Eurographics Conference on Visualization (EuroVis) (2015)
R. Borgo, F. Ganovelli, and I. Viola (Editors)

STAR – State of The Art Report

A Survey of Visualization Systems for Malware Analysis

M. Wagner[1,2], F. Fischer[3], R. Luh[1], A. Haberson[1], A. Rind[1,2], D. A. Keim[3], and W. Aigner[1,2]

[1]St. Poelten University of Applied Sciences, Austria
[2]Vienna University of Technology, Austria
[3]University of Konstanz, Germany

Abstract

Due to the increasing threat from malicious software (malware), monitoring of vulnerable systems is becoming increasingly important. The need to log and analyze activity encompasses networks, individual computers, as well as mobile devices. While there are various automatic approaches and techniques available to detect, identify, or capture malware, the actual analysis of the ever-increasing number of suspicious samples is a time-consuming process for malware analysts. The use of visualization and highly interactive visual analytics systems can help to support this analysis process with respect to investigation, comparison, and summarization of malware samples. Currently, there is no survey available that reviews available visualization systems supporting this important and emerging field. We provide a systematic overview and categorization of malware visualization systems from the perspective of visual analytics. Additionally, we identify and evaluate data providers and commercial tools that produce meaningful input data for the reviewed malware visualization systems. This helps to reveal data types that are currently underrepresented, enabling new research opportunities in the visualization community.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces, I.3.8 [Computer Graphics]: Applications—, K.6.5 [Management Of Computing And Information Systems]: Security and Protection—Invasive software

1. Introduction

Malicious software, or malware, can be defined as "*any software that does something that causes harm to a user, computer, or network*" [SH12]. Examples include viruses, trojan horses, backdoors, worms, rootkits, scareware, or spyware. Malware analysis, in turn, is defined as "*the art of dissecting malware to understand how it works, how to identify it, and how to defeat or eliminate it*" [SH12]. For such an analysis to be effective, accurate detection mechanisms are needed [DKLT14]. These include classical approaches relying on binary signatures that represent certain static portions of a sample's code as well as various behavioral detection techniques relying on an accurate trace of e.g., functions executed by an application during run-time. The number of malicious programs, however, is growing at a tremendous rate. The sheer number of newly discovered malware variants poses a significant challenge to the security community. In the third quarter of 2014 alone, 20 million new samples were discovered [Pan14] which amounts to more than 150,000 pieces of malicious software that need to be triaged every day. What some argue to be a manageable annoyance for personal computer users has the potential to cause severe damage in high-availability environments or safety critical infrastructures.

Because of the overwhelming quantity of samples and the fact that manual analysis by domain experts is very cumbersome, automated data analysis methods are in dire need. In order to automate this process as much as possible, one feasible approach is to specify patterns of particular system call sequences and categorize them as being potentially harmful or harmless [DKLT14]. However, this process cannot be automated completely since domain experts need to be in the loop to identify, correct, and disambiguate intermediate results [WAR*14]. Lee et al. [LSKJ11] show that the use of visualization speeds up the malware detection process significantly. Large amounts of data, complex data analysis requirements, and the combination of automated data analysis with analytical reasoning by domain experts lends itself very well to the notion of visual analytics [TC05,KKEM10]. Visual analytics, "*the science of analytical reasoning facili-*

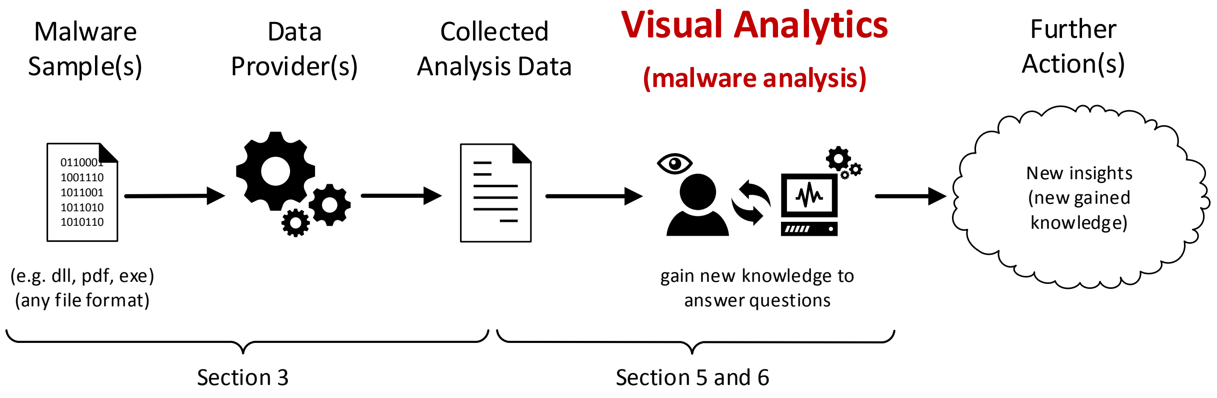


Figure 1: Data collection from malware samples and interactive analysis of these data using visual analytics methods are the main stages of malicious software analysis. Both stages of the process are covered in this survey.

tated by interactive visual interfaces" [TC05, p. 4], is a comparably young field of research. A major tenet of visual analytics states that analytical reasoning is not a routine activity that can be automated completely [Weg97]. Instead it depends heavily on analysts' initiative and domain experience. Furthermore, visual analytics involves automated analysis methods which computationally process large volumes of data and thus complement human cognition.

There are a number of approaches that utilize interactive visual methods for malware analysis. However, there is no survey available that reviews visualization tools for malicious software analysis in a comprehensive and systematic manner. To close this gap, we provide a systematic overview and categorization of the most commonly used visualization techniques for malware analysis.

The main objective of this work is to compare various malware analysis systems and to categorize them based on a number of criteria which are listed in Section 6. Based on the categorization and discussion of different tools, this survey provides a comprehensive overview of various, currently utilized visualization systems for malicious software analysis employed in different areas of malware analysis. Armed with this information, it will become significantly easier for researchers and analysts to identify new research areas and help them focus their efforts in the years to come. In addition to visualization solutions, this survey includes a section discussing and comparing a number of data providers that are part of different malware analysis suites and tools. These data providers are categorized by their main purpose and their provided functionality in terms of data collection (see Section 3.1). We also present challenges and opportunities for future research at the end of the paper.

The overall structure of this report is shown in Figure 1 which follows the general workflow of malware analysis. After discussing related work in Section 2 we focus on data providers (Section 3). These produce data from malware samples and form the basis for visual exploration. We describe our research method and the process of finding and selecting suitable tools in Section 4. Sections 5 and 6 present and compare the surveyed approaches and describe the criteria used for comparison. We conclude in Sections 7 and 8 with a discussion of our findings and present future research challenges in the field of malware visualization systems.

2. Related Work

Even though malware analysis is a prevailing challenge and a worthwhile application domain for visual analytics, we could not identify any academic work surveying this field from a visualization perspective. In the related area of network security, visualization is surveyed by Shiravi et al. [SSG12] who describe 38 different systems divided into 5 different groups of use cases. Like our work, they compare data sources and visualization techniques; however, data sources and analysis objectives differ from those relevant to malware analysis. Conti's book [Con07] covers certain aspects of malware analysis only from the perspective of network traffic analysis. Software Visualization [Die07] shares some data sources and techniques (static and dynamic) but has completely different analysis goals.

There is, however, general literature on *automated techniques* for malware detection and analysis as well as surveys for areas related to malware analysis: Siddiqui et al. [SWL08] provide a compact overview of 19 malware detection approaches using data mining on file features. They categorize them based on the included file properties, the analysis type, and the detection strategy. Complementarily, Egele et al. [ESKK12] survey 18 approaches for dynamic analysis of malware samples and compare them alongside emulation/instrumentation technologies, the granularity of recorded malware behavior, and obfuscation techniques. Furthermore, some of their systems support clustering or automatic report generation. Bazrafshan et al. [BHFH13] survey 22 approaches for heuristic malware detection and categorize them by the data source used. Idika and Mathur [IM07] survey malware detection approaches based on anomalies, specifications, or signatures. In general, the focus of aforementioned surveys is on providing data for subsequent analysis. Section 3 follows a similar approach.

The landscape of mobile malware was surveyed by Felt et al. [FFC*11], who summarized the characteristics of 46 malware samples for iOS, Android, and Symbian operating systems. Additionally, they discussed the effectiveness of preventive measures against such mobile malware. Finally, the topic of port scanning was surveyed by Bou-Harb et al. [BHDA14] and Bhuyan et al. [BBK11].

Ultimately, there is no detailed overview available in the field of visual analytics for malware analysis. Thus, we aim to fill this gap by providing an overview on the state-of-the-art of the available visual analytics approaches and their potential data providers.

3. Data Providers

In this paper, we define *data providers* as standalone tools or commercial suites that statically or dynamically analyze malware and return the collected information for further processing or analysis. Visualization tools use these data as primary input which makes the quality of the provided information paramount to preserving semantic expressiveness. Every *data provider* runs in an *analysis* environment and retrieves *base data* on a certain monitoring level. In the following we explain each term in detail and take a look at some of the most common tools and their analysis capabilities.

Data providers utilize static or dynamic analysis methods (sometimes both) to gather information about a potentially malicious piece of software. *Static analysis* describes techniques that do not require the sample under scrutiny to be actually executed. Depending on the depth of analysis a file may be checked for its basic properties (e.g., file type, checksum), easily extractable information (e.g., strings, DLL import information), or be fully disassembled [KM07]. The analysis environment plays a negligible role for static analyses – the analyst simply chooses a platform compatible with the tools of her choice.
Dynamic analysis goes a step further and executes the file on a host system. Various tools then monitor the execution and log relevant information into an execution trace. This ranges from simple file system operations to a full instruction trace captured through a debugger. The analysis environment is essential for the dynamic approach since the type of data logged depends on both the environment as well as the techniques used to capture system events. Both will be discussed in detail below.

Analysis environments are the foundation of the actual implementation of the respective malware analysis system. Depending on a data provider's capabilities and requirements, these environments may be physical machines, virtual machines, or emulated systems.

Physical machines are bare-metal computers that execute a sample directly in their preinstalled operating system (OS). While physical setups are unlikely to be detected by the malware, the potentially malicious sample is able to directly access the hardware it is running on (usually through a layer of abstraction provided by the OS). It is also important to keep in mind that reinstalling/resetting a physical machine is more time-consuming than resetting a virtualized or emulated environment. Data providers need to be run directly on the real OS using a local user account (usually one with administrative privileges) and therefore need to abide by the system's general rules.
Virtual machines (VMs) can be understood as isolated duplicate of a real machine [Gol74]. For classic VMs, a so-called virtual machine monitor (VMM) manages hardware access and represents this virtual copy to the executed software. This prevents a program from directly interacting with the real hardware but may complicate analysis of malware that utilizes VM evasion techniques to prevent virtualized execution. Like physical machines, VMs are limited to the same architecture as the host machine; the choice of OS, however, is not limited to the host's. Data providers are either run inside the virtualized OS or are part of the VMM. The latter is difficult to detect by the analyzed sample but is limited to the collection of VM state information unless the actual CPU instructions are monitored and correlated to specific API calls as part of a hybrid approach [ESKK12]. Available classic VM solutions include the VMware product line [VMW14], the Xen project [Lin14], and Oracle VM VirtualBox [Ora14].
Emulated systems represent a system that does not share any physical characteristics with the host. In its basic implementation, CPU and memory are fully emulated (i.e., independent and isolated from the physical machine). Since the OS needs to be emulated as well, it is necessary to recreate all functionality (libraries, services, etc.) required to successfully run the sample. A program running in such an environment is not able to access the physical machine in any way but may crash if it requests a resource or function that is not part of the emulation. Full system emulation such as QEMU, on the other hand, also provides emulated hardware and peripherals [Bel05]. This makes it possible to run a full-fledged OS on virtualized hardware that is, unlike a VM, not bound to any specific architecture. Malware may utilize sandbox analysis evasion (and/or detection) techniques to check whether it is being run in an emulated environment. Also, emulation is much more resource-demanding than virtualization and significantly slower than a bare-metal machine. However, since emulation offers full access to the system from the outside, all sample activity can be collected directly from the emulator. Like the VMM-based approach, it is necessary to translate CPU state and memory contents to high-level information such as file or registry operations.

Base data describes the type of data monitored and logged by a provider. There is a multitude of information to be gleaned from static and dynamic analysis, each offering specific insight into the nature and functionality of a malicious program.

The *virus definition* is perhaps the simplest piece of extractable information. The sample's binary code is matched to patterns stored in a signature database of a virus scanner in order to determine if the entire file (checksum) or parts of the code (snippets) are known to be malicious. Many tools include this type of common virus scan to quickly determine a malware sample's category.

Packer information includes used packer designations and general compression information about the sample. Malware authors often use various packing algorithms to obfuscate the program's code and to impede forensic investigation. Many static analysis approaches require the sample to be unpacked in order to yield workable results.

