An Inside Look into the Practice of Malware Analysis

Miuyin Yong Wong[†], Matthew Landen[†], Manos Antonakakis[†], Douglas M. Blough[†], Elissa M. Redmiles[‡], Mustaque Ahamad[†]
Georgia Institute of Technology[†]
United States
Max Planck Institute for Software Systems[‡]
Germany

ABSTRACT

Malware analysis aims to understand how malicious software carries out actions necessary for a successful attack and identify the possible impacts of the attack. While there has been substantial research focused on malware analysis and it is an important tool for practitioners in industry, the overall malware analysis process used by practitioners has not been studied. As a result, an understanding of common malware analysis workflows and their goals is lacking. A better understanding of these workflows could help identify new research directions that are impactful in practice. In order to better understand malware analysis processes, we present the results of a user study with 21 professional malware analysts with diverse backgrounds who work at 18 different companies. The study focuses on answering three research questions: (1) What are the different objectives of malware analysts in practice?, (2) What comprises a typical professional malware analyst workflow?, and (3) When analysts decide to conduct dynamic analysis, what factors do they consider when setting up a dynamic analysis system?

Based on participant responses, we propose a taxonomy of malware analysts and identify five common analysis workflows. We also identify challenges that analysts face during the different stages of their workflow. From the results of the study, we propose two potential directions for future research, informed by challenges described by the participants. Finally, we recommend guidelines for developers of malware analysis tools to consider in order to improve the usability of such tools.

CCS CONCEPTS

• **Security and privacy → Usability in security and privacy**.

KEYWORDS

Malware Analysis;Usable Security

ACM Reference Format:
Miuyin Yong Wong[†], Matthew Landen[†], Manos Antonakakis[†], Douglas M. Blough[†],, Elissa M. Redmiles[‡], Mustaque Ahamad[†] . 2021. An Inside Look into the Practice of Malware Analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3460120.3484759

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3484759

1 INTRODUCTION

The growing volume and sophistication of cyber attacks relies on the ability of malicious actors to take control of victim computers, often by running malicious software on them. Such malicious software, or malware, can be used to exfiltrate sensitive data (e.g., data breaches) or to demand ransom as seen in numerous recent attacks [6, 16]. To combat such attacks, we must understand how they were conducted. To do so, professional malware analysts typically analyze malware samples to understand what a certain malware instance does and how it carries out the actions required for a successful attack.

There has been considerable research in the area of malware analysis that aims to understand malware and develop defenses against it [28, 29, 33, 35–37, 39, 42, 45, 47, 51, 52, 54–56, 61, 66–68, 73, 78, 82]. Such analysis may range from limited static analysis, such as collecting hashes or extracting strings, to a detailed analysis that can help understand the specific tactics used by malware to achieve its goals [24, 64]. There are two main categories of malware analysis; dynamic and static. Dynamic analysis consists of executing a potentially malicious program in a controlled setting and monitoring its actions. In contrast, static analysis techniques are performed without actually executing the sample, but rather focus on analyzing the malicious code. There are also hybrid approaches that leverage both dynamic and static analysis to analyze malware.

Although a significant body of research is devoted to developing new malware analysis techniques [24, 25, 31, 36, 42, 50, 52, 67, 69, 79, 80], and these techniques are used widely, there is limited research on understanding how they are used in practice. Prior work has examined the workflows of software testers and white hat hackers [75] as well as the workflows of reverse engineers [76]. Recently, Votipka et al. presented an analysis of the workflow of reverse engineers by conducting an observational study [76]. The study found that reverse engineers mostly rely on static analysis and utilize some dynamic analysis in later stages of their workflow.

As discussed in prior research, malware analysis spans a broad space of techniques, leveraging a combination of dynamic and static analysis. Since reverse engineering primarily relies on static analysis, the overall process of malware analysis, including important steps associated with dynamic analysis, were not explored in detail in past research studies. To fill this gap, we focus on understanding the process of malware analysis broadly, including how analysis

goals and different methods of analysis (i.e., static and dynamic) impact this process. To do so, we conduct a semi-structured interview study with a diverse group of 21 professional malware analysts from 18 different companies, including Mandiant, Cisco, IBM, and a large financial institution. Although the analysts cannot disclose their company's proprietary information, we gain insights from their personal workflows, which have been shaped by the knowledge that they have accumulated while working in the security industry. Our work seeks to answer the following research questions:

RQ1. What are the different objectives of malware analysts in practice?

RQ2. What comprises a typical professional malware analyst workflow?

RQ3. When analysts conduct dynamic analysis, what factors do they consider when setting up a dynamic analysis system?

Our contributions are divided into three main areas. First, we propose a taxonomy that classifies malware analysts based on the indicators of compromise [1] (IOCs) that the analysts extract in order to achieve their objectives. Second, we identify five common malware analysis workflows. Third, we identify six key decisions that malware analysis practitioners make when setting up their dynamic analysis systems: We detail participants' choices of implementation, virtual analysis platform, environment setup, network communication, execution time, and techniques used to overcome evasive tactics deployed by some malware.

From the workflows, we find that the participant's main objective significantly alters the methodology used to analyze the malware sample. We also observe that as the complexity of a malware sample increases, the likelihood that analysts will require a hybrid approach also increases. The majority of the participants begin with either dynamic analysis or limited static analysis. Later, some participants switch to more in-depth static analysis; either to derive additional information to reconfigure their dynamic analysis systems or to manually reverse engineer the sample. Additionally, this user study allowed us to observe open challenges faced by professional malware analysts. We discuss potential opportunities for translating previous research into practical solutions for practitioners as well as future research directions that could help address the identified challenges.

This paper is structured as follows. Section 2 describes the methodology of our study and section 3 provides details of the participants. Next, we present our taxonomy used to categorize participants in section 4. Section 5 describes the different analysis workflows that we identified and section 6 discusses the configuration decisions made by analysts. Lastly, we present challenges faced by the participants, offer potential ways to help address them, and provide usability recommendations in section 7, and conclude in section 9.

2 METHODOLOGY

In order to answer our research questions, we conducted 21 semi-structured interviews with professional malware analysts. This section describes the recruitment, interview, and data analysis procedures, as well as the limitations of our work. This study was approved by our university's Institutional Review Board.

[1] IOCs are artifacts that indicate potentially malicious activities that analysts seek by analyzing a malware sample [13].

2.1 Recruitment

Three sources were used to recruit professional malware analysts: a curated panel of security professionals who made themselves available for surveys, mailing lists of security organizations, and personal contacts of the co-authors of this paper. In April 2020, an email requesting participation was sent to all three sources containing a description of the study and an explanation of the steps that the participants would have to take to participate.

Participant Selection. Although we only shared the study with people in the cybersecurity field, we needed to ensure that we specifically selected malware analysts for this user study. As such, potential participants completed a 20 question survey (see Appendix A) that collected basic demographic information as well as information regarding participants' area of expertise, technical skills, and job tasks. Among those who fit the purpose of the study, we selected participants who captured a broad range of skill levels, job titles, and job tasks within malware analysis.

2.2 Interview Protocol

We conducted hour-long, semi-structured interviews with each participant via online video conference from May through June 2020. All interviews were conducted by the first author for consistency. The interviewer followed the questions found in Appendix B with the option to ask follow-up questions and skip previously answered questions. The interviews were divided into the following sections.

Introduction and Experience. The interview began by asking the participants to expand on their daily job tasks and experience described in the screening survey. Their responses allowed us to personalize questions during the rest of the interview.

Malware Sources. Next, to better understand our participants' day-to-day work experiences, we asked them questions about the malware samples they analyze. Specifically, we asked how they receive their malware samples, what data is included when they get a sample, and how many samples they receive per day. We also asked how the participants prioritized the samples they analyze and how they detect whether a sample is a variant of a previously known malware.

Analysis Workflow. In this next section, we asked participants to walk us through the steps they take and tools they use to analyze a malware sample. With these questions, we wanted to understand the process that participants use to achieve their analysis objectives. More generally, we wanted to understand the output of their analysis process. Also, we wanted to identify common challenges that malware analysts face during their analysis workflow.

Dynamic Analysis System Configuration. Dynamic analysis is a common process used by a majority of malware analysts. For this reason, we wanted to understand how the participants configure their dynamic analysis system. We asked participants whether they prefer bare metal or virtualized environments[2], commercial or open-source sandboxes[3], and their reasoning behind such preferences. We also asked what operating system (OS) they select when configuring their sandbox, how they configure their network, and

[2] Bare metal refers to physical computer whereas a virtualized environment uses software to simulate a physical computer.

[3] sandboxes are tools that execute and monitor malware samples safely.

whether they configure the user space with specific locations, languages, and/or applications. Our questions aimed to identify any critical settings that our participants consider when setting up their dynamic analysis systems to get malware to reveal its malicious behavior.

After understanding participants' configuration process, we investigated the dynamic analysis process they follow. Specifically, we asked questions regarding the execution of the malware samples. For example: How long and how many times do they run a sample? If they run samples multiple times, do they modify anything in the setup between runs? What data does the execution produce? Lastly, we wanted to identify the challenging and time-consuming tasks that the participants face when performing this setup.

Malware Analysis Evolution. Finally, we asked each participant to describe how their malware analysis approach has evolved over time.

2.3 Data Collection and Analysis

The interviews were recorded and then manually transcribed by the interviewer and analyzed using an iterative open coding methodology [72]. The interviewer, along with another co-author, developed a codebook (Appendix D) based on an independent review of four interviews. Subsequently, the two co-authors independently coded each interview using this codebook. The coders achieved "excellent" Krippendorff's alpha intercoder reliability of 0.823 [57]. Reliability was used to ensure consistency in the interpretation of conceptual themes and workflows across the research team.

2.4 Follow-up Survey

The last step of our methodology was a follow-up survey, sent in September 2020, consisting of three multiple choice questions, shown in Appendix C. The purpose of this survey was to clarify whether the participants configure specific dynamic analysis settings. Further clarification was needed because some participants provided high-level responses in the initial interview. When presenting results about the dynamic analysis environment setup process, we only include findings from participants who responded to the follow-up survey in order to ensure the results are complete.

