MASTER THESIS

Approach on Malware Sophistication Measurement Based on Leaked CIA Documents

Author:
Nils KUHNERT

Supervisors:
Dr. Nhien-An LE-KHAC
Robert MCARDLE

A minor thesis submitted in part fulfillment of the degree of Master of Science in Forensic Computing and Cybercrime Investigation

in the

UCD School of Computer Science

UNIVERSITY COLLEGE DUBLIN

July 30, 2021

UNIVERSITY COLLEGE DUBLIN

Abstract

UCD School of Computer Science

Master of Science in Forensic Computing and Cybercrime Investigation

Approach on Malware Sophistication Measurement Based on Leaked CIA Documents

by Nils KUHNERT

Sophistication of adversaries and malware families is an often used and ambiguously defined term in the cyber threat intelligence (CTI) domain. Also, there is no common ground for sophistication assessments which results in statements biased by the situation, objective and experience. Victims and public relations often tend to overestimate the sophistication of threats as an excuse or marketing instrument while analysts just cannot agree on a sophistication assessment because of their different experience levels. This research project introduces an approach for measuring the sophistication of malware families based on CIA development guidelines leaked in the Vault 7 collection. Converting the guidelines to a list of criteria and adjusting them to the scope of malware reports, this approach can be used as a checklist to create assessments and verify sophistication statements made in malware reports. The application of this approach against 25 malware families was evaluated with data obtained through a community survey. Participant's sophistication estimations aligned with more than half of the assessments and most of the differences are by one category only. Overall the introduced methodology assesses sophistication more critically and the first results are promising, however, the approach is not production-ready, but with additional research and adjustments, it can become a solid base for overall sophistication assessments.

Acknowledgements

First I would like to thank my supervisors, Dr. Nhien-An LE-KHAC and Robert MCARDLE. Without their guidance this thesis would not have been possible. They consistently steered me into the right direction, provided me with their experience and despite Covid-19 being an additional burden to all of us, they always were just a zoom call away.

Then I would like to show my gratitiude to Dr. Timo STEFFENS. Timo initially brought this topic to my attention, was always available for fruitful idea ping-pongs and introduced me to the field of threat intelligence and attribution some years ago.

Also, I would like to thank Daniel PLOHMANN and contributors for Malpedia. It is a very useful platform for researching malware families and especially finding malware reports as well as other source documents to build upon. It was a great help during the work on this thesis.

Without the help of the threat intelligence community, the evaluation of my results would not have been possible. While the survey itself was anonymous and I can not know who participated, I would like to thank you this way.

Last but not least, I would like to thank my family and friends for supporting, motivating and enduring me during this research phase. Their continually backing allowed me to complete this thesis and to challenge the procrastinator which is inside all of us.

Contents

List of Tables

List of Abbreviations

AED	**A**pplied **E**ngineering **D**ivision
API	**A**pplication **P**rogramming **I**nterface
APT	**A**dvanced **P**ersistent **T**hreat
BSC	**B**ase **S**tation **C**ontroller
C2 or **C&C**	**C**ommand and **C**ontrol
CA	**C**ertificate **A**uthority
CCI	**C**enter for **C**yber **I**ntelligence
CIA	**C**entral **I**ntelligence **A**gency
CIRCL	**C**omputer **I**ncident **R**esponse **C**enter **L**uxembourg
COM	**C**omponent **O**bject **M**odel
CTI	**C**yber **T**hreat **I**ntelligence
DES	**D**ata **E**ncryption **S**tandard
DIE	**D**etect **I**t **E**asy
EA	**E**xtended **A**ttribute
EAT	**E**xport **A**ddress **T**able
EDG	**E**ngineering **D**evelopment **G**roup
EDR	**E**ndpoint **D**etection and **R**esponse
ELF	**E**xecutable and **L**inkable **F**ormat
GSM	**G**lobal **S**ystem for **M**obile Communication
IAT	**I**mport **A**ddress **T**able
ICS	**I**ndustrial **C**ontrol **S**ystems
IDS	**I**ntrusion **D**etection **S**ystem
ISO	**I**nternational **O**rganization for **S**tandardization
JSON	**J**ava**S**cript **O**bject **N**otation
MAEC	**M**alware **A**ttribute **E**numeration and **C**haracterization
NDIS	**N**etwork **D**river **I**nterface **S**pecification
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
NOD	**N**etwork **O**perations **D**ivision
NOFORN	**No For**eign **N**ationals
NSA	**N**ational **S**Security **A**gency
OPSEC	**Op**erations **Sec**urity
PDB	**P**rogram **D**atabase
PE	**P**ortable **Executable**
PGP	**P**retty **G**ood **P**rivacy
PLC	**P**rogrammable **L**ogic **C**ontroller

RAT	**R**emote **A**ccess **T**rojan
RC4 (5, 6)	**R**ivest **C**ipher **4** (Also: 5, 6)
RFC	**R**equest **F**or **C**omments
STIX	**S**tructured **T**hreat **I**nformation E**x**pression
UTC	**U**niversal **T**ime **C**oordinated
VB(S)	**V**isual **B**asic (**S**cript)
VFS	**V**irtual **F**ile **S**ystem
WMI	**W**indows **M**anagement **I**nstrumentation
XTEA	E**x**tended **T**iny **E**ncyption **A**lgorithm

Chapter 1

Introduction

Many threat reports and articles mention the sophistication of specific groups or malware families, especially in cases where an advanced persistent threat (APT) is involved. Knowing the sophistication of threat groups or malware families is important in the domain of cyber threat intelligence (CTI) in two different ways. First, the higher the sophistication, the harder it is for defenders to react to an intrusion in a successful way. Knowing the sophistication of adversaries that are part of a target's threat model allows the defenders to prepare adequately which might involve planning additional resources, in case the sophistication is above average. Second, the sophistication of adversaries or malware families is proportional to their operational security (OPSEC), raising the difficulty of analysis. So not only directly affected people need to adjust their resources accordingly, but also analysts and researchers.

While sophistication is a broadly used term in the domain of information security, there is no standardized way of measuring or calculating a sophistication level. Analysts are required to assess sophistication based on their experiences and this results in different analysts having different estimations of sophistication for the same threat group or malware. Additionally, victims of cyberattacks tend to overestimate sophistication rather than acknowledging mistakes made [9]. Other areas where sophistication overestimation of threats and malware typically occur are marketing and news media reporting. The motivation for this research is to create a measurement approach for the sophistication of malware which can be applied by both, analysts writing malware reports and readers of those reports, in order to create and verify an understandable assessment. Additionally, analysts exchanging sophistication assessments can use it as a common ground regardless of their experience level.

The National Institute of Standards and Technology (NIST) describes advanced persistent threats as adversaries with a sophisticated level of experience [50]. This research covers the malware part of said experience and introduces an approach for measuring the sophistication of malware families. Research questions are covered in chapter 3. The initial base for creating the measurement approach is a CIA document leaked within the Vault 7 collection [148]. Containing rule-like requirements for developers, the "tradecraft DOs and DON'Ts" provide rare insights into

the methodology of a sophisticated threat actor. Hence, they were converted to criteria that can be matched against malware analyses. In a second step the criteria were adjusted in order to fit the information scope of malware reports. Beside the adjustment to malware reports, a score calculation procedure is presented in chapter 4, including the assignment to five sophistication levels. In chapter 5 the adjusted criteria set was applied to 25 malware families or more specifically, their public analysis reports. In order to evaluate the assessments, chapter 6 compares them to the data of a conducted survey which participant's estimated the sophistication of the malware families. More than half of the assessments align with estimations from the community, while the ones that differ are assessed lower by the introduced approach and most of them are off by only one category. At the end of chapter 6, differences are discussed in detail and while the approach is not a production-ready to use assessment methodology, the results are promising. Further research topics advancing the introduced approach are describes in chapter 7.

Chapter 2

Literature Survey

The main research area of this project, malware sophistication measurement, is not well researched. While there aren't published studies of that specific area, this research relates to the topics of sophistication in cyber operations, malware threat scoring, i.e. how severe a malware is, and malware capabilities. The following sections introduce published studies in the related areas and differentiate between them and this research project. Of particular interest for this paper is an excerpt of internal CIA documentation for developers working for the Applied Engineering Division (AED). This document, leaked in the context of Vault 7 [148], will be covered in a separate section.

2.1 Sophistication

Sophistication is a rather ambiguously defined, but nevertheless frequently used terminology in cyber security. The Oxford Dictionary defines the sophistication in terms of a machine or a system as "clever and complicated in the way that it works or is presented" [114]. The National Institute for Standards and Technology (NIST) uses the term to define advanced persistent threats as adversaries with sophisticated levels of experience [50]. According to Buchanan, the term *sophistication* is often reduced to a "recognized on sight" type of definition. Sophistication would be used as a synonym to operational success instead of as a quality metric for cyber operations. Buchanan summarizes a sophisticated cyber operation as well-thought, counter-intuitive, complex and hard to execute [9]. Two examples were given as sophisticated malware: Stuxnet and Project Sauron. While the first manipulates programmable logic controllers (PLC) to slowly destroy centrifuges, the latter remained undetected for a long time because of the high level of OPSEC. Buchanan also explains the value of sophistication assessments to defenders as an opportunity to be better prepared for potential adversaries, because of a better understanding of the trade-offs a specific threat has taken to achieve its goals. According to Buchanan, intruders are just as sophisticated as they need to be, so a differentiated understanding of sophistication has also its limits from a strategic perspective. Both, public media and victims of cyber attacks, tend to overestimate the sophistication of threat actors for various reasons. First, the term sophistication seems to be misunderstood in a

way that success of operations indicate a sophisticated adversary. But even well resourced, carefully planned and complex operations by, e.g., security services such as the NSA can fail and the NSA seems to be very sophisticated in cyber operations. Buchanan also describes this paradigm of overestimating adversary sophistication or overstating the capabilities as an excuse by the victims, so they do not need to face their own failing [9]. Also mentioned in that paper is a framework by Dave Aitel [2] which can be used for measuring the "power" of cyber operations. The mentioned framework uses six different categories - Sourcing, Usage, Networking, Testing, Persistence and Operational Security (OPSEC) - as metrics with various criteria. The criteria advance from basic and cheap to implement to more complex ones, suggesting a higher sophistication. Aitel broke down the framework based on his experiences in designing and writing malware implants as there are lots of categories or criteria that could be added.

DePaula and Goel have researched the methodology of assigning a sophistication index to security breaches from 2005 to 2015. The underlying data bases on a survey presented to security professionals which asked about the sophistication of implemented features or criteria and how sophisticated specific incidents were. The paper also mentions the very limited availability of cyber attack sophistication related research in general and, more specifically, missing metrics for sophistication of cyber attacks [22]. The sophistication index introduced by DePaula and Goel concentrates on five different feature types that have to be implemented by cyber attacks to consider them sophisticated: social engineering, remote administration, stealth, zero-day vulnerability exploits, APT. For all feature types the paper includes examples. The index itself can be described as $S_{\text{idx}} \in \mathbb{N}, 0 \leq S_{\text{idx}} \leq 5$ and for every implemented feature type, the index is incremented by one. While this allows to categorize breaches into six different sophistication levels, nuanced differences between attacks can not be considered.

Frye et al [33] implemented various metrics to describe or measure cyber threats. Through attributes, which originate from a generic threat model matrix, threats can be characterized with a common vocabulary. The attributes are grouped into two categories: commitment and resources. Commitment attributes are intensity, stealth and time, whereas resources attributes are technical personnel, cyber knowledge, kinetic knowledge and access. Beside the introduced threat model matrix, the researchers list additional, incident related metrics, one of them is "attack sophistication". Related to the threat attributes stealth, time, cyber knowledge and access, Frye et al measure the sophistication by the amount of systems compromised, multiple accesses to the compromised network and novel methods used (such as unknown vulnerabilities).

2.2 Malware Scoring

Malware scoring or rating is often just used for threat or severity rating of malware samples. Bagnall and French [3] introduced the Malware Rating System (MRS) which scores malware based on the payload potential, proliferation potential and hostility level. Using these three factors, the malware rating can be calculated and gets represented through a number between 0 and 100. The malware can be ranked into one of five categories from "minimal danger" to "catastrophic danger". An additional regional factor allows adjusting the malware rating to the own system landscape and allows considering, e.g., different operating systems.

Malhotra described a methodology for scoring the maliciousness of malware using threat trees which represent the malware's capabilities, behaviours and actions. The specific rating of the malware sample is calculated based on the tree's branch weights [71].

Maasberg, Ko and Bebe [70] developed an approach to malware threat assessment in order to provide a standardized way of threat evaluation. Their method relies on various factors: malware propagation, characteristics, attribution and impact. The sum of all factors is the total score describing the threat of the malware. Malware families are then categorized from one to five, based on their score, where a higher threat category represents a more dangerous threat. The given threat assessment methodology might be less useful for very targeted threats as a high propagation would result in a high rating, however, for sophisticated APTs a high propagation as well as a strong measurable impact is disadvantageous as it makes the malware more visible.

Malware sandbox analysis systems, such as Cuckoo [131], use threat scoring to report the maliciousness of uploaded files to the user. For instance, Cuckoo uses a community maintained repository of signatures where every signature has it's own severity score from 1 to 5. Cuckoo checks uploaded file against the signatures and reports a score based on the severity value given in the matching signature set [144].

2.3 Malware Capability Description

The capability of malware, synonymously used for implemented features and taken actions, are a core information for every malware analyst and are included in every malware report in one form or another. A common vocabulary for describing the capability of malware is needed in order to enable a successful information exchange. The Malware Attribute Enumeration and Characterization (MAEC) language schema provides a methodology for precisely describing malware and the according research [98]. The MAEC project provides a vocabulary that can be used to precisely reflect the environment used during analysis, behaviour of the malware, capability categories of the behaviour recognised, common attributes (such as used encryption algorithms or network protocols), delivery vectors, actions performed

by the malware, malware configuration parameters and many more [97]. MAEC objects are formatted in JavaScript Object Notation (JSON) and are therefore machine readable. Structured Threat Information Expression (STIX), a popular format for exchanging structured Cyber Threat Intelligence (CTI), makes use of MAEC to describe malware [111].

Not only limited to malware, MITRE ATT&CK is a knowledge base and framework for the behaviour of cyber adversaries considering all phases of an attack life cycle. The behavioural model of MITRE ATT&CK includes tactics, techniques and sub-techniques. Tactics can be seen as short-term goals of adversaries during an attack and ATT&CK lists reconnaissance, resource development, initial access, execution, persistence, privilege escalation, defence evasion, credential access, discovery, lateral movement, collection, command and control, exfiltration and impact as so-called enterprise tactics. There is also a similarly structured set of techniques for mobile devices, called mobile tactics. Techniques and sub-techniques represent the method used to achieve a specific goal, e.g. the initial access can be achieved using the technique phishing, or, more precisely, the sub-technique spearphishing attachment which means that a targeted victim opens a malicious file that has been sent as an email attachment by the attacker. ATT&CK also provides links between techniques, sub-techniques and known threat actors that use specific techniques. [132].

In 2020 FireEye researchers published a tool called *capa* which enables analysts to quickly identify a malware's capabilities. The tool uses a method called feature extraction by the authors which extracts information from the portable executable (PE) header, imported libraries and application programming interfaces (API), used mnemonics, constants and strings. In order to recognize specific capabilities, the extracted information are compared to a given rule set [123].

2.4 Distinction of Malware Sophistication Measurement

The above sections describe metrics related to malware threat measurement and forms of capturing malware capabilities, but these cannot be used for measuring the sophistication of malware. The former describe the severity or threat of a malware and facilitate a higher amount of implemented capabilities as well as a greater distribution of infected systems. This is fundamental different to measuring the sophistication of malware. As Buchanan describes it [9], one metric misused for sophistication is often the success of an operation which can be seen as proportional to the amount of overall infected systems. Additionally, the count of implemented capabilities would make a commodity remote access trojan (RAT) that implements a certain set of features very sophisticated, but a single-purpose dropper that implements a variety of OPSEC measures and anti-analysis techniques score less. The latter overlaps with this research as the criteria identified can be seen as some sort of capabilities, too. The difference is that the sophistication criteria developed also

cover different levels of implementation in order to distinguish the quality of implementations or the resources spent for implementation. While MITRE ATT&CK implement sub-techniques, the representation of certain features is not fine enough for sophistication measurement. For example, obfuscation is a central part of anti-analysis features in malware, however, this is just part of single MITRE ATT&CK technique, T1140, despite it is applicable on strings, payloads, configuration data etc. [95]. Also, the methodology of (de)obfuscation is not described further, although the used method can reveal a resource and knowledge gap between two malware families and therefore have a great impact on the assessed sophistication. This shows the use-case of capability representation and sophistication measurement differs.

2.5 CIA Development Tradecraft

In 2017 Wikileaks published a collection of leaked CIA documents called Vault 7. Originating from the Center for Cyber Intelligence (CCI), the documents contain information about the cyber operations of the CIA [148]. Some of the documents are related to the Engineering Development Group (EDG), more specifically to the Applied Engineering Division (AED) which is one of the divisions responsible for developing the CIA's cyber operation tool set [146]. Part of the AED related documents is a document called "Development Tradecraft DOs and DON'Ts" which describes general guidelines for malware developers within the AED. These OPSEC-centric guidelines should prevent attribution and detection of CIA cyber operations. Grouped in six categories (General, Networking, Disk I/O, Dates/Time, PSP/AV and Encryption), the guidelines list directives for the developers with according rationales.

Based on the documents included in Vault 7, CIA cyber operations can be seen as reasonable sophisticated. A document summarizes some of the mistakes made by the Equation Group and how the CIA can avoid their mistakes in own operations [149]. IT security company Kaspersky sees the Equation Group as "probably one of the most sophisticated cyber attack groups in the world" [53]. If the CIA is able to perform equally qualitative operations and produce similar capable tools, they are also as sophisticated. Security company Symantec connects the tools used by a group called Longhorn to the information, specifications and timelines in the Vault 7 leaks and assesses the operations done by the group as sophisticated [49]. However, the Council on Foreign Relations refers to "the technical analysis out there", without specific sources, which assesses the tools as not particular sophisticated [126]. Additionally, according to Check Point, the CIA tools have borrowed code from ordinary malware [14]. So, even though the CIA might not be the most sophisticated cyber threat actor, they seem to constantly refine their operations and learn from mistakes other adversaries made. Using their development guidelines as a starting point for creating a sophistication measurement approach is at least reasonable.

Chapter 3

Statement of Research Problem

3.1 Research Problem

As introduced in Chapter 2, sophistication is a quite ambiguously defined term. However, it is used broadly by news sites and PR departments to justify intrusions [9] as well as analysts in threat reports to describe the maturity or resources of a specific adversary. Typically labeled in three or five categories, either from low to high or from very low to very high, it is not clearly defined what a specific level of sophistication means.

While analysts typically have an understanding of what a specific levels of sophistication might mean, without a common understanding and a methodology on how to measure it understandably and reproducibly, analysts exchanging threat information cannot be sure to talk about the same type or level of sophistication.

Also the sophistication assessment strongly varies between various levels of analysis experience. A structured and reproducible approach is understandable by analysts regardless of their experience level.

To measure the overall sophistication of an adversary, the measurement can be broken down into it's components, e.g.:

- How the adversary sets up and uses their infrastructure
- How they carry out their operations or campaigns
- The victimology of the operation (amount of targets, composition of target set etc.)
- How the operators behave in a network once it is compromised
- How well the adversary writes malware

As there is currently no publicly existing metric or methodology for measuring the sophistication of malware families, this research project covers the latter part and provides a new approach. Using a set of criteria, analysts can review reports or analyze malware and assign a sophistication level based on the matched amount of criteria. Basis for the set of criteria developed in chapter 4 are the guidelines given in the AED document introduced in the previous chapter. While the criteria are similar

to malware capabilities (or capability measurement) already broadly discussed in the community, the underlying idea is not only the existence of specific features, but also how they are implemented. As sophistication is an ambiguous term, this research project uses the following definition for sophisticated malware:

Sophisticated malware implements features that make it harder to analyze, attribute and detect, uses multiple control channels, has a high OPSEC hygiene and exploits (multiple) vulnerabilities.

However, sophistication must always be assessed on a scale and cannot be seen as binary. This is why the approach of this research relies on several criteria described in later chapters. The criteria developed through the research project will reflect and complement the above definition. Some aspects can be covered through a multitude of criteria, e.g., OPSEC hygiene, whereas some aspects are easily measurable, e.g., the amount or type of used exploits. The amount of criteria matched by a malware sample is proportional to the level of sophistication assessed.

3.2 Research Questions and Hypotheses

1. **How can the AED guidelines can be applied for sophistication measurement?** The AED development tradecraft introduced in chapter 2 is a list of recommendations which have an emphasis on OPSEC related measures. Transformed into more compact criteria, the next sections review malware reports for implemented features that align with the given criteria set. The amount of matched features is an indicator of sophistication.

2. **Which of the criteria introduced are useful for sophistication assessment and which are not? Are there other criteria that are essential for evaluation and, hence, should be added?** While the criteria developed from the AED tradecraft document provide a good starting point, there are additional criteria which are useful for sophistication measurement. These can be found in malware reports.

