Android ".dex" and Java ".class." Android programs are assembled into".dex" (Dalvik Executable) scripts, which in effect are zipped onto the computer into a single.apk file. Android will automatically create the ".dex" folders, by converting the compiled applications written in the Java. Dex2Jar reads the dex instruction to dex-ir format and can convert to ASM format. It can also be used to perform some basic deobfuscation. Dex2Jar's core feature is converting an APK classes.dex file to classes.jar, or vice versa. So, using any Java decompiler, it is possible to view the source code of an Android application, and it is fully legible. Here we get files from.class and not the real Java source code that the program author wrote.

**Mobile-Sandbox.** Mobile-Sandbox provides static and dynamic malware analysis for Android OS smartphones. The system is designed to automatically evaluate Android software in two novel ways: (1) by integrating static and dynamic analysis, i.e., static analysis findings are used to direct dynamic

| Detection Tools | | Online scanners and sandboxes | |
|---|---|---|---|
| AnalyzePE | Wrapper for a variety of tools for reporting on Windows PE files. | AndroTotal | Online analysis of APKs against multiple mobile antivirus apps |
| chkrootkit | Linux rootkit detector. | APK Analyzer | Dynamic analysis of APKs |
| Detect-It-Easy | A program for determining types of files. | AVCaesar | Online scanner and malware repository |
| hashdeep | Compute digest hashes with a variety of algorithms. | Cryptam | Analyze suspicious office documents |
| Loki | Host based scanner for IOCs. | Cuckoo Sandbox | Open source sandbox and automated analysis system |
| MASTIFF | Static analysis framework. | Comodo Valkyrie | File verdict system that conducts several analysis using run-time behavior and hundreds of features from a file |
| MultiScanner | Modular file scanning/analysis framework | DeepViz | Multi-format file analyzer with machine-learning classification |
| nsrllookup | A tool for looking up hashes in NIST's National Software Reference Library database. | detux | A sandbox developed to do traffic analysis of Linux malware and capturing IOCs |
| PEV | A multiplatform toolkit to work with PE files, providing feature-rich tools for proper analysis of suspicious binaries. | Document Analyzer | Analysis of DOC and PDF files |
| Rootkit Hunter | Detect Linux rootkits. | DRAKVUF | Dynamic malware analysis system. |
| totalhash.py | Python script for searching in TotalHash.cymru.com database. | File Analyzer | Free dynamic analysis of PE files |
| TrID | File identifier. | firmware.re | Unpacks, scans and analyzes firmware packages |
| YARA | Pattern matching tool for analysts. | Hybrid Analysis | Online malware analysis tool |
| **Memory Forensics Tools** | | IRMA | An asynchronous and customizable analysis platform for suspicious files |
| DAMM | Differential Analysis of Malware in Memory, built on Volatility | Joe Sandbox | Deep malware analysis. |
| evolve | Web interface for the Volatility Memory Forensics Framework | Jotti | Online AV scanner |
| FindAES | Find AES encryption keys in memory | Limon | Sandbox for Analyzing Linux Malware |
| Muninn | A script to automate portions of analysis using Volatility, and create a readable report | Malheur | Automatic sandboxed analysis of malware behavior |
| Rekall | Memory analysis framework (from a Volatility fork). | Malwr | Free analysis with an online Cuckoo Sandbox instance |
| TotalRecall | Script based on Volatility for automating various malware analysis tasks | MASTIFF Online | Online static malware analysis |
| Volatility | Advanced memory forensics framework. | Metadefender.com | Scan a file, hash or IP address for malware |
| WinDbg | Kernel debugger for Windows systems | NoDistribute | Scan files with over 35 anti-viruses. |
| **Packet Analysis Tools** | | NVISO ApkScan | Dynamic analysis of APKs |
| Network Miner | A Network Forensic Analysis Tool (NFAT) for Windows | PDF Examiner | Analyse suspicious PDF files |
| NetworkTotal | Online analysis of .pcap files to detect viruses, worms, trojans and malware. | SEE | "Sandboxed Execution Environment", a framework for building test automation in secured environments |
| PacketTotal | Online engine for analyzing .pcap files and visualizing the network traffic within, useful for malware analysis and incident response. My review | URL Analyzer | Dynamic analysis of URL files |
| Wireshark | Widely-used network protocol analyzer. | VirusTotal | Online analysis of malware samples and URLs |

