Advanced threat intelligence: detection and classification of anomalous behavior in system processes

R. Luh, S. Schrittwieser

With the advent of Advanced Persistent Threats (APTs), it has become increasingly difficult to identify and understand attacks on computer systems. This paper presents a system capable of explaining anomalous behavior within network-enabled user sessions by describing and interpreting kernel event anomalies detected by their deviation from normal behavior.

The prototype has been developed at the Josef Ressel Center for Unified Threat Intelligence on Targeted Attacks (TARGET) at St. Pölten University of Applied Sciences.

Keywords: malware; attack; anomaly detection

Advanced Threat Intelligence: Erkennung und Klassifizierung von anomalem Verhalten in Systemprozessen.

Mit dem Aufkommen von Advanced Persistent Threats (APTs) ist es immer schwieriger geworden, Angriffe auf Computersysteme zu identifizieren und zu verstehen. Diese Arbeit stellt ein System vor, das in der Lage ist, anomales Verhalten innerhalb von Benutzer-Sessions zu erklären, indem es Kernel-Ereignisanomalien beschreibt und klassifiziert, welche durch ihre Abweichung vom Normalverhalten erkannt werden.

Der Prototyp wurde am Josef Ressel-Zentrum für konsolidierte Erkennung gezielter Angriffe (TARGET) an der Fachhochschule St. Pölten entwickelt.

Schlüsselwörter: Schadsoftware; Angriff; Anomalieerkennung

Received October 2, 2019, accepted November 22, 2019 © The Author(s) 2019

1. Introduction

IT systems are threatened by an ever-growing number of cyberattacks. With the emergence of Advanced Persistent Threats (APTs), the focus shifted from off-the-shelf malware to multipartite attacks that are tailored to specific organizations or systems. These targeted threats are driven by varying motivations, such as espionage or sabotage, and often cause significantly more damage [1].

While APTs use malware like most conventional attacks, the level of complexity and sophistication of the malicious programs is usually higher. This is problematic since defensive measures offered by security vendors often utilize primarily signature-based systems, which are effective in the defense against known exploit carriers or ill-considered user actions but struggle with hitherto unknown malware [2]: Traditional misuse detection relies heavily on signature databases that have to be updated whenever a new attack technique or sample is discovered. For emerging threats, such binary patterns of the involved malware are unlikely to exist at the time of attack. Poly- and metamorphic techniques [3] additionally obfuscate malicious software by creating self-altering malware variants that sport differing static appearance for cryptor and payload. Mimicry attacks, on the other hand, might substitute system calls, interweave instructions, or attempt to avoid generating observable events at all, which increases the complexity of dynamic detection efforts [4].

Targeted or not, modern cyber-threats are no longer limited to a single malware executable (i.e. sample) but often comprise multistage attacks that are difficult to spot using only file- and signaturebased malware detection systems. Therefore, it is necessary to explore novel techniques for threat intelligence and APT detection that are augmented with contextual information and provide resilience to various stealth techniques.

Dynamic, behavior-based solutions are a promising means to identify adversary activity on live IT systems. Anomalies signifying a deviation from a known behavioral baseline can be used to detect the threat in its early stages. However, most existing systems do not provide the offending behavioral data to the analyst and contribute little to its interpretation. We argue that closing that semantic gap is a vital next step in holistic (IT) system threat mitigation.

In this article, which is based on the publications "LLR-based sentiment analysis for kernel event sequences" [5] and "AIDIS: Detecting and Classifying Anomalous Behavior in Ubiquitous Kernel Processes" [6], we propose an intrusion detection and classification system capable of explaining anomalous behavior within a networkenabled user session by considering kernel event anomalies identified through their deviation from a set of baseline process graphs. For this purpose, we adapt star structures [7], a less computationally complex, bipartite representation used to approximate the edit distance between two graphs. Baseline templates are generated auto-

Luh, Robert, Fachhochschule St. Pölten, Josef Ressel Zentrum TARGET, Matthias-Corvinus-Straße 15, 3100 St. Pölten, Österreich (E-mail: robert.luh@fhstp.ac.at); Schrittwieser, Sebastian, Fachhochschule St. Pölten, Josef Ressel Zentrum TARGET, Matthias-Corvinus-Straße 15, 3100 St. Pölten, Österreich (E-mail: sebastian.schrittwieser@fhstp.ac.at)

R. Luh, S. Schrittwieser Advanced threat intelligence: detection and classification...



Fig. 1. Overview of intrusion detection and classification system

matically and adapt to the nature of the respective operating system process.