3. **Which of the criteria is usually mentioned in reports and which are less likely to appear in published malware or threat reports? Which criteria need adjustment to fit the analysis scope of published reports? What happens if specific criteria are not mentioned in reports?** Some of the criteria identified will be less useful, because they either match every malware report evaluated or are not implemented by any malware family. Thus, criteria need to be adjusted in order to reflect the common malware implementations and malware reports. For every criteria a default value will be assigned that covers cases in which a criteria is not mentioned in malware reports.

Chapter 4

Adopted Approach

This chapter sequentially describes the methodology development for measuring malware sophistication. Starting with a brief introduction of the AED development guidelines, followed by two sections developing the criteria sets. The first one was directly derived from the CIA document and contains weaknesses found during a short evaluation based on malware reports. The second incorporates adjustments which are useful for assessing sophistication based on malware reports and implements sub-criteria in order to represent various degrees of implemented features.

4.1 CIA Development Guidelines

The "SECRET//NOFORN" (secret - no foreign national) classified document called "Development Tradecraft DOs and DON'Ts" [147] describes general paradigms important for software that should be used during CIA cyber operations. The goal of the guidelines is preventing attribution based on artifacts in the used malware, harden the analysis and make detection more difficult. Developers working for the CIA have to consider the following directives:

1. Strings and configuration data must be obfuscated and should only be deobfuscated in memory. To harden analysis, deobfuscated strings which are no longer needed should be removed from memory.

2. Decryption or deobfuscation routines should not be called upon execution. Otherwise analysts can use a debugger and set a breakpoint right after the routine to dump the decrypted or deobfuscated artifacts.

3. Sensitive data should be removed from memory right after usage in order to prevent plain-text artifacts in memory.

4. Keys used for encryption or obfuscation routines should be unique per deployment. Key reuse allows analysts to cluster samples and result in a faster progression of analysis of other cases involving samples of the same malware family.

5. Build related symbols and manifests should not be included in the final binary in order to harden analysis. The availability of symbols helps analysts to understand the malware's features and build paths are often used for clustering malware.

6. No debugging output should be left in the final release of the tool as it can disclose malware functionality.

7. Functions which would raise the suspicion of analysts should not be imported or called directly. The guidelines mention `WriteProcessMemory`, `VirtualAlloc` and `CreateRemoteThread` for a program that disguises as a text editor as example.

8. Tools written by the CIA should not export sensitive function names. Rather, they should only export ordinals or benign function names as they leave no hints for analysts.

9. Developers have to make sure not creating crash artifacts (crash dump, core dump, blue screens etc.) through implementing unit tests that force program crashes.

10. Target systems should never become unresponsive due to load produced by CIA tooling in order to prevent detection through users or system administrators.

11. Binary size should be minimized. The guideline mentions an ideal binary size of smaller than 150KB.

12. Developers should implement routines for cleaning up infections and cover traces on infected systems. Also, they have to document how the uninstallation procedure works and which side effects there might be. This ensures that there are no artifacts left for analysis.

13. Binaries should not include timestamps that correlate with the typical US working hours in order to harden attribution.

14. Data that can be used to attribute the binary to the CIA or the US government should be omitted.

15. Similarly, the usage of strings containing CIA terminology (e.g. tool or operation names) or US government related strings are not allowed.

16. Strings from the "dirty words"-list should not be included in the binary.

17. All network communications need to be end-to-end encrypted in order to prevent leaking collected data to analysts.

18. Developers should implement additional security mechanism in addition to secure socket layer (SSL) or transport layer security (TLS).

19. A packet replay protection must be implemented by developers.
20. Tools used in CIA operations must use a Request For Comments (RFC) compliant protocol as a blending layer. This prevents network monitoring tools to flag the traffic because of an unusual or unknown protocol.
21. Network protocols used must be implemented in compliance to the according RFC as otherwise analysis tools could flag the network traffic as malformed.
22. Routines that regularly connect to control servers have to use a time and size jitter, so the traffic cannot be easily correlated.
23. Network connections need to be cleaned up after use in order to not leave stale network connections that can be useful for analysts.
24. Developers need to document the forensic footprint their tools generate. Operators can then estimate the risk associated with the tool usage.
25. Programs should not write or cache data unnecessarily. Developers are instructed to consider third party libraries that might write data to disk also.
26. Collection data should not be written to disk in plain-text.
27. CIA tools should encrypt all data written to disk.
28. In case of file deletion, developers have to implement secure deletion routines that wipe filename, timestamps and other metadata as well as file contents from disk. The minimal secure deletion procedure should include at least overwriting the file with zeroes.
29. The amount of disk operations performed by the software should not be alarming to users and administrators.
30. Files have to be completely encrypted and must not include specific headers or magic values which might be useful for analysts as signatures.
31. File paths or names must be configurable at deploy time. This way operators can choose a fitting name while infiltrating a target.
32. In order to limit the size and number of output files, developers must implement configurable maximum file sizes and/or number of files written.
33. Developers must use the coordinated universal time (UTC) to ensure a common timezone to trigger events when expected.
34. Files used by the CIA must not include time or date formats used specifically in the US to harden attribution.
35. During testing not only free antivirus or security products must be used, instead the software should be tested with the broadest possible product range.

36. Security product or anti virus testing should be done with a live internet connection, if suitable.

Additionally, the guidelines include a document of the Network Operations Division (NOD) about encryption requirements. This is not in scope of this research. The next section shows how to convert most of the above paraphrased guidelines into a first criteria set, however, the guidelines that focus on testing and anti virus detection, e.g. 33 and 34, or are partly covered by developer or operator documentation, such as 12, are not useful as criteria. Analysts usually do not have knowledge about testing or documentation processes of adversaries.

4.2 Initial Criteria Set

The following criteria are direct translations from the CIA guidelines and the main requirement for them is comprehensibility. They must be understandable for analysts with various experience levels and applicable for both, malware analyses and malware reports.

IC1 String and configuration data obfuscation or encryption (Guideline 1)

This criteria applies to the use of obfuscation or encryption in malware samples. If any type of obfuscation or encryption method is used, regardless if the methodology is commonly known (various encoding or encryption mechanism) or self-implemented by the perpetrator, this criteria matches a given malware sample.

The goal of obfuscation or encryption in malware is to make the implemented features less obvious and hinder analysis. There are at least two types of obfuscation. Code or control flow obfuscation changes the control flow of programs through adding unnecessary code, e.g., opaque predicates or by obscuring the control flow through, e.g., jump tables or branch functions [4]. High level methods of control flow obfuscation even implement a custom virtual machine [15]. String obfuscation makes strings less or non-readable through various methods, such as hashing, encoding or encryption [62]. Typically, analysts take a look at strings in malware at the beginning of most analysis processes in order to get an early hint on features implemented in the malware, imported libraries or included file and network artifacts [127]. Malware developers use obfuscation methods to make strings at least unreadable or look like random data.

IC2 Deobfuscation or decryption only when needed (Guideline 2)

Similar to IC1, this criteria applies also to protected strings or configuration data in malware samples, however, this criteria matches samples that only decrypt or deobfuscate strings and configuration data on-purpose and not altogether on program start. In case all strings or configuration data are decrypted

or deobfuscated on program start, analysts can obtain the plain text by setting a breakpoint after the routine which decrypts or deobfuscates the cipher text and dumping the process memory during debugging. This is a common analysis practice [127].

IC3 Clear memory after usage (Guideline 3)

The CIA guidelines require the developers not to rely on the operating system to automatically clear memory of sensitive information after program execution and to remove said data themselves. This criteria applies to this guideline and matches malware samples that overwrite memory after the use of certain sensitive variables. This way the plain text is not recoverable by analysts easily and needs them to carefully step through the binary or implement deobfuscation tooling themselves in order to obtain plain text data.

IC4 Unique encryption keys (Guideline 4)

As mentioned in guideline 4, encryption keys should not be reused and this criteria applies to the guideline, however, naturally more than one sample needs to be analyzed in order to verify this criteria. The reuse of encryption keys is useful to analysts in two ways. First, they can create signatures based on the encryption key and use them to detect samples of that malware family and cluster binaries. Second, if a custom encryption algorithm is used, the exact same tools can be used for analyzing additional samples without further adjustments by the analysts.

IC5 Debug symbols stripped (Guideline 5)

In order to harden analysis and reverse engineering of samples, according to guideline 5, all debug information must be stripped from binaries. This criteria matches samples which not contain any debug symbols and related data. Leaving debug information in samples is a major mistake, because with all function names given, the speed of analysis and reverse engineering is enhanced. Also program database (PDB) paths often include usernames or malware names which can be valuable to analysts and can be used for detection and clustering with Yara rules [92] [93] [142]. Sometimes PDB paths include indications which are useful for attribution [130].

IC6 Debugging output stripped (Guideline 6)

Similar to IC5 this criteria applies to debugging related artifacts. As mentioned in guideline 6, debug output can provide valuable information to analysts and must be removed in the final software. Developers use debugging messages to print data about the current status of operation. This criteria matches if a malware sample does not include debug output, either by regular print/logging functions, via `OutputDebugString` Windows application programming interface (API) or similar methods. Debug output can be used by analysts to get

insights of the malware features and processes, but also to detect and cluster samples via Yara rules as done by Novetta for the malware Winnti [110].

IC7 No suspicious imports or function calls (Guideline 7)

Guideline 7 requires the developers not to explicitly import or call API functions that do not fit the feigned functionality of the malware. The according criteria covers this topic more broadly and matches malware samples that do not import functions which are seen as suspicious by the researcher. Examples for suspicious imports are functions that are used for process hollowing, such as `CreateProcess`, `ZwUnmapViewOfSection`, `NtUnmapViewOfSection`, `VirtualAllocEx`, `WriteProcessMemory`, `SetThreadContext`, `ResumeThread` [100]. However, not every suspicious combination of imports lead to malicious behaviour, so the analyst must decide whether a samples import address table (IAT) is suspicious or not. Additionally, signatures can be written based on the IAT and, therefore, the suspicious imports. A common methodology for clustering and detecting malware based on their imports is using the import hash (imphash)[75]. Malpedia provides insight on commonly used Windows API functions by malware [74].

IC8 No export of sensitive function names (Guideline 8)

Developers withing the AED must use benign names or ordinals for exports in portable executable (PE) files. Exports are used to expose internal functionality to external components. While exports are mostly used in dynamic-link libraries (DLL), regular windows executables can use exports, too, as both are PE files and the use of exports (more specifically the `.edata` section) is defined in the PE format [91]. The check for suspicious export names is a common practice in malware analysis [103], however, sometimes legitimate (and signed) binaries are used to load a malicious library that disguises as a benign one. In this case, a usually legitimate export name or the DLL main entry point [90] is used to implement malicious behavior. The malware families used during the Cloud Hopper operation are examples for that [121]. This criteria matches samples in which analysts haven't found suspicious export names or, if based on malware reports, no suspicious exports are mentioned.

IC9 No artifacts on crash (Guideline 9)

To avoid attention by users and system administrators as well as harden incident response and analysis, program crashes of CIA tools must not generate crash related files, such as crash or core dumps, and must avoid blue screens or other user visible dialogues. This paper subsumes these as crash artifacts and the criteria matches if analysts could not find crash artifacts generated by the malware upon a system crash. The artifacts can be used to gather information about the system state when the crash occurred [89]. Guideline 9 require developers to implement checks for crash artifacts in their unit tests, however,

this usually cannot be known or verified by analysts and is therefore out of scope.

IC10 No performance impact (Guideline 10)

In order to lower the chance of detection and to avoid end user suspicion, the tools written by the CIA must not introduce a performance impact to the target system. This criteria matches if analysts can not identify performance impacts by the malware which could occur, e.g., if the software uses a lot of CPU cycles or accesses a lot of data on the disk, however, no public reference to a malware by an advanced persistent threat could be found. The absence of reports mentioning performance impacts introduced by malware lead to the assumption that this criteria is out of scope for usual malware analysis and reporting.

IC11 Small binary (Guideline 11)

According to guideline 11, binaries must be minimized and ideally be smaller than 150KB. The reasoning behind this is the shorter time needed to download, copy, execute and clean-up the malware. Le et al show in their research that the majority of crime malware samples used in their set is smaller than 1MB [65]. However, with statically linked malware, e.g. written in Go [46], on the rise and adapted by APTs, e.g., APT29 deploying Go based WellMess malware [104] [119], this metric is not useful for measuring sophistication.

IC12 Uninstallation or cleaning routine (Guideline 12)

Every artifact that has been left on a target systems can help analysts. In order to avoid that, the guideline 12 requires developers to implement functions which uninstall or remove the implant and its artifacts from the infected machine. Also, developers need to document the procedure and side effects of implant removal so the operators can assess the risk connected to it. The criteria matches a malware sample which has dedicated functionality to delete itself and cleans-up implanted hooks, dropped files etc., however, if the malware consists only of files on the disk and implements various file system related functionality, this does not count as match. A prominent example of uninstallation capability is the Flame (or Flamer) malware which implements a `SUICIDE` component, however, in June 2012 the operators deployed a new component which then deleted and overwrote files associated with the malware [48].

IC13 No attributable timestamps or -zones (Guideline 13)

Guideline 13 makes sure, timestamps in CIA implants can not be attributed to the US. Accordingly this criteria matches if malware does not include valuable timestamps which means they are (or seem) either crafted or missing. Timestamps often are used by analysts to pinpoint the creation date and to estimate the infection time as well as the working hours of the adversary which itself

can lead to attribution indication, as seen, e.g., in the cloud hopper campaign [130] [120]. In addition to the typical compilation timestamp in PE headers, there are also timestamps in the export, resource and debug directories within the PE header, as described by Barabosch [6].

IC14 No attributable strings (Guidelines 14 & 15)

Developers are required to clean the final binary from strings which suggest the involvement of the CIA, US government or other partners in order to harden attribution. The with guidelines 14 and 15 associated criteria applies to samples without strings containing information about the campaign or adversary such as operation keywords or malware names given by the developers. As this research focuses on malware sophistication measurement and not on attribution, false flags [128] are neglected at this point. False flags can effortlessly be implemented as strings and might lead to jumped conclusions. A recent example for false flag strings is an APT29 campaign where Korean strings were used in the PDB path of a dropped PE DLL file [11].

IC15 No suspicious strings (Guideline 16)

According to the guideline document, the AED manages a "dirty words" list which contains strings that must not be used in CIA implants as they would raise suspicion and may lead to a detailed analysis. The criteria matches on samples which do not include suspicious words such as *exploit*, *implant*, *CVE* (Common Vulnerabilities and Exposures) etc. Analysts can use these strings to get insights about the features of a malware, e.g., the targeted vulnerability or which command & control server was used.

IC16 End-to-end encryption (Guideline 17)

To harden the analysis of network traffic and downloaded payloads, developers are required to always use end-to-end encrypted C&C channels. This prevents analysts peeking into network communication and extracting data or payloads. This criteria matches if the malware uses an end-to-end encrypted communication channel. This can also be TLS, if the server is under the adversary's control and not a compromised third-party server.

IC17 Additional encryption layer (to TLS) (Guideline 18)

Guideline 18 requires the developers to implement an additional encryption layer, in order to harden analysis. Intercepting SSL or TLS encrypted traffic is a common analysis practice [24]. A variety of tools, such as MITMproxy [17], Burpsuite [117] or PolarProxy [106], allow to proxy the traffic while replacing the used certificates. Adding another layer of encryption hardens the analysis of intercepted TLS connections.

IC18 Packet replay protection (Guideline 19)

According to guideline 19, tools by the CIA must implement packet replay protection in order to prevent analysts sending previously captured packets to the C2 server. This is often used to gain knowledge about the implemented C2 server, the adversary's infrastructure or to download additional payloads. Common analysis tools support the recording and sending of network packets, such as MITMproxy [17] or Tcpreplay [140]. IC18 matches on malware samples that implement counter measures for packet replay.

IC19 Uses well-known network protocols (Guideline 20)

In order to blend into the common network traffic in target environments, according to guideline 20, developers need to use IETF RFC compliant network protocols. In case a custom protocol is used, the tool must wrap it into a common protocol, such as HTTPS, before sending the data to the C2 server. Custom network protocols might raise attention in intrusion detection systems (IDS) and can be used for writing network traffic signatures, e.g., the malware *Winnti* uses a custom HTTP Request which does not align with typical, RFC compliant requests. This allows analysts to create a signature based on the *HELO* package sent [47]. This criteria matches malware samples which use a common network protocol or blend a custom protocol into a common one.

IC20 Correct usage of network protocols (Guideline 21)

Tools by the AED need to correctly use network protocols and not break compliance. In case a network protocol is not used correctly, the connections might raise suspicion in IDS and network analysis systems which might flag them as faulty. This criteria matches malware samples which implement network features correctly.

IC21 Variable size and timing for network beacons (Guideline 22)

If CIA tools need to connect to their C2 servers regularly, according to guidelines 21, they have to vary timing and size of the messages. The so called *Jitter* hardens pattern based detection of network connections, however, detecting malware which sends beacons to the C2 server is still possible [68]. Malware that implements periodic C2 connections without some kind of jitter can raise the suspicion of network analysts, because the periodic connections can be seen in packet captures (PCAP) or IDS and some tools automatically detect periodic malware beacons. Research by Huynh et al [44] shows that 30 of 31 researched malware samples use periodic network connections to their C2 servers. Criteria 21 matches malware samples which implement a random variance to packet size and timing.

IC22 Connection clean-up (Guideline 23)

Guideline 22 instructs the developers to always clean-up network connections, so analysts cannot obtain additional information from stale network connections. This criteria matches on malware samples which close network connections properly.

IC23 Low forensic disk footprint (Guideline 24)

As analysts usually can not know if a specific documentation is available on the adversary side (compare guideline 24), this criteria matches malware samples that have a low forensic disk footprint. This includes malware which drops its various stages to disk, however, if implants writes files extensively, such as logs or collected data, this criteria does not match. A high forensic disk footprint rises the chance of detection and might leave information which is useful for analysts during forensic examination.

IC24 No temporary data written (Guideline 25)

According to guideline 25, developers have to make sure, that software does not write temporary data unnecessarily and they have to be aware of third-party libraries used in their project which might does. Similar to IC23, this criteria is related to the files written to disk. The difference is that this criteria matches if no temporary data was written to disk by the malware.

IC25 Encrypts data written to disk (Guidelines 26 & 27)

This merges guidelines 26 and 27 which require developers to not write collection data in plain-text to disk as well as generally encrypt every file written. The file encryption hardens the analysis of forensic artifacts. While the encryption key might be found in memory or in the malware itself, analysis will cost more time and the difficulty is raised.

IC26 Secure erase of files (Guideline 28)

According to guideline 28, developers have to make sure to completely wipe files and the according metadata from disk in case of a file deletion. The minimal requirement is to at least overwrite the space occupied by the original file with zeros once. Without overwriting the file, analysts can use digital forensic methods to recover the file even though it is not visible in the file system anymore [10] [34]. The secure deletion of files raises the difficulty for analysis as there are no usable file artifacts left. This criteria matches on malware samples in which analysts have identified secure deletion routines.

IC27 Disk I/O performance (Guideline 29)

Tools must not perform disk I/O operations that cause unresponsiveness in order not to alert the user or system administrators. Accordingly, this criteria matches on malware samples which does not introduce a performance impact to the infected system.

IC28 Complete encryption of files (Guideline 30)

In order to not leave behind indicators which allow associations between encrypted files and the malware itself, guideline 30 requires the developers to implement encryption routines which do not include magic headers or footers and encrypt files completely. This leaves pure data in files which analysts' cannot connect to the malware directly. Additional information, e.g. from logs, is needed to verify the file is connected to a specific malware. Usually headers and footers are used for identifying file types - in this context before decryption. E.g. ransomware uses headers to identify already encrypted files in order to prevent re-encryption, as shown by Gazet [36]. IC28 matches on samples which encrypt files in a way that does not include magic headers or footers.

IC29 File path customization (Guideline 31)

The AED tradecraft instructs developers to allow file path customization at deployment time. This way operators can choose paths which fit the target environment and allow them to lower the chance to raise suspicion. In case of multiple infections, analysts need to search for the malware or artifacts of it in different locations within the file system which hardens the analysis. As analysts usually cannot know whether paths in the malware were changed at deploy time or not, this criteria matches samples which use different paths in their deployments. Naturally, this requires the analysis of multiple samples of the same malware family.

IC30 File size limit (Guideline 32)

Guideline 32 requires developers to implement a limit for output files by size or amount of files written. This helps to avoid situations were the output files are detected by users or administrators because of the space occupied. Also endpoint detection and response (EDR) software can detect malware which writes a lot of data based on the behaviour patterns. Malware samples which implement a file size or file amount limit match this criteria.

IC31 UTC time zone for comparison (Guideline 33)

To avoid the chance of wrong time triggers, guideline 33 instructs developers to always use the coordinated universal time (UTC). As timestamps mostly are represented as (milli- or nano-) seconds counted from a specific baseline [59], the inclusion of time zones in calculations can be unpractical and error prone. In order to make sure triggers fire on time, the use of the same time zone everywhere can help to prevent errors. IC31 matches malware samples which use UTC as time zone for calculations and comparisons.

