

Ali Dehghantanha
Kim-Kwang Raymond Choo *Editors*

Handbook of Big Data and IoT Security



Handbook of Big Data and IoT Security

Ali Dehghantanha • Kim-Kwang Raymond Choo
Editors

Handbook of Big Data and IoT Security



Springer

Editors

Ali Dehghanianha
Cyber Science Lab
School of Computer Science
University of Guelph
Guelph, ON, Canada

Kim-Kwang Raymond Choo
Department of Information Systems
and Cyber Security
The University of Texas at San Antonio
San Antonio, TX, USA

ISBN 978-3-030-10542-6 ISBN 978-3-030-10543-3 (eBook)
<https://doi.org/10.1007/978-3-030-10543-3>

Library of Congress Control Number: 2019934371

© Springer Nature Switzerland AG 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Big Data and Internet of Things Security and Forensics: Challenges and Opportunities	1
Amin Azmoodeh, Ali Dehghantanha, and Kim-Kwang Raymond Choo	
Privacy of Big Data: A Review.....	5
S. Sangeetha and G. Sudha Sadasivam	
A Biometric Analysis of Authentication and Access Control in IoT Devices.....	25
Samuel Grooby, Tooska Dargahi, and Ali Dehghantanha	
Towards Indeterminacy-Tolerant Access Control in IoT	53
Mohammad Heydari, Alexios Mylonas, Vasileios Katos, and Dimitris Gritzalis	
Private Cloud Storage Forensics: Seafile as a Case Study	73
Yee-Yang Teing, Sajad Homayoun, Ali Dehghantanha, Kim-Kwang Raymond Choo, Reza M. Parizi, Mohammad Hammoudeh, and Gregory Epiphaniou	
Distributed Filesystem Forensics: Ceph as a Case Study	129
Krzysztof Nagrabski, Michael Hopkins, Milda Petraityte, Ali Dehghantanha, Reza M. Parizi, Gregory Epiphaniou, and Mohammad Hammoudeh	
Forensic Investigation of Cross Platform Massively Multiplayer Online Games: Minecraft as a Case Study	153
Paul J. Taylor, Henry Mwiki, Ali Dehghantanha, Alex Akinbi, Kim-Kwang Raymond Choo, Mohammad Hammoudeh, and Reza M. Parizi	
Big Data Forensics: Hadoop Distributed File Systems as a Case Study....	179
Mohammed Asim, Dean Richard McKinnel, Ali Dehghantanha, Reza M. Parizi, Mohammad Hammoudeh, and Gregory Epiphaniou	

Internet of Things Camera Identification Algorithm Based on Sensor Pattern Noise Using Color Filter Array and Wavelet Transform ..	211
Kimia Bolouri, Amin Azmoodeh, Ali Dehghantanha, and Mohammad Firouzmand	
Protecting IoT and ICS Platforms Against Advanced Persistent Threat Actors: Analysis of APT1, Silent Chollima and Molerats	225
Samuel Grooby, Tooska Dargahi, and Ali Dehghantanha	
Analysis of APT Actors Targeting IoT and Big Data Systems: Shell_Crew, NetTraveler, ProjectSauron, CopyKittens, Volatile Cedar and Transparent Tribe as a Case Study	257
Paul J. Taylor, Tooska Dargahi, and Ali Dehghantanha	
A Cyber Kill Chain Based Analysis of Remote Access Trojans	273
Reyhaneh HosseiniNejad, Hamed HaddadPajouh, Ali Dehghantanha, and Reza M. Parizi	
Evaluation and Application of Two Fuzzing Approaches for Security Testing of IoT Applications	301
Omar M. K. Alhawi, Alex Akinbi, and Ali Dehghantanha	
Bibliometric Analysis on the Rise of Cloud Security	329
Lim Sze Thiam, Tooska Dargahi, and Ali Dehghantanha	
A Bibliometric Analysis of Botnet Detection Techniques	345
Shehu Amina, Raul Vera, Tooska Dargahi, and Ali Dehghantanha	
Security in Online Games: Current Implementations and Challenges	367
Reza M. Parizi, Ali Dehghantanha, Kim-Kwang Raymond Choo, Mohammad Hammoudeh, and Gregory Epiphaniou	

Contributors

Alex Akinbi School of Computer Science, Liverpool John Moores University, Liverpool, UK

Omar M. K. Alhawi Department of Computer Science, University of Salford, Manchester, UK

Shehu Amina School of Computing, Science and Engineering, University of Salford, Manchester, UK

Mohammed Asim Department of Computer Science, University of Salford, Manchester, UK

Amin Azmoodeh Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

Department of Computer Science & Engineering, Shiraz University, Shiraz, Iran

Kimia Bolouri Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

Kim-Kwang Raymond Choo Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX, USA

Tooska Dargahi Department of Computer Science, School of Computing, Science and Engineering, University of Salford, Manchester, UK

Ali Dehghantanha Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

Gregory Epiphaniou Wolverhampton Cyber Research Institute (WCRI), School of Mathematics and Computer Science, University of Wolverhampton, Wolverhampton, UK

Mohammad Firouzmand Iranian Research Organization for Science and Technology (IROST), Tehran, Iran

Dimitris Gritzalis Department of Informatics, Athens University of Economics and Business, Athens, Greece

Samuel Grooby Department of Computer Science, School of Computing, Science and Engineering, University of Salford, Manchester, UK

Hamed HaddadPajouh Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

Mohammad Hammoudeh School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester, UK

Mohammad Heydari Department of Computing and Informatics, Bournemouth University, Poole, UK

Sajad Homayoun Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran

Michael Hopkins School of Computing, Science, and Engineering, University of Salford, Manchester, UK

Reyhaneh HosseiniNejad Pishtazan Higher Education Institute, Shiraz, Iran

Vasileios Katos Department of Computing and Informatics, Bournemouth University, Poole, UK

Dean Richard McKinnel Department of Computer Science, University of Salford, Manchester, UK

Henry Mwiki School of Computing, Science and Engineering, University of Salford, Manchester, UK

Alexios Mylonas Department of Computing and Informatics, Bournemouth University, Poole, UK

Krzysztof Nagrabski School of Computing, Science, and Engineering, University of Salford, Manchester, UK

Reza M. Parizi Department of Software Engineering and Game Development, Kennesaw State University, Marietta, GA, USA

Milda Petraityte School of Computing, Science, and Engineering, University of Salford, Manchester, UK

S. Sangeetha Department of Information Technology, PSG College of Technology, Coimbatore, Tamil Nadu, India

G. Sudha Sadasivam Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, Tamil Nadu, India

Paul J. Taylor School of Computing, Science and Engineering, University of Salford, Manchester, UK

Yee-Yang Teing Faculty of Computer Science and Information Technology, University Putra Malaysia, Seri Kembangan, Malaysia

Lim Sze Thiam School of Computing, Science and Engineering, University of Salford, Manchester, UK

Raul Vera School of Computing, Science and Engineering, University of Salford, Manchester, UK

Big Data and Internet of Things Security and Forensics: Challenges and Opportunities



Amin Azmoodeh, Ali Dehghanianha, and Kim-Kwang Raymond Choo

Abstract With millions to billions of connected Internet of Things (IoT) devices and systems sending heterogeneous raw and processed data through the IoT network, we need to be able to effectively utilize big data analytical techniques and solutions and ensure the security and privacy of IoT data and services against the broad range of attackers. Further complicating the challenge is the increasing number of nodes and complexity of the IoT network and ecosystem, for example the increasing number and size of audit and security logs and intrusion data to be collected and analyzed. The purpose of this handbook is to explore cyber security, forensics and threat intelligence challenges and solutions relating to IoT and big data.

Keywords Big data · Internet of Things · IoT · Security · Forensic

1 Introduction

Internet of Things (IoT) refers to a wide pervasive network of (smart) Internet-connected devices, such as smart vehicles, embedded systems, smart sensors, and other devices or systems that autonomously sense, store, transfer and process collected data [1, 2]. In recent years, IoT has played an increasingly integrated

A. Azmoodeh

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

Department of Computer Science & Engineering, Shiraz University, Shiraz, Iran

e-mail: azmoodeh@cse.shirazu.ac.ir

A. Dehghanianha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada
e-mail: ali@cybersciencelab.org

K.-K. R. Choo

Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX, USA

role in our society, for example in agriculture [3], health [4], transportation [5], military [6], and so on. However, the prevalence and vast amount of sensitive and private information in a typical IoT network have also attracted the attention of cybercriminals [7]. The complexity of an IoT system and the ecosystem (e.g. big data sources and storage locations) complicates our efforts to secure such big data. The latter generally comprise significant amount of (heterogeneous) information gathered from different sources that require specific techniques and solutions for privacy preserving and forensics investigation [8].

There are a large number of IoT specific and related challenges, such as those due to the distributed and size of IoT networks. For example, cryptography is generally required to ensure the security of a system, although the resource constrain nature of IoT devices may limit the direct application of conventional cryptographic solutions such as key management and storage.

Given the likelihood of IoT devices and systems being targeted in a malicious attack/activity, we will need to ensure that we have the capability for IoT and big data forensics. Unlike IoT security, IoT and big data forensics are generally less under-studied [9]. Challenges associated with IoT and big data forensics include locating, recovering and preserving reliable forensic artifacts [10]. Findings from data analytics and forensic investigation can also inform cyber security strategy formulation, and raise cyber threat awareness (e.g. cyber threat intelligence). Hence, there is a need for robust and effective cyber threat intelligence solutions [11–13].

2 Book Outline

This handbook presents existing state-of-the-art advances from both academia and industry, in IoT and big data security and forensics. The remainder of the book is structured as follows.

The second chapter [14] reviews existing data mining solutions, particularly their application in big data context. In third chapter [15], the authors survey existing authentication and access control for IoT devices, and in fourth chapter [16], Heydari et al. present an IoT access control scheme.

The fifth chapter [17] presents an IoT forensic framework and uses Seafile open source cloud storage as a case study. In the subsequent chapter [18] (sixth chapter), a distributed file system forensic approach is presented, which is used to guide the investigation of Ceph. In seventh and eight chapters [19] and [20], the authors forensically examine Minecraft, a Massively Multiplayer Online Game, and the Hadoop distributed file system environment. In ninth chapter [21], a forensic IoT source camera identification algorithm is presented, which uses the camera's sensor pattern noise from the captured image.

In order to mitigate attacks from Advanced Persistent Threat (APT) actors, Grooby et al. [22] in tenth chapter present a cyber defense triage process for three APT groups targeting IoT infrastructure, namely: *APT1, Molerats, and Silent Chollima* using Diamond Model of Intrusion Analysis and the Lockheed

Martin intrusion kill chain. In eleventh chapter [23], the authors analyze APT actors *Shell_Crew*, *NetTraveler*, *ProjectSauron*, *CopyKittens*, *Volatile Cedar and Transparent Tribe*. In twelfth chapter [24], the authors analyze the characteristics of remote-controlled real-world Trojans using the Cyber Kill Chain.

In thirteenth chapter [25], the authors introduce a method to leverage different crashes discovered from two fuzzing approaches, to boost fuzzers by concentrating on utilized test cases. For evaluation, code coverage is used.

In fourteenth [26] and fifteenth chapters [27], a survey of the cloud security literature published between 2010 and 2018, and a survey of botnet detection approaches, are presented respectively.

Finally, in sixteenth chapter [28], the authors study game security solutions and explain how one may circumvent such solutions.

References

1. X. Li, J. Niu, S. Kumari, F. Wu, A. K. Sangaiah, K.-K. R. Choo, A three-factor anonymous authentication scheme for wireless sensor networks in internet of things environments, *Journal of Network and Computer Applications*. [doi:<https://doi.org/10.1016/j.jnca.2017.07.001>](https://doi.org/10.1016/j.jnca.2017.07.001).
2. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): A vision, architectural elements, and future directions, *Future generation computer systems* 29 (7) (2013) 1645–1660.
3. M. Roopaei, P. Rad, K.-K. R. Choo, Cloud of things in smart agriculture: Intelligent irrigation monitoring by thermal imaging, *IEEE Cloud Computing* 4 (1) (2017) 10–15.
4. S. Walker-Roberts, M. Hammoudeh, A. Dehghantanha, A systematic review of the availability and efficacy of countermeasures to internal threats in healthcare critical infrastructure, *IEEE Access* 6 (2018) 25167–25177. [doi:\[10.1109/ACCESS.2018.2817560\]\(https://doi.org/10.1109/ACCESS.2018.2817560\)](https://doi.org/10.1109/ACCESS.2018.2817560).
5. G. Epiphaniou, P. Karadimas, D. K. B. Ismail, H. Al-Khateeb, A. Dehghantanha, K. K. R. Choo, Non-reciprocity compensation combined with turbo codes for secret key generation in vehicular ad hoc social iot networks, *IEEE Internet of Things Journal* (2017) 1–1. [doi:\[10.1109/JIOT.2017.2764384\]\(https://doi.org/10.1109/JIOT.2017.2764384\)](https://doi.org/10.1109/JIOT.2017.2764384).
6. A. Azmoodeh, A. Dehghantanha, K.-K. R. Choo, Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning, *IEEE Transactions on Sustainable Computing*.
7. M. Conti, A. Dehghantanha, K. Franke, S. Watson, Internet of things security and forensics: Challenges and opportunities (2018).
8. Y.-Y. Teing, D. Ali, K. Choo, M. T. Abdullah, Z. Muda, Greening cloud-enabled big data storage forensics: Syncany as a case study, *IEEE Transactions on Sustainable Computing*.
9. S. Watson, A. Dehghantanha, Digital forensics: the missing piece of the internet of things promise, *Computer Fraud & Security* 2016 (6) (2016) 5–8.
10. Y.-Y. Teing, A. Dehghantanha, K.-K. R. Choo, L. T. Yang, Forensic investigation of p2p cloud storage services and backbone for iot networks: BitTorrent sync as a case study, *Computers & Electrical Engineering* 58 (2017) 350–363.
11. A. Dehghantanha, M. Conti, T. Dargahi, et al., *Cyber threat intelligence*, Springer.
12. A. Azmoodeh, A. Dehghantanha, M. Conti, K.-K. R. Choo, Detecting crypto-ransomware in iot networks based on energy consumption footprint, *Journal of Ambient Intelligence and Humanized Computing* 1–12.
13. H. H. Pajouh, R. Javidan, R. Khayami, D. Ali, K.-K. R. Choo, A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in iot backbone networks, *IEEE Transactions on Emerging Topics in Computing*.

14. S. Sangeetha, G. SudhaSadasivam, Privacy of big data – a review, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. .
15. S. Grooby, T. Dargahi, A. Dehghantanha, A bibliometric analysis of authentication and access control in iot devices, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
16. M. Heydari, A. Mylonas, V. Katos, D. Gritzalis, Towards indeterminacy-tolerant access control in iot, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
17. Y.-Y. Teing, S. Homayoun, A. Dehghantanha, K.-K. R. Choo, R. M. Parizi, M. Hammoudeh, G. Epiphanou, Private cloud storage forensics: Seafile as a case study, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
18. K. Nagrabski, M. Hopkins, M. Petraityte, A. Dehghantanha, R. M. Parizi, G. Epiphanou, M. Hammoudeh, Distributed filesystem forensics: Ceph as a case study, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
19. D. P. J. Taylor, H. Mwiki, A. Dehghantanha, A. Akibini, K. K. R. Choo, M. Hammoudeh, R. M. Parizi, Forensic investigation of cross platform massively multiplayer online games: Minecraft as a case study, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
20. M. Asim, D. R. McKinnel, A. Dehghantanha, R. M. Parizi, M. Hammoudeh, G. Epiphanou, Big data forensics: Hadoop distributed file systems as a case study, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
21. K. Bolouri, A. Azmoodeh, A. Dehghantanha, M. Firouzmand, Internet of things camera identification algorithm based on sensor pattern noise using color filter array and wavelet transform, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
22. S. Grooby, T. Dargahi, A. Dehghantanha, Protecting IoT and ICS platforms against advanced persistent threat actors: Analysis of APT1, Silent Chollima and molerats, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
23. P. J. Taylor, T. Dargahi, A. Dehghantanha, Analysis of apt actors targeting IoT and big data systems: Shell_crew,nettraveler, projectsauron, copykittens, volatile cedar and transparent tribe as a case study, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
24. R. HosseiniNejad, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, A cyber kill chain based analysis of remote access trojans, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
25. O. M. K. Alhawi, A. Akinbi, A. Dehghantanha, Evaluation and application of two fuzzing approaches for security testing of IoT applications, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
26. L. S. Thiam, T. Dargahi, A. Dehghantanha, Bibliometric analysis on the rise of cloud security, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
27. S. Amina, R. Vera, T. Dargahi, A. Dehghantanha, A bibliometric analysis of botnet detection techniques, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.
28. R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, M. Hamoudeh, G. Epiphanou, Security in online games: Current implementations and challenges, in: A. Dehghantanha, K.-K. R. Choo (Eds.), *Handbook of Big Data and IoT Security*, Springer, Cham. <https://doi.org/10.1007/978-3-030-10543-3>.

Privacy of Big Data: A Review



S. Sangeetha and G. Sudha Sadasivam

Abstract Big data has become Buzzword in recent years. It is due to the fact that voluminous amount of structured, semi structured and unstructured data that is generated in the digital era. But, this huge data can be tracked and used for monetary benefits which thwart individual's privacy. Hence numerous fruitful researches are made in privacy preservation. This book chapter lays emphases on the state-of-art privacy preserving data mining mechanisms and reviews the application of these mechanisms in big data environment.

Keywords Big data · Privacy · Hadoop · Spark · PPDM

1 Introduction

Privacy concerns with protecting disclosure of individual's data. Privacy is often confused with security. Security deals with Confidentiality, Integrity and Availability, whereas Privacy deals with appropriate use of an individual's information.

Data mining aims to elicit useful information from data, Data Mining techniques uncover useful business patterns and knowledge buried in large amount of data which is a threat to an individual's privacy. Privacy preservation in data mining aims to preserve data against disclosure. Preservation of privacy in data mining uses algorithms to alter the data so that the sensitive data and knowledge remains private even after the mining process. Privacy preservation preserves privacy during data collection and mining. Protecting data during data collection is called input privacy

S. Sangeetha (✉)

Department of Information Technology, PSG College of Technology, Coimbatore, Tamil Nadu, India

e-mail: ssa.it@psgtech.ac.in

G. Sudha Sadasivam

Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, Tamil Nadu, India

e-mail: gss.cse@psgtech.ac.in

and protection during mining state is called output privacy. A retailer's database on sales information can be collected by the supplier to perform knowledge discovery. The supplier may then favor one retailer over the other retailer. Instead privacy preservation helps in this collaborative mining by hiding retailer and customer information to create a win-win-win situation for retailer, supplier and customer.

Privacy can be protected by various techniques like generalization, suppression and perturbation. All these mechanisms modify the data in order to ensure secrecy. Too much modification of data increases privacy and the resulting data may become totally useless. Too little modification provides very good utility [1] but the privacy of such data cannot be guaranteed.

A noble privacy preserving approach must satisfy some goals including a well balanced privacy utility trade-off, minimal side effects, reduced information loss, high utility of the transformed dataset and ensure zero hiding failure which is denoted by

$$HF = \frac{\#Rp(D')}{\#Rp(D)}$$

where HF = Hiding failure

#Rp = Count of sensitive patterns

D = Original database

D' = Sanitized database

As the technology evolves large amount of data generated by web applications and IoT devices including medical images, genome and social media data gets augmented on a daily basis. This data deluge makes users drown in data but starve for knowledge that call for effective usage of data mining techniques. These mining tasks performed on data questions the user's privacy. Hence this book chapter provides insights on privacy of big data. Section 2 introduces concepts of big data, its framework and technologies; Sect. 3 addresses the architecture of big data system to ingest, collect, store, process, extract and visualize large amount of real time data. Section 4 provides commonly used techniques for privacy preservation in big data systems. Section 5 provides theoretical aspects along with the metrics to measure data quality. The book chapter concludes with summary of privacy preservation framework used in big data application.

2 Big Data

This section provides overview of Big Data. Big Data era was initially governed by three V's—Volume, Velocity and Variety has been identified however in recent days these V's are extended. The following Fig. 1 illustrates the five V's of big data.

Volume: The data is really 'Big'. Data that is generated in various horizons like finance, health care, retail, sensor data, etc. is highly voluminous.



Fig. 1 Five V's of big data

Velocity: Velocity in big data perspective is the speed at which new data is generated and the way in which it is analyzed.

Variety: The variety denotes the diversity of big data that is generated.

Variability: Variability denotes the way in which the big data is captured.

Value: Denotes the importance of the data that is handled.

The five V's of big data can be used to classify a given data as big data. After identifying big data a framework must be devised to handle such big data. The following Sect. 2.1 discusses the big data framework.

2.1 Big Data Framework

To handle large amount of continuously changing real time data a Big Data Framework is needed. Hence this section addresses the Framework necessary for Big Data and Table 1 compares the commonly used Big data framework like Hadoop and Spark. Hadoop is a Big Data framework used to process huge data in distributed computing environment. Hadoop is named by one of the projects creator Doug Cutting, [2] it is a made up name of his kids elephant toy. Apache Hadoop ecosystem is made up of Map Reduce, HDFS, PIG, Hive, Zookeeper, Spark, Mahout, Oozie, HBase.

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware [3]. The primary advantages of HDFS is its high throughput, and fault tolerance.

Map Reduce is the programming paradigm for Big Data computing environment [4]. The model is capable of handling very large datasets using two primary functions mapper and reducer. Mapper processes a key/value pair and generates an intermediary result which is further processed by Reducer. Programs written using these functions are automatically parallelized and executed in large clusters using hadoop as shown in Fig. 2. These jobs runs in thousands of machines and

highly scalable with capability to process massive data. The client submits the job to Job Tracker. Job Tracker is responsible for coordinating the map reduce phases and assigns the submitted jobs to the Task Tracker. In Fig. 2 the job is split among three mappers and the intermediate output produced is further processed by reducer to get the final output.

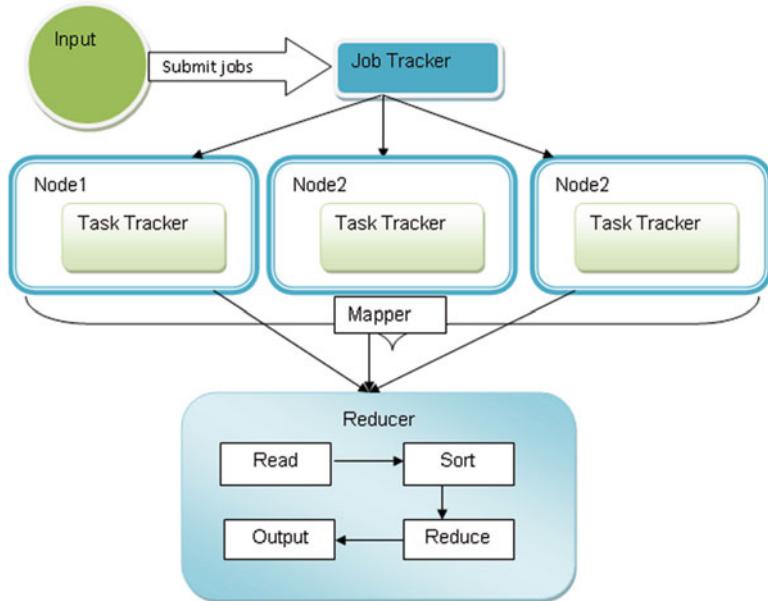


Fig. 2 Map Reduce

Apache Spark is a data processing tool that runs on distributed data collection. In recent days spark is gaining more popularity since it is faster than mapreduce. Apache spark uses a master/slave architecture. The master or driver controls and manages the cluster while the slave or worker are the executors that run individual java processes as shown in Fig. 3.

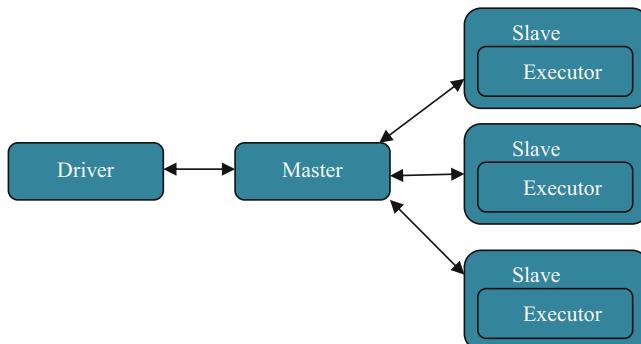


Fig. 3 Spark architecture

As spark performs iterative in-memory processing, it is more suitable for high speed analytics. Hadoop provides software framework for storing and executing applications on clusters. A comparison of Hadoop map-reduce and Spark is summarized in Table 1.

Table 1 Hadoop MapReduce vs Spark

Factors	Hadoop MapReduce	Spark
Data processing	In memory processing is not supported and no real time processing	Supports in-memory computing and real time processing
Speed	Slower than spark	Much faster
Disk read write	High	Low
Workload	Batch	Batch, interactive, iterative, and streaming
Scalability	Support very large datasets	Comparatively lower than hadoop
Ease of use	It is written in Java, and difficult to program. There is no interactive mode available with Hadoop MapReduce	Comes with user friendly API
Graph processing	Algorithms must be coded	Comes with a handy library named GraphX
Fault tolerance	Replication	RDD—Resilient Distributed Datasets
Security	Better security when compared to spark	Infancy
Cost	Uses disk hence cheaper	Needs expensive RAM hence costlier

3 Big Data Architecture

Section 2 discussed about the storage and processing of Big data. To perform real time analytics it is important to collect, ingest, store, process and visualize the data. In all these stages it is essential to protect users privacy. Hence this section elaborates the complete big data privacy preserving architecture. The data analytics architecture is depicted in Fig. 4. Input data is obtained from diverse sources like social media data, sensor data, log files, user profiles, financial information etc. The personal information of an individual such as Aadhar number and voter id must be removed before ingestion. Data ingestion collects and imports data for storing it in a database for further processing, which is performed using tools like Apache Sqoop, Apache nifi, Kafka or flume.