TABLE I. TOOLS AVAILABLE FOR MALWARE DETECTION, MEMORY FORENSICS, PACKET ANALYSIS, SCANNERS AND SANDBOXES.

| Reverse Engineering and Debugging Tools | | strace | Dynamic analysis tool for Linux executables |
|---|---|---|---|
| angr | Platform-agnostic binary analysis framework | Triton | A dynamic binary analysis (DBA) framework |
| bamfdetect | Identifies and extracts information from bots and malware | Udis86 | Disassembler library and tools |
| BARF | Open source multiplatform Binary Analysis and Reverse engineering Framework. | Vivisect | Python tool for malware analysis |
| binnavi | Binary analysis IDE for reverse engineering | X64dbg | Debugger for windows |
| Capstone | Disassembly framework for binary analysis and reversing | **Website Analysis Tools** | |
| codebro | Web based code browser with basic code analysis. | Desenmascara.me | Tool to retrieve metadata from websites |
| dnSpy | .NET assembly editor, decompiler and debugger | Dig | Online dig and other network tools |
| Evan's Debugger (EDB) | Modular debugger with a Qt GUI | dnstwist | Domain name permutation engine for detecting typo squatting, phishing and corporate espionage |
| Fibratus | Windows kernel exploration and tracing tool | Firebug | Firefox extension for web development. |
| GDB | The GNU debugger | IPinfo | Gather information about an IP or domain by searching online resources |
| GEF | GDB Enhanced Features, for exploiters and reverse engineers | Java Decompiler | Decompile and inspect Java apps |
| hackers-grep | Uility to search for strings in PE executables | Java IDX Parser | Parses Java IDX cache files |
| IDA Pro | Windows disassembler and debugger | JSDetox | JavaScript malware analysis tool |
| Immunity Debugger | Debugger for malware analysis | jsunpack-n | Javascript unpacker that emulates browser functionality |
| ltrace | Dynamic analysis tool for Linux executables | Krakatau | Java decompiler, assembler, and disassembler |
| objdump | Static analysis tool for Linux binaries | Machinae | OSINT tool for gathering information about URLs, IPs, or hashes |
| OllyDbg | Debugger for Windows executables | mailchecker | Cross-language temporary email detection library |
| PANDA | Platform for Architecture-Neutral Dynamic Analysis | Malzilla | Analyze malicious web pages. |
| PEDA | Python Exploit Development Assistance for GDB | PunkSpider | Web application vulnerability search engine.My review |
| pestudio | Static analysis tool for Windows executables | RABCDAsm | ActionScript Bytecode Disassembler |
| plasma | Interactive disassembler for x86/ARM/MIPS | SenderBase | Search for IP, domain or network owner |
| PPEE (puppy) | PE file inspector. | Spidermonkey | Mozilla's JavaScript engine, for debugging malicious JS |
| Process Monitor | Advanced monitoring tool for Windows programs | Sucuri SiteCheck | Website Malware and Security Scanner |
| Pyew | Python tool for malware analysis | swftools | Adobe Flash decompiler. |
| Rdare2 | Reverse engineering framework | TekDefense Automator | OSINT tool for gathering information about URLs, IPs, or hashes |
| ROPMEMU | Framework to analyze, dissect and decompile complex code-reuse attacks | xxxswf | Analysis tool for Flash files |
| SMRT | Sublime Malware Research Tool, a plugin for Sublime Text 3 focused on malware analyis. | ZScalar Zulu | Zulu URL Risk Analyzer |

TABLE II. TOOLS AVAILABLE FOR REVERSE ENGINEERING, DEBUGGING, AND WEBSITE ANALYSIS.

analysis and expand coverage of executed code, and (2) by using different logging methods for native API calls [48]. It can evaluate the application with different modules within the static analysis component to get a summary of the program. To achieve this, it uses the VirusTotal service to perform several anti-virus scans, parse the manifest file and finally decompile the application to better identify suspect code. Within the dynamic analysis, it can run the application in an emulator and log every application operation, i.e. it logs both actions performed in the Java Virtual Machine Dalvik and actions performed in native libraries that may be bundled with the application.