2.5 Limitations

Our study has limitations common in exploratory, qualitative research. First, participants may not recall all of the steps of their personal analysis and configuration processes. This limitation is common with studies involving expert tasks [43]. We aimed to mitigate this limitation by asking participants to walk us through their analysis process, per best practice for qualitative interviews [46]. The second limitation is participants' inability to respond to certain questions due to non-disclosure agreements they have with their companies. To mitigate this issue, we asked participants to describe their personal analysis process instead of describing the confidential processes of their companies. Finally, it is possible that our participant group, as a whole, may not cover all of the types of analysts in practice. To partially mitigate this limitation, we recruited participants through several sources to have a diverse group of participants and increase the likelihood that relevant ideas would be stated by more than one participant.

ID	Educ	Yrs	Title	Type of Industry
P1	Ph.D.	5	Director of Research	-
P2	-	14.5	Project Manager - Security Solution Architect	Technology
P3	B.S.	4	Security Engineer	Technology
P4	B.S.	2	Senior Reverse Engineer	Technology
P5	Ph.D.	5	Head of Laboratory	Education
P6	M.S.	5	Senior Analyst	Cybersecurity
P7	Assoc.	10	Principal Threat Researcher / Reverse Engineer	Technology
P8	B.S.	3	Senior Security Engineer	Technology
P9	B.S.	7	Manager, Digital Forensics and Incident Response	Technology
P10	M.S.	2	Malware Research Engineer	Technology
P11	M.S.	6	Sr. Security Analyst	Postal services, Logistics, Transportation, Finance
P12	M.S.	1	Malware Reverse Engineer	Technology
P13	M.S.	-	Staff Threat Intelligence Engineer	Technology
P14	B.S.	4	Computer Information Research Scientist Senior	Technology
P15	College	12	Lead Security Researcher	Technology
P16	B.S.	5	Threat Researcher	Consulting
P17	M.S.	15	Security Researcher	Technology
P18	B.S.	8	Sr. Security Researcher	Technology
P19	M.S.	7	Vice President	Finance
P20	M.S.	10	Malware Researcher	Technology
P21	B.S.	3	Software Engineer Senior	Technology

Table 1: Participants in the study

3 PARTICIPANTS

We received 46 responses to our screening survey and invited 39 of them for an interview. 21 out of these 39 participants responded and participated in an interview. Table 1 shows a breakdown of the participants' education, years of experience with malware analysis, job title, and work industry. Participants have an average of 6 years of experience, with experience levels ranging from 1 year to 15 years. The majority of the participants have a college degree: 8 have a bachelor's degree, 8 a master's degree, and two hold doctorate degrees. Although 18 of the participants have a college degree, many of the participants were not formally taught the skills required for malware analysis. Instead, many of the participants learned how to analyze malware while working in a related field. Although the participants have diverse backgrounds, we observed that the majority have previous experience in either networks, systems, reverse engineering or vulnerability discovery before transitioning into malware analysis. Finally, the participants work for a variety of companies (18 in total) including IBM, Mandiant, Cisco, TwinWave Security, and one of the largest financial institutions.

Tier	ID	Indicators	Main Objective
1	P3	Hash, IP, DN	Blacklist
2	P2	Hash, IP, DN, NHA	Extract Malicious Behavior
	P5	DN, NHA, T	Extract Malicious Behavior
	P8	Hash, IP, DN, NHA	Extract Malicious Behavior
	P10	IP, DN, NHA	Extract Malicious Behavior
	P11	IP, DN, NHA	Extract Malicious Behavior
	P15	NHA	Extract Malicious Behavior
	P16	Hash, IP, DN, NHA	Extract Malicious Behavior
	P21	IP, DN, NHA, T	Extract Malicious Behavior
	P1	IP, DN, NHA	Label Malware Families
	P20	Hash, NHA, T	Label Malware Families
	P4	IP, DN, NHA, T	Generate Report
	P6	Hash, IP, DN, NHA	Generate Report
3	P7	Hash, IP, DN, NHA, T, TTPs	Track TTPs
	P9	Hash,IP, DN, NHA, T, TTPs	Track TTPs
	P12	IP, DN, NHA, T, TTPs	Track TTPs
	P13	IP, DN, NHA, T, TTPs	Track TTPs
	P14	Hash, IP, NHA, T, TTPs	Track TTPs
	P18	IP, DN, NHA, T, TTPs	Track TTPs
	P19	Hash, NHA, T, TTPs	Track TTPs

Table 2: Participants' indicators and tiers. The IOCs required by one tier build upon the IOCs required by the preceding tiers.

4 MALWARE ANALYST TAXONOMY

We address our first research question by analyzing the objectives of our participants. During our analysis, we observed that the high-level objectives of the participants correlated with the IOCs that the participants extract from the malware samples they analyze. Given this observation, we defined a taxonomy that classifies malware analysts into three different groups based on their high-level objectives.

4.1 Malware Analysts' Objectives

We identified six main objectives that our participants discuss. Additionally, we found that participants can have multiple objectives.

Each objective has corresponding IOCs required to achieve them. The security community has defined six IOCs related to malware analysis: hashes, IP addresses (IPs), domain names (DNs), network and host artifacts (NHAs), tools (Ts), and techniques, tactics and procedures (TTPs) [14]. In this subsection, we define each objective our participants reported and identify the corresponding IOCs that are required to achieve the objectives, as shown in Table 2.

Blacklist. Blocking IPs, DNs, and hashes, also known as blacklisting, is the main objective of one of our participants. As P3 explains, the "goal [of blacklisting] is to find out if the malware is causing harm to the client and if it is, then block the file on their blacklists and online through VirusTotal, push blacklists to endpoint devices." The IOCs necessary to achieve the blacklisting objective are hashes, IPs, and DNs.

Extract Malicious Behavior. The most common objective participants have is the extraction of potentially malicious behavior from the malware sample. This objective involves capturing host and network based activities and determining which might be part of a malicious attack. As P10 explains, "part of my role on the team that I'm on is to write behavioral indicators based on the sandbox runs. So, I routinely review indicators from a report and see what we could use to identify malware families or techniques that they use." The participants achieve this objective by monitoring host and network activity such as process behavior and network communication. For example, P6 explains that his analysis is "typically always host or network based. So does it create a mutex? Does it drop any files? Does it modify any files? Does it modify the registry? and what are the characteristics of the file itself? Take the hashes of the file, what type of file it is and the network side would be: Does it communicate with something? If so, what domain, URL, IP address, what type of communication, protocol?"

Label Malware Families. Given that there are multiple versions of malware samples that employ similar attack techniques, also known as malware families, the objective of some participants is to group malware into families to avoid analyzing known malware samples in depth. P1 spends "a lot of time looking at malware and clustering malware based on their network communication similarities. I developed some technologies that use the network communication (http) to detect malware based on how they are communicating over that protocol." Similar to analysts that identify potentially malicious behavior, these analysts extract NHAs to group malware. Additionally, P20 analyzes the tools used by attackers to help exploit their targets. As P20 says, "I then look at the behavior of many of those families, looking at the windows API injections and code similarity, mostly working on doing the aggregations for that."

Generate Report. Oftentimes malware analysts provide reports to their customers or threat intelligence teams with information about the capabilities of a malware attack. P4 provided a brief description saying, "My job is using static and dynamic analysis to analyze the samples and write reports that vary in complexity. Sometimes we do a basic triage of samples and sometimes we do a full exhaustive analysis of samples to send to clients." Similar to the range of information included in the reports these analysts produce, their work requires extracting different IOCs. This can include hashes and IPs as well as analyzing the tools to obtain the capabilities of a sample. When looking at a sample, P12 will "run some tools that pull out metadata and analyze the PE files into the headers, identify if it's a DLL vs. an EXE, is it exporting anything?, does it have any resource that is potentially used to unpack a file?"

Track Tactics, Techniques and Procedures. Another common objective among our participants is to track the indicators of compromise known as TTPs. MITRE defines a set of tactics that cyber attacks have used during attack campaigns and a set of techniques that achieve each tactic [11]. For example, P7 says, "I primarily focus on C2 [Command and Control] protocols for malware. Data structures and algorithms are what I gravitated towards when I first got into malware research. [...] I use those skill sets to group malware families and do more long-term research on the families like Trickbot and Emotet. I use those static skills from looking at their configurations, how they operate, and their C2 protocols to track them over time and when they change." The TTPs that the

analysts extract are often sent to other teams to achieve other objectives such as threat intelligence and incident response. As P12 explains, the team that receives the TTPs he extracts models them and "starts tying the different techniques and building the bigger picture and they might see that different activities are related or from the same campaign and threat actor." Another use for tracking TTPs is to generate proactive defense mechanisms. After reproducing some of the techniques that the malware performs in an isolated environment, P14 can also "actuate [his] test environment in such a way where [he] can turn on certain defense mitigations to see if the malware is still able to perform its intended action."

4.2 Malware Analyst Tiers

After analyzing the participants' objectives, we observed that participants might extract the same set of IOCs when achieving different objectives. As seen in Table 2, participants who extract malicious behavior, label malware families, or generate reports require network and host artifacts. Similarly, the IOC tactics, techniques and procedures is only extracted by participants whose objective is to track TTPs. We used these observations to define a tiered taxonomy that classifies the participants based on the IOCs they extract. Below, we define each tier by the analysts' main objective and required IOCs.

Tier 1 (String-Based IOCs Extraction (SB)). *SB analysts extract easily obtained string-based IOCs such as hashes, domain names, and IP addresses from malware samples to detect when previously seen malware samples re-appear in a large-scale analysis pipeline.*

Tier 2 (Potentially Malicious Activity Identification (PMA)). *PMA analysts focus on identifying potentially malicious activities exhibited by malware samples using network and host artifacts and tools. This analysis process can be leveraged to design methods of malware detection that are more robust than easily modified IOCs, detect different malware samples that exhibit similar activity, or identify the capabilities of malware.*

Tier 3 (Tactics, Techniques, and Procedures Comprehension (TTP)). *TTP analysts perform malware analysis to identify the strategies and intentions behind threat actors' attack campaigns, which is accomplished by understanding the Tactics, Techniques, and Procedures that are used in different pieces of malware and correlating these across multiple attacks.*

After classifying our participants into these tiers, we have 1 participant in tier 1, 12 participants in tier 2, and 7 participants in tier 3.[4] An interesting finding is that we only classified one participant in tier 1. This reinforces recent discussions within the security community about how malware analysis is trending towards behavior-based detection rather than string-based detection because behavior-based methods can protect against undiscovered malware samples and remain effective for longer periods of time [5].