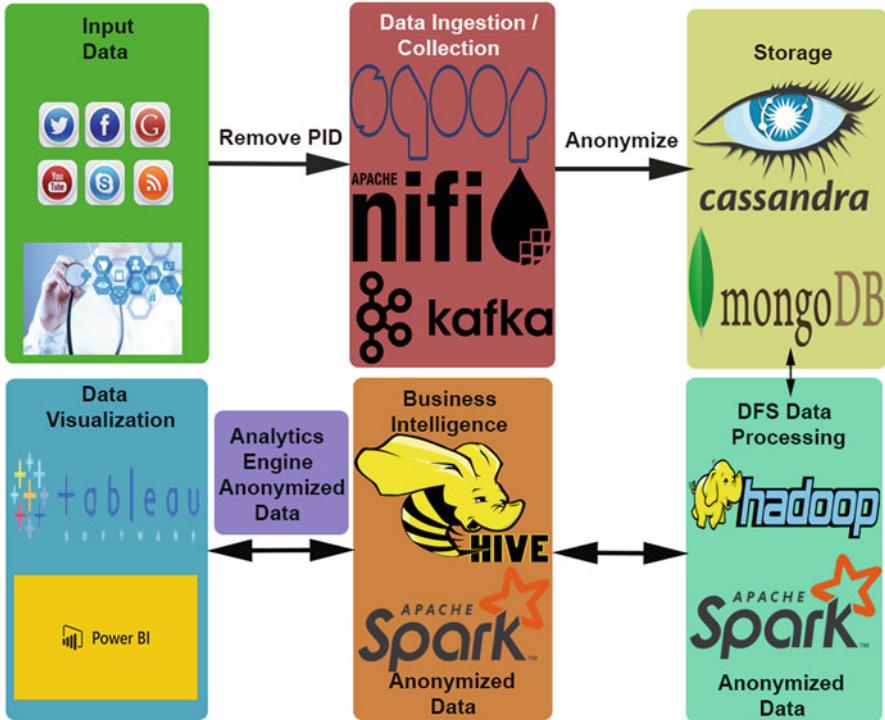


Fig. 4 Privacy preserving big data analytics architecture

The selection of data ingestion tool is based on the type of data that is used. In general Sqoop is used to move bulk data to Hadoop, Nifi and Flume are used to ingest csv file, flat files etc., Kafka can be used for real time data or streaming data processing. Quasi identifiers can be anonymized using any of the privacy preservation techniques and stored in Distributed File System using Casandra or mongo DB. Big Data processing can be performed using Hadoop or Spark. Business intelligence can be obtained using Hive or Spark SQL and given as input to the analytics engine. Tools like Tableau, Microsoft power BI can be used for data visualization and illustrating result. The privacy of the data has to be protected in the entire spectrum of big data in data source, data ingestion, Storage, DFS data processing, Business Intelligence, Analytics engine and Visualization of Big data analytics architecture. Hence the privacy of input data must be guaranteed throughout the lifecycle of big data architecture. Therefore Sect. 4 provides the state-of-art privacy preservation mechanisms.

4 Literature Review

This section outlines the Privacy Preserving data mining techniques that can be applied to big data. Privacy preserving data mining is broadly classified but not limited to the following [5] hierarchy as depicted in Fig. 5.

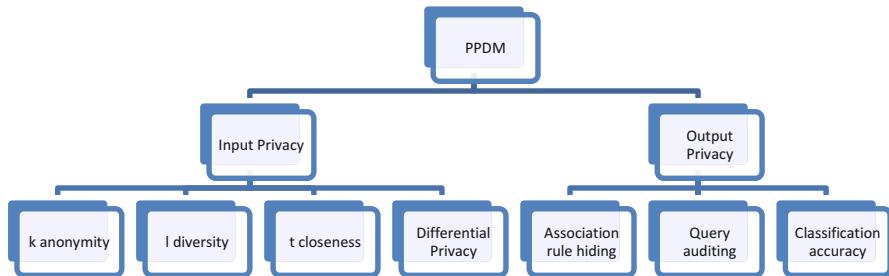


Fig. 5 PPDM hierarchy

In input privacy, one is primarily concerned with publishing anonymized data. In output privacy the result is perturbed or audited in order to preserve privacy. Other than these broad classifications cryptography based mechanisms are also available. Murot Kantarciooglu [6] address the secured multiparty communication in horizontally and vertically partitioned data. Vertically partitioned data has different types of information at each site, each has partial information. In contrast for horizontally partitioned data, every site has complete information on different entities. Cryptography based techniques provide very high privacy for transporting data but these mechanisms does not address secured mining. A data miner can be an adversary and tries to identify individual through mining process. Secured mining ensures privacy even though the miner is an adversary. Hence our prime focus in this literature survey is on input privacy and output privacy.

4.1 Input Privacy

The key emphasis of input privacy is to publish anonymized data. Different approaches used for input privacy is explained below.

4.1.1 K Anonymity

Often hospitals and government organizations release data due to their disclosure policies in enforcement. There are three types of attributes in the data as in Fig. 6.



Fig. 6 Types of attributes

Individual identifiers uniquely identifies individuals e.g.: Aadhar card number, driving license number or voter id number. Hence these attributes are removed before publishing the data. Quasi identifiers are the attributes like street, date of birth, pin code. These quasi identifiers are matched with external database to identify an individual. Even though the individual identifiers are removed these quasi identifiers can be used by the adversary to track the individual. Sensitive attributes for e.g.: in hospital dataset, disease can be a sensitive attribute, ccv number can be a sensitive attribute in financial dataset, and annual income can be a sensitive attribute in census dataset.

In the literature of privacy preservation L. Sweeny [7] has set a remarkable benchmark by presenting k anonymity protection model which is further researched by several authors [8–10]. In her study she proved that quasi identifiers like ZIP code, Birth date and sex can be used to identify an individual. A health insurance company removed individual identifiers and believed that the data is anonymous and provided an open copy to researchers. Later Sweeny purchased voter registration list with name, address, zip code, birth date, and gender of each voter. This information can be linked using ZIP code, birth date and gender to the medical information, thereby For example, governor of Massachusetts at that time and his medical records were in the GIC data. According to the Cambridge Voter list, six people had his particular birth date; only three of them were men; and, he was the only one in his 5-digit ZIP code. Hence Sweeny proved that it is possible to identify an individual by linking two different databases and proposed k anonymity model to overcome the privacy breach.

Definition 1 *The k-anonymity privacy requirement for publishing microdata requires that each equivalence class contains at least k records.*

Table 2 shows an example medical dataset with attributes state, pincode, gender, age and problem. The quasi identifiers are pincode, gender and age. The sensitive attribute is problem. Table 3 shows the anonymized table with **k = 2**.

Table 2 Example medical data

S.No.	Pincode	Gender	Age	Problem
1	231213	M	20	Heart disease
2	231217	F	25	Diabetes
3	231217	F	27	Heart disease
4	247775	M	30	Diabetes
5	225301	M	37	Diabetes
6	221002	M	35	Obesity
7	221004	M	40	Cancer
8	221002	M	43	Cancer
9	221003	M	44	Cancer

Table 3 Anonymized table with $k = 2$

S.No.	Pincode	Gender	Age	Problem
1	23121	M	20–27	Heart disease
2	23121	F	20–27	Diabetes
3	23121	F	20–27	Heart disease
4	24777	M	30–37	Diabetes
5	22530	M	30–37	Diabetes
6	22100	M	30–37	Obesity
7	22100	M	≥ 40	Cancer
8	22100	M	≥ 40	Cancer
9	22100	M	≥ 40	Cancer

In the month of October 2006 [11], Netflix removed the identity of users and released a dataset for improving its recommendation system. Researchers at the university of Texas combined the anonymized Netflix dataset and IMDb database which lead to identification of few users.

The major drawbacks of k -anonymity is protection against attribute disclosure. This is addressed by p -sensitive anonymity by Truta TM [12].

4.1.2 ℓ diversity

A. Machanavajjhala [13] presented two attacks, the homogeneity attack and the background knowledge attack in k anonymity and proposed new privacy model called ℓ -diversity.

Limitations of k Anonymity

Homogeneity Attack Eventhough the anonymized table is published there is a possibility of homogeneity attack. For e.g. when a person is hospitalised, their neighbours know his gender as M, pincode is 221002, age is 43 now running a search query on Table 3 returns tuples 7, 8 and 9. Eventhough the table is anonymized by examining the query results the neighbours can conclude that the person has cancer. This is called homogeneity attack.

Background Knowledge Attack With known quasi identifiers the table is analysed. The matching tuples is further filtered based on the background knowledge and sensitivity of the person is identified. For e.g.: Consider Hana is 25 year old women living in zip code 231217. She is in first equivalence class in Table 3. If an individual knows that Hana has low risk for heart disease then he can conclude that Hana has Diabetes.

k -Anonymity does not protect attacks based on homogeneity attack and background knowledge attack. ℓ -diversity uses the theoretic notation of entrophy, recur-

sive (c, ℓ) diversity, positive disclosure-recursive (c, ℓ) diversity, negative/positive disclosure-recursive (c_1, c_2, ℓ) diversity, and multi attribute ℓ diversity.

Definition 2 An equivalence class is said to have ℓ -diversity if there are at least ℓ “well-represented” diverse values for the sensitive attribute.

A table is said to have ℓ -diversity if every equivalence class of the table has ℓ -diversity. Entropy of an equivalence class E is defined to be

$$\text{Entropy}(E) = - \sum_{s \in S} p(E, s) \log p(E, s)$$

In which S is the domain of the sensitive attribute, and $p(E, s)$ is the fraction of records in E that have sensitive value s . A table is said to have entropy ℓ -diversity if for every equivalence class E , $\text{Entropy}(E) \geq \log \ell$. A. Machanavajjhala shown that a k-anonymized dataset allows attacks due to lack of diversity.

4.1.3 t-Closeness

NinghuiLi [14] addressed the limitations in ℓ diversity and proposed t-closeness based on Earth Mover Distance.

Limitations of ℓ Anonymity

Skewness Attack This attack is created by the overall data distribution when it is skewed. In such a case satisfying ℓ -diversity will not prevent disclosure. Consider the data shown in Table 4 any distinct diversity, entropy or recursive diversity does not prevent information disclosure.

Similarity Attack Although the sensitive values are distinct and their semantic significance is similar there is still an information disclosure.

Table 4 Example medical data

S.No.	Pincode	Gender	Age	Problem
1	231213	M	20	Heart disease
2	231217	F	25	Heart disease
3	231217	F	27	Heart disease
4	247775	M	30	Heart disease
5	225301	M	37	Heart disease
6	221002	M	35	Heart disease
7	221004	M	40	Heart disease
8	221002	M	43	Heart disease
9	221003	M	44	Cancer

Definition 3 An equivalence class is said to have t -closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t . A table is said to have t -closeness if all equivalence classes have t -closeness.

t -closeness is derived for a dataset based on Earth Movers Distance (EMD) which is calculated as follows.

EMD for Numerical Attributes

Let the attribute domain be $\{v_1, v_2, \dots, v_m\}$ the distance between two distributions v_i and v_j is given by,

$$\text{ordereddist}(v_i, v_j) = \frac{|i - j|}{m - 1}$$

EMD for Hierarchical Attributes

For categorical attributes tree hierarchy is constructed. It is based on minimum level to which the two values are generalized to same level in the hierarchy.

$$\text{dist}(v_1, v_2) = \frac{\text{level}(v_1, v_2)}{H}$$

where H denotes height of the tree and $\text{level}(v_1, v_2)$ denotes the height of the lowest common ancestor node of v_1 and v_2 .

The Table 5 is 0.278 t -close on the sensitive attribute disease.

Table 5 t closeness

S.No	Pincode	Gender	Age	Problem
1	2	M	≤ 40	Heart disease
2	2	F	≤ 40	Diabetes
7	2	M	≤ 40	Cancer
4	2	M	≤ 43	Diabetes
8	2	M	≤ 43	Cancer
6	2	M	≤ 43	Obesity
5	2	M	≤ 45	Diabetes
3	2	F	≤ 45	Heart disease
9	2	M	≤ 45	Cancer

4.1.4 Differential Privacy

The goal of Differential privacy is to increase the accuracy of queries from statistical databases and decreasing the possibility of identifying the records of individual [15–17].

Dwork [18] suggest a new measure to capture the increased risk to individuals privacy occurred by participating in a database. Consider a survey is made on urinating in the shower and the questions are asked to 50 people out of them 10 has responded yes and 40 has responded as no. When a new person answers the survey, his answer can be easily identified by comparing the results before his participation and after participation. The original survey result can be considered as D and new result can be considered D'. Now $|D' - D|$ uniquely identifies the answer provided by the individual.

Definition 4 A randomized function K gives ϵ -differential privacy for all datasets D_1 and D_2 differing on at most one element, and all $S \subseteq \text{Range}(k)$ when

$$\Pr[k(D_1) \in S] \leq \exp(\epsilon) \times \Pr[k(D_2) \in S]$$

Differential privacy [DP] provides useful access to data while preventing privacy breaches to individual records using Laplace mechanism and Exponential mechanism. The data analyst queries the database but the query is not directly answered by the database instead an intermediate privacy guard provides response (Fig. 7). The privacy guard analyses the query given by analyst and gets the result from database and adds noise to the results and replies back the analyst with some noisy response [19]. Differential privacy is becoming a popular research area as it guarantees data privacy. Further, popular attacks on privacy like similarity attack, background knowledge attack, skewness attack cannot work for differential privacy. Differential privacy does not cause side effects. Differential Privacy ensures utility as noise addition is minimal thus providing a close approximation of original results. Standard privacy mechanisms like suppression, generalization and perturbation modifies the original data whereas differential privacy leaves original data intact and does not perform any changes to the dataset (Fig. 7).

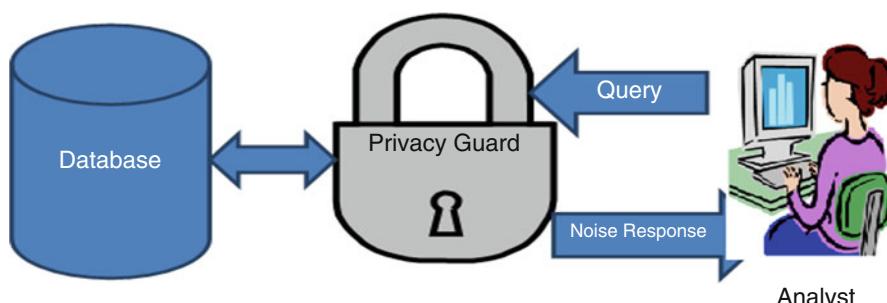


Fig. 7 Differential privacy

4.2 Output Privacy

The key emphasis of output privacy is to perturb the output of various data mining algorithms to ensure privacy. Some of the methods are discussed in the following section.

4.2.1 Association Rule Hiding

S. Verykios [20] proposed five algorithms for sensitive association rule hiding. They have analysed the time complexity of all the algorithms and side effects produced by the algorithms.

An association rule is sensitive if the support and confidence of the rule is greater than a minimum threshold. Support and confidence are the basic measures of any given association rules and is denoted by

$$Supp(A \rightarrow B) = \frac{|A \cap B|}{n}$$

$$Conf(A \rightarrow B) = \frac{Supp(A \cup B)}{Supp(A)}$$

where n—denotes the total number of transactions.

The algorithm identifies the sensitive rules and increases or decreases the support and confidence of the rule thereby the support and confidence becomes less than the minimum threshold. This complete process of modifying the database to hide sensitive rule is called “sanitization”. Hence the output privacy is ensured.

4.2.2 Query Auditing

Query auditing is a privacy preservation mechanism to investigate and avoid private data disclosure from the database. The Subha U. Nabar [21] focus on answering to the queries like sum, min, median and max in statistical databases. The auditing is done on both online and offline modes. In online mode the session is interactive and in offline mode the query response happens at time out intervals. They examine full disclosure and partial disclosure of the database.

4.2.3 Classification Accuracy

Decision tree classification is an important data mining method to establish a classification system. In privacy preserving data mining the challenge is to develop the decision tree from perturbed data which provides a novel reconstruction

procedure that closely approximates the original distribution. R. Agrawal [22] proposed two algorithms namely By Class and Local based on Bayesian procedure. Weiping Ge [23] identified deficiencies in [22] like labelled, Boolean and categorical attribute perturbation is not provided and poor reconstruction accuracy and proposed PPCART algorithm. The PPCART algorithm is based on transition probability matrix.

5 Metrics to Measure Data Quality

All the privacy preservation data mining techniques discussed so far either modifies, perturbs, generalizes or suppresses the data. Further encryption is also performed if cryptographic privacy mechanisms are used. Hence it is important to measure the data quality after applying all these mechanisms. The dissimilarity between the original database D and sanitized database D' with n transactions and A attributes can be measured using a range of techniques. Visualization tools can be used to compare the D and D' or the contents of data can be compared or size of data before and after sanitization can be compared. The more standard formula for comparing dissimilarity is follows

$$Diss(D, D') = \frac{\sum_{i=1}^n |fD(i) - fD'(i)|}{\sum_{i=1}^n fD(i)}$$

where $fD(i)$ is the frequency before sanitization and $fD'(i)$ is the frequency after sanitization.

For aggregation or generalization the amount of information loss (IL) must be measured. For a generalization hierarchy GT with depth h is measured using the following formula

$$IL(D') = \frac{\sum_{i=1}^A \sum_{j=1}^{i=n} \frac{h}{|GT(Ai)|}}{|D| * |A|}$$

6 Summary of Existing Big Data Privacy Preservation Models

All the traditional works of privacy preservation work on limited datasets. Recent advancements have led to voluminous data. Further research is needed to handle such data and powerful algorithms must be developed for the same. The term big data is evolving rapidly and fruitful research is made to solve such big data

Table 6 Summary of big data privacy

S.No.	Research paper	Privacy model/framework	Issue addressed	Disadvantage
1	Privacy-Preserving Big Data publishing [5]	k-anonymity and ℓ -diversity	Proposed the first approach for handling massive data using map reduce	Prone to attacks like homogeneity, background knowledge etc.
2	Privacy Preserving Data Mining on big data computing platform: trends and future [25]	The authors brief the typical big data solutions namely Airavat [26], PRISM [27], and PPLA [28]	Investigates the popular Privacy Preserving Data Mining (PPDM) techniques and highlights its importance in big data platform	The tool requires human participation and does not address multidimensional anonymization
3	Hiding a needle in a haystack: privacy preserving Apriori algorithm in MapReduce framework [29]	Noise generation	Proposes privacy preserving algorithm in MapReduce framework. The fundamental idea is to hide original data in noisy environment to prevent it from privacy breaches	Must be cautious on the amount of noise added which directly affects the data utility
4	Differential Privacy Preserving in Big Data analytics for Connected Health [30]	Differential Privacy	Proposes protection for body area networks in Big Data environment	Differential Privacy provides either very little privacy or very little utility or neither
5	Big healthcare data: preserving security and privacy [31]	HybrEx	Highlighted the crucial need for health care data privacy through the Forbes report [32]. The authors explains the usage of HybrEx in healthcare	The tool requires human participation and does not address multidimensional anonymization
6	Big data privacy: a technological perspective and review [33]	HybrEx	The authors explain the usage of HybrEx with its four categories	The tool requires human participation and does not address multidimensional anonymization
7	Proximity-Aware Local-Recoding Anonymization with MapReduce for Scalable Big Data Privacy Preservation in Cloud [3]	Local recoding anonymization	A scalable two phase clustering approach is proposed based on map reduce	The mechanism proposed should be optimized to handle very large datasets

(continued)

Table 6 (continued)

S.No.	Research paper	Privacy model/framework	Issue addressed	Disadvantage
8	A Discussion of Privacy Challenges in User Profiling with Big Data Techniques: The EEXCESS Use Case [34]	EEXCESS	User profile prevention challenges and how it is achieved using EEXCESS	Only the challenges are discussed. Privacy mechanisms are not elaborated
9	Making Big Data, Privacy, and Anonymization work together in the Enterprise: Experiences and Issues [35]	k-anonymity	Provides the practical use of combining privacy with big data in enterprise environment	Increase in longitude of data and sequential occurrences are not addressed
10	A MapReduce Based Approach of Scalable Multidimensional Anonymization for Big Data Privacy Preservation on Cloud [36]	Mondrian multidimensional anonymity	A scalable median finding algorithm is proposed in map reduce environment	The approach must be extended to explore scalable privacy preservation
11	Differential privacy: its technological prescriptive using big data [37]	Differential privacy	Provides a review of Differential Privacy basics with its current usage and research gaps	Differential Privacy provides either very little privacy or very little utility or neither
12	Big Data Privacy in Biomedical Research [38]	Differential Privacy and Cryptographic techniques	The paper provides a strong emphasis on privacy for biomedical research	Cryptographic mechanism may not provide privacy preserving data mining
13	Protection of Big Data Privacy [39]	Cryptographic techniques and Anonymization techniques	The paper provides survey of anonymization and cryptographic mechanisms that can be used in big data lifecycle	Existing anonymization techniques are surveyed which cannot be used for big data
14	Big Data security and privacy: A review [40]	Cryptographic techniques, Anonymization techniques and iRODs	The paper provides survey of anonymization, cryptographic mechanisms and iRODS	Novel big data privacy methods are not experimented
15	Big Data Privacy in the Internet of Things Era [41]	OpenIoT, Lab of Things (LoT), The Hub of All Things (HAT)	The paper highlights various stakeholders responsible for protecting privacy	Scalable privacy preserving algorithms are not developed

challenges. In addition to previous privacy preserving metrics the scalability of dataset should be given extreme importance. Recent technological frameworks like [24] Map Reduce, Spark etc. are capable of handling such big data. Table 6 highlights the existing big data PPDM techniques.

7 Conclusion

This book chapter provided a thorough study of various privacy preserving data mining techniques and its application in big data environment. Insights on nature of big data and its technological counterparts are provided. Big data analysis tools like hadoop and spark help data scientists to tackle the business challenges by providing solutions for Business Intelligence with dashboards, reports, queries and predictive analytics like optimization, forecasting, statistical analysis. However individual user's privacy must be ensured by data scientist. Hence the privacy preservation mechanisms detailed in this book chapter helps data scientists to achieve privacy preserving data analytics.

The data that is generated can be structured, semi structured, quasi structured and unstructured. Around 80% of data is unstructured hence existing algorithms must be reframed to enable unstructured privacy preserved data handling. Existing cryptography based and randomization based algorithms are designed for smaller datasets. Hence new privacy preserving framework is needed for handling voluminous data that is distributed among several clusters. Few challenges including development of efficient and scalable algorithms suitable for big data, utility of sanitized data, data reminiscence, data provenance, audit trails and privacy guarantee. Future research directions need to address these challenges.

