J Comput Virol Hack Tech (2017) 13:47–85
DOI 10.1007/s11416-016-0273-3

ORIGINAL PAPER

Semantics-aware detection of targeted attacks: a survey

Robert Luh[1,2] · Stefan Marschalek[1] · Manfred Kaiser[1] · Helge Janicke[2] · Sebastian Schrittwieser[1]

Received: 27 October 2015 / Accepted: 7 April 2016 / Published online: 2 May 2016

Abstract In today's interconnected digital world, targeted attacks have become a serious threat to conventional computer systems and critical infrastructure alike. Many researchers contribute to the fight against network intrusions or malicious software by proposing novel detection systems or analysis methods. However, few of these solutions have a particular focus on Advanced Persistent Threats or similarly sophisticated multi-stage attacks. This turns finding domain-appropriate methodologies or developing new approaches into a major research challenge. To overcome these obstacles, we present a structured review of semantics-aware works that have a high potential for contributing to the analysis or detection of targeted attacks. We introduce a detailed literature evaluation schema in addition to a highly granular model for article categorization. Out of 123 identified papers, 60 were found to be relevant in the context of this study. The selected articles are comprehensively reviewed and assessed in accordance to Kitchenham's guidelines for systematic literature reviews. In conclusion, we combine new insights and the status quo of current research into the concept of an ideal systemic approach capable of semantically processing and evaluating information from different observation points.

Keywords Security · Targeted attacks · APT · Malware · Intrusion detection · Semantics-based attack detection

✉ Robert Luh
robert.luh@fhstp.ac.at

1 Josef Ressel Center TARGET, St. Pölten University of Applied Sciences, Matthias-Corvinus St. 15, St. Pölten 3100, Austria

2 Faculty of Technology, De Montfort University, Gateway House 5.37, Leicester LE1 9BH, UK

1 Introduction

IT infrastructures and corporate networks are threatened by a plethora of different attacks. Not long ago, research and industry almost entirely focused on the detection and prevention of widespread malware attacks without a specific target. Signature-based detection techniques have been the de facto standard against this kind of threat throughout the past 30 years and current virus scanners still rely primarily on malware signatures for detection. The fundamental idea behind these techniques is founded on the assumption that one malicious campaign targets thousands or even millions of hosts. Once the payload or carrier has been found on one system, a generic signature or behavior pattern of the threat can be extracted and used on other systems for detection.

In recent years, however, a new generation of attack has emerged. Advanced Persistent Threats (APTs) or Advanced Targeted Attacks (ATAs) can be characterized as tailored to one specific entity. These types of attacks are driven by different motivations and often cause significantly more damage than bulk attacks; often they are performed for espionage or sabotage reasons and are orchestrated by experts. Several cases in recent history have shown that targeted attacks sometimes operate undiscovered by their victims for many months or even years [37,59,63,83,141]. The prime example, Stuxnet, which targeted programmable logic controllers (PLCs) of sensitive industrial systems, was active for at least 3 years until discovery. It utilized four zero-day exploits to achieve its ultimate goal [134]. According to a Symantec study [47], Stuxnet infected close to 100,000 systems across 115 countries. Its quasi successor, Duqu, also targeted industrial control systems (ICS) but was used only for information gathering [78]. Systems of 10 organizations in at least eight countries were reportedly affected [25].

On the espionage side, the Regin trojan is believed to have been used for global, systematic campaigns since at least 2008 [139]. Other examples include Flame [79], Mahdi [130], and Gauss [80]. These strains are currently used for cyber-espionage in Middle Eastern countries and, depending on the variant, are capable of stealing passwords and cookies, recording network traffic, keystrokes, microphone audio, and even entire Sykpe conversations [103].

ATAs and APTs are increasingly affecting less prominent targets as well. In 2013 alone, "economic espionage and theft of trade secrets cost the American economy more than $19 billion. Over the past 4 fiscal years, the number of arrests related to economic espionage and theft of trade secrets overseen by the FBI's Economic Espionage Unit has almost doubled, indictments have more than tripled, and convictions have increased sixfold. Halfway through fiscal year 2013, the number of open investigations is running more than 30 % above the total from 4 years ago" [112].

Modern cyber-threats are no longer limited to a single malware executable but often comprise targeted, multi-stage attacks that are difficult to spot using only file- and signature-based malware detection systems. The reason why these more conventional detection methods are less effective against ATAs is rooted in the fact that targeted attacks are tailor-made to the organization they seek to penetrate: Binary patterns of the responsible malware are unlikely to exist at the time of attack. Anti-virus (AV) products are effective in the defense against known exploit carriers or ill-considered user actions but struggle with hitherto unknown malware [42]. The rise of state-sponsored attacks poses another serious challenge as high-moneyed actors are able to invest significantly more time and effort into designing and developing hard-to-detect malware. Furthermore, government bodies may compel AV vendors to whitelist their respective espionage tools.

This makes it necessary to explore novel techniques for tactical threat intelligence and malicious activity detection on multiple layers, allowing for a defense in depth approach to malware detection through the use of multiple observation points deployed throughout the infrastructure.

This paper provides a systematic overview of four layers of tactical and operational threat intelligence and the various solutions current research offers: *host-based* solutions, *network-based* solutions, *multi-source* solutions, and *general* or *supporting* solutions. The main goal is to present a comprehensive review of proposed approaches that directly or indirectly facilitate target attack detection or analysis through the information they collect. We identify domains and methodologies that can be used in a holistic APT/ATA detection framework. A special focus lies on semantics-aware approaches that are key to distinguishing common from advanced threats, which are well recognized as defeating traditional signature-based detection. The sophisticated and multipartite nature of modern cyber-attacks is one of the main reasons why a defender needs to consider host-based, network-based and hybrid monitoring tools when it comes to capturing suspicious events that can later be analyzed and interpreted. While there are numerous solutions that focus on specific attack aspects such as malware infection or network intrusion, only few consider the greater picture. In order to truly understand a targeted attack, it becomes necessary to not only focus on malicious software or intrusion detection but to use every suitable tool at one's disposal. This paper evaluates existing research retrieved from six academic search engines using an incremental search and refinement process utilized with a view to integrating the results into a holistic framework for APT detection and assessment. Existing models and technologies were methodically categorized by their capabilities, purpose as well as several operational parameters with the aim to establish their applicability to a general concept of an ATA defense framework presented as part of the concluding discussion.

The contribution of this paper includes:

- Concept of an ATA detection framework and introduction of system design checklist specifically tailored to the task of identifying targeted attacks;
- Schema for review and categorization of intrusion detection, analysis, correlation, and threat intelligence domain literature and tools;
- Identification of semantics-aware solutions that could help distinguish common from advanced attacks;
- Comprehensive review of host-based, network-based, and multi-source data providers and analysis approaches that, if utilized accordingly, could contribute to targeted attack detection.

The remainder of the paper is structured as follows: In Sect. 2 we introduce APTs and ATAs, define frequent terms, and provide some background on models and use cases. Section 3 specifies the research questions, review method, and various literature collection criteria. In Sect. 4 we introduce the categorization schema used in the review process. Section 5 provides specific background and reviews the selected papers. Section 6 contains comprehensive comparison charts and breaks down the results. In Sect. 7, we discuss our findings and present the concept of a defense framework that encompasses most APT attack phases.

2 APT taxonomy and semantics

2.1 Modeling advanced persistent threats

APTs are not necessarily limited to a tailored piece of malware, but often encompass various stages with their own respective methods [67,93]. For example, spearfishing and

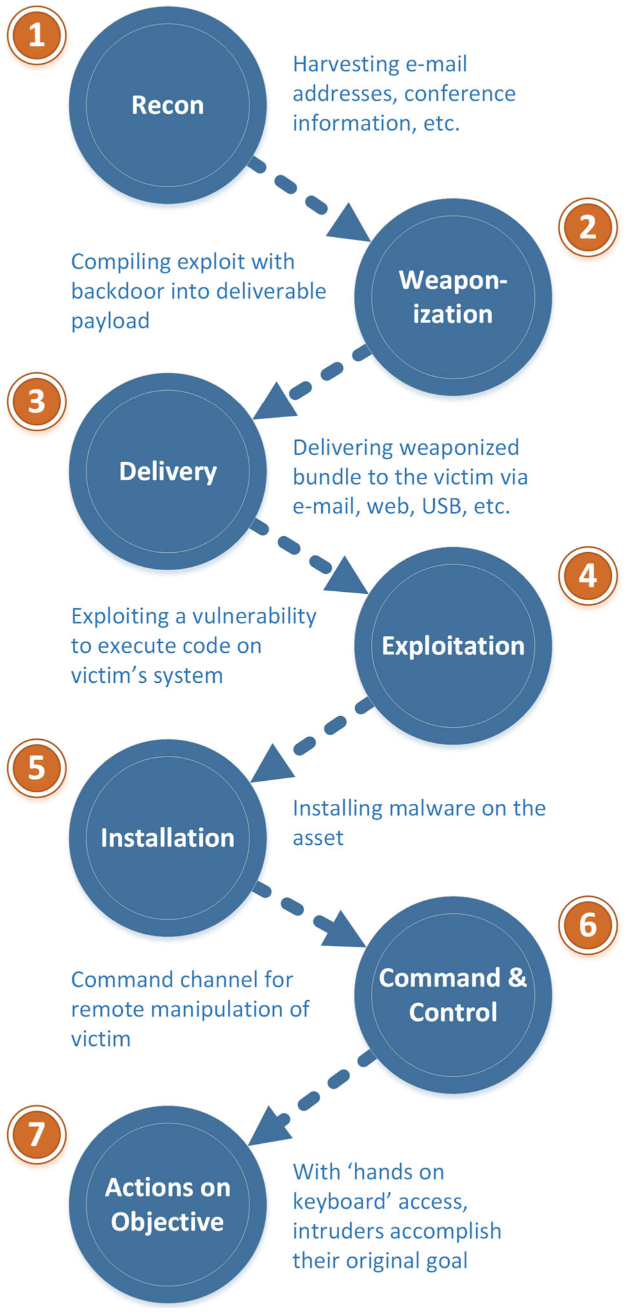

1
Recon
Harvesting e-mail addresses, conference information, etc.
2
Weapon-ization
Compiling exploit with backdoor into deliverable payload
3
Delivery
Delivering weaponized bundle to the victim via e-mail, web, USB, etc.
4
Exploitation
Exploiting a vulnerability to execute code on victim's system
5
Installation
Installing malware on the asset
6
Command & Control
Command channel for remote manipulation of victim
7
Actions on Objective
With 'hands on keyboard' access, intruders accomplish their original goal

Fig. 1 APT phases as defined by [67]

malicious websites are common approaches to delivering a malicious payload to a target system [94,134]. RATs embedded in a Trojan horse are then often used to take control of a machine. In addition, targeted attacks reportedly employ zero-day vulnerabilities [18,134].

In order to better understand the complex nature of targeted attacks and to map specific exploits to a broader goal, APTs are often modeled as a multi-stage process. Hutchins et al. [67] expand on the military concept of target engagement and define seven typical phases, as seen in Fig. 1: *Reconnaissance*, such as network scans, mapping procedures, employee profiling, and search for suitable zero-day exploits, *weaponization*, i.e. the development of targeted malware and the set-up of malicious services, *delivery* via various channels, *exploitation*, such as the activation of malware and the use of the previously weaponized payload to exploit a vulnerability, *installation*, i.e. establishing a persistent presence, *command and control* communication, and *actions on objective*, which include the primary malicious task as well as exfiltration and clean-up activities.

Giuara and Wang [56] as well as De Vries et al. [37] use similar kill chain models to describe APTs – only the naming and some stage boundaries differs. Independent of taxonomy, APTs and ATAs are complex attacks that, unlike bulk malware or fire-and-forget network attacks, carefully consider the target's system environment, security measures, and assets.

Adversary actions in each of the phases are possibly countered by a number of defense mechanisms. For example, the installation on the target system might be detected by a dedicated host-based intrusion detection system (HIDS). Each of the models helps to plan network defense measures as well as illustrates some data providers that are promising to use in an attack detection system. This is the reason why we decided to include solutions from different domains instead of just malware detection tools.

2.2 Semantics-aware and semantics-based approaches

The semantics of an attack is an important factor for its identification and analysis. Generally speaking, semantics is the study of meaning. It is widely used in both technical and non-technical fields such as linguistics, philosophy, and information theory. When applied to computer science, it is often synonymous with determining execution paths of a program. In that case, the meaning of an artificial programming language as opposed to a natural language is evaluated. The mathematical model of computation is defined using one of the following techniques [32,158]:

Denotational semantics describes the formalization of programming languages into mathematical objects. This includes the translation of a phrase into another, strictly formal language. *Operational semantics* encompasses the description of the execution and its correctness by e.g. describing machine state transitions. *Axiomatic semantics* describes meaning and proves correctness through rules of inference that map input to output properties. It is important to note that these strands are highly dependent on each other and are often used in concert.

In this paper we differentiate semantics-aware and semantics-based approaches. We define *semantics-aware* solutions as considerate of the goal, means, and specifics of an attack. The overall premise can be summarized as the analysis of meaning of an attack. Based on this central aspect, we compiled a list of prerequisites that need to be met in order for a defense measure to be considered semantics-aware:

- The solution needs to counteract specific objectives or the overall goal of the attacker by e.g. detecting or preventing an attack or by helping to determine what the adversary wants to achieve (see primary category G in Sect. 4 for more information);
- The solution has to investigate, analyze, detect, or extract the technical means (i.e. techniques) of an attack or attack action through e.g. anomalies or patterns in behavior or code (see primary categories D and A in Sect. 4);
- The techniques, goals, or methods investigated can be mapped to one of the seven APT phases depicted in Fig. 1 (primary category APT).

Semantics-based approaches usually revolve around four key topics we identified as representative for the semantic domain in general: activity context, ontology design, data abstraction, and correlation. As works in these fields are often more universal in nature, the above restrictions do not strictly apply. Instead, we opted to include security solutions that facilitate the development of models, languages and formal definitions or that determine the correctness of information through rules of e.g. inference. We argue that they, while perhaps not directly applicable to a particular scenario, are sufficiently adaptable to benefit the defense against advanced cyber-attacks.

Unlike semantics-aware approaches, the semantics-based category closely adheres to the type definitions introduced at the beginning of this section. For the literature review in Sect. 5, we categorized each paper in accordance to above definitions: Semantics-aware, semantics-based: denotational, operational, or axiomatic.

3 Review method

This study is based on the guidelines for systematic literature reviews by Barbara Kitchenham [82]. The main goal of this paper is to systematically review available literature on the topic of host-based, network-based, and multi-source detection of cyber-attacks with a focus on semantics-aware and semantics-based approaches. The surveyed papers are evaluated by their applicability to the domain of targeted attacks.

3.1 Research questions

Specifically, there are a number of research questions addressed by this paper:

1. Which models, frameworks, formal definitions, and tools exist to describe information system attacks?
2. Which semantics-aware and semantics-based tools and techniques exist to detect and evaluate such attacks?
3. What are promising approaches to ATA detection and how can they be classified?
4. What kind of information is required to identify targeted attacks and how could it help to get a more complete picture of an incident?

To address R1 and R2, we have conducted a search of several scientific databases. This process is detailed in the next subsection. To satisfy R3, we came up with a list of criteria and categories to classify each identified solution. These criteria were combined into structured overview tables that are thoroughly explored in Sect. 6. In response to R4, Sect. 7 conceptually combines the benefits of solutions from several review categories and sketches an ideal defense framework.

3.2 Search process

The search process included a manual keyword search on IEEE, ACM, Scopus, Web of Science, Academic Search Premier and Google Scholar. In the first stage we focused on the following keywords and keyword combinations: "targeted attack {detection, identification, recognition, analysis}"; "{APT, advanced persistent threat, threat, cyber-threat} {detection, identification, recognition}"; "semantic{s}{-}{based} {malware, intrusion} detection"; "SIEM" + "APT" as well as further combinations including the "model" or "framework" suffix.

For data providers, we disregarded all publications released prior to 2003. General papers (e.g. original publications on key topics) were not filtered by date. These initial searches yielded a total of 112 papers. Every publication was perused and summarized.

In stage 2, we assessed and selected all additional references used in the grant proposal document of the corresponding ATA research project[1] at St. Pölten UAS. Stage 2 increased the number of considered papers to 114.

Lastly, we extracted a sample set of relevant references from papers we identified as key publications. This increased the total number of considered works to 123. Below exclusion criteria and quality assessment ultimately brought this number down to 60.

3.3 Inclusion and exclusion criteria

We included all peer-reviewed articles released between 2003 and 2015. We specifically looked for publications that matched the following criteria:

[1] Josef Ressel Center for Unified Threat Intelligence on Targeted Attacks.

- Frameworks, models, methods, formal descriptions and tools of host-based, network-based, and multi-source approaches to data collection and evaluation;
- Papers that explicitly focused on the detection, identification, recognition, and analysis of (targeted) attacks.

Product specification sheets, magazine articles and gray literature were excluded from our search. In addition, we removed the following material:

- Articles which focused excessively on visualization, code manipulation, honeypot technology, and social engineering;
- Papers about data stream processing techniques;
- Marketing texts and general descriptions of commercial malware analysis suites;
- Well-established industry solutions already discussed in dedicated survey papers [45,154];
- Patents, as they typically lack in replicability and presentation.

3.4 Quality assessment

The quality assessment of the reviewed articles is based on a number of questions related to both overall soundness and ATA detection suitability of the presented solutions. Each quality assessment factor Qn was graded from 0 to +1. The composition of the score is detailed below:

1. Were the research questions and the overall goal clearly defined?
   - Research question exists (+0.5)
   - The overall objective and motivation is stated in accordance with the goals identified in Sect. 4.1 and Table 4 (+0.5)
2. Was the attack detection domain introduced and explained?
   - An attack domain (see ’Domain’ in Sect. 4.1 and Table 4) can be clearly assigned (+0.5)
   - A practical application is exemplified (+0.5)
3. Was the approach presented in a clear and replicable manner?
   - Starting point and core process are described (+0.5)
   - Evaluation exists (+0.5)
4. Did the article explain the nature and type of operational data used in the process?
   - Base (origin) data and its representation are stated (+0.5)
   - Result (outcome) data and its representation are stated (+0.5)
5. Can the proposed solution be applied to the detection of targeted attacks?

Specifically, a paper was awarded 1 point for a category when: the research question, motivation and goals were clearly defined (Q1), the domain was well specified (Q2), the approach to solving the stated problem was presented in detail (Q3), the specifics of the input and output data were provided (Q4), and the solution is applicable to the field of targeted attacks or one of its primary aspects (Q5). Since Q5 cannot be objectively computed by evaluating definitive indicators, the score was determined in the course of an expert discussion among at least 4 security researchers per paper.

Complementing the QA score, a domain expert rating ranging from 0 to 5 was added following the initial assessment. This score is the result of a group discussion for each paper (minimum of 3 researchers, excluding the primary author) and mirrors their view on the paper independent from formal criteria. To remove any bias, this second grade was awarded without prior knowledge of the QA score.

In addition to the exclusion criteria, we used the quality assessment to reduce the number of papers ultimately included in the survey. Each article with an overall QA + expert score below 4.5 as well as papers with a Q5 score of zero were removed from the list.

3.5 Data collection

The following data was extracted from each reviewed paper:

- The type of source (journal, conference, book) and full reference;
- Classification of general type of article (model, method, framework, formal definition, or tool);
- Classification of information system domain (host, network, multi-source, or general semantics);
- Classification of data gathering, analysis/detection, and learning techniques used;
- Summary including tags identifying core topics and goals;
- Quality evaluation in regards to the possible contribution to the detection of malicious, ATA-related activities or patterns.

Data extraction and initial checking was done by four security researchers.

