2013 5th International Conference on Cyber Conflict
K. Podins, J. Stinissen, M. Maybaum (Eds.)
2013 © NATO CCD COE Publications, Tallinn

Patterns of a Cooperative Malware Analysis Workflow

Daniel Plohmann
Cyber Defense Research Group
Fraunhofer FKIE
Bonn, Germany
daniel.plohmann@fkie.fraunhofer.de

Sebastian Eschweiler
Cyber Defense Research Group
Fraunhofer FKIE
Bonn, Germany
sebastian.eschweiler@fkie.fraunhofer.de

Elmar Gerhards-Padilla
Cyber Defense Research Group
Fraunhofer FKIE
Bonn, Germany
elmar.gerhards-padilla@fkie.fraunhofer.de

Abstract: In recent years, an ever-increasing number of IT security incidents have been observed, often involving malicious software. In order to cope with the threat posed, it is essential to have a structured analysis workflow for assessment and mitigation.

In this paper, we give a thorough explanation of the malware analysis workflow specified and employed by our team of analysts. It was deducted from observed work patterns and best practices with a strong focus on enabling collaboration, i.e. analyses conducted by multiple analysts in parallel in order to achieve a speed-up. The proposed workflow starts at the point where one or more malware samples have already been extracted. It consists of four phases as a whole, each with its own goals, constraints, and abort conditions.

The first phase aims at gaining an overview of the current situation and specifying goals of the analysis and their respective priorities. The second phase features a preliminary analysis used to sharpen the picture of the threat, using methods of Open Source Intelligence (OSINT) and automated tools in order to obtain a quick assessment enabling first mitigation. In addition, one objective is to facilitate and prepare a more granular dissection of the malware sample, e.g. by unpacking and deobfuscation. The third phase comprises an in-depth analysis relying heavily on reverse engineering of selected parts of the malware. The selection may be influenced by earlier findings or focus on prominent aspects like nesting, functionality, or communication protocols. The final phase builds upon the results of the preceding phases, leading to tailored mitigation concepts for the specimen analysed.

For each of the proposed phases, we give an overview of potential key tools, e.g. helping to gain information or improve collaboration. On a higher level, we highlight challenges to cooperative analysis and our approach to handle them. In this regard, the workflow contains adoptions of principles known from agile software development methodologies. For example, Scrum is used for management of tasks and coordination, aiding the creation of a reproducible and reliable chain of results.

Keywords: *malware analysis, workflow, cooperation*

1. INTRODUCTION

Malware, short for malicious software, is a prevalent tool for digital crime and targeted attacks. Being well-organised, the underground ecosystem around malware constantly unleashes new threats almost entirely aiming at generation of financial gain for miscreants. Additionally, an increasing number of cases including the use of malware for politically motivated espionage and sabotage campaigns [1, 2, 3] have been observed in recent years as well.

The analysis of malware, especially when trying to achieve deeper insights on concrete inner workings, is a time-consuming task. It usually involves notable manual effort which by itself requires significant expertise to be carried out.

In this paper we explain the workflow used by our team of malware analysts, developed on behalf of the German Federal Office for Infomation Security (BSI). The primary goal of this workflow is to speed up in-depth analysis of malware by parallelization of multiple analysts' efforts working in an environment tailored for collaboration. While we are aware of other opportunities for team collaboration, the proposed workflow represents our collection of work patterns and best practices.

Our contributions are the following:

- We identify challenges for the process of cooperative malware analysis
- We propose a workflow designed to overcome these challenges
- We provide an outline of best practices for malware analysis based on our experience

The remainder of paper is structured as follows. Section 2 examines the challenges to cooperative malware analysis. In section 3, an analysis workflow addressing these challenges is described. Section 4 covers related work and section 5 concludes this paper.

2. CHALLENGES TO COOPERATIVE MALWARE ANALYSIS

Being able to effectively conduct malware analysis requires a considerable skill set on its own [4]. This paper focuses on additional challenges posed by close cooperation of multiple analysts working on the same case and how to benefit from joint resources. Useful insights can be taken from the research field of computer-supported cooperative work (CSCW). In this paper, we limit ourselves to two key aspects of collaboration identified by CSCW, awareness and articulation work.

Awareness can be defined as an understanding of the activities of others, providing context for the own activities [5]. Transferred to malware analysis, this can be seen as a need of synchronization of both case specific knowledge and state information among analysts. In consequence, this highlights the importance of a shared space for documentation and proactive signalling of relevant findings. Furthermore, being aware of the individual analysts' proficiencies helps with quick identification of the right contact points in case of specific questions or when delegation of a task is desired.

Articulation work as defined by Strauss [6] is the coordination of lines of work and the interactions necessary for finding work-related agreements [7]. In terms of cooperative malware analysis, this means the breakdown of analysis objectives into manageable tasks that can be processed in parallel. The integration of work results into a consistent product, e.g. documentation in the form of an analysis report or accompanying proof-of-concept code is covered by this as well.