References

1. Bertino, Elisa and Lin, Dan and Jiang, Wei, 2008 A Survey of Quantification of Privacy Preserving Data Mining Algorithms, *Privacy-Preserving Data Mining: Models and Algorithms*, Springer US, 183—205.
2. Hadoop: Toddler Talk Provides Big Data Name <https://www.cnbc.com/id/100769719#>
3. X. Zhang *et al.*, Proximity-Aware Local-Recoding Anonymization with MapReduce for Scalable Big Data Privacy Preservation in Cloud, in *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2293–2307, Aug. 1 2015.
4. Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (January 2008), 107–113.
5. Hessam Zakerzadeh, Charu C. Aggarwal, and Ken Barker. 2015. Privacy-preserving big data publishing. In Proceedings of the 27th International Conference on Scientific and Statistical Database Management (SSDBM '15), Amarnath Gupta and Susan Rathbun (Eds.). ACM, New York, NY, USA, Article 26, 11 pages.
6. Kantarcioglu, Murat, 2008, A Survey of Privacy-Preserving Methods Across Horizontally Partitioned Data, *Privacy-Preserving Data Mining: Models and Algorithms*, Springer US, Pages: 313–335

7. Latanya Sweeney. 2002. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 10, 5 (October 2002), 557–570.
8. Pierangela Samarati and Latanya Sweeney, Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression, 1998.
9. K. LeFevre, D. J. DeWitt and R. Ramakrishnan, Mondrian Multidimensional K-Anonymity, *22nd International Conference on Data Engineering (ICDE'06)*, 2006, pp. 25–25.
10. Hua, Ming and Pei, Jian, 2008, A Survey of Utility-based Privacy-Preserving Data Transformation Methods, Privacy-Preserving Data Mining: Models and Algorithms, Springer US, pages:207–237
11. A. Narayanan and V. Shmatikov, Robust De-anonymization of Large Sparse Datasets, *2008 IEEE Symposium on Security and Privacy (sp 2008)*, Oakland, CA, 2008, pp. 111–125.
12. T. M. Truta and B. Vinay, Privacy Protection: p-Sensitive k-Anonymity Property, *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, Atlanta, GA, USA, 2006, pp. 94–94.
13. Machanavajjhala, Ashwin & Gehrke, Johannes & Kifer, Daniel & Venkitasubramaniam, Muthuramakrishnan. (2006). l-Diversity: Privacy Beyond k-Anonymity. *ACM Transactions on Knowledge Discovery From Data*.
14. NinghuiLi, Tiancheng Li, Suresh Venkatasubramanian, t-Closeness: Privacy Beyond k-Anonymity and ℓ -Diversity, 2007 IEEE 23rd International Conference on Data Engineering, 15–20 April 2007, Istanbul, Turkey.
15. Differential privacy https://en.wikipedia.org/wiki/Differential_privacy.
16. Apple announced that they will be using a technique called “Differential Privacy” (henceforth: DP) to improve the privacy of their data collection practices 2016. <https://blog.cryptographyengineering.com/2016/06/15/what-is-differential-privacy/>.
17. Jun Wang, Shubo Liu, and Yongkai Li. 2016. A review of differential privacy in individual data release. *Int. J. Distrib. Sen. Netw.* 2015, Article 1 (January 2016), 1 pages.
18. Cynthia Dwork. 2006. Differential privacy. In Proceedings of the 33rd international conference on Automata, Languages and Programming - Volume Part II (ICALP'06), Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.), Vol. Part II. Springer-Verlag, Berlin, Heidelberg, 1–12.
19. Microsoft differential privacy for everyone. 2015. http://download.microsoft.com/.../Differential_Privacy_for_Everyone.pdf.
20. V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin and E. Dasseni, 2004, Association rule hiding, in *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 4, pp. 434–447, April 2004.
21. Nabar, Shubha U and Kenthapadi, Krishnaram and Mishra, Nina and Motwani, Rajeev, 2008, A Survey of Query Auditing Techniques for Data Privacy, Privacy-Preserving Data, Springer US, pages: 415—431.
22. Rakesh Agrawal and Ramakrishnan Srikant. 2000. Privacy-preserving data mining. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD '00). ACM, New York, NY, USA, 439–450.
23. Weiping Ge, Wei Wang, Xiaorong Li, and Baile Shi. 2005. A privacy-preserving classification mining algorithm. In Proceedings of the 9th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining (PAKDD'05), Tu Bao Ho, David Cheung, and Huan Liu (Eds.). Springer-Verlag, Berlin, Heidelberg, 256–261.
24. Hadoop Tutorials. 2012. <https://developer.yahoo.com/hadoop/tutorial>.
25. Zhiqiang, Gao & Longjun, Zhang. (2018). Privacy Preserving Data Mining on Big Data Computing Platform: Trends and Future. 491–502.
26. Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. 2010. Airavat: security and privacy for MapReduce. In Proceedings of the 7th USENIX conference on Networked systems design and implementation (NSDI'10). USENIX Association, Berkeley, CA, USA, 20–20.

27. Blass, Erik-Oliver and Di Pietro, Roberto and Molva, Refik and Önen, Melek, 2012, PRISM – Privacy-Preserving Search in MapReduce, Privacy Enhancing Technologies, Springer Berlin Heidelberg, pages:180–200.
28. M. E. Gursoy, A. Inan, M. E. Nergiz and Y. Saygin, Privacy-Preserving Learning Analytics: Challenges and Techniques, in *IEEE Transactions on Learning Technologies*, vol. 10, no. 1, pp. 68–81, Jan.-March 1 2017.
29. Kangsoo Jung, Sehwa Park, and Seog Park. 2014. Hiding a Needle in a Haystack: Privacy Preserving Apriori algorithm in MapReduce Framework. In Proceedings of the First International Workshop on Privacy and Security of Big Data (PSBD '14). ACM, New York, NY, USA, 11–17.
30. Chi Lin, Zihao Song, Houbing Song, Yanhong Zhou, Yi Wang, and Guowei Wu. 2016. Differential Privacy Preserving in Big Data Analytics for Connected Health. *J. Med. Syst.* 40, 4 (April 2016), 1–9.
31. Abouelmehdi, Karim and Beni-Hessane, Abderrahim and Khaloufi, Hayat, 2018, Big health-care data: preserving security and privacy, *Journal of Big Data*, volume 5,number 1, pages 1, 09-Jan 2018.
32. Hill K. How target figured out a teen girl was pregnant before her father did. *Forbes*, Inc. 2012.
33. Jain, Priyank and Gyanchandani, Manasi and Khare, Nilay, 2016, Big data privacy: a technological perspective and review, *Journal of Big Data*, volume 3, number 1, 26-Nov-2016, pages 25.
34. Omar Hasan, Benjamin Habegger, Lionel Brunie, Nadia Bennani, and Ernesto Damiani. 2013. A Discussion of Privacy Challenges in User Profiling with Big Data Techniques: The EEXCESS Use Case. In Proceedings of the 2013 IEEE International Congress on Big Data (BIGDATACONGRESS '13). IEEE Computer Society, Washington, DC, USA, 25–30.
35. J. Sedayao, R. Bhardwaj and N. Gorade, Making Big Data, Privacy, and Anonymization Work Together in the Enterprise: Experiences and Issues, *2014 IEEE International Congress on Big Data*, Anchorage, AK, 2014, pp. 601–607.
36. Xuyun Zhang, Chi Yang, Surya Nepal, Chang Liu, Wanchun Dou, and Jinjun Chen. 2013. A MapReduce Based Approach of Scalable Multidimensional Anonymization for Big Data Privacy Preservation on Cloud. In Proceedings of the 2013 International Conference on Cloud and Green Computing (CGC '13). IEEE Computer Society, Washington, DC, USA, 105–112.
37. Jain, Priyank and Gyanchandani, Manasi and Khare, Nilay, 2018, Differential privacy: its technological prescriptive using big data, *Journal of Big Data*, volume 5, number 1, 13 Apr 2018, pages 15.
38. S. Wang *et al.*, Big Data Privacy in Biomedical Research, in *IEEE Transactions on Big Data*.
39. A. Mehmood, I. Natgunanathan, Y. Xiang, G. Hua and S. Guo, Protection of Big Data Privacy, in *IEEE Access*, vol. 4, pp. 1821–1834, 2016.
40. Matturdi, Bardi & Zhou, Xianwei & Li, Shuai & Lin, Fuhong. (2014). Big Data security and privacy: A review. *China Communications*. January 2014, 11(14), pages: 135–145.
41. C. Perera, R. Ranjan, L. Wang, S. U. Khan and A. Y. Zomaya, Big Data Privacy in the Internet of Things Era, in *IT Professional*, vol. 17, no. 3, pp. 32–39, May–June 2015.

A Bibliometric Analysis of Authentication and Access Control in IoT Devices



Samuel Grooby, Tooska Dargahi, and Ali Dehghantanha

Abstract In order to be considered secure, the devices which make up the Internet of Things (IoT) need access control and authentication methods which are resilient against a wide range of attacks. This paper provides a bibliometric analysis of available academic research papers in this area from 2008 to 2017. We used a dataset of 906 academic papers and analysed the most productive countries, journals, authors and research institutions, as well as looking at the most common research areas, keywords and the most highly cited articles. We then looked at the trends in each country's production finding that overall production is increasing as well as the number of countries contributing. We found that the countries of India, South Korea and USA are rising in their proportional contribution to the dataset whereas the established leader in production, China, is decreasing in dominance. Trends in keyword use showed that the proportion of research relating to Wireless Sensor Networks and RFID technology is decreasing, while the proportion of research into the area of IoT privacy is growing.

Keywords Access Control · IoT · Internet of Things · Authentication

1 Introduction

The “Internet of Things” is an emerging phenomenon in information technology [1]. The term refers to a universal presence of internetworked devices, each with the ability to collect information from, and even perform actions in the real world [2]. The increasing availability of technologies such as high wireless download

S. Grooby · T. Dargahi

Department of Computer Science, School of Computing, Science and Engineering, University of Salford, Manchester, UK

e-mail: S.A.Egunjobi@edu.salford.ac.uk; T.Dargahi@Salford.ac.uk

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

e-mail: ali@cybersciencelab.org

speeds from fifth generation networks [3] and Cloud storage providers [4] means that implementation of this concept is becoming more feasible every year, with the number of IoT devices predicted to number 26 billion by 2020 [5]. To be considered part of the Internet of Things a device must be able to communicate and cooperate with other devices and process data it receives [6]. It must have a physical embodiment (making it a “thing”) and be uniquely identifiable to its peers. Finally, the device must include sensors, actuators or have some other capacity to interact with the physical world [7]. IoT devices have applications in healthcare [8], smart home [9] and critical infrastructure [10]. This high level of integration with our environment means that cyberattacks will become a threat like never before [11]. Hacking groups will have the ability to perform Stuxnet-like attacks, gaining control of a wide range of physical mechanisms that can collect huge amounts of data or do damage in the real world [12]. The massive number of these devices, along with the value their control brings to the attacker and lax security of the current crop means that the IoT will be a frequent target for hackers in the near future [13].

One pillar of the security policy which will prevent such attacks becoming rampant is reliable access control [14]. Each data holding device must be able to recognise whether the source of a request for its data is from who it claims to be (authentication). Once this is established to be true then the device must decide what data the authenticated user is permitted access to (authorisation). Finally, the device must log actions performed by authorised users, so they can be reviewed (accountability). The combination of these three processes is called access control [15]. Unfortunately, the nature of IoT devices makes reliable access control a significant challenge [16]. Since these devices are constantly receiving data from their sensors and each other, they must be able to process huge data streams in real time which requires more computation than normal database access control [17]. Devices must be able to authenticate a large number of heterogeneous systems, all with different amounts of resources available to them [18]. Despite an abundance of resources being more ideal, many IoT devices need to work with lower processing power, memory requirements and energy consumption than conventional systems as well as communicating on lossy networks [19]. To address this issue, researchers are devising new algorithms for key management [20] and authentication [21, 22] in resource constrained systems.

This paper provides a statistical analysis of research into issues and solutions regarding access control in Internet of Things devices, specifically the areas of authentication and authorisation. This analysis is then used to find past trends and predict future ones. Current literature covers either too broad an area or does not provide statistical analysis of its subject, which was our motivation for the current research. In [23] researchers surveyed the available literature on access control in IoT environments, however the paper is a review and applies no bibliographic methodology. In [21] researchers analysed the literature of IoT between the years of 2000 and 2015 using bibliometric techniques and proposes a classification of research themes. The paper covers a very large area and does not focus on research

specifically of IoT security and access control. In [22] scientometric techniques are used to analyse a data set of 19,035 papers on the IoT between 2002 and 2016. This paper also lacks a focus into IoT security, instead reviewing a wider-ranging data set. In [19] researchers present the challenges and solutions in IoT security including access control, however no bibliometric techniques are applied.

This paper attempts to answer the following research questions:

- What trends in research into IoT access control over the last 10 years can be found?
- What possible trends can be predicted in future?

To answer these questions this paper follows the methodology outlined in [24]. By searching for papers containing the key terms “Internet of Things”, “Authentication” and “Access Control” on Web of Science, over 900 papers were found. The dataset was then further filtered to exclude non-English language papers. The analysis of the data set was conducted by extracting metadata from each paper such as country of origin, author, publication journal, publication year, institution, and abstract, and finding relationships between these data points. After the analysis, the paper discusses the classification of access control systems in IoT devices, and then discusses future trends in IoT research.

The structure of this paper is as follows. Section 2 details the research methodology for the collection, pre-processing and analysis of data. Section 3 displays the findings of the analysis. Section 4 categorises the access control systems from the data set. Section 5 concludes the study.

2 Methodology

Bibliometrics refers to the collection and analysis of data relating to academic publications [25]. By examining the metadata of a selection of papers, useful information about a research topic can be acquired [26]. This can include what themes are and aren't commonly investigated, or which authors, countries and institutions produce the highest quality work.

In order to establish a dataset for analysis, the keywords “Internet of Things”, “Authentication” and “Access Control” were chosen. It was decided that all three phrases were needed to find all available papers fitting the scope of this paper. To avoid over-refining the dataset by only including papers which were indexed with all three phrases the following logical expression was developed: (“Internet of Things” AND “Authentication”) OR (“Internet of Things” AND “Access Control”). Using this search term ensured that papers in the dataset could be related to access control or authentication, but all had to be on the topic of the Internet of Things. Searching for this criteria in the Web of Science Core Collection resulted in 945 papers. Our research found that the earliest paper found was published in 2008, exactly 10 years before the writing of this paper. Research from the current year

was excluded since the year is not complete and the data so far would underrepresent the entire year, resulting in 911 papers. The dataset was refined to exclude editorial material, leaving only proceedings papers, journal articles and reviews. Finally, only English language papers were included in the final dataset giving a total of 906 results. These papers were then analysed by looking at the same criteria as in [24]: Country; Research institution; Author; Journal; Research area; Keywords; and the most highly cited articles. By looking at the number of papers in each category we find the productivity and by looking at the number of citations each paper has acquired we can find the influence. Figure 1 shows a flowchart of the methodology. The software Microsoft Excel was used to visualise the analysis.

The database Web of Science was chosen as the source of research for this analysis. Web of Science includes several useful features: A comprehensive search function which allows the user to include or exclude papers based on a logical expression; search results can be sorted or refined by several criteria; the analyse results function automatically creates a table of records in a certain field ordered by count; the citation report function creates a graph of times cited per year for the dataset.

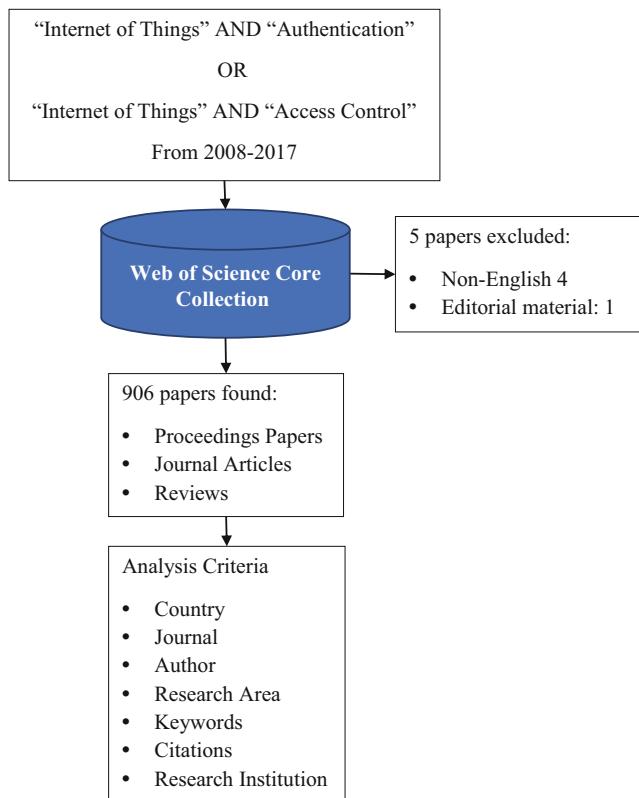


Fig. 1 Flowchart showing the data collection methodology of this paper

3 Findings

In this section we discuss the data we have collected from the Web of Science Database. Web of Science retrieves numerous different data fields from its entries including editors, languages, web of science category, document type, organisation and funding agency. Despite having more avenues for analysis available, our findings are broken down into just seven categories as we believe this will be enough to answer our research questions. The categories are as follows: Country of Origin; Research Area; Research Institution; Author; Journal; Highly Cited Articles and Keywords frequency.

3.1 *Country of Origin*

In this section we discuss the productivity of each country which has made a contribution to the dataset. Table 1 shows the productivity of each country, and their contribution as a percentage of the total dataset. The table also shows the productivity and percentage contribution of each continent. Productivity is defined

Table 1 Productivity of countries

Countries/regions	Publications	Percentage of articles (%)
Africa	34	2.81
Algeria	5	0.41
Libya	1	0.08
Morocco	13	1.08
Nigeria	1	0.08
South Africa	3	0.25
Tunisia	11	0.91
Asia	551	45.61
Bangladesh	3	0.25
India	95	7.86
Indonesia	3	0.25
Iran	9	0.75
Iraq	1	0.08
Japan	32	2.65
Jordan	3	0.25
Kuwait	2	0.17
Lebanon	2	0.17
Malaysia	6	0.50
Pakistan	12	0.99
China	209	17.30
Saudi Arabia	15	1.24
Singapore	14	1.16
South Korea	85	7.04
Taiwan	49	4.06

(continued)

Table 1 (continued)

Countries/regions	Publications	Percentage of articles (%)
Thailand	2	0.17
UAE	7	0.58
Vietnam	2	0.17
Australasia	20	1.66
Australia	20	1.66
Europe	433	35.84
Austria	5	0.41
Belgium	18	1.49
Bulgaria	1	0.08
Belarus	1	0.08
Cyprus	2	0.17
Czech Republic	7	0.58
Denmark	9	0.75
England	40	3.31
Finland	22	1.82
France	50	4.14
Germany	41	3.39
Greece	16	1.32
Hungary	3	0.25
Ireland	7	0.58
Italy	37	3.06
Luxembourg	1	0.08
Netherlands	5	0.41
Norway	7	0.58
Poland	9	0.75
Portugal	11	0.91
Romania	7	0.58
Russia	11	0.91
Scotland	1	0.08
Serbia	1	0.08
Slovenia	8	0.66
Spain	56	4.64
Sweden	26	2.15
Switzerland	23	1.90
Turkey	8	0.66
North America	151	12.50
Canada	27	2.24
Cuba	1	0.08
USA	123	10.18
South America	19	1.57
Brazil	17	1.41
Chile	1	0.08
Colombia	1	0.08

as the number of publications made by a country. It is useful as it shows us where research is coming from; however it is limited as it does not show us the quality or influence of the research. For this other metrics can be used such as number of citations and h-index.

The most productive continent from the data is shown to be Asia with 551 publications making 45.61% of the dataset. The largest contributor from Asia is China, which is the largest contributor from any continent producing 209 publications and 17.3% of the dataset. Other major contributors are India with 95 publications and 7.86% of the dataset and South Korea with 85 publications and 7.04% of the dataset, making these nations the third and fourth most productive nations overall. The remaining 16 Asian countries account for only 29.4% of the continent's total.

The second most productive continent is Europe with 433 publications and 35.84% of the dataset. The major European contributors are Spain, France, Germany, England and Italy which are the fifth, sixth, eighth, ninth and tenth most productive countries in the world respectively. Productivity is more evenly distributed between European countries than Asian countries as Spain, the most productive in Europe, accounts for only 12.93% of the continent's total, whereas China accounts for 37.31% of the Asian total. In addition to this, the top five European countries account for 51.73% of Europe's total whereas the top three Asian countries account for 70.59% of the continent's total.

North America, Africa, Australasia and South America are the third, fourth, fifth and sixth continents respectively. The most productive country from these continents by far is USA with 123 publications and 10.18% of the dataset making it the second most productive country overall. USA is responsible for 81.45% of North American publications.

Table 2 shows the number of citations received by each of the ten most productive countries, as well as the average citations per publication and the countries' h-index. The h-index metric is used to measure the impact of a source of research [27]. It is calculated by ordering research paper by decreasing number of citations and finding the final paper with a number of citations greater than or equal to its position.

Table 2 Citations of most productive countries

Country	Citations	Average citations per publication	H-Index
China	774	3.69	14
USA	446	3.57	12
India	237	2.44	8
South Korea	242	2.85	8
Spain	275	4.91	10
France	54	1.08	5
Taiwan	120	2.45	5
Germany	126	3.07	4
England	130	3.25	6
Italy	1098	29.68	9

The country with the most citations is Italy with 1098, while being only tenth in productivity. Italy has an average of 29.68 citations per publication, with the next highest, Spain, only having 4.91 per publication. Despite having such a huge number of citations Italy only has an h-index of 9, which is fourth on the table behind China, USA and Spain respectively. This suggests that a small number of Italy's publications are responsible for a large number of citations and that removing them would result in a much smaller average. This is shown to be true as the two most cited papers [7, 19] have 739 and 164 citations respectively. Without these papers Italy would have an average of 5.27 citations per publication, while this is still the highest of any country it is a drop of 82.24%.

The country with the highest h-index is China with 14 suggesting it produces a greater volume of influential research than other countries. The sixth most productive country France only has 54 citations, giving it an average of 1.08 per publication which is much lower than the next lowest, India with 2.44 citations per publication. This suggests that despite having a reasonable amount of publications, France has not produced much influential work.

3.2 Research Area

In this section we discuss the different research areas covered by the publications in our dataset. Web of Science attributes each publication with one or more research areas so that researchers can find papers relating to their interests. Table 3 shows the top 25 most common Research Areas attributed to the papers in our dataset.

The most common research area with 653 publications is “Computer Science”, this represents 70.51% of the dataset. This is unsurprising as the dataset concerns the security of IoT devices which is much more likely to pertain to information security than physical security. Information security is encompassed in the field of Computer Science. The second most common research area is “Engineering” with 402 publications, representing 43.41% of the dataset. This can be explained by many of the publications pertaining to the engineering issues with IoT devices. This may concern sensors, actuators, or other physical aspects of the devices. The third most common research area is “Telecommunications” with 356 publications and 38.44% of the dataset. By definition IoT devices need to be able to communicate with other devices at distance, including when authenticating each other, therefore telecommunications are a key area of research for the technology. The fourth most common research area is “Instruments Instrumentation” with 47 papers and 5.07% of the dataset. This area concerns instruments used to measure the physical world, for example sensors which would be used in IoT devices.

Table 3 Research areas

Research areas	Publications	Percentage
Computer Science	653	70.518
Engineering	402	43.413
Telecommunications	356	38.445
Instruments Instrumentation	47	5.076
Chemistry	40	4.32
Electrochemistry	35	3.78
Automation Control Systems	30	3.24
Materials Science	16	1.728
Physics	16	1.728
Mathematics	9	0.972
Medical Informatics	9	0.972
Science Technology Other Topics	9	0.972
Remote Sensing	8	0.864
Transportation	7	0.756
Health Care Sciences Services	6	0.648
Robotics	6	0.648
Education Educational Research	4	0.432
Environmental Sciences Ecology	4	0.432
Operations Research Management Science	4	0.432
Energy Fuels	3	0.324
Optics	3	0.324
Business Economics	2	0.216
Government Law	2	0.216
Mechanics	2	0.216
Social Sciences Other Topics	2	0.216

3.3 Research Institution

This section discusses the research institutions responsible for each publication. From the data we can see which institutions are the most productive. Table 4 shows us the 25 institutions with the highest number of publications, their percentage contribution to the entire dataset, their country and the percentage of the country's total.

The majority of the institutions (13 of the 25) are from Asian countries, with 7 of the 13 being from China, 5 different countries are represented in this set: China, India, South Korea, Saudi Arabia and Singapore. 10 of the 25 are from European countries with 7 different countries represented: Belgium, England, Finland, France, Spain, Sweden and Switzerland. The remaining two institutions, Intel Corporation and University of California System, are from the USA. Most productive research institution is the Chinese Academy of Sciences with 27 publications comprising 2.91% of the dataset and 13% of China's total. The institution on the list which

Table 4 Productivity of institutions

Institution	Publications	Percentage of total	Percentage of country's total (%)	Country
Chinese Academy of Sciences	27	2.916	13	China
Beijing University of Posts Telecommunications	20	2.16	10	China
University of Murcia	18	1.944	32	Spain
Centre National De La Recherche Scientifique CNRS	17	1.836	34	France
Xidian University	15	1.62	7	China
Soongsil University	13	1.404	15	South Korea
Indian Institute of Technology IIT	12	1.296	12	India
University of California System	12	1.296	10	USA
Aalto University	11	1.188	48	Finland
Universite Paris Saclay Comue	11	1.188	22	France
Nanyang Technological University National Institute of Education Nie Singapore	10	1.08	67	Singapore
University of Applied Sciences Arts Western Switzerland	10	1.08	43	Switzerland
Intel Corporation	9	0.972	7	USA
Royal Institute of Technology	9	0.972	35	Sweden
University of Electronic Science Technology of China	9	0.972	4	China
University of Oulu	9	0.972	39	Finland
Wuhan University	9	0.972	4	China
Imec	8	0.864	44	Belgium
Institute of Information Engineering Cas	8	0.864	4	China
Ku Leuven	8	0.864	44	Belgium
Nanjing University of Information Science Technology	8	0.864	4	China
Thapar University	8	0.864	8	India
University of London	8	0.864	20	England
Indian Institute of Technology IIT Kharagpur	7	0.756	7	India
King Saud University	7	0.756	47	Saudi Arabia

is most dominant in its country is the Nanyang Technological University National Institute of Education Nie Singapore which produces 67% of Singapore's entire total with 10 publications.

3.4 Author

In this section we discuss the scholars who have contributed the most research in our dataset. Table 5 shows the ten most productive authors along with the number of publications they have produced, the proportion of the dataset they have contributed and their country of origin. The table shows that the half of the top ten authors are from Asian countries, two being from India and three from China. With 39 publications between them, the top five Asian authors are responsible for 7.07% of the dataset. Of the remaining five authors, four are from Europe and one is from Morocco, Africa. Despite being the second most productive country, USA has no authors in the top ten, suggesting research is more evenly distributed between scholars there.

Table 5 Productivity of authors

Authors	Records	Percentage of total	Country
Antonio J Jara	12	1.296	Spain
Neeraj Kumar	9	0.972	India
Hui Li	9	0.972	China
Yuhong Li	8	0.864	China
Abdellah Ait Ouahman	8	0.864	Morocco
Yue Chen	7	0.756	England
Chandra Sekhar Vorugunti	7	0.756	India
An Braeken	6	0.648	Belgium
Kai Fan	6	0.648	China
Andrei Gurtov	6	0.648	Finland

The most productive author with 12 publications and 1.29% of the dataset is Antonio J Jara from Spain who has 3 more publications than the next highest. Jara's most cited article is titled "Toward a Lightweight Authentication and Authorization Framework for Smart Objects" which proposes a security architecture for IoT devices including lightweight authentication and authorization protocols [28].