File and header information describe a sample's actual type (independent from its cosmetic filename extension) and its code sections. Windows portable executable (PE) files come with a header that contains interesting metadata stored in so-called sections – e.g., the .text section contains the user-written code while .rdata lists import and export information [Mic99].

Library and function imports hint at the functionality that might be utilized by the sample upon execution. Libraries usually contain a number of functions related to a specific area of operation; e.g., the Windows library advapi32.dll aggregates most service manager and registry interaction functions while ws2_32.dll handles low-level networking [RSI12].

CPU instructions and their associated *assembly operations* are the machine code and low-level language representation of a program, respectively. Being a vital part of in-depth reverse-engineering, this base data type offers detailed insight into a sample's core functionality. The program is either disassembled into a trace of sequential instructions to the processor or is dynamically debugged to retrieve register values and identify dormant code.

Unlike function imports, monitoring the actual execution of raw *system and API calls* yields information about the general behavior of a sample. Calls may include wrapper functions that offer a simple interface to the application programmer or native system calls that represent the underlying OS or kernel support functions. Interpreting system calls allow the analyst to identify e.g., file creation, registry modification, socket interaction, or setup routines.

File system operations sum up specific activity on a file object level. The creation, modification, and deletion of files is monitored and logged. While tools usually use system and API call monitoring to discern file system operations, the added layer of abstraction drastically increases the readability of the information.

Registry, process/thread, and *network operations* are semantically and syntactically similar but are usually processed and presented independently. While file interaction, registry operations, and process commands are usually derived only from calls, network activity is either collected through call tracing or by directly monitoring network traffic at the physical or logical interface adapter. Data providers may utilize traffic logging to extract a multitude of handy information such as IP addresses contacted by the infected machine, information on downloaded files, or even plain-text passwords.

3.1. Comparison and Discussion

In the following, we compare specific data providers and their technical capabilities (cmp. Table 1). While the remain-

	Anubis	Cuckoo	CWSandbox	FireEye MAS	Joe Sandbox	ProcMon	APIMon	Generic disassembler	Generic debugger
Analysis mode and environment									
Static analysis support		✓✓		✓	✓✓			✓✓	
Dynamic analysis support	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓		✓✓
Native analysis environment			✓✓		✓✓	✓✓	✓✓	✓✓	✓✓
Virtual machine environment		✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓
Emulation environment	✓✓					✓✓	✓✓	✓✓	✓✓
(Simulated) Internet access	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓		
(Simulated) LAN services	✓✓		✓✓			✓✓	✓✓		
Interface									
Command line interface	✓✓	✓✓		✓✓	✓				
Graphical (web) interface (GUI)	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓
Sample input									
Single file submission	✓✓	✓✓	✓✓	✓✓	✓✓		✓✓	✓✓	✓✓
Folder submission	(✓)	(✓)		✓✓	✓✓				
URL/URI	✓✓	✓✓	✓✓	✓✓	✓✓				
Batch processing	(✓)	(✓)	(✓)	✓✓	(✓)	(✓)	(✓)	(✓)	(✓)
Interactive on-demand analysis	✓					✓✓	✓✓	✓✓	✓✓
Supported input file formats									
Windows executables (.exe)	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓
Windows libraries (.dll)	✓✓	✓✓	✓✓	✓✓	✓✓		✓	✓✓	✓
Microsoft Office files	✓✓	✓✓	✓✓	✓✓	✓✓				
Portable document format (.pdf)	✓✓	✓✓	✓✓	✓✓	✓✓				
Malicious URL scan	✓✓	✓✓	✓✓	✓✓	✓✓				
PHP files (.php)					✓✓				
Java file (.jar)	✓✓	✓✓	✓✓	✓✓	✓✓			✓✓	
Visual Basic scripts (.vbs)					✓✓				
Image files (.jpg, .png,...)		✓✓		✓✓					
Video files (.wmv, .flv,...)	✓	✓		✓✓	✓				
ZIP archive (.zip)	✓✓	✓✓		✓✓	✓✓			✓✓	
Base data									
Virus definition/Malware name	✓✓	✓✓	✓✓	✓✓					
Behavior classification	✓				✓✓				
Packer information					✓			✓	✓
File information/File header		✓✓						✓✓	✓
Library imports/loads	✓✓	✓✓	✓✓		✓✓		✓	✓✓	✓
CPU instructions/assembly					✓✓			✓✓	✓✓
API calls	✓	✓✓	✓	✓	✓	✓	✓✓	✓✓	✓✓
System calls	✓	✓✓	✓	✓	✓	✓	✓✓	✓✓	✓✓
File system operations	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓	✓	✓
Registry operations	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓	✓	✓
Process/thread information	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓	✓	✓
Network activity	✓✓	✓✓	✓✓	✓✓	✓✓	✓	✓	✓	✓
Report output									
PDF report	✓✓		✓✓						
HTML report	✓✓	✓✓	✓✓		✓✓				
XML report	✓✓		✓✓	✓✓	✓✓	✓✓			
TXT report	✓✓			✓✓			(✓)		✓✓
CSV report				✓✓		✓✓			
Native/Proprietary format						✓✓	✓✓	✓✓	✓✓
PCAP network dump	✓✓	✓✓	✓✓	✓✓	✓✓				
JSON report			✓✓	✓✓	✓✓				
Memory dumps		✓✓		✓✓	✓✓				
String dumps		✓✓			✓✓				
Screenshots		✓✓			✓✓				

Table 1: **Comparison of Data Providers** (✓✓... fully implemented, ✓... limited or partial implementation (often due to automated interpretation or the complete lack thereof), (✓) ... supported (through e.g., scripting), but not implemented by default.

der of the paper focuses on visualization approaches and techniques, below tools are assessed by the amount and quality of information they provide for subsequent visualization.

To encompass a meaningful range of environments and base data types, we evaluated a total of 5 static and dynamic analysis suites (which come with their own analysis environment) as well as 4 classes of stand-alone tools. It is important to keep in mind that this is not a strict feature survey or review of available commercial products but an assessment of the data these applications provide. For a better understanding of the different output data structures, we provide some examples on our supplement material webpage (http://mc.fhstp.ac.at/supp/EuroVisStar2015).

Anubis is an automated dynamic analysis tool which evolved from TTAnalyze [BKK06, BMKK06]. Its commercial offshoot is marketed under the name LastLine Analyst. Anubis uses the QEMU emulator [Bel05] to run potentially malicious software on a feature-complete Windows XP OS. A second virtual machine (VM) operates a number of fake network services for the malware to exploit. Since Anubis does not rely on API function hooking [Iva02] or debugging, it is harder to detect by malware than other VM-based solutions using these techniques. Altering the program through function call injection is supported by the tool. Anubis returns a high-level report that lists file, process, registry, and network activity. Its output is best suited for analysts who want a comprehensible overview of a sample's system or network behavior.

Cuckoo Sandbox [Ge14] is a lightweight open source solution for automating the dynamic analysis of files. It is heavily customizable and utilizes other stand-alone tools (e.g., tcpdump and volatility) to generate a complete picture of a program's activity. Cuckoo uses a common VM environment compatible with a range of systems – it is however recommended to use Ubuntu Linux as host and Windows XP for the guest images. The report file returns simple file, registry, and mutex interactions as well as limited static information. Due to its flexible nature, Cuckoo output data can assist visualization efforts in a wide variety of applications, including forensic memory or string dump analysis.

CWSandbox [WHF07] and its commercial successor GFI Sandbox/ThreatAnalyzer are dynamic malware analysis platforms that use either virtual or native (physical) Windows environments. Analysis is based on hooking functions that perform API-level monitoring by rewriting the sample upon load. Like Anubis and Joe Sandbox, it returns a tidied-up list of file system, registry, network, and other OS operations the sample performed.

FireEye Malware Analysis System (MAS) [Fir13,Fir14] is the forensic, lab-based version of the FireEye product line. The MAS supports configurable VM-based analysis of various code formats. It is possible to use preconfigured images with preinstalled software (e.g., Adobe Reader) or put together a custom installation. Unlike most other solutions, FireEye comes bundled with a hardware appliance of varying specifications. The system returns a textual trace that includes general file information, Yara signature matches [Alv15], and malicious alerts (certain API calls, process activity, etc.) triggered by the sample. Compared to the other suites, FireEye offers a slightly less comprehensible overview of malicious behavior and instead relies on a more alert-based approach. On the other hand, the MAS enables efficient automated parsing thanks to its multitude of output formats.

Joe Sandbox [Joe14] is a dynamic malware analysis suite that supports native and VM-based operation. API and system call hooking is performed for each sample; a kernel mode driver hides the necessary function rewriting from the malware under investigation. Joe Sandbox simulates user activity through various AutoIT [Aut14] scripts running directly on the analysis system, allowing for e.g., the automated interaction with an installer. The tool returns a comprehensive list of system activities and collects dropped files as well as a network trace. Static analysis is supported through an optional module. Joe Sandbox offers a good selection of output formats as well as a high level of analysis detail.

Process Monitor (ProcMon) [RC14] is a free file system monitoring tool developed by Mark Russinovich and Bryce Cogswell. Part of Microsoft's SysInternals suite, ProcMon combines non-destructive monitoring and logging of registry and process/thread activity through a device driver loaded at run-time [RSI12]. The tool returns an abstracted view of the system's API activity; its output includes the resource's time and type of access as well as the stack of the respective thread. Since ProcMon is not a malware suite, it does not come with an analysis environment of its own but requires a native or virtual Windows installation to run. The output data provided by ProcMon is especially well-suited for the visualization of processes and threads.

API Monitor (APIMon) is a free tool by Rohitab Batra that offers API and native function monitoring/manipulation functionality. It performs API hooking and supports a large number of native and undocumented calls as well as COM+ interface methods [Bat14]. Unlike most other tools and suites, it offers little in terms of result abstraction; while parameters and return values can be decoded on demand, the resulting trace is not interpreted in any way. This makes the tool very versatile in its application but may require additional processing or filtering prior to visualization. APIMon is well-suited for visualizing call sequences of specific threads. Dornhackl et al. [DKLT14] used a system based on an attributed grammar to automatically process APIMon output and map it to a hierarchical model of malicious behavior.

Generic disassembler and **generic debugger** are data provider categories that summarize the average capabilities of both types of analysis tools. Solutions include applications such as IDA Pro (a widely used disassembler for many different binary file formats), OllyDbg, and WinDbg (Microsoft Windows debuggers). Disassemblers and debuggers generally exist for various architectures and file types; it is recommended to peruse additional resources to identify the desired domain-specific solution. Many visualization tools name IDA Pro [Pan08, ASL12, HLI13, HLKI14] as their primary data source for static information. Using a disassembler or debugger will yield low-level data (e.g., CPU instructions) that is especially useful for image-based techniques and other raw-data visualization.

Discussion: The various tools and suites all come with their unique strengths and weaknesses. While analysis suites usually handle most of the data interpretation and remove excess information automatically, stand-alone tools often require further interpretation by the user. The unfiltered nature of their output, however, often allows for more flexible applications. In the end, the choice of a data provider will be driven by the specific needs of the malware analyst in regards to mode (static vs. dynamic), depth (activity overview or full traces), and output of the respective tool. In many cases, a combination of analysis tools will yield the most satisfying result.

Information in Table 1 was extracted through testing, taken from various analysis reports and documentation as well as from aforementioned literature. Please note that some capabilities may be subject to change since new features might be added to the tool/suite at a later point. On-site testing was performed with a 2010 version of Anubis, FireEye MAS 6.4.0, a 2013 version of Joe Sandbox, Process Monitor 3.1, and API Monitor v2 r-13. The latest Anubis, CWSandbox (ThreatAnalyzer), and Cuckoo (Malwr) sandboxes were assessed through their public web submission frontends.

After discussing some background on the malware analysis process and data providers to collect analysis data we will now investigate visual analytics methods for malware analysis.

4. Research Method

To get a comprehensive overview of visualization methods supporting malicious software analysis systems in the field of IT security, we used a number of *digital libraries* (IEEE Xplore, ACM digital library, Google Scholar, and Academic Research Microsoft). A skeleton of common search terms was used in all of them. To improve our search results we individually refined the different keywords and keyword combinations for each of the used search engines in order to achieve maximum topical coverage. This was necessary since each search engine has its own strengths and weaknesses (e.g., on IEEE Xplore it is possible to structure your own advanced search by selecting different search parameters). All the used search terms and combinations are provided for download on our supplementary material webpage (http://mc.fhstp.ac.at/supp/EuroVisStar2015). Based on the keywords and combinations used, we found about 200 publications.

In a second step, we identified the *authors* of the most relevant papers and refined our search to include other publications by these researchers. Additionally, we visited the homepages of the identified authors to look for additional material related to the research topics. Based on the employed search strategies it was possible to identify more than 220 different scientific papers and articles in the respective area.

In order to sort out inappropriate papers, we perused all the abstracts and the conclusions for relevant information. Through this process, we verified whether the identified papers really fit the main topic of malware analysis systems that make use of visualization methods. Thus, it was possible to reduce the findings to 42 papers. The categorization process and the elimination of inappropriate papers were performed in each search step of the research process.