3.6 Data analysis

The extracted data was analyzed and put into tables in order to provide a quantitative overview:

- Total number of papers and papers per year, addressing R1;
- Number of papers per type of article and domain, addressing R1 and R2;
- Tags describing the key factors of the respective solution appended to each paper, addressing R2 and R3;
- Quality score of all papers as well as an ATA detection applicability rating, addressing R3 as well as Q1 through Q5;
- Full table containing used technology, specific approach, technical properties and capabilities of the solutions introduced in the reviewed articles, addressing R4.

All tables as well as the statistical breakdown of the results can be found in Sect. 6.

4 Review categories

We developed a versatile schema with 4 distinct categories to classify the surveyed articles. Although similar in part to Jacob et al.'s [71] behavioral malware detection taxonomy, our schema included additional properties relevant to APT detection and analysis, e.g. specific input types and general goals that aren't usually seen in malware-centric approaches. Figure 2 presents an overview of the properties investigated. The schema can be freely applied to all topical literature and will help categorize solution capabilities ranging from data collection and analysis to automated learning.

Primary categories, identified by an asterisk, are used for synthesizing the assessed papers and represent tags assigned to each solution. These tags include the general goal G of the solution, the type of threat T, data input type I, detection method D, and analysis technique A. Knowledge generation K is demarcated as either true or false and mentions the solution's learning and classification capabilities, where applicable. The type of semantic affinity S (see Sect. 2.2) is identified as well. This is complemented by the mapping of the technique to one or more of the APT categories (including a brief rationale) by Hutchins et al. [67] (see Sect. 2), thereby satisfying all the semantics-side selection criteria previously defined.

For example, a paper describing a malware detection and behavioral analysis solution based on system events and including a methodology to semantically describe correct sample execution might be tagged as ⋄*G{detection, analysis}; T{malware}; I{event traces}; D{pattern}; A{beha-*

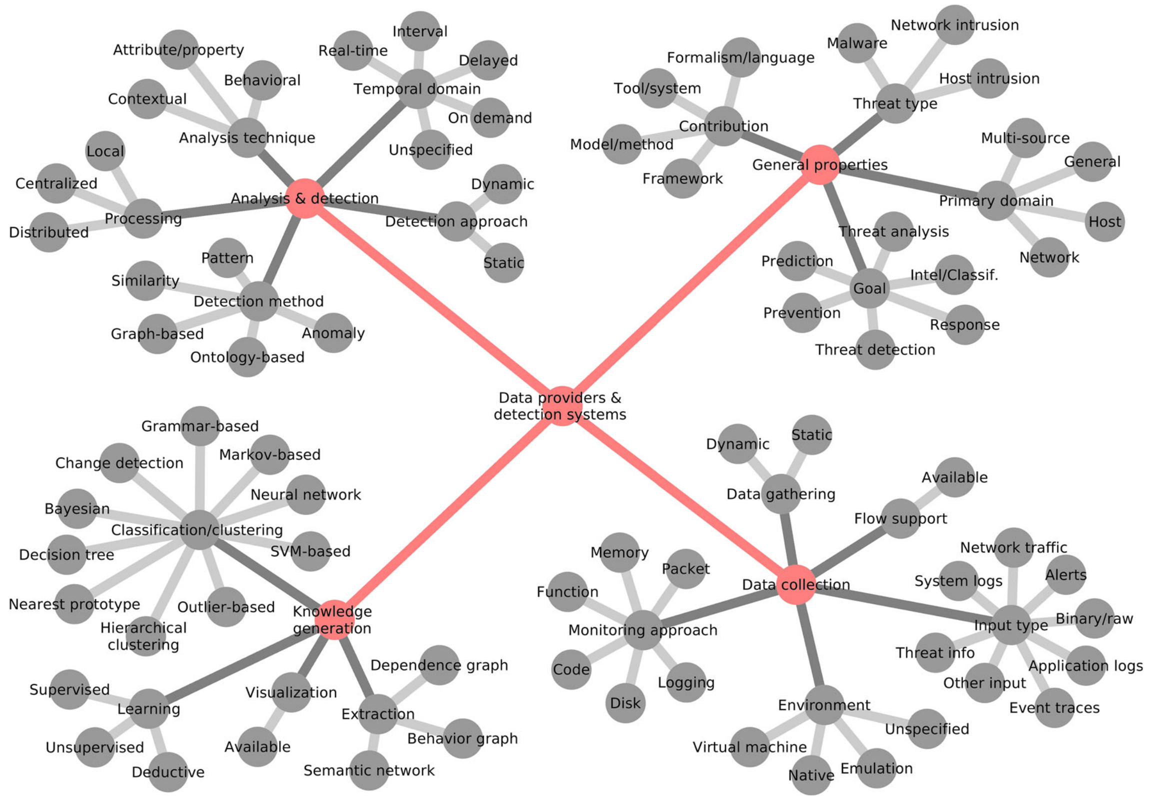


Fig. 2 Categorization of surveyed papers

vioral}; K{no}; S{operational}; APT{4,5,7: host activity}. Such a solution would be best suited to detect local (host) events that are part of APT stage 4 (exploitation), 5 (installation), or 7 (action on target).

A full breakdown of both primary and additional properties can be found in Tables 4, 5, 6, and 7.

4.1 General properties

The 'General' category encompasses high-level objectives and type definitions.

- *Primary domain* – Each paper was assigned a single domain that most closely matches the solution's area of application. Domains follow the subchapters of Sect. 5 and include host, network, multi-source, and general/unspecified solutions.
- *Contribution* – This describes the type of solution introduced in the respective paper. Frameworks, models/methods, formalisms/languages, and tools or systems are possible types of contribution. One solution may fit several categories.
- *Goal $G*$* – This criterion defines the general goal the authors want to achieve. Typically, this can encompass threat prediction, prevention, threat intelligence/classification, threat correlation/event fusion, threat detection, threat analysis, and threat response. The introduced approaches usually have more than one goal.
- *Threat type $T*$* – The threat type describes the general attack the introduced solution attempts to combat. We have separated threat types into malicious code (malware), host intrusion, and network intrusion. A system can counter a specific or several types of threat.

4.2 Data collection

Here, we categorize input data specifics, data gathering, and monitoring techniques.

- *Data gathering* – Distinguishes between static and dynamic data gathering. Data gathering is concerned primarily with the method employed to collect the information from a system or application. Detection of suspicious activity or code is part of the detection approach specified below. A combination of static and dynamic data gathering is unlikely, but not impossible.
- *Monitoring approach* – There are a number of ways to dynamically monitor execution information. Among them is function monitoring, packet monitoring, code monitoring, disk monitoring, memory monitoring, and conventional logging. Specialization on one approach is likely, but not universal.
- *Data input type $I*$* – Defines the kind of input information processed by the respective solution. This may include general threat information, system logs, application logs, network traffic, traces of system events such as API and system calls, binary code or raw data, as well as alerts generated by distributed agents or third-party software like an IDS. Many solutions support several input types.
- *Flow functionality* – Solutions with the capability to record data flows are identified here. This often includes, but is not limited to, network flow-based solutions.
- *Environment* – Depending on the system environment the respective solution can be executed on, we categorize it as running natively – or 'on-device' in case of physical appliances – or as running inside a virtualized environment such as a VM or emulator. Unspecified or environment-neutral solutions are identified as well. Tools can support or utilize several environments.

4.3 Analysis and detection

In this category, we take a look at how suspicious activity or code is recognized and analyzed.

- *Detection approach* – Similar to data gathering, the detection of relevant information can either happen statically or dynamically. This may in some cases differ from data gathering when the process of collection and detection is realized separately. For example, data may be dynamically recorded and subsequently undergo static analysis.
- *Detection method $D*$* – This incorporates general detection methods such as anomaly detection, pattern/misuse detection, similarity detection (e.g. string similarity), graph matching, or ontology-based methodologies. Systems such as pure correlation solutions do not necessarily utilize a particular detection method. Several methods may be employed.
- *Analysis technique $A*$* – Including but not limited to malware, the analysis of data and, by extension, the decision about its relevant properties, can usually happen based on pre-defined attributes (properties, states), through behavioral analysis techniques, or at a contextual (semantic) level. In most cases, solutions use only one approach.
- *Temporal domain* – Depending on their implementation, detection or analysis solutions can offer real-time, delayed but continuous, fixed-interval, or on-demand processing initiated by user command or upon detection of a certain event.
- *Processing* – Data processing itself can be performed locally on the system the data is stored/collected, in a centralized fashion on e.g. a dedicated server, or distributed across several nodes.

4.4 Knowledge generation *K**

A number of solutions offer mechanisms to generate knowledge from the collected and analyzed data.

- *Learning* – (Machine) learning capabilities are identified here. Supervised learning uses available malicious and benign data to determine the difference to a baseline, unsupervised learning tasks the system to determine the deviation by itself, and deductive learning uses inference to draw conclusions based on known premises.
- *Classification and clustering* – This category lists all the identified techniques used to classify or cluster information. We consider support vector machines (SVM), decision trees, neural networks, methods based on nearest-neighbor or nearest-prototype determination, Bayesian techniques and respective statistical, 'belief'-based systems, Markov models or other 'memoryless' statistical models, grammars realized through e.g. grammar parsing, outlier detection, change detection such as state comparison, and hierarchical clustering.
- *Extraction* – In some cases, data is merely extracted for later analysis or visualization. Techniques found in the reviewed papers include dependence graphs, behavior graphs, and semantic (link) networks.
- *Visualization* – As a by-product of knowledge generation, the results of earlier assessment stages can be graphically presented. Solutions offering visualization are identified here. See [154] for further details about security-related visualization systems.

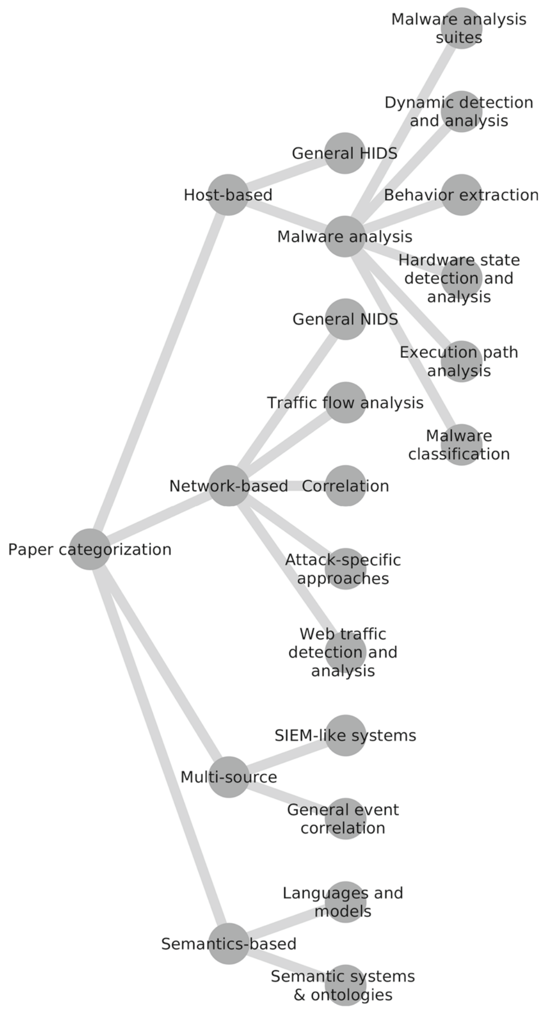


Fig. 3 Paper categorization

5 Review

In this section we assess and summarize all the papers found through the structured search process specified above. Solutions are categorized into host-based, network-based, multi-source, and purely semantic approaches that cannot be attributed to a specific domain. Figure 3 offers an overview of the full categorization used in below subsections.

The beginning of each category section provides some additional background relevant to the respective category. Tags are used to identify primary categories for each paper, providing information about the assessed solution's general approach to data gathering, monitoring, detection and analysis, as well as knowledge generation.

5.1 Host domain

Host-based solutions can be understood as detection systems running on the endpoint. We identified memory-based approaches, numerous behavioral detection and analysis systems, function call monitoring solutions, host-based intrusion detection systems, and more. For many of the tools, *malicious software* (malware) is something of a common denominator:

Most cyber-attacks involve malware smuggled onto the system to perform its sinister deed. Malware can generally be defined as "any software that does something that causes harm to a user, computer, or network" [133]. Examples include viruses, Trojan horses, backdoors, worms, rootkits, scareware, or spyware. Malicious software is known to exploit vulnerabilities of the system it is designed to run on. Flaws in applications can serve as drop vector and may be exploited as part of a privilege escalation routine required for administrative tasks. In the case of ATAs, this often includes hitherto unknown exploits known as zero-days.

Common malware i.e. aims at financial gain through fraud or blackmail; it is delivered to a large number of recipients in hope that a sufficient number of people unknowingly install it on their machines. Malware used for targeted attacks is much more sophisticated and typically includes additional components for e.g. long-term persistent installation and more complex command and control communication than can be found in other malicious software. The main distinguishing factor is its tailoring to a specific environment, however. APT malware is designed to attack a particular device, operating system, or specific application version. This influences the choice of dropping technique, evasion routines, and attacks against specific defensive measures employed by the victim. While this fact makes such software dangerous to only a limited number of systems, the damage caused by targeted malware can be much more severe.

Malware analysis is probably the most widely used and arguably most important class of host-based approaches. Solutions range from analysis suites built to determine a sample's general maliciousness to interpretation of behavior or clustering of potential malware into families. Egele et al. [45] and [68] offer an overview of dynamic malware analysis tools and describe various monitoring and detection techniques. In [154], Wagner et al. expanded on some of the tools and assessed the information they provide. There is a multitude of information to be gleaned from various malware analysis techniques, each offering specific insight into the nature and functionality of a malicious program. This includes the virus definition, packer information like packer designations and general compression information about the sample, file and header information and code sections, library and function imports, CPU instructions and their associated assembly operations, system and API calls executed by the sample, file system operations indicating the creation, modification, and deletion of files, as well as interpreted registry, process, thread, and network operations.

A large number of the data providers surveyed below focus on the detection or analysis of malware. We generally define malware data providers as tools that utilize static or dynamic analysis methods as well as signature- and behavior based detection techniques to gather information about a potentially malicious piece of software [154]:

Static analysis describes techniques that do not require the sample under scrutiny to be actually executed. Depending on the depth of analysis, a file may be checked for its basic properties like file type, checksum, easily extractable information such as null-terminated strings or DLL import information, or be fully disassembled [81]. The analysis environment – bare metal, virtual machine, or emulation – plays a negligible role for static analyses – the analyst simply chooses a platform compatible with the tools of her choice. *Dynamic analysis* goes a step further and executes the file on a dedicated host system. Various tools then monitor the execution and log relevant information into an execution trace. This ranges from simple file system operations to a full instruction list captured through a debugger. The analysis environment is essential for the dynamic approach since the type of data logged depends on both the platform as well as on the techniques used to capture system events.

On the detection side we differentiate between signature-based and behavior-based techniques:

Signature-based approaches are best known for their prominent role in antivirus software and traditional intrusion detection systems. A so-called definition or signature is created to describe an entire file or parts of the code that are known to be malicious [154]. The detection software then compares the appearance of a file or packet to this set of known signatures. Signature-based detection has several shortcomings [42]:

Firstly, obfuscation techniques commonly utilize polymorphic or metamorphic mutation to generate an ever-growing number of malware variants that are different in appearance, but functionally identical. This leads to bloated signature databases and, ultimately, to an overall slowdown of the detection process. Secondly, signature-based techniques only detect malware which has already been identified and analyzed; new species or hitherto unknown variants are often overlooked.

Behavior-based techniques, on the other hand, focus on specific system activities or software behavior typically captured through dynamic analysis. Malicious actions are defined through patterns or behavioral anomalies. Since the behavior-based approach can be semantics-aware, it is largely immune to obfuscation [42]. Its performance is limited, however: While signature matching takes but the fraction of a second, dynamic execution and trace analysis can take several minutes.

The second big area of host-based threat mitigation is *intrusion detection*. IT systems intrusion describes the act of accessing a local or network-based resource without proper authorization. It can basically be defined as computer break-in leaving the targeted system vulnerable to theft or sabotage [87]. In many cases the line between black-hat hacking and malicious software becomes blurred [77]. Most typical intrusions involve malware or tools designed to circumvent security measures of the target system. The attack procedure varies from case to case and can be considered 'targeted'; few intruders act without knowing which system they are attacking.

On the defense side, intrusion detection is primarily concerned with activities deemed illegal by the system operator [87]. These may include digital breaking and entering but generally encompasses all 'malicious actions' aimed at e.g. privilege acquisition or resource abuse. Literature differentiates two classes of intrusion detection systems (IDS): *network-based* and *host-based* [153]. The main differences are the type and number of sources used to detect adversary activity; network-based IDS analyze the traffic transmitted between systems and typically utilize a number of sensors distributed throughout the network. Host-based systems, on the other hand, attempt to discover attacks taking place on the very machine the sensor component is installed on.

There are two approaches to detecting malicious activity through an IDS: Anomaly detection and misuse (pattern) detection [87]. *Anomaly detection* is based on the premise that illegal activity manifests as abnormality and that it can be identified through measuring the variance of certain key metrics. This may include the excessive use of system functions, high resource utilization at unusual times, or other behavior deviating from a defined baseline.

Misuse detection, on the other hand, is based on predefined patterns. Knowledge of an attacker's methods or the expected consequences of an attack are encoded into behavior sequences that can be found by an IDS looking for their occurrence [87]. Examples include sequences of suspicious function calls, certain network packet payloads, or the exploitation of known bugs.

Many of the reviewed papers describe various host-based approaches to locate, identify, categorize, or analyze malicious software. Both semantics-aware as well as a few semantics-based techniques are used. Most focus on determining the maliciousness of a piece of software or attempt to classify a set of samples.

There are also various data providers that fall under the wider definition of an intrusion detection system. Since IDSs can be both host- and network-based, tools range from local process or system call tracers to traffic anomaly detectors utilizing various statistical methods or data mining approaches. Network-based systems are reviewed in Sect. 5.2.

5.1.1 Malware analysis solutions

Inspired by [71], we categorized solutions into malware analysis suites, dynamic and static detection and analysis solutions, hardware state detection and analysis, execution path analysis, behavior extraction systems, as well as malware classification systems.