In joint second place is Neeraj Kumar of India and Hui Li of China with 9 publications and 0.97% of the dataset each. Table 6 shows us the total and average number of citations each top three author has received as well as their H-Indexes.

Table 6 Citation count of most productive authors

Authors	Publications	Citations	Average	H-Index
Antonio J Jara	12	73	6.08	5
Neeraj Kumar	9	121	13.44	5
Hui Li	9	33	3.67	3

The most cited of the three is Neeraj Kumar with 121 citations and an average of 13.44 citations per publication. Despite producing the same amount of publications as Hui li, Kumar has received almost 4 times as many citations, and has even received 48 more than Antonio Jara who has 3 more publications. Kumar's average citation count is more than twice Jara's although they both have an h-Index of 5, suggesting that Kumar's most cited papers may be skewing the average.

3.5 *Journal*

In this section we look at the most productive journals and book series featured in our dataset. This information is useful for researchers as it tells us which journals are most active in the field represented by our dataset and so such series may be the most worth studying. Table 7 shows us the top ten most productive journals, the number of publications they have made, their percentage contribution to the dataset, the number of citations their books have received, their percentage of the total citations received by the dataset.

Table 7 Productivity of journals

Book series titles	Publications	Percentage of total	Citations	Percentage of total (%)	H-Index
Lecture Notes in Computer Science	31	3.348	42	1.216	3
IEEE International Conference on Communications	13	1.404	18	0.521	1
Procedia Computer Science	11	1.188	27	0.781	3
Lecture Notes of the Institute for Computer Sciences Social Informatics and Telecommunications Engineering	10	1.08	1	0.029	1
IEEE Global Communications Conference	9	0.972	8	0.232	1
Communications in Computer and Information Science	8	0.864	36	1.042	1
IEEE Consumer Communications and Networking Conference	7	0.756	0	0.000	0
Applied Mechanics and Materials	6	0.648	4	0.116	1
International Conference on Computer Systems and Applications	6	0.648	0	0.000	0
Advances in Intelligent Systems and Computing	5	0.54	4	0.116	1

At the top of the table is the conference proceedings series “Lecture Notes in Computer Science” with 31 publications making up 3.34% of our publications. It has 42 citations which also makes it the highest cited series in the top ten. Two of the series in our table have no citations, these are “IEEE Consumer Communications and Networking Conference” and “International Conference on Computer Systems and Applications” in sixth and ninth place respectively. This suggests that despite being productive, not all journals in the table are influential and there may not be a correlation.

3.6 Highly Cited Articles

In this section we will discuss the publications in the dataset with the most citations. Table 8 lists the ten most highly cited papers along with the year they were published, the number of times they were cited, their average number of citations per year and the journal they were published in.

The paper in our dataset with the most citations is “Internet of things: Vision, applications and research challenges” from 2012 with 704 citations averaging 100.57 per year since publication. This paper has 553 more citations than the second highest on the list and averages 62.82 more per year, showing that it is clearly an extremely influential paper. The article is far reaching in its contributions and covers applications of the IoT, challenges to its development and the potential applications of the technology. The broad scope of the article is possibly the reason for its extremely high citation count.

Two papers on the list are from the latest year of our dataset 2017, “Anonymous Authentication for Wireless Body Area Networks with Provable Security” in fourth with 67 citations and “Multimedia cloud transmission and storage system based on internet of things” in tenth with 47 citations. These papers have an average per year of 33.5 and 23.5 respectively making them third and fourth of the top ten in that metric. This suggests that these papers may rise high in this table if their rate of gaining citations increases.

The oldest paper on the list is “An Overview of Privacy and Security Issues in the Internet of Things” from 2010. It is sixth on the list with 64 citations and has the lowest average number of citations per year with 7.11. Having been around for 8 years the paper has managed to accrue significant influence.

Looking at the journals of the papers we can see that none of them were published in a top ten most productive journal as seen in Table 7. This is useful information for scholars as it shows that the most productive journals are not necessarily the most influential in the research topic.

Table 8 Most cited publications

Title	Publication year	Total citations	Average per year	Source title
Internet of things: Vision, applications and research challenges	2012	704	100.57	Ad Hoc Networks
Security, privacy and trust in Internet of Things: The road ahead	2015	151	37.75	Computer Networks
A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks	2013	74	12.33	Journal of Network and Computer Applications
Anonymous Authentication for Wireless Body Area Networks with Provable Security	2017	67	33.5	IEEE Systems Journal
A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion	2014	67	13.4	Ad Hoc Networks
An Overview of Privacy and Security Issues in the Internet of Things	2010	64	7.11	Internet of Things
SVELTE: Real-time intrusion detection in the Internet of Things	2013	60	10	Ad Hoc Networks
Cognitive Machine-to-Machine Communications for Internet-of-Things: A Protocol Stack Perspective	2015	59	14.75	IEEE Internet of Things Journal
DTLS based security and two-way authentication for the Internet of Things	2013	52	8.67	Ad Hoc Networks
Multimedia cloud transmission and storage system based on internet of things	2017	47	23.5	Multimedia Tools and Applications

3.7 *Keywords*

In this section we discuss the keywords most frequently found in our dataset. These totals include both author selected keywords and keywords highlighted by Web of Science's KeyWords Plus feature. These keywords are selected by editors who use the titles of referenced papers to choose useful keywords which may have been missed by the author. This information is useful for researchers as it shows which subtopics within the scope of our dataset are being studied the most and where there may be gaps in the research. The most frequently found keywords in the list were collated with other keywords if they both belonged to a broader area. For

example, the 12th most frequent keyword was “key agreement”, the total count for which included instances of “channel based key agreement”, “authenticated key agreement” and “key agreement scheme”.

It can be seen in Table 9 that the most common keyword in the dataset was “internet of things” with 912 mentions. This is understandable as this string was entered as a topic in the Web of Science search function. When searching for topics Web of Science returns any record which contains the search string in either its Title, Abstract, Author Keywords or Keywords Plus field, therefore it is likely that “internet of things” can be found in the Author Keywords or Keywords Plus of a paper in the dataset.

Table 9 Most common keywords

Keyword	Occurrences
Internet of things	912
Security	288
Authentication	242
Wireless sensor networks	177
Access control	138
Cryptography	97
Energy	95
Privacy	57
Cloud computing	48
RFID	48

The second most frequent keyword is “security” with 288 mentions. This is unsurprising as the two search terms access control and authentication are both concepts which are vital to security. Common keywords which were umbrellaed under “security” were: “network security”, “information security”, “security protocol”, “iot security”, “data security” and “RFID security”. Information, data and network security are all relevant to IoT devices as one of their major feature is to communicate data and information to each other over a network, which needs to be done securely. This will be done by the implementation of security protocols such as key agreement, authentication and encryption. “RFID security” is a common keyword as RFID is a precursor technology in the IoT and could be considered as an enabling technology for IoT.

The search terms “authentication” and “access control” appear on the list at third with 242 uses and fifth with 138 uses respectively. Both of these keywords clearly made the list for the same reasons as “internet of things”. Common keywords which contain “authentication” are: “mutual authentication”, “device authentication”, “group authentication” and “broadcast authentication”. Mutual authentication regards two parties authenticating each other at the same time, and is commonly used in machine to machine communication [29] which is integral to IoT devices. “Device authentication” usually refers to methods of device to device authentication, similar to machine to machine authentication, again relevant to IoT devices [30]. Group authentication is a type of authentication where devices are

put into groups and then authenticated as such as opposed to a one to one protocol of multiple devices which would take up greater time and resources [31]. This is useful for IoT devices which are likely to be numerous and have strict resource constraints. Broadcast authentication refers to methods of proving the identity of the originator of a broadcast message is genuine [32]. IoT devices are likely to use broadcasting due to the high diversity of technologies in other devices, therefore a secure broadcast authentication protocol is needed [33].

Common keywords containing “access control” include: “medium access control”(MAC) and “attribute based access control”(ABAC). MAC is part of the datalink layer of the OSI model and provides the first abstraction of the physical hardware of a networking machine. A MAC protocol is used in a Wireless Sensor Network to regulate sensor access to the shared communication channel [34]. ABAC concerns granting access to users based on attributes of the subject and environment [35]. Such schemes could be useful to IoT devices who may want to automatically grant access to other devices of the same type.

In fourth place with 177 occurrences is “wireless sensor networks” (WSN), this along with joint ninth placed “RFID” is a technology valuable to the creation of IoT networks. Sensor nodes collect environmental data and send it through a shared communication channel [36], however they do not receive or process data as an IoT device would. RFID or Radio Frequency Identification is a technology in which lightweight tags are embedded in devices which can be identified at a distance of up to 25 m [37]. An integration of both technologies together may provide solutions to some current issues in IoT networks [38].

In sixth place with 97 occurrences is “cryptography”. Common keywords including cryptography include “elliptic curve cryptography” (ECC) and “lightweight cryptography”. Lightweight crypto algorithms are important for IoT devices which function with limited resources. Encryption can be a particularly resource intensive process [39] and IoT devices must be able to send data which is only readable to the intended recipients in order to be considered secure. ECC is a form of lightweight public key cryptography which promises smaller key sizes [40], allowing its use on devices with less storage space.

The keyword “energy” comes in at seventh place with 95 occurrences. Terms found which include the word energy are: “energy efficiency”, “energy harvesting”, “energy consumption” and “bluetooth low energy”. Efficiency and consumption refer to the usage of energy by IoT devices which is an important consideration as such devices may need to function without external maintenance for long periods of time. Energy harvesting refers to the capture of environmental energy by devices for prolonged functioning, either by solar power, thermal or kinetic energy. “bluetooth low energy” is a technology which allows wireless communication on extremely low power devices [41]. This and similar technologies are crucial to IoT devices where energy efficiency is a concern.

At eighth place comes privacy with 57 occurrences. Keywords relevant to privacy include: “privacy protection”, “privacy preservation”, “data privacy” and “location privacy”. These terms all concern the significant threat to privacy that the IoT imposes. In joint ninth place comes “cloud computing” with 48 occurrences. Cloud

computing provides the required computational and storage resources for the IoT application. Several papers have been written on the IoT applications that considered integration of cloud computing [42]. These include Medical Monitoring Systems [43], independent care systems for people with disabilities [44] and intelligent transport systems [45].

4 Research Trends

In this section we discuss the trends we have found in IoT access control research. In order to do this, we analyse the changes over time in productivity of countries, authors and occurrences of keywords in the publications from our dataset.

4.1 Productivity

There has been a general increase in the production of research into IoT authentication and Access control over the previous 10 years. It can be seen in Fig. 2 that since 2009 the number of publications released has increased every year until 2017, this may be explained by the Web of Science database not having uploaded records for every paper from 2017 at the time of data collection. The year upon year increase in productivity may be explained by an increase in the prevalence of IoT devices on the market [5, 46]. As technology in key areas such as RFID [7, 37, 47] and WSN [36, 44, 48] advances, more devices are available to use and the concept of IoT grows in the public eye, encouraging researchers to take an interest. An increase in diversity of platforms also leads to more security issues [18, 48], which in turn leads to more research being done into possible solutions. Each yearly increase in publications also encourages scholars to build upon existing research as the pool of collective knowledge and resources for researchers has grown.

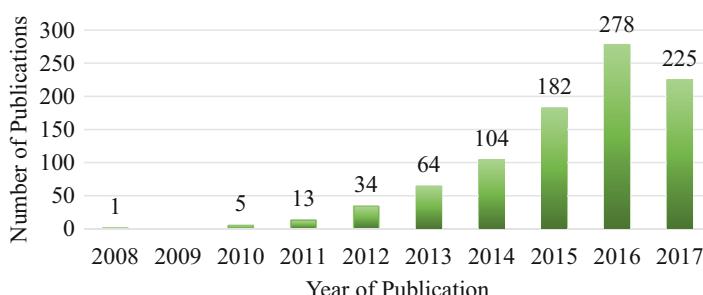


Fig. 2 Number of publications relating to authentication and access control in IoT devices in the previous 10 years

It can also be seen in our data that more countries are taking part each year. Figure 3 illustrates the number of different countries who have published a paper in our dataset each year. Since 2011 the number has increased at a similar rate to the number of publications per year, again with a dip in 2017 where there were fewer publications found on Web of Science than the previous year. Based on our data it could be predicted that the number of publications, and the number of countries publishing them will both continue to increase in the near future.

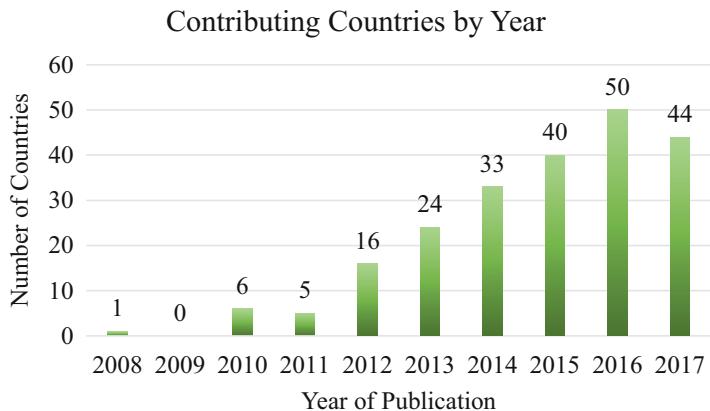


Fig. 3 Number of countries contributing to the dataset by year

Table 10 shows the proportion of each year's publications each country has contributed to the dataset. In the earlier years (2008–2010) very few papers were being published until 2011 where China produced 11 papers, almost twice as many as were published worldwide in the previous 3 years, making it responsible for 84.62% of the years' publications. Here China established itself as a leader in production of IoT access control research and although its proportional yearly contribution has decreased, in real terms its productivity has only increased, and it has the highest contribution to the dataset overall.

In the past 6 years India, South Korea and USA have all increased research production in real terms and also in proportion of the total. This is illustrated in Fig. 4. Due to the rapid yearly increase in production worldwide, Italy is another country with a proportional decrease yet a real increase, Europe's leading producer Spain has remained steady in real terms however has decreased in proportional contribution. From our data it could be predicted that Europe's productivity will likely increase in the near future however the rate of increase will not be enough to

Table 10 Country's contribution to dataset by year

Countries/regions	2008 (%)	2009 (%)	2010 (%)	2011 (%)	2012 (%)	2013 (%)	2014 (%)	2015 (%)	2016 (%)	2017 (%)
Algeria	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.65	0.36
Australia	0.00	0.00	0.00	0.00	2.94	3.13	1.92	0.00	2.52	3.56
Austria	0.00	0.00	0.00	0.00	0.00	0.96	1.10	0.72	0.00	0.00
Bangladesh	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.10	0.00	0.44
Belgium	0.00	0.00	20.00	0.00	2.94	0.00	0.00	4.40	1.08	2.22
Brazil	0.00	0.00	0.00	0.00	0.00	1.56	0.96	3.30	2.16	1.33
Bulgaria	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.36	0.00
Belarus	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.89
Canada	0.00	0.00	0.00	0.00	5.88	1.56	3.85	1.65	2.16	4.89
Chile	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00
Colombia	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44
Cuba	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44
Cyprus	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.36	0.44
Czech Republic	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.80	0.89
Denmark	0.00	0.00	20.00	0.00	2.94	4.69	0.96	0.55	0.72	0.00
England	0.00	0.00	0.00	0.00	2.94	1.56	6.73	3.85	5.04	4.44
Finland	0.00	0.00	0.00	0.00	0.00	3.13	3.85	3.85	2.16	1.78
France	0.00	0.00	0.00	0.00	0.00	1.56	5.77	7.14	7.19	4.44
Germany	0.00	0.00	0.00	0.00	5.88	4.69	7.69	7.69	2.88	2.67
Greece	0.00	0.00	0.00	0.00	2.94	1.56	2.88	0.00	2.88	1.33
Hungary	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.55	0.72	0.00
India	0.00	0.00	0.00	0.00	2.94	6.25	6.73	8.24	14.39	13.33
Indonesia	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.72	0.44
Iran	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.10	1.44	1.33
Iraq	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.36	0.00

(continued)

Table 10 (continued)

Countries/regions	2008 (%)	2009 (%)	2010 (%)	2011 (%)	2012 (%)	2013 (%)	2014 (%)	2015 (%)	2016 (%)	2017 (%)
Ireland	0.00	0.00	0.00	0.00	0.00	0.00	0.96	1.10	1.44	0.00
Italy	0.00	0.00	40.00	0.00	8.82	3.13	3.85	5.49	2.88	3.56
Japan	0.00	0.00	0.00	7.69	5.88	3.13	3.85	0.55	3.60	5.33
Jordan	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.72	0.44	
Kuwait	0.00	0.00	0.00	0.00	0.00	1.56	0.00	0.00	0.36	0.00
Lebanon	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.36	0.44
Libya	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44
Luxembourg	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.55	0.00	0.00
Malaysia	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.55	0.00	2.22
Morocco	0.00	0.00	0.00	0.00	0.00	0.00	1.92	3.30	1.08	0.89
Netherlands	0.00	0.00	0.00	0.00	2.94	1.56	0.00	1.10	0.36	0.00
Nigeria	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44
Norway	0.00	0.00	20.00	0.00	2.94	0.00	1.92	0.00	0.72	0.44
Pakistan	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.55	1.08	3.56
China	0.00	0.00	20.00	84.62	50.00	43.75	21.15	14.84	14.75	28.00
Poland	0.00	0.00	0.00	0.00	0.00	0.00	0.96	3.30	0.72	0.00
Portugal	0.00	0.00	0.00	0.00	0.00	0.00	0.96	1.65	1.80	0.89
Romania	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.55	1.08	0.89
Russia	0.00	0.00	0.00	0.00	0.00	3.13	0.00	0.55	2.16	1.33
Saudi Arabia	0.00	0.00	0.00	0.00	4.69	0.96	0.55	1.08	3.11	

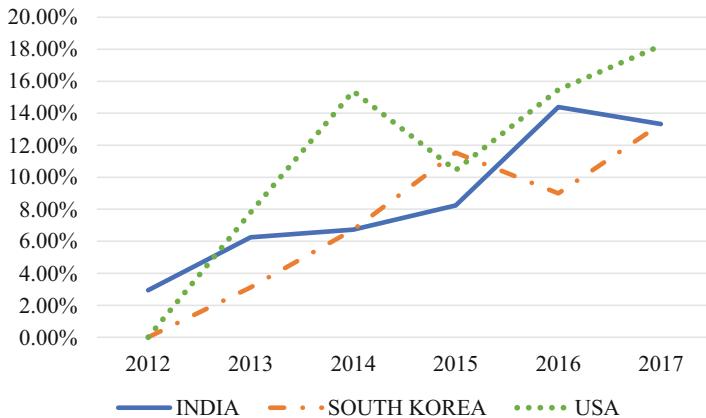


Fig. 4 Percentage contribution of India, South Korea and USA since 2012

stop a proportional decrease due to the high increase in production of countries such as India, South Korea and USA which will contribute the proportional increase in their continents.

4.2 Keywords

By looking at trends in the occurrences of keywords in the research in our dataset we can make predictions about the themes in research which will be conducted in the next few years. We looked at the top ten most occurring keywords from Sect. 3.7 and removed “Internet of Things”, “Security”, “Authentication” and “Access Control” as these are very general and could apply to almost any publication from our dataset. Figure 5 shows the percentage of publications made from 2012 to 2017 which contain the following six most occurring keywords: “Wireless Sensor Networks”, “Cryptography”, “Energy”, “Privacy”, “Cloud” and “RFID”. We chose to refine our data for this section to only include keywords from publications from 2012 onwards as an insufficient volume of publications are available from the previous years of the dataset.

The keyword “Energy”, which primarily refers to energy efficiency or capture by IoT devices has remained relatively steady, fluctuating between 2.94 and 7.81%, suggesting that it continues to be an important research challenge for scholars. After having no occurrences in 2012, the keyword “Cryptography” is found in 14.06% of publications in 2013, however since then it has steadily decreased to 7.56% over time. Originally starting at 8.82% in 2012 the keyword “Cloud” decreased the next year to 1.56%, it then steadily increases to 10.67% making it the third most common keyword in 2017.

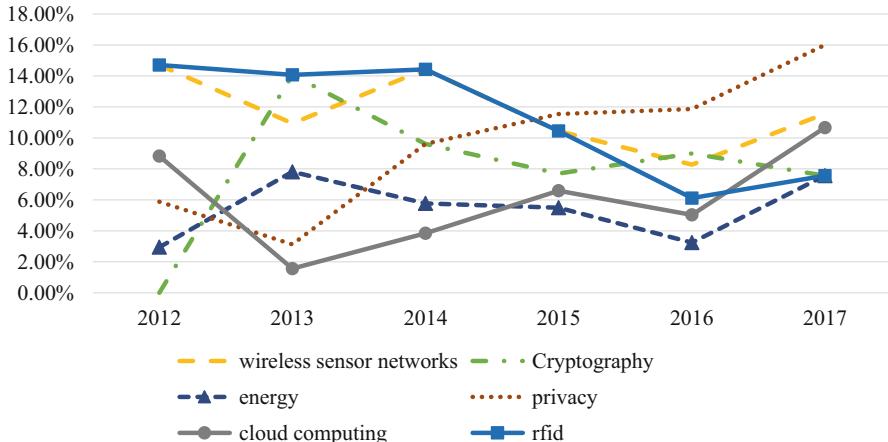


Fig. 5 Percentage of papers containing six most occurring keywords in our dataset

“Wireless Sensor Networks” and “RFID” are both key technologies in the IoT and in 2012 these keywords were the joint most common, occurring in 14.71% of publications each. Since then the proportion of papers containing these keywords has decreased, ending on 11.56% for “Wireless Sensor Networks”, however it is still the second most common keyword in 2017 only behind “Privacy”. The decrease for “RFID” has been more significant with a proportion of 7.56% in 2017, making it joint fourth with “Energy” and “Cryptography”. This may represent a change in the significance of these technologies for the IoT or simply different technologies may be emerging which reduces the share of research for RFID. Electronic Product Code, Near Field Communication and Unique ID are alternative technologies which can be used for the identification of IoT devices [49].

Starting at 5.88% in 2012 the keyword “Privacy” dipped slightly to 3.13% in 2013 before increasing significantly over the next 4 years to 16%, making it the most common keyword of the six chosen in 2017. Privacy is evidently a growing concern with the IoT due to the ubiquitous nature of the technology. Existing privacy issues will only become more severe with the huge increase in applications (e.g., smart home, smart city, smart healthcare, smart airport, smart grid) that the IoT brings [50]. Potentially it will continue to grow as the number of IoT devices increase and it pervades into the lives of the public, bringing attention to the question of what happens to our data.

5 Conclusion and Future Works

In this paper we aimed to identify past trends in research into authentication and access control in the Internet of Things and to use this to predict future ones. In order to do this, we performed a statistical analysis of the metadata of research papers available to us on the Web of Science database. Our findings are as follows:

The most productive continent was found to be Asia, which produced almost half of our dataset, followed by Europe and then North America. Asia has several of the most productive countries, including the most productive country overall, China. Of the top 25 most productive research institutions, seven were from China including the top 2. Three of the top countries, India, South Korea and USA were found to have an increasing proportion of the global production, suggesting they may be world leaders in research into authentication and access control in the Internet of Things in the near future. Europe has the most countries which have contributed to the dataset, however its most productive country, Spain was only fifth overall. North America only has three contributing countries and is dominated by USA which is the second most productive country overall, despite this it only has two of the top 25 most productive research institutions. This suggests that North America may rise in standing as more countries begin to conduct research in future.

The most common research area found in our dataset was Computer Science followed by Engineering and Telecommunications. The most productive author from our dataset was Antonio J Jara of Spain whose research includes a security architecture for IoT devices including lightweight authentication and authorization protocols. The most productive Journal was found to be “Lecture Notes in Computer Science”, a conference proceedings series. The most highly cited article is “Internet of things: Vision, applications and research challenges”, published in Ad Hoc Networks. It is highly influential with many more citations than the next highest and covers applications of the IoT, challenges to its development and the potential applications of the technology.

We collated the most commonly used keywords by authors, finding many papers which relate to Wireless Sensor Networks (WSNs), cryptography, energy efficiency, privacy, cloud computing and RFID technology. We found that the proportion of papers related to WSNs and RFID to be decreasing, possibly due to new alternative technologies, or the fact that research in the area of WSN is mature. The proportion of papers relating to privacy to be increasing as this issue becomes more apparent to the research community.

Finally, we found that overall the amount of research into authentication and access control in the Internet of Things is increasing every year as well as the number of countries who are conducting the research, a trend which we predict to continue as the rate of yearly increase grows.

For future works, it is suggested to conduct a bibliometric analysis of static and dynamic malware analysis techniques with a focus on the specific used tools and investigation approach instead of published papers only such as what suggested in [51]. It would be interesting to find if there is a difference between trend of literature in OSX malware forensics [52] in compare with Android [53] and other types of malware. Moreover, it is interesting to analyse literature trend in security and forensics of specific types of cloud systems such as P2P [54] vs. Cooperative cloud storage [55]. Bibliometric analysis of different types of attack techniques i.e. DDoS [56], Botnets [57], and Ransomware [58] and compare and contrast them would interesting future works as well.