In addition to the results of the search engines, we wanted to make sure to include all papers published at *VizSec (Visualization for Cyber Security)* which is the premier venue for discussing malware visualization systems as it brings together security and visualization experts. To explore VizSec publications, we utilized our publicly-available search interface for VizSec papers (http://vizsec.dbvis.de/) and skimmed through the entirety of publications. In the end, we identified 3 additional papers directly related to malware (most had already been found earlier). Finally, we investigated all the references of the current paper collection to check whether there are any papers still undiscovered.

We eventually identified 25 papers matching our specific topic of malware visualization systems. Some papers present incremental work which leads to the fact that [QL09] is similar to [QL11], because it is an extension journal paper of the same system. Similarly, [HKI14] is related to [HLI13], and [SM14a] to [SM14b]. However, we still decided to include all versions in the survey in order to present an extensive overview of all academic publications that are in the scope of this work.

To classify and categorize the identified papers, we built an interactive web application to gather responses and collect reviews of all the members of our team. The web application directly connects to a shared Zotero collection using the Zotero API [Roy15]. We decided on an extensive list of features and criteria to categorize and review the visualization systems. Two researchers extensively reviewed all the papers. The results were directly entered into our web application which stored them in a database and eventually synchronized them to the Zotero collection in the form of tags. Afterwards, all criteria where no consensus was reached were discussed to agree on a common approach.

The public part of the web application is available at

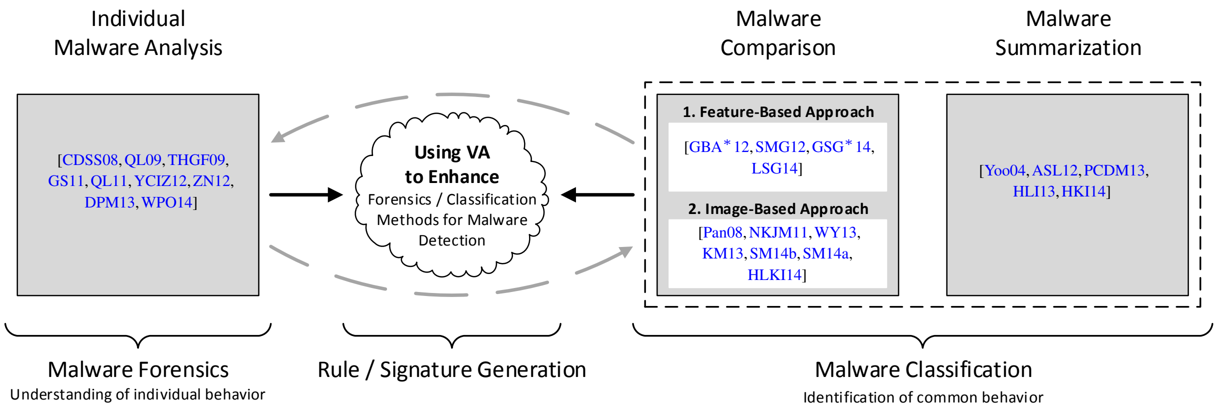

Individual Malware Analysis
[CDSS08, QL09, THGF09, GS11, QL11, YCIZ12, ZN12, DPM13, WPO14]
Using VA to Enhance Forensics / Classification Methods for Malware Detection
Malware Comparison
1. Feature-Based Approach
[GBA*12, SMG12, GSG*14, LSG14]
2. Image-Based Approach
[Pan08, NKJM11, WY13, KM13, SM14b, SM14a, HLKI14]
Malware Summarization
[Yoo04, ASL12, PCDM13, HLI13, HKI14]
Malware Forensics
Understanding of individual behavior
Rule / Signature Generation
Malware Classification
Identification of common behavior

Figure 2: **Malware Visualization Taxonomy** – Categorization of malware visualization systems into three categories, namely (1) Individual Malware Analysis, (2) Malware Comparison, and (3) Malware Summarization. All systems have the ultimate goal to generate rules and signatures for fully-automated malware detection systems. While the first category tackles the problem of understanding the behavior of an individual malware sample for forensics, the latter two focus on the identification of common behavior for malware classification.

(http://malware.dbvis.de/). All tables in this survey can be interactively explored using the mentioned web application.

5. Visualization Systems for Malware Analysis

Based on our literature research, we identified various general trends and objectives prevalent in malware visualization systems. Using visualization obviously helps to understand malware behavior, which is helpful for forensics and *malware detection*. Additionally, visual analysis can help to support the *malware classification* process. Malware detection does mostly refer to the automatic identification of malware (e.g., anti-virus software for end users), however, in more complex scenarios, targeted attacks, or for unknown malware, manual analysis by malware experts is inevitable. Such analysis helps to identify suspicious behavior, to eventually create rules and signatures, which can then be used to improve automated malware detection. Malware classification focuses on the aspect to assign an unknown malware sample to a known group of malware types.

In general, there are two different main goals of malware visualization systems. On the one hand, there are systems for malware forensics which will be used to understand the individual behavior of a malicious malware sample and on the other hand, there are malware classification tools which will be used to identify the common behavior of malware samples. Based on these main groups, we differentiate between three underlying main categories. We developed the *Malware Visualization Taxonomy* (see Figure 2) which represents the three categories:

Individual Malware Analysis: These systems support the individual analysis of primarily single malware samples to gain new insights of its individual behavior related to malware forensics.

Malware Comparison: This category fits to visualization tools that are primarily used for the comparison of n to m malware samples for the identification of common behavior (e.g., the malware family) to support malware classification. In general, we have identified two different subcategories:

- Tools using a **Feature-Based Approach** explore and compare different malware samples based on extracted features. Those tools use various data visualization techniques to compare characteristics with each other.
- The **Image-Based Approach** generates visual images based on binary data or the behavior logs of the malicious software. Eventually, those visual fingerprints are compared using computer vision techniques.

Malware Summarization: Systems of this category summarize the behaviors of n different malware samples to identify similarities and to gain new insights of their common behavior.

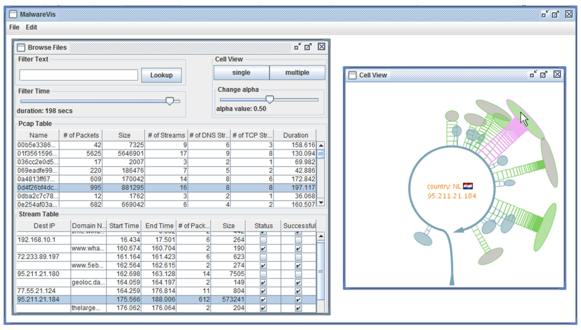
As sketched in Figure 2, eventually, one or several malware analysis tools can be used in combination to generate rules and signatures for malware samples or malware families based on the generated insights. Additionally, the increasing use of visual analytics methods will enhance the forensics and classification methods for malware detection.

Discussion: From the taxonomy as seen in Figure 2, it becomes obvious that 9 tools focus on individual malware analysis, 11 on malware comparison, and 5 on malware summarization to provide visual summaries of large amounts of malware samples and their characteristics. Additionally, it is interesting to see that only 4 tools for malware comparison are using primarily the feature-based approach, while 7 focus on image-based approaches.

Based on the various publication years, it becomes apparent that using malware characteristics (based on features extracted through static and dynamic malware analysis) is becoming more common since 2013 and that fewer systems focus on individual malware analysis (malware forensics). Most of the research for individual malware analysis was performed between 2004 and 2012. In the past 10 years, visualization seems to be used more often to generate image-like representations of malware samples which are then used for visual comparisons.

5.1. Visualization for Individual Malware Analysis

The first group contains visualization systems geared towards the extensive analysis of *individual* malware samples [CDSS08, QL09, THGF09, GS11, QL11, YCIZ12, ZN12, DPM13, WPO14]. Zhuo and Nadjin [ZN12], for example, focus on only one specific type of malware behavior – the network activity of a malware sample – which is then visualized by a glyph-like chart as can be seen in Figure 3. This specific feature can be explored in great detail which is not possible in other, less specialized visualization tools.



duration: 198 secs

alpha value: 0.50

Pcap Table

Name	# of Packets	Size	# of Streams	# of DNS Str...	# of TCP Str...	Duration
00b5e3386...	42	7325	9	6	3	158.616
01f3561596...	5625	5646901	17	9	8	130.094
036cc2e0d5...	17	2007	3	2	1	69.982
069eadfe99...	220	186476	7	5	2	42.886
0a4813ff67...	609	170042	14	8	6	172.842
0d4f26bf4dc...	995	881295	16	8	8	197.117
0dba2c7c78...	12	1762	3	2	1	36.068
0e254af03a...	682	669042	6	4	2	160.507

Stream Table

Dest IP	Domain N...	Start Time	End Time	# of Pack...	Size	Status	Successful
192.168.10.1		16.434	17.501	6	264	☐	☐
	www.wha...	160.674	160.704	2	190	✓	✓
72.233.89.197		161.164	161.423	6	623	☐	☐
	www.5eb...	162.564	162.615	2	274	✓	✓
95.211.21.180		162.698	163.128	14	7505	☐	☐
	geoloc.da...	164.059	164.197	2	149	✓	✓
77.55.21.124		164.259	176.814	11	804	✓	✓
95.211.21.184		175.566	188.006	612	573241	✓	✓
	thelarge...	176.062	176.064	2	204	✓	✓

Figure 3: **Individual Malware Analysis** – This interactive system visualizes network activity of an individual malware sample [ZN12]. *Image © 2012 ACM, Included here by permission.*

Other tools consider various features at the same time, but still focus on the individual analysis of single malware samples. Trinius et al. [THGF09] use treemaps and so-called thread graphs, as seen in Figure 4, to visually analyze system calls executed by the selected malware. While basic comparison is also possible with most of the tools in this category (e.g., using multiple instances of the same tool), they do not specifically support bulk analysis.

Future Research Directions: The visual analysis of individual malware samples leads the analyst to a better understanding of the specific behavior and can help to judge if an unknown sample is indeed malicious or not. However, current work could be improved with respect to malware detection, because many of those tools do not include classification methods to compare the observed behavior to the behavior of known malware types. In the future we expect more visual analytics tools to combine individual malware analysis with automated methods and to incorporate methods to directly relate and compare findings with behavior of known or previously analyzed samples. Automatic highlighting of important or possibly malicious aspects, would help the analyst to quickly focus on most suspicious behavior first to reduce the time needed for manual analysis.

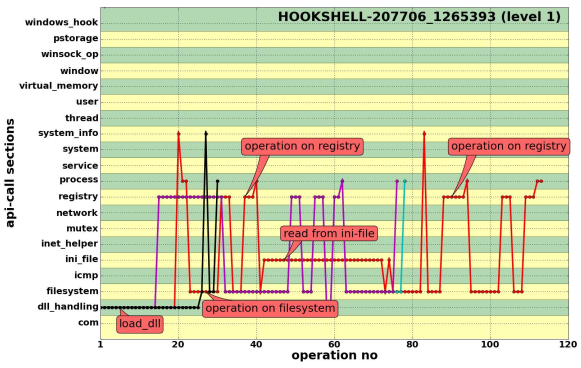


Figure 4: **Individual Malware Analysis** – Visual representation of system calls issued over time by an individual malware sample. *Image © 2009 IEEE. Reprinted, with permission, from [THGF09].*

5.2. Visualization Support for Malware Comparison

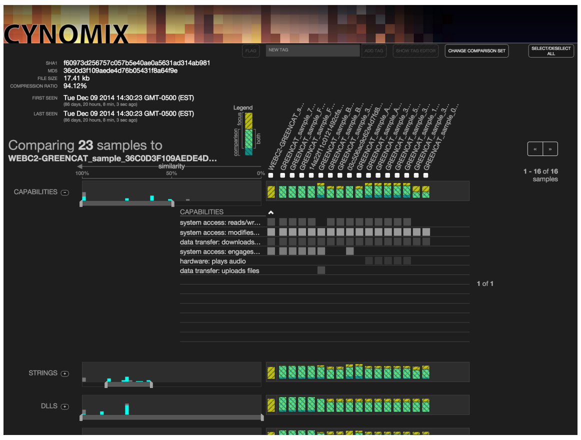
While the individual analysis is needed to get a deep understanding of a malware sample, the comparison with already known malware samples is crucial for malware classification. On the one hand, this step helps to reduce the number of samples that need time-consuming manual analysis. On the other hand, comparison with other samples can help to identify groups or malware families. All the systems which are represented in this category use visualizations to enhance the comparison of *n* with *m* malware samples for the identification of their common behavior (e.g., to identify related samples, find the correct malware family). Technically, we distinguish between feature-based and image-based approaches.

5.2.1. Feature-Based Approach

Feature-based approaches [GBA*12, SMG12, GSG*14, LSG14] use visual analytics techniques to let the user filter, search, compare, and explore a wide range of properties extracted during analysis. These systems provide means to compare malware samples based on their similarities of features.