### C. Other Analysis Tools

**Wireshark.** Wireshark, a network monitoring application once known as Ethereal, records packets and shows them in human-readable format in real time. It intercepts traffic and transforms the binary data to a readable format for users. Wireshark includes filters, color coding, and other features which allow individuals to dig deep into network traffic and inspect individual packets. It is the leading network traffic analyzer in the world, and an essential tool for any skilled security or device administrator. This free software allows people to track network traffic in real time, and is often the best tool on any network for troubleshooting problems. Common issues that Wireshark can deal with troubleshooting include lost messages, latency issues and malicious network operation. It enables network data to be held under a microscope and offers resources for filtering and digging into that information, zooming into the root cause of the issue. It is used by management to detect defective network equipment that lose packets, latency problems caused by machines transmitting traffic around the world and data exfiltration or even intrusion attempts against any entity.

**VirusTotal.** Virustotal is a service that analyzes suspicious files and URLs and helps to detect viruses, worms, trojans and all kinds of malware detected by antivirus engines quickly. In addition to a variety of methods for removing signals from the studied material, VirusTotal inspects products with over 70 antivirus scanners and URL / domain blacklisting services. Every person can use their browser to pick a file from their device, and submit it to VirusTotal. VirusTotal offers various methods for uploading data, including the default public web portal, desktop uploaders, browser extension and a programmatic API. The web interface has the greatest scanning priority among the forms of application which are available to the public. The specifications can be made using the HTTP-based public API in any programming language. It also offers a variety of other functions, including the VirusTotal Community: a network that allows users to report on files and URLs and exchange comments with each other. This can be helpful in detecting malicious content and also in finding false positives – regular and harmless objects identified as dangerous by one or more scanners.

**SysAnalyzer.** SysAnalyzer is an open source tool developed to provide an interactive resource for malware researchers to easily compile, analyze, and monitor the behavior that a binary performed when operating on the network. It is an interactive framework for the malcode run time analysis that tracks different aspects of device and method states. SysAnalyzer was designed to allow analysts to rapidly create a detailed report. SysAnalyzer's main components function off of comparing device snapshots over a given user time interval. Similar to a live monitoring system, the reason a snapshot method was used is to reduce the amount of data analysts need to wade through when performing their research. By using a snapshot method, audiences can easily display only the persistent changes that have been identified since the first run of the application.

**Malzilla.** Malzilla is a useful malware hunting tool for analyzing websites containing malicious code. Web pages that contain exploits often use a sequence of redirects and obfuscated code to make it difficult for someone to track them. This allows users to access websites and obtain all of their source code, such as wget, without visiting the site and potentially damaging their device. This program has the option of switching user agent and picking the user's own referrer. This shows the full list of webpages for browsers and all the headers for HTTP. It also has proxy features, complex decoders and, most notably, javascript code deobfuscation, all in one program.

## V. Conclusion

In this paper we had surveyed an overview of techniques and tools for detecting and analyzing the malware. In particular, a light has been thrown on various tools available for malware detection, memory forensics, packet analysis, scanners/sandboxes, reverse engineering, debugging, and website analysis. Since most of the existing surveys usually focus on a specific subset of the standard, this paper provides a thorough study of tools for detecting and analyzing malware with a clear understanding of domain specific analysis.

## References


[1] S. K. Talukder, M. I. I. Sakib, and M. M. Rahman, "Model for e-government in bangladesh: A unique id based approach," in *2014 International Conference on Informatics, Electronics Vision (ICIEV)*, May 2014, pp. 1–6.

[2] S. Talukder and B. Carbunar, "When friend becomes abuser: Evidence of friend abuse in facebook," in *Proceedings of the 9th ACM Conference on Web Science*, ser. WebSci '17. New York, NY, USA: ACM, June 2017. [Online]. Available: http://doi.acm.org/10.1145/3091478.3098869

[3] S. K. Talukder, M. I. I. Sakib, and M. M. Rahman, "Digital land management system: A new initiative for bangladesh," in *2014 International Conference on Electrical Engineering and Information Communication Technology*, April 2014, pp. 1–6.