Since we want to move away from sample-centric analysis that requires knowledge about a particular suspicious binary, we additionally use a machine learning approach to sentiment analysis based on the log likelihood ratio (LLR) test [8] to identify ubiquitous Windows kernel processes that contribute most to an automated malicious/benign classification. This helps narrow down the computational scope of the aforementioned anomaly detection component by providing a small pool of processes worthy of increased scrutiny.

Ultimately, we implemented smart anomaly classification through a set of competency questions applied to graph template deviations and evaluated the approach using both Random Forest [9] and linear kernel Support Vector Machines [10]. The determined attack classes are ultimately mapped to a novel APT attacker/defender model that considers actions, actors, as well as assets and mitigating controls, thereby enabling decision support and contextual interpretation of past and ongoing attacks. Figure 1 depicts the proposed multi-stage system in its entirety. This article focuses on the components "Sentiment Analysis" and "Star Graph Analysis".

1.1 Related work

For an in-depth look at related and similar work refer to Luh et al.'s literature survey [11] as well as the system's core publication [6].

2. Methodology

The methodology chosen to investigate our approach to targeted attack detection and classification is *experimental, constructive* research based on large-scale data analytics as well as top-down threat modeling.

Our research is founded on a formal top-down and a technical bottom-up component. This article focuses on the latter; namely data collection, knowledge extraction, and anomaly analysis, which corresponds to the aforementioned system components. Figure 2 summarizes the overall methodical approach.

We specifically explore data monitoring and processing, behavioral analysis, as well as anomaly detection and classification. Our system comprises three stages:

- Observation of actively used computer systems as part of the initial data monitoring process;
- Extraction of behavioral patterns from the collected data to determine the most relevant processes and features in regard to anomaly detection;
- White-box anomaly detection and granular classification to spot attacks and classify them in accordance to their attack stage or pattern.

3. Advanced threat detection and classification

This article focuses on the technical component of the proposed system. Work detailing the formal approach can be found in [12] and [13]. In the following we discuss the three stages highlighted in Fig. 2: Data collection, process behavior extraction, and anomaly detection with classification.

3.1 Observation: data collection and processing

In our research, we opted to use *kernel monitoring data* of moderate abstraction, which represents a compromise between raw events or API calls and high-level alerts based on often problematic patterns. Our final data selection comprised abstracted process and file system activity as well as network functions invoked on the endpoint. The instrument used for collecting the data is a monitoring agent prototype dubbed "Sonar". It runs as part of the Windows kernel and utilizes SSDT hooking [14, 15] to remain undetected. Sonar collects process, thread, image load, file, registry, and network events in the form of abstracted API and system calls. These events describe i.a. process launches or terminations, access to local resources, or the retrieval of system configuration information. Other functions represent the establishing of network connections or general interaction with external resources – all of which can be considered an event in the context of a process.





Table 1. Timeline of events as chronological trace ordered by process occurrence

Event type	Argument (operation)
ProcessEvent	Start process A with PID 220
ImageLoadEvent	Load image "process.exe"
RegistryEvent	Write file "document.txt"
ProcessEvent	Start process B with PID 224
RegistryEvent	Modify registry key "HKLM/"
ProcessEvent	Terminate process with PID 220
ProcessEvent	Start process B with PID 224
ImageLoadEvent	Load image "library.dll"
NetworkEvent	Connect to IP address 1.2.3.4

To maintain chronology as well as context, each individual event captured by the Sonar agent is time-stamped and can be linked to a specific process or thread through its respective (parent) process ID. This allows us to construct trees of individual processes as well as entire system sessions, as discussed at length in [16]. See Fig. 3 for an example process tree.