Individual exploration of these features is also possible, but is much more limited, compared to the previous category. While some of the tools of the previous category were specifically designed to do an in-depth analysis of network activity or to fully explore the temporal sequence of system calls, feature-based malware comparison tools try to focus on a broad set of different features and characteristics, and try to make them all accessible to the analysts. This leads to more abstract representations, higher aggregation levels, and eventually less details for individual features (e.g., ignoring the temporal aspects of network connectivity).

Figure 5 shows a screenshot of a visual analytics system by Gove et al. [GSG*14] used to interactively explore and compare large sets of characteristics or attributes of samples in malware corpora.


SHA1 f60973d256757c057b5e40ae0a5631ad314ab981
MD5 36c0d3f109aede4d76b05431f8a64f9e
FILE SIZE 17.41 kb
COMPRESSION RATIO 94.12%
FIRST SEEN Tue Dec 09 2014 14:30:23 GMT-0500 (EST)
(86 days, 20 hours, 8 min, 3 sec ago)
LAST SEEN Tue Dec 09 2014 14:30:23 GMT-0500 (EST)
(86 days, 20 hours, 8 min, 3 sec ago)

Comparing **23** samples to
WEBC2-GREENCAT_sample_36C0D3F109AEDE4D...

CAPABILITIES
system access: reads/wr...
system access: modifies...
data transfer: downloads...
system access: engages...
hardware: plays audio
data transfer: uploads files

Figure 5: **Comparison of Malware Characteristics** – Identifying similar malware samples to a focus sample by comparing them along different sets of characteristics (e.g., capabilities) [GSG*14]. *Image courtesy of Robert Gove.*

The advantage of such approaches is that the analyst can directly compare various features. This helps to understand in which features malware binaries are related and in which they are not. However, on the other hand it is harder to get a quick visual overview of occurring patterns.

Future Research Directions: The comparison of characteristics helps to visually enhance the malware classification process in various ways. Tools in this category also focus on the question of which features can be extracted and used for comparison. Comparing such malware characteristics helps to identify related samples based on similarity metrics and to identify the common behavior of the explored samples for classification. Especially, the possibility to compare many different features at once and the possibility to apply standard methods from the field of data analysis (e.g., MDS, PCA, clustering) opens a promising research direction. Using visual interfaces to guide the analyst in the selection of features seems to be a good way to better support malware classification. Such visual analytics interfaces would eventually help to define better classifiers to improve malware classification models.

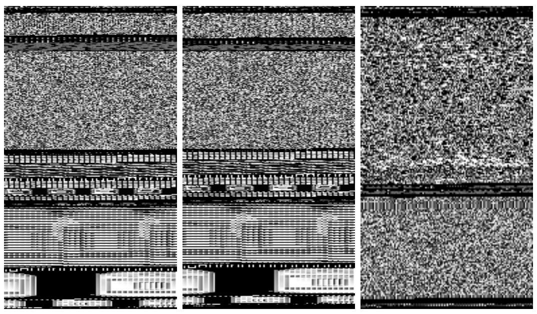

(a) FakeRean.D (b) FakeRean.E (c) Mebroot

Figure 6: **Comparison of Malware Images** – Visualizing malware executables as grayscale images is a common technique to visually identify similarities with low computation costs. *Image by the authors.*

5.2.2. Image-Based Approach

Image-based approaches [Pan08, NKJM11, WY13, KM13, SM14a, SM14b, HLKI14] have in common that they use visual mappings to render an image for each malware sample.

For example, the analyst might need to correlate a given suspicious file to a cluster of malware variants in order to associate the file to a specific malware family. Similar images can be visually clustered using either a manual or an automatic approach based on algorithms from the areas of computer vision and image processing. Some systems visualize the binary data and directly map the (raw) byte-code representation or respective entropy values to an image (e.g., [NKJM11, HLKI14]). We applied this technique to variants of the *FakeRean* malware as seen in Figure 6a. We use this to detect similar images representing related malware samples (Figure 6b). These particular malware samples can be visually distinguished from Figure 6c, which represents a *Mebroot* malware sample, sharing no visual patterns with the other malware family.

Nataraj et al. [NYPZ11] extract various texture features from such images, to eventually use them for classification. The advantage of this technique is, that it can be applied to any file and can be computed efficiently, which is important for large malware corpora. While classification accuracy is quite comparable for many malware variants, the approach is limited because it does not make use of any dynamic analysis and only relies on the actual bytes found in the binaries. Another problem is, that the visual impression is strongly dominated by possible images embedded in the resource section of an executable, which could be avoided by malware authors to create less obvious visual patterns.

To overcome this drawback, the approach was extended to visualize disassembled CPU instructions or API calls (e.g., [Pan08, SM14a, SM14b]) in a similar way, however, resulting in higher computation costs.

Future Research Directions: One possible future research direction could be the implementation of interaction methods to segment a region of interest or to characterize these texture patterns. Automated image comparison would help analysts to visually identify common code portions or specific instruction blocks within a sample. This information could be used to directly highlight relevant sections in the image. Additionally, the integration and combination of image- and feature-based methods could be promising. Image-based methods using static analysis together with a probability score can be used as efficient first step in a classification pipeline. Afterwards, the more expensive feature-based methods together with dynamic analysis would only be applied to those samples, which share less distinctive image representations, eventually leading to a more scalable classification process.

5.3. Visualization Support for Malware Summarization

While this category is more diverse, the associated tools all provide primarily some kind of summarization capability for a large number of malware samples within the visualization [Yoo04, ASL12, PCDM13, HLI13, HKI14]. Some identify a visual mask that is common for all selected samples (e.g., [Yoo04]) as seen in Figure 7. Others summarize and extract a single combined representative out of many malware variants (e.g., [HLI13, HKI14]). Finally, some use visual representations to show hierarchical clusters [PCDM13] or use heatmaps to visually represent kernels used for a support vector machine classifier to summarize and eventually classify malware samples [ASL12].

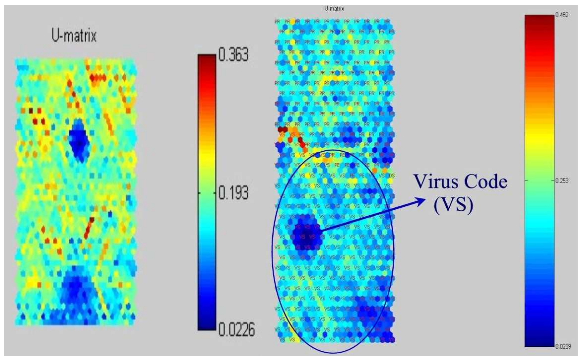


Figure 7: **Visualization Support for Malware Summarization** – A self-organized map is calculated and visually represented by the system to summarize many malware variants to extract common regions. With this technique it is possible to create a topologically ordered data mapping [Yoo04]. *Image © 2004 ACM, Included here by permission.*

Future Research Directions: The combination of different types of base data and data provider analysis modes are frequently stated as future work in this category. This will result in larger amounts and more heterogeneous data as input for visualization systems. Another direction into larger amounts of data can be the comparison of malware families as a whole based on their summarization. Finally, the integration of malware summarization with malware comparison and malware forensics using semantic zoom for example is a promising direction.

6. Categorization and Comparison

To provide a systematic overview of the findings from our literature research, we decided to consistently categorize all tools by the type of provided data, used visualization techniques [Kei02], mapping and representation space [AMST11], temporal aspects [AMST11], interactivity, and problems/actions ("Why?") [Mun14]. Thus, all the used categorizations are based on well-established taxonomies used in the visualization community and are described in detail in this section.

6.1. Data Providers in Visual Analytics

This section highlights the common denominator of visualization tools and malware data providers (cmp. Section 3). The input requirements of every visualization tool correspond to report output formats used by data providers. As mentioned above, *base data* describes the actual type of information gleaned from malware analyses – it basically determines the specific kind of monitored system activity or program code to be subsequently visualized.

Table 2 shows the base data visualized by the various solutions while Table 3 lists the respective data processing formats (provider output formats) of each tool introduced in Section 3.1. Using this information, an analyst can simply choose the desired type and format and pick a suitable data provider as well as visualization solution. Alternatively, the table might be used as reference for tool capabilities and its general approach.

Discussion: It is important to keep in mind that many visualization approaches utilize data gathered internally, e.g., through direct processing of a sample's binary. To encompass this vertical integration, the initial two base data categories were slightly altered: *raw virus definition* specifies that the tool uses the actual virus definition (instead of its plain-text abstraction as it is the case for most data providers) while *raw file* (a sample's binary/hexadecimal or ASCII representation) replaces the preliminary behavior classification done by some dynamic analysis suites. A newly added category is *memory/driver I/O*, describing e.g., RAM read and write operations as well as driver I/O activity captured by specialized data providers or directly via the VMM. Other, minor adaptations include the removal of PDF reports (parsing PDF files is usually not feasible) and the addition of the *raw/binary* category for direct sample input. Samples using the raw data format for processing or input generally include

	[Yoo04]	[Pan08]	[CDSS08]	[QL09]	[THGF09]	[NKJM11]	[GS11]	[QL11]	[YCIZ12]	[GBA*12]	[ZN12]	[SMG12]	[ASL12]	[PCDM13]	[HLI13]	[WY13]	[KM13]	[DPM13]	[SM14b]	[HLKI14]	[HKI14]	[SM14a]	[GSG*14]	[WPO14]	[LSG14]
Raw virus definition	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Raw file (direct input)	✓	✓	✓	-	-	✓	-	-	-	-	-	-	✓	✓	✓	-	✓	✓	-	✓	✓	-	-	-	✓
Packer information	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-
File information/File header	✓	✓	-	-	-	✓	-	-	-	-	-	-	✓	✓	-	-	✓	✓	-	✓	-	-	✓	-	-
Library imports/loads	-	✓	-	-	✓	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	✓	-	-
CPU instructions/assembly	-	✓	-	✓	-	-	-	✓	✓	-	-	-	✓	-	✓	-	-	-	-	-	✓	-	-	-	-
API calls	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	✓	-	-	-	✓	-	✓	✓	✓	✓	-
System calls	-	-	-	-	-	-	✓	-	-	✓	-	-	✓	-	✓	✓	-	-	-	-	✓	-	✓	✓	-
File system operations	-	-	-	✓	✓	-	✓	✓	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	✓	-
Registry operations	-	-	-	-	✓	-	✓	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	✓	-
Process/thread information	-	-	-	-	✓	-	✓	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	✓	-
Network activity	-	-	-	-	✓	-	✓	-	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	✓	✓	-
Resource utilization	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Memory/driver I/O	-	-	-	✓	-	-	-	✓	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-

Table 2: **Base Data** – This table provides an overview of the base data that is used as input for the various malware visualization systems. As discussed in Section 3, the data is collected by various data providers or the tool itself.

	[Yoo04]	[Pan08]	[CDSS08]	[QL09]	[THGF09]	[NKJM11]	[GS11]	[QL11]	[YCIZ12]	[GBA*12]	[ZN12]	[SMG12]	[ASL12]	[PCDM13]	[HLI13]	[WY13]	[KM13]	[DPM13]	[SM14b]	[HLKI14]	[HKI14]	[SM14a]	[GSG*14]	[WPO14]	[LSG14]
HTML format	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
XML format	-	-	-	-	✓	-	-	-	-	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-
TXT format (plain text)	-	-	-	-	-	-	✓	-	-	✓	-	-	✓	-	✓	-	-	-	✓	-	✓	✓	-	-	-
CSV format	-	-	-	-	-	-	-	-	-	✓	-	✓	✓	-	-	-	-	-	✓	-	-	✓	-	-	-
Native/Proprietary format	✓	✓	-	-	-	-	-	-	-	-	-	✓	✓	-	✓	-	-	-	-	✓	-	-	✓	✓	-
PCAP/network traffic	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
JSON format	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Raw/binary	✓	✓	✓	✓	-	✓	-	✓	✓	-	-	-	-	✓	✓	✓	✓	✓	-	✓	✓	-	✓	-	✓
Memory dumps (raw)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
String dumps	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-
Images (pictures)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓

Table 3: **Data Format** – Visualization systems use various data formats as input data, generated by the data providers.

certain data provider functionality and do not rely on external applications. Also note that some of the listed capabilities are only implied by the respective authors; not every format or type of base data is accurately specified.

6.2. Visualization Techniques

For the categorization of the different visualization techniques we used the "*Information Visualization and Data Mining*" taxonomy by Keim [Kei02]. More precisely, we focused on the part discussing visualization techniques. Based on this taxonomy it is possible to divide the used techniques into 5 generalized categories:

- **Standard 2D/3D Displays:** Includes visualization techniques like *x-y (x-y-z) plots* (e.g., scatter plots), *bar charts*, and *line graphs* [Kei02].
- **Geometrically-transformed Displays:** This category aims to visualize interesting transformations of multidimensional datasets (e.g., *scatter plot matrices* [And72], *node-link diagrams*, *parallel coordinates* [Kei02], *stardinates* [LMP05]).
- **Iconic Displays:** The attributes of multidimensional data items are mapped onto the features of an icon for the representation (e.g., *chernoff faces* [Che73]), *needle icons*, *star icons*, *stick figure icons* [PG98], *color icons*, and *tile bars*).
- **Dense Pixel Display:** Each data point is mapped to a colored pixel and they are grouped into adjacent areas that represent individual data dimensions. (e.g., *matrix* visualizations).
- **Stacked Display:** Representations for hierarchical data (e.g., *hierarchical stacking*, *treemaps*, *neighborhood treemaps* [DSF*14] also called *Nmaps*) and hierarchical layouts for multidimensional data (e.g., *dimensional stacking* [LWW90].)