[4] S. Talukder, I. I. Sakib, F. Hossen, Z. R. Talukder, and S. Hossain, "Attacks and defenses in mobile ip: Modeling with stochastic game petri net," in *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*. IEEE, 2017, pp. 18–23.

[5] S. Corporation, "Internet security threat report," Symantec, shorturl.at/aqKO7, 2019.

[6] M. Labs, "Mcafee labs threats reports," McAfee, shorturl.at/izEJU, 2019.

[7] T. Blazytko, M. Contag, C. Aschermann, and T. Holz, "Syntia: Synthesizing the semantics of obfuscated code," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 643–659.

[8] A. Damodaran, F. Di Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 1, pp. 1–12, 2017.

[9] B. Anderson, C. Storlie, and T. Lane, "Improving malware classification: bridging the static/dynamic gap," in *Proceedings of the 5th ACM workshop on Security and artificial intelligence*, 2012, pp. 3–14.

[10] S. Talukder, S. Witherspoon, K. Srivastava, and R. Thompson, "Mobile technology in healthcare environment: Security vulnerabilities and countermeasures," *arXiv preprint arXiv:1807.11086*, 2018.

[11] C. Raghuraman, S. Suresh, S. Shivshankar, and R. Chapaneri, "Static and dynamic malware analysis using machine learning," in *First International Conference on Sustainable Technologies for Computational Intelligence*. Springer, 2020, pp. 793–806.

[12] A. Arora, S. K. Peddoju, and M. Conti, "Permpair: Android malware detection using permission pairs," *IEEE Transactions on Information Forensics and Security*, 2019.

[13] M. Alazab, M. Alazab, A. Shalaginov, A. Mesleh, and A. Awajan, "Intelligent mobile malware detection using permission requests and api calls," *Future Generation Computer Systems*, 2020.

[14] S. Talukder and B. Carbunar, "Abusniff: Automatic detection and defenses against abusive facebook friends," in *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[15] X. Xiao, S. Zhang, F. Mercaldo, G. Hu, and A. K. Sangaiah, "Android malware detection based on system call sequences and lstm," *Multimedia Tools and Applications*, vol. 78, no. 4, pp. 3979–3999, 2019.

[16] E. Schulte, J. Ruchti, M. Noonan, D. Ciarletta, and A. Loginov, "Evolving exact decompilation," in *Workshop on Binary Analysis Research (BAR)*, 2018.

[17] J. Upchurch and X. Zhou, "Variant: a malware similarity testing framework," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2015, pp. 31–39.

[18] M. Ghiasi, A. Sami, and Z. Salehi, "Dynamic vsa: a framework for malware detection based on register contents," *Engineering Applications of Artificial Intelligence*, vol. 44, pp. 111–122, 2015.

[19] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2015, pp. 11–20.

[20] J. Sexton, C. Storlie, and B. Anderson, "Subroutine based detection of apt malware," *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 4, pp. 225–233, 2016.

[21] P. Khodamoradi, M. Fazlali, F. Mardukhi, and M. Nosrati, "Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms," in *2015 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS)*. IEEE, 2015, pp. 1–6.

[22] L. Chen, T. Li, M. Abdulhayoglu, and Y. Ye, "Intelligent malware detection based on file relation graphs," in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*. IEEE, 2015, pp. 85–92.

[23] A. Caliskan-Islam, R. Harang, A. Liu, A. Narayanan, C. Voss, F. Yamaguchi, and R. Greenstadt, "De-anonymizing programmers via code stylometry," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 255–270.

[24] B. Gu, Y. Fang, P. Jia, L. Liu, L. Zhang, and M. Wang, "A new static detection method of malicious document based on wavelet package analysis," in *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*. IEEE, 2015, pp. 333–336.

[25] M. Nar, A. G. Kakisim, M. N. Yavuz, and İ. Soğukpinar, "Analysis and comparison of disassemblers for opcode based malware analysis," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2019, pp. 17–22.