In order to overcome these challenges, we have developed a workflow to handle cooperative malware analysis by adopting components of the Scrum methodology [8] known from agile software development. The three pillars of Scrum are transparency, inspection, and adaption. Internally, transparency forces analysts to create and experience awareness while providing a clear view on the current status of investigation to the outside. Frequent inspection of documentation artefacts propagates knowledge in the team and enables emergence as analysts may encounter artefacts to which they can contribute. Adaption allows controlling progress towards the defined analysis goals. The roles as defined for a Scrum team are treated not as consequently as intended in the concept. The person in charge of keeping contact with a client becomes Product Owner of the analysis. The role of the Scrum Master is taken by different team members on a per case basis. For details on the duties of these roles, please refer to [8].

Scrum as a process management framework has several features that perfectly fit for malware analysis such as being lightweight and flexible. The course of investigation usually has only little foresight, which causes a strong need of short-term decisions based on new findings. Having an iterative progress with incremental results allows keeping focus on superordinate analysis goals and frequently partitioning workload into distinct tasks, thus avoiding plural effort on the same objectives. If a task has been finished by an analyst, it is peer-reviewed by another member of the team. By this, knowledge is shared in the team and a higher quality standard is assured. A typical Scrum task board as used to visualise tasks and their progress is shown in Figure 1, in this example for the analysis of a malware's C&C protocol.

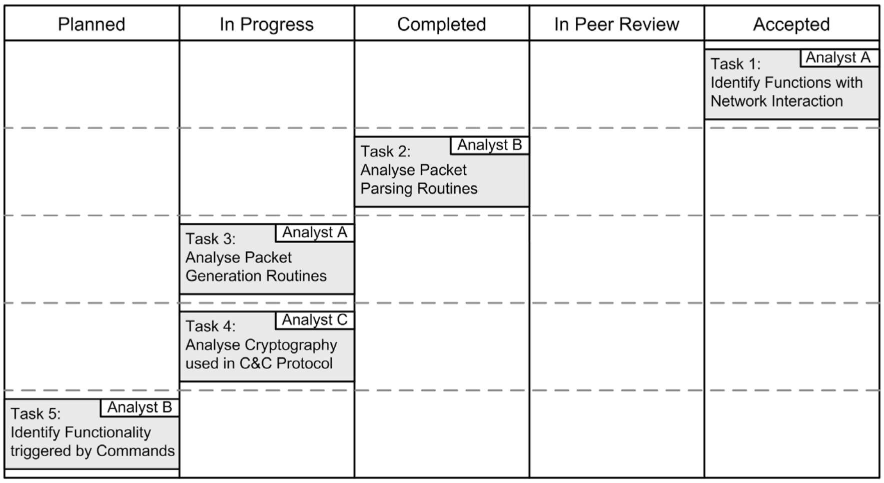
Planned
In Progress
Completed
In Peer Review
Accepted
Task 1: Identify Functions with Network Interaction
Analyst A
Task 2: Analyse Packet Parsing Routines
Analyst B
Task 3: Analyse Packet Generation Routines
Analyst A
Task 4: Analyse Cryptography used in C&C Protocol
Analyst C
Task 5: Identify Functionality triggered by Commands
Analyst B

Figure 1. A typical view on a Scrum board as provided by common task management systems.

Tasks are scheduled to be worked on in so called sprints, a time-box framed by an obligatory sprint planning and sprint review meeting. These meetings are a very effective instrument for overcoming the challenges of awareness and articulation work as defined earlier. While planning meetings are used to define which of the remaining tasks are selected for the upcoming sprint, how complex they are and how they should be approached, reviews are used to inspect accomplished analysis work and to gather feedback on it. The duration of sprints should be chosen in compliance with given analysis time frames. After completion, a sprint retrospective is held to enable inspection of the workflow as a methodology itself, in order to successively gather input for possible adjustments and improvements.

To further enable collaboration, one or more tools covering three classes of functional aspects should be available within the workflow. In the following they will be described as distinct services, while all aspects could as well be served by just one tool. First, a documentation system such as a wiki is needed. It should be accessible both by analysts (read/write permission) and all other parties involved (read permissions). The documentation system is used for all kinds of note taking as well as continuous delivery of the analysis report for a case. Second, a tool supporting task management and being compatible with the Scrum-like workflow allows tracking progress. Third, a case or file repository is used as storage for intermediately generated data. It also serves as version control system for own code created in order to support the analysis.

Social aspects of teamwork that may also influence the analysis performance are out of scope of this paper.

3. MALWARE ANALYSIS WORKFLOW

In this section, our malware analysis workflow is described in detail. It consists of four phases. Each has increasing analysis depth and a scope on delivering details towards defined goals. The phases are not to be seen as completely disjoint but with a partial overlap. The workflow as a whole has been designed with the intention to support a thorough and thus long-term investigation of the characteristics of selected malware families but may also prove useful as component in incident management framework. An overview of the workflow is shown in Figure 2.