Discussion: Our findings are summarized in Table 4. It is interesting that stacked displays and iconic displays are not commonly used in this domain. More research in appropriate glyph design seems to be promising because of the compactness of such visualization techniques. Most analysis support tools use standard 2D displays. Trinius et al. [THGF09] use treemap representations to analyze system call operations for

	[Yoo04]	[Pan08]	[CDSS08]	[QL09]	[THGF09]	[NKJM11]	[GS11]	[QL11]	[YCIZ12]	[GBA*12]	[ZN12]	[SMG12]	[ASL12]	[PCDM13]	[HLI13]	[WY13]	[KM13]	[DPM13]	[SM14b]	[HLKI14]	[HKI14]	[SM14a]	[GSG*14]	[WPO14]	[LSG14]
Standard 2D Display	-	-	-	✓	✓	-	✓	✓	-	✓	-	-	-	-	-	✓	-	✓	✓	✓	-	✓	✓	✓	✓
Standard 3D Display	-	✓	-	✓	-	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Geometrically-transformed Display	✓	✓	-	✓	-	-	✓	✓	✓	-	✓	-	-	✓	-	-	-	-	-	-	-	-	-	✓	✓
Iconic Display	-	-	-	-	-	-	-	-	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	✓	-	-
Dense Pixel Display	✓	-	✓	-	-	✓	-	-	-	-	-	✓	✓	-	✓	-	✓	✓	✓	✓	✓	✓	✓	-	-
Stacked Display	-	-	-	-	✓	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	✓	-	-

Table 4: **Visualization Techniques** – A general overview of the most frequently used types of visualization techniques based on the taxonomy of Keim [Kei02].

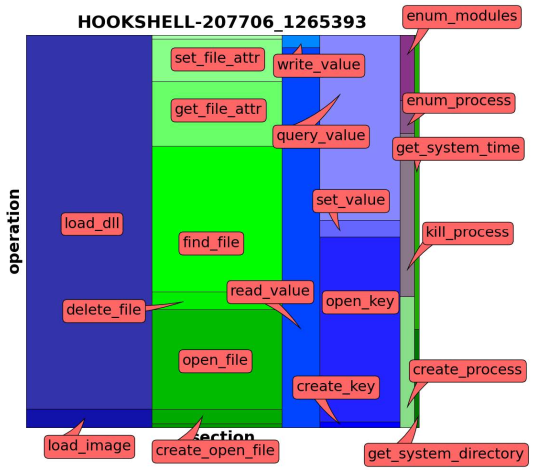


Figure 8: **Malware Treemap** – Visualization providing an overview of the most frequently used system call operations for an individual malware sample. It is evident that this sample uses calls from 6 out of 20 sections with operations from the "dll handling" and "filesystem"sections occuring most often. *Image © 2009 IEEE. Reprinted, with permission, from [THGF09].*

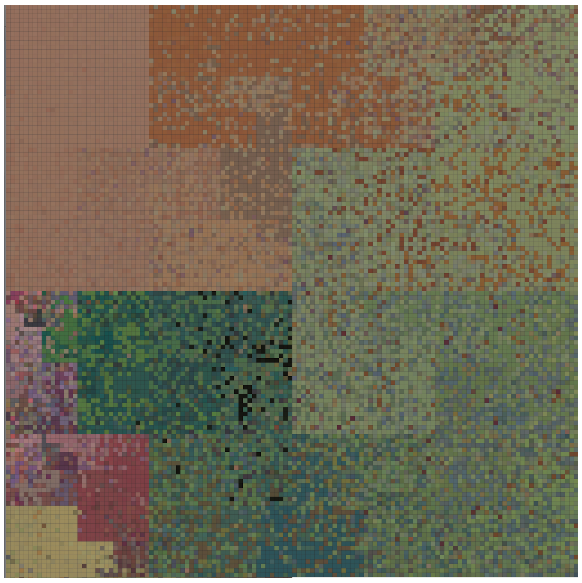


Figure 9: **Dense Pixel Displays** – Each pixel shape represents a malware sample. Similar malware samples are arranged next to each other and assigned similar color values to visualize commonalities [SMG12]. *Image courtesy of Josh Saxe.*

individual malware samples, as seen in Figure 8. Interestingly, a total number of 13 tools use visualization techniques which fall into the category of dense pixel displays. The reason for this is that a wide range of tools depict malware samples as image-like representations (Figure 6) which can already be interpreted as dense pixel displays. Dense pixel displays are also used to convey similarities between malware samples as can be seen in Figure 9 [SMG12].

6.3. Representation Space & Mapping to Time

While a large variety of visual representations are possible for analysis of malware data, we can categorize these by two fundamental dichotomies: the dimensionality of the representation space and whether physical time is used to convey data [AMST11].

In general, the **representation space** of a visualization can be either *2D* (two-dimensional) or *3D* (three-dimensional). There is no consensus in the community as to which of the representations is generally better suited for visualization [AMST11].

- **2D** visualization uses the two available dimensions (x-axis and y-axis) of a computer display, whereby all visual elements are described in respect to these coordinates (e.g., dots, lines, circles and arcs).
- **3D** visualizations additionally use the third dimension (the z-axis) for the representation of a geometry. This implies that the visualization gets more complex by the use of volumetric structures, as for example seen in Figure 10.

Mapping to time adds the temporal dimension of time to be used as part of a slide show or animation. This dynamic approach lends itself to a time-to-time mapping that can be

	[Yoo04]	[Pan08]	[CDSS08]	[QL09]	[THGF09]	[NKJM11]	[GS11]	[QL11]	[YCIZ12]	[GBA*12]	[ZN12]	[SMG12]	[ASL12]	[PCDM13]	[HLI13]	[WY13]	[KM13]	[DPM13]	[SM14b]	[HLKI14]	[HKI14]	[SM14a]	[GSG*14]	[WPO14]	[LSG14]
Mapping ► Static	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Mapping ► Dynamic	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Dimensionality ► 2D	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Dimensionality ► 3D	-	✓	-	✓	-	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 5: **Mapping & Representation Space** – An overview of used representation spaces in visualization systems. Almost all tools focus on static mappings.

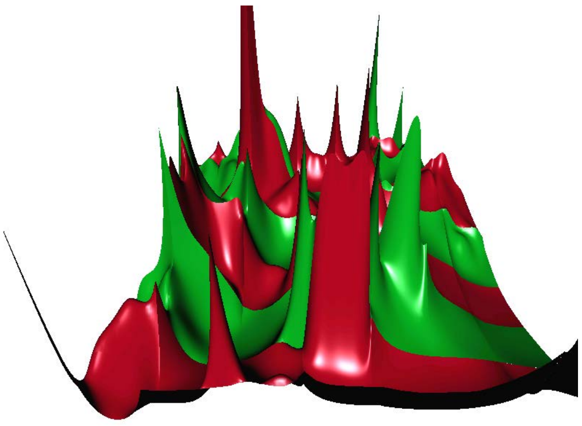


Figure 10: **3D Visualization** – Panas proposes 3D visualization to generate a specific visual signature that helps to identify anomalies within a malware family. For the data visualization, Panas mapped the number of control transfer instructions (x-axis), the number of data transfer instructions (y-axis) and the number of instructions (z-axis) [Pan08]. *Image © 2008 ACM, Included here by permission.*

projected to various visual variables. In addition, streaming data sources or progressively updated automated analysis can lead to a dynamic mapping [AMST11].

- **Static:** The data is mapped to display space and other visual variables that do not change over time. Static mapping does not exclude interactivity – it can still be modified through user interaction.
- **Dynamic:** Displayed data changes over physical time without the need for user interaction.

Discussion: Table 5 summarizes the findings with respect to representation space and mapping to time. Obviously, most of the tools focus on static mapping. The reason for this might be that many of the tools use base data that does not consider chronological order. Another reason might be that a dynamic representation makes it harder for the analyst to focus on specific characteristics gleaned through static analysis. Only the tool by Yee et al. [YCIZ12] uses a more dynamic mapping. They provide a visual debugger with node-link diagrams which can be used to replay the control flow of a malware sample (Figure 11). On the representation space side, 4 out of 25 tools map the data to a 3D representation space in addition to a 2D visualization. The remainder utilizes 2D representation only. Only one tool [Pan08] (see Figure 10) solely uses 3D representation of the analysis data. Additionally, only two tools ([YCIZ12] and [WPO14]) provide a static and dynamic mapping to time. In contrast to static mapping, physical time (dynamic mapping) can be used to encode data. Therefore, several frames will be rendered for the time steps in the data so that a 1:1 mapping could be implemented between time steps and frames. However, in practice this is not always realizable.

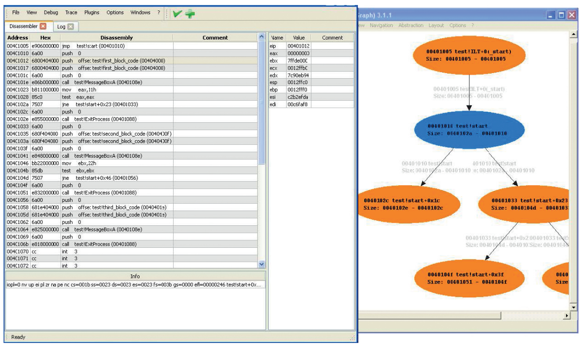


Figure 11: **Dynamic Mapping** – A visual debugger for malware analyis using node-link diagrams with replay capabilities to show the execution flow of a malware sample over time [YCIZ12]. *Image courtesy of Chen Lee Yee.*

6.4. Temporal Aspects

Time-oriented data plays an important role in malware analysis: For example, the execution order of system or API calls is relevant to identify certain behavior patterns (such as the creation and subsequent deletion of files). The time passed between two specific calls could also be of importance. Time can be modeled in different ways depending on analysis goals. For our categorization, we use a selection from time-oriented data aspects introduced by Aigner et al. [AMST11].

	[Yoo04]	[Pan08]	[CDSS08]	[QL09]	[THGF09]	[NKJM11]	[GS11]	[QL11]	[YCIZ12]	[GBA*12]	[ZN12]	[SMG12]	[ASL12]	[PCDM13]	[HLI13]	[WY13]	[KM13]	[DPM13]	[SM14b]	[HLKI14]	[HKI14]	[SM14a]	[GSG*14]	[WPO14]	[LSG14]
Scale ► Ordinal	-	-	-	✓	✓	-	✓	✓	✓	✓	✓	-	-	-	✓	-	-	-	✓	-	✓	✓	-	-	-
Scale ► Discrete	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Scale ► Continuous	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Arrangement ► Linear	-	-	-	✓	✓	-	✓	✓	✓	✓	✓	-	-	-	✓	-	-	-	✓	-	✓	✓	-	✓	-
Arrangement ► Cyclic	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Granularity ► None	-	-	-	✓	✓	-	✓	✓	✓	✓	-	-	-	-	✓	-	-	-	✓	-	✓	✓	-	-	-
Granularity ► Single	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Granularity ► Multiple	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Time primitives ► Instant	-	-	-	✓	✓	-	✓	✓	✓	✓	-	-	-	-	✓	-	-	-	✓	-	✓	✓	-	✓	-
Time primitives ► Interval	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Time primitives ► Span	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 6: **Temporal Aspects** – An overview of used time primitives. It is interesting to see that only 12 of the reviewed systems focus on temporal aspects, while the others do not specifically focus or do not convey temporal aspects of the malware behavior in the visual representations.

6.4.1. Scale

- **Ordinal:** The *ordinal* time domain represents only relations between time aspects without anchoring or quantification (e.g., before, after).
- **Discrete** time domains are also able to model distances and can be represented by a mapping of time values to a set of integers.
- **Continuous:** With this type of time model, a mapping to real numbers is possible.

6.4.2. Arrangement

- **Linear:** The *linear* form corresponds to our perception of time, from the past to the future (like a timeline) whereby each element has a predecessor and a successor.
- **Cyclic:** If the data is composed of a set of recurrent time values, we are talking about *cyclic* arrangement (e.g., the 4 seasons of the year).

6.4.3. Granularity and Calendars

- **None:** If time values are not mapped (divided) by any kind of granularity (e.g., years, quarters, months and so on), the system will have no granularity.
- **Single** granularity describes the mapping of the time values to only 1 type of granular unit (e.g., years or months).
- **Multiple:** With the mapping to *multiple* granularities, it is possible to divide the time values into years, quarters, months and so on. Such a mapping is referred to as a calendar.

6.4.4. Time primitives

- **Instant:** A single point in time is called an *instant* (e.g., January 16, 2015).
- **Interval:** An *interval* is a portion of a time between two *instants* (e.g., beginning and end).
- **Span:** A *span* is an unanchored primitive which represents a directed duration of time (e.g., 4 hours) in terms of a number of granules in a given granularity.