While *malware analysis suites* are mostly established commercial solutions and thereby not included as per the criteria defined in Sect. 3, they are an important class of data providers that need to be mentioned. Malware suites do not per se run on the endpoint: Tools like Anubis/LastLine Analyst [14,70], Cuckoo Sandbox [35], CWSandbox/Threat Analyzer [142,157], Joe Sandbox [75], and FireEye MAS [50] run in their own, usually virtualized environment. They employ function hooking and kernel mode drivers to record and report system and/or API calls executed by the sample under scrutiny. The recorded activity is then interpreted and returned as human-readable summary. Malware analysis suites are a good starting point for general dynamic analysis and are undoubtedly the inspiration for some of the solutions introduced below. See [45] and [154] for more information. ⋄*Representative tags: G{detection, analysis}; T{malware}; I{event traces, network traffic}; D{pattern}; A{behavioral}; K{no}; S{aware}; APT{3-7: host and network activity}.*

Dynamic malware detection and analysis is covered by a multitude of articles: Grégio et al. [60] introduce BehEMOT, a non-intrusive malware behavior analysis system. It uses both a native and emulated system environment that promises to circumvent sandbox detection. Functions and their arguments as well as network events are captured using system call interception via System Service Dispatch Table (SSDT) hooking: The SSDT is a list of memory addresses that each correspond to a system call. If hooked, it becomes possible to redirect calls to an altered copy of the function; a technique useful for monitoring activity or modifying results. Similar to commercial suites like Anubis [70] or Joe Sandbox [75], BehEMOT abstracts specific calls to a general operation type (e.g. 'open process'). ⋄*Tags: G{detection, analysis}; T{malware}; I{event traces, network traffic}; D{pattern}; A{behavioral}; K{no}; S{aware}; APT{3-7: host and network activity}.*

Fukushima et al. [53] developed another behavior-based detection approach: Suspicious process behavior is identified through specific system and temporary directories accessed by the sample as well as the creation of certain registry keys responsible for e.g. automated startup. Registration or deletion of uninstall information is considered as well. Fukushima's system utilizes Procmon [126] as primary data provider. Unlike BehEMOT's post-monitoring abstraction [60], Procmon abstracts system calls in a non-transparent manner prior to analysis. For this reason, many additional calls are not considered. This arguably increases performance while potentially reducing accuracy. ⋄*Tags: G{detection, analysis}; T{malware, host intrusion}; I{event traces}; D{pattern}; A{behavioral}; K{no}; S{aware}; APT{4,5,7: host activity}.*

In [66], the authors introduce the Taiwan Malware Analysis Net (TWMAN), an ontology system for behavioral malware analysis. The VM-based dynamic analysis system is built around the TRUMAN sandnet [40] that logs contacted IP addresses, created, changed, and deleted files, as well as registry activity. While some features are taken from CWSandbox [142], others are comparable to the generally more detailed BehEMOT approach [60]. The file output is akin to a list of changes to the original system state. An OWL-based ontology built in Protégé [136] describes the impact and approximate activity of each sample and includes a rough categorization into different types of malware. Actors, assets, and activity beyond abstracted malware function calls are not considered in the proposed ontology. ⋄*Tags: G{intelligence, analysis}; T{malware}; I{event traces}; D{ontology}; A{behavioral, contextual}; K{no}; S{operational}; APT{3-7: host and partial network activity}.*

The work of Chiang and Tsaur [24] introduces dedicated, ontology-based behavioral analysis for mobile malware. Their approach is based on the TOVE project by Fox et al. [51] and considers infection routes, potential damage, and propagation capabilities via e.g. Bluetooth. While the authors do not directly propose a practical implementation, their work can be seen as example of a domain-specific, ontology-based approach that could be transported to the information system environment. ⋄*Tags: G{analysis}; T{malware}; I{-}; D{ontology}; A{bahavioral, contextual}; K{no}; S{denotational}; APT{-: general approach}.*

Static malware detection and analysis is less concerned with application behavior, but primarily considers a sample's code structure. Zhang et al. [164] introduce a detection technique for polymorphic code through static identification of its self-decryption functionality. Limited emulation of instructions is used in conjunction with recursive traversal of loops to find the starting location of the respective routine. Since Zhang et al. address a very specific property of only a subclass of malicious programs, their approach could potentially complement systems that attempt to identify other function entry points [44] or illegitimate code paths [20]. ⋄*Tags: G{detection}; T{malware, host intrusion}; I{network traffic, raw}; D{pattern}; A{behavioral}; K{no}; S{aware}; APT{3,5: partial network and initial program activity}.*

Dube et al. [44] take a more general approach. Their system, MaTR, focuses on target recognition of malware attacks through static heuristic features. These non-instruction-based heuristics include structural anomalies such as non-standard section names, characteristics such as the presence of unpacking routines, entry points outside the code section, and unusual numbers of function imports and exports. MaTR utilizes supervised machine learning based on bagged decision trees. Dube's research contrasts the n-gram approach found in other surveyed articles [16,23,124], where sequences of byte patterns of various lengths are assessed. ⋄*Tags: G{detection, analysis}; T{malware}; I{raw}; D{anomaly}; A{behavioral}; K{yes: supervised learning, decision tree classification}; S{aware}; APT{5: structural program properties}.*

One of the most cited works is undoubtedly by Christodorescu et al. [28]. The authors describe a malware detection algorithm that incorporates instruction semantics, an approach that uses resilient pattern matching which can deal with slight variations in code. Activities such as decryption loops and search operations for certain strings are considered malicious behavior indicators, which are specified by templates and depicted as control-flow graphs (CFG). Christodorescu et al. also developed a tool based on IDA Pro [64]: It takes a binary, disassembles it, constructs a CFG, produces an intermediate form using abstract machine language, and determines a possible match. In a later paper, Preda et al. [36] prove that the concept of semantic malware detectors such as [28] is a sound one and can indeed defeat numerous obfuscation techniques. ⋄*Tags: G{detection, analysis}; T{malware}; I{raw}; D{pattern}; A{behavioral, contextual}; K{no}; S{aware}; APT{4,5,7: program activity}.*

Sharif et al. [132] introduce Eureka, a framework for facilitating static analysis of obfuscated code. The sample is first executed to trigger its unpacking procedures, then captured and dumped for static analysis. Here, API resolution comes into play: Eureka identifies subroutines and builds a control flow graph for every function. Calls are then extracted from the image of the previously packed sample. The combination of static and dynamic analysis addresses the issue that packed or encrypted samples are hard to analyze in their original state. Letting the respective routines execute before taking a closer look on the disassembled code promises an unimpeded view on the malware in question – with any of the above-mentioned static analysis tools. ⋄*Tags: G{analysis}; T{malware}; I{event traces, raw}; D{pattern, ontology}; A{attribute, behavioral}; K{yes: behavior graph extraction, visualization}; S{aware}; APT{5: initial program activity}.*

Hardware state detection and analysis solutions encompass raw operation monitoring applications that are often VM-centric. Fore example, the approach described in [74] focuses specifically on virtual machines. VMwatcher constructs an on-system semantic view on files, processes, and kernel modules to bridge the gap between the inside and outside view on a system. Jiang et al. use virtual machine introspection (VMI) to monitor the raw disk and memory states of a VM in a transparent and tamper-resistant fashion. This ultimately eliminates the need to install malware detection software on the guest machine and could enable further research into system state analysis in virtual environments. ⋄*Tags: G{detection}; T{malware}; I{event traces, raw}; D{-}; A{attribute}; K{no}; S{aware}; APT{4,5,7: raw host activity}.*

Payne et al. [114] also focus on VMI-based monitoring. Their proposed solution, Lares, is primarily intended to protect other defense measures such as antivirus programs and intrusion detection systems by providing an isolated API monitoring environment. Lares could be combined with other tools that would benefit from the additional protection against evasion or tampering. ⋄*Tags: G{detection}; T{malware, host intrusion}; I{event traces, raw}; D{-}; A{-}; K{no}; S{aware}; APT{4,5: raw host activity, secondary protection}.*

Mankin and Kaeli [97] propose a similar disk monitoring system: Their implementation, the Disk I/O analysis engine (DIONE), intercepts and interprets disk access operations using a sensor that resides in the Xen hypervisor outside the guest OS. Thanks to this design, DIONE is more resilient against many conventional attacks and various obfuscation techniques. Attacks against the hypervisor itself [115] and cross-VM attacks [9,125] remain an issue, however. Unlike Jiang's VMwatcher [74], DIONE is capable of monitoring the NTFS filesystem. ⋄*Tags: G{detection}; T{malware}; I{raw}; D{-}; A{-}; K{no}; S{aware}; APT{4,5: raw disk operations, secondary protection}.*

Execution path analysis explores or defines code paths that are dynamically investigated during execution. Miles et al. [102] focus on the interrelationship among malware instances to discover new connections between actors, machines, and malware: Code, semantically similar procedures of code, and API call execution/event log traces are compared to identify similarities. This includes websites, e-mail messages and PE file headers. Names, addresses, and certain constants are generalized using BinJuice [89]. The discovery system itself uses concolic execution: It is able to explore multiple execution paths by using an SMT solver to create new input values for subsequent runs. Miles' prototype system, VirusBattle, further includes an unpacking routine as well as monitoring capabilities utilizing VMI. With a focus on threat intelligence and attribution, Miles' approach could be combined with other, more detection-centric methods to enhance its capabilities. ⋄*Tags: G{intelligence, analysis}; T{malware}; I{event traces, raw}; D{similarity}; A{attribute, contextual}; K{no}; S{operational}; APT{3-7: host and partial network activity, web, e-mail}.*

Another interesting approach is presented by Xu et al. [159]. They introduce a waypoint mechanism in the form of markers on the execution path that a process must follow to provide trustworthy control flow information for subsequent anomaly monitoring. Waypoints, for which Xu et al. use the context of the currently active function of the code, are generated through static analysis and later imported as traps into the kernel. A system utilizing this method would be able to detect mimicry attacks [153], i.e. the interleaving of malicious calls in benign sequences. ⋄*Tags: G{detection}; T{malware, host intrusion}; I{raw}; D{anomaly}; A{contextual}; K{no}; S{operational}; APT{4,5: program flow}.*

PECAN [20] is a dynamic anomaly detector that identifies unusual program behavior through the definition of legitimate code paths. It has been developed to identify bugs that use valid executions but violate the programmer's expectations. The authors introduce both a training and monitoring module: The system scans for specific security functions that can affect the system outside the Java VM, considers the calling context of a call, and checks for anomalous sequences. This is achieved through probabilistic calling context (PCC), which appends a unique number that represents the location a program is called from [19]. Clients then query that number at every system call to determine whether the context is known or suspicious. While PECAN is a solution specific to Java, the concept of context can also be found in system calls [86]. Synergies to process-graph-based systems are likely, but have yet to be investigated. ⋄*Tags: G{detection}; T{malware}; I{raw}; D{anomaly}; A{bahavioral, contextual}; K{supervised learning}; S{operational}; APT{4,5: program flow}.*

Behavior extraction solutions often focus on graphs and subgraphs of certain behavioral commonalities. Often similar to classification approaches (see below), they usually focus on the generation of knowledge.

Kwon and Lee [88] introduce BinGraph, a discovery method for metamorphic malware. API calls are extracted and converted to a hierarchical behavior graph. Extracted subgraphs represent common behavior and can be considered a 'semantic signature' – Kwon and Lee assume that the same API sequences occur in metamorphic variants of the same malware strain. To counter isomorphism, node types are abstracted into categories such as process, registry, memory, or socket operations. Each operation can be further split into groups that denote the specific action (open, close, read, write) performed. Graphs are matched to search for subgraphs in newly submitted traces. ⋄*Tags: G{detection, analysis}; T{malware}; I{event traces}; D{graph}; A{attribute}; K{yes: behavior graph extraction}; S{aware}; APT{4-7: host activity, sockets}.*

In [155], the Wang et al. describe a malware feature extraction system based on application behavior. System calls and their dependencies are mapped to a graph and then tainted to trace the passing on of parameters. Calls of interest include file, registry, process, and network functions. Single-step debugging is used to keep track of the tainted information. Semantic analysis is used on certain, not further specified calls. The taint approach sets this work apart from other solutions and allows the authors to extract calls that are actively used by malware. ⋄*Tags: G{intelligence, analysis}; T{malware}; I{event traces}; D{graph}; A{behavioral, contextual}; K{yes: dependence graph extraction}; S{aware}; APT{4-7: host and partial network activity}.*

Another graph-based approach, DAVAST, is described by Wüchner et al. [156]: The authors visualize trace information as quantitative data flow graphs where files, sockets, processes, and more are represented as nodes while the edges depict the execution flow. Both online and offline analysis is supported. Rules define normal behavior as well as malicious activities. Abstraction of e.g. memory addresses is used to reduce processing complexity. Unlike most other solutions, Wüchner et al. use time slices to further reduce the size of their graphs. Since DAVAST primarily focuses on visualization, further automated processing it is not explored. It is currently unknown if Wüchner's approach [156] could be used in combination with e.g. Wang's feature extraction system [155] or other graph-based approaches such as BinGraph [88] or the system introduced by Dolgikh [41]. ⋄*Tags: G{detection, analysis}; T{malware, host intrusion}; I{event traces}; D{pattern}; A{behavioral}; K{yes: visualization}; S{aware}; APT{3-7: host activity, sockets, e-mails}.*

Dolgikh et al. [41] conduct behavioral analysis capable to automatically create application profiles for both malicious and benign samples. Their system considers recorded API calls that are subsequently transformed into a labeled graph representing a stream of system calls. Graphs are compressed using the Graphitour algorithm [116] used in genetic data processing. When applied to benign software, the resulting dataset describes rules that represent normalcy – functionality of system calls that can be seen as benign behavior. Ultimately, each node in the rule graph is modeled as part of a Colored Petri Net [73]. ◇*Tags: G{intelligence, detection}; T{malware}; I{event traces}; D{anomaly, graph}; A{behavioral}; K{yes: supervised learning, classification grammar, behavior graph extraction}; S{aware}; APT{4,5,7: host activity}.*

Graph mining is another interesting approach to extracting behavior. Next to developing a language for specifying malicious behavior through system call dependencies, Christodorescu et al. [27] came up with a method to mine behavioral data from trace files of benign and malicious samples. Both are collected during dynamic analysis and linked through their parameters. Christodorescu's system, dubbed 'MiniMal', constructs dependence graphs for later comparison. To keep the classification process nonspecific to individual samples, the graphs are generalized into so-called minimal contrast subgraphs describing the smallest possible malicious subgraph that does not occur in benign sequences. Other works build on Christodorescu et al.'s concept: Fredrikson et al. [52] introduce a graph mining method for extracting significant behavioral specifications used to describe classes of programs. Their system, Holmes, aims to extract specifications that distinguish malicious from benign applications on a system call level. ◇*Tags: G{intelligence, detection}; T{malware}; I{event traces}; D{graph}; A{behavioral}; K{yes: deductive learning, dependence graph extraction, visualization}; S{aware, denotational}; APT{4,5,7: host activity}.*

Touching the network domain, Jacobs et al. [72] present Jackstraws, a system designed to identify command and control (C2) communication. Unlike other network-centric approaches, Jackstraws captures host activity through dynamic analysis performed by tools such as Anubis [70] and associates network communication to local malware activity. Association is achieved through behavior graph modeling of data flows between individual system calls. Graph templates for C2 pattern similarity matching are mined from a known set based on a technique introduced by Yan and Han [162], followed by a clustering stage. While Jackstraws is potentially vulnerable to certain obfuscation and mimicry attacks [153], its unique approach to detecting C2 traffic makes it particular interesting to APT detection and knowledge generation efforts. ◇*Tags: G{intelligence, detection}; T{malware}; I{event traces}; D{pattern. similarity, graph}; A{behavioral, contextual}; K{yes: supervised learning, graph clustering, behavior graph extraction}; S{aware}; APT{6: C2 network activity}.*

Malware classification solutions found in literature usually cluster dynamic analysis traces or static function use. The primary purpose of malware classification is the generation of knowledge through sample similarity assessment. Unlike behavior extraction (see above), classification does not necessarily yield discriminative patterns for subsequent use.

Riek et al. [124,145] describe a clustering and classification approach for malware behavior traces generated by dedicated dynamic analysis tools (in their case the malware sandbox 'CWSandbox' [142]). The focus lies on API and system calls as well as their arguments. The reports are embedded in a vector space in order to enable similarity assessment based on geometry. Hierarchical clustering is then used to identify groups of malware displaying similar behavior. The distance to a representative prototype is determined and stored. Incremental analysis allows for comparing new samples to existing clusters. For better granularity, the authors introduced the optional Malware Instruction Set (MIST) format: The behavior of a binary is described as a sequence of instructions similar to CPU opcodes. Each level of MIST allows for additional details to be included or omitted on demand. Riek et al.'s work later became 'Malheur' [123], a system capable of rapidly processing function call n-grams. Malheur's performance and flexibility makes it a useful tool for pre-classifying samples without having to convert existing traces. However, since Riek's approach does not extract patterns like e.g. Christodorescu's work [27], Malheur's performance and accuracy comes at the expense of semantic expressiveness. ◇*Tags: G{intelligence}; T{malware}; I{event traces}; D{-}; A{behavioral}; K{yes: unsupervised learning, nearest prototype clustering}; S{aware}; APT{4-7: support solution}.*

In [13], Bayer et al. present a scalable behavior-based malware clustering system. Dynamic analysis is performed using Anubis [70] and extended with taint tracking functionality that marks out-arguments and return values and monitors them for changes. The resulting trace files are then generalized into profiles which characterize sample behavior. Ultimately, the abstracted data is clustered using the locality sensitive hashing (LSH) algorithm based on the work of Indyk and Motwani [69]. Despite the fact that Riek's approach is more accurate in terms of F-measure [124], the extraction of profiles is a distinct advantage over pure classification systems like Malheur. ◇*Tags: G{intelligence}; T{malware}; I{event traces}; D{pattern, similarity}; A{behavioral}; K{yes: nearest prototype classification}; S{aware}; APT{4-7: host and limited network activity}.*

5.1.2 Host-based intrusion detection systems

Host-based intrusion detection systems focus less on the tool/malware utilized by the attacker but rather on anomalous activities registered on the system. For example, Mutz et al. [86] describe a classical approach to detecting anomalies in system call sequences. Their system is designed to detect attacks against privileged applications. To this end, it analyzes the relation between system call arguments and calling contexts. Function return addresses gathered from the application call stack are used to add cohesion. Among the anomalies considered are string length, character distribution, and the use of certain non-printable characters. As part of Mutz' prototype implementation, a Linux kernel module monitors system calls through a wrapper function that logs relevant activity before actual execution. ⋄*Tags: G{detection}; T{host intrusion}; I{event traces, raw}; D{anomaly}; A{behavioral, contextual}; K{yes: supervised learning, Markov-based classification}; S{aware}; APT{4,5,7: host activity}.*

Anomalies in system call patterns are a predominant theme: Creech and Hu [34] introduce a host-based anomaly detection method that uses discontiguous (unconnected) system call patterns. A context-free grammar describes benign and malicious call traces; learning is achieved through an Extreme Learning Machine (ELM), a new type of fast-learning artificial neural network. Several decision engines were tested and compared by the authors, making the paper a good starting point for the selection of learning algorithms applicable to system call sequences. ⋄*Tags: G{intelligence, detection}; T{host intrusion}; I{event traces}; D{anomaly}; A{contextual}; K{yes: supervised learning, neural network/ Markov-/grammar-based classification}; S{aware w/ denotational elements}; APT{4,5,7: host activity}.*

A way to formally describe a data flow without losing information about the accessing resource is described in Chaturvedi's work [23]. Their model can capture certain properties of the sample and identifies anomalies hinting at malicious use of functions or configuration files. Among the considered properties are command line arguments and environment specifics such as local variables. ⋄*Tags: G{detection}; T{malware, host intrusion}; I{event traces}; D{anomaly}; A{behavioral}; K{yes: supervised learning, behavior graph extraction}; S{aware}; APT{4,5,7: host activity}.*

Based on this approach, Bhatkar et al. [16] propose anomaly detection within the data flow of an application: In addition to calls, call arguments are considered and used to establish a temporal context. Unary relations define the properties of an argument in the form of a specific value or range, while binary relations establish the relationship between two event arguments. This model allows to search for e.g. certain paths or file extensions and is able to compare them to each other. With its focus on learning as well as matching control flows, both Bhatkar's [16] and Chaturvedi's [23] systems exemplify and discuss the use of n-gram and execution graph methods. ⋄*Tags: G{detection}; T{malware, host intrusion}; I{event traces}; D{pattern, ontology}; A{contextual}; K{yes: supervised learning, Markov-based extraction}; S{aware, denotational}; APT{4,5,7: host activity}.*

Ou et al. [113] emphasize the issue that attacker and user actions are often syntactically similar and are therefore hard to distinguish from one another. They propose the formal modeling of uncertainties to counter false-positives and use various system monitoring data sources to score them. The introduced reasoning engine is designed to determine whether a system is actually compromised. Statements written in the logic programming language Prolog [31] describe the rules and aim to emulate the reasoning process of a human administrator. ⋄*Tags: G{prediction, detection}; T{host intrusion}; I{system/app logs, network traffic, event traces}; D{-}; A{-}; K{no}; S{denotational}; APT{3-7: general approach}.*

Kumar and Spafford [87] take a step back and present a pattern matching model for misuse intrusion detection, which is not based on the premise that intrusive activity always manifests as an abnormality. Instead, misuse detection uses knowledge about attacks such as known exploits as well as patterns and monitors the system for the occurrence of these patterns. The approach proposed in their paper defines the patterns as state transition graphs that are an adaptation of Colored Petri Nets [73]. Start and final states as well as the paths in between are matched by the net. Like most pattern-based approaches, Kumar's system only looks for known behavior. ⋄*Tags: G{detection}; T{host intrusion}; I{-}; D{pattern}; A{behavioral}; K{no}; S{aware}; APT{4,5,7: general approach}.*

Andersson et al. [6] focus on code injection that can be mapped to DLL injection attacks. Their framework uses process tracing and DLL hooking based on the Microsoft Detours library: Potentially executable instructions are identified through Snort [30] and Fnord [22] and are sent to a monitored environment where they are dynamically scanned for shellcode. Two methods were employed by the authors: NOP detection similar to [28], and executable code identification. Andersson's framework is an example for how suspicious traffic data can be automatically forwarded by an IDS to a dynamic host-based analysis tool. ⋄*Tags: G{detection, analysis}; T{host intrusion}; I{network traffic, event traces}; D{pattern}; A{behavioral}; K{no}; S{aware}; APT{4-7: partial host and network activity}.*

5.2 Network domain

Network-based approaches primarily include generic network-based intrusion detection systems (NIDS), specific attack detection systems, traffic flow analysis solutions as well as detection systems for malicious web traffic.