For the workflow described in this paper, we assume that the starting point of analysis is an already extracted suspicious file. It is assumed that the initial compromise point has already been identified, thus the process of how this file was obtained, e.g. by means of digital forensics, is out of scope for this paper. Note that the workflow also can be easily adapted to cover forensic activities.

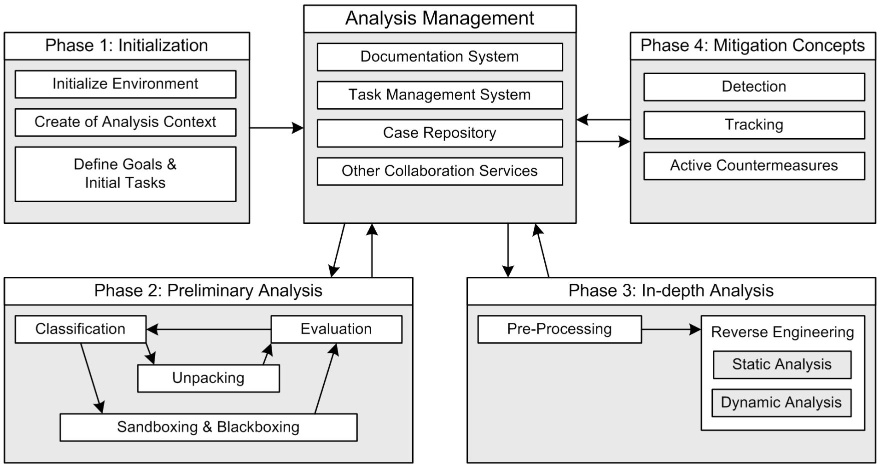

Figure 2. Overview on the proposed cooperative malware analysis workflow.

During all phases and their respective steps, written documentation is produced already alongside the analysis, covering the concrete intention and scope of the analysis activities, a brief description of the procedure taken and its outcome. Additionally, automatable log files such as network traces are always recorded as it is not known whether these are repeatable at a later point in time. Furthermore, the notes taken increase the understanding among analysts involved. The documentation allows assessment of the current analysis progress as well. We use a report template as basis for documentation in order to maintain consistency in our reports. The template may be extended to suffice the needs of special cases.

Where possible, illustrations are used to clarify findings as they can be considered a valuable supplement [9]. The documentation system also serves as a growing knowledge database that can be queried against.

When handling incidents, it is also of interest to collect evidence on whether the malware subject to analysis is part of a targeted attack or the infection occurred rather by chance and is potentially related to digital crime, e.g. through a mass campaign using spam to infect as many targets as possible. A targeted attack is often indicated by the nature of the malware used and may be uncovered by identifying additional compromises with similar malware within the environment of the originally affected system. This investigation is often performed by forensics.

In the following, dynamic analysis will refer to techniques requiring execution of the code subject to analysis, whereas static analysis operates without execution.

A. ANALYSIS ENVIRONMENT

Before outlining the workflow, a typical cooperative work environment tailored to our workflow is presented. The structure of this environment shall maximise flexibility. By design it is not bound to static infrastructure in order to suffice the requirements of possible incident response tasks that may be connected to investigations outside of our own facilities. An overview is given in figure 3.

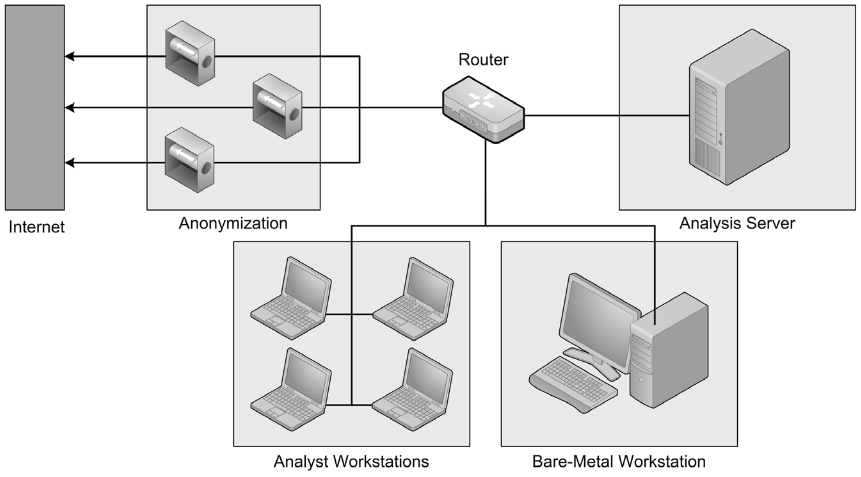

Figure 3. Structure of the work environment the workflow was developed for.

Every analyst uses his own workstation, which usually is a laptop in order to ensure mobility. The main requirement to the hardware is being able to run multiple virtual machines in parallel. This allows simulation of small networks within one workstation. The virtual machine images used for analysis are derived from identical templates per laptop in order to ensure reproducibility across the different hardware instances. Operating systems used for analysis are oriented on the most common variants found in the wild [10], equipped with typical consumer software (e.g. document processors, web browsers) in different patch levels. Furthermore, each workstation has a baseline of similar commonly used analysis tools as needed during the different phases. A template of this basic setup is maintained centrally and can be rapidly rolled out to new devices. Should the malware be virtualization aware to a degree that seriously aggravates the progress, analysts can temporarily shift to bare-metal machines.