5 MALWARE ANALYST WORKFLOWS

The main results of this study center around a detailed analysis of participants' workflows. In this section, we discuss the different stages of a malware analyst's process and highlight the most common workflows of the participants. Additionally, we elaborate on how participants within the same tier have commonalities within their workflows. The findings of this analysis answer the second research question (RQ2).

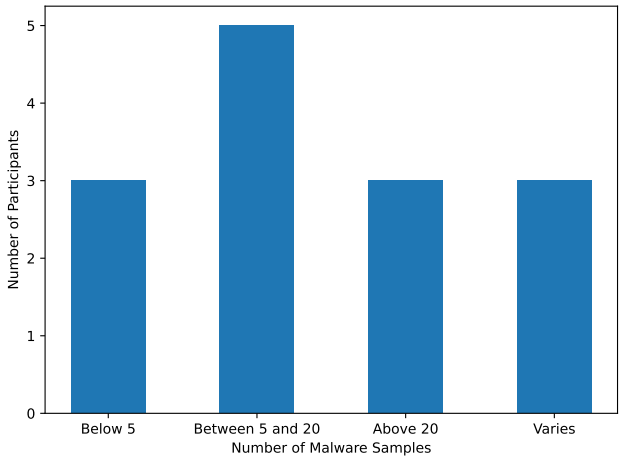


Figure 1: Volume of malware samples analyzed per week

We found that the workflow of a malware analyst consists of three key steps: (1) obtaining malware samples, (2) prioritizing samples for analysis, and (3) in-depth malware analysis when indicated. Next, we describe each of these steps and how they vary between analysts we interviewed.

5.1 Obtaining Malware Samples

The first step in an analyst's workflow is to find malware samples to analyze. These samples can come from a variety of sources including clients, open-source repositories, honeypots, security products, and other internal analysis teams. The tier 1 participant reported getting their samples from clients. The client pays for a service that monitors their perimeter and detects malicious behavior, mainly within the network traffic, emails and their attachments, and files found on the company's devices.

When samples require a more in-depth analysis, they will usually be escalated to a tier 2 analyst. For example, P2 says "Any malware samples that are too complicated to be solved with basic string analysis or requires unpacking usually ends up coming to my team." There are also companies that provide incident response services where clients hire them to investigate and remediate a cyber intrusion. As part of this process, incident responders search the client's enterprise for malware and send any samples they find to tier 2 analysts within the incident response company. Additionally, 7 of the tier 2 participants supplement their samples from clients with additional malware found in repositories such as VirusTotal [2], Reversing Labs [15], Malware Bazaar [10], The Zoo [17], AnyRun [4], Malshare [9], Hybrid Analysis [7], Malpedia [8], and Twitter [18]. One reason for this practice is that analysts want to have a broad range of samples to analyze so that they can generate signatures with a wider coverage.

[4]We were not able to classify P17. During P17's interview, he focused on a subsection of his overall workflow, excluding the details needed to determine his tier.

Participants in tier 3 look for new, potentially harmful malware that requires an in-depth analysis in order to develop preventive measures to protect against these samples. To be truly preventive, these analysts are not able to rely on samples sent from clients because these samples likely come from infections that have already occurred. To obtain interesting samples, 5 participants in tier 3 scrape websites like Pastebins and disposable emails, and look in repositories, similar to the ones mentioned in tier 2.

Although the patterns above are fairly consistent among our participants, we found that the data source is also dependent on the company the analysts work with. As P2 noted, "I have worked in some roles where you don't research very much malware and others where I can research millions of files in a day. So, what I've come to realize is that process is very much indicative of the individual company and the process they have developed." Finally, in Figure 1, we specify the amount of malware samples that the participants analyze per week. As seen in the figure, the majority of the participants analyze anywhere between 5 and 20 malware samples. Just as the data source depends on the company, the amount of malware samples also varies due to the current threat landscape.

5.2 Prioritization

The volume of malware samples has grown exponentially, making it challenging for malware analysts to process all of the samples they see. As a result, analysts must prioritize which samples to analyze. A common method of prioritization used by our participants is to assign a lower priority to samples that are variants of known malware samples. To determine whether a sample is a variant of a known malware family, analysts use static and dynamic methods to scan samples for known IOCs. In static analysis, automated systems generate hashes, and extract strings, including IPs, and domain names. These string-based IOCs can then be cross-referenced with blacklists to identify variants. In dynamic analysis, participants run the samples and search for previously created IOCs that capture activities observed in previous samples. P12 states that they "use things like yara rules or create signatures so that [they] can submit it through [their] sandbox [and] ... figure out what [malware family it belongs to]." If a sample does not match any indicators, it remains in the analyst's queue for further analysis. Lastly, 10 of our participants assign a higher priority to newer malware samples. As P2 said, "When I come in on any given day, or anytime to research malware, I'm really only concerned about the new stuff because malware dies really quickly. It dies within, oftentimes hours, or at most days."

During this process, some analysts may need to examine the malware's code or strings. However, malware samples are frequently encrypted to prevent this type of analysis. Therefore, an important task in the malware analysis workflow is reversing this encryption, commonly referred to as "unpacking." 7 participants utilize dynamic analysis to unpack samples. For example, P14 says "if the malware is packed or obfuscated, you're either gonna spend time reversing the encryption or obfuscation functions and that's a pain. I hate doing that. It's much easier to run it and let it decrypt itself." 2 participants utilize external, automated unpacking services such as UnpacMe [19] to get the unpacked sample. As P10 describes "Another service that I use is called UnpacMe, which is an up and coming malware unpacking service, and it's really awesome. A lot of times, it will get you to the lowest stage executable really quickly that can be really helpful". Lastly, 3 participants also utilize a hybrid approach by debugging the sample mainly with ollydbg [12].

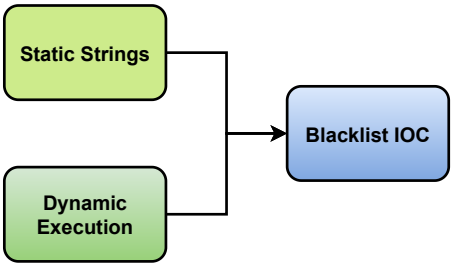


Figure 2: Tier 1 Participant Workflow; P3

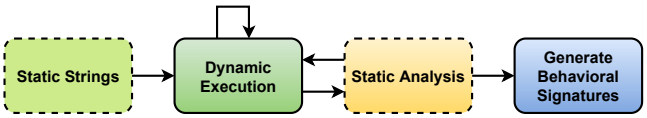
5.3 Main Analysis Process

When an analyst decides to analyze a malware sample in detail, they follow a specific workflow that allows them to reach their objective. To identify the analysis processes of our participants, we studied the overall workflow described in the interviews. Next, we identified commonalities between the workflows of different participants. These commonalities allowed us to merge similar workflows. Finally, we selected the five most common workflows of the participants, shown in Figures 1-3[5], where the blue boxes display the participants' main objective. The rest of this section describes these workflows, including the analysis steps and the thought processes of the participants.

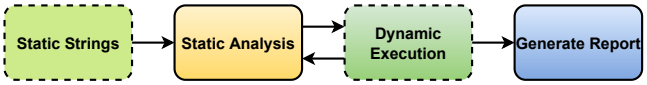
The first workflow is shown in Figure 2. This workflow starts with either a potentially malicious executable or document. P3 analyzes executables by statically extracting strings. This process can be accomplished by running the `strings` command or opening the sample in a disassembler such as IDA. These strings are then used to blacklist the sample. P3 said he "will block it in our network and we will make sure that domain and hash will be blocked in our local firewall." When analyzing a document, P3 opts to perform dynamic analysis on the sample and monitor the results, looking for IOCs such as IPs and domains. These IOCs are put into firewall blacklists to detect previously seen samples.

The second workflow, shown in Figure 3a, is used by participants who utilize some static analysis within their workflow, but emphasize dynamic analysis. During dynamic analysis, participants monitor the environment to identify what actions the sample takes. As P15 mentions, "process tree, file access rewrites, registry, injects creation are the minimum." There are a number of tools that are able to collect this information. One example is P2, who is "always going to run ProcMon and process explorer because I want to know if other processes were created on the fly. I'm always going to run Wireshark because I want to know if network traffic is going to be done, snapshot of my box because I want to know what changes have been made, and when I'm done I run Clonezilla to revert." This information is used to create behavioral signatures, which P10 explains are "able to match on what the malware does like what files that it writes or registry keys that it creates, sets or modifies.

[5]P5 and P21 did not follow any of the five most common workflows and, as such, are not listed in the figures.



(a) Tier 2A Workflow; P1, P2, P8, P10, P11, P15, P16, P17



(b) Tier 2B Workflow; P4, P6, P20

Figure 3: Tier 2 Participant Workflows. The dashed boxes represent optional steps in the workflow process.

(a) 3A workflow; P9, P12, P13

(b) 3B workflow; P7, P14, P18, P19

Figure 4: Tier 3 Participant Workflows. The dashed boxes represent optional steps in the workflow process.

[...] The most promising [tool] is called sigma and they let us write behavioral indicators and also write signatures on log files." One of the benefits of behavioral signatures is that they "encompass more than one [sample] that I load" as P2 discusses.

If analysts following this workflow are not able to get the malware to reveal its malicious behavior with dynamic analysis, they will often resort to static analysis. P1 explains his process: "first you run it in a dynamic environment and you see: Does it output anything? Does it look like it's running as you think it should? [...] To understand it better, run it on bare metal to see if there is anything in that dynamic environment that it's detecting. [...] If you still can't get anything there, the next step is reverse engineering. So, doing manual analysis with IDA pro and unpacking and disassembling to try and understand what's going on. That is the tiered approach."

The third workflow, shown in Figure 3b, also utilizes a combination of static and dynamic analysis, but emphasizes static analysis. Static analysis is used by these participants to conduct a more in-depth analysis of the tools used by the attackers. P20 starts with "an API scout over it; meaning I extract all of the statically referenced window API functions and basically do a similarity of them because the spectrum of the API functions that I use is quite static for families because it basically relates to the semantics that are encoded by the programmer." This type of analysis allows participants to label samples with their malware family and understand the tools, which improves the detection of different variations of the sample.