Although usually classified as NIDS (see above), network traffic analysis systems deserve special attention. They come in two distinct flavors [135]: *Packet inspectors* that analyze the payload of certain (or all) packets, and *flow-based detection systems*. The latter focus on communication patterns instead of individual packets. Such patterns typically include source and destination IP addresses, port numbers, timestamps and transmission duration, as well as the amount of data and number of packets sent. Flow monitoring systems like compatible IOS routers usually export Netflow [29] or IPFIX [144] records and transmit them to a central node for analysis.

Detailed packet inspection requires significant processing capacities and may cause slowdowns on even the most powerful networks. For this reason, many solutions focus primarily on header information and analyze only specific packets such as HTTP requests and responses relevant for malicious web traffic detection. Flow-based systems are generally faster but ignore packet payloads. In addition, they are reliant on external analysis systems that amalgamate and interpret the collected data.

Both approaches may utilize pattern- and anomaly-based detection (see HIDS above). For example, a large number of SSH connections to a specific destination IP address could match the pattern of a brute-force attack. Significant amounts of data sent during the out-of-hours period, on the other hand, could constitute a data leakage anomaly.

A growing number of data providers use flow-based detection to complement classical packet inspection. It is often used to identify Denial of Service (DoS) attacks and other suspicious communication patterns. Packet scanning, on the other hand, aims at finding potentially malicious payload data. This makes it the direct equivalent to signature-based malware detection.

Many network-based systems use publicly available datasets to test the capabilities of an NIDS. Examples include the 2000 DARPA set [104] and KDD 1999 [146]. On the payload side, ADMmutate and the Clet engine [76] are often used to generate (polymorphic) shellcode.

5.2.1 Network-based intrusion detection systems

General NIDS are represented in a large number of papers. Several of those include a semantic component. For example, Abdoli and Kahani [1] describe a distributed IDS that can extract semantic relations between attacks using an ontology based on the Semantic Web [150], a collection of W3C formats for data exchange. Like similar solutions [24,66], they utilize Protégé [136] for ontology design and SPARQL [152] for querying. Their system uses Java-based JENA [8] agents to collect IPs, ports, protocols, and connection status information. A dedicated master node interprets the data and attempts to find indicators for embedded malware, buffer overflows, password attacks, or ongoing DoS activity. ⋄*Tags: G{detection}; T{malware, host intrusion, network intrusion}; I{network traffic}; D{pattern, ontology}; A{attribute, contextual}; K{no}; S{aware}; APT{1,3-7: network activity with payload detection}.*

Many papers are based on or inspired by Christodorescu et al.'s work ([28], see 'Static malware detection and analysis): In [129], the authors describe two sliding-window-based schemes used to automatically generate malware signatures: a fixed-size scheme and a more flexible variable length scheme which stops at certain, predefined patterns. In combination, they augment Christodorescu's approach with a traffic classifier and a binary detection and extraction module capturing polymorphic shellcode. Like a part of Andersson's less successful host-based approach [6], their work focuses on the presence of 'no operation' (NOP) and NOP-like instructions. ⋄*Tags: G{correlation, detection}; T{malware, host intrusion, network intrusion}; I{network traffic}; D{pattern}; A{attribute, contextual}; K{no}; S{aware}; APT{3,4,6: network activity with payload detection}.*

Hirono et al. [65] present another, more architecture-centered approach. They propose a distributed IDS that uses a transparent proxy able to analyze internal network traffic in an isolated environment. The system primarily uses signature-based detection to identify malware propagation, spamming, or DoS attacks. Suspicious binaries are automatically extracted and forwarded to a dynamic malware analysis sandbox. The decision whether a sample is suspicious is made by Snort [30] complemented by an off-the-shelf virus scanner. Similarly, Andersson [6] uses the Snort IDS for initial decision-making. Unlike Andersson and Scheirer [129], Hirono et al. do not attemt to identify shellcode but rather focus on the extraction and analysis process of the flagged binaries. ⋄*Tags: G{detection, analysis}; T{malware, host intrusion, network intrusion}; I{network traffic, event traces}; D{pattern}; A{attribute}; K{no}; S{aware}; APT{3,7: network activity with normal malware scan}.*

Correlation of intrusion events is a vital part of most multi-agent IDS systems. Chien et al. [26] introduce a primitive-attack (PA)-based correlation framework able to detect multi-stage attacks. IDS alerts from various tools such as Snort [30] are extracted, stored and evaluated. Through time window correlation complemented by port and IP matching, the system is able to identify e.g. network scans and denial of service attacks. The authors construct attack templates, mapping abstracted goals to specific attacker actions such as reconnaissance, penetration, and other unauthorized activity. ⋄*Tags: G{correlation, detection}; T{network intrusion}; I{alerts}; D{ontology}; A{contextual}; K{no}; S{axiomatic}; APT{1,3,6,7: partial network activity}.*

Zhu and Ghorbani [165] present an alert correlation technique that also considers attacker strategies. Their classification encompasses a neural network as well as a support-vector machine (SVM) [33] approach. Their system suggests relationship of alerts: For example, consequences of one event are mapped to the prerequisites of another in order to construct scenarios that consider both correlation strength and probability. In addition to IP similarities and matching ports, the frequency of alerts is assessed. In the end, an attack graph is generated from the extracted attacker strategies. ◇*Tags: G{prediction, intelligence, correlation, detection}; T{network intrusion}; I{alerts}; D{similarity, ontology}; A{contextual}; K{yes: supervised learning, SVM and neural network classification, behavior graph extraction}; S{aware}; APT{1,3,6: network activity}.*

A more recent paper by AlEroud and Karabatis [2] (also see [62]) presents a layered attack detection system that utilizes semantics and context through a semantic network describing relationships between attacks. Their approach uses Conditional Entropy Theory – the uncertainty of one event given another [131] – to create attack context profiles that filter out non-relevant events. In addition, the Anderberg similarity coefficient measure [43] is used to detect similarities in binary network data. For example, the ‘host context’ profile considers type, vulnerabilities, operating system, applications, and services when filtering. ◇*Tags: G{prediction, correlation, detection}; T{host intrusion, network intrusion}; I{network traffic}; D{ontology}; A{contextual}; K{yes: supervised learning, Bayesian classification, semantic network extraction}; S{aware}; APT{1,3-7: general approach}.*

5.2.2 Traffic flow analysis solutions

Flow-based approaches are increasingly used to detect network attacks. Sperotto et al. [135] offer a good overview of traffic flow IDS systems. Over the years, a number of different solutions have been proposed:

Münz and Carle [110] present TOPAS, a traffic flow and packet analysis system compatible with Cisco NetFlow and IPFIX. TOPAS provides a framework for user-defined detection modules operating in real-time. The data used is netflow-specific in nature: Source and destination IP addresses/ports as well as protocols are considered. The system's detection algorithm encompasses threshold-based detection via pre-defined values that need to be exceeded, principal component classifiers (PCC) to detect anomalies in multivariate time series, outlier detection through the comparison of a sample to previously learned, normal behavior, and rule learning through a classification extracted from these “good” and “evil” training sets. ◇*Tags: G{detection, analysis}; T{network intrusion}; I{network traffic}; D{anomaly, pattern}; A{contextual}; K{yes: supervised learning, outlier classification}; S{aware}; APT{1,6: network flows}.*

Vance's work [148] is one of the few approaches that focus directly on APTs: He describes a flow-based monitoring system that uses statistical analysis of captured network traffic data to detect anomalies. He uses change detection through sketch-based measurement [85] to identify flows that hint at command & control traffic, data mining, or exfiltration activities. Volume, timing, and packet size are of primary interest; the respective baseline and subsequent analysis consider packet and traffic throughput, number of concurrent flows, TCP/IP SYN and RST packets, flow duration, and the current time. ◇*Tags: G{intelligence, detection}; T{network intrusion}; I{network traffic}; D{anomaly}; A{contextual}; K{yes: change classification}; S{aware}; APT{6: specific network flows}.*

Another example of flow-based intrusion detection is described in [3]: The authors' approach utilizes probabilistic semantic link networks (SLN) synonymous to graphs using similarity values to describe node connections. The target's IP address, time and duration of communication, as well as various other features such as protocols and flags are mined from network traffic and the corresponding flows. Common characteristics are translated into link weight between the respective nodes of the graph. ◇*Tags: G{prediction, intelligence, correlation, detection}; T{host and network intrusion}; I{system logs, app logs, network traffic, alerts}; D{similarity}; A{contextual}; K{yes: decision tree classification, semantic network extraction}; S{axiomatic}; APT{1,3,6: general approach}.*

5.2.3 Web traffic detection and analysis systems

Web service attacks are often part of the reconnaissance stages of a targeted attack or aim to publicly disclose sensitive information. More often than not, business-critical infrastructure can be accessed through exposed web portals that allow privileged users to monitor or configure backend systems. Because of their high visibility, web servers are also frequent targets of defacement attacks. For these reasons, a number of security solutions focus on the detection of malicious HTTP traffic:

Razzaq et al. [121] describe a system able to detect and classify web application attacks. Threats are specified through semantic rules that establish the context: Both attack consequences and common application properties such as protocol use are evaluated. The system analyzes the user part of an HTTP request (header, message) to detect authentication bypass attacks, DoS, probing, cookie stealing attacks, and more. The authors developed an ontological model (see also [120]) using a description logic based on OWL [100] and validated through OntoClean [61]. The inference rules were implemented using the Apache JENA framework [8]. ◇*Tags: G{intelligence, correlation, detection, analysis}; T{network intrusion}; I{network traffic}; D{ontology}; A{contextual}; K{yes: deductive learning}; S{axiomatic}; APT{1,3,6,7: specific network traffic}.*

Earlier work of Razzaq et al. [119] describes another interesting semantic approach. The authors introduce an application-level IDS using a Bayesian filter to find malicious scripts in HTTP protocol traffic. URL, parameters, cookies, etc. are considered. This 'spam filter' assigns values to specific keywords and generates a score. Potential attacks are matched with attack descriptions stored in an external database. ◇*Tags: G{detection}; T{network intrusion}; I{network traffic}; D{ontology}; A{contextual}; K{yes: deductive learning, Bayesian classification}; S{aware}; APT{3,6: specific network traffic}.*

Another semantic intrusion detection system is presented by Sangeetha and Vaidehi [128]. It defines malicious behavior as rules that consider source, frequency of occurrence, and parts of the HTTP packet content. Rules are represented as BNF grammar [48]. Fuzzy Cognitive Mapping [84] is used for attack prediction. The authors' traffic sniffer and interpreter system was implemented as part of a client-server infrastructure. ◇*Tags: G{prediction, detection, analysis}; T{network intrusion}; I{network traffic}; D{pattern, similarity}; A{contextual}; K{no}; S{aware, denotational}; APT{3,6: specific network traffic}.*

SpuNge [12] is a system explicitly designed to detect targeted attacks through behavior clustering (similar behavior with respect to malicious resources used) and location/industry correlation. The analysis focus lies on URLs – its framework is able to detect machines that are part of the same attack. This is achieved through hierarchical clustering and string similarity measurement utilizing the Levenshtein distance [92]. SpuNge determines host distance (hostname similarity) and request distance (request path similarity) and groups the processed requests accordingly. ◇*Tags: G{intelligence, correlation, detection}; T{host and network intrusion}; I{network traffic}; D{similarity}; A{contextual}; K{yes: hierarchical clustering}; S{aware}; APT{1,3,6: specific network traffic}.*

Thakar et al. [140] also focus on the analysis and extraction of traffic patterns. Unlike SpuNge, their work revolves around the extraction of signatures that can later be used by an IDS. Specifically, the authors log SOAP [151] traffic and extract information such as client identifiers, IP addresses, ports, and certain strings in HTTP requests and response packets. The data is then clustered using an SVM-based classifier [33]. Extraction and analysis utilizes the Longest Common Substring (LCS) algorithm [10]. ◇*Tags: G{intelligence, analysis}; T{network intrusion}; I{network traffic}; D{pattern}; A{attribute}; K{yes: SVM-based classification}; S{aware}; APT{1,3,6: specific network traffic}.*

Zarras et al. [163] introduce BotHound, a detection method for malware communicating over HTTP. The system automatically generates models for benign and malicious requests and classifies new traffic in real-time. The primary goal is to discover bot traffic and C2 communication through suspicious header chains (sequences of HTTP headers) and HTTP templates that encompass content data such as IP addresses, ports, transported file types, and more. Like in [12], string similarity is measured using the Levenshtein distance [92]. ◇*Tags: G{detection}; T{malware}; I{network traffic}; D{pattern, similarity}; A{attribute, behavioral}; K{yes: supervised learning}; S{aware}; APT{6,7: specific network traffic}.*

5.2.4 Attack-specific approaches

There are a number of network attack detection systems that focus on a specific type of threat. Because of their prominence, (distributed) *Denial of Service* (D/DoS) attacks are of special interest. One such solution is introduced by Gamer et al. [55]. Their proposed attack detection system is placed on routers and focuses on DDoS attacks and malware propagation. Network traffic is sampled and then refined in several stages: At the first level of granularity, Gamer's approach only considers the overall number of packets and attempts to find anomalous changes in volume. Level two differentiates DDoS from worm propagation by analyzing target subnets. Protocol anomalies hinting at a specific attack are detected in stage three. This e.g. includes the anomalous ratio between incoming and outgoing packets that typically accompanies a DDoS attack. ◇*Tags: G{detection}; T{network intrusion}; I{network traffic}; D{anomaly}; A{behavioral}; K{no}; S{aware}; APT{3,7: specific network traffic}.*

'Vanguard' [95] is a detection system addressing low-rate and random-interval DoS attacks. Luo at al.'s approach is formal in nature: They propose a detection scheme for polymorphic DoS attacks that registers anomalies in TCP traffic. The decision is primarily based the ratio between incoming data and outgoing ACK packets. Vanguard has been implemented as Snort [30] preprocessor plug-in, presenting a more specific and better tested approach than Gamer's model [55]. However, its narrow focus on low-rate DoS attacks limits its use; the applicability of the algorithm to other types of DoS attacks has not yet been explored. ◇*Tags: G{detection}; T{network intrusion}; I{network traffic}; D{anomaly}; A{behavioral}; K{no}; S{aware}; APT{3,7: specific network traffic}.*

On the ontology side, the system by Ansarinia et al. [7] provides an interesting take on the detection of DDoS attacks. The authors model prerequisites and consequences (e.g. unwanted disclosure) of such attacks and automatically generate an attack ontology based on Mitre's CAPEC [105], CWE [108], and CVE [107] threat information. Events and IDS logs are combined and converted to a single format: CEE [106]. In the end, it is possible to ontologically describe how a vulnerability is comprised of certain weaknesses and how exploiting them leads to a successful attack. ⋄*Tags: G{detection}; T{network intrusion}; I{threat info, alerts}; D{pattern, ontology}; A{contextual}; K{yes: deductive learning}; S{denotational, axiomatic}; APT{1,3,6: general approach}.*

5.3 Multi-source domain

Multi-source approaches typically focus on data fusion and event correlation. The primary categories within this domain are SIEM-like solutions and log correlation systems:

Logs are record files created by a wide range of devices and applications. Their primary purpose is non-repudiation through event auditing as well as system diagnostics. *Log analysis* systems retrieve log files and assess their contents. This can again be done using pattern or anomaly detection (see IDS) and will typically yield simplified alerts or a list of anomalous log entries. Their simple yet versatile nature makes bulk-processing easy and feasible for a wide range of applications.

Log analysis is often used in conjunction with multi-source event fusion. Solutions such as *Security Information and Event Management* (SIEM) systems take log files of e.g. several intrusion detection systems, traffic flow analyzers, and OS event logs to correlate or visualize attacks. SIEM systems usually do not monitor assets on their own; they merely process logs, alerts, and other monitoring reports generated and supplied by other tools. Multi-source (log) analysis is the closest thing to an attack interpretation system currently on the market.

SIEM systems and SIEM-like event fusion tools have become increasingly important in today's cyber-defense. SIEM development has spawned open source solutions like OSSIM [4] as well as various commercial products. Combined with the assessment of conventional log files, multi-source event aggregation and correlation is a promising new approach to understanding cyber-attacks.