The analysts' workstations are complemented by an analysis server enabling collaboration. As described in section 2, cooperative malware analysis faces challenges that can be partly overcome by providing services to the analysts. The analysis server hosts the documentation and task management system, as well as the case repository. In our implementation of the workflow, we use the following three tools to reflect these services:

- Confluence [11], a web-based wiki as documentation system
- JIRA [12], a web-based project tracking tool as task management system
- Stash [13], a web-based Git [14] management software as case repository

If the workflow is more targeted towards incident handling and less for in-depth analysis, the freeware RTIR [15] is a considerable option. Furthermore, special services such as collabREate (see section 3.D) can be hosted on this machine.

All machines are linked by a router, configured as a firewall. Internet Access is provided through a consumer uplink, optionally achieving anonymization through Tor [16] into a VPN provider in order to be able to control the outside origin IP address. This is useful when investigating threats with regional limitations. However, this uplink or permission to use the Internet for research at all depends on the nature of the investigated case.

B. PHASE 1: INITIALIZATION

The goal of the first phase is to set up the framework for the start of a new analysis case. Optimal starting conditions are created for the team by using baselined analysis templates. It is important to continuously maintain these templates during non-analysis periods in order to be able to start the actual analysis without any delay.

1) Initialization of the Work Environment

During the first step, the work environment is prepared. A new repository is created for hosting files that will accumulate during the work on this case. Depending on the tasks performed, for example this can include one more malicious executables or droppers, memory dumps taken during execution, network traces recorded while examining the malware, short snippets of code aiding the analysis, and related documents obtained during research. In parallel, a new space is created within the documentation system in order to allow collaboration on the documentation to be created for the case. This setup takes at most a few minutes.

2) Creation of Analysis Context

Next, in an initial meeting the scene for the analysis case can be set. The goal is to create context and a roadmap for the team of analysts. The information known on the incident thus far is discussed and inserted in the documentation system. This includes details on when and how the incident was recognised, all data available to the analysts such as potentially relevant files and network traffic recordings as well as the original role of the affected system in order to derive a threat scenario, e.g. a laptop used for travelling, a workstation, or a web server facing the Internet. Points of contact and communication channels are defined in case of questions that may arise. Constraints for the analysis with regard to certain search terms, network access for the malware sample etc. are defined in case of a need for confidentiality.

3) Definition of Analysis Goals and Initial Tasks

The final step for the first phase is a definition of overall analysis goals. Example goals are the extraction of indicators of compromise (IOC) to allow detection on related systems, the identification of remote communication entities or a description of the malware's functionality. The more specific these goals are, the more constrained and focused an analysis can be performed. Each goal should be coupled to an abort criterion, which can be either a timeframe or result and is agreed upon between analyst team and client. At this stage, potential obstacles that may arise during analysis should be outlined. Goals can also be adjusted based on findings later on. From these goals, initial tasks are derived and scheduled in the first sprint planning. The first sprint usually targets to complete the preliminary analysis as defined in phase 2.

C. PHASE 2: PRELIMINARY ANALYSIS

After framing the case in the first phase, a preliminary analysis can be conducted. The goal of this phase is to obtain a first impression of the characteristics and effects caused by the malware to assess its threat potential. This phase also serves as a fundament for all further analysis phases by providing indicators, which can influence steps taken in the following phases. All interim results produced in this and later phases are directly shared with relevant parties such as the security operations centre through the documentation system to allow taking mitigative actions. This is a common good practice as described in [17] and [18]. The second phase of the workflow iterates around three central activities: classification, behavioural analysis, and unpacking to enable deeper analysis. Iterations are closed with an evaluation of the current analysis state.

1) *Classification*

The step of classification during this phase targets the malware sample as a whole. The focus lies on static analysis in a cursory manner, using tools to automatically extract and assess features. This involves various techniques that can be compared for establishment of a hypothesis on the identity of the malware. In case of a well known specimen, there might be publicly available report coverage on this family that can be incorporated into the case, aiding the speed of analysis. Calculated hashes of suspicious file serve as unique fingerprints. Identification of the file format and examination of its fields (e.g. PE header of an executable) as well as statistical measures such as file entropy constitute the file's outer appearance. Some tools provide detection mechanisms for well-known protection schemes or can help with identifying the programming language or compiler used. Strings and other constants present in the data ascertainable by pattern matching sharpen the picture, especially "low hanging fruits", such as domain names, IP addresses, suspicious registry keys and file names, or similar. It has to be noted that these features are usually not immediately visible due to packing. In this case, a memory dump using a framework like Volatility [19] can serve as a rough approximation to unpacking as detailed later in this section. Furthermore, scans with locally available commodity antivirus software or a matching against the database of online services such as VirusTotal [20] can give hints on the family. Additionally, outstanding data fragments such as very expressive strings can be examined with methods of Open Source Intelligence [21]. However, it has to be kept in mind that such data can always be forged in order to mislead an analyst.