In this workflow, static analysis is also used to identify capabilities of the sample and write different reports about it. P4 mentions that "some people want to know all the capabilities like what can [the malware] do; Can it log keystrokes? Can it respond to backdoors? Can it take screenshots?" These insights are compiled to write reports about the malware, both for internal use by other teams, such as threat intelligence, and to respond to external requests from customers. As previously mentioned, participants in this workflow occasionally utilize dynamic analysis. P6 provided an example of when he would use dynamic analysis, "I had some samples ... they were packed in a certain way that I couldn't see. In that case, I put it in a VM and ran it and saw that it's definitely that family. That's an example of when I would do dynamic."

The fourth workflow, shown in Figure 4a, is used to analyze multiple steps of a malware attack. Attackers can combine multiple stages into a complete attack. This workflow identifies the connections between these stages. Participants who use this workflow want to identify the TTPs used throughout the attack, which requires dynamic analysis with behavioral monitoring. For example, P13 said, "we've got a couple different dynamic analysis tools. I'll run through those if it's an executable [...] makes it easier to triage and get what I want out of the document." However, if the malware evades the analysis or does not show it's complete malicious behavior, static analysis is used to figure out what triggers are required for the malware to download and execute the second or third payload. As P12 explains, "a lot of times the second stage requires the downloader to actually load it and do something to get it to run. So, if you're running a sandbox that can collect and can detect the second stage but it runs it separately, that may not work. You may just get a file that doesn't execute. So, you need it to do one stream where it downloads each stage and is able to put that in the same VM and memory so that the other components run." The identified attack techniques are often sent to threat intelligence teams who model threat actors. P12 says these teams "pull a lot of the information and figure out the different aspects of what they do when they model it. They use a graph database to model and look for trends. [...] As they model, they start tying the different techniques and building the bigger picture and they might see that different activities are related or are from the same campaign and threat actor."

The final workflow in Figure 4b provides another method for extracting TTPs over a longer period of time by emulating the malware. As P7 explains, emulating the malware involves "reversing the C2 protocols and then using that reversed C2 protocol to harvest information from live C2s, whether that is payloads or targets, whatever I want to get from the C2s." This analysis can reveal how a group's malware moves through different infrastructure while it is developed, obfuscated, and eventually deployed during attacks. This valuable knowledge allows security specialists to understand the threats they are likely to face in the near future and prepare appropriate defenses to stop such threats before they can damage their organization. By adopting this approach, organizations can reduce the number of cyber attacks that they fall victim to and improve their security posture.

5.4 Factors that Lead to Differences in Workflows

The detailed findings from this section reinforce our conclusion in section 4 that the primary factors that affect the workflows are the main objectives and the extracted IOCs. Additionally, we found two additional influencing factors. First, we found that the type of malware sample being analyzed affects the workflow. When samples are classified as variants, the participants' workflow is less extensive than the analysis of novel malware because the analysts already have an understanding of the malware family. While workflow 3a and 3b begin to explore the steps required to conduct a more in depth analysis, we leave a more detailed investigation for future work. The second additional influencing factor is the obfuscation mechanisms used by the sample. When analyzing malware with more sophisticated obfuscation, 4 of our participants prefer to use dynamic analysis to deobfuscate the sample. In contrast, when analyzing malware samples that utilize known obfuscation techniques, 2 participants prefer to skip dynamic analysis and analyze the sample statically.

6 DYNAMIC ANALYSIS SYSTEM SETUP

In section 5, we learned that all of the common malware analysis workflows involve dynamic analysis. Due to this widespread use, we investigated how different participants configure their dynamic analysis system. During the interviews, we asked participants to walk us through their configuration process. From the responses, we learned that one of the most important factors in performing quality dynamic analysis is the careful setup and configuration of the analysis system. Specifically, we identified five main steps that analysts complete when they set up their dynamic analysis system and their reasoning behind how they complete each step.

6.1 Virtualization Vs. Bare Metal

When performing dynamic malware analysis, an analyst must decide whether to run samples in a virtual environment or directly on a bare metal machine. All of the participants use virtual environments for dynamic analysis. One of the main reasons for this is that bare metal analysis requires more physical resources, As P1 mentioned, "It's costly to run things on bare metal" (previous work has also discussed this limitation [54]). Also, 5 participants emphasized the fact that bare metal analysis does not scale well to a large number of malware samples. This is especially true for the participant in tier 1 because he frequently needs to analyze many samples. However, if malware does not show its expected behavior in a virtual environment, 4 participants re-analyze the sample on a bare metal system. As P1 mentioned, the only time he uses bare metal is for "unique cases; you can't run everything."

6.2 Open-Source Vs. Commercial Dynamic Analysis Tools Vs. Custom Virtual Machines

The three types of virtual dynamic analysis platforms that analysts use are commercial sandboxes, open-source sandboxes, and personal analysis environments that are generally built with virtual machines (VMs). When participants decide to analyze malware in a virtual environment, they have to decide which platform to use. From our interviews, we discovered that the more in-depth the analysis becomes, the less likely it will be done with commercial tools, primarily because of their lack of configurability.

Specifically, 3 of our more experienced participants noted that they want the ability to set many different parameters of the environment and add additional features. For example, P19 said that if he "cannot put personal signatures into the sandbox, that can be a problem." Some of the other features that our participants want in their sandboxes include the flexibility of running samples with either a 32 or 64 bit architecture (P19), adding custom sensors to collect activity from different malware types such as JavaScript or Visual Basic (P19), and adding custom code when they develop their own features (P16). Unlike commercial sandboxes, open-source solutions allow analysts to personalize their setup and are free to use. Although participants appreciate the flexibility of open-source sandboxes, one of their main critiques is that they require a significant amount of effort to set up correctly. Also, 3 participants mentioned that the setup process requires a deep technical understanding of how sandboxes function.

In contrast to open-source sandboxes, 4 participants say that commercial sandboxes are easier to use. Although we emphasize the importance and challenges of setting up dynamic analysis systems, some commercial sandboxes provide a preconfigured environment. As P12 stated, security companies "have done a lot of research to figure out what products should go in that and how to configure it to get it to run optimally." The malware analysts at these companies also provide constant feedback to the sandbox engineering team to address new evasion techniques and incorporate new detection signatures. However, the decision about what new features are added to the sandbox is made by its developers, not the malware analysts who use it. Therefore, it can be challenging for users to get additional features added to the sandbox because this may not be the main driving force of the developers.

Due to the previously mentioned disadvantages of both commercial and open-source sandboxes, 9 participants set up personal analysis environments. Generally, these environments are composed of VMs with analysis tools installed. As P12 mentioned, "I like to do my own dynamic analysis. I will run it in my VM and run different tools that collect information." We were surprised when P21 said that he had "never done anything with cloud sandboxing like Cuckoo. I'm not terribly informed on what they do. I've just done things with VM because they are easy to set up."

6.3 Setting Up Environment Targeted by Malware

One of the key decisions an analyst has to make is how to set up the execution environment, including selecting the operating system, time zone, programs installed, and files present, such that malware samples will try to attack it. Table 3 and Table 4 show the features that participants change when configuring a general or specific environment, respectively. To provide the most accurate results, we only include participants who responded to our follow-up survey to prevent inaccuracies due to a participant not mentioning a feature they do configure during their interview.

ID	OS	Microsoft Office	Web Browser	Adobe Acrobat	Libraries	Timezone	Language	Usernames	File Names	Browsing History	Populate Files	Other
P9	Windows & Linux	✓	✓	✓	✓	✓	✓	✓	✓	✓		
P10	Windows	✓	✓	✓						✓	✓	Shared drives
P16	Windows 7, 8.1, 10 & Android		✓	✓	✓		✓	✓	✓		✓	
P18	Windows 7, 10, or XP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Admin vs. normal user
P20	Windows	✓		✓	✓			✓	✓	✓	✓	.net, MSVC runtimes

Table 3: General environment features: For the participants who strive to create a general dynamic analysis environment, a check (✓) signifies that the participant sets the corresponding configuration in their environment.

ID	Timezone	Language	Filenames	Browser History	Populate Files
P2	✓				
P4			✓		✓
P5	✓	✓			
P13			✓		
P19				✓	✓

Table 4: Specific environment features: For the participants who configure specialized environments for running individual malware samples, a check (✓) signifies that the participant configures the corresponding environment feature for individual samples.

Some malware rely on features in the environment to complete malicious actions. An example of this is phishing emails with malicious documents attached. The environment must have Microsoft Office installed to run the commands that are embedded in the file. Therefore, to see some of the behaviors of the sample, the dynamic analysis environment must be configured in a certain way. One method, described by 10 participants, for analyzing such samples is to configure a general environment that can analyze many samples without modifying the environment. For example, P13 said that "scattering a couple files and some basic software [...] is slightly more than bare minimal," which was also confirmed in the follow-up survey. This is especially true when samples are being executed for the first time because the analysts do not know what environment the sample requires until they do a more in-depth analysis. After the first execution in a general environment, the analysts will then decide whether they want to continue testing out different dynamic analysis configurations or utilize static analysis.

When the participants decide to continue using dynamic analysis, 5 participants configure their environments either to replicate the target of specific samples or to avoid evasion techniques. Specifically, some samples will evade analysis by checking the values of environment variables such as the language or time zone and not performing malicious tasks if it is not running on its targeted system. Such evasive malware pose additional challenges that analysts must decide how to handle. These participants choose to handle evasive malware by configuring the analysis environment specifically for a sample. For example, P1 stated that he runs malware samples in a "similar operating system to what [his] customers were running" in order to emulate, as closely as possible, the malware's target environment. This creates an analysis environment that the malware is more likely to attack, allowing the analyst to observe more of the sample's malicious behavior. Table 4 shows the features that these participants set for individual samples. Each feature can be tied to a different evasion technique seen in malware samples. For example, malware samples may target a specific geographic region. P2 executes samples in "different regions around the world and he runs that same url or that same payload through each region looking for different characteristics" because he does not know "what the attacker is looking for or who they are targeting." Similarly, malware that exfiltrates data may only take specific files, so if these files are not present during the analysis, the real behavior of the sample will not be captured.