Ultimately, all process trees are converted to traces in order to enable word-based processing. Table 1 shows an example based on the tree depicted in Fig. 3.

Each of our system's components use such traces for their pattern extraction and anomaly detection routines. Despite the function abstraction implemented by SONAR, manually interpreting the depicted activity becomes a daunting task. Therefore, automated solutions are necessary to process the newly generated timelines.

3.2 Extraction: determining process behavior

We use an approach akin to sentiment analysis [17] to generate initial knowledge about relevant OS processes and to determine the most expressive process candidates for investigation. At the same time, this first stage computes a first benign/malicious score that provides us with a tendency towards general harmfulness for the provided dataset. In the following, we highlight the main properties of the sentiment analysis component, which has been initially disseminated in [5]. The likelihood ratio (LR) test employed is a statistical method used to test model assumptions, namely the quality of fit of a reference (null) and an alternative model, whereby the simpler model can be understood as a special case of the more complex one [8]. The goal of a model is to find parameter values that in turn maximize the value of the likelihood function, which is equivalent to finding the set of parameters that make the data most likely [18]. The LR test relies less on the assumption that a certain variable (i.e. word) is distributed normally throughout a text than e.g. chi-squared or z-score tests. When computing the occurrence of rarely observed events – which are at the core of many a malicious trace – likelihood ratio tests show significantly better results than the mentioned alternatives.

Dunning [8] defines the *likelihood ratio* for a hypothesis as the ratio of the maximum value of the likelihood function over the subspace represented by the hypothesis to the maximum value of the likelihood function L over the entire parameter space, where Ω is the entire parameter space and Ω_0 is the hypothesis being tested.

$$LR = \frac{\max_{\omega \in \Omega_0} L(\omega; k)}{\max_{\omega \in \Omega} L(\omega; k)}$$

As mentioned above, the basis for sentiment analysis are kernel event traces reordered to maintain process and thread context. After tokenization, we extract bigrams from the collected sequences. The general goal of this stage is to find useful features in a large amount of system traces, in particular events that occur in combination. We compute the LLR score for each individual n-gram to highlight collocations characteristic for sequences of malicious and benign system events. This enables the analyst to ultimately rank the words of a corpus by their domain relevance and subsequently extract processes that are more likely to be involved in adversarial behavior.

The formulas presented in Table 2 (matrix K) are responsible for counting token occurrences in the given distribution. The different values are defined as the number of times both event tokens occur together (k_{11}), the number of times each event token has been observed independently from the other (k_{12} and k_{21} , depending on their position in the bigram), and the number of times the tokens were not present at all (k_{22}).



Fig. 3. Example tree of a process launched by the kernel

Table 2. Event occurrence matrix K

	А	!A
В	$k_{11} = k (AB)$	$k_{12} = k (!AB)$
iB	$k_{21} = k \left(A ! B \right)$	$k_{22} = k \left(!A !B \right)$

Once these counts have been determined, it becomes possible to compute the log likelihood ratio (LLR) score. Applied to the assembled matrix K, LLR becomes

 $LLR(K) = 2S(K) \cdot (H(K) - H(k_{11} + k_{12}k_{21} + k_{22}))$

$$-H(k_{11}+k_{21}k_{12}+k_{22}))$$

where S(K) represents the sum of all matrix elements k_{ij} in K and H denotes the function computing the Shannon entropy [19]. I.e. given an $m \times n$ matrix $X = (x_{ij})$, we have:

$$H(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\frac{x_{ij}}{S(X)} \cdot \log\left(\frac{x_{ij}}{S(X)}\right) \right)$$

with

$$S(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}$$

Using this methodology, we are able to extract a list of bigrams that represent likely collocations, signifying their semantic relationship. Based on these calculations we can then determine the association of bigrams to different corpora of known benign and malicious events. With a significantly high number of known benign and malicious kernel event traces at one's disposal it becomes possible compile a dictionary of suspect and valid events.

The dictionary compilation process itself is similar to the one described above. We determine the occurrence of known malicious ngrams in a benign sample set. In addition to LLR-based scoring performed for opposing corpora, we calculate the occurrence of partial and full malicious bigrams in the benign corpus. This results in a new LLR score that describes the association of each bigram to both sets of data.