2) *Behavioural Analysis*

The goal of behavioural analysis is to gain an insight on how the malware affects the system during and after infection. Both sandboxing and blackboxing can be either performed in parallel to get a perspective from different tools or blackboxing can be used to explore the pointers obtained from sandboxing in more depth. Fully automated sandboxing can give valuable hints on the malware's interaction with the victim system. Results are usually stored machine-readable and accompanied by comprehensive presentation of analysis results. However, sandboxes may lack granularity desired by the analyst. Regardless of the approach, a selection of optionally fake network services should be provided to the analysis system to potentially increase the malware's execution depth by allowing basic network availability checks to succeed. This can be achieved e.g. with InetSim [22]. Nevertheless it has to be kept in mind that the execution behaviour of malware in a sandboxed or blackboxed environment can drastically differ and have more expressiveness if the malware has access to its C&C channel.

3) Unpacking

The third step in the second phase is unpacking, if necessary. Oftentimes, a proprietary protection scheme is used on the malware in order to evade detection by antivirus software and aggravate analysis [23]. However, due to the commercialization of the malware economy, "crypting" of binaries is offered as a service. In many cases the packing layers only wrap the original malware which is then loaded and executed directly from memory, similar to the technique described in [24]. As a result, by intercepting execution at the right moment, the original binary can be recovered from memory prior to its execution. In some cases, the recovery process involves manually reconstructing parts of the original binary e.g. fixing its API imports. Another step of unpacking is removing additional layers of obfuscation if applied to the binary. The recovery of a "clean" unpacked binary is crucial for a success in the following phases and thus deserves special attention. Optimally, the recovered binary is executable after freeing it from its protection schemes. For the further explanation of the workflow, we assume that a binary ready for deep analysis has been recovered in this step and consider details of actual unpacking process as out of scope of this paper.

4) Evaluation

As has been mentioned in the beginning, the three steps are iterated in multiple evaluation stages, as an unpacked malware sample can provide additional insight and can be more accurately classified and behavioural results can be incorporated into classification.

D. PHASE 3: IN-DEPTH ANALYSIS

The third phase shifts the analyst's view from the outside to the inside of the malware, towards the actual code level. The goal of this phase is gaining detailed understanding of the inner workings of the malware. A solid understanding of these aspects serves as a basis for threat assessment and successful mitigation. In this phase, both static analysis of disassembled code as well as dynamic analysis by debugging selected code fragments are used. The proposed workflow heavily relies on reverse engineering via static analysis. The preferred tool for this type of analysis is Hex Rays' IDA Pro [25] and its ecosystem of extensions and plug-ins such as the Hex Rays Decompiler [26] that can convert disassembly to pseudo code text resembling the C programming language.

1) Pre-Processing

As reverse engineering on machine level usually is very time-consuming, it is worthwhile spending time on a pre-processing step for reducing the expected analysis effort. This can be done e.g. by trying to identify known algorithms imported from library code, recognition of formerly documented functions by matching their characteristics against repositories of already known functions. Another technique is prioritising the analysis on potentially interesting portions of the disassembly, e.g. by extracting the occasions of system API usage, hinting at the higher-level semantics represented by the respective fractions of the code. This is possible because it can be assumed that a compiled binary roughly reflects the structure of its source code [27].

The identification of library code is supported by IDA Pro's FLIRT [28]. Additionally to the shipped modules, further signature databases are available [29]. Another approach for identification is automating queries with constants found in disassembled code against search engines [30], for example Google code.

Similar to the use of publicly available library code, code reuse in malware can be regularly observed as well. This goes without saying for specimen of the same malware family advancing over time. Therefore, the recognition of formerly documented functions is a promising attempt to reduce reverse engineering effort. Zynamics' BinCrowd [31] was such an approach, using the BinDiff technology for a centralised repository, but the product has been discontinued. Out of their own need, CrowdStrike have made the CrowdRE tool and repository publicly available [32]. This service allows the exchange of annotations created with the Hex-Rays Decompiler plug-in, based on both exact and fuzzy matching of functions. As the service is only available in conjunction with the proprietary database provided by CrowdStrike, limitations to application of this service can arise when working on classified cases. Instead of trying to match individual functions against a growing repository, the tool collabREate [33] focuses on keeping annotations consistent among the IDA Pro databases used by instances of multiple analysts. The data exchange happens in real time and thus contributes to synchronization of the analyst's view. A direct transfer of annotations from one IDA database to another based on function matching can be achieved with BinDiff, which is useful for migrating annotations along consecutive versions of related malware.

Besides the incorporation of existing knowledge into the case, heuristics can be applied in order to increase orientation within the code. IDAscope [34] is an IDA Pro plug-in that aims at helping to identify interesting parts of the binary. By analysing the frequency and usage of selected API calls for all functions, it can lead to semantically interesting locations. For instance, the presence of API calls for network and file access within the same function can be taken as an indicator for download functionality. The plug-in furthermore implements detection of cryptographic algorithms both based on signatures and a heuristic approach as described in [35].