IC32 Not attributable time format (Guideline 34)

As some time and date formats are used only in specific countries [12], guideline 34 require tools to include more standardized and less attributable time

and date formats. An alternative to consider is the date format standardized by the International Organization for Standardization (ISO), ISO 8601 [45], which is a global understood format in comparison to the US native format. This criteria matches samples which do not include an attributable time format.

Two criteria related to guidelines 35 and 36 were not included in the above list because the use of anti-virus or other security products in testing processes cannot be known or validated by analysts. The use of anti-virus detection ratios could be a good indicator, however, the anti-virus detection change over time and new signatures are created every day. Analysts would assess these criteria on different bases and that makes achieving a consensus difficult.

4.3 Review of Initial Criteria Set

There are multiple issues with the initial criteria set introduced in section 4.2 which need to be fixed in order to use the criteria set for malware sophistication measurement.

First, some criteria need different levels of implementation in order to represent various degrees of sophistication correctly. E.g., IC1 matches on samples which obfuscate or encrypt strings. But there are various ways of string obfuscation and some might be easier to implement than others. The initial criteria set does not take this into consideration.

Second, there are always information missing in malware reports, however, there are criteria which are not mentioned in the malware report set used for this research (see chapter 5) at all. As defined in chapter 3, in addition to provide a criteria set for sophistication measurement for analysts, the result of this research projects must also be usable with malware reports. Usually analysts do not test and analyze what happens if a malware crashes on the system and therefore crash artifacts are not a common observation mentioned in malware reports (see IC9). So criteria that are not commonly observed must be discarded.

Third, some of the criteria cover multiple features developers implement for OPSEC reasons, e.g., IC1 covers strings and configuration obfuscation, but there are additional components of malware that can be obfuscated, such as later payload stages and the overall control-flow of the malware. So in order to value different kinds of obfuscation, the criteria need to reflect that.

Fourth, there are features and criteria which are not in the AED tradecraft document, but are commonly implemented in malware and represent additional resources put into the development process. Examples for common features are the use of exploits or a very specific targeting, such as programmable logic controllers (PLC) or other industrial control system (ICS) components.

Last, if a sophistication assessment is based solely on malware reports, analysts need to know how to handle missing information. This means, there must be default values for criteria that cannot be checked, based on the most likely scenario,

e.g., analysts would typically mention debug artifacts left in the malware and if this information is not included in reports, debug artifacts were likely removed by the developers.

4.4 Refined Criteria Set

This section revises the initial set of criteria with regard to the shortages identified in the previous section. The criteria IC9, IC10, IC18, IC20, IC22, IC23, IC27 and IC31 are removed from the set, because they are rarely mentioned in malware reports and therefore are less useful as criteria. IC1 is divided into multiple criteria in order to reflect multiple obfuscation methods. Additional criteria were added to represent other commonly implemented features or techniques in malware. In order to add more granularity, some criteria are complemented by sub-criteria. This allows to show different levels if implementation. Also, every criteria has a default value that can be used in case the assessment is based on malware reports and they do not mention a specific criteria explicitly. The following criteria represent the final set for this project:

C1 String obfuscation

This criteria is derived from the initial IC1 and matches on samples which implement obfuscation for strings. Three sub-criteria are available:

C1.1 String obfuscation through malformed strings

C1.2 String obfuscation through functions provided by libraries

C1.3 String obfuscation using self-implemented routines

C1.1 refers to obfuscation implementations where strings are malformed which means that the order of characters is wrong, additional characters were added etc. Deobfuscation methods in this case often reverse the obfuscation through character replacement. C1.2 addresses malware samples that use known obfuscation functions or methods such as various encoding or encryption algorithms. This covers, e.g., Base64 encoding, AES encryption etc. C1.3 covers self-implemented or modified given routines, e.g., custom encryption algorithms or algorithms with modified parameters such as a replaced AES s-boxes [20]. It is important to notice that C1.3 is not meant to be more secure than C1.2, but additional resources were spent in order to implement it.

In case a malware report does not mention string obfuscation explicitly, none of the C1 criteria matches.

C2 Binary/payload packing or encryption

Derived from the initial IC1, this criteria matches samples which are packed or encrypted as well as samples which contain an encrypted payload. This criteria includes two sub-criteria:

C2.1 Binary or payload is packed or encrypted with well-known tools or algorithms

C2.2 Binary or payload is packed or encrypted with custom or modified tools

C2.1 includes commodity packers, self-extracting (SFX) or makeself archives [116], or common encryption algorithms. Commodity packers in this case also include more complex packers which are based on virtual machines such as Themida [112] or VMProtect [15], because they involve no additional development work by the malware developers. C2.2 is comparable to C1.3 and matches on samples which use custom encryption algorithms.

If a malware report does not mention a packed sample or payload, none of the C2 criteria match.

C3 Control-flow obfuscation

Also derived from initial IC1, this criteria is related to obfuscated control-flows or code. Two sub-criteria describe the different levels of control-flow obfuscation:

C3.1 Control-flow obfuscation using well-known tools

C3.2 Control-flow obfuscation using custom tools

Well-known tools in terms of control-flow obfuscation are for example ConfuserEx [51] or Invoke-Obfuscation [8]. C3.2 matches on custom implemented code obfuscation mechanisms, example methodologies can be found in the literature survey by Balakrishnan and Schulze [4].

C4 Configuration obfuscation

C4 refers to the obfuscation or encryption of included configuration data. The available sub-criteria are similar to the ones in C1:

C4.1 Configuration obfuscation through malformed strings

C4.2 Configuration obfuscation through functions provided by libraries

C4.3 Configuration obfuscation using self-implemented routines

Malware configurations are often used by analysts to extract C2 IPs or domains and they are often extracted at scale with sandbox systems that support configuration extraction such as CAPE (Configuration And Payload Extraction) sandbox [113]. This shows the importance of malware configurations and the necessity to include their protection mechanisms in sophistication assessments as a specific criteria. Even though the mechanisms are similar to C1, the methods used for encrypting strings and configuration data must not necessarily overlap.

C5 Purpose-oriented string deobfuscation/decryption

C5 is an additional criteria to C1. Strings can be decrypted the moment they are needed or together at the beginning of the program execution. If strings are decrypted shortly after the program starts, analysts can set a breakpoint in the debugger and dump the memory to receive all decrypted strings at once. This criteria has two sub-criteria:

C5.1 Purpose-oriented string deobfuscation/decryption

C5.2 Purpose-oriented string deobfuscation/decryption and clearing

While C5.1 covers the purpose-bound deobfuscation or decryption process, C5.2 matches on malware samples which remove the plain-text strings after use from memory.

If a malware report does not mention purpose-oriented deobfuscation, but the malware implements general string obfuscation, C5.1 is matched. This reflects a routine which is called every time access to a an obfuscated variable is needed which is a common use case.

C6 Stripped binary

This criteria merges the initial build or debug related criteria IC5 and IC6. Malware samples which do not include

- Function names
- Debug artifacts (PDB paths, `OutputDebugString` statements, ...)
- Verbose printed or logged messages

match this criteria. If a malware report does not mention explicitly that the binary is stripped and none of the above items are either included in the binary or mentioned in the report, the criteria matches.

C7 Obfuscated imports

The initial criteria IC7 is the base for this. A common method in malware is using hashes for importing libraries and functions. This can be done through looping over available libraries and comparing the values [145] [7], however, this criteria applies also to other methods used in order to evade import-based detection. The following sub-criteria are available:

C7.1 Obfuscated imports through obfuscation methods used in C1

C7.2 Obfuscated imports through well-known API hashing or import evasion methods

C7.3 Obfuscated imports through unknown or custom API hashing or import evasion methods

As the obfuscation can also base on regular string obfuscation methods, C7.1 reflects the use of them in this scenario. C7.2 matches on samples that use well-known hashing algorithms for import obfuscation or other well-known methods for evading direct imports. Similarly, C7.3 matches on unknown or custom hashing algorithms as well as previously unknown methods for covertly importing functions from libraries. An examples for covert importing methods is Stuxnet. Section 5.1 describes how Stuxnet hooks Windows APIs to evade import- or behavior-based detection.

In case a malware report does not mention obfuscated imports, none of the above sub-criteria matches.

C8 No sensitive exports

Analog to IC8 this criteria matches if a samples does not export suspicious functions or a malware report does not explicitly mentions suspicious exports. C8 has no sub-criteria.

C9 Uninstallation/cleaning functionality

As former IC12, C9 refers to dedicated uninstallation or cleaning routines implemented in malware samples. As mentioned before, generic file deletion functionality is not sufficient. By default, C9 is not matched.

C10 No attributable strings

This criteria is exactly like the initial IC14 and does not have any sub-criteria. Malware samples which do not include strings that can be used to attribute the malware at various levels (campaign, threat actor, country or organization, cf. [130]) match this criteria.

C11 No suspicious strings

Like initial IC15 this criteria covers suspicious terms used in malware. Samples that do not include terms that raise the suspicion of analysts match this criteria. Providing an exhaustive list of suspicious words is not possible. In general, suspicious words can be related to CVE numbers, branded vulnerabilities [118], general hacker terminology such as *hack*, *exploit*, *implant* etc., but overall the suspicion regarding specific terminology is subjective to the analyst. This criteria has no sub-criteria and in case it is not mentioned in a malware report, the criteria matches.

C12 Use of well-known network protocols

Exactly like the initial criteria IC19, this matches malware samples which use well-known network protocols instead of niche or custom ones. The well-known protocols can be used as blending layer for other network protocols. This criteria matches also malware reports which do not mention it explicitly as well-known protocols are most likely the case and analysts would mention

custom protocols because they are conspicuous as they can be flagged by networking tools such as Wireshark [150].

C13 Additional encryption layer

This criteria is derived from the initial criteria IC17. There are two sub-criteria to measure different levels.

C13.1 Additional symmetric encryption

C13.2 Additional asymmetric/hybrid encryption

The first sub-criteria matches if a sample uses an additional encryption layer within the C2 communication and needs to share the key via the same connection. C13.2 matches malware samples which use an asymmetric (e.g. RSA) or hybrid (e.g. PGP) encryption approach.

In case a malware report does not mention an additional encryption layer, none of the above criteria matches.

C14 Unique encryption keys

Analog to the initial criteria IC4, malware samples which use unique encryption keys per deployment match this criteria. This applies to all occasions where an encryption key is used, not just string or payload decryption processes, but also C2 communication.

If a malware report does not mention the use of unique key material, this criteria does not match. Usually encryption keys are one of the easiest ways to cluster malware samples if a key was reused. The assumption that analysts have examined samples regarding key reuse seems reasonable.

C15 No temporary data written

This criteria merges the initial criteria IC23 and IC24, and matches malware samples which do not write temporary files, such as collected data, to disk. Files written by malware leave valuable artifacts to analysts and although they can be overwritten the easiest prevention is not writing them. In case this criteria is not mentioned in a malware report, the according malware sample does not match this criteria.

C16 Encryption of written files

Initial criteria IC25 and IC28 are merged in this criteria as two different levels:

C16.1 Encryption of written files

C16.2 Encryption of written files without magic header or footer

C16.2 is the more strict criteria matching on samples which implement encryption without adding a magic header or footer to the encrypted file. Samples matching C16.1 would allow finding files encrypted by the malware through

searching for the magic header or footer. If no encryption of written files is mentioned in malware reports, none of the above criteria match.

C17 Secure deletion

As exact replacement of initial criteria IC26, this criteria matches on malware samples which implement routines for secure deleting or overwriting files. By default this criteria is not matched.

C18 Use of exploits

An additional criteria identified in section 4.3 is the use of exploits. This criteria implements 4 sub-criteria.

C18.1 Use of exploits for publicly known vulnerabilities

C18.2 Use of exploits for recently published vulnerabilities

C18.3 Use of exploit for an unknown vulnerability

C18.4 Use of exploits for multiple unknown vulnerabilities

C18.1 refers to the use of exploits for vulnerabilities which are publicly known for quite some time. An example for this are lure documents generated by the *8.t* or *Royal Road* RTF exploit builder. These include exploits for vulnerabilities in the Microsoft Equation Editor which were disclosed in 2017 and 2018 [60]. Although the vulnerabilities were disclosed quite some time ago, they are still in use as recently reported by Check Point [13]. C18.2 matches on samples which exploit vulnerabilities which are also called 1-days or n-days. They were publicly disclosed recently. C18.3 and C18.4 refer to the use of exploits for 0-day vulnerabilities, i.e. unknown vulnerabilities which are not disclosed publicly, either one or multiple of them.

In general exploits age which means that time as a variable need to be taken into consideration. Because of that, the correct criteria must be matched according to the assessed or known time of malware creation and not based on the time of analysis.

In case a malware report does not mention exploits, none of the above criteria match.

C19 Signed malware

This criteria applies to all malware binaries which have a signature and has two sub-criteria.

C19.1 Malware has an expired or invalid signature

C19.2 Malware has a valid signature

The first one matches malware samples which are signed, but the certificate is expired or invalid. The latter applies to malware samples which have a correct

and valid signature. As certificates always involve time, similar to C18, the assessed or known creation timestamp needs to be used for assessment.

If a malware report does not mention a signature explicitly, none of the above matches.

C20 Modular architecture

A modular architecture allows extending malware during operations. This way, operators can deploy an additional target specific payload without the need to recompile the complete malware or to create a new one. There are different scenarios possible, but every time a malware loads additional payloads which, both, rely on routines in the main payload for execution and extend the functionality, this criteria matches. Important to notice is that this criteria does not apply for dropper which load the next stage from a C2 server. While developers can change the payload, a dropper is not a modular malware approach itself. Modular malware can be extended during operations by the operators themselves without the help of an additional developer.

This criteria does not match by default.

C21 Persistence

An important criteria missing in the initial criteria set is the specific persistence method used in malware. This criteria implements two sub-criteria.

C21.1 Well-known persistence method

C21.2 Exceptional persistence method

C21.1 applies to all malware samples which use regular persistence methods. Hunkeler published examples of frequently used persistence methods [43]. The paragraph of techniques which are difficult to detect contains examples of persistence methods that match C21.2, e.g., Duqu 2.0, which does not implement persistence on infected clients, because a highly available server in the same network reinfects the clients, if needed [56]. While rootkits often use installed services to start, they are considered exceptional as they implement additional measures to hide the malware although services as persistence methods would match C21.1.

In case malware reports do not mention persistence, none of the above sub-criteria matches.

C22 Specific targeting

Targeting is a big difference between measuring malware severity and sophistication. Usually the higher the amount of infections, the higher the score from threat or severity rating mechanisms assessed to malware samples. In contrast to the presented threat rating mechanism in chapter 2, sophisticated malware

often targets niche products on the same platform or even products on other processor architectures. This criteria uses two sub-criteria.

C22.1 Targeting of uncommon software on the same platform

C22.2 Targeting of uncommon software and architectures

The first sub-criteria matches on malware which targets niche or sector-specific software. Forcepoint published an article in 2018 which describes malware targeting AutoCAD projects [107]. This matches C22.1. The latter sub-criteria cover an even more specific targeting, where the malware ultimately infects a complete different architecture. Stuxnet [27] or Regin [57] [133] are examples for this category of malware.

If a malware report does not explicitly state a specific targeting, none of the above criteria matches.

C23 Uncommon network features

Some malware implement uncommon network features, e.g., in order to provide other infected clients a proxy connection to the C2 server or to intercept packets to or from specific services. Every time a malware implements networking related features which seem to be uncommon, this criteria matches. Examples that match this criteria are the Winnti malware which implements a kernel driver listening to a specific magic header in network packets [110] or the X-Agent malware which provides an alternative routing for C2 channel messages and enables isolated systems to send messages through message queues on USB flash drives [77].

Per default, this criteria does not match and must be mentioned in malware reports explicitly.

C24 Parameter customization

Similar to criteria C20, modular architecture, this criteria is also related to operational adjustments in the malware which do not need developers to change actual code. Mostly this is done through specific configuration files or blocks embedded in the malware which contain, e.g., the C2 server's address or specific file paths used on the target system. This criteria matches malware families which do not use hardcoded parameters, but instead use configurations that can be changed from deployment to deployment. Sometimes researching customization requires analysis of more than one malware sample of the same malware family.

In case a malware report does not mention customization of malware samples this criteria does not match.

C25 Multiple C2 channels

This criteria matches on samples which have one or multiple C2 channels according to the three sub-criteria:

C25.1 One C2 channel

C25.2 Two C2 channels

C25.3 Three or more C2 channels

Per default, none of the above criteria matches, reports need to mention the use of one or multiple C2 channels explicitly.

C26 Limited execution

In order to narrow down the malware propagation, sometimes the execution of malware is limited to specific environments through various measures. This criteria includes two sub-criteria:

C26.1 Roughly limited execution

C26.2 Strongly limited execution

C26.1 matches samples which include measures to allow execution on systems based on, e.g., keyboard layout or display language. The used limiting parameters still are quite broad. Contrary, C26.2 matches malware samples that are bound to specific domains or hosts. Exemplary, this criteria can be used in combination with C2, where the malware drops a payload which is symmetrically encrypted using the target's hostname as encryption key. The sample would match both, C2 and C26 in that case.

If a malware report does not mention specific execution limits, C26 does not match.

4.5 Score Calculation

After the criteria have been applied to a specific malware family, the score (S) can be calculated through the sum of criteria values that matched a sample divided by the sum of all 26 criteria values. The value of a specific criteria is dependent on the matched sub-criteria and can be described as function $v(C_{i.j})$, shown in equation 4.1. Equations 4.2 and 4.3 show examples for matched criteria $C_{1.2}$ and C_6.

$$v(C_{i.j}) = \begin{cases} j, & \text{if criteria has a matched sub-criteria} \\ 1, & \text{if criteria matched and has no sub-criteria} \\ 0, & \text{else} \end{cases} \tag{4.1}$$

$$v(C_{1.2}) = 2 \tag{4.2}$$

$$v(C_6) = 1 \tag{4.3}$$

For the mentioned score calculation the sum of all criteria values is needed. This means applying $v(C_{i.j})$ for every 26 criteria and sum the result as shown in 4.4. In this case, j is always the maximum available sub-criteria.

$$\sum_{i=1}^{26} v(C_{i.j}) = v(C_{1.3}) + v(C_{2.2}) + ... + v(C_{24}) + v(C_{25.3}) + v(C_{26.2}) = 46 \tag{4.4}$$

Equation 4.5 shows the score calculation using the sum of all criteria from above.

$$S = \frac{\sum_{i=1}^{26} v(C_i)}{46} \tag{4.5}$$

Developers implement some features more often than other features therefore a weight is applied to reduce the significance of features that are generally rather implemented and to increase the significance of criteria that are rarely implemented or implemented in malware families that are reported to be sophisticated. The used weights, however, are not final and should be subject to further research. The weighted score is calculated through applying the weight on every summed criteria value and, similar to the unweighted score, dividing it through weighted value of all criteria. Table 4.1 shows the weights used in this research project.

Similar to equation 4.4, the total weighted value can be summed through applying $v(C_{i.j})$ and multiplying it with the according weight, as shown in equation 4.6.

$$\sum_{i=1}^{26} v(C_{i.j}) * w_i = v(C_{1.3}) * 1.0 + ... + v(C_{23}) * 2.0 + ... + v(C_{26.2}) * 1.0 = 56.5 \tag{4.6}$$

Equation 4.7 shows the calculation of the weighted score with the total weighted value of 56.5.

$$S_w = \frac{\sum_{i=1}^{26} v(C_i) * w_i}{56.5} \tag{4.7}$$

Sometimes, specific features are not implemented consistently or malware reports are not precise enough to assess if a criteria would match a specific sample completely. In these cases, half-matching criteria is possible through subtracting 0.5 from the value, e.g., if a malware securely wipes files most of the times, but in a specific scenario, the file is not wiped, but just deleted instead. This would qualify to set the value of C17 to 0.5. Similarly this can be applied to sub-criteria. If a malware reports assesses the use of a zero-day but is not confident in doing so, the value for C18 is 2.5. $v(C_{i.j})$ can be extended to support this, as shown in equation 4.8.