References

1. G. Epiphanou, P. Karadimas, D. K. Ben Ismail, H. Al-Khateeb, A. Dehghantanha, and K.-K. R. Choo, "Non-Reciprocity Compensation Combined with Turbo Codes for Secret Key Generation in Vehicular Ad Hoc Social IoT Networks," *IEEE Internet Things J.*, pp. 1–1, 2017.
2. M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things security and forensics: Challenges and opportunities," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan. 2018.
3. S. Li, L. Da Xu, and S. Zhao, "5G Internet of Things: A Survey," *J. Ind. Inf. Integr.*, p., 2018.
4. Y.-Y. Teing, D. Ali, K. Choo, M. T. Abdullah, and Z. Muda, "Greening Cloud-Enabled Big Data Storage Forensics: Syncany as a Case Study," *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2017.
5. I. Yaqoob *et al.*, "The rise of ransomware and emerging security challenges in the Internet of Things," *Comput. Networks*, vol. 129, pp. 444–458, 2017.
6. A. Azmoekeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *J. Ambient Intell. Humaniz. Comput.*, pp. 1–12, Aug. 2017.
7. D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.
8. S. Walker-Roberts, M. Hammoudeh, and A. Dehghantanha, "A Systematic Review of the Availability and Efficacy of Countermeasures to Internal Threats in Healthcare Critical Infrastructure," *IEEE Access*, 2018.
9. M. Alaa, A. A. Zaidan, B. B. Zaidan, M. Talal, and M. L. M. Kiah, "A review of smart home applications based on Internet of Things," *J. Netw. Comput. Appl.*, vol. 97, pp. 48–65, 2017.
10. A. Azmoekeh, A. Dehghantanha, and K.-K. R. Choo, "Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning," *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2018.
11. S. Watson and A. Dehghantanha, "Digital forensics: the missing piece of the Internet of Things promise," *Comput. Fraud Secur.*, vol. 2016, no. 6, pp. 5–8, Jun. 2016.
12. H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A Deep Recurrent Neural Network Based Approach for Internet of Things Malware Threat Hunting", Future Generation Computer System," *Futur. Gener. Comput. Syst.*, 2017.
13. M. Conti, T. Dargahi, and A. Dehghantanha, "Cyber Threat Intelligence: Challenges and Opportunities," Springer, Cham, 2018, pp. 1–6.
14. M. Hopkins and A. Dehghantanha, "Exploit Kits: The production line of the Cybercrime economy?", in *2015 2nd International Conference on Information Security and Cyber Forensics, InfoSec 2015*, 2016.
15. V. Suhendra, "A Survey on Access Control Deployment," in *Security Technology*, 2011, pp. 11–20.
16. D. Kiwia, A. Dehghantanha, K.-K. R. Choo, and J. Slaughter, "A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence," *J. Comput. Sci.*, Nov. 2017.
17. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence," *IEEE Trans. Emerg. Top. Comput.*, pp. 1–1, 2017.
18. R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Comput. Networks*, vol. 57, no. 10, pp. 2266–2279, 2013.
19. S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Networks*, vol. 76, pp. 146–164, 2015.
20. M. R. Abdmeziem and D. Tandjaoui, "An end-to-end secure key management protocol for e-health applications," *Comput. Electr. Eng.*, vol. 44, pp. 184–197, 2015.

21. D. Mishra, A. Gunasekaran, S. J. Childe, T. Papadopoulos, R. Dubey, and S. Wamba, “Vision, applications and future challenges of Internet of Things: A bibliometric study of the recent literature,” *Ind. Manag. Data Syst.*, vol. 116, no. 7, pp. 1331–1355, 2016.
22. J. Ruiz-Rosero, G. Ramirez-Gonzalez, J. M. Williams, H. Liu, R. Khanna, and G. Pisharody, “Internet of Things: A Scientometric Review,” *Symmetry (Basel.)*, vol. 9, no. 12, 2017.
23. A. Ouaddah, H. Mousannif, A. A. Elkalam, and A. A. Ouahman, “Access control in the Internet of Things: Big challenges and new opportunities,” *Comput. Networks*, vol. 112, pp. 237–262, 2017.
24. M. F. A. Razak, N. B. Anuar, R. Salleh, and A. Firdaus, “The rise of ‘malware’: Bibliometric analysis of malware study,” *J. Netw. Comput. Appl.*, vol. 75, pp. 58–76, 2016.
25. J. Baldwin, O. M. K. Alhwai, S. Shaughnessy, A. Akinbi, and A. Dehghantanha, *Emerging from the cloud: A bibliometric analysis of cloud forensics studies*, vol. 70. 2018.
26. J. Gill, I. Okere, H. HaddadPajouh, and A. Dehghantanha, *Mobile forensics: A bibliometric analysis*, vol. 70. 2018.
27. M. A. Meyers and H. Quan, “The use of the h-index to evaluate and rank academic departments,” *J. Mater. Res. Technol.*, vol. 6, no. 4, pp. 304–311, Oct. 2017.
28. J. L. Hernández-Ramos, M. P. Pawłowski, A. J. Jara, A. F. Skarmeta, and L. Ladid, “Toward a Lightweight Authentication and Authorization Framework for Smart Objects,” *IEEE J. Sel. Areas Commun.*, vol. 33, no. 4, pp. 690–702, 2015.
29. G. Tuna, D. G. Kogias, V. C. Gungor, C. Gezer, E. Taşkin, and E. Ayday, “A survey on information security threats and solutions for Machine to Machine (M2M) communications,” *J. Parallel Distrib. Comput.*, vol. 109, pp. 142–154, 2017.
30. R. Mayrhofer and H. Gellersen, “Spontaneous mobile device authentication based on sensor data,” *Inf. Secur. Tech. Rep.*, vol. 13, no. 3, pp. 136–150, 2008.
31. C. Lai, H. Li, R. Lu, and X. (Sherman) Shen, “SE-AKA: A secure and efficient group authentication and key agreement protocol for LTE networks,” *Comput. Networks*, vol. 57, no. 17, pp. 3492–3510, 2013.
32. K. Grover and A. Lim, “A survey of broadcast authentication schemes for wireless networks,” *Ad Hoc Networks*, vol. 24, pp. 288–316, 2015.
33. N. Ruan and Y. Hori, “DoS attack-tolerant TESLA-based broadcast authentication protocol in Internet of Things,” in *2012 International Conference on Selected Topics in Mobile and Wireless Networking*, 2012, pp. 60–65.
34. S. Kosunalp, Y. Chu, P. D. Mitchell, D. Grace, and T. Clarke, “Use of Q-learning approaches for practical medium access control in wireless sensor networks,” *Eng. Appl. Artif. Intell.*, vol. 55, pp. 146–154, 2016.
35. H. S. G. Pussewalage and V. A. Oleshchuk, “Attribute based access control scheme with controlled access delegation for collaborative E-health environments,” *J. Inf. Secur. Appl.*, vol. 37, pp. 50–64, 2017.
36. N. Khalil, M. R. Abid, D. Benhaddou, and M. Gerndt, “Wireless sensors networks for Internet of Things,” in *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2014, pp. 1–6.
37. Y. Duroc and S. Tedjini, “RFID: A key technology for Humanity,” *Comptes Rendus Phys.*, vol. 19, no. 1, pp. 64–71, 2018.
38. J. V. V. Sobral *et al.*, “A framework for enhancing the performance of Internet of Things applications based on RFID and WSNs,” *J. Netw. Comput. Appl.*, vol. 107, pp. 56–68, 2018.
39. B. J. Mohd, T. Hayajneh, K. M. A. Yousef, Z. A. Khalaf, and M. Z. A. Bhuiyan, “Hardware design and modeling of lightweight block ciphers for secure communications,” *Futur. Gener. Comput. Syst.*, vol. 83, pp. 510–521, 2018.
40. K. Mahmood, S. A. Chaudhry, H. Naqvi, S. Kumari, X. Li, and A. K. Sangaiah, “An elliptic curve cryptography based lightweight authentication scheme for smart grid communication,” *Futur. Gener. Comput. Syst.*, vol. 81, pp. 557–565, 2018.
41. M. Gentili, R. Sannino, and M. Petracca, “BlueVoice: Voice communications over Bluetooth Low Energy in the Internet of Things scenario,” *Comput. Commun.*, vol. 89–90, pp. 51–59, 2016.

42. K. B. Raja, R. Raghavendra, M. Stokkenes, and C. Busch, "Multi-modal authentication system for smartphones using face, iris and periocular," in *2015 International Conference on Biometrics (ICB)*, 2015, pp. 143–150.
43. Y. Liu, B. Dong, B. Guo, J. Yang, and W. Peng, "Combination of Cloud Computing and Internet of Things (IOT) in Medical Monitoring Systems," *Int. J. Hybrid Inf. Technol.*, vol. 8, no. 12, pp. 367–376, 2015.
44. C. Dores, L. P. Reis, and N. V Lopes, "Internet of things and cloud computing," in *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, 2014, pp. 1–4.
45. J. A. Guerrero-ibanez, S. Zeadally, and J. Contreras-Castillo, "Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and internet of things technologies," *IEEE Wirel. Commun.*, vol. 22, no. 6, pp. 122–128, 2015.
46. F. Lauria, "How to footprint, report and remotely secure compromised IoT devices," *Netw. Secur.*, no. December 2017, pp. 10–16, Dec. 2017.
47. G. Roussos and V. Kostakos, "rfid in pervasive computing: State-of-the-art and outlook," *Pervasive Mob. Comput.*, vol. 5, no. 1, pp. 110–131, 2009.
48. M. Turkanović, B. Brumen, and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion," *Ad Hoc Networks*, vol. 20, pp. 96–112, 2014.
49. P. P. Ray, "A survey on Internet of Things architectures," *J. King Saud Univ. - Comput. Inf. Sci.*, 2016.
50. J. Lopez, R. Rios, F. Bao, and G. Wang, "Evolving privacy: From sensors to the Internet of Things," *Futur. Gener. Comput. Syst.*, vol. 75, pp. 46–57, 2017.
51. A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, *Machine learning aided static malware analysis: A survey and tutorial*, vol. 70. 2018.
52. H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "Intelligent OS X malware threat detection with code inspection," *J. Comput. Virol. Hacking Tech.*, 2017.
53. N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided Android malware classification," *Comput. Electr. Eng.*
54. Y.-Y. Teing, A. Dehghantanha, K. K. R. Choo, and L. T. Yang, "Forensic investigation of P2P cloud storage services and backbone for IoT networks: BitTorrent Sync as a case study," *Comput. Electr. Eng.*, 2016.
55. Y.-Y. Teing, A. Dehghantanha, K.-K. R. Choo, T. Dargahi, and M. Conti, "Forensic Investigation of Cooperative Storage Cloud Service: Symform as a Case Study," *J. Forensic Sci.*, vol. 62, no. 3, pp. 641–654, May 2017.
56. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing," *Eurasip J. Wirel. Commun. Netw.*, vol. 2016, no. 1, 2016.
57. S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha, and R. Khayami, *BoTShark: A deep learning approach for botnet traffic detection*, vol. 70. 2018.
58. A. D. James Baldwin, Omar Alhawi, *Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection*. Cyber Threat Intelligence- Springer Book, 2017.

Towards Indeterminacy-Tolerant Access Control in IoT



Mohammad Heydari, Alexios Mylonas, Vasileios Katos, and Dimitris Gritzalis

Abstract The ultimate goal of any access control system is to assign precisely the necessary level of access (*i.e.*, no more and no less) to each subject. Meeting this goal is challenging in an environment that is inherently scalable, heterogeneous and dynamic as the Internet of Things (IoT). This holds true as the volume, velocity and variety of data produced by wireless sensors, RFID tags and other enabling technologies in IoT introduce new challenges for data access. Traditional access control methods that rely on static, pre-defined access policies do not offer flexibility in dealing with the new challenges of the dynamic environment of IoT, which has been extensively studied in the relevant literature. This work, defines and studies the indeterminacy challenge for access control in the context of IoT, which to the best of our knowledge has not been studied in the relevant literature. The current access control models, even those that introduce some form of resiliency into the access decision process, cannot make a correct access decision in unpredicted scenarios, which are typically found in IoT due to its inherent characteristics that amplify indeterminacy. Therefore, this work stresses the need for a scalable, heterogeneous, and dynamic access control model that is able cope with indeterminate data access scenarios. To this end, this work proposes a conceptual framework for indeterminacy-tolerant access control in IoT.

Keywords IoT · Internet of things · Access control

M. Heydari · A. Mylonas · V. Katos

Department of Computing and Informatics, Bournemouth University, Poole, UK
e-mail: mheydari@bournemouth.ac.uk; amylonas@bournemouth.ac.uk;
vkatos@bournemouth.ac.uk

D. Gritzalis (✉)

Department of Informatics, Athens University of Economics and Business, Athens, Greece
e-mail: dgrit@aueb.gr

1 Introduction

Internet of Things (IoT), which is defined as the “worldwide network of interconnected objects” [1], extends connectivity from human-to-machine to machine-to-machine communication. IoT offers large scale of integration of heterogeneous networks and devices, which apart from the obvious opportunities, introduces great security and privacy challenges. These challenges are not new as they have been well-studied in the relevant literature in different IoT domains (such as e-Health, Smart City, Smart Grid) [2–6]. Among the various security challenges in IoT, access control is a crucial and open challenge [7].

Amongst the inherent characteristics of IoT, we consider that *scalability*, *heterogeneity*, *interoperability*, *dynamism* and *resource sharing* exaggerate the security challenges that are related to the field of access control. This holds true as:

Scalability stems from the exponential growth in IoT which also results to an increased network connectivity requirement. According to Gartner, the number of Internet-connected devices will reach 20–50 billion devices by 2020 [8]. As a result, this exacerbates the security challenges in IoT by requiring more effort and resources required by the security controls (such as the access control mechanism) to address them [9].

Heterogeneity and *interoperability* in IoT derive from the different technologies and networks (such as RFID, WSN, GSM) that exist in IoT. Thus, enabling seamless and secure integration of these different platforms is a challenge, as the degree of complexity increases dramatically when different technologies are merged to form a complex network. Similarly, interoperability brings new challenges to the field of data access control [10]. For example, in Vehicle-To-Vehicle (V2V) communication moving from one geographical domain (e.g. UK) to another (e.g. France) can cause data access problem, due to the interoperability issues between inter-domain public key infrastructures [11].

Dynamism in IoT stems from the fact that the interconnected things need to interact with each other in a real-time manner. Therefore, the need for an appropriate response to rapid changes in the physical world which are caused by those interactions poses new technological challenges not only for access control but also for any context-aware services [12].

Resource sharing in different IoT domains (e.g. smart grids, smart city) is inevitable [13, 14]. For example, resource sharing in grid environments through Virtual Organizations (VOs) creates added value by improving performance with less investment. Besides advantages it poses permission misuse and insider threats against VOs [13]. The same threats can be exposed to smart city where traffic information is shared or in V2V communication where two or more vehicles share their resources (e.g. information about their locations) [14].

Data communication loss due to the network or device failure in IoT enabling technologies like Wireless Sensor Network (WSN), and Radio-Frequency Identification (RFID) leads to incomplete and imprecise data. Incompleteness and imprecision inherent in the above-mentioned sources can hinder precise access control decisions [15].

The focus of this work is to introduce “*Indeterminacy*”, a new and neglected challenge in IoT that affects access control. Indeterminacy in IoT comes to play where an access control decision needs to be made based on incomplete or imprecise information. In other words, indeterminacy is introduced in the access control mechanism when the required information for access decision is not sufficient or not precise enough to be used in order to make a correct access decision.

We regard that a subset of the abovementioned inherent IoT characteristics exaggerate the indeterminacy in access control. In specific, *dynamism* may result in indeterminacy because real-time tracking of the rapid changes (joining, disjoining, displacement of entities) is not easily achievable in a scalable environment like IoT. Therefore, the lack of information caused by the inability of tracking those changes results in indeterminacy. Scalability can increase dynamism in a way that having complete information to make access decision is impossible. Network and Service dependency in a *heterogeneous* environment like IoT can cause delay and latency in network delivery. If a real-time access decision depends on the information, which is delivered with delay or suffers from latency, it will suffer from indeterminacy in access control. Finally, regarding the inherent heterogeneity of IoT introduces different sources of data communication loss. For example, data may be lost in RFID due to the following reasons [16]: (1) Missing reading caused by tag collision, metal/liquid effect. (2) Data inconsistency caused by reading data from various readers simultaneously. (3) Ghost data caused by frequency reflection in the reading area. Incompleteness and imprecision inherent in the above-mentioned sources are conceived as the main causes of indeterminacy [15].

In this context, this work contributes by: studying the adaptability of traditional and emerging access control models in IoT, defining indeterminacy, uncertainty and ambiguity in access control for IoT, analyzing resilient access control paradigms and proposing a conceptual framework to address indeterminacy in access control in the context of IoT.

The rest of the chapter is organized as follows: Sect. 2, discusses the preliminaries in the field of access control. Section 3 analyses the concept of uncertainty and ambiguity. This section also defines these concepts in the context of access control and discusses different paradigms in the field of resilient access control. Finally, the conceptual framework is proposed at the end of this chapter to cover both uncertainty and ambiguity in the access control. The chapter concludes in Sect. 4.

2 Background

2.1 An Introduction to Access Control

Access control is a mechanism by which system resources can be used only by authorized entities based on a policy (RFC 4949 Internet Security Glossary [17]). An entity may be a user, program or process. Access control consists of the following functions [18]:

1. *Authentication*, which is defined as a verification process to check whether the credentials of an entity is valid. In some texts, *identification* was introduced as the process of identity verification, which should be completed before the authentication process.
2. *Authorisation*, which is defined as a process of granting the access right to an entity to use a resource.
3. *Auditing*, which is defined as the process of reviewing the access control system records and activities to detect any security breach. Auditing is necessary to ensure that the defined access policies are compliant with the operational security of the system.

An access control system has three basic elements [19]: (1) the subject is an entity that wants to access an object. (2) The object, which is a resource and a target of access request by subjects, and (3) access right policy, which defines the way in which an object can be accessed by subjects. Any access control system should meet the main security objectives, known as CIA: *confidentiality* by preventing unauthorised access to resources, *integrity* by preventing resources to be modified without required permission, and *availability* by assuring that the resource can be accessible and usable on demand by only authorized subject. Furthermore, an access control system may have some of the following characteristics, which are often used to evaluate the access control system [20]:

1. *Delegation*, which is the act of granting an access right (full or part of) from one subject to another subject in the system.
2. *Revocation*, namely the act of removing from a subject the access right to a resource.
3. *Granularity*, i.e., the level of details that can be used for making access decision. If the required details are explicit and limited, then the type of granularity is referred as *coarse-grained*. On the contrary, *fine-grained* access control needs more details such as subject or object attributes to make a decision about the access and govern it.
4. *Flexibility* is the ability of the access control system to adapt itself to a different situation and to govern both planned and spontaneous interactions between subjects and objects.
5. *Scalability*, i.e., the ability of an access control system to be extensible in terms of the number of subjects, objects and access right policies. Another dimension of scalability is the ability of an access control system to extend its structure and scope of operation.
6. *Lightweight*, which reflects the computational complexity or volume of network traffic that an access control mechanism imposes to the system.
7. *Heterogeneity*, which is defined as the ability of the access control system to be used in different domains, platforms, networks and technologies.
8. *Context-aware*, namely the ability of the access control system to use contextual attributes of the environment such as time and location to make an access decision.

In the field of IoT, any access control system must be scalable, heterogeneous, lightweight and context-aware due to the characteristics of IoT itself.

2.2 Access Control Models

Since Lampson's access matrix was introduced in the late 1960s, a number of access control models have been proposed. This subsection briefly describes traditional access control models before moving to emerging access control models. In a number of texts, DAC, MAC, Biba, BLP, Clark-Wilson, Chinese Wall and RBAC were classified as traditional access control models while ABAC is titled as "relatively recent" access control model [21]. This work refers to ABAC and other models that were proposed after ABAC as *emerging* access control models. It also discusses their adaptability in a scalable, heterogeneous and dynamic environment like IoT.

The *traditional access control* models include:

Discretionary Access Control (DAC) In DAC the access policy for any object is defined based on the discretion of the object's owner [22]. The earliest implementation of DAC was the Access Control List (ACL), proposed in 1971. Modern operating systems like Windows utilize this ACL-based approach. Contrary to Mandatory Access Control, in DAC a subject with certain access permission is able to delegate and revoke its permission to another subject [23].

Mandatory Access Control (MAC) In MAC, which was proposed in 1973, the access policy is enforced by the system and the access to a resource is granted if the security clearance of the subject is greater than the security level of the object. The subject who has the clearance to access the object cannot delegate her access to another subject or make any change in the access policy. In this model, multi-level security (MLS) structure is defined by the system [24]. Although traditional MAC protects the confidentiality of information, it cannot protect the integrity of information since subjects with lower clearance can modify the objects that are accessible by the subjects with higher clearance [24]. To address this problem, Bell-La Padula model embodies two major principles: (a) *no-read-up*, meaning that resources can be read only by subjects with clearance greater than or equal to the resource's classification, and (b) *no-write-down* meaning that resources can be written only by subjects with clearance less than or equal to resource's classification. In contrary to DAC, another major drawback of MAC is that a subject cannot delegate or revoke its access rights to another subject [25]. Security-Enhanced Linux (SELinux) and Mandatory Integrity Control are two examples of using MAC.

Bel La Padula (BLP) BLP model was proposed in 1973 to focus on the confidentiality of data. For this reason, BLP enforces two main security policies known as "*No Read Up*" and "*No Write Down*". "*No Read Up*" ensures that read access is granted if the security level of the subject must dominate the security classification

of the object. The “No Write Down” is defined for both “append” and “write” operations. In the former, “No Write Down” ensures that the security level of objects must dominate the security level of the subject. For the latter, it ensures that security levels of subject and objects are equal. BLP supports both MAC, by determining the access rights from the security levels associated with subjects and objects, and DAC, by governing access rights based on the access matrix.

Biba model [26] was proposed in 1977 to ensure the integrity of data. For this reason, Biba rules control the transfer of data between integrity levels. To meet this goal, subjects cannot read objects of lower integrity level, which is known as “No Read Down” policy. Also, subjects in lower integrity level cannot get write access to the objects at higher integrity level, which is known as the “No Write-Up” policy.

Clark-Wilson model was proposed [27] in 1989 to protect the integrity and uses programs as a layer between users and data items. Users are authorized to execute a specific set of programs and data items can be accessed only via those programs. The focus of this model is on the security requirements of commercial applications.

Chinese Wall access control model was proposed in 1989 to avoid conflicts of interest when dealing with different subjects [28]. Conflicts arise when some companies are in competition and want to access or have accessed in the past the same objects. The model can address the “conflict of interest” for both MAC and DAC. The Chinese-Wall policy combines commercial discretion with legally enforceable mandatory controls.

Role Based Access Control (RBAC) Ferraiolo et al. [29] proposed RBAC in 1992 to address the management complexity of DAC and MAC [30]. In RBAC access is regulated based on the roles of the individuals within an organization. In other words, individuals performing specific roles can request access to specific resources that are necessary for this role. RBAC supports scalability and granularity and enforces the *principle of least privilege* [31]. According to the principle of least privilege, a subject should operate using the minimum set of privileges necessary to complete the task. Enforcing this principle mitigates the damage of unintended errors. Furthermore, RBAC supports separation of duties by ensuring that at least two different individuals are responsible for carrying out the various steps of any critical task [32].

The emerging *access control* models include:

Attribute-Based Access Control (ABAC) In the ABAC model when a subject request access to an object, the decision will be made based on the subject’s attributes, the object’s attributes and the environment’s attributes [33]. ABAC is widely used in the current Internet because it supports fine-grained access policies [34].

Capability Based Access Control (CapBAC) CapBAC was introduced in [35] and governs access requests based on tokens. The subject must have a valid token to request access to a resource and the token must be tamper proof. In this chapter, we classify those capability-based access control as “emerging access control” that benefit from using lightweight encryption algorithms like Elliptical Curve

Cryptography (ECC) to create an attribute-based encryption (ABE) access control models. In CapBAC, the subject needs to show the resource owner its token prior to performing corresponding resource request operations. This model has been used in several large-scale projects like IoT@WORK.¹

Access Control based on Usage Control (UCON) In this model, an access decision will be made using three factors [36]: (a) *authorization rules* which define the access policy based on the subject and object attributes not only prior to the access but also during the access, (b) *Obligations* which is responsible to verify mandatory requirements for a subject before or during an access, and (c) *conditions* that evaluate current environment or system status for usage function. The most important aspect of this model is that if the access attributes change while the access is granted, and this change leads to a security breach, then the granted access is revoked, and the usage is cancelled. This holds because the subject and object attributes are mutable. Mutable attributes can change as a consequence of an access [37].

Organizational Based Access Control Model (OrBAC) OrBAC proposed by A. Kalam et al. [38] is an extension of the RBAC model in a way that organization is considered as a new dimension. Contrary to DAC, MAC and RBAC, in this model policies are not restricted to static access rules only, but also include contextual rules related to access permissions, prohibitions, obligations and recommendations.

As discussed earlier, the above-mentioned access control models have been introduced to address a number of challenges in technological paradigms that preceded the IoT. As two recent works suggest [13, 20], to build or evaluate the suitability of an access control model in IoT the following criteria must be taken into consideration:

1. **Dynamism:** If the access decision must change, due to the changes in the subject, object or environment attributes, while the access is granted, then, the access control system is classified as dynamic. Otherwise, if the changes do not affect the access decision, then the access control system is static. Considering dynamism in IoT access control models is important, due to the rapid changes of contextual parameters that occur in this paradigm.
2. **Scalability:** Scalability in access control must be evaluated by three dimensions, namely an access controls has: (a) *Subject/Object (entities) scalability* if increasing the number of entities does not lead to an overhead in processing time or workload, (b) *Policy rules scalability*: if increasing the number of access rules does not lead to overhead in terms of processing time or workload., and (c) *Extensibility* if it has the ability to extend its structure to cover more sub-systems and domains. The third form of scalability can be achieved through building de-centralized structure rather than centralized structure in scalable environments like IoT.