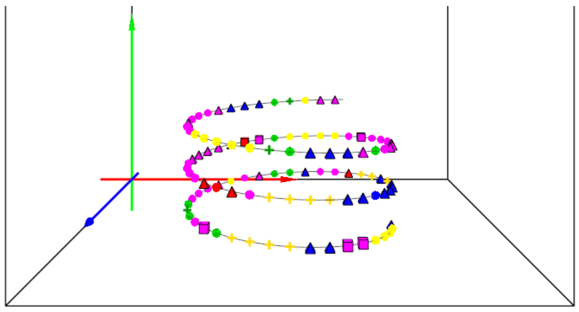


Figure 12: **Interaction** – This tool represents an ordered sequence of the malicious actions using an iconic representation in a spiral form. For the data exploration it is possible to zoom in and out, rotate, tilt, select different behavior slices, view the textual logs and compare it with other available behavioral data [GBA*12]. *Image courtesy of André Grégio.*

Discussion: Interestingly, only 12 out of 25 presented tools use temporal aspects. All these 12 tools have a *linear* arrangement, 11 use an *ordinal* time scale (see Table 6) and only the tool by Wüchner et al. [WPO14] uses a *discrete* time scale. Only 2 tools use an *interval* based time primitive, whereby the [ZN12] tool uses a *single* granularity (as seen in Figure 3) and the [WPO14] tool uses *multiple* granularities. The remaining 10 visualization systems use *instants* as time primitives and do not feature any granularity.

6.5. Interactivity

For the categorization of the systems' interactive capabilities we explored whether interaction techniques such as zooming, filtering, panning, details on demand, or brushing/linking are available (e.g., [Shn96, TC05, KMS*08]). Additionally, we tried to find out if it is possible to switch dynamically between different visual data representations.

Discussion: The main issue with this category was that

	[Yoo04]	[Pan08]	[CDSS08]	[QL09]	[THGF09]	[NKJM11]	[GS11]	[QL11]	[YCIZ12]	[GBA*12]	[ZN12]	[SMG12]	[ASL12]	[PCDM13]	[HLI13]	[WY13]	[KM13]	[DPM13]	[SM14b]	[HLKI14]	[HKI14]	[SM14a]	[GSG*14]	[WPO14]	[LSG14]
Interaction	-	-	✓	✓	-	-	✓	✓	✓	✓	✓	✓	-	-	-	✓	-	✓	-	-	-	-	✓	✓	✓
No Interaction	✓	✓	-	-	✓	✓	-	-	-	-	-	-	✓	✓	✓	-	✓	-	✓	✓	✓	✓	-	-	-

Table 7: **Interactivity** – An overview of the level of interactivity in the visualizations used by the tools.

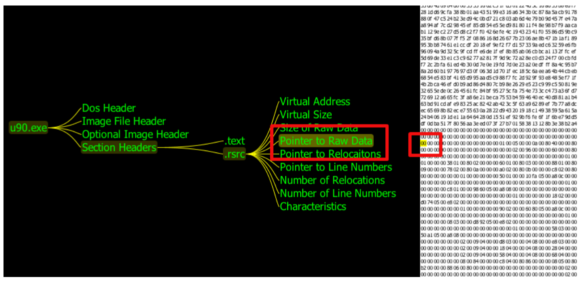


Figure 13: **Interaction** – Linking the hex editor on the right to an interactive tree representation enhances navigation and understanding of malware header data [DPM13]. *Image courtesy of John Donahue.*

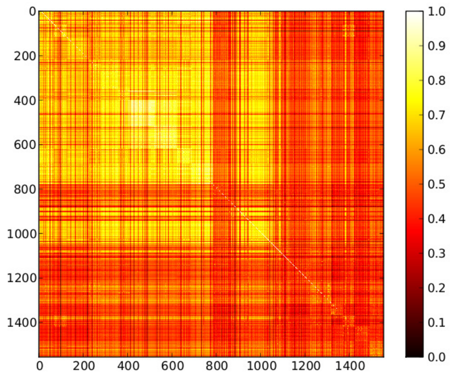


Figure 14: **No Interaction** – Example for a non-interactive dense pixel visualization showing similarity between 780 malware samples (top left) and 776 benign samples (bottom right) [ASL12]. *Image © 2012 ACM, Included here by permission.*

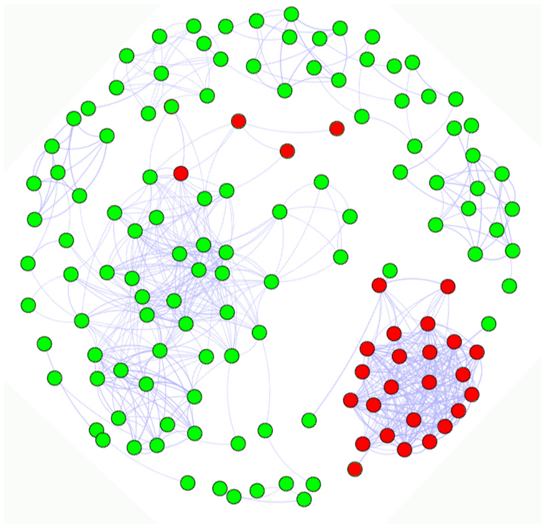


Figure 15: **No Interaction** – Malware samples are arranged in a node-link diagram with edge weights based on how many antivirus systems label them in the same category. The red nodes belong to a known malware family.The placement of the nodes were calculated automatically and the user could not interact with them [GS11]. *Image courtesy of André Grégio.*

many of the papers did not specifically describe which of the aforementioned features they actually support. Most of the time, the tools were only dubbed as *interactive* in general without offering a more detailed explanation. Therefore, we decided to limit the categorization to whether the system supports any kind of interaction (see Table 7) without going into detail. Based on this simple categorization we found that 13 out of 25 tools support interaction [CDSS08, QL09, GS11, QL11, YCIZ12, GBA*12, ZN12, SMG12, WY13, DPM13, GSG*14, WPO14, LSG14]. An example representation for an interactive analysis tool can be seen in Figures 12 and 13. A non-interactive solution is depicted in Figures 14 and 15.

6.6. Problems/Actions ("Why?")

Brehmer and Munzner [BM13, Mun14] proposed a multi-level typology to describe abstract visualization tasks that may be performed by a user. The abstraction of domain-specific vocabulary helps to identify similarities between these tasks but it is hard to abstract them in a comparable way. Munzner defined three levels of actions to describe a user's goal: *analyze* (top-level), *search* (mid-level) and *query* (bottom-level). The typology translates all needed domain-specific terms into a generic terminology and thus fits well

	[Yoo04]	[Pan08]	[CDSS08]	[QL09]	[THGF09]	[NKJM11]	[GS11]	[QL11]	[YCIZ12]	[GBA*12]	[ZN12]	[SMG12]	[ASL12]	[PCDM13]	[HLI13]	[WY13]	[KM13]	[DPM13]	[SM14b]	[HLKI14]	[HKI14]	[SM14a]	[GSG*14]	[WPO14]	[LSG14]
Analyze ► Consume ► Discover	-	-	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	-	✓	-	✓	-	✓	✓	✓	-	✓	✓	✓	✓
Analyze ► Consume ► Present	✓	-	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	-	-	-	✓	✓	-	✓	✓	-	✓	✓	✓	✓
Analyze ► Consume ► Enjoy	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Analyze ► Produce ► Annotate	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	✓	✓	-
Analyze ► Produce ► Record	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Analyze ► Produce ► Derive	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Search ► Lookup	-	-	-	✓	-	-	✓	✓	✓	-	✓	✓	-	-	-	-	-	✓	-	-	-	-	✓	-	-
Search ► Browse	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	✓	-	✓
Search ► Locate	-	-	✓	✓	✓	-	✓	✓	✓	-	✓	✓	-	-	-	-	-	✓	-	-	-	-	✓	-	-
Search ► Explore	-	-	-	-	-	-	✓	-	-	-	✓	✓	-	-	-	✓	-	-	-	-	-	-	✓	✓	-
Query ► Identify	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	✓	✓	-
Query ► Compare	✓	✓	-	✓	✓	✓	✓	✓	-	✓	✓	✓	-	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-	✓
Query ► Summarize	✓	-	-	-	-	-	-	-	-	-	-	✓	✓	✓	✓	-	-	-	-	-	✓	-	-	-	-

Table 8: **Problems/Actions ("Why?")** – The analysis based on the main actions supported by the visualization systems helps to identify gaps and unexplored research areas.

into our classification framework. In the following, we describe how we applied these abstract actions to the context of malware visualization [Mun14].

6.6.1. Analyze

Consume: Information that has previously been generated is *consumed* by the user [Mun14]. This subcategory is divided into the following three types of actions:

- **Discover:** Describes the generation and verification of hypotheses using visual exploration as well as the gaining of new knowledge about the presented data.
- **Present:** Incisive information communication or *storytelling* based on the visualized data.
- **Enjoy:** Refers to casual usage or pure enjoyment of visualization without specific goals or needs at hand.

Produce: In this case, the user's intent is to generate new material or output which will be used as input for further (usually visualization-related) tasks [Mun14].

- **Annotation:** Textually or graphically annotates the visualization. These annotations are typically made by hand (e.g., tagging the points of a scatter plot).
- **Record:** Captures or saves selected visualization elements (e.g., everything that makes sense to store). In contrast to the *annotate* action, the *record* action saves relevant visualization data as a persistent element.
- **Derive:** Produces new data elements which are based on existing data elements. Thus, it is possible to derive new attributes from existing information or to transform one data type into another.

6.6.2. Search

All the actions which were presented in the *analyze* area require *search* activities for the elements of interest that are described in this mid-level area. Munzner divided this area into four categories whereby the identifier and the location of the target elements are known or not [Mun14]. In this survey, we used the *malware sample* as the *target* while the *malware characteristics* were used as the *location*.

- **Lookup:** The user knows what she is looking for and were it can be found (target and location are known). Applied to malware visualization, the analyst loads a specific malware sample to analyze a predetermined set of characteristics. Such a lookup action would help to e.g., confirm a suspicion or to investigate certain properties of the sample in order to to eventually understand its behavior.
- **Locate:** The user knows what she is searching for but not where she has to search (location is unknown but target is known). Applied to malware visualization, the analyst again loads a specific malware spample to analyze. However, its relevant behavior features have yet to be discovered. That means that the target of interest is known, but the location and characteristics need to be located.
- **Browse:** The user does not know the exact identity of the target she is looking for but she knows the location where it can be found. Applied to malware visualization, the analyst is in this case interested in many different malware samples. The precise target is unknown; she merely knows what characteristics to look for.
- **Explore:** The user does not know what she is looking for and she also does not know where she has to search (target and location are unknown). Applied to malware visualization, the analyst is interested in many different malware samples and does not have a specific target in mind. It is also unclear which of the available characteristics are relevant, so the precise location to look at is also unknown.

6.6.3. Query

Once a (set of) target(s) for a search is identified, additional information will be queried as part of this bottom-level goal. Munzner [Mun14] named three different types of queries which are described below:

- **Identify** refers to a single target. If the search result is a known target (usually found by *lookup* or *locate* operations), the *identify* query returns the characteristics of the target.

- **Compare** actions consider multiple targets. A *comparison* query activity takes more sophisticated techniques than an *identify* query activity since the specifics of the comparison need to be determined.
- **Summarize** actions apply to all possible targets. A "*comprehensive view of everything*" or a "*summary display of everything*" should be provided [Mun14].

Discussion: Table 8 offers an overview of the main focus of the visualization systems in terms of supported actions. Systems that feature the *summarize* action are especially suited for the corresponding category discussed in Section 5.3. Identifying a specific malware sample is not a commonly used action used in the reviewed systems. Most systems instead focus on comparing given malware samples to a larger set, e.g., to assign them to certain malware families.

7. Discussion and Future Challenges

Having surveyed and systematically compared the state of the art in visualization systems for malware analysis we can extract a number of findings and propose challenges for future work. These results provide particular guidance for visual analytics professionals working in the domain but also benefit both the visualization and IT security communities.

Bridge between categories: In Section 5 we identified three categories of malware visualization systems tackling different sub-problems of malware forensics and classification at the levels of individual malware samples, comparison of malware samples, and common features summarized from malware families. It is surprising that these categories cleanly partition the state-of-the-art in malware visualization. Furthermore, the prevalence of systems using malware samples either individually (9) or in comparison (11) is evident in comparison to systems working with the summaries of malware families (5), which is in sharp contrast to the domain literature's emphasis on increasing number of malware, malware families and variants in the wild (e.g., [LSKJ11, BHFH13, DKLT14]). Since there is a common goal of generating rules or signatures, it can be assumed the potential target users of all three visualization system categories overlap. Thus, future malware visualization systems should investigate comprehensive designs: for example to switch perspective between summarization and comparison or to semantically zoom into individual analysis mode. Likewise the integration of common features of malware families can be integrated into individual malware forensics to make it more expressive.