A surprising result is that although tier 3 participants look at specific malware samples, they do not spend as much time or effort customizing their environment. As P19 described, "I probably wouldn't bother. It would be interesting, but at that point [...] I would just statically analyze and reverse engineer it." However, in the follow-up survey, P19 clarified that he occasionally sets the browser history and populates files.

Another approach that 2 participants discussed for setting up dynamic analysis systems is to use an automatic script that randomizes the features of a general, realistic environment that were previously mentioned. This approach not only reduces the setup time, but can also help prevent malware from fingerprinting the analysis system. Due to the randomness of the environment, malware cannot identify specific characteristics of the analysis environment to search later to identify that they are being analyzed.

6.4 Configuring Network Communication

The next consideration of a dynamic analysis environment is how to configure the network connection. 17 participants believe that capturing the network behavior of malware when it is allowed to connect to the real Internet provides insights that are beneficial in conducting their analysis. These participants are aware of the potential risks of malware samples causing harm to other Internet users and take the necessary precautions to avoid causing serious

harm. The precautions that participants take are similar to what previous work has taken, such as blocking outbound communication from specific protocols commonly used for denial-of-service attacks [20, 24, 32, 41, 64, 65].

In contrast, 2 participants from incident response and threat intelligence teams are concerned that malware may trace them back to their company. One possible solution for this problem is to analyze the malware through VPNs that appear to originate in a location other than where the analysts currently are. However, these participants argue that this technique does not protect against the attackers tracing the analysts back to their company. For example, P12 states that "If it's part of an incident response engagement, we don't do that because we don't want to tip off the threat actor that we are investigating." An alternative option is network emulation, which allows incident responders to send synthetic network responses to the malware and analyze some network behavior without anything escaping their analysis environment and alerting the attackers that their malware is being analyzed. This technique can also be helpful when a C2 server is down and analysts want to observe how the malware would communicate with it.

6.5 Dynamic Analysis Execution Time

When performing dynamic analysis, participants must decide how long to run each malware sample. Through our interviews, we discovered that 15 participants run samples for a short time period (less than 5 minutes) because executing a sample requires resources, which are limited. To run all the samples that tier 1 and tier 2 participants collect, the time allocated to each must be small. Also, participants like P18 argue that "a low execution time limits the amount of damage that can [be] done."

However, there are exceptions to this short execution time. 4 participants in tiers 2 and 3 tend to run their samples for multiple hours. One reason that participants prefer longer execution time is that changing the flow of time gives malware an artifact that they can use to evade the analysis system. Therefore, to observe the true behavior of the malware, these participants state that they do not alter the time. Finally, in tier 3, 3 participants perform longitudinal studies of malware, lasting for months. A long-term analysis allows these participants to understand the motivation and tactics behind the attack. Finally, 2 participants try to create tools that allow them to monitor an attacker's C2 infrastructure and gain first hand access to malware being distributed by the attacker. This allows them to be proactive by enhancing defensive security measures to protect against new techniques and avoid being compromised when these techniques are eventually deployed. P18 stated that their system "can tell within minutes if any spam campaign is sent or if they updated the configuration."

6.6 Overcoming Evasion

The last component of the dynamic analysis workflow we investigated was how analysts handle evasive malware samples. 16 participants in tiers 2 and 3 utilize techniques such as debugging and patching malware to get evasive malware to reveal its malicious behavior during dynamic analysis. For example, P17 stated that "the debugger can be used to bypass the evasion checks that are anti-virtualization." Once the participants identify the evasion techniques with signatures such as Yara, they can use a debugger to manually skip over them and execute the malicious behavior. This process has been partially automated by tools such as CAPE Sandbox [1]. Similarly, 4 participants choose to patch the malware, completely removing evasive checks.

A different approach for handling evasive malware mentioned by a few participants is to utilize existing open-source solutions such as Al-Khaser [3]. Al-Khaser is a tool that runs in a sandbox and provides a list of evasion techniques that the sandbox is susceptible to. Our interviews also confirm that practitioners and researchers manipulate the system time to bypass time-based evasive checks as extensively studied in past work [38, 39, 63, 81].

7 DISCUSSION

After examining how the participants perform malware analysis in practice, we identified challenging steps in their workflows. We believe that these challenges faced by practitioners can be ameliorated by the malware analysis research community. First, we identify previous research that has not yet fully transitioned into practice. Second, we highlight remaining research questions and propose future directions that leverage the expert domain knowledge gained in this study to help answer these questions. Third, we present recommendations for improving the usability of malware analysis systems.

7.1 Transitioning Research to Practice

In this section, we identify research that has the potential to ease some of the challenges our participants face in practice. As previously discussed, one of the most common challenges with dynamic malware analysis is determining how to get a malware sample to reveal its malicious behavior. P13 said that one of the biggest challenges is "probably if it doesn't run, trying to figure out why it doesn't run. It seems that dynamic analysis for the things that run tend to do a pretty good job, but if it doesn't run or produce any output; that rabbit hole you have to go on." While analysts have ways to handle a variety of evasion techniques, there is no single solution to get a sample to execute its malicious instructions. One technique that participants use to avoid being evaded is by creating an environment that is appealing to the malware. Another strategy to avoid evasion that has been explored in research is to analyze the malware in more transparent environments. A considerable amount of research in this area could be transitioned into practice [39, 52, 71, 83, 84]. These methods try to defeat fingerprinting by removing artifacts in the analysis environment that malware can associate with the analysis system. If these methods could be transitioned into practice, practitioners could analyze their malware in these environments and, due to the increased transparency, malware samples would be less likely to evade analysis. This method could allow analysts to capture more of the sample's malicious behavior. Although these are promising analysis techniques, their use in practice may not be straightforward. As P21 said, "I read a bunch of academic papers, but what I often found was that a lot of them were in a lab scenario where you control the inputs and outputs. Extrapolating that into something that I can use is difficult." The transparency of these methods prevents malware from detecting

that it is being run in a controlled environment however, they do not analyze the malware in its targeted environment. This study has shown that configuring the analysis environment to mimic its targeted environment is critical to get some malware samples to reveal their malicious behavior. In the next section, we propose future research that has the potential to address these remaining challenges.

Determining whether a sample is a variant of a previously analyzed sample is a common task that most malware analysts have to perform to avoid conducting an in-depth analysis of a known sample. From the interviews, 14 participants stated that this remains a challenging process. For example, when asked what strategies P2 uses to identify malware variants, he replied, "I'm going to say there are no good tools or resources, at least in the free space, that help in that arena." Although this remains a challenge in practice, it is important to note that there has been a considerable amount of research to address this problem, including methods that leverage dynamic features [24, 27, 68], static features [42], or a combination of the two [44]. These features are used to classify or cluster malware samples into their respective families automatically. By combining these techniques with the large amount of malware samples that practitioners have access to, analysts could build clusters that divide samples into different families. These clusters could then be used later to determine whether an unknown sample fits into a known malware family. By adopting these techniques, practitioners could reduce the amount of time spent determining whether samples are variants, and instead focus on analyzing the novel samples.

Additionally, 8 of the participants stated that the task of determining whether two malware functions are similar is one of the more time-consuming tasks they need to perform. As P21 stated, "Figuring out if two binaries have the same function is a really, really hard problem that I worked on to get something close to right. Being able to say that these are reliable results and putting a stamp [saying that those] two match is difficult." There has been past research that addresses the problem of function similarity [59, 62, 77]. Some of these solutions propose a hybrid method with dynamic and static analysis, while others use machine learning based approaches to identify similar functions. The algorithms proposed in these papers could be implemented as a new analysis feature as a first step towards addressing malware function similarity.

Another challenging task identified in practice is distinguishing between malicious and benign behaviors found in dynamic analysis execution outputs. As P6 said, "To use a sandbox you need to have a good idea of what is baseline behavior. [...] You can't just put it in a sandbox and take all the results. You need to be able to look at it and say that's regular windows and adobe behavior. That's something that can trick people a lot." One approach to address this problem is to compare the system calls generated by a malware sample to the system calls generated by benign applications when executed in a dynamic analysis system. Christodorescu [35] provides such an algorithm that practitioners could use to compute the differences between these system traces, and automatically separate malicious behavior from normal system behavior.

The last time-consuming process that we identified could benefit from past research is unpacking malware samples. Unpacking malware samples is a task that 7 participants not only describe as time-consuming, but 3 participants also state that the effort of unpacking does not provide them any useful information about the malware. In fact, they would rather spend their time on more valuable tasks such as analyzing the functions of a malware sample. There are many research systems that address various types of unpacking. Such systems include PolyUnpack [69], Rambo [74], and BinUnpack [34]. These systems focus on automating the process of unpacking and have the potential to help analysts in practice.

7.2 Future Research Directions

In addition to the challenges discussed in subsection 7.1, we identified additional challenges that participants faced, which potentially pose new research questions. In this section, informed by the expertise of our participants, we define some future research directions that may help fill existing gaps in these areas.

How to configure a dynamic analysis environment that replicates a malware's targeted environment. As seen in section 6, configuring the dynamic analysis environment to match the malware sample's targeted environment can be crucial to observe a sample's full, malicious behavior. Although previous work done by Brumley et. al. [30] identifies the triggering events that cause a malware sample to execute its malicious behavior, it is still unclear how to automatically configure the runtime environment of a dynamic analysis system such that it is an appealing target for the sample. In practice, we found that many participants use static analysis to determine what features are required in the dynamic analysis environment to trigger the malicious behavior. One potentially impactful research direction is building an integrated hybrid analysis system to empower analysts. It could leverage previous research such as GoldenEye [79], which proposed a method to identify the targeted victim's environment. This system first uses taint analysis combined with manual analysis to identify conditional statements that are sensitive to the environment. Next, it executes the malware in multiple environments, one for each possible return value of the APIs that query the environment. Although this method is more efficient compared to its predecessor, X-Force [67], it requires significant memory to run a sample on multiple environments in parallel.

The exploration of concurrent use of both static and dynamic analysis in an integrated environment could be promising. First, static analysis would automatically search the malware for conditional checks of the targeted system. Then, symbolic analysis could be applied to solve for the environmental conditions of the malware's targeted system. After finding the necessary environmental conditions, this system would suggest and set the corresponding values in the dynamic analysis environment and re-run the malware sample. In contrast to GoldenEye [79], this system would only execute the sample in a single environment, which would reduce the amount of memory and computational resources required for the analysis. Additionally, by incorporating feedback from malware analysts, better predictions of the targeted environment could be made over time, leading to faster analysis times. If a malware sample was packed, this tool would have to apply currently available unpackers [34, 69, 74] to obtain a sample that can be statically analyzed.