¹At: www.probe-it.eu

3. **Heterogeneity/Interoperability:** In IoT, entities have dependencies and their workflows are tightly convergent, which increases complexity. For this reason, any access control breach in IoT can be more disruptive compared to traditional networks [39]. Furthermore, as IoT is composed of different platforms, enabling technologies and domains, designing an access control model to regulate access inter/intra domains or technologies is a must.
4. **Context-Aware:** It refers to the ability of the access control system to take contextual attributes to make an access decision. Considering contextual parameters in access decision brings flexibility in terms of tracking subject, object and environment changes if those changes have impacts on the decision.

The above evaluation criteria uncover limitations in the models (both the traditional and the emerging), making them inapplicable to IoT. As summarized in Table 1, the traditional access control models do not support the abovementioned criteria and thus cannot fit IoT. With regards to the emerging access control models, *RBAC* does not satisfy the *interoperability* criterion [40], as it cannot support the definition of roles among heterogeneous networks with different platforms and domains. Furthermore, due to the inherent *scalability* of IoT, defining a vast number of roles and associated permission rules is impossible and leads to “role explosion”.

RBAC also does not take *contextual* parameters into account during access decision. Despite the advantages of *ABAC*, *i.e.*, fine-grained access control, ease of use in a collaborative environment and flexibility, the adaption of *ABAC* in IoT is hindered due to overhead. Specifically, applying *ABAC* in IoT has a limitation in terms of computational overhead because its authorization process has high complexity, due to the consideration of attributes of subject, object and environment in the access decision. Thus, applying *ABAC* in a highly dynamic, real-time environment like IoT is infeasible due to the computational complexity that arises from the number of rules that rapidly increase with entities and contextual attributes, which may change frequently [41–43].

CapBAC is a coarse-grained access control model and does not consider contextual attributes therefore cannot satisfy the *flexibility* criterion. Moreover, even by applying lightweight cryptography algorithms, like Elliptic-curve cryptography (ECC), using *CapBAC* brings overhead to the system in *scalable* scenarios. Another concern about *CapBAC* is the distribution of token in *heterogeneous* networks that is not straightforward. Also, the model fails the *interoperability* criterion as the model cannot be applied in a heterogeneous environment. The reason is that each domain has a dedicated public key infrastructure (PKI) and there is a trust issue in inter-domain interaction between those PKIs [44]. *UCON* has the same limitations as *ABAC* in terms of scalability and extensibility. Finally, *OrBAC* suffers from the same limitation in terms of policy rules scalability as *RBAC*, as well as fails the *interoperability* criterion.

The above evaluation, which is summarized in Table 1, highlights the need of a new access control that supports mentioned characteristics for IoT domains.

Table 1 Evaluation of traditional and emerging access control models against IoT specific criteria

	Scalability			Heterogeneity Interoperability	Dynamism	Context-Aware
	Entity	Policy Rule	Extensibility			
DAC	-	-	✓	-	-	-
MAC	-	-	-	-	-	-
RBAC	✓	-	-	-	-	-
CapBAC	-	✓	-	-	-	-
ABAC	-	-	✓	✓	✓	✓
UCON	-	-	-	✓	-	✓
OrBAC	✓	✓	✓	-	-	-

3 Indeterminacy in Access Control

Indeterminacy has not had the attention that deserves as a challenge in IoT, compared to other challenges that are well-studied in the relevant literature, such as scalability, heterogeneity, interoperability and dynamism [2–6]. However, as this work stresses indeterminacy should be considered when making an access control decision in IoT. Otherwise, if the decision is based on deterministic rules regardless of the indeterminacy concept it can lead to a binary decision (Access/Deny), which does not fit in dynamic environment like IoT.

According to [45], there are at least two facets for indeterminacy: *uncertainty* and *ambiguity*. In this work we consider that uncertainty is caused by the lack of information about the likelihood of an event occurring. Also, ambiguity is caused by the lack of precision in the required information to make a decision. In the rest of this section, uncertainty and ambiguity in access control are discussed.

3.1 Uncertainty in Access Control

For many years the word “Randomness” was used to describe the probabilistic phenomenon. According to [45], Knight and Keynes started using the “uncertainty” term for the first time in 1921 and 1936 respectively. They made great progress to break the monopoly of probability theory [45]. Since then uncertainty draws attentions in different disciplines (e.g. Economy, Management). As mentioned earlier, *uncertainty* is caused by the lack of information about the occurrence of an event. In other words, uncertainty refers to a state in which the following question cannot be answered in a deterministic way: *What event will occur?* According to [46], three types of uncertainty exist, namely:

- *Aleatory Uncertainty*, which refers to an observed phenomenon that occurs randomly, therefore it is impossible to predict.
- *Epistemic Uncertainty*, which refers to the lack of information and knowledge about the properties and conditions of the phenomenon.
- *Inconsistent Uncertainty*, which refers to conflicting testimonies. The notion of inconsistency here describes the case where there is “too much” information available but this information is logically inconsistent.

In the context of a system, uncertainty leads to a situation in which an analyst cannot describe or foresee an event in the system due lack of information about it. Therefore, the main motivation for measuring uncertainty is decision making [47]. The relevant literature includes five main approaches to measure uncertainty, namely: (a) Probability theory, (b) Information Theory, (c) Evidence theory [48, 49], (d) Possibility theory [50], and (e) Uncertainty theory [51].

To define uncertainty in access control, it is essential to differentiate between its effect in the authentication and authorization phases of access control. It should be noted that as discussed in Sect. 2, an access control typically includes three phases: authentication (which encompasses identification), authorization and auditing. Auditing deals with the analysis of the other two phases to detect security violations, thus we consider that uncertainty cannot be considered in this phase.

In this work we define uncertainty for authentication and authorization as follows:

Uncertainty in authentication stems from the incompleteness of information regarding the likelihood of whether the acceptance of an authentication request leads to an incident. For instance, assume that “Alice” attempts to authenticate to a system. We assume that authenticating her, endangers the system (the access exposes an asset to a threat) with a probability 60%.

Uncertainty in authorization stems from the incompleteness of information regarding the likelihood that an authorized action of a subject could lead to an incident. Assume that “Bob” is an authenticated subject in the system and he wants to perform the “append” operation on a file. We assume that this action could cause harm with a probability of 30%.

In this work we consider that access control methods used in IoT need to treat uncertainty to be classified as uncertainty-resilient access control. This holds true, as the access control approaches that are based on a set of deterministic policies defined by system administrator cannot effectively deal with the unpredicted data access scenarios [52]. In other words, making a binary access decision based on deterministic information (e.g. credential, tokens) cannot cope with the new challenges in access control in a scalable and dynamic environment like IoT. To meet this goal in authentication and authorization, three uncertainty-resilient models have been proposed in the literature [53, 54]: (a) Break-The-Glass Access Control (b) Optimistic Access Control, and (c) Risk-Aware Access Control.

Break-the-Glass Access Control Ferreira [55] proposed a model called Break-The-Glass (BTG) to allow policy overrides. The aim of this model is to allow

unanticipated access to be provided in unexpected situations. The main application of this method is in the emergency situations in the healthcare system [56]. One of the most important problems with the BTG is the scalability of policy overriding. By increasing the number of policy overriding in a scalable environment like IoT, the access monitoring and misuse detection become impossible [57]. Another concern about BTG arises from interoperability in heterogeneous environments. Specifically, in a heterogeneous environment consisting of different networks and platforms one entity may be assigned with an overriding policy to handle emergency condition and at the same time be denied by another overriding policy from another domain. These conflicts highlight the need to accurately determine the overall decision is vital [39].

Optimistic Access Control In cases such as emergency healthcare services, the capability of an access control system to provide openness and availability is more necessary than confidentiality [58]. In this context, optimistic access control has been proposed, which assumes that most access requests will be legitimate. An optimistic approach permits the requesters to exceed their normal access rights. Therefore, putting additional control layer to protect the asset from misuse is recommended for optimistic access control. This approach suffers from the lack of scalability in terms of policy rules. This holds true as implementing defence layers in a scalable environment needs additional resources and causes computational complexity. The optimistic access control also suffers from the lack of interoperability in heterogeneous environment, as defining exceptions for access scenarios that fall between two or more domains with conflict of interests is not straightforward [53].

Risk-Aware Access Control (RAAC) The closest concept to uncertainty is “Risk”. In one hand, risk is defined as a measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of: (a) the adverse impacts that would arise if the circumstance or event occurs; and (b) the likelihood of occurrence [59]. On the other hand, uncertainty is defined as the lack of information about the likelihood of an event occurring. Therefore, “likelihood of event occurring” is common between these two concepts. Moreover, risk considers the impact of an event. For example, the value of risk in authentication phase is assessed using the following formula:

$$\text{Risk of Access request} = \frac{\text{Likelihood of incident per access request}^*}{\text{Impact of the incident}}$$

As discussed in Section 1, traditional and emerging access control models (*e.g.*, RBAC, CapBAC, ABAC) do not address uncertainty in access control. Therefore Risk-Aware access control models were proposed to address the uncertainty challenge. In these models, the risk of the access request (in authentication phase) or the risk of an action made by the subject using a permission rule is calculated to determine whether the access to a resource should be granted or the action should be permitted [60]. The main task in RAAC is to quantify the level of

risk based on the risk-benefit calculation. There are four types of the risk-benefit calculations for access control [61, 62], namely: (a) risk that focuses on information leak upon granting access to a subject, (b) risk that focuses on the cost of resource unavailability for a system upon denying access to a resource, (c) the benefit of granting access that focus on resource sharing possibility that may reduce risk, and (d) benefit of the denying access that focuses on decreasing the chance of information leak. One of the main challenges in RAAC is quantifying and calculating the risk. RAAC models use different risk metrics that calculate the value of the risk in access control system, such as: action severity, object sensitivity, benefit of access [63, 64]. There are five classes for calculating the risk [65]:

1. *Class A*: The risk is calculated based on three parameters: (1) Likelihood of threat (L_T), (2) Vulnerability of the asset (V), (3) Impact of the threat (I_T), Score of the risk = $L_T \times V \times I_T$
2. *Class B*: The risk is calculated based on the security requirements of the asset. In this method the vulnerability related to the asset (V) and the impact of the threat (I_T) are involved in the calculation of the risk: Score of the risk = $V \times I_T$
3. *Class C*: The risk is calculated in conjunction with the financial loss considerations. To meet this goal, likelihood of the threat (L_T) and the average financial loss caused by the threat (F_T) are taken into consideration: Score of the Risk = $L_T \times F_T$
4. *Class D*: In this class, the risk is calculated only for critical assets. The way of calculating the score of the risk for critical assets is the same as Class B.
5. *Class E*: The concepts of threat and vulnerability are combined to create a new concept called “Incident”. The score of the risk for this class is calculated using the likelihood of the incident (L_I) and the impact of the incident (I_I): Score of the risk: $L_I \times I_I$

According to [66], RAAC methods can be divided into two types, *non-adaptive* and *adaptive*. In *non-adaptive*, the calculated risk value for each access request remains unchanged even if any risk factor value changes during access session. This means that the access control mechanism cannot follow the changes of risk parameters after granting access. On the contrary, in the Adaptive approach the calculated risk value for each access request may change upon changes in the risk factor value or the detection of abnormal behaviour. Therefore, adaptive RAAC continuously monitor the activities to detect any suspicious event after granting access.

Most existing RAAC approaches rely on manual processes and thus are unable to provide high degree of automation in risk assessment [67–69]. This lack of automation in risk assessment leads to the requirement of manual configuration from analysts, which is costly, error-prone, and vulnerable to a social engineering attack. Moreover, the related work on RAAC that provides a conceptual framework [70] or focuses on only a domain-specific solution cannot be reused in other knowledge domains, and supports only restricted outcomes (*i.e.*, permit or deny) [71]. Therefore, these approaches suffer from lack of generalizability.

3.2 Ambiguity in Access Control

The terms “Uncertainty” and “Ambiguity” are used interchangeably in a number of studies [72, 73]. More importantly, the proposed approaches to address them are used also used interchangeably. This stems from the fact that current there is no clarity in the differentiation between these two concepts.

Aristotle first mentioned ambiguity, which is caused by imprecise information, in his works [74]. The theories that were discussed in the Section 3.1 fail to predict a system with imprecise or vaguely formulated information. This imprecise information may arise from several reasons, like the complexity of the system [75]. We deal with ambiguity when the answer to this question is not clear: *What is the occurring event?* To model and present the ambiguity, the fuzzy set theory was proposed by Zadeh [76]. With regards to access control, we define ambiguity in authentication and authorization as follows:

Ambiguity in authentication stems from the lack of precision in the information regarding the subject requesting to authenticate in the system. In other words, it aims to answer the question of ‘to what extent we can trust the subject in order to authenticate him?’. Assume that “Alice” attempts to authenticate and according to her access history, she has successfully authenticated seven times in the past out of her ten attempts. If the average of successful authentication successes in the system is 60%, then Alice is classified as “trusted” subject.

Ambiguity in authorization stems from the lack of precision in information regarding whether or not the subject performs an action aligned with the purpose that the subject was authorized for. Assume that “Bob” is an authenticated subject in the system and he wants to perform the “append” operation on a file. We assume that our system detects that the trustworthiness reequipment of the file is higher than the trustworthiness of Bob at the time of access. As a result, the “append” operation on the resource is classified as “Untrusted” action for Bob and the action is stopped.

Traditionally, ambiguity has been evaluated with the use of *trust* analysis. There are two classes of trust analysis [13, 77], (a) *credential-based trust analysis* and (b) *behavioural-based trust analysis*. In credential-based trust analysis, trust is established by verifying certain credentials. Upon the credential verification trust is established and access rights to different resources are granted, based on the access policy. This class of trust analysis is widely used in access control systems with static and pre-defined policies. Behavioral based trust analysis uses the past behavior of the subjects to predict how the subject will behave in the future. In authentication, behavioral based analysis takes the history profile of successful and unsuccessful access requests for a given subject as a metric to calculate its trust value. Reputation-based and recommendation-based trust are the subsets of this class of trust analysis that use cumulative knowledge about the past behavior of a subject [78].

Trust metrics can vary from one domain to another. In the context of access control, the following metrics have been used in the literature [79–82]: (a) Number

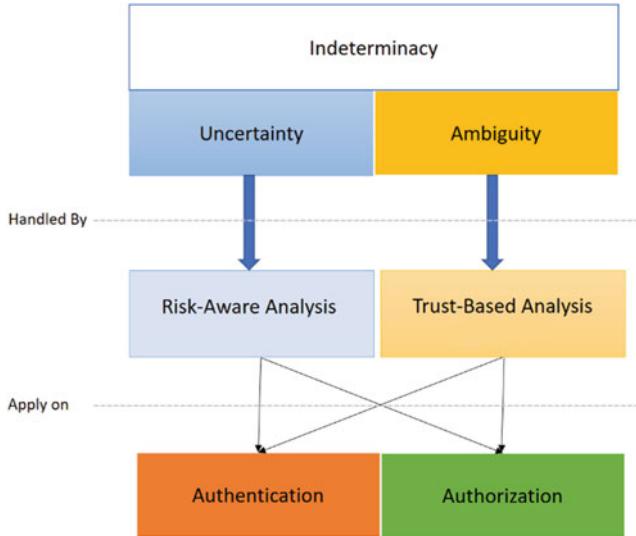


Fig. 1 Proposed framework for indeterminacy handling in the field of IoT Access Control

of authentication/authorization failures, (b) subject anonymity, (c) frequency of access request, and (d) degree of trustworthiness for subject and object.

3.3 *Proposed Framework*

As discussed in the Sects. 3.1 and 3.2, uncertainty and ambiguity cannot be treated in the same way as they are different in nature. Uncertainty must be treated as a risk-based problem. Also, ambiguity focuses on situations in which the assumptions necessary for decision making are not met. In other words, if the decision is made based on imprecise information we are in the vague state of decision making.

For this reason, we propose a framework to handle indeterminacy in the field of access control (see Fig. 1). As depicted in Fig. 1, indeterminacy encompasses uncertainty and ambiguity. According to the relevant literature that has been discussed in Sect. 3.1, uncertainty should be handled using risk assessment. Ambiguity also needs to be addressed using trust-based analysis. Selecting the right method for risk assessment and also for trust-based analysis is necessary to tackle with indeterminacy in access control, but we leave this for future work.

4 Conclusion

In this chapter, traditional and emerging access control were analysed against a set of criteria to see whether they fit in IoT. Then, besides well-studied IoT challenges in the field of access control, we introduced indeterminacy in IoT. We have also discussed how known IoT characteristics, such as scalability, heterogeneity and dynamism, exaggerate indeterminacy in IoT and make real-time access decision for such an indeterminate environment difficult. Three resilient access control paradigms including break-the-glass, optimistic access control and risk-aware access control were discussed to see whether they are able address indeterminacy in access control. Among them, risk-aware access control paradigm is able to address only one facet of indeterminacy i.e. uncertainty. This holds true as risk and uncertainty have one thing in common: “likelihood of an event occurring”. We discussed that another pillar for indeterminacy, i.e. ambiguity, can be handled through trust-based analysis. Finally, we defined indeterminacy (uncertainty and ambiguity) for authentication and authorization phases of access control and proposed a conceptual framework for handling indeterminacy.

References

1. C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context aware computing for the Internet of Things: A survey”, *IEEE Communication surveys and tutorials*, vol. 16, no. 1, 2014.
2. Wei Zhou, Yan Jia, Anni Peng, Yuqing Zhang, and Peng Liu, “The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved,” *IEEE Internet of Things Journal*, pp. 1–11, 2018.
3. Elisa Bertino, Kim-Kwang Raymond Choo, Dimitrios Georgakopoulos, Surya Nepal, “Internet of Things (IoT): Smart and Secure Service Delivery,” *ACM Transactions on Internet Technology*, vol. 16, no. 4, pp. 22–29, 2016.
4. Francesco Restuccia, Salvatore D’Oro and Tommaso Melodia, “Securing the Internet of Things in the Age of Machine Learning and Software-defined Networking,” *IEEE Internet of Things*, vol. 1, no. 1, p. IEEE Early Access Service, 2018.
5. H. Reza Ghorbani; M. Hossein Ahmadzadegan, “Security challenges in internet of things: survey,” in *IEEE Conference on Wireless Sensors (ICWiSe)*, 2017.
6. Mario Frustaci; Pasquale Pace; Gianluca Alois; Giancarlo Fortino, “Evaluating critical security issues of the IoT world: Present and Future challenges,” *IEEE Internet of Things Journal*, pp. 2327–4662, 2017.
7. C. Zhang and R. Green, “Communication Security in Internet of Thing: Preventive measure and avoid DDoS attack over IoT network,” in *IEEE Symposium on Communications & Networking*, 2015.
8. A. Nordrum, “The Internet of Fewer Things,” *IEEE Spectrum*, vol. 10, pp. 12–13, 2016.
9. Yuankun Xue, Ji Li, Shahin Nazarian, and Paul Bogdan, “Fundamental Challenges Toward Making the IoT a Reachable Reality: A Model-Centric Investigation,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 3, 2017.
10. Raffaele Giaffreda; Luca Capra; Fabio Antonelli, “A pragmatic approach to solving IoT interoperability and security problems in an eHealth context,” in *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, 2016.

11. Yanping Li; Yanjiao Qi; Laifeng Lu, "Secure and Efficient V2V Communications for Heterogeneous Vehicle Ad Hoc Networks," in *International Conference on Networking and Network Applications (NANA)*, 2017.
12. Bo Cheng, Member, IEEE, Ming Wang, Shuai Zhao, Zhongyi Zhai, Da Zhu, and Junliang Chen, "Situation-Aware Dynamic Service Coordination in an IoT Environment," *IEEE/ACM Transactions On Networking*, vol. 25, no. 4, pp. 2082–2095, 2017.
13. Sadegh Dorri, Rasool Jalili, "TIRIAC: A trust-driven risk-aware acces control framework for Grid enviroments," *Future Generation Computer Systems*, vol. 55, pp. 238–254, 2016.
14. Jiawen Kang, Rong Yu, Xumin Huang, Magnus Jonsson, Hanna Bogucka, Stein Gjessing, and Yan Zhang, "Location Privacy Attacks and Defenses in Cloud-Enabled Internet of Vehicles," *IEEE Wireless Communications*, pp. 52–59, 2016.
15. Vilem Novák, Irina Perfilieva, Antonin Dvorak, "What is fuzzy modelling?," in *Insight into Fuzzy Modeling*, Wiley, 2016, pp. 3–9.
16. Dong Xie, Yongrui Qin, Quan Z. Sheng, "Managing Uncertainties in RFID Applications: A Survey," in *11th IEEE International Conference on e-Business Engineering*, 2014.
17. "Information on RFC 4949," IETF, 1 1 2018. [Online]. Available: <https://www.rfc-editor.org/info/rfc4949>. [Accessed 1 1 2018].
18. William Stallings, "Access Control," in *Computer Security, principles and practice*, Pearson, 2017.
19. D. Gollmann, "Access Control," in *Computer Security*, Wiley, 2011.
20. Aafaf Ouaddah, Hajar Mousannif, Anas Abou Elkalam, Abdellah Ait Ouahman, "Access control in the Internet of Things: Big challenges and new opportunities," *Elsevier Computer Networks*, vol. 112, pp. 237–262, 2017.
21. William Stallings, Lawrie Brown, "Access Control," in *Computer Security: Principles and Practice, 3rd Edition*, Pearson, 2015, pp. 113–154.
22. D. Gollmann, "Chapter 5: Access Control," in *Computer Security*, John Wiley & Sons, 2011.
23. Jin, X., Krishnan, R., & Sandhu, R., "A Unified Attribute-Based Access Control Model Covering DAC, MAC And RBAC," *Springer Lecture Notes in Computer Science: Data and Applications Security and Privacy*, vol. 7371, pp. 41–55, 2012.
24. R.S. Sandhu and P. Samarati, "Access control: Principle and practice," *IEEE Communication Magazine*, vol. 32, pp. 40–48, 1994.
25. Vijayakumar, H., Jakka, G., Rueda, S., Schiffman, J., & Jaeger, T., "Integrity Walls: Finding Attack Surfaces from Mandatory Access Control Policies," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, 2012.
26. K. J. Biba, "Integrity consideration for secure computer systems. Technical Report," The MITRE Corporation, Bedford, MA, 1977.
27. D. Clark, and D. Wilson, "A comparison of commercial and military computer security policy," in *IEEE Symposium on Security and Privacy*, 1987.
28. D. F. C. Brewer and M. J. Nash., "The Chinese Wall security policy.,," in *In Proceedings of 1989 IEEE symposium on Security and Privacy*, 1989.
29. D. K. Ferraiolo, D. Kuhn, "Role Based Access Control," in *15Th International Computer Security Conference*, 1992.
30. V. Suhendra, "A Survey on Access Control Deployment," in *International Conference on Security Technology (FGIT)*, 2014.
31. Lagutin, D., Visala, K., Zahemszky, A., Burbridge, T., & Marias, G. F, "Roles and Security in a Publish/Subscribe Network Architecture," in *IEEE Symposium on Computers and Communications (ISCC)*, 2012.
32. A. Singh, "Role Based Trust Management Security Policy Analysis," in *International Journal of Engineering Research and Applications (IJERA)*, 2012.
33. W.W. Smari, P. Clemente, J.-F. Lalande, "An extended attribute based ac- cess control model with trust and privacy: application to a collabora- tive crisis management system," *Future Generation of Computer System*, vol. 31, pp. 147–168, 2014.
34. Li, J., Chen, X., Li, J., Jia, C., Ma, J., & Lou, W, "Fine-Grained Access Control System Based on Outsourced Attribute-Based Encryption," *Springer Computer Security*, vol. 8134, pp. 592–602, 2014.