Criteria	Description	Weight
C1	String obfuscation	1.0
C2	Binary/payload packaging or encryption	1.0
C3	Control-flow obfuscation	1.0
C4	Configuration obfuscation	1.0
C5	Purpose-oriented string deobfuscation/decryption	1.0
C6	Stripped binary	1.0
C7	Obfuscated imports	1.0
C8	No sensitive exports	0.5
C9	Uninstallation/cleaning functionality	1.0
C10	No attributable strings	1.0
C11	No suspicious strings	0.5
C12	Use of well-known network protocols	0.5
C13	Additional encryption layer	1.0
C14	Unique encryption keys	2.0
C15	No temporary data written	1.0
C16	Encryption of written files	2.0
C17	Secure deletion	2.0
C18	Use of exploits	2.0
C19	Signed malware	2.0
C20	Modular architecture	1.0
C21	Persistence	0.5
C22	Specific targeting	2.0
C23	Uncommon network features	2.0
C24	Parameter customization	1.0
C25	Multiple C2 channels	1.0
C26	Limited execution	1.0
	Sum	**31**

TABLE 4.1: Criteria weights used in this research project.

$$v(C_{i.j}) = \begin{cases} j, & \text{if criteria has a matched sub-criteria} \\ j - 0.5, & \text{if criteria has a partly matched sub-criteria} \\ 1, & \text{if criteria matched and has no sub-criteria} \\ 0.5, & \text{if criteria matched partly and has no sub-criteria} \\ 0, & \text{else} \end{cases} \tag{4.8}$$

Based on the given weighted score or the weighted sum, the malware can be assigned to one of the five available categories of sophistication: very low, low, medium, high, very high. Baseline for scaling the category ranges, is the weighted sum of all first sub-criteria (AFS) which is the same as the sum of all weights (see table 4.1) as shown in equation 4.9. Table 4.2 show the categories assigned based on a 20% interval of the AFS.

$$\text{AFS} = \sum_{i=1}^{26} w_i = 31 \tag{4.9}$$

% of AFS	**Weighted sum**	**Weighted score**	**Category**
$\geq 80\%$	≥ 24.8	≥ 0.44	Very High
$\geq 60\%$	≥ 18.6	≥ 0.33	High
$\geq 40\%$	≥ 12.4	≥ 0.22	Medium
$\geq 20\%$	≥ 6.2	≥ 0.11	Low
$< 20\%$	< 6.2	< 0.11	Very Low

TABLE 4.2: Sophistication categories based on AFS percentage.

Chapter 5

Description of Results

This chapter includes the application of the methodology introduced in chapter 4. Based on publicly available analysis reports, 25 APT malware families are matched against the criteria set. Although there is no standardized measurement method for sophistication assessment available, based on the statements in analysis reports, the set of malware families were chosen to be as heterogeneous as possible with regard to the level of sophistication. The set consist of the following malware families (notation adopted from sources):

- BabyShark (Visual Basic Script based)
- BadNews
- DoubleFantasy
- Duqu 2.0
- EquationDrug
- EvilGnome (Executable and Linkable Format (ELF) binary)
- FatDuke
- Flame
- KASPERAGENT
- Kazuar
- LightNeuron
- MATA
- MICROPSIA
- MoleNet
- OceanDrive
- PlugX
- POWERSTATS (Powershell based)

- Pteranodon (mostly Visual Basic and batch script based)
- Regin
- SharpStage
- Spyder
- Stuxnet
- Winnti
- X-Agent
- XDSpy

The following sections describe how the presented approach is applied to the malware families. Information about the malware families were solely taken from reports mentioned as source, no own malware analysis was done[1]. A survey was conducted to collect a sample of sophistication assessments from the threat intelligence and malware analysis community. The last section of this chapter introduces the aforementioned survey. A checklist which can be used for applying the introduced methodology can be found in appendix A.

5.1 Detailed Examination of Stuxnet

Exemplary, the introduced approach will be applied on the malware Stuxnet in detail. Stuxnet is the highest scoring malware in the set of malware families used in this research project and therefore is a good example for describing how to match the criteria. Base for the assessment is the Stuxnet dossier written by Falliere, Murchu and Chien [27].

The malware Stuxnet targets systems which are used to program PLCs in order to manipulate industrial control systems. To infect targets, various vulnerabilities [79], some of them zero-days [81] [82], and infection vectors are used, including a peer-to-peer updating mechanism. In addition to bypassing security products, Stuxnet installs a rootkit to hide its binaries from the system. Similarly, it also hides the modified code on PLCs which can be seen as PLC rootkit. The main payload of Stuxnet is a DLL file which is included in a loader executable. After extracting and mapping the DLL to memory, the loader calls an exported function, identified by ordinal. The main payload DLL includes various resources, such as further payloads, template files which can be used to create exploits or configuration files. In order to prevent behavior-based detection or blocking of the DLL loading mechanism, Stuxnet hooks various functions in `Ntdll.dll` and calls `LoadLibrary` with fake file names which do not exist in the file system. The crafted file names are then caught by the

[1] In chapter 6, Detect It Easy was used on MoleNet and SharpStage in order to verify the obfuscation method, as obfuscation was mentioned, but details were missing in the malware report.

hooked functions and mapped to other locations in memory. Addresses of functions are retrieved via `GetProcAddress`. As a security measure, various anti-virus products are enumerated and if no bypass is available, injection processes are canceled. Otherwise the target process varies based on the security product installed. Stuxnet uses obfuscation or encryption in some variants for strings (URLs) and in general for some modules and configuration data as well as C2 traffic, however, encrypted modules are not further described by the source. The following criteria can be applied to the malware report:

C1 String obfuscation
URLs were moved to the installer component and "crudely" obfuscated at one point. While the obfuscation method is not clearly stated in the report, it does not apply to all versions of Stuxnet. Based on this limitations, half-matching C1.1 or C1.2 is possible. "Crude" can relate to malware in general or to Stuxnet specifically. For this assessment a known method based on given functions (C1.2) is assumed.

C2 Binary/payload packing or encryption
Various functionality of Stuxnet involves dropping encrypted malware binaries. The type of encryption is not further described, so a common methodology is assumed here which matches C2.1.

C3 Control-flow obfuscation
Not mentioned.

C4 Configuration obfuscation
Mentioned multiple times, Stuxnet uses encryption configuration blocks. While the type of encryption is not further mentioned, a common methodology is assumed in this case. C4.2 matches this characteristics.

C5 Purpose-oriented string deobfuscation
Related to C1, this matches C5.1 partly, as this applies only to the specific versions including obfuscated URLs.

C6 Stripped binary
Not specifically mentioned. Default value applies.

C7 Obfuscated imports
Stuxnet hooks functions typically used by `LoadLibrary` and calls the function with fake and non-existent filenames. The returned handle points to an address in memory which was set by the hooked functions based on the fake filenames used. This not a common method for importing libraries and functions, therefore this matches C7.3.

C8 No sensitive exports
Only ordinals are used by Stuxnet. This matches the original requirement by the CIA guidelines which are represented by C8.

C9 Uninstallation/cleaning functionality
Stuxnet provides uninstallation functionality via exported function in the DLL.

C10 No attributable strings
Not mentioned in the report.

C11 No suspicious strings
Not mentioned in the source document.

C12 Use of well-known protocols
Stuxnet's command and control traffic uses HTTP, so this criteria matches.

C13 Additional encryption layer
For C&C communication, data is encrypted with 31 Byte XOR keys which matches C13.1.

C14 Unique encryption keys
No match as Stuxnet uses static keys in all samples.

C15 No temporary data written
Stuxnet writes lots of files to the temporary directory, i.e. this criteria does not match.

C16 Encryption of written files
Stuxnet encrypts files written to disk. The malware reports suggest that there is no header or footer available which can be used to identify the encrypted file's association to Stuxnet. This matches C16.2.

C17 Secure deletion
Not mentioned, therefore not matched.

C18 Use of exploits
Multiple exploits are used by Stuxnet, two are mentioned to be zero-days. Therefore the highest sub-criteria, C18.4, matches.

C19 Signed malware
This changed over the threat lifetime, but some samples were validly signed, however, as this does not apply to all samples, criteria C19.2 is partially matched.

C20 Modular architecture
Stuxnet does not have a modular architecture.

C21 Persistence
While the persistence is achieved via installation of one Stuxnet component as a service, it implements rootkit functionality and hides other Stuxnet related files. Rootkits match C21.2 despite relying on an installed service.

C22 Specific targeting
As a malware which targets PLCs in order to manipulate other ICS components, it matches C22.2.

C23 Uncommon network features
The malware report mentions infections via peer-to-peer RPC functionality in order to reach the actual target systems, as they likely do not have internet access. This counts as a match for this criteria.

C24 Parameter customization
As mentioned already in C4, Stuxnet uses configuration blocks which contain customizable configuration data.

C25 Multiple C2 channels
Stuxnet only uses one actual C2 channel which is HTTP. C25.1 matches this behavior.

C26 Limited execution
The malware uses a rough limitation for stopping the malware's propagation. After a given date, Stuxnet does not run anymore. This matches C26.1.

Table 5.1 shows all matching criteria and the score of 0.68. According to the introduced scheme, Stuxnet is a very high sophisticated malware.

Criteria	Value	Criteria	Value
C1	1.5	C14	0
C2	1	C15	0
C3	0	C16	2
C4	2	C17	0
C5	0.5	C18	4
C6	1	C19	1.5
C7	3	C20	0
C8	1	C21	2
C9	1	C22	2
C10	1	C23	1
C11	1	C24	1
C12	1	C25	1
C13	1	C26	1
Sum			30.5
Weighted sum			38.5
Weighted score			0.68
Category			Very High

TABLE 5.1: Matching criteria for the Stuxnet malware.

5.2 BabyShark

The malware family BabyShark is based on VisualBasic script. Malicious document templates download a Microsoft HTML Application (HTA) file from the C2 server. The included VBS requests additional data from the C2 server, decodes the answer and executes code. First, registry settings are altered which enable the execution of macros. Second, the malware executes commands to obtain system information. Third, the base64 encoded file which contains the results is uploaded to the C2 server. Last, the malware registers a Powershell command as a command processor and creates two scheduled tasks which run the `taskkill` command via `cmd.exe`. As the scheduled task start the Windows command line interpreter, every time the scheduled task is started, it triggers the previously registered command processor [115]. Table 5.2 shows the matched criteria. With a score of 0.08, BabyShark is a very low sophisticated malware.

5.3 BadNews

The BadNews called malware is dropped by documents which use one of two publicly known vulnerabilities from 2015 and 2017 [108] [109]. Communicating and downloading additional information over HTTP, the malware is able to execute various commands provided by the C2 server, however, the C2 server's IP must be downloaded and decrypted from a dead drop resolver [102] before. In order to store information, the malware creates four different files in the `%TEMP%` directory. Traffic from the C2 server is encrypted with AES-128, however, according to [67], uploads to the C2 server are plain-text. Table 5.3 shows the matching criteria for BadNews. The malware has a weighted score of 0.13 and is assigned to the category low sophisticated.

5.4 DoubleFantasy

In the analysis scenario Kaspersky researchers report about, the implant Double Fantasy by the Equation Group was distributed in 2009 using CDROMs with photo slideshows [52]. Leveraging three vulnerabilities [80] [83] [84] for privilege escalation - two of them were zero days in 2009 - the malware creates privilege persistence for the local user through adding them to the administrators group. The first payload of the implant is a DLL file which extracts a XOR encrypted configuration from one PE resource. After the malware checked the availability of some anti-virus products, the main payload DLL is dropped to the `%system%` directory. Using a custom loading mechanism instead of `LoadLibrary`, the implant loads various libraries and persists itself through Component Object Model (COM) object hijacking during the installation process. In order to not break the original functionality of the hijacked

COM object, the persisted DLLs implement necessary functions and exports to forward calls to the correct COM object. Kaspersky concludes that DoubleFantasy is a validator-type trojan which is used to deploy further malware after reconnaissance done on the target system. However, the trojan itself is capable of running commands from the C2 server. Communication is additionally RC6 encrypted and the trojan is configured to delete itself after a specified number of days. The first payload is configured to remove itself after a reboot. Table 5.4 shows the matching criteria. With a score of 0.45 the implant is very high sophisticated.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	1	C15	0
C3	0	C16	0
C4	0	C17	0
C5	0	C18	0
C6	0	C19	0
C7	0	C20	0
C8	0	C21	1
C9	0	C22	0
C10	1	C23	0
C11	1	C24	0
C12	1	C25	1
C13	0	C26	0
Sum			6
Weighted sum			4.5
Weighted score			0.08
Category			Very Low

TABLE 5.2: Matching criteria for the BabyShark malware.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	0	C15	0
C3	0	C16	0
C4	0	C17	0
C5	0	C18	1
C6	1	C19	0
C7	0	C20	0
C8	1	C21	1
C9	0	C22	0
C10	1	C23	0
C11	1	C24	0
C12	1	C25	1
C13	0.5	C26	0
Sum			8.5
Weighted sum			7.5
Weighted score			0.13
Category			Low

TABLE 5.3: Matching criteria for the BadNews malware.

Criteria	Value	Criteria	Value
C1	0	C14	1
C2	1	C15	1
C3	0	C16	0
C4	2	C17	0
C5	0	C18	4
C6	1	C19	0
C7	3	C20	0
C8	1	C21	2
C9	1	C22	0
C10	1	C23	0
C11	1	C24	1
C12	1	C25	1
C13	1	C26	0
Sum			23
Weighted sum			25.5
Weighted score			0.45
Category			Very High

TABLE 5.4: Matching criteria for the DoubleFantasy malware.

5.5 Duqu 2.0

While Kaspersky is not sure about the initial infection vector that has being used for Duqu 2.0, they assess it is possible that a zero day vulnerability was exploited to deploy the malware [85]. Similarly, the implant uses another zero day at that time to propagate across the network as a Microsoft Windows Installer Package (MSI) [86]. In order to extract the encrypted payload, the operator need to pass the key as a parameter to `msiexec.exe`. The adversaries used different keys, algorithms and filenames for each operation. Kaspersky lists Camellia, AES, Extended Tiny Encryption Algorithm (XTEA), RC4 and XOR for encryption as well as LZJB, LZF, FastLZ and LZO as compression algorithms. The backdoor is deployed to computers if needed and only resides in memory. In order to achieve persistence, the adversary infects a server which has a high availability and recompromises clients from the server. The malware family consists of multiple layered and encrypted payloads, and uses a modular approach to run many different plugins. On 64-bit platforms another zero-day vulnerability is exploited to circumvent the need of a signed kernel driver and to escalate privileges to kernel level [87]. Kaspersky describes the actor as one of the highest sophisticated groups [56]. With a score of 0.57 (see table 5.5) the malware family is assessed as very high sophisticated.

Criteria	Value	Criteria	Value
C1	0	C14	1
C2	1	C15	0
C3	0	C16	0.5
C4	2	C17	1
C5	0	C18	4
C6	1	C19	0
C7	3	C20	1
C8	1	C21	2
C9	0.5	C22	0
C10	0	C23	1
C11	1	C24	1
C12	1	C25	1
C13	1	C26	2
Sum			27
Weighted sum			32
Weighted score			0.57
Category			Very High

TABLE 5.5: Matching criteria for the Duqu 2.0 malware.

5.6 EquationDrug

EquationDrug is a malware platform used by the Equation Group. Delivered by a dropper, such as DoubleFantasy, the malware uses a modular approach to run different kinds of plugins. Kaspersky calls the implant a mini operating system with kernel- and user-mode. However, compared to DoubleFantasy, EquationDrug has a lower OPSEC level. Some components includes strings which are both, suspicious and attributable. Also, one component writes a temporary log file either to disk or to the registry. In case it is written to disk, the malware violates C15, however for other modules encrypted virtual file systems (VFS) and in-memory storage are used for storing files. EquationDrug also implements a network driver which listens for a specific magic header and is capable of running shellcode directly from network packets, remove the driver and associated files, generate new RC5 session keys and change the process injection target. The malware persists through the kernel driver which is also a rootkit. There are modules for flashing disk firmware available, but not further described in the source [54]. Table 5.6 shows the matching criteria. Due to the mentioned lower OPSEC in comparison to DoubleFantasy and missing exploits as the malware is delivered at the end of the infection chain, the score is 0.40 and the malware is assigned to the high sophisticated category.

Criteria	Value	Criteria	Value
C1	2	C14	1
C2	1	C15	0.5
C3	0	C16	2
C4	2	C17	0
C5	1	C18	0
C6	1	C19	0
C7	0	C20	1
C8	1	C21	2
C9	1	C22	0
C10	0	C23	1
C11	0	C24	1
C12	1	C25	1
C13	1	C26	0
Sum			20.5
Weighted sum			22.5
Weighted score			0.40
Category			High

TABLE 5.6: Matching criteria for the EquationDrug malware.

5.7 EvilGnome

The only Linux malware in this research, EvilGnome, is delivered using a makeself self-extracting archive [116] and contains all metadata, e.g., the exact commands and paths used by the adversary. For installation, an included setup script copies the not-stripped payload to the user directory and establishes persistence via cron job. Upon start, the malware loads the not encrypted binary configuration file and starts a thread for capturing screenshots, scanning the file system and collecting files, and connecting the C2 server. Output generated by the modules is RC5 encrypted either stored in the user directory or directly send to the C2 server. Table 5.7 shows the matched criteria and the score of 0.14 which assigns the malware to the low sophistication category.

5.8 FatDuke

FatDuke is one of the main payloads used by APT29 also known as *the Dukes* [30]. The malware persists using a run key in the registry [94] and contains a base64 encoded configuration as PE resource which includes the C2 server address, an AES encryption key used for C2 communication, the names of cookies to exfiltrate and other values. According to ESET, the binaries are highly obfuscated using string stacking, opaque predicates, junk code and junk strings. The analysts assess that the adversary behind the malware has the resources to create their own obfuscator instead of relying on publicly available and commercial tools. Two C2 channels are available which include the capability to connect via named pipes using other infected systems. The malware is able to delete itself in a secure manner. Table 5.8

shows the matched criteria and the score. Based on the methodology introduced in this paper, the malware is assessed as high sophisticated.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	1	C15	0
C3	0	C16	1
C4	0	C17	0
C5	0	C18	0
C6	0	C19	0
C7	0	C20	0
C8	1	C21	1
C9	0.5	C22	0
C10	0	C23	0
C11	0	C24	1
C12	1	C25	1
C13	1	C26	0
Sum			8.5
Weighted sum			8
Weighted score			0.14
Category			Low

TABLE 5.7: Matching criteria for the EvilGnome malware.

Criteria	Value	Criteria	Value
C1	1	C14	1
C2	0	C15	1
C3	2	C16	0
C4	0	C17	1
C5	1	C18	0
C6	1	C19	0
C7	1	C20	0
C8	1	C21	1
C9	1	C22	0
C10	1	C23	1
C11	1	C24	1
C12	1	C25	2
C13	1	C26	0
Sum			20
Weighted sum			21
Weighted score			0.37
Category			High

TABLE 5.8: Matching criteria for the FatDuke malware.

5.9 Flame

The Flame [37] or sKyWIper [63] named malware is a complex toolkit which implements a lot of functionality. Completely deployed the malware is nearly 20MB in size and consists of up to 20 plugins. It has the capabilities to spread across air-gapped environments and to deliver malicious Windows updates through acting as man in the middle (MITM). The adversary behind Flame was able to hijack the Windows updating procedure and provide signed malware to computers that try to download Windows updates. Using a valid certificate created through a MD5 hash collision showcases the resources of the perpetrator [124] [129]. The Laboratory of Cryptography and System Security (CrySyS Lab) mentions multiple custom encryption algorithms used in the malware. One particular component of the malware is responsible for removing all data related to the intrusion. In 2015, Symantec reports about a new module deployed to systems infected by Flame which wiped the malware itself [48]. The malware persists as Local Security Authority (LSA) authentication package. Table 5.9 shows the matched criteria as well as the weighted score. The malware is assessed as very high sophisticated.

Criteria	Value	Criteria	Value
C1	0	C14	1
C2	2	C15	0
C3	0	C16	2
C4	3	C17	0
C5	0	C18	2.5
C6	0	C19	1
C7	0	C20	1
C8	1	C21	2
C9	1	C22	0
C10	0	C23	1
C11	1	C24	1
C12	1	C25	1
C13	1	C26	0
Sum			22.5
Weighted sum			27.5
Weighted score			0.49
Category			Very High

TABLE 5.9: Matching criteria for the Flame malware.

5.10 KASPERAGENT

KASPERAGENT is compressed using the legitimate MPRESS packer [76] and disguises itself as a media player. The malware uses malformed strings for obfuscation and connects to the hard-coded C2 server domain via HTTP. Most of the analyzed samples are pure droppers for further payloads, however, some of them implement additional functionality such as exfiltrating saved passwords from browsers, taking screenshots, logging keys, executing remote commands and exfiltrate files from the target as encrypted ZIP archive. The researchers were able to find PDB strings in the malware [5]. Based on the weighted score of 0.11 shown in table 5.10, the malware is assessed as low sophisticated.

5.11 Kazuar

Assessed to be used by the threat group Turla, Kazuar is an implant protected by code-obfuscation through the publicly available ConfuserEx tool. The malware logs its activities verbosely to AES encrypted files on disk. Persistence is achieved through installing a new service in an environment with no user interface or in environments with graphical user interface (GUI) through a link in the startup directory or various registry keys. Kazuar implements the capability to connect to C2 servers via HTTP(S) or FTP(S), however, the analysts have only observed the use of HTTP in the wild. The malware expects the C2 server to respond with XML-formatted data which contain tasks for the implant to process. One of the commands available is

called *suicide* and while it suggests an uninstallation of the implant, it is not implemented. Kazuar provides an uncommon network feature which exposes the malware's functionality via an HTTP API to other systems in the network [66]. With the amount of matched criteria in table 5.11 and the score of 0.27 Kazuar is medium sophisticated.

Criteria	Value	Criteria	Value
C1	1	C14	0
C2	1	C15	0
C3	0	C16	0
C4	0	C17	0
C5	1	C18	0
C6	0	C19	0
C7	0	C20	0
C8	1	C21	1
C9	0	C22	0
C10	0	C23	0
C11	1	C24	0
C12	1	C25	1
C13	0.5	C26	0
Sum			8.5
Weighted sum			6.5
Weighted score			0.12
Category			Low

TABLE 5.10: Matching criteria for the KASPERAGENT malware.