Integrate different data sources: Malware analysis is based on a wide range of base data collected by data providers under different analysis modes (Section 3). As malware gets more sophisticated in detecting and avoiding analysis, there is increasing need to combine different data providers – for example to combine static and dynamic analysis. This involves not only supporting different data formats but also handling the resulting heterogeneous data in a suitable way, for example through multiple coordinated views.

Problem characterization and abstraction for tailored visualization: Many systems use visualization only superficially and rely on standard displays. However, these visual representation methods are limited in their visual scalability. Yet there is a potential for novel or adapted representation methods to cater the special needs of malware analysis. Problem-driven visualization research thrives from interdisciplinary collaboration with domain experts but needs to start from a solid problem characterization and abstraction as base for design and evaluation [Mun14, SMM12, MA14]. Such research on the requirements for malware visualization can constitute an independent contribution to research (e.g., [WAR*14]).

Involve expert knowledge through interaction: For keeping up with the large number and dynamic evolution of malware families, malware analysts need to continuously adapt the settings of their visualization systems. Interactivity is a key strength of visualization systems allowing domain experts to take other points of view with immediate feedback [Shn96, TC05, KKEM10]. However, most malware analysis systems surveyed here are very limited in this regard – only 13 of 25 system reported any evidence for interaction. Even if these deficits in interaction are a recurrent theme in visualization research, malware analysis in particular can profit from more extensive interaction and annotation features as it is a very knowledge-intensive job. It should even be considered to provide knowledge-oriented interactions allowing to externalize knowledge that can subsequently be used in the analysis process to improve analysts' performance [SSS*14].

Intertwine analytical methods with visualization: Currently most systems build their visual metaphors directly on the output of the data providers and only few systems such as Saxe et al. [SMG12] use additional analytical methods to classify or cluster the data. Following the visual analytics agenda [TC05, KKEM10], analytical methods must be considered alongside visual representation methods for scalable and problem-tailored visualization solutions. Furthermore, analytical methods should not be treated as a black box but should allow adaption by experts through interaction [MPG*14].

8. Conclusion

In this survey we presented the currently used data providers as well as a systematic review of visualization systems for malware analysis. In the first step, we categorized the data providers in regards to their analysis approach, their environment as well as their input and output data formats.

Each analysis system was then assigned to one of the 3 main categories, depending on their general approach to processing and visualization, which were defined in the *Malware Visualization Taxonomy* as presented in Figure 2. We also categorized these systems by their input files and formats, the visualization techniques utilized, the representation space and the mapping to time, certain temporal aspects, their interactive capabilities, and the different types of available user actions. Many of the surveyed systems gather analysis data internally and base their analysis on the sample's binary code. Others use external data providers to retrieve specific properties. In terms of visualization techniques we discovered that stacked displays and iconic displays are not commonly used in the malware domain; most tools utilize static 2D displays to support the analyst. Dynamic or 3D representations are rare – only 4 of the explored systems are able to map data to a 3D representation space. Regarding the used representation space and the mapping to time, we found out that most of the systems use a static mapping. On the temporal side we determined that only 12 out of 25 analysis systems consider time at all. All time-aware systems use a linear arrangement. 11 out of these 12 tools use an ordinal timescale and one uses a discrete one. Most of the tools use the *instant* time primitive. Only one tool uses the *interval* primitive and one other tool uses *instant* and *interval* primitives. In relation to the granularities, only 2 tools out of 12 are using these primitives, whereby one tool uses a single granularity and one tool uses multiple granularities. Surprisingly, only 13 of the surveyed systems support interaction while the remainder relies solely on non-interactive representations. Of the available user actions, *discover*, *present* and *compare* operations are the most common. It is interesting to see that the identification of specific malware samples is usually not a priority. All the results of this survey are publicly available for interactive exploration on our supplementary website found at http://malware.dbvis.de/.

Furthermore, we defined various future challenges and perspectives in Section 7 to further improve visual analytics for malware analysis to eventually help to enhance cyber security.

Acknowledgments

This work was supported by the Austrian Science Fund (FWF) via the KAVA-Time project (P25489) and the Austrian Federal Ministry for Transport, Innovation and Technology via KIRAS project (836264). Additionally, it was partially supported by the DFG Priority Programme "Scalable Visual Analytics: Interactive Visual Analysis Systems of Complex Information Spaces" (SPP 1335).

References

[Alv15] ALVAREZ V. M.:. YARA – the pattern matching swiss knife for malware researchers [online]. 2015. URL: http://plusvic.github.io/yara/ [cited 2015-04-07]. 5

[AMST11] AIGNER W., MIKSCH S., SCHUMANN H., TOMINSKI C.: *Visualization of time-oriented data*. Human-computer interaction series. Springer, 2011. 10, 12, 13

[And72] ANDREWS D. F.: Plots of high-dimensional data. *Biometrics 28*, 1 (1972), 125–136. doi:10.2307/2528964. 11

[ASL12] ANDERSON B., STORLIE C., LANE T.: Improving malware classification: Bridging the static/dynamic gap. In *Proc. 5th ACM Workshop on Security and Artificial Intelligence, AISec* (2012), ACM, pp. 3–14. doi:10.1145/2381896.2381900. 6, 7, 10, 11, 12, 13, 14, 15, 16

[Aut14] AUTOIT CONSULTING LTD.:. Autoit [online]. 2014. URL: https://www.autoitscript.com/site/autoit/ [cited 2014-12-29]. 5

[Bat14] BATRA R.:. API Monitor [online]. 2014. URL: http://www.rohitab.com/apimonitor [cited 2014-12-29]. 5

[BBK11] BHUYAN M. H., BHATTACHARYYA D., KALITA J.: Surveying port scans and their detection methodologies. *The Computer Journal 54*, 10 (Oct. 2011), 1565–1581. doi:10.1093/comjnl/bxr035. 3

[Bel05] BELLARD F.: QEMU, a fast and portable dynamic translator. In *Proc. USENIX Annual Technical Conf., ATEC* (2005), USENIX Association, Berkeley, CA, USA, pp. 41–46. 3, 5

[BHDA14] BOU-HARB E., DEBBABI M., ASSI C.: Cyber scanning: A comprehensive survey. *IEEE Communications Surveys Tutorials 16*, 3 (2014), 1496–1519. doi:10.1109/SURV.2013.102913.00020. 3

[BHFH13] BAZRAFSHAN Z., HASHEMI H., FARD S., HAMZEH A.: A survey on heuristic malware detection techniques. In *5th Conf. Information and Knowledge Technology, IKT* (2013), pp. 113–120. doi:10.1109/IKT.2013.6620049. 2, 17

[BKK06] BAYER U., KRUEGEL C., KIRDA E.: TTAnalyze: A tool for analyzing malware. In *Proc. 15th Annual Conf. European Institute for Computer Antivirus Research, EICAR* (2006). 5

[BM13] BREHMER M., MUNZNER T.: A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2376–2385. doi:10.1109/TVCG.2013.124. 15

[BMKK06] BAYER U., MOSER A., KRUEGEL C., KIRDA E.: Dynamic analysis of malicious code. *Journal in Computer Virology 2*, 1 (2006), 67–77. 5

[CDSS08] CONTI G., DEAN E., SINDA M., SANGSTER B.: Visual reverse engineering of binary and data files. In *Visualization for Computer Security, Proc. VizSec* (2008), Goodall J. R., Conti G., Ma K.-L., (Eds.), LNCS 5210, Springer, pp. 1–17. doi:10.1007/978-3-540-85933-8_1. 7, 8, 11, 12, 13, 14, 15, 16

[Che73] CHERNOFF H.: The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association 68*, 342 (1973), 361–368. doi:10.1080/01621459.1973.10482434. 11

[Con07] CONTI G.: *Security data visualization: graphical techniques for network analysis*. No Starch Press, 2007. 2

[Die07] DIEHL S.: *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, Berlin, 2007. 2

[DKLT14] DORNHACKL H., KADLETZ K., LUH R., TAVOLATO P.: Malicious behavior patterns. In *Proc. IEEE 8th Int. Symp. Service Oriented System Engineering* (2014), pp. 384–389. doi:10.1109/SOSE.2014.52. 1, 5, 17

[DPM13] DONAHUE J., PATURI A., MUKKAMALA S.: Visualization techniques for efficient malware detection. In *Proc. 2013*

IEEE Int. Conf. Intelligence and Security Informatics, ISI (2013), pp. 289–291. doi:10.1109/ISI.2013.6578845. 7, 8, 11, 12, 13, 14, 15, 16

[DSF*14] DUARTE F., SIKANSI F., FATORE F., FADEL S., PAULOVICH F.: Nmap: A novel neighborhood preservation space-filling algorithm. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2063–2071. doi:10.1109/TVCG.2014.2346276. 11

[ESKK12] EGELE M., SCHOLTE T., KIRDA E., KRUEGEL C.: A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys 44*, 2 (2012), 6:1–6:42. doi:10.1145/2089125.2089126. 2, 3

[FFC*11] FELT A. P., FINIFTER M., CHIN E., HANNA S., WAGNER D.: A survey of mobile malware in the wild. In *Proc. 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM* (2011), ACM, pp. 3–14. doi:10.1145/2046614.2046618. 3

[Fir13] FIREEYE INC.: *FireEye MAS 6.4.0 Operator's Guide*, 2013. 5

[Fir14] FIREEYE INC.:. FireEye malware analysis [online]. 2014. URL: https://www.fireeye.com/products/malware-analysis.html [cited 2014-12-29]. 5

[GBA*12] GRÉGIO A. R. A., BARUQUE A. O. C., AFONSO V. M., FILHO D. S. F., GEUS P. L. D., JINO M., SANTOS R. D. C. D.: Interactive, visual-aided tools to analyze malware behavior. In *Computational Science and Its Applications, ICCSA*, LNCS 7336. Springer, 2012, pp. 302–313. doi:10.1007/978-3-642-31128-4_22. 7, 8, 11, 12, 13, 14, 15, 16

[Ge14] GUARNIERI C., ET AL.: *Cuckoo Sandbox Book*, Release 1.2-dev ed., 2014. 5

[Gol74] GOLDBERG R. P.: Survey of virtual machine research. *Computer 7*, 6 (1974), 34–45. 3

[GS11] GRÉGIO A. R. A., SANTOS R. D. C.: Visualization techniques for malware behavior analysis. In *Proc. Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense X* (2011), vol. 8019, SPIE, pp. 801905:1–801905:9. doi:10.1117/12.883441. 7, 8, 11, 12, 13, 14, 15, 16

[GSG*14] GOVE R., SAXE J., GOLD S., LONG A., BERGAMO G.: SEEM: A scalable visualization for comparing multiple large sets of attributes for malware analysis. In *Proc. 11th Workshop on Visualization for Cyber Security, VizSec* (2014), ACM. doi:10.1145/2671491.2671496. 7, 8, 9, 11, 12, 13, 14, 15, 16

[HKI14] HAN K., KANG B., IM E. G.: Malware analysis using visualized image matrices. *The Scientific World Journal 2014* (2014), 15. doi:10.1155/2014/132713. 6, 7, 10, 11, 12, 13, 14, 15, 16

[HLI13] HAN K., LIM J. H., IM E. G.: Malware analysis method using visualization of binary files. In *Proc. Research in Adaptive and Convergent Systems, RACS* (2013), pp. 317–321. doi:10.1145/2513228.2513294. 6, 7, 10, 11, 12, 13, 14, 15, 16

[HLKI14] HAN K. S., LIM J. H., KANG B., IM E. G.: Malware analysis using visualized images and entropy graphs. *Int. Journal of Information Security* (2014), 1–14. doi:10.1007/s10207-014-0242-0. 6, 7, 9, 11, 12, 13, 14, 15, 16

[IM07] IDIKA N., MATHUR A. P.: A survey of malware detection techniques. Technical Report, Purdue University, 2007. 3

[Iva02] IVANOV I.: API hooking revealed. *The Code Project* (2002). 5

[Joe14] JOE SECURITY LLC:. JoeSandbox [online]. 2014. URL: http://www.joesecurity.org [cited 2014-12-29]. 5

[Kei02] KEIM D.: Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 1–8. doi:10.1109/2945.981847. 10, 11, 12

[KKEM10] KEIM D., KOHLHAMMER J., ELLIS G., MANSMANN F. (Eds.): *Mastering the information age: solving problems with visual analytics*. Eurographics Association, 2010. 1, 17

[KM07] KENDALL K., MCMILLAN C.: *Practical malware analysis*. Mandiant, 2007. 3

[KM13] KANCHERLA K., MUKKAMALA S.: Image visualization based malware detection. In *Proc. 2013 IEEE Symp. Computational Intelligence in Cyber Security, CICS* (2013), pp. 40–44. doi:10.1109/CICYBS.2013.6597204. 7, 9, 11, 12, 13, 14, 15, 16

[KMS*08] KEIM D. A., MANSMANN F., SCHNEIDEWIND J., THOMAS J., ZIEGLER H.: Visual Analytics: Scope and challenges. In *Visual Data Mining*, Simoff S. J., Böhlen M. H., Mazeika A., (Eds.), LNCS 4404. Springer, Berlin, 2008, pp. 76–90. doi:10.1007/978-3-540-71080-6_6. 14