35. J.B. Dennis, E.C. Van Horn, “Programming semantics for multiprogrammed computations,” *ACM Communication*, vol. 3, pp. 143–155, 1966.
36. A. Lazouski, F. Martinelli, P. Mori, “Usage control in computer security: a survey,” *Elsevier Journal of Computer Science*, vol. 4, 2010.
37. X. Zhang, M. Nakae, M.J. Covington, R. Sandhu,, “Toward a usage-based security framework for collaborative computing systems,” *ACM Transaction on Information system security*, vol. 11, 2008.
38. A. Kalam, R. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswart, A. Miege, C. Saurel, G. Trouessin, “Organization based access control,” in *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 2003.
39. Srdjan Marinovic, Robert Craven, Jiefei Ma, “Rumpole: A Flexible Break-glass Access Control Model,” in *The ACM Symposium on Access Control Models and Technologies (SACMAT)*, Austria, 2011.
40. Syed Zain R. Rizvi Philip W. L. Fong, “Interoperability of Relationship- and Role-Based Access Model,” in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, 2016.
41. Sun Kaiwen Yin Lihua, “Attribute-Role-Based Hybrid Access Control in the Internet of Things,” in *Web Technologies and Applications*, Springer, 2014.
42. Sun Kaiwen Yin Lihua, “Attribute-Role-Based Hybrid Access Control in the Internet of Things,” in *International Conference on Web Technologies and Applications. APWeb*, 2014.
43. Prosunjit Biswas, Ravi Sandhu, Ram Krishnan, “Attribute Transformation for Attribute-Based Access Control,” in *Proceedings of the 2nd ACM International Workshop on Attribute-Based Access Control*, 2017.
44. Bayu Anggorojati; Ramjee Prasad, “Securing communication in inter domains Internet of Things using identity-based cryptography,” in *International Workshop on Big Data and Information Security (IWBiS)*, 2017.
45. Y. Sakai, “J. M. Keynes on probability versus F. H. Knight on uncertainty: reflections on the miracle year of 1921,” *Springer Japan Association for Evolutionary Economics*, 2016.
46. Zhiguo Zeng, Rui Kang, Meilin Wen and Enrico Zio, “A Model-Based Reliability Metric Considering Aleatory and Epistemic Uncertainty,” *IEEE Access Journal*, vol. 5, 2017.
47. T. Aven and E. Zio, “Some considerations on the treatment of uncertainties in risk assessment for practical decision making,” *Reliability Engineering & System Safety*, vol. 96, no. 1, pp. 64–74, 2011.
48. A. P. Dempster, “Upper and Lower Probabilities Induced by a Multivalued Mapping,” *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325–339, 1967.
49. G. Shafer, *A mathematical theory of evidence*, Princeton University, 1976.
50. Baudrit, C. and Dubois, D., “Practical representations of incomplete probabilistic knowledge,” *Elsevier Journal of Computational Statistics & Data Analysis*, vol. 51, no. 1, 2006.
51. L. B, *Uncertainty Theory*, Springer, 2017.
52. Mirza, N. A. S., Abbas, H., Khan, F., & Al Muhtadi, “Anticipating Advanced Persistent Threat (APT) countermeasures using collaborative security mechanisms,” in *IEEE International Symposium on Biometrics and Security Technologies (ISBAST)*, 2014.
53. S. Savinov, “A Dynamic Risk-Based Access Control Approach: Model and Implementation,” *PhD Thesis, University of Waterloo*, 2017.
54. F. Salim, “Approaches to Access Control Under Uncertainty,” *PhD Thesis, Queensland University of Technology*, 2012.
55. A. Ferreira, R. Cruz-Correia and L. Antunes, “How to Break Access Control in a Controlled Manner,” in *19th IEEE International Symposium on Computer-Based Medical Systems*, 2006.
56. Htoo Aung Maw, Hannan Xiao, Bruce Christianson, and James A. Malcolm, “BTG-AC: Break-the-Glass Access Control Model for Medical Data in Wireless Sensor Networks,” *IEEE Journal Of Biomedical And Health Informatics*, , vol. 20, no. 3, pp. 763–774, 2016.
57. Schefer-Wenzl, S., & Strembeck, M., “Generic Support for RBAC Break-Glass Policies in Process-Aware Information Systems,” in *28Th Annual ACM Symposium on Applied Computing*, 2013.

58. D. Povey, “Optimistic Security: A New Access Control Paradigm,” in *ACM workshop on New security paradigms*, 1999.
59. Patrick D. Gallagher, “NISP SP800-30 Guide for Conducting Risk Assessment,” NIST, 2012.
60. Molloy, I., Dickens, L., Morisset, C., Cheng, P. C., Lobo, J., & Russo, A., “Risk-Based Security Decisions under Uncertainty,” in *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*, 2012.
61. Fugini, M., Teimourikia, M., & Hadjichristofi, G., “A web-based cooperative tool for risk management with adaptive security,” *Elsevier Journal of Future Generation Computer Systems*, 2015.
62. Molloy, I., Dickens, L., Morisset, C., Cheng, P. C., Lobo, J., & Russo, A., “Risk-Based Security Decisions under Uncertainty,” in *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*, 2012.
63. Hany F. Atlam, Ahmed Alenezi, Robert J. Walters, Gary B. Wills, Joshua Daniel, “Developing an adaptive Risk-based access control model for the Internet of Things,” in *IEEE International Conference on Internet of Things (iThings)*, 2017.
64. Hemanth Khambrammettu, Sofiene Boulares, Kamel Adi, Luigi Logrippo, “A framework for risk assessment in access control systems,” *Elsevier Computers and Security*, vol. 39, pp. 86–103, 2013.
65. Gritzalis D., Giulia Iseppi, Alexios Mylonas and Vasilis Stavrou, “Exiting the Risk Assessment maze: A meta-survey,” *ACM Computing Surveys*, 2018.
66. Khalid Zaman Bijon, Ram Krishnan, Ravi Sandhu, “A framework for risk-aware role based access control,” in *IEEE Conference on Communications and Network Security (CNS)*, 2013.
67. Giuseppe Petracca, Frank Capobianco, Christian Skalka, Trent Jaeger, “On Risk in Access Control Enforcement,” in *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*, Indianapolis, Indiana, USA, 2017.
68. Divya Muthukumaran, Trent Jaeger, and Vinod Ganapathy, “Leveraging “Choice” to Automate Authorization Hook Placement.,” in *ACM Conference on Computer and Communications Security*, 2012.
69. Sooel Son, Kathryn S. McKinley, and Vitaly Shmatikov, “Fix Me Up: Repairring Access-Control Bugs in Web Applications,” in *Proceedings of the 20th Annual Network and Distributed System Security Symposium*, 2013.
70. Salehie, M., Pasquale, L., Omoronyia, I., Ali, R., & Nuseibeh, B., “Requirements-driven adaptive security: Protecting variable assets at runtime,” in *20th IEEE International Conference on Requirements Engineering Conference (RE)*, 2012.
71. Zhao, Z., Hu, H., Ahn, G. J., & Wu, R., “Risk-aware mitigation for MANET routing attacks.,” *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 250–260, 2012.
72. Nick Firoozye, Fauzian Arrif, *Managing Uncertainty Mitigation Risk*, Springer, 2016.
73. J. Bancroft, *Tolerance of Uncertainty*, Author House, 2014.
74. J. Barnes, *The Complete Works of Aristotle: The Revised Oxford Translation*, Princeton, 1984.
75. “Towards Fuzzy Type Theory with Partial Functions,” *Springer Journal of Advances in Fuzzy Logic and Technology*, 2018.
76. L.A. Zadeh, “Fuzzy sets,” *Information and Control*, vol. 8, no. 3, 1965.
77. Ava Ahadipour, Martin Schanzenbach, “A Survey on Authorization in Distributed Systems: Information Storage, Data Retrieval and Trust Evaluation,” in *The 16th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (IEEE TrustCom-17)*, 2017.
78. Loubna Mekouar, Youssef Iraqi, Raouf Boutaba, “Reputation-Based Trust Management in Peer-to-Peer Systems: Taxonomy and Anatomy,” in *Handbook of Peer-to-Peer Networking*, Springer, 2009, pp. 689–732.
79. “CASTRA: Seamless and Unobtrusive Authentication of Users to Diverse Mobile Services,” *IEEE Internet of Things Journal*, vol. Early Access, pp. 1–16, 2018.
80. Guoyuan Lin; Danru Wang; Yuyu Bie; Min Lei, “MTBAC: A mutual trust based access control model in Cloud computing,” *IEEE Communication*, vol. 11, no. 4, 2014.

81. Zheng Yan, Xueyun Li, Mingjun Wang and Athanasios V. Vasilakos, “Flexible Data Access Control Based on Trust and Reputation in Cloud Computing,” *IEEE TRANSACTIONS ON CLOUD COMPUTING*, vol. 5, no. 3, pp. 485–498, 2017.
82. Lan Zhou, Vijay Varadharajan, and Michael Hitchens, “Trust Enhanced Cryptographic Role-Based Access Control for Secure Cloud Data Storage,” *IEEE Transactions On Information Forensics And Security*, vol. 10, no. 11, pp. 2381–2395, 2015.

Private Cloud Storage Forensics: Seafile as a Case Study



Yee-Yang Teing, Sajad Homayoun, Ali Dehghantanha, Kim-Kwang Raymond Choo, Reza M. Parizi, Mohammad Hammoudeh, and Gregory Epiphaniou

Abstract Cloud storage forensics is an active research area, and this is unsurprising due to the increasing popularity of cloud storage services (e.g., Dropbox, Google Drive, and iCloud). Existing research generally focuses on public cloud forensics (e.g., client device forensics), rather than private cloud forensics (e.g., both client and server forensics). In this paper, we aim to address the gap by proposing a framework for forensics investigations of private cloud storage services. The

Y.-Y. Teing

Faculty of Computer Science and Information Technology, University Putra Malaysia, Seri Kembangan, Malaysia

S. Homayoun

Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran

e-mail: S.Homayoun@sutech.ac.ir

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada
e-mail: ali@cybersciencelab.org

K.-K. R. Choo

Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX, USA

e-mail: raymond.choo@utsa.edu

R. M. Parizi

Department of Software Engineering and Game Development, Kennesaw State University, Marietta, GA, USA

e-mail: rparizi1@kennesaw.edu

M. Hammoudeh

School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester, UK
e-mail: M.Hammoudeh@mmu.ac.uk

G. Epiphaniou

Wolverhampton Cyber Research Institute (WCRI), School of Mathematics and Computer Science, University of Wolverhampton, Wolverhampton, UK
e-mail: G.Epiphaniou@wlv.ac.uk

proposed framework elaborates on procedures and artefact categories integral to the collection, preservation, analysis, and presentation of key evidential data from both client and server environments. Using the proposed framework to guide the investigation of Seafile, a popular open-source private cloud storage service, we demonstrate the types of client and server side artefacts that can be forensically recovered.

Keywords Forensic science · Digital forensics · Cloud storage forensics · Private cloud investigation · Cloud server forensics · Seafile forensics

1 Introduction

Cloud computing has been trending for the past few years, and cloud storage services are popular with both individual and organizational users [1, 2]. Vendor revenue from the sales of cloud infrastructure was reportedly USD 7.6 billion in the third quarter of 2015, representing a strong year-over-year growth by 23.0%. It is further predicted that the cloud-computing market will be worth more than USD 107 billion, driving 17% of the IT product spending by 2017.

As with any consumer technologies, cloud computing such as cloud storage services can be exploited by criminals [3–7]. For example, public cloud servers can be compromised with the aims of leaking and exfiltrating sensitive data outsourced to the data [8], facilitating distributed denial of service attacks [9, 10] and resources attacks [11, 12], launching side channel attacks [13], hiding of criminal tracks [14], storing incident materials [15], sharing illegitimate contents [16] and other criminal endeavours (e.g., storing incriminating data in the cloud). Data leakage can also be a result of malicious insiders within the cloud service providers [17]. For example, a rogue administrator may abuse their privileges to obtain sensitive corporate data for insider trading, while a disgruntled employee may exact revenge by sabotaging data hosted in the cloud [18]. The Federal Bureau of Investigation (FBI) also revealed that cyber incidents involving former or disgruntled employees resulted in losses of approximately USD 3 million. There have also been concerns raised by researchers and users about the potential for cloud service providers (CSPs) to infer customers' private data, such as transactions and medical records [19–21]. Such concerns are compounded by the revelations of sealed search warrants and the National Security Agency (NSA)'s surveillance programs (e.g., PRISM and MUSCULAR) that allegedly enable government agencies to collect user data from CSPs without prior consent of the data owner [22]. Thus, it is not surprising that preserving the privacy of user data in public clouds is an active research focus [23–25]. Due to the concerns associated with the use of public cloud services, private cloud services are increasingly preferred by organisations, particularly larger organisations (e.g. universities and fortune 500 organisations) and organisations dealing with sensitive user data (e.g. healthcare and banking industries) [19–21, 26].

Cloud services, public or private, are rich sources of evidential data in the event of a criminal investigation (e.g. cyber incident such as security breach) or civil litigation (e.g. theft of intellectual property from data stored in a public/private cloud service [27, 28]). As noted by Martini and Choo [29], the forensic investigation of different cloud services and models could entail different requirements and challenges. In private cloud forensics, the focus is not only on acquisition of evidential data from the client devices but also on the cloud hosting instance [30]. A failure to contemplate the cloud computing architecture could result in crucial evidence being missed or even corrupted [31, 32]. The situation is complicated by the deployment of chunking mechanisms for effective data deduplication [33]. As data structures are often heterogenous (dependent on vendor-specific data models), traditional evidence collection tools, techniques and practices are unlikely to be adequate [34]. For example, a forensic investigator would need to have the breadth and depth understanding of the different file system structures and tools in order to collect evidence from the wide range of cloud services, such as Seafire server [35–37], and client devices that have been used to access or synchronise with the cloud services [38, 39] (e.g. personal computers, Android devices, iOS devices, and other mobile devices).

In addition, in a private cloud deployment, the configurations will most likely vary between organisations [40] and as such, the investigation would necessitate the organisation's involvement during data preservation and collection [41]. The chain of custody could be broken when the employees acting as first-responders are not trained to preserve evidence according to the court accepted standards [42]. On the other hand, organisations may choose not to cooperate in a criminal investigation due to exacting privacy regulations [43] (e.g. in the recent unlocking case of San Bernardino shooter's iPhone). In the worst case scenario, the organisation deploying the private cloud may even collude with the suspect to erase and destroy evidential data (e.g. deleting evidential data on the server, modifying access logs, or physically damaging the server) for fear of reputational damage [44]. The challenges are further compounded in a decentralised data storage setting [29, 45–47], cross-jurisdictional investigation [48–51], virtualisation [52], large-scale data processing [53–56, 56, 57], the use of encryption, poor documentation [58], and bring-your-own-device policy.

Hence, in this paper, we seek to contribute to the body of knowledge in this active field of research. Specifically, we extend the framework of [59] and present a conceptual framework for forensic investigation of private cloud storage services. We then demonstrate the utility of the framework in investigating by investigating Seafire, a popular cloud storage solution. By attempting to answer the following questions in this research, we provide an in-depth understanding of the residual artefacts of forensics value in private cloud forensic investigation.

1. What residual artefacts can be found on hard drive and physical memory after a user accessed the Seafire desktop client application?
2. What residual artefacts of forensics interest can be detected after a user accessed the Seafire mobile client application?

3. What data of forensics value can be located from a server hosting the Seafile private cloud storage service?
4. What artefacts of forensics interests can be recovered from the network communications between Seafile client and server?

The structure of the paper is as follows: In the next two sections, we discuss “Related Work” and present our conceptual “Private Cloud Forensics Framework”, respectively. In “Background”, we provide an overview of Seafile private cloud storage service. The research environment is described in “Experiment Setup”. We then present the findings of Seafile client and server forensic investigations in “Client Forensics” and “Server Forensics (Divide and Conquer)”, respectively, before concluding the paper in the last section.

2 Related Work

Marty [60] proposed a central log collector incorporating the log entries of forensic interests from server-side application components including Django, JavaScript, Apache, and MySQL. However, Zawoad and Hasan [61] argued that the concept does not solve issues relating to the collection of network logs and file metadata. By leveraging the Application Programming Interfaces (API), forensic researchers such as Quick et al., Dykstra and Sherman [62], Gebhardt and Reiser [63], Zawoad et al. [64], and Martini and Choo [65] developed frameworks and prototype implementations that support remote collection of residual evidences from virtual machines. While data collection using APIs could reduce the involvement of CSPs in investigation, such data collection methods could be limited by the APIs’ feature sets and modify the organisation’s logs [65]. Dykstra and Sherman [66] assessed the effectiveness of collecting evidence remotely from the Amazon EC2 servers using popular remote data acquisition agents and reported EnCase Servlet and FTK Remote Agent as the most time effective data acquisition methods. In another study, Thethi and Keana [67] found that FTK Imager Lite and Amazon Web Services (AWS) snapshot functionality provided the most time effective data collection method for ephemeral and persistent storage volumes, respectively. However, the remote data acquisition agents typically require a root privilege in the host OS; hence, potentially compromising the evidence’s chain of custody [66].

Other cloud applications such as XtreemFS, VMWare, Evernote [68] Amazon S3 [68], Dropbox [68, 69], Google Drive [68, 70], Microsoft Skydrive [71], Amazon Cloud Drive [72], BitTorrent Sync [73], SugarSync [40], Ubuntu One [30], huBic [41], Symform [15], Mega [16], ownCloud [74], as well as mobile cloud apps [34, 75], have also been examined. These forensic studies demonstrated the types of artefacts that could be forensically recovered from the storage media in desktop computers, laptops, and mobile client devices [76]. A number of studies also

demonstrated that it is possible to recover parts of the cloud synchronizing history and synchronised files from unstructured datasets such as memory dumps, slack space, free space, and swap files in plain text [77]. Quick and Choo [78] studied the integrity of data downloaded from the web and desktop clients of Dropbox, Google Drive, and Skydrive and identified that the act of downloading data from client applications does not tamper the contents of the files, despite changes in file creation/modification times. Quick and Choo [69–71] also identified that data erasing tools such as Eraser and CCleaner could not completely remove the data remnants from Dropbox, Google Drive, and Microsoft SkyDrive.

In the literature, the majority of existing studies focus on client-side investigation of public cloud services, and there are only a small number of forensic studies that examine private cloud solutions [29, 46] (25, 70)—see Table 1. To the best of our knowledge, there is no existing study that examines Seafile. This is the gap we aim to address in this paper.

Table 1 Summary of related work on cloud forensics

Work	Public	Private
Martini and Choo [29]		✓
Marty [60]	✓	
Dykstra and Sherman [62]	✓	
Gebhardt and Reiser [63]	✓	
Zawoad et al. [64]	✓	
Martini and Choo [65]	✓	
Dykstra and Sherman [66]	✓	
Thethi and Keana [67]	✓	
Chung et al. [68]	✓	
Quick and Choo [69]	✓	
Quick and Choo [70]	✓	
Quick and Choo [71]	✓	
Hale [72]	✓	
Farina et al. [73]	✓	
Shariati et al. [40]	✓	
Shariati et al. [30]	✓	
Blakeley et al. [41]	✓	
Teing et al. [15]	✓	
Daryabar et al. [16]	✓	
Martini and Choo [74]		✓
Martini et al. [79]	✓	
Daryabar et al. [34]	✓	
Quick and Choo [78]	✓	

3 A Conceptual Private Cloud Forensics Framework

A competent practitioner should adhere to generally accepted principles, standards, guidelines, procedures, and best practices [80]. Mckemmish [81], for example, suggested four rules for digital forensics investigation, namely: minimal handing of the original, account for any changes, comply with the rules of evidence, and finally not to exceed one's knowledge (e.g. a practitioner competent in Android device forensics may not necessarily be competent in BlackBerry device forensics). There are overlaps between Mckemmish's four rules and those presented by United Kingdom Association of Chief Police Officers (ACPO). The latter's four rules for digital investigation stated that no action should change data, when it is necessary to access original data the persons accessing data should be competent to do so, a record of processes should be made, and the investigator in charge is responsible to ensure the principles are adhered to. The NIST digital forensic guidelines [80] prescribed that digital forensics should be undertaken based on the four basis phases: collection, examination, analysis, and reporting.

The first cloud forensic framework was proposed by Martini and Choo [59], which was derived based upon the frameworks of NIST [80] and McKemmish [81]. The framework was used to investigate XtreemFS [29], VMWare [65], Amazon EC2 [72], ownCloud [74], etc. Quick et al. [46] further extended and validated the four-stage framework using Dropbox, Google Drive and SkyDrive [71]. Chung et al. [68] proposed a methodology for cloud investigation on Windows, Mac OS, iOS, and Android devices. The methodology was then used to investigate Amazon S3, Google Docs, and Evernote. Scanlon et al. [82] outlined a methodology for remote acquisition of evidence from decentralised file synchronisation network and utilised it to investigate BitTorrent Sync [73, 83]. By adapting the adversary model widely used in network security and cryptography, Do et al. [84] proposed the first adversary model for digital forensics. They then demonstrated how such an adversary model can be used to investigate mobile devices, such as Android smartwatch [85] and Android apps [79]. Ab Rahman et al. [86] posited the importance of proactive forensic readiness and proposed a conceptual forensic-by-design framework to integrate forensics tools and best practices in the design and development of cloud systems.

A great majority of existing cloud forensics frameworks are either too generic, or rigid to accommodate the customised nature of private cloud. Moreover, the frameworks are client-centric and have failed to provide a link between the client side investigation and the cloud hosting instance (e.g., how the client artefacts could assist with the collection of evidence from the server or cloud hosting environment), which is essential in private cloud investigations. Hence, we extend the cloud forensics frameworks of Martini and Choo [59] and Quick et al. [46] to present our conceptual private cloud forensics framework—see Fig. 1.

The framework comprises seven phases, as follows:

1. *Readiness*: This phase is concerned with understanding the architecture and technologies involved in deploying and maintaining a private cloud infrastruc-

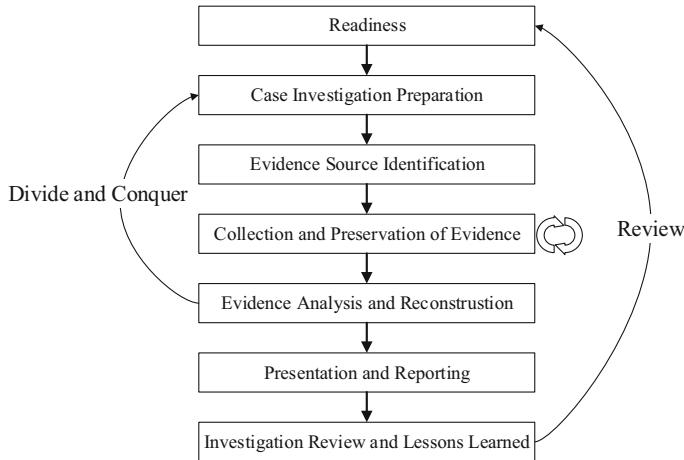


Fig. 1 Our conceptual private cloud forensics framework

ture such as data model, software/hardware components, network header, file format, programming syntax, and commands. As it is not realistic to expect any practitioner to be well-versed in every cloud computing technology and the underlying infrastructure (e.g. servers and client devices), a research or review of the documentation is essential to familiarise one with the potential approaches, tools or expertise required in the collecting and analysing of data, scoping the investigation boundaries, setting up the forensic workstation, or even refuting any defense that a piece of evidence was associated with the target cloud platform during a case trial (e.g., the third and fourth rules of the Daubert Standard require a forensic procedure to be documented/published for peer review and generally acceptable in the pertinent scientific community [87]). As most of private cloud solutions are fully or partially open source, there is usually a sufficient wealth of information available on the vendor website, Github or Wiki pages, as well as other publications (e.g. technical reports and academic publications). Alternatively, the practitioners can reach out for additional expertise from the vendor (if the vendor is willing to assist in investigation) or other forensic researchers familiar with the target cloud platform. Additionally, an ongoing development/refreshing of skills through training, academic resources, and reviewing of case-studies can help address competency issues relating to the cloud evidence handling.

2. *Case Investigation Preparation:* In this phase, the investigators prepare search warrants, with the assistance of the forensic practitioners. The latter also helps to gather information regarding the organisation-specific cloud deployment, determine the potential effects on investigation, and identify third-party service providers (if any) for cooperation. For data hosted overseas, the practitioner and investigator may need to rely upon mutual assistance from the host country [80]. Preparation also includes other aspects of investigations, such as defining roles

and responsibilities and customising forensic workstation to extract and preserve evidences stored in an uncommon or proprietary format. For example, forensic investigation of the BitTorrent Sync cloud applications may require a bencode editor to parse the bencode configuration files.

3. *Evidence Source Identification:* In this phase, possible sources of evidence are identified and seized. The phase typically commences with the identification of client devices such as desktop computers, laptops and mobile devices. During the second (or subsequent) iteration, the practitioner identifies the remote hardware (typically a cloud hosting server, peer node, etc.) that the client devices were connected to, based on the corresponding IP addresses, node identifications (IDs), computer names, and other relevant identifying information recovered from the client devices. The identity of the suspicious host could be validated through contacting the IP address owner (e.g. via the WHOIS query), seeking assistance from the Internet Service Provider (ISP), researching the IP address history (via the network traffic, Security Operation Centre logs, incident response archives, Internet search engines, etc.), and looking for clues in the data or log files.
4. *Collection and Preservation of Evidence:* In this phase, the evidence is collected and preserved. Although the data collection and preservation processes are structured and separated as different forensic phases in Martini and Choo's [59] framework, we contend that the processes should be interchangeable to provide the flexibility to accommodate the framework in different data acquisition scenarios and stages. The general process begins with creating a bit-for-bit image (preservation) of the target node(s), such as desktop/mobile clients and server hosting the cloud instance. In the scenarios involving virtualisation, physical acquisition of the server can be facilitated by exporting the virtual hard disk and memory from the host machine. Once the file system is preserved, a forensic browser can be used to extract potential evidence from the working copy(ies), including cloud instances with an uncommon or proprietary format (bencode file, MySQL database, etc) for analysis using vendor-specific tools in latter phases (hence, preserve and collect). In the situation when physical acquisition is not possible, this phase commences with logical acquisition (collection) of the visible data structures on the target system. As the structure of the data directory may vary it is recommended to fully copy user's data directory. If the directory is huge and too complex to be copied in full, then the practitioners may copy directories matching keywords of interest (e.g. name of application, vendor, or specific components) provisioned during the Readiness phase. The data collection can be undertaken locally (on the target system), or remotely via API calls or a remote acquisition software (e.g. Forensic Toolkit—FTK, and Remote Agent [88]). Collected data can be preserved into a logical container, such as Encase Evidence (E01), Encase Logical Evidence (L01), FTK Imager (AD1), or X-Ways Evidence File Container (CTR) format (hence, collect and preserve). Common categories of artefacts from client devices are as follows:

- Synchronisation and file management metadata—The property information used to facilitate the synchronisation process between the client(s) and the hardware hosting the cloud environment. These include the node/peer IDs, device names, owner's name, version and licensing information (containing the vendor's contact details), Unified Resource Locators (URL) to the main servers, session IDs, filenames and IDs for the sync directories/files, share keys (if any), and other unique identifying properties useful for user profiling, identifying the corresponding nodes, and conducting file-to-device mapping during the latter phase. The version information can prove useful in the scoping of the investigation. For example, if Seafile version 4.1 or later is used, then the practitioner should pay attention to the trash folder for the deleted files. The synchronisation and file management metadata can be typically located in application-specific configuration files, log files, and supporting files (documentations, user manuals, help files etc.) in the install, sync, or data directory. In Windows OS, additional property information can potentially be located in the Windows registry; Mac OS, in the OSX property lists (PLIST) files such as in /%Library%/Preferences/; and Linux OS, in the application-specific configuration files in /User/<User Profile>/.config/. The property information can also vary according to applications. Additionally, records of the running processes (e.g., process identifiers, process names, and creation times) could be recovered using the ps command provided by UNIX-based OSs.
- Authentication and encryption metadata—Artefacts describing the encryption algorithm, public/private keys or certificates, encryption salt, initialization vectors (IVs), password hashes, login URL, and possibly plaintext credentials that could provide the opportunity to login to the user's cloud account. Other than the cloud application generated instances, the practitioners can use a third-party utility to dump and extract password hashes from the password vaults of the OSs and web browsers (e.g. if the user saves the login credentials in the web browser).
- Cloud transaction logs—Records of user actions (install, uninstall, create user account, file upload, file download, file modification, file delete, login, logoff, etc.) made to the cloud instance, alongside the directory path and the timestamp information, which can often be located in the data directory. However, the data is likely to be in different formats ranging from raw log files to clear text, XML, PLIST (in a Mac OSX), and a variety of database file formats. Other potential sources of transaction logs/caches in the Windows OS include the registry, \$LogFile, MFT, UsnJrnl, prefetch files, and shortcuts; in the Linux OS are Zeistgeist, Gnome's recently-used.xbel, and other log files in /var/log/; in the Mac OSX OS are the Quick Look thumbnail cache, preference files (in /%Library%/Preferences/), and other log files in /private/var/logs/. Web browsing history and other web-based transactions artefacts are also helpful remnants during investigations.