Based on these scores we can assign sentiment to each identified bigram determined through the occurrence of the sequence and the maximum LLR measured for each corpus. The result is a normalized sentiment rating *s* ranging from +1.0 (benign) to -1.0 (malicious).

Scoring is achieved by means of both the benign and malicious scores determined through a comparison with the sentiment dictionary compiled in the previous stage. The final score *s* is determined by whether the root-LLR scores of the unknown event sequence lean toward the harmless or suspicious end of the spectrum.

LLR-scored sequence combinations have proven to be good indicators for both individual programs as well as entire system sessions. Simultaneously, the resulting scores give us a good idea which ubiquitous kernel processes are the most likely candidates for closer examination.

3.3 Detection and classification: interpreting anomalies

Once monitoring candidates have been identified, we utilize star structures to create a by-process representation of event sequences that encompass single process launch behavior, its full run time, or even entire multi-process system sessions. Star structures are a means to reduce the complexity of a known NP problem to polynomial complexity [7, 20]. Instead of searching entire system session graphs for matching patterns, the star structure approach breaks down the computation into a triplet of nodes (vertices) connected by a labeled edge, denoted as G = (U, V, E). The labels of both vertices and edges are used as basis for minimal cost calculation of samesize star structures. Specifically, we utilize bipartite graph matching based on the Hungarian (Kuhn-Munkres) algorithm [21], namely an adaptation of Hu's approach [7] which combines n bipartite graphs into one star representing a single process and all its activity. The process becomes the natural center vertex of a star-shaped graph, where the outer vertices represent the individual actions performed within its context and the edges describe the type of operation performed (e.g. create, modify, delete). Figure 4 shows a simplified example.

This method for determining the minimal edit distance between two star-shaped graphs is used as the foundation for context-aware anomaly detection utilizing supervised learning on a per-process basis. The cost of required transformation operations can be used to determine the event-level deviation between instances of the same process. In order to automatically determine thresholds for each observed process, we create one or several templates from a benign environment in one of 4 different ways: 1) *Perfect match* extracts identical events found in each iteration of a process and assembles an entirely new graph, 2) *Majority mode* picks the most common



Fig. 4. Example event representation for process sychost.exe (central node)

base graph from the input set and converts it to a template without altering its contents, 3) *Prototype extraction* uses the Malheur algorithm [22, 23] to extract not one, but several prototypes representative of the various aspects of a single process, and 4) *Similarity hashing* measured document similarity using the MinHash algorithm [24].

Armed with one or several templates for each process, we can now check unknown graphs against the predetermined thresholds and extract events responsible for the deviation. To classify these anomalous events, we map them to CAPEC attack patterns describing concrete adversary behavior [13]. Features are extracted through a list of 200 competency questions, which include simple Boolean queries into the presence of events over another event (e.g. if the number of thread terminations exceed the number of thread spawns) as well as decisions based on the presence of certain activity tags describing the base functionality of a loaded image (e.g. networking, authentication, user interface, kernel, etc.).

For the discrimination of anomaly traces, both binary 'benign' vs. 'malicious' and multi-class classification is used, focusing on *Ran-dom Forest* and *Linear Kernel Support Vector Machines* (SVM). Ultimately, our system maps the resulting verdict (e.g. 'anomaly belongs to class CAPEC-112') and the anomaly report itself to the model disseminated in [13], thereby building our knowledge base of labeled attacks that can then be associated a goal, stage, likely actor, possible countermeasure, and more.

4. Evaluation

The prototype of the system was implemented in a test-bed environment consisting of 13 physical Windows computers used on and off by developers and IT personnel of a medium business over the course of half a year. One additional virtual Windows instance was utilized for dynamically monitoring malicious software and automated targeted attacks on demand.