[26] P. Vadrevu and R. Perdisci, "Maxs: Scaling malware execution with sequential multi-hypothesis testing," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 771–782.

[27] T. Wüchner, M. Ochoa, and A. Pretschner, "Robust and effective malware detection through quantitative data flow graph metrics," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2015, pp. 98–118.

[28] A. A. E. Elhadi, M. A. Maarof, and B. Barry, "Improving the detection of malware behaviour using simplified data dependent api call graph," *International Journal of Security and Its Applications*, vol. 7, no. 5, pp. 29–42, 2013.

[29] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraș, "The dropper effect: Insights into malware distribution with downloader graph analytics," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1118–1129.

[30] A. Mohaisen, O. Alrawi, and M. Mohaisen, "Amal: High-fidelity, behavior-based automated malware analysis and classification," *computers & security*, vol. 52, pp. 251–266, 2015.

[31] W. Mao, Z. Cai, D. Towsley, and X. Guan, "Probabilistic inference on integrity for access behavior based malware detection," in *International Symposium on Recent Advances in Intrusion Detection*. Springer, 2015, pp. 155–176.

[32] N. Kawaguchi and K. Omote, "Malware function classification using apis in initial behavior," in *2015 10th Asia Joint Conference on Information Security*. IEEE, 2015, pp. 138–144.

[33] A. Raff, D. Peri, and A. Lotem, "System and methods for malware detection using log based crowdsourcing analysis," Aug. 27 2019, uS Patent 10,397,246.

[34] X. Jiang, B. Mao, J. Guan, and X. Huang, "Android malware detection using fine-grained features," *Scientific Programming*, vol. 2020, 2020.

[35] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Dl-droid: Deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, p. 101663, 2020.

[36] W. Wang, M. Zhao, and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 8, pp. 3035–3043, 2019.

[37] I. Sogukpinar, "Analysis and evaluation of dynamic feature-based malware detection methods," in *Innovative Security Solutions for Information Technology and Communications: 11th International Conference, SecITC 2018, Bucharest, Romania, November 8–9, 2018, Revised Selected Papers*, vol. 11359. Springer, 2019, p. 247.

[38] G. Laurenza, L. Aniello, R. Lazzeretti, and R. Baldoni, "Malware triage based on static features and public apt reports," in *International Conference on Cyber Security Cryptography and Machine Learning*. Springer, 2017, pp. 288–305.

[39] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 625–642.

[40] M. Howard, A. Pfeffer, M. Dalai, and M. Reposa, "Predicting signatures of future malware variants," in *2017 12th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2017, pp. 126–132.

[41] Z. Feng, S. Xiong, D. Cao, X. Deng, X. Wang, Y. Yang, X. Zhou, Y. Huang, and G. Wu, "Hrs: A hybrid framework for malware detection," in *Proceedings of the 2015 ACM International Workshop on International Workshop on Security and Privacy Analytics*, 2015, pp. 19–26.

[42] M. Asquith, "Extremely scalable storage and clustering of malware metadata," *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 2, pp. 49–58, 2016.

[43] M. Polino, A. Scorti, F. Maggi, and S. Zanero, "Jackdaw: Towards automatic reverse engineering of large datasets of binaries," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2015, pp. 121–143.

[44] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Comput. Surv.*, vol. 44, no. 2, Mar. 2008. [Online]. Available: https://doi.org/10.1145/2089125.2089126

[45] D. Uppal, V. Mehra, and V. Verma, "Basic survey on malware analysis, tools and techniques," *International Journal on Computational Sciences & Applications (IJCSA)*, vol. 4, no. 1, p. 103, 2014.

[46] M. Labs, "2019 state of malware," Malwarebytes, shorturl.at/bjtP8, 2019.

[47] I. Lapowsky, "Malware last 10 years," AV-TEST, shorturl.at/yzN01, 2020.

[48] M. Spreitzenbarth, F. Freiling, F. Echtler, T. Schreck, and J. Hoffmann, "Mobile-sandbox: having a deeper look into android applications," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1808–1815.