[Lin14] LINUX FOUNDATION:. Xen project [online]. 2014. URL: http://www.xenproject.org [cited 2014-12-29]. 3

[LMP05] LANZENBERGER M., MIKSCH S., POHL M.: Exploring highly structured data: a comparative study of stardinates and parallel coordinates. In *Proc. Int. Conf. Information Visualisation* (2005), pp. 312–320. doi:10.1109/IV.2005.49. 11

[LSG14] LONG A., SAXE J., GOVE R.: Detecting malware samples with similar image sets. In *Proc. 11th Workshop on Visualization for Cyber Security, VizSec* (2014), ACM. doi:10.1145/2671491.2671500. 7, 8, 11, 12, 13, 14, 15, 16

[LSKJ11] LEE D., SONG I. S., KIM K., JEONG J.-H.: A study on malicious codes pattern analysis using visualization. In *Proc. Int. Conf. Information Science and Applications, ICISA* (2011), pp. 1–5. doi:10.1109/ICISA.2011.5772330. 1, 17

[LWW90] LEBLANC J., WARD M., WITTELS N.: Exploring n-dimensional databases. In *Proc. 1st IEEE Conf. Visualization, Vis* (1990), pp. 230–237. doi:10.1109/VISUAL.1990.146386. 11

[MA14] MIKSCH S., AIGNER W.: A matter of time: Applying a data–users–tasks design triangle to visual analytics of time-oriented data. *Computers & Graphics 38* (2014), 286–290. doi:10.1016/j.cag.2013.11.002. 17

[Mic99] MICROSOFT CORPORATION: *Microsoft Portable Executable and Common Object File Format Specification*. Microsoft Corporation, 1999. 4

[MPG*14] MÜHLBACHER T., PIRINGER H., GRATZL S., SEDLMAIR M., STREIT M.: Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1643–1652. doi:10.1109/TVCG.2014.2346578. 17

[Mun14] MUNZNER T.: *Visualization Analysis and Design*. A K Peters Ltd, Boca Raton, 2014. 10, 15, 16, 17

[NKJM11] NATARAJ L., KARTHIKEYAN S., JACOB G., MANJUNATH B. S.: Malware images: Visualization and automatic classification. In *Proc. 8th Int. Symp. Visualization for Cyber Security, VizSec* (2011), ACM, pp. 4:1–4:7. doi:10.1145/2016904.2016908. 7, 9, 11, 12, 13, 14, 15, 16

[NYPZ11] NATARAJ L., YEGNESWARAN V., PORRAS P., ZHANG J.: A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In *Proc. 4th ACM Workshop on Security and Artificial Intelligence, AISec* (2011), pp. 21–30. doi:10.1145/2046684.2046689. 9

[Ora14] Oracle Corporation:. Oracle VirtualBox [online]. 2014. URL: https://www.virtualbox.org [cited 2014-12-29]. 3

[Pan08] Panas T.: Signature visualization of software binaries. In *Proc. 4th ACM Symp. Software Visualization, SoftVis* (2008), pp. 185–188. doi:10.1145/1409720.1409749. 6, 7, 9, 11, 12, 13, 14, 15, 16

[Pan14] PandaLab: *Quarterly Report*. Tech. rep., 2014. 1

[PCDM13] Paturi A., Cherukuri M., Donahue J., Mukkamala S.: Mobile malware visual analytics and similarities of attack toolkits (malware gene analysis). In *Proc. Int. Conf. Collaboration Technologies and Systems, CTS* (2013), pp. 149–154. doi:10.1109/CTS.2013.6567221. 7, 10, 11, 12, 13, 14, 15, 16

[PG98] Pickett R., Grinstein G.: Iconographic displays for visualizing multidimensional data. In *Proc. 1988 IEEE Int. Conf. Systems, Man, and Cybernetics* (1998), vol. 1, pp. 514–519. doi:10.1109/ICSMC.1988.754351. 11

[QL09] Quist D., Liebrock L.: Visualizing compiled executables for malware analysis. In *Proc. 6th Int. Workshop on Visualization for Cyber Security, VizSec* (2009), pp. 27–32. doi:10.1109/VIZSEC.2009.5375539. 6, 7, 8, 11, 12, 13, 14, 15, 16

[QL11] Quist D. A., Liebrock L. M.: Reversing compiled executables for malware analysis via visualization. *Information Visualization 10*, 2 (2011), 117–126. doi:10.1057/ivs.2010.11. 6, 7, 8, 11, 12, 13, 14, 15, 16

[RC14] Russinovich M., Cogswell B.:. Process monitor v3.1 [online]. 2014. URL: http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx [cited 2014-12-29]. 5

[Roy15] Roy Rosenzweig Center for History and New Media:. dev:web_api:v3:basics [zotero documentation] [online]. 2015. URL: https://www.zotero.org/support/dev/web_api/v3/basics [cited 2015-04-07]. 6

[RSI12] Russinovich M., Solomon D., Ionescu A.: *Windows Internals, Part 1: Covering Windows Server 2008 R2 and Windows 7*, 6th ed. Microsoft Press, 2012. 4, 5

[SH12] Sikorski M., Honig A.: *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*, 1 ed. No Starch Press, 2012. 1

[Shn96] Shneiderman B.: The eyes have it: a task by data type taxonomy for information visualizations. In *Proc. IEEE Symp. Visual Languages* (1996), pp. 336–343. doi:10.1109/VL.1996.545307. 14, 17

[SM14a] Shaid S., Maarof M.: Malware behavior image for malware variant identification. In *Proc. 2014 Int. Symp. on Biometrics and Security Technologies, ISBAST* (2014), pp. 238–243. doi:10.1109/ISBAST.2014.7013128. 6, 7, 9, 11, 12, 13, 14, 15, 16

[SM14b] Shaid, S.Z.M., Maarof, M.A.: Malware behaviour visualization. *Jurnal Teknologi 70*, 5 (2014), 25–33. doi:10.11113/jt.v70.3512. 6, 7, 9, 11, 12, 13, 14, 15, 16

[SMG12] Saxe J., Mentis D., Greamo C.: Visualization of shared system call sequence relationships in large malware corpora. In *Proc. 9th Int. Symp. Visualization for Cyber Security, VizSec* (2012), ACM, pp. 33–40. doi:10.1145/2379690.2379695. 7, 8, 11, 12, 13, 14, 15, 16, 17

[SMM12] Sedlmair M., Meyer M., Munzner T.: Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2431–2440. doi:10.1109/TVCG.2012.213. 17

[SSG12] Shiravi H., Shiravi A., Ghorbani A.: A survey of visualization systems for network security. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1313–1329. doi:10.1109/TVCG.2011.144. 2

[SSS*14] Sacha D., Stoffel A., Stoffel F., Kwon B. C., Ellis G., Keim D.: Knowledge generation model for visual analytics. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1604–1613. doi:10.1109/TVCG.2014.2346481. 17

[SWL08] Siddiqui M., Wang M. C., Lee J.: A survey of data mining techniques for malware detection using file features. In *Proc. 46th Ann. Southeast Regional Conference on XX* (2008), ACM-SE 46, pp. 509–510. doi:10.1145/1593105.1593239. 2

[TC05] Thomas J. J., Cook K. A. (Eds.): *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE, 2005. 1, 2, 14, 17

[THGF09] Trinius P., Holz T., Gobel J., Freiling F.: Visual analysis of malware behavior using treemaps and thread graphs. In *Proc. 6th Int. Workshop on Visualization for Cyber Security, VizSec* (2009), pp. 33–38. doi:10.1109/VIZSEC.2009.5375540. 7, 8, 11, 12, 13, 14, 15, 16

[VMW14] VMWare Inc.:. Vmware workstation [online]. 2014. URL: http://www.vmware.com/products/workstation/features.html [cited 2014-12-29]. 3

[WAR*14] Wagner M., Aigner W., Rind A., Dornhackl H., Kadletz K., Luh R., Tavolato P.: Problem characterization and abstraction for visual analytics in behavior-based malware pattern analysis. In *Proc. 11th Workshop on Visualization for Cyber Security, VizSec* (2014), ACM, pp. 9–16. doi:10.1145/2671491.2671498. 1, 17

[Weg97] Wegner P.: Why interaction is more powerful than algorithms. *Communications of the ACM 40*, 5 (1997), 80–91. doi:10.1145/253769.253801. 2

[WHF07] Willems C., Holz T., Freiling F.: Toward automated dynamic malware analysis using CWSandbox. *IEEE Security and Privacy 5*, 2 (2007), 32–39. doi:10.1109/MSP.2007.45. 5

[WPO14] Wüchner T., Pretschner A., Ochoa M.: Davast: Data-centric system level activity visualization. In *Proc. 11th Workshop on Visualization for Cyber Security, VizSec* (2014), ACM, pp. 25–32. doi:10.1145/2671491.2671499. 7, 8, 11, 12, 13, 14, 15, 16

[WY13] Wu Y., Yap R. H. C.: Experiments with malware visualization. In *Proc. 9th Int. Conf. Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA* (2013), Flegel U., Markatos E., Robertson W., (Eds.), LNCS 7591, Springer, pp. 123–133. doi:10.1007/978-3-642-37300-8_7. 7, 9, 11, 12, 13, 14, 15, 16

[YCIZ12] Yee C. L., Chuan L. L., Ismail M., Zainal N.: A static and dynamic visual debugger for malware analysis. In *Proc. 18th Asia-Pacific Conf. Communications, APCC* (2012), pp. 765–769. doi:10.1109/APCC.2012.6388211. 7, 8, 11, 12, 13, 14, 15, 16

[Yoo04] Yoo I.: Visualizing windows executable viruses using self-organizing maps. In *Proc. ACM Workshop on Visualization and Data Mining for Computer Security, VizSec* (New York, NY, USA, 2004), ACM, pp. 82–89. doi:10.1145/1029208.1029222. 7, 10, 11, 12, 13, 14, 15, 16

[ZN12] Zhuo W., Nadjin Y.: MalwareVis: Entity-based visualization of malware network traces. In *Proc. 9th Int. Symp. Visualization for Cyber Security, VizSec* (2012), ACM, pp. 41–47. doi:10.1145/2379690.2379696. 7, 8, 11, 12, 13, 14, 15, 16

Biography

Markus Wagner is a research associate at the Institute of Creative\Media/Technologies, St. Poelten University of Applied Sciences (Austria) and a Ph.D. student at the Faculty of Informatics of Vienna University of Technology. His research activities involve knowledge-assisted visual analytics methods for behavior-based malware analysis. Markus studied Game Engineering and Simulation at the University of Applied Sciences Technikum Wien where he received his M.Sc. degree in 2013 and Industrial Simulation at St. Poelten University of Applied Sciences where he received his B.Sc. degree in 2011.

Fabian Fischer is a research associate and Ph.D. student at the Computer Science Department of the University of Konstanz (Germany). He has received his M.Sc. degree in Information Engineering by the same university in 2011, where he also received his B.Sc. in 2008. Since 2008, he published many papers in the field of visual analytics, especially with respect to network security and threat detection. In 2013 and 2014, he was part of the organizing committee as poster chair of VizSec, which is an IEEE VIS workshop focusing specifically on visualization for cyber security.

Robert Luh is a research associate at the Institute of IT Security Research at St. Poelten University of Applied Sciences (Austria). He earned his master's degree in Information Security in 2013 and is currently starting his Ph.D. at DeMontfort University in Leicester (UK). Robert's current research revolves around the analysis and interpretation of malicious behavior in software.

Andrea Haberson is a student researcher at the Institute of Creative\Media/Technologies, St. Poelten University of Applied Sciences (Austria).

Alexander Rind is a research associate at the Institute of Creative\Media/Technologies, St. Poelten University of Applied Sciences (Austria) and lecturer at Vienna University of Technology. His research activities involve knowledge-assisted visual analytics methods for behavior-based malware analysis, interaction in visual analytics, and visualization of electronic health records. Alexander studied Business Informatics in Vienna and Lund and received his M.Sc. degree from Vienna University of Technology in 2004.

Daniel Keim is a full professor and head of the Information Visualization and Data Analysis Research Group in the University of Konstanz's Computer Science Department. Keim received a habilitation in computer science from the University of Munich. He has been program cochair of the IEEE Information Visualization Conference, the IEEE Conference on Visual Analytics Science and Technology (VAST), and the ACM SIGKDD Conference on Knowledge Discovery and Data Mining. He is on the steering committees of IEEE VAST and the Eurographics Conference on Visualization.

Wolfgang Aigner is a professor at St. Poelten University of Applied Sciences (Austria). His main research interests include visual analytics and information visualization. Wolfgang Aigner received his Ph.D. degree and habilitation from Vienna University of Technology in 2006 and 2013. He authored and co-authored several dozens of peer-reviewed articles as well as the book "Visualization of Time-Oriented Data" (Springer, 2011) that is devoted to a systematic view on this topic.