5.3.1 SIEM-like systems

SIEM-like systems are prototypical of the multi-source domain. One of the earliest multi-source approaches was introduced by Gorodetski et al. [58]. Their general paper on multi-agent system (MAS) technology for IDS encompasses attack simulation and intrusion detection learning. A model mapping attacker intentions to actions as well as targets is introduced and formally described by the authors. The proposed simulator considers network traffic data, data from the OS audit trail, system logs, and application audit data. Combined attacks with e.g. shared source IP addresses are detected through pattern matching on pre-processed input streams. Specific learning algorithms are mentioned but not explained in detail. ⋄*Tags: G{intelligence, detection}; T{host and network intrusion}; I{system logs, app logs, network traffic, alerts}; D{pattern}; A{behavioral}; K{yes: supervised learning, Bayesian classification, visualization}; S{aware, denotational}; APT{1,3-7: general approach, attack simulation}.*

A general APT attack model following the intrusion kill chain [67] is presented by Bhatt and Gustavsson [17]: The logging module collects security events as well as various logs and submits them for analysis. Malware forensics is part of the framework but not specified in detail. A dedicated intelligence module is responsible for event correlation and searching. An experimental implementation was realized on a 5-node Apache Hadoop cluster and tested with constructed log files. ⋄*Tags: G{intelligence, analysis}; T{malware, host and network intrusion}; I{system logs, app logs, alerts}; D{pattern}; A{behavioral}; K{no}; S{aware, denotational}; APT{3,5-7: general log analysis approach}.*

The system proposed in [99] aims to establish real-time situational awareness through semantic event fusion to detect multi-stage attacks. Event streams from intrusion detection systems are correlated with pre-defined alert templates and mapped to various categories (e.g. scan or intrusion), services (e.g. web, FTP), protocol stacks, and consequences (e.g. DoS). Attack criticality is also modelled. Unlike Bhatt's primarily time-based approach [17], Mathew et al.'s basis for correlation are similarities of IP addresses and semantically linked events. The authors implemented their system using the model editor FUME [98] and the fusion engine INFERD [137] on an emulated OSIS network [57]. ⋄*Tags: G{intelligence, correlation}; T{network intrusion}; I{system logs, app logs, alerts}; D{pattern, similarity}; A{contextual}; K{no}; S{aware}; APT{1,3,6: network activity}.*

Atighetchi et al. [11] present 'Gestalt', a cyber-information management system that simplifies the access to event data stored on various systems. Unlike classical SIEM solutions, Gestalt leaves the data where it was generated. The focus lies on forensics: the actual methods and techniques required to access the data are abstracted and described using a new Cyber Defense Language (CDL). All information is provided via a single interface accessible from a central management workstation – something that is achieved through the use of the Asio tool suite [15] and the web ontology language OWL [100]. Queries are submitted using SPARQL [152]. ⋄*Tags: G{intelligence, correlation, response}; T{host and network intrusion}; I{system logs, app logs, network traffic, event traces, alerts}; D{ontology}; A{contextual}; K{no}; S{denotational}; APT{1,3-7: multi-system data correlation}.*

Sadighian et al. [127] propose an alert fusion approach that incorporates public vulnerability data (CVE, NVD) and contextual attack information such as network configurations, host settings, application requirements, and user-specific configurations. This data is retrieved from a dedicated configuration management system. The collected information is then converted to a unified format and combined with common IDS alerts. A pre-populated set of ontologies for alerts, context information, and vulnerabilities is used as basis for the subsequent decision process. Fusion rules are built using SWRL, a language based on the OWL description logic (OWL-DL). In contrast to other solutions, Sadighian's system focuses on alerts collected by different sensors but describing the same event. This potentially reduces redundant and irrelevant information. ⋄*Tags: G{detection, analysis}; T{network intrusion}; I{threat info, network traffic, alerts}; D{ontology}; A{contextual}; K{no}; S{denotational}; APT{1,3,6: network activity event fusion rules}.*

5.3.2 Event correlation solutions

The second focus of multi-source data analysis is correlation. Unlike SIEM systems, the focus here lies more on the underlying correlation algorithms and less on data generation or management considerations. This promises high synergy potential between solutions of the two categories.

In [54], the authors describe the application of data mining techniques to identify patterns in data. A combination of association analysis, which aims to discover interesting relationships, and similarity-based propagation analysis is used to fuse log events into semantic incidents. Host data, user data, and events from both the OS and antivirus software are used as input for classification. ⋄*Tags: G{intelligence, correlation}; T{malware, host intrusion}; I{system logs, app logs, event traces}; D{pattern, similarity}; A{contextual}; K{yes: deductive learning}; S{aware}; APT{4,5: host event discovery}.*

Langeder [91] introduces a framework for dynamic threat recognition and combines it with a proof-of-concept classification comparing Bayes, SVM, and decision trees. Patternized rules are extracted from a training environment and include attributes such as IP addresses, time, HTTP status and request information, ports, users, and more. Best results were achieved with the SVM approach; however, processing performance was only assessed for small data sets. Further domain testing is required to generically compare the various classification methods. ⋄*Tags: G{intelligence, correlation}; T{network intrusion}; I{app logs}; D{pattern, anomaly}; A{contextual}; K{yes: unsupervised learning, SVM, decision tree, Bayesian classification}; S{aware}; APT{1,3,6: network event classification}.*

Bordering the domain of network-based correlation solutions, the work by Debar and Wespi [39] addresses IDS weaknesses such as event flooding, lack of context, false alerts and lack of scalability. The authors propose an intrusion detection architecture that correlates the output of several host-based and network-based probes to produce a condensed view of an incident. Next to the definition of conceptual and operational requirements, an alert class hierarchy considering both target and probes is presented: Generic information and probe status messages are combined with victim host information such as process or port details. ⋄*Tags: G{intelligence, correlation, analysis}; T{host and network intrusion}; I{alerts}; D{-}; A{contextual}; K{no}; S{aware}; APT{3-7: host and network alert correlation}.*

Strasburg et al. [138] introduce S-MAIDS, a semantic model for automated tuning, correlation, and response selection in IDSs based on observable attack indicators the authors call ‘signals’. Each signal is decomposed into a domain/characteristic such as the high-level protocol used (e.g. TCP), a type constraint (e.g. integer), and an value (e.g. 80). The proposed model is formalized using OWL. Cross-system correlation was assessed using IIS log messages and the output of a Netflow-aware system. As a reasoning-based ontology, S-MAIDS requires predefined attack responses to be present in the knowledge base. ⋄*Tags: G{intelligence, correlation, response}; T{host and network intrusion}; I{app logs, network traffic}; D{-}; A{contextual}; K{no}; S{aware, denotational}; APT{3,6: app and network event correlation}.*

5.4 Semantic domain

Unlike the other three categories, this section focuses solely on formal definitions and general ontology models. Only approaches that specifically revolve around formal definitions, ontologies, and other semantic approaches are reviewed here. Also, this section includes articles that cannot be clearly attributed to one of the other domains. Semantics-aware solutions designed to support attack detection on the host or network can be found in the respective subsections above.

A number of models and some select data providers focus on semantics-based detection of cyber-attacks or malicious behavior in general. Most of the time, the term ‘semantics’ is used very loosely and only refers to the process of assigning meaning to e.g. specific patterns of system functions or network packets. Other solutions identify sequences of code that produce identical results.

While almost every solution discussed in this paper can be considered semantics-aware, this particular section focuses on semantics-based approaches per our definition found in Sect. 2.2.

5.4.1 General semantic systems and ontologies

Semantic systems and ontologies have found resonance with the information security community a good while ago. Landwehr et al. [90] were among the first to define a taxonomy of computer program security flaws. While that was not an ontology per se, their work laid the foundation for later attack models. Raskin and Nirenburg [117] eventually introduced a semantic approach to information security in regards to the unification of terms and nomenclature. Today, several application-independent semantic systems can be found in the literature:

Razzaq et al. [120] describe a general approach to ontology-based attack detection and argue its suitability for web application security purposes. While they also propose a domain-specific model (see [121] in the overview tables), the paper primarily introduces general ontology engineering methodologies such as 'Methontology' [49]. A layered model for ontology design is presented.

The paper by Anagnostopoulos et al. [5] is a workable example for the application of semantics to general intrusion scenarios. The authors seek to classify and predict attacker intentions using a Bayesian classifier and a probabilistic inference algorithm. Their semantic model includes both legitimate and illegitimate actors, the formal characterization – dubbed behavior – of the actor, activities in the form of sequential events, concrete commands issued, and an overall state of attack triggered by specific commands. ⋄*Tags: G{prediction, intelligence}; T{host and network intrusion}; I{-}; D{pattern, ontology}; A{contextual}; K{yes: deductive learning, Bayesian classification}; S{axiomatic, denotational}; APT{3-7: general approach}.*

Yan et al. [160], on the other hand, focus on the conversion of raw sensor alerts into a machine-understandable format in order to enable easier data fusion. They advertise the use of a Principal-subordinate Consequence Tagging Case Grammar (PCTCG) that considers object, location, method, cause, 'has object' and 'is part of' rules, attack stage, and attack consequence of an intrusion. Together with a 2-atom semantic network [149], their system is able to generate attack scenario classes that can be extracted and used as detection templates for e.g. IDS systems. ⋄*Tags: G{intelligence}; T{host and network intrusion}; I{alerts}; D{ontology}; A{contextual}; K{yes: deductive learning, grammar-based classification, semantic network extraction}; S{axiomatic, denotational}; APT{3-7: general approach}.*

5.4.2 Languages and models

There are a number of languages that form the basis for many a semantic model. Meier [101] offers a good overview of approaches by introducing a model of attack signatures for use on pattern detection systems. It is based on a metamodel for the semantics of database events by Zimmer and Unland [166]. The author identifies several types of information relevant for misuse detection: Exploit languages used to encode attacker actions, event languages that represent information to be analyzed by an IDS (e.g. HiPAC, SNOOP, NAOS, and ACOOD), detection languages used to describe signatures (rule-based (e.g. P-BEST), state-transition-based languages (e.g. STATL, IDIOT IDS), algebraic languages (e.g. LAMBDA, ADeLe, and Sutekh), response languages (alert information), and report languages such as the Intrusion Detection Message Exchange Format (IDMEF) [38].

For example, Totel et al. [143] correlate events from different IDS sources to combat the usually high number of false positives. The authors developed ADeLe, a language specifically tailored to describe exploits and attacks from the target's perspective in addition to the intrusion response. Their correlation model supports event sequences, (non-)occurrence, recurring events, and time constraints. While ADeLe is not a data provider or analysis system, its event correlation capabilities make it especially useful for describing multi-stage attacks. ⋄*Tags: G{intelligence, response}; T{host and network intrusion}; I{-}; D{-}; A{-}; K{no}; S{denotational}; APT{3-7: attack description language}.*

6 Results

In this section we investigate the general and statistical findings derived from this survey of 60 domain articles. This includes scoring per the quality assessment introduced in Sect. 3.4, a list papers identified as especially good fit for APT detection efforts, as well as the full categorization (see Sect. 4) of all solutions.

6.1 Findings summary

The mean date of publication of the surveyed papers is 2009. There are two deviations from the rule set by the publication date constraint (2003): the key papers [87] and [39] were included for their overall contribution to the field despite their more advanced age. See Table 1 for a breakdown of publication dates.

Interestingly, the focus on specific techniques changed little over the years. For example, system event traces are still widely used as basis for threat detection and analysis. Of the 30 papers published more recently (after 2010), 40 % process system events. This stands in contrast to a 41.7 % overall ratio. The same can be observed for a number of categories, including learning techniques. Even the use of semantics-based approaches did not significantly increase. Shy of 57 % of the post-2010 papers can be classified as at least partially context-based; this is only slightly higher than the overall average of 53.3 %. The trend towards semantic solutions is recognizable, but not notably so.

In contrast, 9 of the 13 top-scored (score > 7) papers that have been awarded the highest score in APT domain applicability (Q5) were published in recent years. This shows that the focus is slowly shifting towards the detection and analy-

Table 1 Publication date breakdown

Paper	Year	Paper	Year	Paper	Year	Paper	Year
[39]	2001	[26]	2007	[53]	2010	[2]	2013
[58]	2003	[110]	2007	[66]	2010	[12]	2013
[159]	2004	[164]	2008	[20]	2010	[127]	2013
[143]	2004	[27]	2008	[128]	2010	[138]	2013
[161]	2004	[113]	2008	[99]	2010	[121]	2014
[6]	2005	[129]	2008	[52]	2010	[102]	2014
[23]	2005	[114]	2008	[60]	2011	[156]	2014
[5]	2005	[132]	2008	[72]	2011	[148]	2014
[28]	2005	[24]	2009	[41]	2012	[3]	2014
[165]	2005	[1]	2009	[155]	2012	[91]	2014
[55]	2006	[119]	2009	[88]	2012	[17]	2014
[95]	2006	[54]	2009	[34]	2012	[11]	2014
[16]	2006	[13]	2009	[7]	2012	[65]	2014
[74]	2007	[124]	2009	[97]	2012	[163]	2014

sis of targeted attacks, even though many of the articles were not authored specifically with APTs in mind. As prime examples for techniques that can more easily be transferred to this new class of threat, these 13 papers are of special interest to answering research question R4 thanks to their determined scores. See Table 2 for a complete list.

In addition to particularly APT-relevant articles, highly graded works are of special interest as well. In Table 3, we take a closer look at papers scored 8.0 or higher. Because of the constraints defined in Sect. 3, all of the research has been classified with a TA (Q5) score of at least 0.5.

Delving deeper into the individual stages of a targeted attack, we see that exploitation, installation, C2, and action stages are roughly equally represented. We see 34 to 37 methodologies that directly or indirectly contribute to phases 4-7. The reason can be found in the nature of common malware: A larger number of malicious software variants include functionality that is comparable to the operations implied by the APT kill chain seen in Fig. 1. Initial (exploit) code execution, installation activity, remote communication and payload execution is not necessarily unique to targeted attacks and can be spotted by a wide range of tools that consider host or network activity.

The delivery phase (3) is in the scope of 50 % of the surveyed papers. However, as delivery may include common e-mail communication and local device activity, most of the solutions only indirectly consider it – more focused work specifically targeting delivery actions is still rare.

Table 2 Papers scoring highest for TA relevance (Q5 = 1.0; Overall >7.0)

Paper	Key research	Domain	Score
[72]	Host-side C2 traffic identification and behavior extraction through graph mining (Jackstraws)	Host	8.5
[124]	Heuristic classification of dynamically generated application traces (Malheur)	Host	8.5
[41]	Automatically created application profiles through graph-based function/parameter tracing	Host	7.0
[102]	Connection discovery for actors, machines, and malware through code semantics and function behavior (VirusBattle, BinJuice)	Host	7.0
[113]	Emulation of human reasoning process to distinguish user from attacker actions	Host	7.0
[121]	Detection and classification of web app attacks by means of an ontological model	Network	8.5
[163]	Detection of C2 traffic through malicious and benign HTTP traffic templates (BotHound)	Network	7.5
[7]	Attack ontology generation based on threat intelligence information and event fusion	Network	7.0
[12]	Behavior clustering through industry/location correlation based on URL strings (SpuNge)	Network	7.0
[148]	Flow-based traffic monitoring system capable of statistical anomaly detection	Network	7.0
[11]	Ontology-based data management system for central forensic data analysis (Gestalt)	Multi-source	7.5
[58]	Attacker intention modelling and traffic simulation system considering multi-source data	Multi-Source	7.0
[143]	Event correlation through attack/exploit language describing the target's view (ADeLe)	General	7.0

Table 3 Papers with the highest overall score (Overall >8.0)

Paper	Key research	Domain	Score
[28]	Static malware detection algorithm incorporating semantics and depicted as control-flow graphs	Host	8.5
[72]	Host-side C2 traffic identification and behavior extraction through graph mining (Jackstraws)	Host	8.5
[124]	Heuristic classification of dynamically generated application traces (Malheur)	Host	8.5
[27,52]	Malware classification and behavior extraction through dependence graphs (MiniMal, Holmes)	Host	8.0
[86]	Host-based function anomaly detection system with added context	Host	8.0
[156]	Data flow graphs depicting execution flow of various objects (DAVAST)	Host	8.0
[121]	Detection and classification of web application attacks through ontological model	Network	8.5

Reconnaissance (phase 1) is covered by 15 solutions, most of which are capable of analyzing web traffic that could yield insight into suspicious scanning activity or targeted information mining from publicly accessible resources. Stage 2 (weaponization) is currently not in the scope at all, as it takes place solely on the attacker's systems. Here, alternative approaches such as active intelligence into code reuse or post-attack attribution need to be investigated.

Taking a closer look at the distinctive semantic categories defined in Sect. 2.2, we most often see semantics-aware solutions (44 papers). This is followed by solutions that include formal denotations of languages or rules (denotational semantics, 14 papers). Axiomatic and operational semantics are still rarely used (6 and 4 papers, respectively).

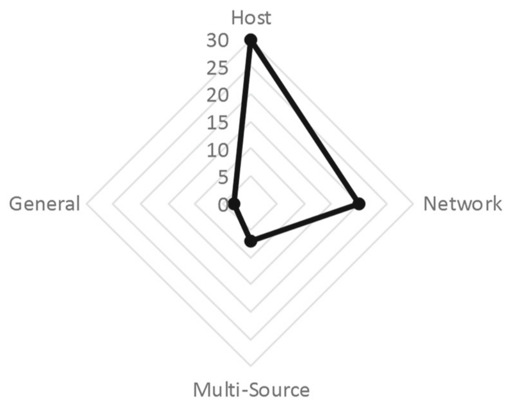
In Sect. 7, we discuss the implications of these findings and present a model of a workable ATA detection system based on key components identified in the surveyed literature.

6.2 Statistics

Primary domain (see 'General categorization') is a unique property used for initial *categorization*. Each article can be assigned a single domain. Of the 60 papers surveyed, 30 present a host-based solution, 20 focus on the network, 7 describe multi-source approaches, and only 3 could not be classified or were independent of a specific application.

Contributions most often encompass methods or models and a (prototypical) tool or system (39 papers each). 19 papers introduce a formalism or language while 15 papers revolve around a more general framework. See Fig. 4 for a breakdown chart and Table 4 for a full overview.

The goals of the respective solutions noticeably lean towards threat detection: 42 papers discuss approaches specifically aimed at detecting attacks or spotting attack indicators. Threat intelligence or classification components could be found in 26 articles. The actual analysis of a threat – done usually to determine its nature or goal – is an integral part of 19 papers, followed by 14 works that also consider threat correlation or event fusion. Only 6 solutions attempt to predict threats, and not a single one focuses on preventive measures. Threat response is also rarely found: only three articles claim to support follow-up activities to a previously detected attack.



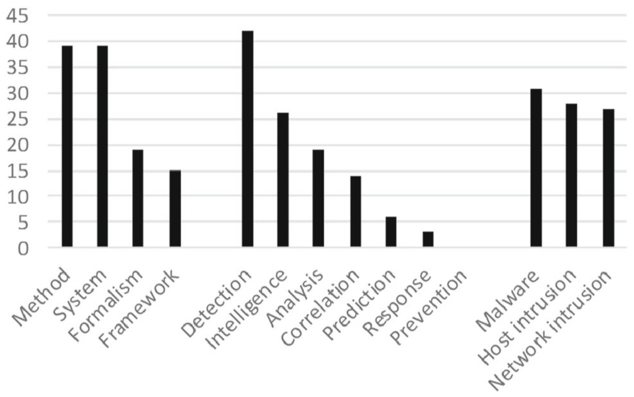


Fig. 4 (a) domain focus, and (b) general categories

Table 4 General category by paper

		[1]	[2]	[3]	[5]	[6]	[7]	[140]	[11]	[12]	[13]	[16]	[17]	[20]	[23]	[26]	[24]	[28]	[27]	[34]	[39]	[41]	[44]	[53]	[55]	[58]	[60]	[65]	[66]	[74]	[87]
Domain	Host					✓					✓	✓		✓	✓		✓	✓	✓	✓		✓	✓	✓			✓		✓	✓	✓
	Network	✓	✓	✓			✓	✓		✓						✓									✓	✓		✓			
	Multi-source								✓				✓								✓										
	General				✓																										
Contribution	Framework					✓		✓	✓	✓			✓								✓					✓					
	Model/method	✓	✓	✓	✓		✓					✓		✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓			✓	✓	✓
	Formalism		✓		✓		✓				✓	✓			✓			✓	✓	✓		✓									
	Tool/system	✓		✓		✓		✓	✓	✓	✓			✓		✓		✓					✓	✓	✓	✓	✓	✓		✓	
Goal	Prediction		✓	✓	✓																										
	Prevention																														
	Intelligence			✓	✓			✓	✓	✓	✓		✓						✓	✓	✓	✓				✓					
	Correlation		✓	✓					✓	✓						✓					✓										
	Detection	✓	✓	✓		✓	✓			✓		✓		✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓
	Analysis					✓		✓					✓				✓	✓			✓		✓	✓			✓	✓	✓		
	Response								✓																						
Threat type	Malware	✓									✓	✓	✓	✓	✓		✓	✓	✓			✓	✓	✓			✓	✓	✓	✓	
	Host intr.	✓	✓	✓	✓	✓			✓	✓		✓	✓		✓					✓	✓			✓		✓		✓			✓
	Network intr.	✓		✓	✓		✓	✓	✓	✓			✓			✓					✓				✓	✓		✓			