Criteria	Value	Criteria	Value
C1	0	C14	1
C2	1	C15	0
C3	0	C16	1
C4	0	C17	1
C5	0	C18	0
C6	1	C19	0
C7	0	C20	1
C8	1	C21	1
C9	0	C22	0
C10	0.5	C23	1
C11	1	C24	0
C12	1	C25	2
C13	0	C26	0
Sum			13.5
Weighted sum			15.5
Weighted score			0.27
Category			Medium

TABLE 5.11: Matching criteria for the Kazuar malware.

5.12 LightNeuron

Similarly belonging to the Turla threat group, the LightNeuron malware is a Microsoft Exchange backdoor which installs itself as malicious transport agent [78] in an Exchange environment. This way, the implant is able to process incoming mails before they are forwarded to the user. After logging date and sender, the backdoor calls the companion DLL in order to execute a command, modify or block the e-mail or just do nothing. The companion DLL exports three functions which are named suspiciously in more recent samples. Strings in the library are decrypted at the beginning of program execution, then it decrypts the configuration file which is encrypted with RSA1024/AES256 in a hybrid way. The RSA key used is available hard-coded in the binary. In addition to the configuration file, there is a rule file available which describes the handling of incoming or outgoing e-mails. The adversary is able to send commands embedded in pictures via JPG or PDF attachments, the command output can be sent back using steganography similarly [28]. Table

5.12 shows the weighted score of 0.35 and the matched criteria. LightNeuron is high sophisticated.

Criteria	Value	Criteria	Value
C1	2	C14	1
C2	0	C15	0
C3	0	C16	1
C4	2	C17	0
C5	0	C18	0
C6	1	C19	0
C7	0	C20	0
C8	0.5	C21	2
C9	0.5	C22	1
C10	1	C23	1
C11	1	C24	1
C12	1	C25	1
C13	1	C26	0
Sum			18
Weighted sum			19.75
Weighted score			0.35
Category			High

TABLE 5.12: Matching criteria for the LightNeuron malware.

5.13 MATA

MATA is a multi-platform malware targeting Windows, Linux and MacOS, and assessed to be deployed through a loader stage as AES encrypted payload. Configuration data is either provided AES encrypted in the registry or, if not available, hardcoded in the binary itself. On execution, the malware can load up to 15 plugins, either from disk, from a given HTTP(S) URL or from the C2 server. The developers call the C2 mechanism MataNet and the implants implement both, server and client functionality. There are plugins for routing traffic between MataNet nodes and providing a proxy service available. There is no persistence method mentioned in the source [55]. With a weighted score of 0.28 MATA is assessed as medium sophisticated. Table 5.13 shows the matched criteria.

Criteria	Value	Criteria	Value
C1	0	C14	1
C2	1	C15	1
C3	0	C16	0
C4	1.5	C17	1
C5	0	C18	0
C6	1	C19	0
C7	0	C20	1
C8	1	C21	0
C9	0	C22	0
C10	0	C23	1
C11	1	C24	1
C12	1	C25	1
C13	1	C26	0
Sum			14.5
Weighted sum			16
Weighted score			0.28
Category			Medium

TABLE 5.13: Matching criteria for the MATA malware.

5.14 MICROPSIA

The information stealer MICROPSIA is packed multiple times and delivered as RAR archive. The other packaging layers involved are UPX and RAR SFX which contains ultimately the actual malware itself. It is capable of capturing keyboard input and screenshots as well as exfiltrating files with Microsoft Office extensions. Prior to exfiltration, the files are packed and encrypted using WinRAR and a hard-coded key. In addition to the malware reported by Bar and Lancaster [5], Flossman and Scott [32] observed malware that is based on the initial MICROPSIA code base, however, the general capabilities have not changed much. Added capabilities are command execution, remote downloading files to execute and base64 encoding to obfuscate C2 communications. Table 5.14 shows the score of 0.11 and therefore MICROPSIA is low sophisticated.

5.15 MoleNet

According to Cybereason [19], MoleNet is an obfuscated downloader written in .NET. Aside from downloading the next stage payload, the malware implements profiling of the operating system, anti-debugging checks and persistence via registry run key [94]. The used type of obfuscation is not mentioned, so for the assessment control-flow obfuscation is assumed. The matching criteria are describe in table 5.15. A weighted score of 0.09 results in the category very low sophisticated.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	1	C15	0
C3	0	C16	0
C4	0	C17	0
C5	0	C18	0
C6	1	C19	0
C7	0	C20	0
C8	1	C21	1
C9	0	C22	0
C10	0	C23	0
C11	1	C24	0
C12	1	C25	1
C13	1	C26	0
Sum			8
Weighted sum			6
Weighted score			0.11
Category			Low

TABLE 5.14: Matching criteria for the MICROPSIA malware.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	0	C15	0
C3	1	C16	0
C4	0	C17	0
C5	0	C18	0
C6	1	C19	0
C7	0	C20	0
C8	1	C21	1
C9	0	C22	0
C10	0	C23	0
C11	1	C24	0
C12	1	C25	1
C13	0	C26	0
Sum			7
Weighted sum			5
Weighted score			0.09
Category			Very Low

TABLE 5.15: Matching criteria for the MoleNet malware.

5.16 OceanDrive

The RAT OceanDrive uses Google Drive as C2 mechanism and is delivered as RAR archive in spearphishing mails [99]. Along the malware, there is a decoy executable available in the archive which is started before the malicious program and displays a password form in order to mimic a decryption process. OceanDrive copies itself to the startup directory for persistence on the initial run. Using hard-coded credentials, the malware receives commands from the Google Drive, executes them and writes the output into a file in the Google Drive directory. The sample described by Hacquebord and Remorin implements no encryption or obfuscation mechanisms [39]. OceanDrive is a very low sophisticated malware, based on the matched criteria in table 5.16.

5.17 PlugX

While there are reports about multiple versions of PlugX available [64] [134] [139], this assessment focus on the core PlugX variant [16]. The typical PlugX loading process uses a legitimate executable, a malicious loader DLL as well as an encrypted file which contains the main payload. Persistence is achieved via registry run key or installation as a service. According to the Computer Incident Response Team Luxembourg (CIRCL), the malware sample analyzed makes use of logging functionality extensively and writes the output to the user profile directory. Capabilities and features are implemented in their own C++ source files, e.g. `XPlugOption.cpp` or

`XPlugDisk.cpp`, which shows a modular-like architecture, however, the features are statically build into the malware, not qualifying to match criteria C20 completely. Based on the score of 0.13 (see table 5.17), the PlugX variant analyzed by CIRCL is assigned to the category low sophisticated.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	0	C15	1
C3	0	C16	0
C4	0	C17	0
C5	0	C18	0
C6	0	C19	0
C7	0	C20	0
C8	1	C21	1
C9	0	C22	0
C10	0	C23	0
C11	0	C24	0
C12	1	C25	1
C13	0	C26	0
Sum			5
Weighted sum			3.5
Weighted score			0.06
Category			Very Low

TABLE 5.16: Matching criteria for the OceanDrive malware.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	1	C15	0
C3	0	C16	0
C4	2	C17	0
C5	0	C18	0
C6	0	C19	0
C7	0	C20	0.5
C8	1	C21	1
C9	0	C22	0
C10	0	C23	0
C11	1	C24	0
C12	1	C25	2
C13	0	C26	0
Sum			9.5
Weighted sum			7.5
Weighted score			0.13
Category			Low

TABLE 5.17: Matching criteria for the PlugX malware.

5.18 POWERSTATS

POWERSTATS is a Powershell based implant delivered via Visual Basic script. The malware uses not further specified string obfuscation and dead code blocks to harden analysis. After collecting system information and uploading the data to the C2 server, the malware tries to download a specific file containing further Powershell code to execute from the C2 server in an endless loop. In case an error occurs during C2 communication, the malware deletes itself [69]. Table 5.18 shows the matched criteria for POWERSTATS and the weighted score of 0.11. The malware is low sophisticated.

5.19 Pteranodon

The Gamaredon Pteranodon malware family consists of multiple nested self-extracting (SFX) archives delivered via word macros in injected remote templates [101]. After the SFX archives executed the included scripts which decrypt and unpack the next stage, a .NET binary, several legitimate office DLL files and Word as well as Excel macros. The libraries are used to create document files containing the macros which

are executed afterwards. Persistence is achieved by creating two scheduled tasks. A later stage, which was not analyzed in the malware reports, can only be executed after decoding it with a key based on the targets drive serial number [138] [61]. Table 5.19 shows the weighted score of 0.11 and the according sophistication of low.

Criteria	Value	Criteria	Value
C1	2	C14	0
C2	0	C15	0
C3	1	C16	0
C4	0	C17	0
C5	0	C18	0
C6	0	C19	0
C7	0	C20	0
C8	0	C21	0
C9	1	C22	0
C10	0	C23	0
C11	1	C24	0
C12	1	C25	1
C13	0	C26	0
Sum			7
Weighted sum			6
Weighted score			0.11
Category			Low

TABLE 5.18: Matching criteria for the POWERSTATS malware.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	1	C15	0
C3	1	C16	0
C4	0	C17	0
C5	0	C18	0
C6	0	C19	0
C7	0	C20	0
C8	1	C21	1
C9	0	C22	0
C10	0	C23	0
C11	0	C24	0
C12	1	C25	1
C13	0	C26	1.5
Sum			7.5
Weighted sum			6
Weighted score			0.11
Category			Low

TABLE 5.19: Matching criteria for the Pteranodon malware.

5.20 Regin

Regin is a modular malware platform. Relying on multiple stages which load the next encrypted payloads, the loading chain is different for 32- and 64-bit. On 32-bit systems, the malware loads next stages from extended attributes (EA) within the NTFS file system [88] or encrypted data from the registry. Additionally, the 32-bit version loads a kernel driver before the dispatcher is loaded from the virtual file system (VFS). The 64-bit version loads following stages from unallocated space right after the last partition on the hard disk. Instead of using a kernel driver, the dispatcher DLL file is directly loaded by the second 64-bit stage. The platform uses fake certificates which are trusted because of an added certificate authority (CA) during deployment stage as well as RC5 encryption with a key shared across samples. Regin is highly extensible and modular implementing various capabilities in different modules, including a plugin for logging commands entered on a base station controller (BSC) of a Global System for Mobile Communication (GSM) provider. The malware implements more than three C2 channels. While the initial infection vector is unknown, both sources assess that the exploitation of zero-day vulnerabilities is likely. Symantec reports Regin originated from Yahoo messenger in one

analyzed infection [57] [133]. Table 5.20 shows the matched criteria and the assessed sophistication as very high.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	1	C15	0
C3	0	C16	1.5
C4	3	C17	1
C5	0	C18	2.5
C6	1	C19	1
C7	0	C20	1
C8	1	C21	2
C9	1	C22	2
C10	0	C23	1
C11	1	C24	1
C12	1	C25	3
C13	1	C26	0
Sum			26
Weighted sum			32.5
Weighted score			0.58
Category			Very High

TABLE 5.20: Matching criteria for the Regin platform.

5.21 SharpStage

The SharpStage .NET malware implements various features, such as downloading and dropping additional payloads, accessing the Dropbox-API to download or upload files, execute commands via Powershell, capturing the screen and fingerprint the target system using Windows Management Instrumentation (WMI) queries. Multiple samples have different persistence mechanisms using well-known methods via registry run keys or scheduled tasks. Additionally, multiple samples share the same string used as a mutex name. Cybereason mentions code obfuscation, but provides no further description or examples. SharpStage checks on start if an Arabic keyboard is installed and only continues execution if one is available. One of the samples mentioned by Cybereason implements a decoy functionality and drops a document upon execution [19]. Table 5.21 shows the criteria which match SharpStage and the weighted score of 0.10. The malware is considered as very low sophisticated.

5.22 Spyder

Spyder consists of a loader which persists through DLL search order hijacking [96] and contains the main payload which is not encrypted, but the PE header was filled with zeros. The developers made the initialization routine artificially complicated in

order to harden static analysis, however, no further obfuscation methods were mentioned by the analysts. Also, the C2 communication is not additionally encrypted, but checks are implemented in order to verify the integrity of C2 server answers. The report mentions encrypted modules which can be decrypted and executed by the backdoor, but provides no further information about them. Nevertheless, the architecture of Spyder can be seen as modular. Some of the samples contain lots of debug strings that are printed via `OutputDebugStringA` if they are related to C2 communications. Log messages related to other functionality are either also printed via `OutputDebugStringA` or XOR encrypted, based on configuration parameters [23]. Based on the amount of criteria matched in table 5.22, the malware is assessed as low sophisticated.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	0	C15	0
C3	1	C16	0
C4	0	C17	0
C5	0	C18	0
C6	0.5	C19	0
C7	0	C20	0
C8	1	C21	1
C9	0	C22	0
C10	0	C23	0
C11	1	C24	0
C12	1	C25	1
C13	0	C26	1
Sum			7.5
Weighted sum			5.5
Weighted score			0.10
Category			Very Low

TABLE 5.21: Matching criteria for the Sharp-Stage malware.

Criteria	Value	Criteria	Value
C1	0	C14	0
C2	0.5	C15	0
C3	2	C16	0
C4	0	C17	0
C5	0	C18	0
C6	0.5	C19	0
C7	0	C20	1
C8	1	C21	1
C9	0	C22	0
C10	0	C23	0
C11	1	C24	1
C12	1	C25	1
C13	0	C26	0
Sum			9.5
Weighted sum			7.5
Weighted score			0.13
Category			Low

TABLE 5.22: Matching criteria for the Spyder malware.

5.23 Winnti

While there are reports about other variants and versions of Winnti [58] [122] [143], the base for this assessment mainly is the report of Novetta [110]. The Winnti dropper requires a command-line argument which is used as a key for decrypting the DES-encrypted payload. After extracting the service component, the dropper extracts and decrypts included configuration data. Via call to `rundll32.exe` the dropper runs the service component which starts the installation process. During the process, the service component extracts the Winnti engine, adds a correct PE header to the stripped binary and calls the installation routine. The engine executable is never written to disk. Beside moving previously extracted payloads to the system

directory, the engine creates a new service for persistence. The worker component supports two C2 channels, the Winnti custom TCP protocol directly or encapsulated within HTTP(S). RAT functionality is not included in the Winnti components and depends on various plug-ins downloaded from the C2 server. Haruyama describes a kernel rootkit available in some worker components [40]. The driver hooks network device interface specification (NDIS) protocol handlers in order to create a covert C2 channel. Based on the matched criteria in table 5.23, Winnti is a high sophisticated malware.

Criteria	Value	Criteria	Value
C1	0	C14	1
C2	1	C15	0
C3	0	C16	0
C4	2	C17	0
C5	0	C18	0
C6	0	C19	2
C7	0	C20	1
C8	1	C21	1
C9	0	C22	0
C10	0.5	C23	1
C11	1	C24	1
C12	1	C25	2
C13	1	C26	2
Sum			18.5
Weighted sum			20.5
Weighted score			0.36
Category			High

TABLE 5.23: Matching criteria for the Winnti malware.

5.24 X-Agent

Base for the X-Agent assessment is the work from Mehta, Leonard and Huntley [77], the CHOPSTICK analysis included in the APT28 report by FireEye [31] and the Sednit reports by ESET including an analysis of the X-Agent source code for Linux [25]. In addition, ESET describes updated features of X-Agent at a later time [26]. CHOPSTICK is the name FireEye uses for the malware X-Agent. The modular backdoor is written in C++ and supports two C2 channels, HTTP and POP3/SMTP. HTTP communication is plain-text and the mail channel is TLS encrypted, however, in both cases the messages are XOR- or RC4-encrypted with the key included in the message. X-Agent has the capability to map message queues to USB drives. This way air-gapped systems can contact the C2 server without having internet access and this qualifies as a third C2 channel as well as an uncommon network feature. Isolated systems can be infected using the same USB drives as they contain an autorun

file created by the malware. While the backdoor has the capability to store configuration data RC4 encrypted in the registry, Mehta et al describe that most X-Agent samples do not use this functionality. Persistence is achieved through one of the registry run keys [94]. X-Agent version 4 includes an update to the string obfuscation methodology which is unique per sample instead of the previously used XOR-based encryption. Table 5.24 shows the weighted score of 0.23 and the assessed medium sophistication based on the matched criteria.

Criteria	Value	Criteria	Value
C1	2	C14	0
C2	0	C15	0
C3	0	C16	1
C4	0	C17	0
C5	0	C18	0
C6	0	C19	0
C7	0	C20	1
C8	1	C21	1
C9	0	C22	0
C10	0.5	C23	1
C11	1	C24	0
C12	1	C25	3
C13	1	C26	0
Sum			13
Weighted sum			13
Weighted score			0.23
Category			Medium

TABLE 5.24: Matching criteria for the X-Agent malware.

5.25 XDSpy

XDSpy originally is the group name for an adversary rather than the tool set used, but for this research, the main payload *XDDown* as well as the plugins *XDRecon*, *XDList*, *XDMonitor*, *XDUpload*, *XDLoc* and *XDPass* are subsumed under the name *XDSpy*. The orchestrator component of XDSpy persists via registry run key and is responsible for downloading and loading the plugins. Faou mentions the use of stack strings and shifted characters for string obfuscation as well as dynamically resolved Windows APIs after 2013. The malware uses a hard-coded date-based kill switch and exits if a given date is exceeded. The module download is additionally encrypted with a 2 byte XOR key while some of the modules encrypt their data with the RC4 cipher and different hard-coded keys. Also, screenshots seem to be the only output files which are encrypted before written to disk. The malware does not implement persistence for plugins. As they are dropped into temporary directories, they need to be downloaded every time the temporary data got deleted and user logs in. Additionally, XDSpy implements the uncommon feature of tracking a users

location by measuring the signal strength of nearby WiFi access points [29]. The malware is assessed as medium sophisticated and the matching criteria are listed in table 5.25.

Criteria	Value	Criteria	Value
C1	1	C14	0
C2	0	C15	0
C3	0	C16	0.5
C4	0	C17	0
C5	1	C18	0
C6	1	C19	0
C7	1	C20	1
C8	1	C21	1
C9	0	C22	0
C10	0	C23	1
C11	1	C24	0
C12	1	C25	1
C13	1	C26	1
Sum			13.5
Weighted sum			13
Weighted score			0.23
Category			Medium

TABLE 5.25: Matching criteria for the as XDSpy subsumed malware.

5.26 Sophistication Assessment Survey

The goal of the conducted survey is to collect the estimated sophistication of malware families in combination with a self-assessment of the participant's experience as well as their confidence level towards specific assessments. This allows to apply weights to emphasize a high level of experience and confidence, but also enables the participation of beginner-level analysts that might not be very confident regarding their assessment. Additionally participants need to choose their knowledge source which also applies as a weight towards the sophistication assessment. Participants need to answer the following self-assessment questions:

1. **How would you describe your role?**
   Possible answers: Malware Analyst, Threat Intelligence Analyst, Other (custom answer by participant)

2. **For how long have you been researching malware?**
   Possible answers: <1 year, 1-5 years, 6-10 years, >10 years

3. **How would you estimate your malware analysis experience level?**
   Possible answers: Beginner, Intermediate, Advanced, Expert

The survey had a total of 39 participants of various experience levels, while the majority has more than five years of experience (>61%) and see themselves at least on an advanced experience level (>58%). Tables 5.26, 5.27 and 5.28 show the answers to the self-assessments questions of the survey.

Role description	**Amount of participants**
Malware Analyst	16
Threat Intelligence Analyst	17
Other	6
Thereof considering themselves doing both	*4*
Total	**39**

TABLE 5.26: Survey participant's job role descriptions

Experience in years	**Amount of participants**
<1 year	3
1-5 years	12
6-10 years	12
>10 years	12
Total	**39**

TABLE 5.27: Survey participant's malware analysis experience in years

Experience level	Amount of participants
Beginner	3
Intermediate	13
Advanced	18
Expert	5
Total	**39**

TABLE 5.28: Survey participant's malware analysis experience level

Next, participants need to state their knowledge source for specific malware families. As stated above, this allows the application of weight depending on the knowledge source, but also skipping unknown malware families in the survey. Participants were able to choose from the following answers per malware family:

- Detailed malware analysis (D)
- Malware analysis (A)
- Reading analysis reports (R)
- Hearsay (H)
- Unknown (U)

Table 5.29 shows the amount of answers given per malware family and knowledge level, the table headings were shortened to fit the page. The amount of participants which answered varies per malware family. Some participants skipped questions. The mean knowledge score ($K_{\text{mean}} \in \mathbb{R}, 1 \leq K_{\text{mean}} \leq 5$) relies on the value per category, from 5 for detailed malware analysis to 1 for unknown. It is important to notice that not necessarily every participant that voted for a knowledge source also voted for a sophistication estimation later, however, participants who voted for a malware family which is unknown to them are removed from the survey results by setting the weight to 0 (described in detail in section 6.1).