2) Reverse Engineering

After these steps of pre-processing, the further analysis strategy depends strongly on the individual case.

Dynamic analysis is used to complement static analysis wherever it appears to be of avail. For example, observing the generation of system-dependent dynamic values or processing of communication through debugging the respective code fragments can drastically speed up figuring out details of the algorithms used. Debugging can also be used to prove reasoning about functional aspects that may have been made during static analysis. Well-proven debuggers for this task are e.g. OllyDbg [36] and WinDbg [37] on Windows and gdb [38] on Linux operating systems.

In general, there exist three main categories the analysis can be directed towards. They immediately benefit typical goals of the final phase. The three categories are: nesting strategy, functional capabilities including potential spreading mechanisms, and the communication protocols used.

An exact understanding of the malware's nesting strategy allows the creation of tailored detection or even protection methods based upon its indicators of compromise. This is covered in more detail with the description of detection in phase 4.

The in-depth analysis of functional capabilities can serve as a base for threat assessment. Identifying information stealing capabilities gives a clue on potentially exfiltrated data. It is noteworthy that an analysis through reverse engineering can reveal functionality hidden in the binary that has not yet been observed active in the wild or during prior analysis phases, e.g. during sandboxing. The reason for this is that most functionality in malware is triggered by specific commands that have to be given by the actor using the malware. Further functionality of interest can be update or downloading behaviour, potentially widening the degree of compromise.

Finally, understanding the communication protocol and likely used cryptographic routines enables the analyst to decipher traffic generated by the malware or imitate the malware in order to extract information from the C&C entity. Furthermore, this part of analysis may reveal potential backup communication channels that have to be taken into concern when planning countermeasures.

In our workflow, we use both collabREate and CrowdRE for synchronization and data exchange within IDA Pro. While collabREate's real time updates serve as immediate notifications indicating the functions currently being examined by the individual analysts, CrowdRE is used to submit the documentation of decompiled functions after their analysis is finished.

In addition to this synchronization on a technical level, periodic but time-boxed meetings in person or via voice based conferencing software are used to exchange information on the latest progress. This procedure is an adoption of the stand-up meetings known from Scrum [8]. The intervals between those meetings depend on the time criticality of the case, with typically one up to four meetings per day. The meetings are timeboxed with about 15 minutes shared among all analysts in which they report their findings oriented on the following three questions:

- What did I accomplish during the last time box?
- What am I going to do in the next time box?
- What problems may I face in my analysis?

E. PHASE 4: PROVIDING EVIDENCE AND LONG-TERM MITIGATION CONCEPTS

To not have the malware analysis case end in itself, the final phase aims at providing tailored mitigation concepts for the malware specimen. Evidence in form of a written documentation on the analysis case is already available at this point as it has been created throughout the former phases alongside analysis. The mitigation concepts considered for this paper represent only a selection of possibilities and cover the following aspects: detection, tracking, and active countermeasures. Management of tasks connected to the mitigation strategies can again be achieved with the adapted Scrum. It has to be noted that parts of the proposed methods are potentially in conflict with given law in some countries and should be considered by legitimated authorities such as law enforcement only.

1) Detection

As has been mentioned in the description of phase 3, a thorough understanding of the malware's nesting strategy can lead to comprehensive detection methods, extending a pure signature based approach. In case of deterministic IOCs, reliable detection is possible and even the creation of a temporary tool serving as a virtual vaccine can be considered to protect other machines [39]. Network based detection nowadays is increasingly challenging as it has to be assumed that the traffic generated by malware is completely encrypted. One common product of in-depth analysis of the communication protocol is the identification of C&C entities such as domain names or IP addresses used as point of contact. While being less generic, these serve as primary indicators for network based detection. In some cases protocol characteristics such as fixed ports or characteristic data fragments e.g. caused by use of a static encryption key can be identified that are sufficient for detection. A custom traffic decrypter tailored to the malware is also of high value in case of available full packet captures recorded during the incident.

2) *Tracking*

A prerequisite to tracking a malware's C&C channel is the understanding of its communication protocol as obtained during phase 3. One possible goal of tracking is being able to monitor the channel for commands, updates, and changes to the C&C infrastructure. Potential use cases are the extraction of templates for spam mails to build better filters [40], get knowledge about announced targets of DDoS attacks [41], or track the evolution of a malware specimen by mining new malware samples and analysing the differences to preceding versions. In case of a distributed architecture (P2P botnet), implementation of a crawler [42] or sensor [43] is an option in order to globally identify infected machines and inform affected parties.

3) *Active Countermeasures*

As far as active countermeasures are concerned, various options exist. Based on the identified C&C points of contact, an abuse notification to the responsible registrars and hosting providers can be issued. In case of neglect of these notifications, an attempt can be made to orchestrate a takedown supported by a court of law [44]. In case of a P2P-based C&C channel, there may be the option to abuse protocol characteristics in order to achieve a sinkholing effect or partition the network until it is rendered unusable.