The repository of data used for evaluating the system included a total of 125 GiB of traces with more than 1.3 billion individual events across all monitored processes. Another 4.3 million (4.5 GiB) events were recorded on the aforementioned analysis VM. For these malicious traces, we executed a total of 1,995 APT malware samples and attack software, ranging from DarkComet [25] and other, unnamed Remote Access Trojans (RATs) to various crypto-miners and tools such as ShoulderSurfer.¹ Since our system is not primarily used for malware classification but considers behavior independently, most monitored attack activity is not attributed to specific

Tahlo 3	Roculte	of the	anomaly	classification
idule 5.	nesults	or the	anomaiv	Classification

	Classes	OOB error	Accuracy	Карра	C-value	Time (s)
RF	2	0.26%	99.77%	-	_	142.1
SVM	2	-	99.82%	99.24%	1	70.4
SVM grid	2	-	99.83%	99.28%	0.25	1899.8
RF	n	4.96%	91.37%	-	-	224.8
SVM	n	-	95.53%	94.67%	1	412.0
SVM grid	n	-	95.73%	94.87%	1.75	8180.4

sample families. We instead use a CAPEC-based classification [26] to describe patterns for e.g. reconfiguring the system or disabling security mechanisms. Altogether, we classified our data into 23 CAPEC attack patterns, a "benign" class for non-malicious software, and two classes representing crashing or idle malware [6].

4.1 Results summary

Next to the compound system (named "AIDIS Core" below), all individual components can be employed in standalone mode and have been tested with standard event traces as described in Sect. 3. Performance was assessed using the traditional accuracy score as main performance indicator. Figure 5 shows a summary. Note that "SE-QUIN" is an optional component discussed in [27].

For both binary and CAPEC multi-class classification, the core classification system utilizing SVM with hyperplane optimization offers the highest accuracy with 99.82% and 95.73%, respectively. We used 10-fold cross validation and a 70–30 split between training and validation data in our experiments. Table 3 provides further details.

Key features as per mean decrease in accuracy/Gini turned out to be the count of error function libraries imported, the use of Windows user management and universal app functions, high (system) registry interaction, operations related to log files, network activity in general, as well as the import of data access and diagnostics functions.

In summary, many of the most relevant features are related to image load and registry operations. The reason for this can be found to a degree in the selection of data used in the experiment: With a focus on the initial 10 s of activity, it is expected to see numerous events pertaining to the dynamic linking of libraries [28], which is generally more widely used than static or runtime linking in both malware and benign software. Registry events typically represent

¹https://wikileaks.org/ciav7p1/cms/page_524353.html.

Process	Description	Min	Mean	Max	Std. dev.	Total #
conhost	Console window host process	-0.895	0.016	0.607	0.104	485932
csrss	Win32 user-mode subsystem	-1.000	0.039	0.836	0.171	2952
explorer	Explorer shell and file manager	-1.000	0.003	1.000	0.008	756709
searchindexer	Windows search and indexing	-1.000	0.002	1.000	0.047	310956
smss	Session manager subsystem	-1.000	-0.236	1.000	0.416	53126
svchost	Generic host process	-1.000	0.005	1.000	0.012	1612797
taskhost	Generic host process for libraries	-1.000	0.022	0.983	0.048	409818

Table 4. Primary process candidates for monitoring



Fig. 5. Classification accuracy comparison of the core system and its individual components

initialization tasks or changes to certain settings, something that is often seen in the early stages of operation as well. Interestingly, file events were found to be generally underrepresented during the start-up of compromised process instances in particular: Only 108 events in the selection described malicious file operations, as opposed to 40,538 events in the benign svchost.exe corpus. As a result, the lack of specific file operations might be a strong indicator of manipulation – something that has to be considered in future feature selection.

As a standalone system, the *sentiment analysis* component performed well (accuracy of 98.2%) in comparison to similar n-grambased implementations such as AccessMiner [29], which achieved an average accuracy rate of 89.5% using file operation trigrams of benign applications as baseline. This supports our decision of observing several event types in synergy and of basing our decision on a statistical test rather than the number of n-grams deviating from a baseline.