Family	**D** (5)	**A** (4)	**R** (3)	**H** (2)	**U** (1)	$K_{\mathbf{mean}}$
BabyShark	0	7	6	12	7	2.41
BadNews	0	5	11	5	11	2.31
DoubleFantasy	2	5	14	6	5	2.78
Duqu 2.0	2	5	19	3	4	2.94
EquationDrug	2	2	17	7	4	2.72
EvilGnome	1	4	5	8	14	2.06
FatDuke	4	5	11	8	5	2.85
Flame	2	3	23	3	2	3
KASPERAGENT	1	3	8	6	14	2.09
Kazuar	7	3	15	7	1	3.24
LightNeuron	7	5	12	6	2	3.28
MATA	0	6	9	3	14	2.22
MICROPSIA	1	3	7	7	14	2.06
MoleNet	0	3	8	5	16	1.94
OceanDrive	6	5	6	3	12	2.69
PlugX	9	8	10	4	2	3.55
POWERSTATS	8	3	9	3	9	2.94
Pteranodon	2	7	4	9	11	2.39
Regin	5	1	19	6	2	3.03
SharpStage	0	1	8	6	17	1.78
Spyder	0	2	6	2	22	1.63
Stuxnet	4	0	25	3	1	3.09
Winnti	4	7	16	4	2	3.21
X-Agent	6	8	12	4	3	3.3
XDSpy	2	2	11	5	12	2.28

TABLE 5.29: Knowledge sources for specific malware families.

Similarly to the knowledge source, the confidence of estimation also defines the weight of assessments by the participants. Both of them motivate participants to give their sophistication estimation even if they are not entirely sure. Otherwise participants would rather skip an estimation because they have not analyzed the malware in detail or themselves at all. The confidence levels available to the participants are:

- Confident (C)
- Somewhat confident (S)
- Not confident (N)

Table 5.30 shows the confidence level per malware family. Similar to table 5.29, the mean confidence score ($C_{\text{mean}} \in \mathbb{R}, 1 \leq C_{\text{mean}} \leq 3$) relies on the values assigned to the confidence levels, from 3 for a confident assessment to 1 for a non-confident assessment.

Family	**C** (3)	**S** (2)	**N** (1)	$C_{\mathbf{mean}}$
BabyShark	5	7	0	2.42
BadNews	4	5	1	2.3
DoubleFantasy	11	2	2	2.6
Duqu 2.0	11	4	2	2.53
EquationDrug	10	2	3	2.47
EvilGnome	3	2	6	1.73
FatDuke	10	1	1	2.75
Flame	12	5	2	2.53
KASPERAGENT	3	5	2	2.1
Kazuar	9	5	0	2.64
LightNeuron	9	4	1	2.57
MATA	6	3	1	2.5
MICROPSIA	5	2	2	2.33
MoleNet	4	2	3	2.11
OceanDrive	9	4	0	2.69
PlugX	12	4	0	2.75
POWERSTATS	6	4	1	2.45
Pteranodon	6	3	2	2.36
Regin	9	7	2	2.39
SharpStage	1	2	4	1.57
Spyder	1	3	1	2
Stuxnet	11	5	2	2.5
Winnti	10	5	1	2.56
X-Agent	10	5	0	2.67
XDSpy	3	3	3	2

TABLE 5.30: Confidence level of sophistication assessment for specific malware families.

Next to the confidence level, the participants were asked to give an estimation about the sophistication of malware families in the previously introduced categories:

- Very High (VH)
- High (H)
- Medium (M)
- Low (L)
- Very Low (VL)

The table 5.31 shows the amount of votes per sophistication category as well the mean sophistication score ($S_{\text{mean}} \in \mathbb{R}, 1 \leq S_{\text{mean}} \leq 5$). As with previous mean scores, S_{mean} also relies on the numerical value assigned to each category ranging from 5 (very high sophisticated) to 1 (very low sophisticated).

Family	**VH** (5)	**H** (4)	**M** (3)	**L** (2)	**VL** (1)	**Mean**
BabyShark	0	0	7	4	1	2.50
BadNews	0	1	2	7	0	2.40
DoubleFantasy	9	7	1	0	0	4.47
Duqu 2.0	19	0	0	0	0	5.00
EquationDrug	12	4	1	0	0	4.65
EvilGnome	0	0	6	5	2	2.31
FatDuke	2	10	3	0	0	3.93
Flame	16	5	0	0	0	4.76
KASPERAGENT	0	0	3	5	2	2.10
Kazuar	0	12	6	1	0	3.58
LightNeuron	4	8	6	0	0	3.89
MATA	0	7	4	0	0	3.64
MICROPSIA	0	0	2	5	2	2.00
MoleNet	0	0	0	7	2	1.78
OceanDrive	0	1	6	7	2	2.38
PlugX	0	6	9	4	0	3.11
POWERSTATS	0	0	4	7	2	2.15
Pteranodon	0	0	4	6	3	2.08
Regin	17	5	0	0	0	4.77
SharpStage	0	0	2	5	0	2.29
Spyder	0	4	0	1	0	3.60
Stuxnet	19	1	0	0	0	4.95
Winnti	1	14	4	0	0	3.84
X-Agent	1	10	7	1	0	3.58
XDSpy	0	2	7	1	0	3.10

TABLE 5.31: Sophistication assessments for specific malware families and unweighted mean sophistication score.

Chapter 6

Evaluation and Discussion of Results

This chapter includes the evaluation of survey results and the comparison with the sophistication assessments from chapter 5. It explains the differences between estimated sophistication from participants of the survey and the assessed sophistication using the approach of this research project and points out specific aspects of the assessment approach which need further adjustments.

6.1 Survey Evaluation

In order to emphasize a higher experience level of participants as well as detailed analysis of malware families and high confidence in the sophistication estimation, weights need to be applied upon the raw survey results. The weights were chosen up from a minimum of 0.5 to a maximum of 2.0 which halves or doubles the impact of specific aspects. Experience level, experience time and knowledge source can reduce and enhance the overall weight of assessments while the confidence of assessment is either 1.0 for a confident estimation or lower for non-confident estimations, so the confidence can not enhance the effective weight of an assessment done by participants.

Table 6.1 shows the weights used for the experience level of participants. Intermediate is the base weight with 1.0. Less experienced analysts as well as more experienced analysts need a different weight and the step size in this case is 0.50. The overall range therefore is from 0.50 for beginner level analyst to 2.00 for expert level analysts. Similarly, the experience time for analysts uses the same ranges. Less than one year experience is weighted with 0.50 while more than 10 years of experience are weighted with 2.00. While there is an overlap between the level of experience and the time spent as an analyst, resulting in an overemphasis of experts with a considerable experience time, it allows also to emphasize participants which are solid analysts for quite some time, but would hesitate to consider themselves as advanced. Table 6.2 shows the weights applied based on the amount of time spent analyzing malware.

Experience level	Weight
Beginner	0.50
Intermediate	1.00
Advanced	1.50
Expert	2.00

TABLE 6.1: Weights applied through the experience level of participants.

Experience time	Weight
<1 year	0.50
1-5 years	1.00
6-10 years	1.50
>10 years	2.00

TABLE 6.2: Weights applied through the experience time of participants.

Another important weight is the source of knowledge. Analysts can only be really sure about an sophistication assessment, if it bases on a detailed analysis of the specific malware. This is the reason, why only analysis and detailed analysis emphasize the weight of an assessment. While reading analysis reports state that analyst themselves collected information about a specific malware family, reports can be incomplete or take a specific perspective only. This is why assessments from this research are only examples as they also base on malware reports. Hearsay has the lowest weight for obvious reasons. Table 6.3 shows all weights related to the knowledge source.

Knowledge source	Weight
Unknown	0.00
Hearsay	0.50
Reading analysis reports	0.75
Malware analsis	1.50
Detailed malware analysis	2.00

TABLE 6.3: Weights applied through the knowledge source of malware families.

The confidence weights range is from 0.50 to 1.00 based on the amount of confidence assessed. For completeness, table 6.4 show the confidence weights.

Confidence	Weight
Not confident	0.50
Somewhat confident	0.75
Confident	1.00

TABLE 6.4: Weights applied through the confidence of estimation.

Table 6.5 shows the survey results after applying the weights. The tendency column shows whether the majority of participants voted for the upper (i.e. high

sophisticated) or lower (i.e. low sophisticated) half which is useful for detecting high weighted outliers. Table 6.6 shows the distribution of the survey results. Already noticeable is the missing amount of very low sophisticated malware families. This is because only a few participants voted for very low sophistication in total and these were outnumbered and -weighted by other participant's votes. The next section (6.2) compares the survey results to the assessments done with the approach introduced in this thesis.

Family	Sophistication	Tendency
BabyShark	Medium	↓
BadNews	Low	↓
DoubleFantasy	Very High	↑
Duqu 2.0	Very High	↑
EquationDrug	Very High	↑
EvilGnome	Low	↓
FatDuke	High	↑
Flame	Very High	↑
KASPERAGENT	Medium	↓
Kazuar	High	↑
LightNeuron	High	↑
MATA	High	↑
Micropsia	Low	↓
MoleNet	Low	↓
OceanDrive	Low	↓
PlugX	Medium	↑
POWERSTATS	Low	↓
Pteranodon	Low	↓
Regin	Very High	↑
SharpStage	Low	↓
Spyder	High	↑
Stuxnet	Very High	↑
Winnti	High	↑
X-Agent	High	↑
XDSpy	Medium	↑

TABLE 6.5: Weighted survey results.

Sophistication level	Amount of survey results
Very high	6
High	7
Medium	4
Low	8
Very low	0
Total	25

TABLE 6.6: Distribution of survey results.

6.2 Result Evaluation

This section shows, describes and explains differences between the estimated sophistication by the survey participants and the sophistication assessed by the approach introduced in this thesis. Noticeable is that more than half of all malware families were voted very high or high sophisticated while the weighted survey results do not include any votes for very low sophistication. The raw survey data contain votes in the category very low sophisticated, however, in the weighted survey results these were outperformed or outweighted (because of experience or confidence) by other votes. Table 6.7 shows the mismatching sophistication categories. The colors emphasize the amount of difference. Yellow represents a difference by one category, while orange means that the results are off by two categories.

Family	Voted	Assessed	Off by
BabyShark	Medium	Very Low	**2**
BadNews	Low	Low	0
DoubleFantasy	Very High	Very High	0
Duqu 2.0	Very High	Very High	0
EquationDrug	Very High	High	1
EvilGnome	Low	Low	0
FatDuke	High	High	0
Flame	Very High	Very High	0
KASPERAGENT	Medium	Low	1
Kazuar	High	Medium	1
LightNeuron	High	High	0
MATA	High	Medium	1
Micropsia	Low	Low	0
MoleNet	Low	Very Low	1
OceanDrive	Low	Very Low	1
PlugX	Medium	Low	1
POWERSTATS	Low	Low	0
Pteranodon	Low	Low	0
Regin	Very High	Very High	0
SharpStage	Low	Very Low	1
Spyder	High	Low	**2**
Stuxnet	Very High	Very High	0
Winnti	High	High	0
X-Agent	High	Medium	1
XDSpy	Medium	Medium	0

TABLE 6.7: Comparison between assessed and voted sophistication levels per malware family. Yellow color means a difference by one category, orange by two.

The result comparison shows almost a complete match for very high sophisticated malware with only EquationDrug off by one category. The other categories show multiple mismatches, some by two categories. Remarkably, the mismatches between the sophistication levels are only in one direction. The estimated sophistication voted by the survey participants is higher than the assessed sophistication

using the approach introduced in this research project. This shows that the methodology in combination with the chosen criteria takes a more critical perspective in assessing sophistication levels. The following paragraphs analyze the survey data as well as the assessment of the specific malware families further in order to legitimate the assessment done or to expose weaknesses in the chosen methodology, starting with the families which have a difference of two categories compared to the estimated sophistication from the survey. It is important to notice that the assessments entirely base on malware reports and therefore the underlying information situation may differ from analysts that have done a detailed malware analysis on specific malware families. Also, some criteria might be missing in the malware reports because they were not part of the analysis goals and thus these malware families might be assessed to a lower sophistication category.

BabyShark is, according to the introduced methodology, a very low sophisticated malware. First, it is important to remark that the various stages of BabyShark do not use PE binaries. The criteria developed throughout this research have a focus on PE executables, however, POWERSTATS is also not using Windows binaries and has the same category assigned as the participants of the survey voted. Some of the criteria, such as C7 or C8, do not match per default, but can only match on PE files in general. Second, the tendency of votes is in the lower spectrum of categories, however, most votes (58%) assigned the medium sophistication category to BabyShark. This would also mean that the malware would need to match a similar amount of criteria as Kazuar (see section 5.11) or X-Agent (see section 5.24). Third, as a script based malware, analysts can directly start examining the samples without the need of disassembly or decompilation. As, based on the analysis report, there does not seem to be any further anti-analysis measures taken, BabyShark cannot be considered a medium sophisticated malware.

The **Spyder** malware is used by groups which also use the famous (see e.g. [136] [137] [135]) Winnti malware and therefore might be considered similarly sophisticated. While Winnti is high sophisticated, based on the matched criteria, Spyder is only low sophisticated. This might be due to missing information from analysis reports, but Spyder seems to implement way less anti-analysis and OPSEC related features. Also, something similar uncommon like the network driver used by Winnti is missing. Only two of the participants have analyzed Spyder themselves and only one of them is confident in their assessment. Because of the missing features in the malware and flaws, such as plain-text logging strings, the malware cannot be considered high sophisticated.

EquationDrug differs from the survey results by one category and is assessed to be a high sophisticated malware. In comparison to other malware families which are rated very high sophisticated, the main scoring difference is a missing zero-day exploit. Other families incorporate the use of exploits for infection (code execution) or privilege escalation, however, EquationDrug is already the main payload used in campaigns. At the point when EquationDrug is rolled out, there is no need to

escalate privileges as they are already achieved by a previous dropper component, such as DoubleFantasy. Main payloads likely implement more features than a dropper or a malware which is a combination of both, but the amount of implemented capabilities is not in scope for this research. As mentioned in chapter 2 and 3, there is a challenge in using the amount of capabilities for sophistication measurement. Malware implementing lots of capabilities can outperform implants which focus on anti-analysis and OPSEC measures resulting in biased sophistication ratings. Based on the set of criteria defined in this research project and the information given in malware reports, EquationDrug cannot assessed as very high sophisticated, however, the malware can be seen as an edge case which might not be covered by the current criteria set used for the assessment in this project.

The implant **KASPERAGENT** just crossed the category boundary with a score of 0.12 and therefore is assessed low sophisticated. The majority of participants voted low sophisticated while some participants with a greater weight voted medium sophisticated, thus the overall voted result is medium sophisticated. The malware report does not state any uncommon functionality. Obfuscation measures are malformed strings as well as a legitimate packer tool. Compared to other medium sophisticated assessed malware families, e.g. Kazuar, MATA or X-Agent, there is a considerable gap in functionality which does not qualify KASPERAGENT to be medium sophisticated. Additionally, the malware MICROPSIA which matches a similar amount of criteria was voted low sophisticated by the survey participants.

Kazuar was voted to be high sophisticated, but scored less in the assessment. As this malware is associated with the Turla group and as they use also high sophisticated malware such as LightNeuron, the participants might be biased because of other malware belonging to the group. The majority (63%) of participants voted Kazuar as high sophisticated. While only 74% of participants which voted for Kazuar have specified their confidence, 64% of them are confident in their estimation. Compared to LightNeuron, Kazuar implements less anti-analysis features, i.e. only using a commodity obfuscator, and the overall implemented capabilities seem rather basic. Also, Kazuar implements verbose logging to symmetric encrypted files which can be examined by analysts as the key is available in the .NET based executable. Kazuar lacks too many features and OPSEC measures to be considered high sophisticated.

The malware **MATA** was voted high sophisticated while it is assessed medium sophisticated in the application of the introduced approach. Based on the raw survey data, 63% voted for high sophistication and 71% are confident in their assessment. 57% of the participants which voted high sophistication have analyzed MATA themselves. The remaining 36% of voters estimated MATA to be medium sophisticated with 50% of them being confident in their decision. Also 50% of the medium confidence voters analyzed the malware themselves. However, there is a significant difference to other malware which is assessed and voted as high sophisticated, such as FatDuke, LightNeuron or Winnti. Missing uninstallation functionality, attributable strings, just a single C2 channel, missing persistence and not encrypted hard-coded

configuration data are key differences to other high sophisticated malware families. The latter two might be due to missing information in the analysis report, however, based on the information given, the malware misses features which would be needed for a higher score and therefore a higher sophistication category.

MoleNet is, based on the assessment, very low sophisticated. The survey participants estimated MoleNet to be low sophisticated. 75%[1] of the participants which estimated MoleNet's sophistication voted for low and the remaining 25% voted for very low. 50% are confident in the low sophistication of MoleNet. Only one participant per category analyzed the malware themselves. With a score of 0.09 there is only a single criteria match missing in order to cross the boundary to low sophistication. Due to unprecise information regarding the type of obfuscation, there might be missing criteria matches in case there are multiple obfuscation features applied. The typical obfuscator for .NET samples would be represented by C3. According to Detect It Easy (DIE) [41] [42], the available sample[2] on Malpedia [72] was obfuscated with a commercial commodity obfuscator called Eazfuscator.NET [35]. The mentioned sample was obtained from a public sandbox and also mentioned in the malware analysis report [19]. This suggests that the amount of obfuscation criteria is correctly set. Without the addition of other features the category very low sophisticated seems reasonable.

The Google Drive using malware **OceanDrive** was the least sophisticated assessed malware in this research. With a score of 0.06 it is assessed very low sophisticated. As it is an .NET based malware which does not implement any protection features and includes hard-coded credentials it lacks sophistication, as also assessed by Hacquebord and Remorin [39]. The voted estimations to higher categories of sophistication might be based of the relation to APT28, however, the malware itself must be seen as very low sophisticated based on the straightforward implementation and lacking anti-analysis features.

PlugX is assessed low sophisticated based on the information available from the malware report [16]. Here is important to notice that this reports covers the PlugX version available in 2013. Very likely the malware was further developed and later published reports include information from newer samples, however, the CIRCL report was the most detailed one available. The survey participants estimated PlugX to be medium sophisticated. According to the raw survey data, 32% voted with confidence for high sophisticated and 47% voted for medium sophisticated whereof 55% are confident in their estimation. The remaining 21% voted for low sophistication with mixed confidence. Based on the information available in the malware report, the malware achieved a score of 0.13. Heavy weighted criteria such as unique encryption keys, encryption of written files, secure file deletion etc. are not mentioned in the report. There is a considerable gap between other medium assessed malware

[1] This does not align with the given table in section 5.26 because one of the participants which voted low sophisticated also voted that the malware family is unknown to them. The vote was removed during analysis because of a weight value of 0.

[2] SHA256: f323a150d7597f46d29eb3a3c56f74e11d18caf164f9176c8c1b2fa0031cc729

families, e.g. MATA or XDSpy, and PlugX. Assessing PlugX in the state reported by CIRCL as low sophisticated seems reasonable, however, a more modern PlugX version implementing additional anti-analysis and OPSEC-related features would score differently.

The malware **SharpStage** is assessed very low sophisticated with a score of 0.10. Thus the malware would be assessed low sophisticated if an additional criteria matches as the boundary is 0.11. The survey participants estimate the malware to be low sophisticated. 71% of the participants which voted for SharpStage estimate low sophistication with 20% of them being confident in their estimation. The remaining 29% voted medium sophistication and none of the participants is confident in their estimation. Based on the implemented anti-analysis and OPSEC features, SharpStage can be compared to the MoleNet malware. The difference is limited execution, as SharpStage checks for the availability of Arabic language and runs only on target systems where it is installed. Screenshots from the analysis report show that the variable names are not removed and therefore lead to a fast analysis process after the initial code-obfuscation is defeated. While the type of obfuscation is not further described in the report, a sample[3] available on Malpedia [73] uses, according to DIE, the same .NET obfuscator as the MoleNet sample described. Because of the similar set of analysis-hardening features the assessment of SharpStage to be very low sophisticated seems reasonable.

According to the introduced approach and the set of criteria, **X-Agent** is medium sophisticated with a score of 0.23. Based on the survey results, the community sees X-Agent as high sophisticated. According to the unweighted survey data, 53% percent of the participants which voted for X-Agent estimate the malware to be high sophisticated and 70% are confident in the estimation. 36% of participants voted medium sophistication whereof 71% are confident. The remaining 10% split half into confident votes for low and very high sophistication. While the information sources are also analysis reports, the majority of participants (63%) have analyzed the malware themselves. X-Agent is a multi-platform malware used by the group APT28, known for breaching the Bundestag [38] and the Democratic National Commitee [18] [125], which might be the reason why the majority of participants voted for high sophistication. In comparison to other high sophisticated assessed and voted malware families, such as FatDuke, LightNeuron or Winnti, there are considerable gaps in the identified criteria. Additionally, the malware report by Mehta, Leonard and Huntley [77] is from 2014 and covers X-Agent version 2 which is not the most recent version. ESET researchers [25] [26] describe in their research that X-Agent version 3 implement additional obfuscation and encryption measures, however, these are already included in the assessment. The general feature-set seems to be similar in both versions. Given the considerable gap between other high sophisticated malware families and X-Agent, categorizing the malware as high sophisticated is not reasonable.