The mitigation concepts presented likely require the creation of proof-of-concept or production code in order to be carried out. The utilization of Scrum as process management paradigm in the proposed workflow as well as the structure of the analysis environment support this naturally and allow seamless mixing or shifting of tasks with a focus from analysis to software development. For software development, it should be adhered to known good practices [45]. Especially providing tests for code can serve both as illustrative examples of usage and ensure stability of rapidly prototyped projects.

4. RELATED WORK

In this paper, we addressed challenges to cooperative malware analysis and proposed a structured workflow to overcome them. We are not aware of directly comparable work, addressing both aspects of human collaboration and different progressive phases of malware analysis.

In [46], Wedum describes a systematic approach to malware analysis divided in three phases. An overview of common methodologies used in malware analysis is given by Sikorski in [47] and an explanation of selected tools and techniques with their respective use cases is provided by [48]. In [49], Willems and Freiling give a survey on reverse engineering and countermeasures. Song et al. have developed a BitBlaze, a system for binary analysis [50] usable in the context of malware.

A selection of frameworks for synchronization of analysis results [31, 32, 33] has been already covered in section 3.D.

A comparative survey on automated dynamic analysis systems has been performed by Egele et al. [51]. Blaszczyk discusses automation versus in-depth malware analysis in [52].

5. CONCLUSION

In this paper we gave a thorough overview of a proposed cooperative malware analysis workflow. By adapting selected elements of the Scrum methodology, challenges originating from collaboration such as need for synchronization and work partitioning have been targeted.

A major issue with an efficient workflow in general is the lack of inter-operability of analysis and documentation tools. Oftentimes, the output of analysis tools has to be extensively edited in order to be usable in a report.

Another area with high potential for improvement is further semi-automation of static analysis as described in phase 3. Currently, best practices for recovering details of functionality from binary code are connected to tedious human efforts. Further exploitation of methods for recognising and classifying the structure of the program and its control flow may speed up the analysis by providing better orientation and understanding the relationship of functions and interactions.

There already exist examples of malware specimen, where an effective mitigation can only be derived from a deep understanding of their communication protocol strategy due to the nature of their distributed command and control channels [53, 54]. Thus, we firmly believe it is worthwhile and necessary to research into the optimization of a workflow for malware analysis, especially targeting families as a whole.

REFERENCES

All websites successfully retrieved on January 09th, 2013.

[1] W32.Stuxnet Dossier. Falliere, N., O Murchu, L., Chien, E. Symantec Whitepaper, 2011.

[2] Protecting Your Critical Assets - Lessons Learned from "Operation Aurora". McAfee, 2010.

[3] Shamoon the Wiper - Copycats at Work. Kaspersky GReAT, 2012.

[4] The Malware Analysis Body of Knowledge (MABOK). Valli, C., Brand, M. Published by: Edith Cowan University, School of Computer and Information Science, 2008.

[5] Awareness and coordination in shared workspaces. Dourish, P., Belotti, V. In: Proceedings of the 1992 ACM conference on Computer-supported cooperative work (CSCW), 1992.

[6] Work and the Division of Labor. Strauss, A. In: The Sociological Quarterly 26 (1), 1985.

[7] Continual Permutations of Action. Strauss, A. Published by: Aldine de Gruyter, 1993.

[8] The Official Scrum Rulebook. Schwaber, K., Sutherland, J. E-book, published 2011 on: http://www.scrum.org/Scrum-Guides

[9] Syntactic Theory of Visual Communication. Lester, P. Essay, published 2006 on: http://commfaculty.fullerton.edu/lester/writings/viscomtheory.html

[10] OS Platform Statistics and Trends. http://w3schools.com/browsers/browsers_os.asp

[11] Confluence, Team Collaboration Software. http://atlassian.com/software/confluence

[12] JIRA, Issue and Project Tracking Software. http://www.atlassian.com/software/jira

[13] Stash, Enterprise Git Repository Management. http://www.atlassian.com/software/stash

[14] Git – distributed version control system http://git-scm.com

[15] RT for Incident Response. http://bestpractical.com/rtir

[16] Tor Project. https://www.torproject.org

[17] Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. Hutchins, E., Cloppert, M., Amin R. In: The 6th International Conference on Information-Warfare & Security, 2010.

[18] Exploring Security Countermeasures along the Attack Sequence, Sakuraba, T., Domyo, S., Chou, B., Sakurai, K.. In: Proceedings of the 2nd International Conference on Information Security and Assurance ISA, 2008.

[19] The Volatility Framework. https://www.volatilesystems.com/default/volatility

[20] VirusTotal. http://virustotal.com

[21] NATO Open Source Intelligence Handbook, 2001.

[22] InetSim: Internet Services Simulation Suite. http://www.inetsim.org/

[23] Binary-Code Obfuscations in Prevalent Packer Tools. Roundy, Miller, B. Published by: University of Wisconsin, 2012.