To determine not only a preliminary anomaly score but identify processes more likely to exhibit malicious behavior, we used statistical evaluation of close to 2,000 unique processes that were launched during the lifetime of more than 10,000 benign and close to 2,000 malicious system sessions. This yielded a list of 7 processes (Table 4) that were determined to be of greatest significance. In order to perform meaningful data mining, the process with the most recorded events (126.1 million) was chosen for all subsequent analyses: sv-chost.exe, the Windows generic host (service host) process [30].

Star anomaly detection by itself resulted in a workable accuracy of at least 89.3% and up to 96.2%, depending on the number of templates and the kind of threshold optimization used. Utilizing multiple templates for complex processes such as svchost.exe is highly recommended: In our experiments, doing so increased accuracy by at least 2 percentage points across the board. Still, optimizing the threshold instead of using the mean star graph distance deviation should be used sparingly to avoid overfitting. The key contribution of the (standalone) star anomaly system is undoubtedly the new insights into process coverage: Our results show that 88.48% of all malicious activity is reflected in the first 10 s of activity of the svchost.exe generic host process. This finding has the potential to significantly contribute to IDS efficiency for both new and existing solutions.

5. Conclusion

In this article, we presented the core components of a star structurebased intrusion detection and classification system able to detect and explain anomalous deviations in operating system process behavior. The returned output of detailed state changes as well as a tendency towards a specific APT stage and attack pattern is expressed through the mapping of semantic key factors to a dedicated attacker–defender model. At the same time, the model suggests specific measures intended to counter any observed attack.

The process was prototypically implemented and successfully tested using real-world process data captured on more than a dozen company workstations over half a year. Ultimately, 99.8% of star structure anomalies were correctly identified as benign or malicious, with a solid 95.7% accuracy in multi-class scenarios that seek to associate each anomaly with a distinct CAPEC attack pattern. Furthermore, we have shown that 88.3% of close to 2,000 attacks could be accurately identified by observing and classifying just one generic Windows process for a mere 10 seconds, thereby eliminating the ne-

cessity to monitor each and every (unknown) process existing on a system.

For future research, we aim to focus on strategy inference utilizing the model in combination with anomalous events tagged by our classification system to compute optimal responses to a wide range of attacks. This will enable analysts to employ our solution as expert system supporting both risk management and organizational threat mitigation while being provided detailed technical assessments about individual stages of an intrusion.

Acknowledgements

Open access funding provided by FH St. Pölten – University of Applied Sciences. The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/ licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- 1. Symantec Security Response Team (2018): Internet security threat report. Symantec.
- Dornhackl, H., Kadletz, K., Luh, R., Tavolato, P. (2014): Malicious behavior patterns. In 2014 IEEE 8th international symposium on service oriented system engineering, SOSE.
- O'Kane, P., Sezer, S., McLaughlin, K. (2011): Obfuscation: the hidden malware. IEEE Secur. Priv., 9, 41–47.
- Wagner, D., Soto, P. (2002): Mimicry attacks on host-based intrusion detection systems. In Proceedings of the 9th ACM conference on computer and communications security.
- Luh, R., Schrittwieser, S., Marschalek, S. (2017): LLR-based sentiment analysis for kernel event sequences. In 2017 IEEE 31st international conference on advanced information networking and applications, AINA.
- Luh, R., Janicke, H., Schrittwieser, S. (2019): AIDIS: detecting and interpreting anomalous behavior in ubiquitous kernel processes. J. Comput. Secur. https://doi.org/10.1016/j.cose.2019.03.015.
- Hu, X., Chiueh, T-c., Shin, K. G. (2009): Large-scale malware indexing using functioncall graphs. In Proceedings of the 16th ACM conference on computer and communications security.
- Dunning, T. (1993): Accurate methods for the statistics of surprise and coincidence. Comput. Linguist., 19, 61–74.

Authors



Robert Luh

was awarded his master's degree in information security at St. Pölten University of Applied Sciences (UAS) in 2013. He recently defended his PhD thesis on attack recognition and interpretation of anomalous events in IDS monitoring data at DeMontfort University in Leicester. In his role as lecturer and researcher at St. Pölten UAS and its Institute of IT Security Research he mainly focuses on threat

modeling, adversary and malicious software behavior, intrusion detection, and machine learning.