How to automatically analyze multistage malware. Some participants who use workflow 3a, seen in Figure 4a, need to analyze multi-stage malware attacks. However, participants explain that getting a sample to execute all of its stages at once can be quite challenging. Running each stage independently is not effective because there can be dependencies between the different stages that must be taken into consideration. Our study participants analyze these campaigns by using dynamic analysis to run each stage and static analysis to step through the process of downloading the next stage and determining how it injects the payloads to ensure that each stage runs as intended. A promising technique to address this challenge is concolic execution, which uses a combination of both symbolic and concrete analysis. Symbolic analysis could automate the process of determining what inputs and settings each stage requires to run as intended. Then, the concrete part of the analysis could use the newly found inputs to properly execute the malware and retrieve the next stage. The software testing community has previously proposed a technique that leverages a combination of dynamic and symbolic execution to execute targeted paths in a program [23, 31]. Although this research proposes methods that are similar to what we suggest, applying them to malware analysis presents new challenges because malware samples frequently evade analysis. Combining previous work with the strategies that the participants use to analyze multistage malware offers a potential way to automate additional analysis steps of these campaigns.

7.3 Usability Recommendations

From our participants' explanations of the disadvantages of current tools, we identify two important aspects that developers of malware analysis tools should keep in mind. Lack of customization of tools is the main disadvantage that participants mentioned in section 6. According to many analysts, it is important for tools such as dynamic analysis sandboxes to allow users to customize the settings of the analysis system, such as installed programs as well as file paths and their content. As mentioned in subsection 6.2, some analysts find that commercial dynamic analysis tools lack critical features such as the type of architecture malware can be executed on (Ex: 32bit vs. 64bit) and support for malware samples that target different platforms such as JavaScript or Visual Basic. During the interviews, participants also mentioned that they use a wide range of tools to collect all of the information they need. As P2 says "I use VMs, VPN to hide my network, Wireshark, Process Explorer to monitor processes, ProcMon to monitor what the process is doing [...] there is also ProcDOT and all sort of other tools to analyze those dumps after you get them so there are countless tools." Malware analysts would benefit from a dynamic analysis tool that has a more comprehensive and flexible monitoring program, which collects all of the IOCs and presents them to the analyst in a standard format rather than asking analysts to run different tools to obtain each type of indicator separately.

8 RELATED WORK

User Studies. A sizable body of prior work has examined the workflows of software engineers (e.g., [40, 49]), including engineers writing security software [21, 22]. However, only a few prior works have examined how security-specific engineers conduct their work. Most related to our work, Votipka et al. [76] studied the workflow of reverse engineers through an observational study of security professionals. Their work develops a formalized workflow that describes how a reverse engineer completes the task of understanding the operation of a piece of software that the professional is unfamiliar with. Our study confirms the finding of Votipka et al. that static analysis is often used when reverse engineering a malware sample. In contrast to Votipka et al. [76], the participants in our study are all malware analysts, rather than reverse engineers. Additionally, we find that the workflows extracted from our participants include analysis tasks that are not present in the workflow of reverse engineers described in Votipka's work. Finally, our results contribute a discussion of the dynamic analysis configuration process that malware analysts use, a topic that is also not addressed in prior work.

Another related study by Votipka et al. [75] investigates the differences between the vulnerability discovery processes of software testers and white-hat hackers. Its methodology consists of semi-structured interviews with participants from both the hacker and tester communities, and they use their results to recommend improvements to vulnerability discovery processes of both communities. Our methodology is similar to the one used for this study, but we focus on how malware analysis is performed in practice as opposed to vulnerability discovery.

Malware Analysis. A major challenge of dynamic malware analysis is the proper configuration of the analysis environments such that malware samples will reveal as much of their malicious behavior as possible. Kharaz et. al. [50] automatically creates a realistic environment as part of their ransomware detection tool. To make an attractive environment for the ransomware, the authors design scripts that create files with believable content pulled from the internet, names with real words, and file metadata with reasonable values.

Another approach to overcome evasion is to design analysis systems that are transparent to the malware [39, 52, 71, 83, 84]. These methods try to defeat fingerprinting by removing artifacts in the analysis environment that malware can associate with the analysis system. Other methods focus on detecting when malware stalls before performing malicious activities, and force the malware to execute a path that avoids the code that stalls execution [56].

Some analysis systems execute a single sample in multiple environments in an attempt to get a greater percentage of the malware's true behavior to execute [26, 54, 63, 67]. A common environment included in such systems is bare metal. When malware runs outside of a virtual machine, fingerprinting methods are less effective because the artifacts collected are from a physical machine instead of a virtual environment [53, 54, 84]. Furthermore, multiple environment techniques have been used to automatically detect any type of evasion techniques [26, 52, 58]. In order to help malware analysts determine if their analysis system is vulnerable to evasion, Jing compares artifacts from an analysis environment with a standard environment to determine the artifacts found in the analysis environment but not in production [48].

As the interviews with the participants show, malware analysts sometimes turn to static analysis techniques when they want to reverse engineer a sample or learn more about how to configure

their dynamic environment. Researchers have also applied static analysis to determine the malware targeted environment. Xu [79] analyzes targeted malware using parallel execution environments and automatically adjusts the environment settings that the sample is looking for. First, static analysis is used to identify environment query APIs and determine the possible return values. Then, the sample is dynamically analyzed in parallel environments, each with different environment settings for the possible return values. While executing, the system sets environment settings such that the sample executes the most interesting paths. To identify inputs that will trigger malicious behavior, Brumley [30] proposes a hybrid approach that matches malware triggers to conditional checks in the binary and treats the variables involved in such checks as symbolic. Then, the malware is executed such that the instructions that operate on symbolic variables are symbolically executed and other instructions are executed concretely. This approach allows analysts to identify inputs that trigger the malware and then dynamically execute the sample with the identified inputs to observe its malicious behavior.

There are other works that leverage hybrid analysis in ways our participants did not discuss. Hybrid analysis has been used to find dormant malicious behaviors [36], discover vulnerabilities [23], and find software bugs [31]. A different application of hybrid analysis is hybrid concolic testing [60]. This work uses random testing to access program paths that reach deep within the program, then symbolic analysis is applied to explore parts of the program near that execution path.

Finally, static analysis can be leveraged independently for malware analysis. Symbolic analysis is one such technique that involves exploring many paths of a program symbolically and using constraint solvers to determine whether different paths are feasible and what inputs will cause that path to be executed [25, 70]. Since malware is frequently obfuscated, it can be challenging to apply these static analysis techniques in malware analysis. Such obstacles have been previously explored, including symbolic analysis of obfuscated code [80] and extraction of the semantics of obfuscated code [29]. Also, when analyzing complicated symbolic constraints in malware, one method solves a subset of the constraints through decomposing them and recombining the produced inputs [31].

9 CONCLUSION

In this user study, we interviewed 21 malware analysts to get a better understanding of how malware analysis is done in practice. These interviews shed light on the diversity of malware analysis, which is reflected in the broad range of workflows that our participants follow as well as the amount of effort they put into configuring their dynamic analysis systems. Our analysis highlights six different components of dynamic analysis systems that practitioners focus on when setting up dynamic analysis systems and how their setup process differs depending on their tier. Based on the user study, we identified challenging tasks that the participants face during their analysis process. We first discuss promising existing academic research that could be transitioned to practice and then propose future research directions that could help address the remaining challenges.

REFERENCES

[1] Capev2. https://github.com/kevoreilly/CAPEv2.
[2] Virustotal. https://virustotal.com.
[3] al-khaser. URL https://github.com/LordNoteworthy/al-khaser.
[4] Any-run - interactive online malware sandbox. https://any.run.
[5] How antivirus softwares are evolving with behaviour-based malware detection algorithms. https://analyticsindiamag.com/how-antivirus-softwares-are-evolving-with-behaviour-based-malware-detection-algorithms.
[6] Equifax says cyberattack may have affected 143 million in the u.s. https://nytimes.com/2017/09/07/business/equifax-cyberattack.html.
[7] Free automated malware analysis service. https://hybrid-analysis.com.
[8] Malpedia, . https://malpedia.caad.fkie.fraunhofer.de/.
[9] Malshare, . https://malshare.com.
[10] Malware bazaar, . https://bazaar.abuse.ch.
[11] Mitre att&ck. https://attack.mitre.org/matrices/enterprise/.
[12] ollydbg. http://www.ollydbg.de/.
[13] Openioc: Back to the basics. https://www.fireeye.com/blog/threat-research/2013/10/openioc-basics.html.
[14] Pyramid of pain. https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html.
[15] Reversing labs. https://reversinglabs.com.
[16] Target missed warnings in epic hack of credit card data. https://bloom.bg/2KjElxM.
[17] thezoo - a live malware repository. https://github.com/ytisf/theZoo.
[18] Twitter. https://twitter.com.
[19] Unpacme. https://unpac.me.
[20] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 17th ACM SIGCOMM*, pages 41–52, Stanford, CA, Aug. 2006.
[21] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky. You get where you're looking for: The impact of information sources on code security. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 289–305. IEEE, 2016.
[22] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky. How internet resources might be helping you develop faster but less securely. *IEEE Security & Privacy*, 15(2):50–60, 2017.
[23] D. Babić, L. Martignoni, S. McCamant, and D. Song. Statically-directed dynamic automated test generation. In *Proceedings of the 2011 International Symposium on Software Testing and Analysis*, pages 12–22, 2011.
[24] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of internet malware. In *Proceedings of the 9th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, pages 178–197, 2007.
[25] R. Baldoni, E. Coppa, D. C. D'elia, C. Demetrescu, and I. Finocchi. A survey of symbolic execution techniques. *ACM Computing Surveys (CSUR)*, 51(3):1–39, 2018.
[26] D. Balzarotti, M. Cova, C. Karlberger, E. Kirda, C. Kruegel, and G. Vigna. Efficient detection of split personalities in malware. In *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb.–Mar. 2010.
[27] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2009.
[28] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2011.
[29] T. Blazytko, M. Contag, C. Aschermann, and T. Holz. Syntia: Synthesizing the semantics of obfuscated code. In *Proceedings of the 25th USENIX Security Symposium (Security)*, pages 643–659, Vancouver, BC, Canada, Aug. 2017.
[30] D. Brumley, C. Hartwig, Z. Liang, J. Newsome, D. Song, and H. Yin. Automatically identifying trigger-based behavior in malware. In *Botnet Detection*, pages 65–88. Springer, 2008.
[31] J. Caballero, P. Poosankam, S. McCamant, D. Babi ć, and D. Song. Input generation via decomposition and re-stitching: Finding bugs in malware. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 413–425, 2010.
[32] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: the commoditization of malware distribution. In *Proceedings of the 20th USENIX Security Symposium (Security)*, San Francisco, CA, Aug. 2011.
[33] R. Canzanese, S. Mancoridis, and M. Kam. Run-time classification of malicious processes using system call analysis. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 21–28. IEEE, 2015.
[34] B. Cheng, J. Ming, J. Fu, G. Peng, T. Chen, X. Zhang, and J.-Y. Marion. Towards paving the way for large-scale windows malware analysis: Generic binary unpacking with orders-of-magnitude performance boost. In *Proceedings of the 24rd ACM Conference on Computer and Communications Security (CCS)*, pages 395–411, Toronto, Canada, Oct. 2018.