[24] Dynamic Forking of Win32 EXE. http://www.security.org.sg/code/loadexe.html

[25] Hex-Rays IDA Pro. www.hex-rays.com/products/ida/

[26] Hex-Rays Decompiler Plugin. http://www.hex-rays.com/products/decompiler/

[27] Recovering the Toolchain Provenance of Binary Code, Rosenblum, N., Miller, B., Zhu, X. In: Proceedings of the International Symposium on Software Testing and Analysis (ISSTA), 2011.

[28] IDA FLIRT Technology: In-Depth.
http://www.hex-rays.com/products/ida/tech/flirt/in_depth.shtml

[29] IDA FLIRT Signatures. http://woodmann.com/collaborative/tools/index.php/Category:IDA_FLIRT_Signatures

[30] RE-Google. http://regoogle.carnivore.it/

[31] BinCrowd. http://www.zynamics.com/bincrowd.html

[32] CrowdRE. https://crowdre.crowdstrike.com

[33] CollabREate. http://www.idabook.com/collabreate

[34] IDAscope. http://idascope.pnx.tf

[35] Dispatcher: Enabling Active Botnet Infiltration using Automatic Protocol Reverse-Engineering. Caballero, J., Poosankam, P., Kreibich, C., Song, D. In: Proceedings of the 16th ACM conference on Computer and communications security (CCS), 2009.

[36] OllyDbg. http://http://www.ollydbg.de/

[37] Debugging Tools for Windows (WinDbg).
http://msdn.microsoft.com/en-US/windows/hardware/gg463009/

[38] The GNU Project Debugger. http://www.gnu.org/software/gdb/

[39] Using Infection Markers as a Vaccine against Malware Attacks. Wichmann, A. Gerhards-Padilla, E. In: Proceedings of the 2nd workshop on Security of Systems and Software resiLiency (3SL) in conjunction with IEEE International Conference on Cyber, Physical and Social Computing (CPSCom), 2012.

[40] Spamcraft: An Inside Look At Spam Campaign Orchestration. Kreibich, C., Kanich, C., Levchenko, K., Enright, B., Voelker, G., Paxson, V., Savage, S. In: Prcoceedings of the 2nd Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET), 2009.

[41] Active Botnet Probing to Identify Obscure Command and Control Channels. Gu, G., Yegneswaran, V., Porras, P., Stoll, J., Lee, W. In: Proceedings of the Annual Computer Security Applications Conference (ASAC), 2009.

[42] Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F. In: Prcoceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET), 2008.

[43] Towards Complete Node Enumeration in a Peer-To-Peer Botnet. Kang, B., Chan-tin, E., Lee, C.P., Tyra, J., Kang, H., Nunnery, C., Wadler, Z., Sinclair, G., Dagon, D., Kim, Y. In: Proceedings fo the ACM Symposium on Information, Computer & Communication Security (ASIACCS), 2009.

[44] Bredolab Takedown, Another Win for Collaboration. Williams, J., 2010. http://blogs.technet.com/b/mmpc/archive/2010/10/26/bredolab-takedown-another-win-for-collaboration.aspx

[45] Clean Code, A Handbook of Agile Software Craftsmanship. Martin, R. Published by: Prentice Hall International, 2008

[46] Malware Analysis; A Systematic Approach. Wedum, P. Published by: Norwegian University of Science and Technology, Department of Telematics, 2008.

[47] Practical Malware Analysis. Sikorski, M. Published by: No Starch Press, 2012.

[48] Malware Analyst's Cookbook. Ligh, M., Adair, S., Hartschein, R. Published by: John Wiley & Sons, 2010.

[49] Reverse Code Engineering - State of the Art and Countermeasures. Willems, C., Freiling, F. In: it - Information Technology 54 (2), 2012.

[50] BitBlaze: A New Approach to Computer Security via Binary Analysis. Song, D., Brumley, D., Yin, H., Caballero, J., Jager, I., Kang, M., Liang, M., Newsome, J., Poosankam, P., Saxena, P. In: Proceedings of the 4th International Conference on Information Systems Security (ICISS), 2008.

[51] A Survey on Automated Dynamic Malware Analysis Techniques and Tools, Egele, M., Scholte, T., Kirda, E., Kruegel, C. In: ACM Computing Surveys 44 (2), 2012.

[52] Automation vs. In-depth Malware Analysis. Blaszczyk, A. Essay, published 2012 on: http://www.hexacorn.com/blog/2011/11/21/automation-vs-in-depth-malware-analysis

[53] What we know (and learned) from the Waledac takedown. Williams, J., 2010. http://blogs.technet.com/b/mmpc/archive/2010/03/15/what-we-know-and-learned-from-the-waledac-takedown.aspx

[54] The Lifecycle of Peer-to-Peer (Gameover) ZeuS. Stone-Gross, G., 2012. http://www.secureworks.com/cyber-threat-intelligence/threats/The_Lifecycle_of_Peer_to_Peer_Gameover_ZeuS/