Table 4 continued

		[88]	[91]	[95]	[97]	[99]	[102]	[72]	[110]	[86]	[113]	[54]	[114]	[119]	[121]	[124]	[127]	[128]	[129]	[132]	[138]	[143]	[148]	[155]	[52]	[156]	[159]	[160]	[163]	[164]	[165]
Domain	Host	✓			✓		✓	✓		✓	✓		✓			✓				✓				✓	✓	✓	✓			✓	
	Network			✓					✓					✓	✓		✓	✓	✓				✓						✓		✓
	Multi-source		✓			✓						✓									✓										
	General																					✓						✓			
Contribution	Framework		✓			✓			✓			✓	✓			✓			✓	✓											
	Model/method	✓					✓	✓			✓	✓		✓	✓		✓	✓			✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	Formalism			✓						✓	✓				✓			✓			✓	✓			✓			✓			
	Tool/system		✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓					✓	✓	✓		✓	✓	✓	✓
Goal	Prediction										✓							✓													✓
	Prevention																														
	Intelligence		✓			✓	✓	✓				✓			✓	✓					✓	✓	✓	✓	✓			✓			✓
	Correlation		✓			✓						✓			✓		✓		✓		✓										✓
	Detection	✓		✓	✓			✓	✓	✓	✓		✓	✓	✓		✓	✓	✓				✓		✓	✓	✓		✓	✓	✓
	Analysis	✓					✓		✓						✓			✓		✓				✓		✓					
	Response																				✓	✓									
Threat type	Malware	✓			✓		✓	✓				✓	✓			✓			✓	✓				✓	✓	✓	✓		✓	✓	
	Host intr.		✓							✓	✓	✓	✓						✓		✓	✓				✓	✓	✓		✓	
	Network intr.		✓	✓		✓			✓					✓	✓		✓	✓	✓		✓	✓	✓					✓			✓

Table 5 Data collection capabilities by paper

		[1]	[2]	[3]	[5]	[6]	[7]	[140]	[11]	[12]	[13]	[16]	[17]	[20]	[23]	[26]	[24]	[28]	[27]	[34]	[39]	[41]	[44]	[53]	[55]	[58]	[60]	[65]	[66]	[74]	[87]
Data gathering	Dynamic	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	?	✓	✓		✓	✓	✓	✓	✓	✓	✓	
	Static																	✓					✓								
Approach	Function					✓					✓	✓			✓				✓	✓		✓		✓			✓	✓	✓	✓	
	Packet	✓	✓	✓		✓		✓		✓															✓	✓	✓	✓			
	Code													✓																	
	Disk																													✓	
	Memory										✓								✓											✓	
	Logs			✓			✓						✓								✓					✓					
Input type	Threat info						✓																								
	System logs			✓					✓				✓													✓					
	App logs			✓					✓				✓													✓					
	Netw. traffic	✓	✓	✓		✓		✓	✓	✓															✓	✓	✓	✓			
	Events traces					✓			✓		✓	✓			✓				✓	✓		✓		✓			✓	✓	✓	✓	
	Binary/raw													✓				✓					✓							✓	
	Alerts			✓			✓		✓				✓			✓					✓					✓					
Flow support	Available			✓								✓			✓							✓									
Environment	Native	✓	✓					✓		✓					✓			✓			✓						✓	✓			
	VM					✓							✓	✓				✓										✓	✓	✓	
	Emulation										✓							✓									✓				
	Unspecified			✓			✓					✓							✓	✓		✓		✓	✓	✓					

Table 5 continued

		[88]	[91]	[95]	[97]	[99]	[102]	[72]	[110]	[86]	[113]	[54]	[114]	[119]	[121]	[124]	[127]	[128]	[129]	[132]	[138]	[143]	[148]	[155]	[52]	[156]	[159]	[160]	[163]	[164]	[165]
Data gathering	Dynamic	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓			✓		✓
	Static						✓																				✓			✓	
Approach	Function	✓					✓	✓		✓			✓			✓				✓				✓	✓	✓	✓				
	Packet			✓					✓					✓	✓		✓	✓	✓		✓		✓						✓	✓	
	Code									✓										✓							✓				
	Disk				✓								✓																		
	Memory												✓																		
	Logs							✓									✓				✓										
Input type	Threat info																✓														
	System logs					✓					✓	✓																			
	App logs		✓			✓					✓	✓									✓										
	Netw. traffic			✓					✓		✓			✓	✓		✓	✓	✓		✓		✓						✓	✓	
	Event traces	✓					✓	✓		✓	✓	✓	✓			✓				✓				✓	✓	✓					
	Binary/raw				✓		✓			✓			✓							✓							✓			✓	
	Alerts					✓											✓											✓			✓
Flow support	Available								✓												✓		✓				✓				
Environment	Native								✓					✓	✓	✓	✓	✓	✓				✓								
	VM						✓	✓					✓			✓				✓						✓			✓		
	Emulation				✓			✓																✓	✓						
	Unspecified	✓		✓						✓											✓						✓				

Table 6 Detection and analysis capabilities by paper

		[1]	[2]	[3]	[5]	[6]	[7]	[140]	[11]	[12]	[13]	[16]	[17]	[20]	[23]	[26]	[24]	[28]	[27]	[34]	[39]	[41]	[44]	[53]	[55]	[58]	[60]	[65]	[66]	[74]	[87]
Detection appr.	Dynamic	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	?		✓		✓	✓	✓	✓	✓	✓	✓	✓
	Static																	✓					✓								
Detect. method	Anomaly											✓		✓	✓					✓		✓	✓		✓						
	Pattern	✓			✓	?	✓	✓			✓		✓					✓						✓		✓	✓	✓			✓
	Similarity			✓						✓	✓																				
	Graph																		✓			✓									
	Ontology	✓	✓		✓		✓		✓			✓				✓	✓												✓		
Analysis techn.	Attribute	✓						✓																				✓		✓	
	Behavioral					?					✓		✓	✓	✓		✓	✓	✓			✓	✓	✓	✓	✓	✓		✓		✓
	Contextual	✓	✓	✓	✓		✓		✓	✓		✓		✓		✓	✓	✓		✓	✓								✓		
Temporal dom.	Real-time																								?					✓	
	Delayed					?			?																			✓			
	Interval							✓																							
	On demand	✓								✓	✓		✓	✓	✓	✓		✓	✓			✓	✓	✓			✓		✓		✓
	Unspecified		✓	✓	✓	✓	✓					✓					✓			✓						✓					
Processing	Local			✓	?							?		✓	✓		✓	✓	✓	?		?	✓	✓							✓
	Centralized		?	✓		✓	?	✓	✓	✓	✓					✓									?	✓	✓	✓	✓	✓	
	Distributed	✓											✓																		

Table 6 continued

		[88]	[91]	[95]	[97]	[99]	[102]	[72]	[110]	[86]	[113]	[54]	[114]	[119]	[121]	[124]	[127]	[128]	[129]	[132]	[138]	[143]	[148]	[155]	[52]	[156]	[159]	[160]	[163]	[164]	[165]
Detection appr.	Dynamic	✓		✓	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓		✓
	Static						✓												✓	✓										✓	
Detect. method	Anomaly		✓	✓					✓	✓													✓				✓				
	Pattern		✓			✓		✓	✓			✓						✓	✓	✓						✓			✓	✓	
	Similarity					✓	✓	✓				✓						✓											✓		✓
	Graph	✓						✓																✓	✓						
	Ontology													✓	✓		✓			✓								✓			✓
Analysis techn.	Attribute	✓					✓												✓	✓									✓		
	Behavioral			✓				✓		✓						✓				✓				✓	✓	✓			✓	✓	
	Contextual		✓			✓	✓	✓	✓	✓				✓	✓		✓	✓	✓		✓		✓	✓			✓	✓			✓
Temporal dom.	Real-time			✓		✓			✓					✓	✓			✓					✓			✓			✓		
	Delayed				✓														✓												✓
	Interval																														
	On demand	✓					✓	✓								✓	✓		✓	✓				✓	✓	✓	✓			✓	✓
	Unspecified									✓																		✓			
Processing	Local	?		?			✓			✓						✓			✓	✓							✓	✓		✓	
	Centralized				✓	✓		✓	✓					✓	✓		✓	✓					✓	✓	✓	✓			✓		?
	Distributed																														

Table 7 Knowledge generation through learning and classification

		[2]	[3]	[5]	[7]	[140]	[12]	[13]	[16]	[20]	[23]	[27]	[34]	[41]	[44]	[58]	[88]	[91]	[110]	[86]	[119]	[121]	[124]	[132]	[148]	[155]	[156]	[160]	[163]	[165]	[54]	[72]	[52]
Learning	Supervised	✓							✓	✓	✓		✓	✓	✓	✓			✓	✓									✓	✓	✓	✓	
	Unsupervised																	✓					✓										
	Deductive			✓	✓							✓									✓	✓						✓			✓		
Classification/	SVM					✓												✓												✓			
clustering	Decision tree		✓												✓			✓															
	Neural netw.												✓																	✓			
	Nearest protot.							✓															✓										
	Bayesian	✓		✓												?		✓			✓												
	Markov								✓				✓							✓													
	Grammars												✓	✓														✓					
	Outlier																		✓														
	Change detect.																								✓								
	Hierarch. clust.						✓																✓										
Extraction	Depend. graph											✓														✓							✓
	Behavior graph										✓			✓			✓							✓						✓		✓	
	Semantic netw.	✓	✓																									✓					
Visualization	Available											✓		✓		✓								✓			✓						

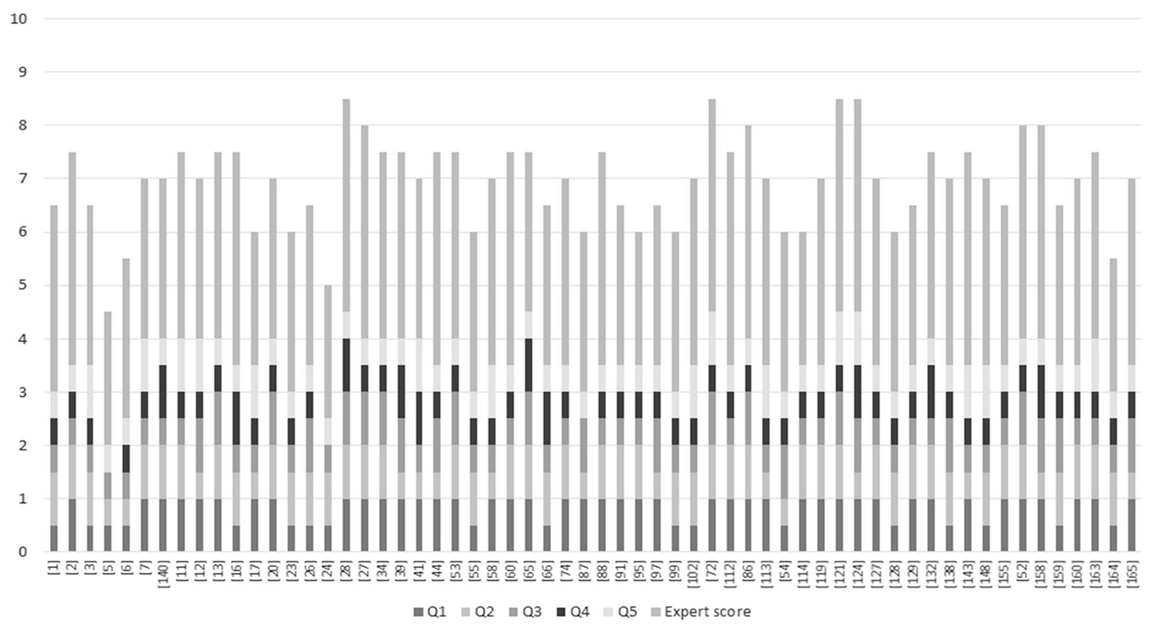


Fig. 5 QA and overall scores for each surveyed paper

In contrast to the diverse goals of the surveyed approaches, the type of threats combated is almost evenly distributed between malware (31 papers), host intrusions (28 papers), and network intrusions (27 papers). This shows that countering each means of attack is considered equally important by the research community. See Table 4 for details.

Data collection is distinctively dominated by dynamic methods. 51 papers describe dynamic techniques to acquire input data. While data gathering capabilities do not necessarily mean that the solution records information by itself, a large number of solutions (47 papers) do in fact include monitoring functionality or at least a means to directly import results from integrated tools.

Static retrieval is used by 5 solutions, whereas 5 approaches do not collect data at all. 23 solutions dynamically monitor functions that are often logged as event traces. In contrast, 21 approaches sniff, record, or interpret network packets. Conventional logging is used in 8 cases. Code, disk, and memory monitoring are used least often – only three to four systems directly observe respective activities. In terms of type, capabilities mostly correspond to the approach identified above: We most often see system event traces (25 papers) and network traffic (23 papers). External alerts (11 papers), system and application logs (7 and 9 papers, respectively), and direct binary (raw) data input (11 papers) are used less often. General threat information is only collected in 2 cases.

Flow functionality is usually found in network-domain articles. Of the 47 papers that monitor information, 8 utilize data flow processing.

With the exception of emulation (7 papers), monitoring environments almost equally encompass native (17) and VM (14) systems. Interestingly, 14 papers do not specify a particular environment. The reason can be found in the nature of the works – many solutions are not yet part of an operational system but only roughly sketch a proposed functionality. Also, there exist a number platform-independent implementations. See Table 5 for details.

A total of 56 papers describe either *analysis* or *detection* functions – or both. Similar to data gathering, detection also leans towards dynamic techniques were the attack or sample under scrutiny is executed on an isolated system or is let loose in a testing network. More often than not, this analysis environment is also the one that monitors abovementioned data.

Pattern-based systems are most common (24 papers). Anomaly detection is used in 13 cases. This is only surpassed by ontologies – 15 solutions use various ontology models, languages, and tools to describe and detect a multitude of threats. Since this paper expressively focuses on at least semantics-aware approaches, this number is not surprising. In accordance, distinctively semantics-based or context-aware solutions are described in over 50 % of the papers. Less formal behavioral approaches (26 papers) are almost as common. At the same time the trend shows that attribute-based detection is on the decline (9 papers). This mirrors the opinion of many researchers that signature-based detection found for example in AV software is slowly becoming obsolete [21,96,111,118].

On the processing side, on-demand analysis (28 papers) on either the local system (23 papers) or a central server (29 papers) is most widely used. Still, a total of 11 works claim to have achieved real-time processing. See Table 6 for details.

Knowledge generation is part of 32 of the surveyed solutions. Supervised learning is most widely used (14 papers), followed by deductive learning (7) and unsupervised approaches (2 papers). The specific methods used are diverse and do not lean towards one specific approach. Bayes-based systems are slightly more common than e.g. outlier or change detection (5 vs. 1 paper). On the extraction side, we most often see behavior graphs. Only five solutions offer post-analysis visualization. Part of the reason is the explicit exclusion of pure visualization systems. See Table 7 for a list of papers with knowledge generation capabilities.

6.3 Scores

Quality assessment was conducted through scores ranging from 0 to 10. The average score awarded to the reviewed papers is 6.94. Eight of the articles received an excellent score of 8 points or higher, while 38 papers were given a solid 7-point or above rating. Interestingly, the QA average (3.59 of 5) is slightly higher than the average expert rating (3.35).

In direct response to the QA questions, we have calculated average scores and assessed the number of papers that achieved the maximum score of 1 point for the respective category (see Fig. 5). It stands to notice that the attack domain (Q2) was usually well defined. 44 papers were given the full point; the average score was close to 0.87. In contrast, the definition and explanation of the type of operational data (Q4) used by the solutions was often found lacking. Only ten papers received the full score, resulting in an average grade of only 0.56. In regards to targeted attacks, 15 papers were found to be especially well-suited for APT defense-related objectives. See Fig. 5 for a full score breakdown of all reviewed papers and Tables 2 and 3 for top picks.

7 Discussion

7.1 Research questions

In this study, we have addressed 4 specific research questions that are now evaluated in this section. The overarching goal of the paper was to provide an overview of approaches and methods that could be employed to strengthen an organization's defense against advanced persistent threats and targeted attacks in general. Effective solutions tailored to the ATA/APT realm are very rare, presenting researchers with ample opportunity to develop specialized solutions that transport conventional cyber-defense mechanisms to this new threat domain.

In answer to R1 ("Which models, frameworks, formal definitions, and tools exist to describe information system attacks?"), we have identified various models, frameworks, formal definitions, and tools that describe information system attacks. The solutions and techniques they employ can serve as foundation for the design or technical implementation of a system capable of identifying targeted attacks at various levels: the host, the network, or a custom combination thereof. Armed with this knowledge, future research can be focused and prioritized in accordance to the user's specific needs in terms of data formats, specific approaches to gathering and monitoring as well as knowledge generation capabilities. A detailed evaluation and overview of all reviewed solutions can be found in Sect. 6.

In response to R2 ("Which semantics-aware and semantics-based tools and techniques exist to detect and evaluate attacks?"), we have highlighted solutions that use a semantic approach to perform their primary task. Semantics-aware tools are of significant interest when they establish attack context (see also contextual analysis technique in Table 6) and thereby enable research into the still problematic differentiation between common and targeted attacks. Semantics-based approaches offer additional insight and are often able to link actors and assets with a specific attack action. Enabled by the semantic categorization introduced in Sect. 2.2, we have highlighted solutions that focus on execution correctness/state transition, correlation through rules or I/O inference, as well as denotational approaches. This can help to e.g. better understand attacks and their consequences, identify high-level goals, or determine existing flaws in the defender's security design or implementation.

Techniques useful to analyzing targeted attacks are identified through their Q5 score as well as their APT stage affinity tag. This addresses research question R3 ("What are promising approaches to ATA detection and how can they be classified?") by highlighting solutions deemed promising for APT/ATA detection and analysis. With semantics-based methodologies in the minority, there is still room for improvement in the areas of formalization as well as axiomatic and operational semantics. Next to awareness of meaning, knowledge generation was determined to be of utmost importance. Many papers offer functionality enabling the extraction of certain behavioral particularities that support the process of learning typical APT activity.

In accordance to R4, we determined which information is actually most helpful to the general task of detecting and understanding ATAs by seeking to encompass all primary categories (see 4. In combination with the aforementioned APT stage tags, it becomes possible to maximize the scope through diversity of approach. With the aim to put together a system design checklist for a successful holistic defense strategy (see Table 8 for details), we infer that:

– The highest possible number of primary goals (G) should to be considered for maximum breadth of defense;

– Every threat type T needs to be countered as they are all part of a typical ATA;
– Each data input type I should be taken into consideration for a complete view on the system and its neighboring network infrastructure;
– Detection methods D and analysis techniques A need to be diverse to maximize obfuscation and evasion resilience;
– Knowledge generation K should be part of at least one solution per threat type so that the defender does not only detect, but also learn from adversary action on both host and network level;
– Each APT stage per Hutchins' model [67], possibly excluding weaponization, should be covered.

This can either be achieved through a combination of approaches – ultimately requiring correlation – or by utilizing a system general enough to be flexible in its application. The necessary information to assemble such a defense framework can be extracted from the reviewed and scored papers and is further detailed in the subsection below. We focus specifically on the kind of information required to identify targeted attacks and show how a system considering most of the APT stages as well as a diverse range of data collection and analysis methods could look like. To this end, we conceptualized the approach starting from the choice of data provider (data collection approach as summarized in Table 5) up to the correlation and interpretation of events. The resulting roadmap is intended to assist further research into applied ATA defense and other domain-aware countermeasures.

7.2 APT defense framework

The approaches discussed in this study can be used to assemble a model of a system capable of dealing with a variety of targeted attack scenarios. While other defense mechanisms only return isolated events or counter single attacks, a holistic system would interpret the collected information and come to a more detailed verdict. Each satisfied item in our checklist would increase the confidence in the result.

The following roadmap details a suggested design process for a conceptual ATA defense system based on the solutions reviewed in this survey. Please keep in mind that a concrete technical implementation is considered future research and not part of this paper.