[35] M. Christodorescu, S. Jha, and C. Kruegel. Mining specifications of malicious behavior. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 5–14, 2007.

[36] P. M. Comparetti, G. Salvaneschi, E. Kirda, C. Kolbitsch, C. Kruegel, and S. Zanero. Identifying dormant functionality in malware programs. In *Proceedings of the 31th IEEE Symposium on Security and Privacy (Oakland)*, pages 61–76, Oakland, CA, May 2010.

[37] W. Cui, M. Peinado, Z. Xu, and E. Chan. Tracking rootkit footprints with a practical memory analysis system. In *Proceedings of the 21st USENIX Security Symposium (Security)*, pages 601–615, Bellevue, WA, Aug. 2012.

[38] Z. Deng, X. Zhang, and D. Xu. Spider: Stealthy binary program instrumentation and debugging via hardware virtualization. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 289–298, 2013.

[39] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, pages 51–62, Alexandria, VA, Oct. 2008.

[40] D. Ford and C. Parnin. Exploring causes of frustration for software developers. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 115–116, 2015. doi: 10.1109/CHASE.2015.19.

[41] M. Graziano, C. Leita, and D. Balzarotti. Towards network containment in malware analysis systems. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 339–348, 2012.

[42] M. Hassen, M. M. Carvalho, and P. K. Chan. Malware classification using static analysis based features. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2017.

[43] E. Hollnagel. *Handbook of cognitive task design*. CRC Press, 2003.

[44] M. Ijaz, M. H. Durad, and M. Ismail. Static and dynamic malware analysis using machine learning. In *2019 16th International bhurban conference on applied sciences and technology (IBCAST)*, pages 687–691. IEEE, 2019.

[45] G. Jacob, R. Hund, C. Kruegel, and T. Holz. Jackstraws: Picking command and control connections from bot traffic. In *Proceedings of the 20th USENIX Security Symposium (Security)*, San Francisco, CA, Aug. 2011.

[46] S. A. Jacob and S. P. Furgerson. Writing interview protocols and conducting interviews: tips for students new to the field of qualitative research. *Qualitative Report*, 17:6, 2012.

[47] N. Jagpal, E. Dingle, J.-P. Gravel, P. Mavrommatis, N. Provos, M. A. Rajab, and K. Thomas. Trends and lessons from three years fighting malicious extensions. In *Proceedings of the 24th USENIX Security Symposium (Security)*, pages 579–593, Washington, DC, Aug. 2015.

[48] Y. Jing, Z. Zhao, G.-J. Ahn, and H. Hu. Morpheus: automatically generating heuristics to detect android emulators. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 216–225, 2014.

[49] B. Johnson, R. Pandita, J. Smith, D. Ford, S. Elder, E. Murphy-Hill, S. Heckman, and C. Sadowski. A cross-tool communication study on program analysis tool notifications. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, page 73–84, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342186. doi: 10.1145/2950290.2950304. URL https://doi.org/10.1145/2950290.2950304.

[50] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda. {UNVEIL}: A large-scale, automated approach to detecting ransomware. In *Proceedings of the 24th USENIX Security Symposium (Security)*, pages 757–772, Washington, DC, Aug. 2015.

[51] D. Kim, A. Majlesi-Kupaei, J. Roy, K. Anand, K. ElWazeer, D. Buettner, and R. Barua. Dynodet: Detecting dynamic obfuscation in malware. In *Proceedings of the 14th Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, pages 97–118, 2017.

[52] D. Kirat and G. Vigna. Malgene: Automatic extraction of malware analysis evasion signature. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 769–780, 2014.

[53] D. Kirat, G. Vigna, and C. Kruegel. Barebox: efficient malware analysis on bare-metal. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 403–412, 2011.

[54] D. Kirat, G. Vigna, and C. Kruegel. Barecloud: bare-metal analysis-based evasive malware detection. In *Proceedings of the 23rd USENIX Security Symposium (Security)*, San Diego, CA, Aug. 2014.

[55] C. Kolbitsch, T. Holz, C. Kruegel, and E. Kirda. Inspector gadget: Automated extraction of proprietary gadgets from malware binaries. In *Proceedings of the 31th IEEE Symposium on Security and Privacy (Oakland)*, pages 29–44, Oakland, CA, May 2010.

[56] C. Kolbitsch, E. Kirda, and C. Kruegel. The power of procrastination: detection and mitigation of execution-stalling malicious code. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pages 285–296, Chicago, Illinois, Oct. 2011.

[57] J. R. Landis and G. G. Koch. An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers. *Biometrics*, 33 (2):363–374, 1977. ISSN 0006341X, 15410420. URL http://jstor.org/stable/2529786.

[58] M. Lindorfer, C. Kolbitsch, and P. M. Comparetti. Detecting environment-sensitive malware. In *International Workshop on Recent Advances in Intrusion Detection*, pages 338–357. Springer, 2011.

[59] B. Liu, W. Huo, C. Zhang, W. Li, F. Li, A. Piao, and W. Zou. αdiff: cross-version binary code similarity detection with dnn. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 667–678, 2018.

[60] R. Majumdar and K. Sen. Hybrid concolic testing. In *29th International Conference on Software Engineering (ICSE'07)*, pages 416–426. IEEE, 2007.

[61] L. Martignoni, E. Stinson, M. Fredrikson, S. Jha, and J. C. Mitchell. A layered architecture for detecting malicious behaviors. In *International Workshop on Recent Advances in Intrusion Detection*, pages 78–97. Springer, 2008.

[62] J. Ming, D. Xu, Y. Jiang, and D. Wu. Binsim: Trace-based semantic binary diffing via system call sliced segment equivalence checking. In *Proceedings of the 25th USENIX Security Symposium (Security)*, pages 253–270, Vancouver, BC, Canada, Aug. 2017.

[63] A. Moser, C. Kruegel, and E. Kirda. Exploring multiple execution paths for malware analysis. In *Proceedings of the 28th IEEE Symposium on Security and Privacy (Oakland)*, pages 231–245, Oakland, CA, May 2007.

[64] Y. Nadji, M. Antonakakis, R. Perdisci, and W. Lee. Understanding the prevalence and use of alternative plans in malware with network games. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2011.

[65] M. Neugschwandtner, P. M. Comparetti, and C. Platzer. Detecting malware's failover c&c strategies with squeeze. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2011.

[66] S. Palahan, D. Babić, S. Chaudhuri, and D. Kifer. Extraction of statistically significant malware behaviors. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 69–78, 2013.

[67] F. Peng, Z. Deng, X. Zhang, D. Xu, Z. Lin, and Z. Su. X-force: Force-executing binary programs for security applications. In *Proceedings of the 23rd USENIX Security Symposium (Security)*, pages 829–844, San Diego, CA, Aug. 2014.

[68] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov. Learning and classification of malware behavior. In *Proceedings of the 5th Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, pages 108–125, 2008.

[69] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee. Polyunpack: Automating the hidden-code extraction of unpack-executing malware. In *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC)*, pages 289–300, 2006.

[70] E. J. Schwartz, T. Avgerinos, and D. Brumley. All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask). In *2010 IEEE symposium on Security and privacy*, pages 317–331, Oakland, CA, May 2010.

[71] C. Spensky, H. Hu, and K. Leach. Lo-phi: Low-observable physical host instrumentation for malware analysis. In *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2016.

[72] P. Srivastava and N. Hopwood. A practical iterative framework for qualitative data analysis. *International journal of qualitative methods*, 8(1):76–84, 2009.

[73] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel. Botfinder: Finding bots in network traffic without deep packet inspection. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 349–360, 2012.

[74] X. Ugarte-Pedrero, D. Balzarotti, I. Santos, and P. G. Bringas. Rambo: Run-time packer analysis with multiple branch observation. In *Proceedings of the 13th Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, pages 186–206, 2017.

[75] D. Votipka, R. Stevens, E. Redmiles, J. Hu, and M. Mazurek. Hackers vs. testers: A comparison of software vulnerability discovery processes. In *Proceedings of the 39th IEEE Symposium on Security and Privacy (Oakland)*, pages 374–391, San Jose, CA, May 2018.

[76] D. Votipka, S. Rabin, K. Micinski, J. S. Foster, and M. L. Mazurek. An observational investigation of reverse engineers' processes. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1875–1892, 2020.

[77] S. Wang and D. Wu. In-memory fuzzing for binary code similarity analysis. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 319–330. IEEE, 2017.

[78] M. Xu and T. Kim. Platpal: Detecting malicious documents with platform diversity. In *Proceedings of the 25th USENIX Security Symposium (Security)*, pages 271–287, Vancouver, BC, Canada, Aug. 2017.

[79] Z. Xu, J. Zhang, G. Gu, and Z. Lin. Goldeneye: Efficiently and effectively unveiling malware's targeted environment. In *International Workshop on Recent Advances in Intrusion Detection*, pages 22–45, 2014.

[80] B. Yadegari and S. Debray. Symbolic execution of obfuscated code. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 732–744, Denver, Colorado, Oct. 2015.