- Liaw, A., Wiener, M., et al. (2002): Classification and regression by randomforest. R News, 2, 18–22.
- 10. Cortes, C., Vapnik, V. (1995): Support-vector networks. Mach. Learn., 20, 273–297.
- Luh, R., Marschalek, S., Kaiser, M., Janicke, H., Schrittwieser, S. (2016): Semanticsaware detection of targeted attacks: a survey. J. Comput. Virol. Hacking Tech., 13, 17–85.
- Luh, R., Schrittwieser, S., Marschalek, S. (2016): TAON: an ontology-based approach to mitigating targeted attacks. In Proceedings of the 18th international conference on information integration and web-based applications services.
- Luh, R., Temper, M., Tjoa, S., Schrittwieser, S. (2018): APT RPG: design of a gamified attacker/defender meta model. In Proceedings of the 4th international conference on information systems security and privacy, ICISSP 2018.
- Grégio, A. R. A., Fernandes Filho, D. S., Afonso, V. M., Santos, R. D. C., Jino, M., Geus, P. L. (2011): Behavioral analysis of malicious code through network traffic and system call monitoring. In SPIE defense, security, and sensing.
- Sebastian, E., Robert, L., Sebastian, S. (2017): The evolution of process hiding techniques in malware-current threats and possible countermeasures. J. Inf. Process. (58). https://doi.org/10.2197/ipsjip.25.866.
- Marschalek, S., Luh, R., Kaiser, M., Schrittwieser, S. (2015): Classifying malicious system behavior using event propagation trees. In Proceedings of the 17th international conference on information integration and web-based applications services.
- Gamon, M. (2004): Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In Proceedings of the 20th international conference on computational linguistics.
- UCLA Institute for Digital Research and Education: How are the likelihood ratio, Wald, and Lagrange multiplier (score) tests different and/or similar?
- Shannon, C. E. (1948): A mathematical theory of communication. Bell Syst. Tech. J. 27, 379–423.
- Zeng, Z., Tung, A. K. H., Wang, J., Feng, J., Zhou, L. (2009): Comparing stars: on approximating graph edit distance. Proc. VLDB Endow., 2, 25–36).
- Kuhn, H. W. (1955): The Hungarian method for the assignment problem. Nav. Res. Logist. Q., 2, 83–97.
- 22. Trinius, P., Willems, C., Holz, T., Rieck, K. (2009): A malware instruction set for behavior-based analysis.
- Rieck, K., Trinius, P., Willems, C., Holz, T. (2011): Automatic analysis of malware behavior using machine learning. J. Comput. Secur., 19, 639–668.
- Broder, A. Z. (1997): On the resemblance and containment of documents. In Compression and complexity of sequences proceedings.
- 25. Kujawa, A. (2012): You dirty RAT! Part 1: DarkComet.
- MITRE Corporation (2015): CAPEC common attack pattern enumeration and classification (CAPEC).
- Luh, R., Schramm, G., Wagner, M., Janicke, H., Schrittwieser, S. (2018): SEQUIN: a grammar inference framework for analyzing malicious system behavior. J. Comput. Virol. Hacking Tech., 14, 1–21.
- 28. Franz, M. (1997): Dynamic linking of software components. Computer, 30, 74-81.
- Lanzi, A., Balzarotti, D., Kruegel, C., Christodorescu, M., Kirda, E. (2010): Access-Miner: using system-centric models for malware protection. In Proceedings of the 17th ACM conference on computer and communications security.
- Russinovich, M. E., Solomon, D. A., Ionescu, A. (2012): Windows internals. Upper Saddle River: Pearson Education.



Sebastian Schrittwieser

was awarded his doctorate at TU Wien in 2014. His dissertation revolved around the topic of code analysis and obfuscation. Since 2015, Sebastian heads the Josef Ressel Center for Unified Threat Intelligence on Targeted Attacks, which explores novel techniques for detecting and mitigating targeted attacks on IT infrastructures. He is a full-time permanent professor (FH) and scientific head of the In-

stitute of IT Security Research at St. Pölten UAS. His main research interests are static code analysis, code obfuscation, malware detection, and digital forensics.