We differentiate the following stages: Threat definition and modeling, formalization and ontology building, as well as data provider selection.

7.2.1 Threat definition and modeling

Prior to implementation, the defending organization will have to define both assets and threats. For threat definition, we found the model by Giuara and Wang [56] (see Fig. 1) and the more common cyber kill chain by Hutchins et al. [67] to be simple yet effective solutions for modeling targeted attacks. The decision of which model to use largely depends on personal preference and data exchange requirements: While the cyber kill chain model considers command and control activity and weaponization as separate stages, Giuara's model is more detailed when it comes to the collection of data. Reconnaissance, exploitation, operation, and exfiltration stages are mostly identical, albeit named differently at times. Both models can be used in conjunction with MITRE's APT-enabled Structured Threat Information eXpression (STIX) data exchange format [109], which was developed to represent threat information in a comprehensive manner.

Before existing data sources can be combined and interpreted, the defender also needs to evaluate their own assets in order to determine likely targets. This organizational step goes hand in hand with an assessment of possible attacker goals and methods. The resulting top-down view on a potential targeted attack is the foundation for subsequent ontology building and goal mapping. For initial actor and asset evaluation, we propose the use of goal modeling techniques: KAOS [122], GRL [147], and the i* Strategic Dependency (SD) model [46] are promising candidates for implementation. Subsequent technical solutions will want to add reasoning to the mix – this is where ontologies come into play.

7.2.2 Formalization and ontology building

Ontologies were identified as a promising way of approaching the challenge of formalizing threats and threat responses in a semantics-aware manner. Evaluating the top results shown in Tables 2 and 3, we can see that ontologies and related semantics-based methods are among the more highly scored solutions [7,11,102,113,120].

The information security community has only in recent years begun to truly embrace the concept. Originally a discipline of philosophy, ontologies in information science have become a formal approach to describing data types, properties, and interrelationships of entities within a specific domain. Their reasoning capabilities and data formats set them apart from semantics-unaware relational databases. Depending on general requirements and desired granularity, system designers can choose from numerous languages or systems. Data formats and languages include RDF, OWL, OWL-DL, and SWRL, among others. In the reviewed papers, ontology building and design often relies on established implementations like Protégé [136] or Apache JENA [8]. Queries to an ontological system are usually written in SPARQL [152].

For an ATA detection system to succeed, research will have to bridge the gap between pure formalization and event interpretation by combining various data sources into a domain-complete attack ontology. Only then can we collect all available information and assign it a place in the greater picture. Fox et al. [51] emphasize that an ontology needs to be designed with a number of competency question in mind. Each question is representative for the information the completed ontology is supposed to answer. For designing an APT defense system, these questions would have to revolve around data providers and specific attack activities:

- Which data providers do I need to utilize in order to be able to detect a satisfyingly high number of targeted attack stages?
- Which data provider is able to detect which attack stage or activity?
- What are the techniques used in an activity and how do they translate to data provider events?
- Is the current activity indicative of a targeted attack, and if yes, which?
- Who are the actors of the attack and which assets are targeted/affected?

Fused into one system and complemented by a suitable reasoning engine such as the ones offered by various Protégé plugins [136], such an ontology would be able to answer straightforward, natural-language questions about any hypothetical or suspected ongoing attack. For this to work, it is necessary to fill the ontology with detailed knowledge of the attacks and assets identified in design stage 1. Only then can we move on to the selection and implementation of data providers.

7.2.3 Data provider selection

Once the ATA ontology has been defined and supplied with the necessary knowledge as well as inference rules, we can begin to link each attack stage or activity to specific events. As argued by Hutchins et al. [67], countering the different stages of an attack will require both analysis and detection systems. The asset owner will have to choose a number of data providers capable of collecting both host-based as well as network-based information. It is prudent to employ systems that, put together, encompass as many types of input types (see Table 5) as possible.

The reasons are manifold: Reconnaissance and delivery will likely involve networked resources and may not even register on the host. Since different layers of an organization's network may be accessed or analyzed by the attacker without him actually copying or manipulating resources, the traffic passing through the LAN as well as the implied general behavior is of particular interest. Flow-based approaches would help identify anomalous communication patterns while suspicious web traffic could be a sign of initial probing activities. Alerts that are harmless by themselves could be correlated to other findings using a dedicated SIEM-like system.

Sooner or later during exploitation, installation, or action stages, the attacker will interact with a local system. Here, host monitoring and malware detection comes into play. Again, the defender will not have to reinvent the wheel to protect himself from isolated malicious activity. The true challenge is registering events that do not appear to be harmful by themselves, but might be part of a larger attack. While data collection and correlation are well-researched, mapping individual events to a greater picture is still a challenge. This study shows that attack semantics is a term used in many a work, but that there is often a significant gap between promise and delivery.

Source data will also be one of the key success factors of any practical APT defense implementation – inadequate or too low a number of data providers will make it nigh impossible to identify attacks affecting specific or multiple targets. On the other hand, it might not be feasible to include too many sources lest the system would experience slowdowns or a decrease in interpretation accuracy. An effective implementation will have to carefully consider available solutions and hand-pick a small number of providers suited to the respective task. This survey identified monitoring and data classification [12,27,121,124,148,156] as well as attack description, profiling and extraction to be a vital part of this stage of ATA identification [41,72,143,156]. Various detection systems spread across all the primary detection domains will help to assemble the picture [28,148,163].

To further support data provider selection and to offer a simplified view on the surveyed solutions, we introduce a design checklist based on the categorization used in this study (see Table 8). Each key property has been awarded a minimum coverage threshold (in square brackets) that should be satisfied by the chosen data providers.

Once data providers have been selected based on specific needs, the actual semantic engine can be built around the objective of detecting, explaining, and locating targeted attacks. Developing such a system will undoubtedly be a major research challenge in the coming years.

7.3 Limitations of the study

This study slightly deviated from the guidelines presented by Kitchenham [82] as the search process was only partially automated and relied on some manual corrections to search strings and sources. We also refrained from including papers beyond the 50th result of the respective search engine. This happened only rarely since, thanks to search term refinement, the number of results was usually lower than that.

The post-review exclusion of articles with a low total score is also not cited as standard practice but helped to keep the

Table 8 System design checklist

	Prediction	Prevention	Intelligence	Correlation	Detection	Analysis	Response
G met [5/7]							
	Malware	Host intr.	Network intr.		Attribute	Behavior	Context
T countered [3/3] K for T [3/3] Correlation for T [3/3]				A used [3/3] G for A [3/3] Corr.: A [3/3]			
	Threat info	System logs	App logs	Netw. traffic	Events traces	Binary/raw	Alerts
I incorporated [5/7]							
	Recon	Weaponiz.	Delivery	Exploitation	Installation	C2	Actions
APT stage covered [5/7]							

G goal, *T* threat type, *A* analysis technique, *K* knowledge generation, *I* input data type

number of surveyed papers to a manageable minimum. Furthermore, we violated the publication year constraint in two cases where we felt that an exception was appropriate for the sake of completeness. Please see Table 1 for a full date breakdown.

To limit the scope of the paper, the articles reviewed in this survey have solely been chosen using the criteria defined in Sect. 3. Additional literature catering to specific topics not covered by the initial search terms was usually not considered, even if certain aspects were later identified as potentially relevant, such as solutions focusing primarily on knowledge generation.

8 Conclusion

The detection of advanced targeted attacks is highly interwoven with more conventional approaches to intrusion or malware detection. In this study, we identified 60 articles that introduce methods, models, frameworks, formalisms, or systems that could potentially contribute to the defense against APTs and other multi-stage cyber-attacks.

With ontologies and general semantics-based approaches, an increasing number of attack models have found their way into the field of information security. The shift towards ATA-aware solutions is noticeable, but not as pronounced as suggested by many a title. We identified only a total of 13 papers (see Table 2) that could contribute significantly to the fight against advanced attackers, while the remainder provides tools in the fight against more common or specific cyber-threats. Still, each of the reviewed articles has something unique to offer and should be considered when developing new systems.

To simplify prioritization, the introduced categorization enables researchers to conveniently browse for solutions best suited to their particular endeavor. This is complemented by the design concept of an defense framework which uses input from various data sources to detect and analyze a targeted attack. Thanks to our simple but comprehensive checklist, the task of designing such a system has become considerably easier.

8.1 Future research

This study introduces a number of future research opportunities. Based on data provider classification, it becomes possible to determine the exact type and nature of practical solutions needed to detect APTs with confidence. We are currently working on an OWL-based ontology encompassing categorized data providers, involved actors, utilized techniques, and concrete system events all linked together by a cyber kill chain model. Especially the interpretation of truly complex, multi-stage attacks is still largely unexplored. In future works, we will thoroughly define the requirements and capabilities of a full APT detection system and will introduce a detailed model that builds upon the results of this structured survey.

Minimizing the number of data providers needed to interpret attacks without compromising the accuracy of the results will be another predominant research challenge in the near future. New monitoring and analysis tools will have to consider ATA scenarios from the get-go and increasingly utilize data correlation in order to bring the three domains of host-based, network-based, and multi-source detection closer together.

Acknowledgements Open access funding provided by FH St. Pölten - University of Applied Sciences. The financial support by the Austrian Federal Ministry of Science, Research and Economy and the National Foundation for Research, Technology and Development is gratefully acknowledged.

References

1. Abdoli, F., Kahani, M.: Ontology-based distributed intrusion detection system. In: Computer Conference, 2009. CSICC 2009, 14th International CSI, IEEE, pp. 65–70 (2009)
2. AlEroud, A., Karabatis, G.: A system for cyber attack detection using contextual semantics. In: 7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing, pp. 431–442, Springer, New York (2013)
3. Aleroud, A., Karabatis, G.: Context Infusion in Semantic Link Networks to Detect Cyber-attacks: A Flow-Based Detection Approach. pp. 175–182. IEEE (2014). doi:10.1109/ICSC.2014.29
4. Alienvault: OSSIM: The Open Source SIEM | AlienVault. https://www.alienvault.com/products/ossim. Accessed 29 July 2015
5. Anagnostopoulos, T., Anagnostopoulos, C., Hadjiefthymiades, S.: Enabling attack behavior prediction in ubiquitous environments. In: Pervasive Services, 2005. ICPS'05, Proceedings of International Conference on, pp. 425–428. IEEE (2005)
6. Andersson, S., Clark, A., Mohay, G., Schatz, B., Zimmermann, J.: A framework for detecting network-based code injection attacks targeting Windows and UNIX. In: Computer Security Applications Conference, 21st Annual, pp. 10. IEEE (2005)
7. Ansarinia, M., Asghari, S.A., Souzani, A., Ghaznavi, A.: Ontology-based modeling of DDoS attacks for attack plan detection. In: 2012 6th International Symposium on Telecommunications (IST), pp. 993–998. IEEE (2012)
8. Apache Software Foundation: Apache JENA. https://jena.apache.org/. Accessed 27 July 2015
9. Apecechea, G.I., Inci, M.S., Eisenbarth, T., Sunar, B.: Fine grain Cross-VM Attacks on Xen and VMware are possible! IACR Cryptology ePrint Archive p. 248 (2014)
10. Apostolico, A., Guerra, C.: The longest common subsequence problem revisited. Algorithmica **2**(1–4), 315–336 (1987)
11. Atighetchi, M., Griffith, J., Emmons, I., Mankins, D., Guidorizzi, R.: Federated Access to Cyber Observables for Detection of Targeted Attacks. pp. 60–66. IEEE (2014). doi:10.1109/MILCOM.2014.15
12. Balduzzi, M., Ciangaglini, V., McArdle, R.: Targeted attacks detection with spunge. In: 11th Annual International Conference on Privacy, Security and Trust (PST), 2013, pp. 185–194. IEEE (2013)
13. Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable, Behavior-Based Malware Clustering. In: NDSS, vol. 9, pp. 8–11. Citeseer (2009)
14. Bayer, U., Kruegel, C., Kirda, E.: TTAnalyze: A Tool for Analyzing Malware. EICAR, pp. 180–192 (2006)
15. BBN Technologies: Asio BBN. http://asio.bbn.com. Accessed 27 July 2015
16. Bhatkar, S., Chaturvedi, A., Sekar, R.: Dataflow anomaly detection. In: 2006 IEEE Symposium on Security and Privacy, pp. 15–pp. IEEE (2006)
17. Bhatt, P., Yano, E.T., Gustavsson, P.: Towards a Framework to Detect Multi-stage Advanced Persistent Threats Attacks. pp. 390–395. IEEE (2014). doi:10.1109/SOSE.2014.53
18. Bilge, L., Dumitras, T.: Before we knew it: an empirical study of zero-day attacks in the real world. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 833–844. ACM (2012)
19. Bond, M.D., McKinley, K.S.: Probabilistic calling context. In: ACM SIGPLAN Notices, vol. 42, pp. 97–112. ACM (2007)
20. Bond, M.D., Srivastava, V., McKinley, K.S., Shmatikov, V.: Efficient, context-sensitive detection of real-world semantic attacks. In: Proceedings of the 5th ACM SIGPLAN Workshop on Programming Languages and Analysis for Security, p. 1. ACM (2010)
21. Bott, E.: Why malware networks are beating antivirus software. http://www.zdnet.com/article/why-malware-networks-are-beating-antivirus-software/. Accessed 29 July 2015
22. Caswell, B., Beale, J.: Snort Intrusion Detection 2.0. Syngress (2003)
23. Chaturvedi, A., Bhatkar, S., Sekar, R.: Improving attack detection in host-based IDS by learning properties of system call arguments. In: Proceedings of the IEEE Symposium on Security and Privacy, Citeseer (2005)
24. Chiang, H.S., Tsaur, W.J.: Ontology-based Mobile Malware Behavioral Analysis. Da-Yeh University, Changhua (2009)
25. Chien, E., OMurchu, L., Falliere, N.: W32. Duqu: the precursor to the next stuxnet. In: Proceedings of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET) (2012)
26. Chien, S.H., Chang, E.H., Yu, C.Y., Ho, C.S.: Attack subplan-based attack scenario correlation. In: 2007 International Conference on Machine Learning and Cybernetics, vol. 4, pp. 1881–1887. IEEE (2007)
27. Christodorescu, M., Jha, S., Kruegel, C.: Mining specifications of malicious behavior. In: Proceedings of the 1st India Software Engineering Conference, pp. 5–14. ACM (2008)
28. Christodorescu, M., Jha, S., Seshia, S., Song, D., Bryant, R.E.: others: Semantics-aware malware detection. In: Security and Privacy, 2005 IEEE Symposium on, pp. 32–46. IEEE (2005)
29. Cisco: Cisco IOS NetFlow. http://cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html. Accessed 29 July 2015
30. Cisco: Snort.Org. https://www.snort.org/. Accessed 29 July 2015
31. Colmerauer, A., Roussel, P.: The Birth of Prolog. In: History of programming languages–II, pp. 331–367. ACM (1996)
32. Consel, C., Danvy, O.: Static and dynamic semantics processing. In: Proceedings of the 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 14–24. ACM (1991)
33. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). doi:10.1007/bf00994018
34. Creech, G., Hu, J.: A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. Comput. IEEE Trans. **63**(4), 807–819 (2014)
35. Cuckoo Foundation: Automated Malware Analysis - Cuckoo Sandbox. http://www.cuckoosandbox.org/. Accessed 29 July 2015
36. Dalla Preda, M., Christodorescu, M., Jha, S., Debray, S.: A semantics-based approach to malware detection. ACM Trans. Program. Lang. Syst. **30**(5), 1–54 (2008)
37. De Vries, J., Hoogstraaten, H., van den Berg, J., Daskapan, S.: Systems for Detecting Advanced Persistent Threats: A Development Roadmap Using Intelligent Data Analysis. In: Cyber Security (CyberSecurity), 2012 International Conference on, pp. 54–61. IEEE (2012)
38. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format (IDMEF). https://www.ietf.org/rfc/rfc4765.txt. Accessed 29 July 2015
39. Debar, H., Wespi, A.: Aggregation and correlation of intrusion-detection alerts. In: Recent Advances in Intrusion Detection, pp. 85–103. Springer, New York (2001)

40. Dell SecureWorks: Truman. http://www.secureworks.com/cyber-threat-intelligence/tools/truman/. Accessed 29 July 2015
41. Dolgikh, A., Nykodym, T., Skormin, V., Birnbaum, Z.: Using behavioral modeling and customized normalcy profiles as protection against targeted cyber-attacks. In: Computer Network Security, pp. 191–202. Springer, New York (2012)
42. Dornhackl, H., Kadletz, K., Luh, R., Tavolato, P.: Malicious behavior patterns. In: 2014 IEEE 8th International Symposium on Service Oriented System Engineering (SOSE), pp. 384–389. IEEE (2014)
43. Duarte, J.M., Santos, J.B., Melo, L.C.: Comparison of similarity coefficients based on RAPD markers in the common bean. Genet. Molecular Biol. **22**(3), 427–432 (1999)
44. Dube, T.E., Raines, R.A., Grimaila, M.R., Bauer, K.W., Rogers, S.K.: Malware target recognition of unknown threats. IEEE Syst. J. **7**(3), 467–477 (2013). doi:10.1109/JSYST.2012.2221913
45. Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv. (CSUR) **44**(2), 6 (2012)
46. Eric, S.Y.: Social Modeling and i*. In: Conceptual Modeling: Foundations and Applications, pp. 99–121. Springer, New York (2009)
47. Falliere, N., Murchu, L., Chien, E.: W32.Stuxnet.Dossier. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier. Accessed 18 Sept 2015
48. Farrell, J.A.: http://www.cs.man.ac.uk/~pjj/farrell/comp2.html#EBNF. Accessed 29 July 2015
49. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: Methontology: from ontological art towards ontological engineering. In: AAAI Symposium on Ontological Engineering. American Association for Artificial Intelligence (1997)
50. FireEye: FireEye Malware Analysis. https://www.fireeye.com/products/malware-analysis.html. Accessed 29 July 2015
51. Fox, M.S., Barbuceanu, M., Gruninger, M.: An organisation ontology for enterprise modelling: preliminary concepts for linking structure and behaviour. In: Proceedings of the 4th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1995, pp. 71–81. IEEE (1995)
52. Fredrikson, M., Jha, S., Christodorescu, M., Sailer, R., Yan, X.: Synthesizing near-optimal malware specifications from suspicious behaviors. In: 2010 IEEE Symposium on Security and Privacy (SP), pp. 45–60. IEEE (2010)
53. Fukushima, Y., Sakai, A., Hori, Y., Sakurai, K.: A behavior based malware detection scheme for avoiding false positive. In: Secure Network Protocols (NPSec), 2010 6th IEEE Workshop on, pp. 79–84. IEEE (2010)
54. Gabriel, R., Hoppe, T., Pastwa, A., Sowa, S.: Analyzing Malware Log Data to Support Security Information and Event Management: Some Research Results. pp. 108–113. IEEE (2009). doi:10.1109/DBKDA.2009.26
55. Gamer, T., Scholler, M., Bless, R.: A granularity-adaptive system for in-network attack detection. In: Proceedings of the IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation, pp. 47–50 (2006)
56. Giura, P., Wang, W.: A context-based detection framework for advanced persistent threats. In: Cyber Security (CyberSecurity), 2012 International Conference on, pp. 69–74. IEEE (2012)
57. GlobalSecurity.org: Open Source Information System (OSIS). http://www.globalsecurity.org/intell/systems/osis.htm. Accessed 29 July 2015
58. Gorodetski, V., Kotenko, I., Karsaev, O.: Multi-agent technologies for computer network security: attack simulation, intrusion detection and intrusion detection learning. Comput. Syst. Sci. Eng. **18**(4), 191–200 (2003)
59. Greenberg, A.: Russians fingered for 'Uroburos' spy malware campaign, went undetected for years - SC Magazine. http://www.scmagazine.com/russians-fingered-for-uroburos-spy-malware-campaign-went-undetected-for-years/article/336570/. Accessed 29 July 2015
60. Grégio, A.R., Fernandes Filho, D.S., Afonso, V.M., Santos, R.D., Jino, M., de Geus, P.L.: Behavioral analysis of malicious code through network traffic and system call monitoring. In: SPIE Defense, Security, and Sensing. International Society for Optics and Photonics (2011)
61. Guarino, N., Welty, C.A.: An overview of OntoClean. In: Handbook on ontologies, pp. 201–220. Springer, New York (2009)
62. He, P., Karabatis, G.: Using semantic networks to counter cyber threats. In: Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on, pp. 184–184. IEEE (2012)
63. Herman, L.: Malware Attack at US Health Organization Went Undetected for 2 Years. http://www.hackbusters.com/news/stories/187232-malware-attack-at-us-health-organization-went-undetected-for-2-years. Accessed 20 Oct 2015
64. Hex-Rays: IDA: About. https://www.hex-rays.com/products/ida/. Accessed 29 July 2015
65. Hirono, S., Yamaguchi, Y., Shimada, H., Takakura, H.: Development of a Secure Traffic Analysis System to Trace Malicious Activities on Internal Networks. pp. 305–310. IEEE (2014). doi:10.1109/COMPSAC.2014.41
66. Huang, H.D., Chuang, T.Y., Tsai, Y.L., Lee, C.S.: Ontology-based intelligent system for malware behavioral analysis. In: Fuzzy Systems (FUZZ), 2010 IEEE International Conference on, pp. 1–6. IEEE (2010)
67. Hutchins, E.M., Cloppert, M.J., Amin, R.M.: Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Lead. Issues Inf. Warfare Secur. Res. **1**, 80 (2011)
68. Idika, N., Mathur, A.P.: A survey of malware detection techniques. Technical report 286, Department of Computer Science, Purdue University, USA (2007)
69. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing, pp. 604–613. ACM (1998)
70. International Secure Systems Lab: Anubis - Malware Analysis for Unknown Binaries. https://anubis.iseclab.org/. Accessed 29 July 2015
71. Jacob, G., Debar, H., Filiol, E.: Behavioral detection of malware: from a survey towards an established taxonomy. J. Comput. Virol. **4**(3), 251–266 (2008)
72. Jacob, G., Hund, R., Kruegel, C., Holz, T.: Jackstraws: Picking command and control connections from bot traffic. In: USENIX Security Symposium, vol. 2011. San Francisco, CA, USA (2011)
73. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer, Dordrecht (2009)
74. Jiang, X., Wang, X., Xu, D.: Stealthy malware detection through vmm-based out-of-the-box semantic view reconstruction. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 128–138. ACM (2007)
75. Joe Security: Agile Malware Analysis - Joe Sandbox Desktop. http://www.joesecurity.org/joe-sandbox-desktop. Accessed 29 July 2015
76. Julisch, K., Kruegel, C.: Detection of Intrusions and Malware, and Vulnerability Assessment. In: 2005, Proceedings of 2nd International Conference, DIMVA 2005, Vienna, Austria, July 7-8, Springer, New York (2005)
77. Kabiri, P., Ghorbani, A.A.: Research on intrusion detection and response: a survey. IJ Netw. Secur. **1**(2), 84–102 (2005)