[3]SHA256: 69af17199ede144d1c743146d4a7b7709b765e57375d4a4200ea742dabef75ef

6.3 Assessment Approach Observations and Discussion

This section summarizes the results of the previous section and highlights as well as discusses strengths and flaws of the introduced approach. The following observations are covered:

- Overall the introduced approach produces practical results which mostly corresponds to community estimations
- Criteria are not overall useful for malware targeting other operating systems than Windows
- Script-based malware is not covered by the criteria well enough
- General capabilities and features of malware are not represented by the current criteria set and therefore not considered at all
- The confidence of assessments might vary and cannot be expressed through the categories alone

The majority of assessments (56%) align with the estimations of the survey participants. Section 6.2 explains why some of the differing results are debatable and shows some edge cases. If sophistication ratings differ, the assessment results are always showing a lower sophistication score than the survey results. The approach introduced in this research project seems to be more critical, but overall produces usable results, given that there currently is no standardized approach on sophistication measurement.

Important to notice here is that the criteria set used for assessing the sophistication in this research project must not be static. Rather the criteria must evolve over time as techniques implemented in malware do, too. The first two observations cover different types of malware. The criteria set used in the assessments are adapted for Windows PE files. Criteria C7 and C8 for example are not useful for non-PE files which do not have an Import Address Table (IAT) or Export Address Table (EAT). Additionally, C6, stripped binary, is not useful for script-based malware. While other binary executable formats, such as ELF, have similar features and criteria can adapted to fit executables from various operating systems and platforms, script-based malware could need a complete own set of criteria to allow precise measurement. However, the script-based malware families assessed in this research project can be assigned the lower sophistication spectrum and the assessment aligned mostly with the community estimation from the survey. As discussed in the above section, only BabyShark was voted outside the lower spectrum. While coverage of other file types is desirable, the majority of all malicious software are Windows PE files. At the time of writing, VirusTotal lists 8,701,215 PE executable, 1,587,419 PE DLL and 1,364,012 DOS executable files uploaded in the last seven days,

exceeding other file types considerably [141]. Another community focusing on exchanging malware samples between researchers, Malware Bazaar, lists 141,919 PE executable and 40,174 PE DLL files which together are 52.7% of the collection [1].

Including more general capabilities of malware into the assessment is difficult. While implementing features in malware generally need resources, there still is a considerable difference between feature-complete commodity RATs and highly targeted malware developed by intelligence services. The first implements features to enhance the user experience of the operator while the latter implements anti-analysis capacities in order to harden detection, analysis and attribution. The approach used in this research project mostly ignores more general features and concentrates on anti-analysis and OPSEC measures in malware. However, there are edge cases where the general feature set must have an impact on the sophistication assessment. An example for this is EquationDrug which uses lots of modules implementing various features. As the malware is the last stage dropped during an infection chain, it lacks features that would potentially match criteria, such as exploits. Finding an approach for creating balanced criteria based on general malware capabilities is a topic for future research.

Similar to the knowledge resource question in the survey it is important to remark the base for a sophistication assessment. Even with the use of a common approach for the measurement itself, there is a clear difference between an assessment based on a single report, the rough analysis of one malware sample and the detailed analysis of multiple malware samples belonging to the same malware family. A solution could be combining the assessment base with the assessment result in a similar fashion to the NATO admiralty code (as mentioned in [21] and [105]). The admiralty code uses numbers from one to six to represent the credibility of information and letters from A to F to represent the reliability of a source. Table 6.8 shows the available values of the admiralty code.

Reliability	**Meaning**	**Credibility**	**Meaning**
A	Completely reliable	1	Confirmed by other sources
B	Usually reliable	2	Probably True
C	Fairly reliable	3	Possibly True
D	Not usually reliable	4	Doubtful
E	Unreliable	5	Improbable
F	Reliability cannot be judged	6	Truth cannot be judged

TABLE 6.8: Different levels used in NATO admiralty code.

The NATO admiralty code matrix can be adapted to represent the knowledge source as various levels and the five categories of sophistication. Highest reliability requires detailed malware analysis of multiple samples. Other gradations are detailed analysis of a single sample, rough malware analysis, malware reports and hearsay. Tables 6.9 and 6.10 show the abbreviations and meanings of the adapted admiralty code. Assessments from chapter 5 are based on malware reports and

therefore the assessment result for, e.g., Stuxnet is D5 whereas the adapted admiralty code representation for OceanDrive is D1. An assessment based on detailed analysis of multiple malware samples which results in medium sophistication is A3.

Sophistication level	Meaning
1	Very low
2	Low
3	Medium
4	High
5	Very high

TABLE 6.9: Different sophistication levels for the admiralty code adoption.

Knowledge source	Meaning
A	Completely reliable: detailed analysis of multiple malware samples belonging to the same malware family
B	Likely reliable: detailed analysis of a single malware sample
C	Usually reliable: rough analysis of one or multiple malware samples belonging to the same malware family
D	Fairly reliable: malware reports
E	Unreliable: hearsay

TABLE 6.10: Reliability part of admiralty code adapted to knowledge sources.

Chapter 7

Conclusion

This chapter summarizes the research project, concludes the results and points out further research topics. The problem approached in this research is the absence of a standardized sophistication measurement methodology for malicious software. Along with the definition of the ambiguous term "sophistication" used throughout this project, three specific research questions were introduced in chapter 3. The approach on sophistication measurement strongly relies on the leaked CIA developer guidelines included in Vault 7 leaks. As the CIA can be considered highly sophisticated, the document provides rare insights into the methodology of a threat actor of this level and a set of criteria was derived from most of the guidelines. As some of them are tied to processes along documentation and preparation of cyber operations, the information of them are usually unknown to analysts and therefore are useless as criteria. During a first evaluation of the criteria set, weaknesses were discovered and discussed in chapter 4. Based on the evaluation results, a second criteria set was created implementing adjustments and additional criteria unrelated to the CIA document. The approach then was applied to a set of 25 malware families in order to assess their sophistication levels. Chapter 5 describes the results and shows matches in detail. Additionally, the conducted survey was introduced and the data collected was presented. The answers given by participants then were weighted based on their self-assessed experience in malware analysis and their confidence in order to evaluate the previously done assessment of malware families. While most of the assessments align with the community estimations, differences were described and analyzed in chapter 6, ending with a discussion about the observations and potential solutions.

The results achieved with the introduced approach are promising. While there is no standard methodology for measuring sophistication in malware available, the majority of families assessed align with the data obtained in the survey. In case of differences between assessment and survey data, sophistication of malware families assessed by the introduced approach were always below the voted ones, however, not all participants of the survey used the complete range of sophistication as the weighted results do not include the lowest category at all.

While the results show a certain applicability of the introduced approach, there are still limitations. As different types of malware have not only common, but also

very different kind of characteristics a single set of criteria cannot generally cover various types without facilitating specific types. This research includes assessments of various script based malware families as well a Linux malware which are not covered well by criteria created with Windows PE binaries in mind. General assessments with criteria sets favoring a specific type of malware therefore lowers the score of other types resulting in biased sophistication measurements. Another limitation is the ignoring of more general malware features. Because anti-analysis and OPSEC features were focused on by the chosen criteria set, other malware features were not taken into consideration. While this research shows that sophistication measurement is possible without them generally, there are edge cases where the result of the sophistication measurement is questionable because of the resources spent by developers implementing features not covered by the criteria set.

The sophistication of malware is just one part of a bigger picture along with the used malware infrastructure, operating procedures of adversaries, campaign victimology, operator behavior etc. Future research includes completing the picture with providing measurement approaches for every component which leads to the measurement of adversary sophistication. Other topics for future work are implementing a measurement method for general malware capabilities in order to consider them as a criteria while maintaining balance and not favor big amounts of implemented features alone and the creation of criteria sets suitable for other types of malware, so the assessment does not favor Windows PE files anymore.

Appendix A

Criteria Checklist

The following checklist can be used to apply the introduced approach in combination with the chosen criteria set.

Criteria		**Matches**				**Weight**	**Results**
C1	String Obfuscation	C1.1	C1.2	C1.3		1.0	
C2	Binary/payload packing/encryption	C2.1	C2.2			1.0	
C3	Control-flow obfuscation	C3.1	C3.2			1.0	
C4	Configuration obfuscation	C4.1	C4.2	C4.3		1.0	
C5	Purpose-oriented deobfuscation	C5.1	C5.2			1.0	
C6	Stripped binary	C6.1				1.0	
C7	Obfuscated imports	C7.1	C7.2	C7.3		1.0	
C8	No sensitive exports	C8.1				0.5	
C9	Uninstallation functionality	C9.1				1.0	
C10	No attributable strings	C10.1				1.0	
C11	No suspicious strings	C11.1				0.5	
C12	Well-known network protocols	C12.1				0.5	
C13	Additional encryption layer	C13.1	C13.2			1.0	
C14	Unique encryption keys	C14.1				2.0	
C15	No temporary data written	C15.1				1.0	
C16	Encryption of written files	C16.1	C16.2			2.0	
C17	Secure deletion	C17.1				2.0	
C18	Use of exploits	C18.1	C18.2	C18.3	C18.4	2.0	
C19	Signed malware	C19.1	C19.2			2.0	
C20	Modular architecture	C20.1				2.0	
C21	Persistence	C21.1	C21.2			0.5	
C22	Specific targeting	C22.1	C22.2			2.0	
C23	Uncommon network features	C23.1				2.0	
C24	Parameter customization	C24.1				1.0	
C25	Multiple C2 channels	C25.1	C25.2	C25.3		1.0	
C26	Limited execution	C26.1	C26.2			1.0	
					Sum		
					(Divided by 56.5) **Score**		

TABLE A.1: Criteria checklist

Bibliography

[1] Abuse.ch. *MalwareBazaar Statistics*. URL: https://bazaar.abuse.ch/statistics/#filetype (visited on 07/22/2021).

[2] Dave Aitel. *CyberSecPolitics: Useful Fundamental Metrics for Cyber Power*. June 2016. URL: https://cybersecpolitics.blogspot.com/2016/06/useful-fundamental-metrics-for-cyber.html (visited on 02/15/2021).

[3] Robert J Bagnall and Geoffrey French. *The Malware Rating System (MRS)^{TM}*. 2001. URL: http://dodccrp.org/events/6th_ICCRTS/Tracks/Papers/Track7/105_tr7.pdf.

[4] Arini Balakrishnan and Chloe Schulze. *Code Obfuscation Literature Survey*. 2005.

[5] Tomer Bar and Tom Lancaster. *Targeted Attacks in the Middle East Using KASPERAGENT and MICROPSIA*. Apr. 2017. URL: https://unit42.paloaltonetworks.com/unit42-targeted-attacks-middle-east-using-kasperagent-micropsia/ (visited on 07/09/2021).

[6] Thomas Barabosch. *The malware analyst's guide to PE timestamps*. Jan. 2021. URL: https://0xc0decafe.com/malware-analyst-guide-to-pe-timestamps/ (visited on 04/12/2021).

[7] Mantvydas Baranauskas. *Windows API Hashing in Malware*. URL: https://www.ired.team/offensive-security/defense-evasion/windows-api-hashing-in-malware (visited on 07/05/2021).

[8] Daniel Bohannon. *Invoke-Obfuscation*. July 2021. URL: https://github.com/danielbohannon/Invoke-Obfuscation (visited on 07/04/2021).

[9] Ben Buchanan. *The Legend of Sophistication in Cyber Operations*. 2017.

[10] Brian Carrier. *File system forensic analysis*. Boston, Mass. ; London: Addison-Wesley, 2005. ISBN: 978-0-321-26817-4.

[11] Damien Cash et al. *Suspected APT29 Operation Launches Election Fraud Themed Phishing Campaigns*. May 2021. URL: https://www.volexity.com/blog/2021/05/27/suspected-apt29-operation-launches-election-fraud-themed-phishing-campaigns/ (visited on 07/01/2021).

[12] Mona Chalabi. *Why do Americans write the month before the day?* Dec. 2013. URL: http://www.theguardian.com/news/datablog/2013/dec/16/why-do-americans-write-the-month-before-the-day (visited on 07/03/2021).

[13] Check Point Research. *SharpPanda: Chinese APT Group Targets Southeast Asian Government With Previously Unknown Backdoor*. June 2021. URL: https://research.checkpoint.com/2021/chinese-apt-group-targets-southeast-asian-government-with-previously-unknown-backdoor/ (visited on 07/05/2021).

[14] Check Point Software Technologies. *Three Key Takeaways from WikiLeaks' Release of CIA Documents*. Mar. 2017. URL: https://blog.checkpoint.com/2017/03/10/three-key-takeaways-from-vault-7-leak/ (visited on 07/04/2021).

[15] Seokwoo Choi et al. "x64Unpack: Hybrid Emulation Unpacker for 64-bit Windows Environments and Detailed Analysis Results on VMProtect 3.4". In: *IEEE Access* 8 (2020), pp. 127939–127953. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3008900.

[16] Computer Indicent Response Center Luxembourg. *Analysis Report (TLP:WHITE): Analysis of a PlugX variant (PlugX version 7.0)*. 2013. URL: https://circl.lu/assets/files/tr-12/tr-12-circl-plugx-analysis-v1.pdf (visited on 07/10/2021).

[17] Aldo Cortesi et al. *mitmproxy: A free and open source interactive HTTPS proxy*. 2010. URL: https://mitmproxy.org/.

[18] Crowdstrike. *Our Work with the DNC: Setting the record straight*. June 2020. URL: https://www.crowdstrike.com/blog/bears-midst-intrusion-democratic-national-committee/ (visited on 07/19/2021).

[19] Cybereason. *Molerats in the Cloud: New Malware Arsenal Abuses Cloud Platforms in Middle East Espionage Campaign*. 2020. URL: https://www.cybereason.com/hubfs/dam/collateral/reports/Molerats-in-the-Cloud-New-Malware-Arsenal-Abuses-Cloud-Platforms-in-Middle-East-Espionage-Campaign.pdf (visited on 07/10/2021).

[20] Joan Daemen and Vincent Rijmen. *The Rijndael Block Cipher*. 1999. URL: https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf#page=1 (visited on 07/04/2021).

[21] Department of the Army. *Field Manual No. 2-22.3 (FM 34-52) - Human Intelligence Collector Operations*. 2006. URL: https://fas.org/irp/doddir/army/fm2-22-3.pdf.

[22] Nic DePaula and Sanjay Goel. *A Sophistication Index for Evaluating Security Breaches*. 2016.

[23] Doctor Web. *Study of the Spyder modular backdoor for targeted attacks*. 2021. URL: https://st.drweb.com/static/new-www/news/2021/march/BackDoor.Spyder.1_en.pdf.

[24] María Erquiaga, Sebastián García, and Carlos Garcia Garino. "Observer Effect: How Intercepting HTTPS Traffic Forces Malware to Change Their Behavior". In: *Computer Science – CACIC 2017*. Communications in Computer and Information Science. New York, NY, USA: Springer International Publishing, Jan. 2018, pp. 272–281. ISBN: 978-3-319-75213-6. DOI: 10.1007/978-3-319-75214-3_26.

[25] ESET. *En Route with Sednit*. 2016. URL: https://www.welivesecurity.com/wp-content/uploads/2016/10/eset-sednit-part-2.pdf (visited on 07/14/2021).

[26] ESET. *Sednit update: How Fancy Bear Spent the Year*. Dec. 2017. URL: https://www.welivesecurity.com/2017/12/21/sednit-update-fancy-bear-spent-year/ (visited on 06/19/2021).

[27] Nicolas Falliere, Liam O Murchu, and Eric Chien. *W32.Stuxnet Dossier*. 2011. URL: https://www.wired.com/images_blogs/threatlevel/2011/02/Symantec-Stuxnet-Update-Feb-2011.pdf (visited on 06/09/2021).

[28] Matthieu Faou. *Turla LightNeuron: One email away from remote code execution*. 2019. URL: https://www.welivesecurity.com/wp-content/uploads/2019/05/ESET-LightNeuron.pdf (visited on 07/09/2021).

[29] Matthieu Faou. *XDSpy: Stealing government secrets since 2011*. Oct. 2020. URL: https://www.welivesecurity.com/2020/10/02/xdspy-stealing-government-secrets-since-2011/ (visited on 07/14/2021).

[30] Matthieu Faou, Mathieu Tartare, and Thomas Dupuy. *Operation Ghost: The Dukes aren't back - they never left*. 2019. URL: https://www.welivesecurity.com/wp-content/uploads/2019/10/ESET_Operation_Ghost_Dukes.pdf (visited on 07/09/2021).

[31] FireEye. *APT28: A Window into Russia's Cyber Espionage Operations?* 2014. URL: https://www2.fireeye.com/rs/fireye/images/rpt-apt28.pdf (visited on 07/14/2021).

[32] Michael Flossman and Michael Scott. *Technical threat report: Arid Viper*. 2021. URL: https://about.fb.com/wp-content/uploads/2021/04/Technical-threat-report-Arid-Viper-April-2021.pdf.

[33] Jason Frye et al. *Cyber threat metrics*. 2012.

[34] Ricardo Kléber Martins Galvão. *Computer Forensics with The Sleuth Kit and The Autopsy Forensic Browser*.

[35] Oleksiy Gapotchenko. *Eazfuscator.NET – .NET Obfuscator and Optimizer*. URL: https://www.gapotchenko.com/eazfuscator.net (visited on 07/18/2021).

[36] Alexandre Gazet. "Comparative analysis of various ransomware virii". In: *Journal in Computer Virology* 6.1 (Feb. 2010), pp. 77–90. ISSN: 1772-9890, 1772-9904. DOI: 10.1007/s11416-008-0092-2. URL: http://link.springer.com/10.1007/s11416-008-0092-2 (visited on 07/03/2021).

[37] Alexander Gostev. *The Flame: Questions and Answers*. May 2012. URL: https://securelist.com/the-flame-questions-and-answers/34344/ (visited on 07/09/2021).

[38] Claudio Guarnieri. *Digital Attack on German Parliament: Investigative Report on the Hack of the Left Party Infrastructure in Bundestag*. June 2015. URL: https://netzpolitik.org/2015/digital-attack-on-german-parliament-investigative-report-on-the-hack-of-the-left-party-infrastructure-in-bundestag/ (visited on 07/19/2021).

[39] Feike Hacquebord and Lord Alfred Remorin. *Pawn Storm's Lack of Sophistication as a Strategy*. Dec. 2020. URL: https://www.trendmicro.com/en_us/research/20/l/pawn-storm-lack-of-sophistication-as-a-strategy.html (visited on 03/22/2021).

[40] Takahiro Haruyama. *Winnti Polymorphism*. 2016. URL: https://hitcon.org/2016/pacific/0composition/pdf/1201/1201%20R2%201610%20winnti%20polymorphism.pdf (visited on 07/13/2021).

[41] horsicq. *horsicq/Detect-It-Easy*. July 2021. URL: https://github.com/horsicq/Detect-It-Easy (visited on 07/18/2021).

[42] horsicq. *horsicq/DIE-engine*. July 2021. URL: https://github.com/horsicq/DIE-engine (visited on 07/18/2021).

[43] Andreas Hunkeler. *Collection of malware persistence information*. May 2021. URL: https://github.com/Karneades/malware-persistence (visited on 07/06/2021).

[44] Ngoc Anh Huynh, Wee Keong Ng, and Hoang Giang Do. "On periodic behavior of malware: experiments, opportunities and challenges". In: *11th International Conference on Malicious and Unwanted Software (MALWARE)*. Oct. 2016. DOI: 10.1109/MALWARE.2016.7888733.

[45] International Organization for Standardization. *ISO 8601 — Date and time format*. URL: https://www.iso.org/iso-8601-date-and-time-format.html (visited on 07/03/2021).

[46] Intezer. *Year of the Gopher*. 2021. URL: https://www.intezer.com/wp-content/uploads/2021/02/Intezer-2020-Go-Malware-Round-Up.pdf (visited on 07/01/2021).

[47] Jason Zhang and Stefano Ortolani. *HELO Winnti: Attack or Scan?* Sept. 2019. URL: https://www.lastline.com/labsblog/helo-winnti-attack-scan/ (visited on 07/25/2021).

[48] A. L. Johnson. *Flamer: Urgent Suicide*. 2012. URL: https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=b17110bd-8387-4c38-b27e-7c875ca98021&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments (visited on 06/12/2021).

[49] A. L. Johnson. *Longhorn: Tools used by cyberespionage group linked to Vault 7*. Oct. 2017. URL: https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=7ca2e331-2209-46a8-9e60-4cb83f9602de&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments (visited on 07/04/2021).

[50] Joint Task Force Transformation Initiative. *Managing information security risk: organization, mission, and information system view*. Tech. rep. NIST SP 800-39. Edition: 0. Gaithersburg, MD: National Institute of Standards and Technology, 2011. DOI: 10.6028/NIST.SP.800-39. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-39.pdf (visited on 02/15/2021).

[51] Martin Karing. *ConfuserEx 2*. URL: https://mkaring.github.io/ConfuserEx/ (visited on 07/04/2021).