[81] L.-K. Yan, M. Jayachandra, M. Zhang, and H. Yin. V2e: combining hardware virtualization and softwareemulation for transparent and extensible malware analysis. In *Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments*, pages 227–238, 2012.

[82] T.-F. Yen and M. K. Reiter. Traffic aggregation for malware detection. In *Proceedings of the 5th Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, pages 207–227, 2008.
[83] F. Zhang, K. Leach, K. Sun, and A. Stavrou. Spectre: A dependable introspection framework via system management mode. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 1–12, 2013.
[84] F. Zhang, K. Leach, A. Stavrou, H. Wang, and K. Sun. Using hardware features for increased debugging transparency. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (Oakland)*, pages 55–69, San Jose, CA, May 2015.

ACKNOWLEDGMENTS

We thank Alex Bardas and the reviewers for their helpful feedback, Daniel Votipka for recruitment assistance and our participants for providing valuable insights.

The second author's work is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2039655. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

APPENDIX

A Survey Questionnaire

Background and Experience.

- Which of the following definitions best describes you? (Please select all that apply) Malware engineer: I work on configuring the malware sandbox analyzers and/or develop programs to process the inputs and outputs of these analyzers. Ex: prioritizing, clustering, parsing, preventing evasion techniques, capturing system calls. Malware analyst: I analyze malware samples to understand their functionality, potential impact and origin.
- [If malware engineer] How many years have you worked as a malware engineer?
- [If malware analyst] How many years have you worked as a malware analyst?
- Please select your most highly qualified skills or specialty areas (please check all that apply): Operating Systems, Networks, Compilers, Computer Architecture, Programming, Cryptography, Virtualization, Scalability, Data Analysis, Sandboxing, Reverse Engineering, Assembly code, Signature creation

Job Description.

- What is your current or most recent job title?
- What is the end goal of your threat/malware analysis work given by your employer ? (please check all that apply) Attribution, Forensics, Recovery remediation, Detection, Classification, Signature creation, Indication of Compromise, Research, N/A, Other
- Please tell me the primary activities you perform day-to-day as part of your job.
- How often do you use the following malware analysis techniques? Dynamic Analysis, Static Analysis, Other
- Is your job somehow associated with malware sandboxing? If so, how ? (Please check all that apply) Collecting malware specimens for the sandbox, Prioritizing the malware specimens that get sent to the sandbox, Configuring the malware sandbox, Categorizing, clustering the malware specimens from sandbox output (usually with ML feature selection), Manually analyzing the output of malware sandbox, Providing feedback to update the sandbox

Demographics.

- Please specify the gender with which you mostly identify. Woman, Man, Non-binary, Prefer not to answer
- Please specify your age range. 18-29, 30-39, 40-49, 50-59, 60-69, >70, Prefer not to answer
- Please specify your ethnicity. American Indian or Alaska Native, Asian, Black or African American, Hispanic or Latino, White, Prefer not to answer.
- Please specify your highest degree level of education. High school credit, no diploma, High school graduate, diploma or equivalent, College credit, no diploma, Trade/Technical/Vocational Training, Associate degree, Bachelor's degree, Master's degree, Doctorate degree, Prefer not to answer
- If you did get a degree please specify your major.
- If you are currently employed, please specify the business sector in which you are currently working for. Technology, Government, Healthcare, Retail, Construction, Education, Finance, Arts, Other.

B Interview Questions

Background and Experience.

- Could you tell me a little bit about what you do in your profession?
- How long would you say you've been doing this?
- What got you into this profession? Have you always been doing this?

Malware Sources.

- Great, so these next set of questions will be mainly about the malware specimens that you receive. Can you walk me through the steps you have to take to get the malware samples and how you decide which ones to analyze? Imagine that you have just started your day at work and are trying to figure out where to begin.
- How do you receive malware specimens that need to be analyzed?
- Roughly how many malware specimens do you get per day?
- What types of malware specimens do you usually encounter (spyware, virus, trojan, keylogger, rootkit, botnets, ransomware, worms, malvertising…)?
- Do you analyze all of these malware specimens or only a subset of them? How do you decide which malware to analyze? How do you prioritize which malware to analyze first?
- What data about the malware do you generally have available before starting your analysis? How do you prioritize the various data types you look at? Why do you prioritize them this way? What data do you consider to be the most important or helpful in doing your analysis?
- Do you determine whether a malware is a variation of a previously seen malware? If yes, how do you do it?

- What percentage of the malware would you say are variants of malware you have previously seen and what percentage are previously unseen?

Dynamic Analysis System Configuration.

- Can you walk me through the steps you take to set up your sandbox environment? Imagine that you were teaching me to set up a sandbox.
- How do you set up your sandbox environment?
- Do you use a commercial, open-source or custom sandbox? Is there a reason why you made that decision?
- Is your hardware bare-metal, virtualized or emulated? Why did you choose that hardware?
- What type of operating system(s) do you use? Why did you choose that/those operating system(s)?
- How do you decide on an OS configuration? For example, do you configure features such as location, time zone, users, groups and services?
- What type of applications do you set up in the user space?
- How do you configure the outgoing network of a sandbox?
- How long do you run your malware specimen for? How did you get this set time?
- Are there any crucial environmental features that you have found necessary to get malware to reveal its malicious behavior?
- Do you execute malware samples multiple times? Why? If so, do you execute them in a different environment? What features are different in the different environments that you use?
- Can you tell me about how you do malware reports?
- What parts of the sandbox configuration takes up the most time?
- What parts of the sandbox configuration process do you find more challenging?

Analysis Workflow.

- Are there any specific tools that you utilize to do malware analysis? How did you learn to use these tools? How do you decide which tool and in which order to utilize during your analysis of a specific malware sample/family?
- Can you walk me through the steps you take to manually analyze a malware sample? Imagine that you were teaching me.
- What is the first thing you look for?
- Are there specific indicators that you look for?
- How did you learn/develop this process? +
- Do you have an SOP (Standard Operating Procedure) or documented procedure on how you approach analyzing a malware specimen?
- What goal are you trying to achieve by the end of your analysis?
- How do you know when you have concluded the analysis process?
- What parts of the process do you find more challenging?
- Where do you generally seek for help when you run into these challenges?
- Is there information you wished you had available that isn't in the report generated by the malware sandbox analyzer?
- Do you perform the same analysis steps when looking at an unknown malware versus a variant of a previously seen malware? If the answer is no, please describe the differences between the two.

Evolution.

- Considering your past experiences in malware analysis, has the analysis process changed over time? If so, what parts have changed? What do you think has been the biggest factor that contributed to these changes? (Ex: business model, personal goals, malware type…)
- Is there any other topic that I haven't touched on which you would like to elaborate on?

C Follow-up Survey

- Do you or your team install the following settings in your dynamic analysis environment? Microsoft office, Web browser, Adobe acrobat, Software libraries – Do you or your team add any specific software libraries in the analysis environment?
- Do you or your team configure or modify the following settings in your dynamic analysis environment? Location, Time, Languages, Username, File names, Browsing history, Populate files
- Which of the following settings do you configure per individual sample? Location, Time, Languages, Username, File names, Browsing history, Populate files

D Codebook

The codes used to analyze the participant interviews are presented in Table 5.

High-Level Codes	Subcodes
Runtime: Refers to the amount of time the participants run the malware sample in the dynamic analysis system.	Minutes, Hours, Weeks, Short
Number of runs: Refers to the amount of times the participant runs a malware sample in the dynamic analysis system.	Numbers
Number of malware samples: Refers to the number of malware samples that a participant.	Per day, Per week , Per month
Data source: Refers to the source of malware samples that participants analyze.	Clients, Repository, Virus total, Pastebin, Twitter, Blog posts, Open source, Crawl
Fresh malware samples: Refers to the amount of time since a malware sample was released.	New, Today, Novel, First
Prioritization of malware samples: Refers to the process that participants use to determine the order in which they analyze their malware samples.	Priority, Customer, Harmful, FIFO, Risk, Novel, Complexity, New, Damage
Malware variants: Refers to the process that participants use to determine whether an unknown sample is a variant of a known malware.	Variant, Family, Cluster, Campaign
Simulated network: Refers to whether the participant simulates the network when performing dynamic analysis.	Network, Simulation
Use of open source or commercial sandboxes: Refers to whether participants use open source and/or commercial dynamic analysis tools.	Open Source, Commercial, Joe Sandbox, Any.run, Cape Sandbox, Virtualbox, Fireeye
Preference of open vs commercial sandbox: Refers to the participants preference between open source dynamic analysis tools and commercial dynamic analysis tools.	Open Source, Commercial, Joe Sandbox, Any.run, Cape Sandbox, Virtualbox, Fireeye
Use of bare metal: Refers to the participants use of bare metal for dynamic analysis.	Laptop, VM, Sandbox, Cloud, Server
Operating system: Refers to whether the participants discussed what operating system that they use for dynamic malware analysis.	OS, Windows, Linux, Unix
Applications installed in dynamic analysis systems: Refers to the applications that participants install in the dynamic analysis system.	Web browser, Microsoft, Java, Adobe, Libraries
Environment Settings: Refers to the configuration of the environment within the dynamic analysis system.	Libraries, Timezone, Language, Usernames, User privilege
Mimic real user: Refers to whether participants configured their dynamic analysis environments to appear as if the system was used by a real user.	Browser history, Files, Documents, Directories, Usage, Names
Monitoring of the dynamic analysis execution: Refers to the process used to monitor the execution of the malware sample in a dynamic analysis system.	Procmon, Hooking, Syscalls, Registry, Files created, Network activity, Pcap, Logs
Evasion: Refers to whether participants discussed strategies they use to analyze evasive malware samples.	Evasion, Encryption, Packing, Obfuscation, Detection, Sleep, Debug, Breakpoint, Skip, Patch
Static analysis process and tools: Refers to the participants static analysis process and tools used.	Static, Analysis, Tool, Ghidra, IDA, Reverse Engineer, Unpack, Decrypt, Disassembler, Decode, Decrypt, Script, Injection, Code, Binary, Function, Registry Keys, Logs
Generate signatures: Refers to how the participants generate signatures.	Signature, IDS, Network behavior, System Behavior, Tool, IP, Domain, Hash, Tactics, Techniques, Procedure, Capabilities

Table 5: Codebook