78. Kaspersky Lab: Duqu: Steal Everything. http://www.kaspersky.com/about/press/major_malware_outbreaks/duqu. Accessed 29 July 2015
79. Kaspersky Lab: What is Flame Malware | Definition and Risks | Kaspersky Lab. http://www.kaspersky.com/flame. Accessed 29 July 2015
80. Kaspersky Lab's Global Research & Analysis Team: Gauss: Abnormal Distribution - Securelist. https://securelist.com/analysis/36620/gauss-abnormal-distribution/. Accessed 29 July 2015
81. Kendall, K., McMillan, C.: Practical malware analysis. In: Black Hat Conference, USA (2007)
82. Kitchenham, B.A.: Procedures for undertaking systematic reviews. Tech. rep. Computer Science Department, Keele University (2004)
83. Knight, S.: Sophisticated malware dubbed 'The Mask' went undetected for the past seven years - TechSpot. http://www.techspot.com/news/55640-sophisticated-malware-dubbed-the-mask-went-undetected-for-the-past-seven-years.html. Accessed 29 July 2015
84. Kosko, B.: Fuzzy cognitive maps. Int. J. Man-Mach. Stud. **24**(1), 65–75 (1986)
85. Krishnamurthy, B., Sen, S., Zhang, Y., Chen, Y.: Sketch-based change detection: methods, evaluation, and applications. In: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, pp. 234–247. ACM (2003)
86. Kruegel, C., Lippmann, R., Clark, A. (eds.): Recent advances in intrusion detection. In: 10th International Symposium, RAID 2007, Gold Coast [i.e. Coast], Australia, September 5–7, 2007: Proceedings on No. 4637 in Lecture Notes in Computer Science. Springer, Berlin (2007)
87. Kumar, S., Spafford, E.H.: A pattern matching model for misuse intrusion detection. In: Proceedings of the 17th National computer Security Conference, pp.11–21 (1994)
88. Kwon, J., Lee, H.: Bingraph: Discovering mutant malware using hierarchical semantic signatures. In: Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on, pp. 104–111. IEEE (2012)
89. Lakhotia, A., Preda, M.D., Giacobazzi, R.: Fast location of similar code fragments using semantic 'juice'. In: Proceedings of the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop, p. 5. ACM (2013)
90. Landwehr, C.E., Bull, A.R., McDermott, J.P., Choi, W.S.: A taxonomy of computer program security flaws. ACM Comput. Surv. (CSUR) **26**(3), 211–254 (1994)
91. Langeder, S.: Towards Dynamic Attack Recognition for SIEM. Ph.D. thesis, St. Poelten University of Applied Sciences (2014)
92. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Doklady **10**, 707–710 (1966)
93. Li, F., Lai, A., Ddl, D.: Evidence of advanced persistent threat: A case study of malware for political espionage. In: Malicious and Unwanted Software (MALWARE), 2011 6th International Conference on, pp. 102–109. IEEE (2011)
94. Line, M.B., Zand, A., Stringhini, G., Kemmerer, R.: Targeted Attacks Against Industrial Control Systems: Is the Power Industry Prepared? pp. 13–22. ACM Press (2014). doi:10.1145/2667190.2667192
95. Luo, X., Chan, E.W., Chang, R.K.: Vanguard: a new detection scheme for a class of TCP-targeted denial-of-service attacks. In: Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP, pp. 507–518. IEEE (2006)
96. MacDonald, N.: Is Antivirus Obsolete? http://blogs.gartner.com/neil_macdonald/2012/09/13/is-antivirus-obsolete/. Accessed 29 July 2015
97. Mankin, J., Kaeli, D.: DIONE: a flexible disk monitoring and analysis framework. In: Research in Attacks, Intrusions, and Defenses, pp. 127–146. Springer, New York (2012)
98. Mathew, S., Giomundo, R., Upadhyaya, S., Sudit, M., Stotz, A.: Understanding multistage attacks by attack-track based visualization of heterogeneous event streams. In: Proceedings of the 3rd International Workshop on Visualization for Computer Security, pp. 1–6. ACM (2006)
99. Mathew, S., Upadhyaya, S., Sudit, M., Stotz, A.: Situation awareness of multistage cyber attacks by semantic event fusion. In: Military Communications Conference, 2010-MILCOM 2010, pp. 1286–1291. IEEE (2010)
100. McGuinness, D.L., Van Harmelen, F.: OWL web ontology language overview. W3C Recomm. **10**(10), 2004 (2004)
101. Meier, M.: A model for the semantics of attack signatures in misuse detection systems. In: Information Security, pp. 158–169. Springer, New York (2004)
102. Miles, C., Lakhotia, A., LeDoux, C., Newsom, A., Notani, V.: VirusBattle: State-of-the-art malware analysis for better cyber threat intelligence. In: Resilient Control Systems (ISRCS), 2014 7th International Symposium on, pp. 1–6. IEEE (2014)
103. Mills, E.: A who's who of Mideast-targeted malware. http://www.cnet.com/news/a-whos-who-of-mideast-targeted-malware/. Accessed 18 Sept 2015
104. MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation. http://www.ll.mit.edu/ideval/data/. Accessed 29 July 2015
105. MITRE Corporation: CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC). https://capec.mitre.org/. Accessed 22 Sept 2015
106. MITRE Corporation: Common Event Expression: CEE, A Standard Log Language for Event Interoperability in Electronic Systems. https://cee.mitre.org/. Accessed 29 July 2015
107. MITRE Corporation: CVE - Common Vulnerabilities and Exposures (CVE). https://cve.mitre.org/. Accessed 22 Sept 2015
108. MITRE Corporation: CWE - Common Weakness Enumeration. https://cwe.mitre.org/. Accessed 22 Sept 2015
109. MITRE Corporation: STIX - Structured Threat Information Expression | STIX Project Documentation. https://stixproject.github.io/. Accessed 22 Sept 2015
110. Münz, G., Carle, G.: Real-time analysis of flow data for network attack detection. In: Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on, pp. 100–108. IEEE (2007)
111. Mowbray, T.J.: Cybersecurity: Managing Systems, Conducting Testing, and Investigating Intrusions. Wiley, New York (2013)
112. Munsey, C.: Economic Espionage: Competing For Trade By Stealing Industrial Secrets. https://leb.fbi.gov/2013/october-november/economic-espionage-competing-for-trade-by-stealing-industrial-secrets. Accessed 15 Sept 2015
113. Ou, X., Rajagopalan, R., Sakthivelmurugan, S.: A practical approach to modeling uncertainty in intrusion analysis. In: Technical Report, Department of Computing and Information Sciences, Kansas State University(2008)
114. Payne, B.D., Carbone, M., Sharif, M., Lee, W.: Lares: An architecture for secure active monitoring using virtualization. In: Security and Privacy, 2008. SP 2008. IEEE Symposium on, pp. 233–247. IEEE (2008)
115. Perez-Botero, D., Szefer, J., Lee, R.B.: Characterizing hypervisor vulnerabilities in cloud computing servers. In: Proceedings of the 2013 International Workshop on Security in Cloud Computing, pp. 3–10. ACM (2013)
116. Peshkin, L.: Structure induction by lossless graph compression. arXiv:cs/0703132 (2007)
117. Raskin, V., Hempelmann, C.F., Triezenberg, K.E., Nirenburg, S.: Ontology in information security: a useful theoretical foundation

and methodological tool. In: Proceedings of the 2001 Workshop on New Security Paradigms, pp. 53–59. ACM (2001)

118. Raywood, D.: Failure to detect Flame marks the end of signaturebased antivirus. http://www.scmagazineuk.com/failure-to-detect-flame-marks-the-end-of-signature-based-anti-virus/article/243505/. Accessed 29 July 2015

119. Razzaq, A., Ahmed, H.F., Hur, A., Haider, N.: Ontology based application level intrusion detection system by using bayesian filter. In: Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on, pp. 1–6. IEEE (2009)

120. Razzaq, A., Anwar, Z., Ahmad, H.F., Latif, K., Munir, F.: Ontology for attack detection: an intelligent approach to web application security. Comput. Secur. **45**, 124–146 (2014). doi:10.1016/j.cose.2014.05.005

121. Razzaq, A., Latif, K., Ahmad, H.F., Hur, A., Anwar, Z., Bloodsworth, P.C.: Semantic security against web application attacks. Inf. Sci. **254**, 19–38 (2014). doi:10.1016/j.ins.2013.08.007

122. Respect-IT: Kaos tutorial. http://www.objectiver.com/fileadmin/download/documents/KaosTutorial. Accessed 27 July 2015

123. Rieck, K.: Malheur - Automatic Analysis of Malware Behavior. http://www.mlsec.org/malheur/. Accessed 27 July 2015

124. Rieck, K., Trinius, P., Willems, C., Holz, T.: Automatic analysis of malware behavior using machine learning. J. Comput. Secur. **19**(4), 639–668 (2011)

125. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM conference on Computer and communications security, pp. 199–212. ACM (2009)

126. Russinovich, M.: Process Monitor. https://technet.microsoft.com/en-us/library/bb896645.aspx. Accessed 29 July 2015

127. Sadighian, A., Zargar, S.T., Fernandez, J.M., Lemay, A.: Semantic-based context-aware alert fusion for distributed Intrusion Detection Systems. In: 2013 International Conference on, Risks and Security of Internet and Systems (CRiSIS), pp. 1–6. IEEE (2013)

128. Sangeetha, S., Vaidehi, V.: Fuzzy aided application layer semantic intrusion detection system - FASIDS. Int. J. Netw. Secur. Appl. **2**(2), 39–56 (2010). doi:10.5121/ijnsa.2010.2204

129. Scheirer, W., Chuah, M.C.: Syntax vs. semantics: competing approaches to dynamic network intrusion detection. Int. J. Secure. Netw. **3**(1), 24–35 (2008)

130. Seculert: Mahdi - The Cyberwar Savior? http://www.seculert.com/blog/2012/07/mahdi-cyberwar-savior.html. Accessed 29 July 2015

131. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mob. Comput. Commun. Rev. **5**(1), 3–55 (2001)

132. Sharif, M., Yegneswaran, V., Saidi, H., Porras, P., Lee, W.: Eureka: A framework for enabling static malware analysis. In: Computer security-ESORICS 2008, pp. 481–500. Springer, New York (2008)

133. Sikorski, M., Honig, A.: Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. No Starch Press, San Francisco (2012)

134. Sood, A.K., Enbody, R.J.: Targeted cyberattacks: a superset of advanced persistent threats. IEEE Secur. Privacy **1**, 54–61 (2013)

135. Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., Stiller, B.: An overview of IP flow-based intrusion detection. IEEE Commun. Surv. Tutorials **12**(3), 343–356 (2010). doi:10.1109/SURV.2010.032210.00054

136. Stanford Center for Biomedical Informatics Research: Protégé. http://protege.stanford.edu/. Accessed 29 July 2015

137. Stotz, A., Sudit, M.: INformation fusion engine for real-time decision-making (INFERD): a perceptual system for cyber attack tracking. In: Information Fusion, 2007 10th International Conference on, pp. 1–8. IEEE (2007)

138. Strasburg, C., Basu, S., Wong, J.S.: S-MAIDS: A Semantic Model for Automated Tuning, Correlation, and Response Selection in Intrusion Detection Systems. pp. 319–328. IEEE (2013). doi:10.1109/COMPSAC.2013.57

139. Symantec: Regin: Top-tier espionage tool enables stealthy surveillance. http://www.symantec.com/connect/blogs/regin-top-tier-espionage-tool-enables-stealthy-surveillance. Accessed 15 Sept 2015

140. Thakar, U., Dagdee, N., Varma, S.: Pattern analysis and signature extraction for intrusion attacks on web services. Int. J. Netw. Secur. Appl. **2**(3), 190–205 (2010). doi:10.5121/ijnsa.2010.2313

141. The Hacker News: Harkonnen Operation–Malware Campaign that Went Undetected for 12 Years. http://thehackernews.com/2014/09/harkonnen-operation-malware-campaign_16.html. Accessed 29 July 2015

142. ThreatTrack Security: Dynamic Malware Analysis Tools, Malware Sandbox - ThreatAnalyzer - ThreatTrack Security. http://www.threattracksecurity.com/enterprise-security/malware-analysis-sandbox-software.aspx. Accessed 29 July 2015

143. Totel, E., Vivinis, B., Mé, L.: A language driven intrusion detection system for event and alert correlation. In: Proceedings at the 19th IFIP International Information Security Conference, Kluwer Academic, Toulouse, pp. 209–224. Springer, New York (2004)

144. Trammell, B., Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. https://tools.ietf.org/html/rfc7011. Accessed 29 July 2015

145. Trinius, P., Willems, C., Holz, T., Rieck, K.: A malware instruction set for behavior-based analysis. Tech. Rep. TR-2009-07, University of Mannheim (2009)

146. University of California: KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed 29 July 2015

147. University of Toronto: GRL Syntax. http://www.cs.toronto.edu/km/GRL/grl_syntax.html. Accessed 27 July 2015

148. Vance, A.: Flow based analysis of Advanced Persistent Threats detecting targeted attacks in cloud computing. In: Infocommunications Science and Technology, 2014 1st International Scientific-Practical Conference Problems of, pp. 173–176. IEEE (2014)

149. Vanderwende, L.H., Loritz, D.: The analysis of noun sequences using semantic information extracted from on-line dictionaries. Ph.D. thesis, Georgetown University (1995)

150. W3C: Semantic web. http://www.w3.org/standards/semanticweb/. Accessed 29 July 2015

151. W3C: SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). http://www.w3.org/TR/soap12/. Accessed 22 July 2015

152. W3C: SPARQL 1.1 Overview. http://www.w3.org/TR/sparql11-overview/. Accessed 29 July 2015

153. Wagner, D., Soto, P.: Mimicry attacks on host-based intrusion detection systems. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 255–264. ACM (2002)

154. Wagner, M., Fischer, F., Luh, R., Haberson, A., Rind, A., Keim, D., Aigner, W., Borgo, R., Ganovelli, F., Viola, I.: A Survey of Visualization Systems for Malware Analysis. In: EG Conference on Visualization (EuroVis)-STARs, pp. 105–125. EuroGraphics (2015)

155. Wang, R., Jia, X., Nie, C.: A Behavior Feature Generation Method for Obfuscated Malware Detection. pp. 470–474. IEEE (2012). doi:10.1109/CSSS.2012.124

156. Wüchner, T., Pretschner, A., Ochoa, M.: DAVAST: data-centric system level activity visualization. pp. 25–32. ACM Press (2014) doi:10.1145/2671491.2671499

157. Willems, C., Holz, T., Freiling, F.: Toward automated dynamic malware analysis using cwsandbox. IEEE Secur. Privacy **2**, 32–39 (2007)

158. Winskel, G.: The formal semantics of programming languages: an introduction. Foundations of computing, 5th edn. MIT Press, Cambridge, MA (2001)
159. Xu, H., Du, W., Chapin, S.J.: Context sensitive anomaly monitoring of process control flow to detect mimicry attacks and impossible paths. In: RAID, pp. 21–38. Springer, New York (2004)
160. Yan, W., Hou, E., Ansari, N.: Extracting attack knowledge using principal-subordinate consequence tagging case grammar and alerts semantic networks. In: Local Computer Networks, 2004. 29th Annual IEEE International Conference on, pp. 110–117. IEEE (2004)
161. Yan, W., Hou, E., Ansari, N.: A description logic based approach for IDS security information management. In: Advances in Wired and Wireless Communication, 2005 IEEE/Sarnoff Symposium on, pp. 25–28. IEEE (2005)
162. Yan, X., Han, J.: gspan: graph-based substructure pattern mining. In: Data Mining, 2002. ICDM 2003. Proceedings of 2002 IEEE International Conference on, pp. 721–724. IEEE (2002)
163. Zarras, A., Papadogiannakis, A., Gawlik, R., Holz, T.: Automated generation of models for fast and precise detection of HTTP-based malware. In: 2014 12th Annual International Conference on, Privacy, Security and Trust (PST), pp. 249–256. IEEE (2014)
164. Zhang, Q., Reeves, D.S., Ning, P., Iyer, S.P.: Analyzing network traffic to detect self-decrypting exploit code. In: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications security, pp. 4–12. ACM (2007)
165. Zhu, B., Ghorbani, A.A.: Alert correlation for extracting attack strategies. Ph.D. thesis, Citeseer (2005)
166. Zimmer, D., Unland, R.: On the semantics of complex events in active database management systems. In: 1999, Proceedings of 15th International Conference on, Data Engineering, pp. 392–399. IEEE (1999)