[52] Kaspersky Global Research and Analysis Team. *Equation Group: from Houston with love*. 2015. URL: https://securelist.com/equation-group-from-houston-with-love/68877/ (visited on 07/08/2021).

[53] Kaspersky Global Research and Analysis Team. *Equation Group: Questions and Answers*. 2015. URL: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064459/Equation_group_questions_and_answers.pdf (visited on 03/07/2021).

[54] Kaspersky Global Research and Analysis Team. *Inside the EquationDrug Espionage Platform*. 2015. URL: https://securelist.com/inside-the-equationdrug-espionage-platform/69203/ (visited on 07/08/2021).

[55] Kaspersky Global Research and Analysis Team. *MATA: Multi-platform targeted malware framework*. July 2020. URL: https://securelist.com/mata-multi-platform-targeted-malware-framework/97746/ (visited on 07/09/2021).

[56] Kaspersky Global Research and Analysis Team. *The Duqu 2.0: Technical Details*. 2015. URL: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07205202/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf (visited on 07/06/2021).

[57] Kaspersky Global Research and Analysis Team. *The Regin Platform: Nation-State Ownage of GSM Networks*. 2014. URL: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08070305/Kaspersky_Lab_whitepaper_Regin_platform_eng.pdf (visited on 07/06/2021).

[58] Kaspersky Global Research and Analysis Team. *Winnti: More then just a game*. 2013. URL: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/20134508/winnti-more-than-just-a-game-130410.pdf (visited on 07/13/2021).

[59] Mike Kelly. *Unix Time and Windows Time*. Jan. 2009. URL: https://docs.microsoft.com/en-us/archive/blogs/mikekelly/unix-time-and-windows-time (visited on 07/03/2021).

[60] Rintaro Koike and Shota Nakajima. *An Overhead View of the Royal Road*. 2020. URL: https://github.com/nao-sec/materials/raw/master/JSAC%2BCPRCon2020/An_Overhead_View_of_the_Royal_Road.pdf (visited on 07/05/2021).

[61] Vitali Kremez. *Pro-Russian CyberSpy Gamaredon Intensifies Ukrainian Security Targeting*. Feb. 2020. URL: https://labs.sentinelone.com/pro-russian-cyberspy-gamaredon-intensifies-ukrainian-security-targeting/ (visited on 07/11/2021).

[62] Georgy Kucherin, Igor Kuznetsov, and Costin Raiu. *Sunburst backdoor – code overlaps with Kazuar*. Jan. 2021. URL: https://securelist.com/sunburst-backdoor-kazuar/99981/ (visited on 02/24/2021).

[63] Laboratory of Cryptography and System Security (CrySyS Lab). *sKyWIper (a.k.a. Flame a.k.a. Flamer): A complex malware for targeted attacks*. 2012. URL: https://www.crysys.hu/publications/files/skywiper.pdf (visited on 07/09/2021).

[64] Tom Lancaster and Esmid Idrizovic. *Paranoid PlugX*. June 2017. URL: https://unit42.paloaltonetworks.com/unit42-paranoid-plugx/ (visited on 07/10/2021).

[65] Quan Le et al. *Deep Learning at the Shallow End: Malware Classification for Non-Domain Experts*. July 2018. URL: https://www.researchgate.net/publication/323548814_Deep_Learning_at_the_Shallow_End_Malware_Classification_for_Non-Domain_Experts.

[66] Brandon Levene, Robert Falcone, and Tyler Halfpop. *Kazuar: Multiplatform Espionage Backdoor with API Access*. May 2017. URL: https://unit42.paloaltonetworks.com/unit42-kazuar-multiplatform-espionage-backdoor-api-access/ (visited on 07/09/2021).

[67] Brandon Levene, Josh Grunzweig, and Brittany Barbehenn. *Patchwork Continues to Deliver BADNEWS to the Indian Subcontinent*. Mar. 2018. URL: https://unit42.paloaltonetworks.com/unit42-patchwork-continues-deliver-badnews-indian-subcontinent/ (visited on 07/07/2021).

[68] Ruud van Luijk. *Hunting for beacons*. Jan. 2020. URL: https://blog.fox-it.com/2020/01/15/hunting-for-beacons/ (visited on 07/02/2021).

[69] Daniel Lunghi and Jaromir Horejsi. *MuddyWater Resurfaces, Uses Multi-Stage Backdoor POWERSTATS V3 and New Post-Exploitation Tools*. June 2019. URL: https://blog.trendmicro.com/trendlabs-security-intelligence/muddywater-resurfaces-uses-multi-stage-backdoor-powerstats-v3-and-new-post-exploitation-tools/ (visited on 07/11/2021).

[70] M. Maasberg, M. Ko, and N. L. Beebe. "Exploring a Systematic Approach to Malware Threat Assessment". In: *49th Hawaii International Conference on System Sciences (HICSS)*. ISSN: 1530-1605. Jan. 2016, pp. 5517–5526. DOI: 10.1109/HICSS.2016.682.

[71] Asheer Malhotra. "Calculating Malware Severity Rating using Threat Tree Analysis". Thesis. Mississippi State University, June 2020. URL: https://ir.library.msstate.edu/handle/11668/17666 (visited on 04/18/2021).

[72] Malpedia. *MoleNet (Malware Family)*. URL: https://malpedia.caad.fkie.fraunhofer.de/details/win.molenet (visited on 07/18/2021).

[73] Malpedia. *SharpStage (Malware Family)*. URL: https://malpedia.caad.fkie.fraunhofer.de/details/win.sharpstage (visited on 07/19/2021).

[74] Malpedia. *WinAPI Usage Frequencies*. URL: https://malpedia.caad.fkie.fraunhofer.de/stats/api_dll_frequencies (visited on 07/25/2021).

[75] Mandiant. *Tracking Malware with Import Hashing*. en. Jan. 2014. URL: https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html (visited on 04/10/2021).

[76] MATCODE Software. *MPRESS - Free high-performance executable packer for PE32, PE32+, .NET, MAC-OS-X*. URL: https://www.autohotkey.com/mpress/mpress_web.htm (visited on 07/09/2021).

[77] Neel Mehta, Billy Leonard, and Shane Huntley. *Peering Into the Aquarium: Analysis of a Sophisticated Multi-Stage Malware Family*. 2014. URL: https://s3.documentcloud.org/documents/3461560/Google-Aquarium-Clean.pdf (visited on 07/06/2021).

[78] Microsoft. *Microsoft 365 Documentation: Transport agents in Exchange Server*. 2021. URL: https://docs.microsoft.com/en-us/exchange/mail-flow/transport-agents/transport-agents (visited on 07/09/2021).

[79] Microsoft. *Microsoft Security Bulletin MS08-067*. 2008. URL: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2008/ms08-067 (visited on 07/13/2021).

[80] Microsoft. *Microsoft Security Bulletin MS09-025*. 2009. URL: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2009/ms09-025 (visited on 07/08/2021).

[81] Microsoft. *Microsoft Security Bulletin MS10-046*. 2010. URL: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2010/ms10-046 (visited on 07/13/2021).

[82] Microsoft. *Microsoft Security Bulletin MS10-061*. 2010. URL: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2010/ms10-061 (visited on 07/13/2021).

[83] Microsoft. *Microsoft Security Bulletin MS12-034*. 2012. URL: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2012/ms12-034 (visited on 07/08/2021).

[84] Microsoft. *Microsoft Security Bulletin MS13-081*. 2013. URL: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2013/ms13-081 (visited on 07/08/2021).

[85] Microsoft. *Microsoft Security Bulletin MS14-058*. 2014. URL: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2014/ms14-058 (visited on 07/08/2021).

[86] Microsoft. *Microsoft Security Bulletin MS14-068*. 2014. URL: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2014/ms14-068 (visited on 07/08/2021).

[87] Microsoft. *Microsoft Security Bulletin MS15-061*. 2015. URL: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2015/ms15-061 (visited on 07/08/2021).

[88] Microsoft. *Open Specifications: [MS-FSCC]: NTFS Attribute Types*. 2020. URL: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-fscc/a82e9105-2405-4e37-b2c3-28c773902d85 (visited on 07/11/2021).

[89] Microsoft. *Windows Developer Documentation: Crash Dump Analysis*. May 2018. URL: https://docs.microsoft.com/en-us/windows/win32/dxtecharts/crash-dump-analysis (visited on 07/01/2021).

[90] Microsoft. *Windows Developer Documentation: DllMain entry point*. July 2020. URL: https://docs.microsoft.com/en-us/windows/win32/dlls/dllmain (visited on 07/01/2021).

[91] Microsoft. *Windows Developer Documentation: PE Format*. Mar. 2021. URL: https://docs.microsoft.com/en-us/windows/win32/debug/pe-format (visited on 07/01/2021).

[92] Microsoft. *Windows Developer Documentation: Symbol Files - Win32 apps*. May 2018. URL: https://docs.microsoft.com/en-us/windows/win32/debug/symbol-files (visited on 07/25/2021).

[93] Steve Miller. *Definitive Dossier of Devilish Debug Details – Part One: PDB Paths and Malware*. Aug. 2019. URL: https://www.fireeye.com/blog/threat-research/2019/08/definitive-dossier-of-devilish-debug-details-part-one-pdb-paths-malware.html (visited on 03/14/2021).

[94] MITRE Corporation. *Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder, MITRE ATT&CK® Sub-technique T1547.001*. 2020. URL: https://attack.mitre.org/techniques/T1547/001/ (visited on 07/09/2021).

[95] MITRE Corporation. *Deobfuscate/Decode Files or Information, MITRE ATT&CK® Technique T1140*. 2020. URL: https://attack.mitre.org/techniques/T1140/ (visited on 07/09/2021).

[96] MITRE Corporation. *Hijack Execution Flow: DLL Search Order Hijacking, MITRE ATT&CK® Sub-technique T1574.001*. 2020. URL: https://attack.mitre.org/techniques/T1574/001/ (visited on 07/12/2021).

[97] MITRE Corporation. *MAEC 5.0 Specification - Vocabularies*. Oct. 2017. URL: https://maecproject.github.io/releases/5.0/MAEC_Vocabularies_Specification.pdf (visited on 02/18/2021).

[98] MITRE Corporation. *MAEC Overview*. 2017. URL: https://maecproject.github.io/documentation/overview/ (visited on 06/01/2021).

[99] MITRE Corporation. *Phishing: Spearphishing Attachment, MITRE ATT&CK® Sub-technique T1566.001*. 2021. URL: https://attack.mitre.org/techniques/T1566/001/ (visited on 07/10/2021).

[100] MITRE Corporation. *Process Injection: Process Hollowing, MITRE ATT&CK® Sub-technique T1055.012*. 2020. URL: https://attack.mitre.org/techniques/T1055/012/ (visited on 06/30/2021).

[101] MITRE Corporation. *Template Injection, MITRE ATT&CK® Technique T1221*. 2020. URL: https://attack.mitre.org/techniques/T1221/ (visited on 07/11/2021).

[102] MITRE Corporation. *Web Service: Dead Drop Resolver, MITRE ATT&CK® Sub-technique T1102.001*. 2020. URL: https://attack.mitre.org/techniques/T1102/001/ (visited on 07/07/2021).

[103] K. A Monnappa. *Learning Malware Analysis: Explore the Concepts, Tools, and Techniques to Analyze and Investigate Windows Malware*. 2018. ISBN: 978-1-78839-752-0.

[104] National Cyber Security Centre. *Advisory: APT29 targets COVID-19 vaccine development*. 2020. URL: https://www.ncsc.gov.uk/files/Advisory-APT29-targets-COVID-19-vaccine-development-V1-1.pdf (visited on 07/01/2021).

[105] NATO Science and Technology Organization. *Assessment and Communication of Uncertainty in Intelligence to Support Decision-Making*. 2020. URL: https://www.sto.nato.int/publications/STO%20Technical%20Reports/STO-TR-SAS-114/$$TR-SAS-114-ALL.pdf.

[106] Netresec. *PolarProxy - A transparent TLS proxy created primarily for incident responders and malware researchers*. URL: https://www.netresec.com/?page=PolarProxy (visited on 07/02/2021).

[107] Robert Neumann. *AutoCAD Malware - Computer Aided Theft*. Nov. 2018. URL: https://www.forcepoint.com/blog/x-labs/autocad-malware-computer-aided-theft (visited on 07/06/2021).

[108] NIST. *NVD - CVE-2015-2545*. 2018. URL: https://nvd.nist.gov/vuln/detail/CVE-2015-2545 (visited on 07/07/2021).

[109] NIST. *NVD - CVE-2017-0261*. 2019. URL: https://nvd.nist.gov/vuln/detail/CVE-2017-0261 (visited on 07/07/2021).

[110] Novetta. *Winnti Analysis*. 2015. URL: https://www.novetta.com/wp-content/uploads/2015/04/novetta_winntianalysis.pdf (visited on 06/30/2021).

[111] OASIS Open. *STIX Version 2.1 Specification Document*. 2021. URL: https://docs.oasis-open.org/cti/stix/v2.1/cs02/stix-v2.1-cs02.pdf (visited on 06/01/2021).

[112] Oreans. *Themida Overview*. URL: https://www.oreans.com/Themida.php (visited on 07/04/2021).

[113] Kevin O'Reilly. *What is CAPE? — CAPE Sandbox v2.1 Book*. URL: https://capev2.readthedocs.io/en/latest/introduction/what.html (visited on 07/04/2021).

[114] Oxford Learners Dictionary. *Definition "sophisticated"*. URL: https://www.oxfordlearnersdictionaries.com/us/definition/english/sophisticated (visited on 05/29/2021).

[115] Palo Alto Networks Unit42. *New BabyShark Malware Targets U.S. National Security Think Tanks*. Feb. 2019. URL: https://unit42.paloaltonetworks.com/new-babyshark-malware-targets-u-s-national-security-think-tanks/ (visited on 07/07/2021).

[116] Stephane Peter. *makeself - Make self-extractable archives on Unix*. URL: https://makeself.io/ (visited on 07/04/2021).

[117] Howard Poston. *Intercepting HTTPS traffic with Burp Suite*. June 2019. URL: https://resources.infosecinstitute.com/topic/intercepting-https-traffic-with-burp-suite/ (visited on 07/02/2021).

[118] John-Paul Power. *Celebrity vulnerabilities: A short history of bug branding*. July 2017. URL: https://medium.com/threat-intel/bug-branding-heartbleed-14ef1a64047f (visited on 07/05/2021).

[119] PwC UK. *How WellMess malware has been used to target COVID-19 vaccines*. July 2020. (Visited on 07/01/2021).

[120] PwC UK and BAE Systems. *Operation Cloud Hopper*. 2017. URL: https://www.pwc.co.uk/cyber-security/pdf/cloud-hopper-report-final-v4.pdf.

[121] PwC UK and BAE Systems. *Operation Cloud Hopper: Technical Annex*. 2017. URL: https://www.pwc.co.uk/cyber-security/pdf/cloud-hopper-annex-b-final.pdf (visited on 04/10/2021).

[122] QuoIntelligence. *WINNTI GROUP: Insights From the Past*. Apr. 2020. URL: https://quointelligence.eu/2020/04/winnti-group-insights-from-the-past/ (visited on 07/13/2021).

[123] Moritz Raabe and Willi Ballenthin. *capa: Automatically Identify Malware Capabilities*. July 2020. URL: https://www.fireeye.com/blog/threat-research/2020/07/capa-automatically-identify-malware-capabilities.html (visited on 06/02/2021).

[124] Mary-Beth Samekh. *Lessons learned from Flame, three years later*. 2015. URL: https://securelist.com/lessons-learned-from-flame-three-years-later/70149/ (visited on 07/09/2021).

[125] Secureworks. *Russian Threat Group Targets Clinton Campaign*. June 2016. URL: https://www.secureworks.com/blog/russian-threat-group-targets-clinton-campaign (visited on 07/19/2021).

[126] Adam Segal. *Wikileaks and the CIA: What's in Vault7?* Mar. 2017. URL: https://www.cfr.org/article/wikileaks-and-cia-whats-vault7 (visited on 07/04/2021).

[127] Michael Sikorski and Andrew Honig. *Practical Malware Analysis*. San Francisco: No Starch Press, Incorporated, Jan. 2012. ISBN: 978-1-59327-290-6.

[128] Florian Skopik and Timea Pahi. "Under false flag: using technical artifacts for cyber attack attribution". In: *Cybersecurity* 3.1 (Mar. 2020), p. 8. ISSN: 2523-3246. DOI: 10.1186/s42400-020-00048-4. URL: https://doi.org/10.1186/s42400-020-00048-4 (visited on 07/01/2021).

[129] Alex Sotirov. *Analyzing the MD5 collision in Flame*. 2012. URL: https://trailofbits.files.wordpress.com/2012/06/flame-md5.pdf (visited on 07/09/2021).

[130] Timo Steffens. *Attribution of Advanced Persistent Threats*. English. 1st ed. Bonn: Springer Vieweg, 2020. ISBN: 978-3-662-61312-2.

[131] Stichting Cuckoo Foundation. *Cuckoo Sandbox - Automated Malware Analysis*. URL: https://cuckoosandbox.org/ (visited on 05/31/2021).

[132] Blake E. Strom et al. *MITRE ATT&CK: Design and Philosophy*. 2020. URL: https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf (visited on 06/02/2021).

[133] Symantec. *Regin: Top-tier espionage tool enables stealthy surveillance*. 2014. URL: https://docs.broadcom.com/doc/regin-top-tier-espionage-tool-15-en (visited on 03/07/2021).

[134] Gabor Szappanos. *PlugX: The Next Generation*. 2014. URL: https://www.sophos.com/en-us/medialibrary/pdfs/technical%20papers/plugx-thenextgeneration.pdf (visited on 07/10/2021).

[135] Hakan Tanriverdi et al. *Winnti: Attacking the Heart of the German Industry*. July 2019. URL: https://web.br.de/interaktiv/winnti/english (visited on 07/16/2021).

[136] Dmitry Tarakanov. *Games are over: Winnti is now targeting pharmaceutical companies*. Jan. 2015. URL: https://securelist.com/games-are-over/70991/ (visited on 07/16/2021).

[137] Mathieu Tartare and Martin Smolár. *No "Game over" for the Winnti Group*. May 2020. URL: https://www.welivesecurity.com/2020/05/21/no-game-over-winnti-group/ (visited on 07/16/2021).

[138] Davide Testa, Luigi Martire, and Antonio Pirozzi. *Cyberwarfare: A deep dive into the latest Gamaredon Espionage Campaign*. Feb. 2020. URL: https://yoroi.company/research/cyberwarfare-a-deep-dive-into-the-latest-gamaredon-espionage-campaign/ (visited on 07/11/2021).

[139] Shusei Tomonaga. *Analysis of a Recent PlugX Variant - "P2P PlugX"*. Jan. 2015. URL: https://blogs.jpcert.or.jp/en/2015/01/analysis-of-a-r-ff05.html (visited on 07/10/2021).

[140] Aaron Turner and Fred Klassen. *Tcpreplay - Pcap editing and replaying utilities*. URL: https://tcpreplay.appneta.com/ (visited on 07/02/2021).

[141] VirusTotal. *VirusTotal Statistics*. URL: https://www.virustotal.com/gui/stats (visited on 07/22/2021).

[142] VirusTotal. *YARA - The pattern matching swiss knife for malware researchers*. URL: https://virustotal.github.io/yara/ (visited on 06/29/2021).

[143] VMware. *CB TAU Threat Intelligence Notification: Winnti Malware 4.0*. Sept. 2019. URL: https://blogs.vmware.com/security/2019/09/cb-tau-threat-intelligence-notification-winnti-malware-4-0.html (visited on 07/13/2021).

[144] Aaron Walker, Muhammad Faisal Amjad, and Shamik Sengupta. "Cuckoo's Malware Threat Scoring and Classification: Friend or Foe?" In: *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. Las Vegas, NV, USA: IEEE, Jan. 2019, pp. 0678–0684. ISBN: 978-1-72810-554-3. DOI: 10.1109/CCWC.2019.8666454. URL: https://ieeexplore.ieee.org/document/8666454/ (visited on 04/19/2021).

[145] Lars A. Wallenborn. *API-Hashing in the Sodinokibi/REvil Ransomware – Why and How? – nullteilerfrei*. Nov. 2019. URL: https://blag.nullteilerfrei.de/2019/11/09/api-hashing-why-and-how/ (visited on 07/05/2021).

[146] Wikileaks. *CIA Organization Chart*. 2017. URL: https://wikileaks.org/ciav7p1/files/org-chart.png (visited on 05/31/2021).

[147] Wikileaks. *Development Tradecraft DOs and DON'Ts*. Mar. 2017. URL: https://wikileaks.org/ciav7p1/cms/page_14587109.html (visited on 02/16/2021).

[148] Wikileaks. *Vault7 Press Release*. 2017. URL: https://wikileaks.org/ciav7p1/ (visited on 05/31/2021).

[149] Wikileaks. *What did Equation do wrong, and how can we avoid doing the same?* Mar. 2017. URL: https://wikileaks.org/ciav7p1/cms/page_14588809.html (visited on 07/04/2021).

[150] Wireshark Foundation. *Wireshark website*. URL: https://www.wireshark.org/ (visited on 07/05/2021).