- Data storage—Actual file storage or temporary holding places that may be relied upon for recovery of the synced files if the cloud environment is not accessible or local copies of files have been deleted (e.g. the client application generated sync/archive directories, web browsing caches, thumbnail caches, trash directories, and unallocated space).
 - Network captures—Copies of the network packets transmitted between client and the cloud hosting instance. In all scenarios, the artefacts may comprise the IP addresses and unified resource identifiers (URIs) of the cloud hosting instance, as well as records of the connected IP addresses, port numbers, socket information, and corresponding timestamp information [69–71]. In the event when the network traffic is unencrypted, then the practitioner can potentially recover copies of the transferred files and a full set of the HTTP request/response headers in plain text, providing the login credentials, property information of the transferred files (in the HTTP message body), references to the relevant web directories (in the ‘Referer’ HTTP header) and other relevant cloud transaction details.
 - Volatile memory dumps—Volatile memory is a potential source of the above mentioned artefacts [77]. The volatile memory captures should be undertaken prior to the file system acquisition and shutting down of the machine. Practitioners should also consider swap files as a potential source for overloaded volatile memory. Common categories of artefacts from private cloud servers are as follows:
 - Administrative and file management metadata—The server application generates instances storing records of the clients’ synchronisation and file management metadata as well as account information (e.g., login usernames, email addresses, password hashes, encryption salt, and other encryption properties) that can be used to support the evidence recovered from the client devices. The data can generally be found in database, XML, and other proprietary-format files. In cases when physical acquisition of the server is not feasible, then database collection can be instantiated by copies of the database dump files.
 - Cloud logging and authentication data: Records of cloud transaction and authentication events/requests generated by the clients along with identifying information such as node IDs, logical addresses, session IDs, and timestamps that will facilitate provenance determination. These records can be typically located in the cache or log files of the target cloud service, Database Management System (DBMS), web server (Apache, Nginx etc.), and other application components (Django, Javascript etc.) [60].
 - Data repositories—The data uploaded by the users to the offsite cloud repositories (typically present in encrypted chunks or blocks) which enables the practitioners to recover synced files from the cloud hosting environment.
5. Evidence Analysis and Reconstruction: In this phase, collected evidences are processed, analysed, and interpreted in a forensically sound manner. It is during this phase that the practitioners attempt to recover information useful

for establishing connections between the suspect and the crime. The evidential data may potentially reveal location(s) of the server or corresponding nodes, and subsequently lead to a second or more iterations of the process.

6. Presentation and Reporting: This phase remains very similar to the frameworks of NIST [80] and McKemmish [81], which involves legal presentation of the collected evidential data in a court of law in a unified format. In general, the investigation report should include information on the expertise and qualifications of the presenter, the credibility of the processes employed to produce the evidence being tendered, description of utilised tools and applications, and any limitations to avoid falsification of conclusions being made [89–92].
7. Investigation Review and Lessons Learned: Finally, the practitioners review the investigative process, determine any problems that may need to be remediated, and document the lessons learned for future references. The learned lessons are great resources for the readiness phase of future investigations.

4 Background

Seafile [] is an open-source file hosting system created in 2009. There are reportedly more than 200,000 users, including large organisations such as Wuppertal Institute for Climate, Environment and Energy, University of Strasbourg, and Royal Belgian Institute of Natural Sciences [93]. The self-hosted version comes in two editions, namely: a “community edition” (released under the terms of the GNU General Public License) and a “professional edition” (released under a commercial license). The main difference being that the professional edition provides additional features such as selective sync, full text file search, lightweight Directory Access Protocol (LDAP) group syncing, audit logging, and AWS S3 backend support [93]. The cloud service can be accessed using a web browser or a client application (‘Seahub’ web application) available for personal computers running Windows, Mac OS, or Linux OS and mobile devices running iOS, Windows Phone 8, or Android OS. However, only the non-mobile application supports group discussion, private messaging, and file sharing (private, group, public, or anonymous).

The users are required to create an admin account during the server setup, which can be used to manage (add, delete, change password, etc.) the user accounts and synchronisation settings. Seafile organises the sync data in libraries (Seafile term for sync folders) [94], and the users may use the server-created default library (‘My Library’) or create new libraries. Each library has an owner that has the ability grants read-only or read-write access to the library or individual documents in the library. Libraries shared with read-only permission are only limited to one-way synchronisation, where modifications made at the client will not be synced with the server. The libraries can be encrypted locally using the built-in library encryption feature [95], but the encryption feature is only supported by desktop and iOS clients. The newer versions of Seafile deployment (version 4.1 or later) use HTTP protocol by default, but users can manually enable syncing in HTTPS.

The desktop client keeps track of the libraries in four main lists, which are ‘My Libraries’, ‘Synced Libraries’, ‘Recently Updated’ and ‘Private Shares’. The ‘My Libraries’ list keeps track of the libraries associated with the user currently logged in from the target system; ‘Synced Libraries’ list displays the libraries that were synced with the target system; ‘Recently Updated’ list holds the recently updated libraries along with the latest update times; and the ‘Private Shares’ list shows the libraries shared by other users with the currently logged in user. In Seafile version 4.0.0, the cloud file browser was added to enable browsing of unsynced libraries with the desktop clients.

Seafile keeps track of files in the libraries internally using a GIT-like data model, which consists of repo (data model term for the library), branch, commit, FS, and blocks [35]. A Seafile 4.0 server is generally built upon the following three modular components [96]:

- Seahub (django): the web interface of the Seafile server, which runs applications within Python HTTP server gunicorn. It can be deployed behind a reverse proxy such as Apache or Nginx.
- Seafile server (seaf-server): the data service daemon that handles file upload/down-load/syncing.
- Ccnet server (ccnet-server): the Remote Procedure Call (RPC) service daemon for enabling communication among multiple components. Users are required to setup a database for each of the above components, using MySQL or SQLite the backend option [97]. For SQLite, the database files can be located under the server’s data directory at:
 - %ccnet%/PeerMgr/usermgr.db: contains user information.
 - %ccnet%/GroupMgr/groupmgr.db: contains group information.
 - %seafolder-data%/seafolder.db: contains library metadata.
 - %seafolder-data%/seahub.db: contains tables used by the web front end (seahub).

For MySQL, the databases are created by the administrator; hence, the names can vary according to deployments. The database names suggested in the Seafile Server manual are as follows:

- ccnet-db: contains user and group information.
- seafolder-db: contains library metadata.
- seahub.db: contains tables used by the web front end (seahub). Although the sync files are saved in blocks in the repositories, the administrator can gain ‘read-only’ access to the unencrypted libraries using the ‘seaf-fuse’ extension on the server running Seafile server application version 4.2.0 or later [36]. On the other hand, the ‘seaf-fsck’ [37] extension can be used to repair the corrupted libraries and export the libraries externally from the repositories.

5 Experimental Setup

The scope of the client forensics is to determine the synchronisation and file management and encryption and authentication metadata, as well as cloud transaction logs (as defined in our research) produced by the Seafile client (version 4.2.X) and mobile (version 1.X) applications. The scope of the server forensics is to determine the administrative and file management metadata, cloud logging and authentication data, and data repositories for Seafile server application community edition version 4.2.3.

In this research, we created eight [9] fictional accounts to imitate the role of administrators and clients—see Table 2. Each account was assigned a unique ‘display icon’ and username, which was not used within the application to ease identification of the user role.

Our testing environment consists of two VMware Workstations (VMs) for the servers, three [3] VMs for the desktop clients, and a default factory-restored Android and iOS device; involving the OS combinations as detailed in Table 3. The VMs were hosted using VMware Fusion Professional version 7.0.0 (2103067) on a Macbook Pro running Mac OS X Mavericks 10.9.5, with a 2.6 GHz Intel Core i7 processor and 16 GB of RAM. As explained by Quick and Choo [69–71], using physical hardware to undertake setup, erasing, copying, and re-installing would have been an onerous exercise. Moreover, a virtual machine allows room for error by enabling the test environment to be reverted to a restore point should the results be unfavourable. Nevertheless, using a mobile emulator on a computer desktop may omit the hardware features of a physical mobile device, leading to unrealistic information [98]. To this regard, physical mobile devices were used in our research. The workstations were configured with minimal space in order to reduce the time required to analyse the considerable amounts of snapshots in the latter stage. The mobile devices were jailbroken/rooted to enable root access to the devices.

In the third step, we conducted a predefined set of activities to simulate various real world scenarios of using Seafile based on the scope of research, such as setting up the cloud hosting servers, installing and uninstalling the client applications, logging in and out, as well as accessing, uploading, downloading, viewing, deleting, and unsyncing the sample files using the client/web applications. Additional experiments were conducted for the web application for file sharing, group discussions, private conversations, and account deletion. Similar to the approaches of Quick and Choo [69–71], the 3111th email message of the University of California Berkeley Enron email dataset (downloaded from http://bailando.sims.berkeley.edu/enron_email.html) was used to create the sample files and saved in .RTF, .TXT, .DOCX, .JPG (print screen), .ZIP, and .PDF formats, providing a basis for replicating the experiments in future. A new folder was created to house the set

Table 2 System configurations for Seafile forensics

<i>Windows server</i>	
	Operating system: Microsoft Windows 8.1 Enterprise (build 9600)
	Virtual memory size: 2 GB
	Virtual storage size: 20 GB
	Web server: Nginx 1.9.3
	Database server: MySQL version 5.6
	Web browser: Microsoft Internet Explorer version 11.0.9600.17351
	Server application: Seafile server application for Windows version 4.2.3
	IP address/URL: HTTP://192.168.220.180
<i>Clients</i>	
Windows client	Operating system: Windows 8.1 Professional (Service Pack 2, 64-bit, build 9600). Virtual memory size: 2 GB Virtual storage size: 20 GB Web browsers: Microsoft Internet Explorer version 11.0.9600.17351 (IE), Mozilla Firefox version 13.0 (MF), and Google Chrome version 39.0.2171.99m (GC); each of which was installed on a separate snapshot. Client application: Seafile client application for Windows version 4.2.8
Mac OS client	Operating system: Mac OS X Mavericks 10.9.5 Virtual memory size: 1 GB Virtual storage size: 60 GB Web browser: Safari version 7.0.6 (9537.78.2). Client application: Seafile client application for Mac OS version 4.2.7
<i>Ubuntu server</i>	
	Operating system: Ubuntu 14.04.1 LTS Virtual memory size: 1 GB Virtual storage size: 20 GB Web server: Nginx 1.9.3 Database server: MySQL version Web browser: Mozilla Firefox for Ubuntu version 31.0 Server application: Seafile server application for Windows version 4.2.3 64-bit IP address/URL: HTTP://192.168.220.170
<i>Clients</i>	
Ubuntu client	Operating system: Ubuntu 14.04.1 LTS Virtual memory size: 1 GB Virtual storage size: 20 GB Web browser: Mozilla Firefox for Ubuntu version 31.0 Client application: Seafile client application for Linux version 4.2.5 64-bit
iOS client	Device: iPhone 4 Operating system: iOS 7.2.1, jailbroke with Pangu8 v1.1 Client application: Seafile Pro for iOS version 1.9.1
Android client	Device: HTC One X Operating system: Android KitKat 4.4.4 rooted using Odin3 v.185 Client application: Seafile client application for Android version 1.8

Table 3 User details for Seafile experiments

Email	Password	Role
alice70707@gmail.com	alicepassword	Administrator
fbcchelper@gmail.com	fbcchelper	Administrator
windowsclient@gmail.com	windowsclient	Windows OS client
macclient@gmail.com	macclient	Mac OS client
androidmobile@gmail.com	androidmobile	Android OS client
iosmobile@gmail.com	iosmobile	iOS client
internetexplorer@gmail.com	internetexplorer	Accessing the web application using Internet Explorer browser.
googlechrome@gmail.com	googlechrome	Accessing the web application using Google Chrome browser.
mozillafirefox@gmail.com	mozillafirefox	Accessing the web application using Mozilla Firefox browser.

of sample files for the unencrypted libraries (one each for all the workstations/devices investigated), and renamed as ‘WindowsClientNotEncrypted’, ‘UbuntuClientNotEncrypted’, ‘MacClientNotEncrypted’, ‘IEClientNotEncrypted’, ‘GoogleChromeClientNotEncrypted’, ‘MozillaClientNotEncrypted’, ‘iOSClientNotEncrypted’, and ‘AndroidClientNotEncrypted’ on the respective workstations/devices. The process was repeated for the encrypted libraries ‘WindowsClientEncrypted’, ‘UbuntuClientEncrypted’, ‘MacClientEncrypted’, ‘IEClientEncrypted’, ‘GoogleChromeClientEncrypted’, ‘MozillaClientEncrypted’, ‘iOSClientEncrypted’, and ‘AndroidClientEncrypted’, before the libraries being synced across the corresponding workstations/devices.

For the purpose of this research, we used HTTP as the carrying protocol to enable the study of the network’s inner workings. Wireshark was deployed on the host machine to capture the network traffic from the server for each scenario. After each experiment, a snapshot was undertaken of the VM workstations prior to and after being shutdown, enabling acquisition of the volatile memory at a later stage in the former. For the mobile devices, we acquired a bit-for-bit image of the internal storage using dd over SSH/ADB Shell. The experiments were repeated thrice (at different dates) to ensure consistency of findings. The final step was to setup the forensic workstation using the tools in Table 4.

Table 4 Tools prepared for Seafile forensics

Tool	Usage
Work	Public
FTK Imager version 3.2.0.0	To create a forensic image of the .VMDK files.
dd version 1.3.4-1	To produce a bit-for-bit image of the mobile devices' internal storage and .VMEM files.
Autopsy 3.1.1	To parse the file system, produce directory listings, as well as extracting/analysing the files, Windows registry, swap file/partition, and unallocated space of the forensic images.
emf_decrypter.py	To decrypt the iOS images for analysis.
raw2vmdk	To create .VMDK files for the raw dd images.
HxD version 1.7.7.0	To conduct keyword searches in the unstructured datasets.
Volatility 2.4	To analyse the running processes (using the pslist function), network statistics (using the netscan function), and detecting the location of a string (using the yarascan function) in the physical memory dumps.
SQLite Browser version 3.4.0	To view the contents of SQLite database files.
Wireshark version 1.10.1	To analyse network traffic.
Network Miner version 1.6.1	To analyse and carve network capture files.
Whois command	To determine the registration information of an IP address.
Photorec 7.0	To data carve the unstructured datasets.
File juicer 4.45	To extract files from files.
Nirsoft Web Browser Passview 1.19.1	To recover the credential details stored in web browsers.
Nirsoft cache viewer, ChromeCacheView 1.56, MozillaCacheView 1.62, IECacheView 1.53	To analyse the web browsing cache.
BrowsingHistoryView v.1.60	To analyse the web browsing history.
Thumbcacheviewer version 1.0.2.7	To examine the Windows thumbnail cache.
Windows Event Viewer version 1.0	To view the Windows event logs.
Console Version 10.10 (543)	To view the log files.
Windows File Analyser 2.6.0.0	To analyse the Windows prefetch and link files.
Plist Explorer v1.0	To examine the Apple Property List (PLIST) files.
chainbreaker.py	To extract the master keys stored in the Mac OS Keychain dump.
NTFS Log Tracker	To parse and analyse the \$LogFile, \$MFT, and \$UsnJrn1 New Technology File System (NTFS) files.
MySQL version 5.6	To access the MySQL database.
seaf-fuse.sh and seaf-fsck.sh	To access unencrypted libraries on the Seafile server.

6 Client Forensics

In this section, we identify, preserve, collect, and analyse the artefacts from the desktop and mobile clients. Evidence Source Identification: The physical hardware of interest was identified, which contained the virtual disk and memory files created in our research. The mobile devices used in this research were an HTC One X running Android KitKat 4.4.4 and an Apple iPhone 4 running iOS version 7.1.2. Collection and Preservation of Evidence: A forensic copy was created of each VMDK file and mobile device image in E01 container, and VMEM file in raw image (dd) file format. An MD5 and SHA1 hash value was calculated for each original copy and subsequently verified for each working copy. The decision to instantiate the hard disks and physical memory dumps with the virtual disk and memory files was to prevent the datasets from being adulterated with the use of memory/image acquisition tools [69–71].

The process was followed by the collection of test data that matched the search terms of relevance (e.g., ‘Seafile’, ‘ccnet’, and ‘seahub’ as identified in the Readiness phase) in the hard disk images, but held formats unsupported by the Autopsy forensic browser for analysis using the tools of relevance in the latter phase, such as: SQLite database files, PLIST files, prefetch files, event files, shortcuts, thumbnail cache, \$MFT, \$LogFile, \$UsnJrnl, as well as web browsers’ data folders/files (e.g., %AppData%\Local\Microsoft\Windows\TemporaryFiles\index.dat, %AppData%\Local\Microsoft\Windows\WebCache, %AppData%\Local\Google\Chrome, %AppData%\Roaming\Mozilla\Firefox, and %AppData%\Local\Microsoft\Windows\WebCache of the Windows desktop clients). The volatile data was collected using the Volatility tools, Photorec file carver, and HxD Hex Editor for the physical memory dumps, and Wireshark and Netminer network analysis software for the network captures.

6.1 Evidence Analysis and Reconstruction

The data collected was analysed for synchronisation and file management metadata, encryption and authentication metadata, as well as cloud transaction records, which are essential for determining the data storage and logical location of the cloud hosting server.

6.2 Sync and File Management Metadata

The Seafile desktop client’s sync and file management metadata was predominantly located in %Users% \ < Windows Profile > \Seafile\seafile – data,

/home/ < User Profile > /Seafolder/.seafolder – data/, and /Users/ < User Profile > /Seafolder/.seafolder – data/ of the Windows 8.1, Ubuntu, and Mac OS clients respectively (henceforth seafolder-data directory). The */%seafolder – data%/repo.db* database cached a wealth of property information of the libraries added to the desktop clients in the ‘RepoProperty’ table, such as the owners’ email addresses, server’s URLs/IP addresses, folder paths (worktree), and other identifying information as illustrated in Fig. 2; each library is identified by a repo ID.

repo_id	key	value
Filter	Filter	Filter
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	download-head	000
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	token	6d0196a89b0867b193d324cef6cbe0c72f419ddf
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	email	alice70707@gmail.com
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	relay-address	192.168.220.180
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	relay-port	10001
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	relay-id	cc4bf7c1308276069e5eef70a7f1760704629137
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	server-url	http://192.168.220.180
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	worktree	C:/WindowsClientEncrypted
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	remote-head	2e867b460498480d323e505e04a8b9476fade80b
cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479	local-head	3cba016ca1b3cb68b86b3a25ac66206b34759314

Fig. 2 The Seafolder desktop client’s sync and file management metadata

Within the ccnet.conf file in *%Users% < User Profile > ccnet /home/<User Profile>/ccnet/, and /Users/<User Profile>/ccnet/* (henceforth ccnet directory) of the Windows, Ubuntu, and Mac OS desktop clients (respectively) we observed the username (prefixed with ‘USER_NAME =’) and peer ID (prefixed with ‘ID =’) for the desktop clients. The peer ID is the device-unique SHA1 hash derived from the non-HTTP syncing private key [99], and hence a potential source of information for file-to-device mapping.

Examinations of the mobile clients determined that the Seafolder mobile client applications maintained a different data directory structure in comparison to the desktop clients. The Android app held a list of account signatures in the ‘StarredFileCache’ table of */data/data/com.seafolder/seadroid2/databases/data.db* database. The account signature is a unique alphanumerical string used to define the users logged in from the app, which could prove useful for user-to-file mapping e.g., correlating the account signatures with the ‘RepoDir’ and ‘FileCache’ tables (of *data.db* database) to determine the library names and file IDs associated with a user account (respectively), along with the directory paths.

6.3 Authentication and Encryption Metadata

Analysis of the %seafolder-data%/accounts.db database (of the desktop clients) and /data/data/com.seafile.seadroid2/shared_prefs/latest_account.xml file (of the Android client) located records of the server's URLs, email addresses, as well as last-used authentication tokens for the accounts which logged in from the client applications in the 'Account' table. We also observed the server application's version and email address of the currently logged in user in the 'ServerInfo' table of the accounts.db database (or /data/data/com.seafile.seadroid2/databases/accounts.db on the Android client). An inspection of the /.config/Seafile/Seafile Client.config and /Users/<User Profile>/Library/Preferences/com.seafile.Seafile Client.plist files on the Ubuntu and Mac OS desktop clients (respectively) determined that caches of the server's URL(s) could be located in the 'UsedServerAddresses' entry.

Analysis of the Seafile iOS client revealed additional authentication and encryption information such as the plain text login and library encryption passwords in the /private/var/mobile/Applications/<Universally Unique Identifier (UUID) for the Seafile iOS app]/Library/Preferences/group.com.seafile.seafilePro.plist file (see Fig. 3). When the login credentials and encryption password were saved in the web browsers, it was possible to recover the credential information or password using Nirsoft Web Browser Passview [100].

Dictionary		
Dictionary	http://192.168.1.170/alice70707@gmail.com	
AnsiString	token	427ac3a2dea4ca99f5baf50b6284b93ff389cd7c
AnsiString	password	alicepassword
AnsiString	username	alice70707@gmail.com
AnsiString	link	http://192.168.1.170
Integer	usage	300544
Integer	total	-2
AnsiString	VERSION	1.9.1
Dictionary	http://192.168.1.170/alice70707@gmail.com/settings	
Boolean	autoSync	True
AnsiString	DEFAULT-USER	alice70707@gmail.com
Array	ACCOUNTS	
Dictionary		
AnsiString	url	http://192.168.1.170
AnsiString	username	alice70707@gmail.com
AnsiString	DEFAULT-SERVER	http://192.168.1.170
Dictionary	repopassword	encrypteddrive
	28b36023-cb44-4cc6-a049-c154adba1370	

Fig. 3 Analysis of the Seafile iOS client

Further inspections of directory listings located the base64-encoded private key for the non-HTTP syncing protocol [99] in the /%ccnet%/mykey.peer file. We also located the peer ID alongside the base64-encoded public key for the desktop clients in the /%ccnet%/misc/<peer ID>file, following the terms 'peer/' and 'pubkey' respectively. However, only the 'RepoKeys' table of the repo.db database held the keys and initialisation vectors (IVs) for the library encryption keys [95].

6.4 Cloud Transaction Logs

Examinations of the desktop clients indicated the user's last login time in the 'lastVisited' table column of the 'Account' table (of accounts.db database). Alternatively, it was observed that the last login time could be located in the 'LAST_USED' property of the /Users/<User Profile>/Library/Preferences/com.apple.spotlight.plist file on the Mac OS client. Within the 'DeletedRepo' and 'GarbageRepos' tables of the repo.db database we located a list of libraries IDs for the deleted libraries.

Analysis of the commit files (%seafie-data%/storage/commits/<Repo ID>/<first two characters of the commit ID>/<remaining of the commit ID>) revealed the synchronisation history associated with the libraries; each user action created a commit file describing the action (e.g., addition and deletion of files along with the filename references) made to libraries, email address and peer ID of the user who initiated the action, and corresponding timestamp in Unix epoch format in the 'description', 'creator_name', 'creator', and 'ctime' properties, correspondingly. The commit files also held other property information associated with the libraries such as the commit IDs, magic tokens, library names, repo IDs, and other information as shown in Fig. 4 [101]. Since the commit directory %seafie-data%/storage/commits/<Repo ID> is unique to the library, the creation time could reflect the library initiation time. Although analysis of the %ccnet%/logs/seafie.log was not as conclusive as the commit files, we could estimate the accessed and library sync times from the entries "[07/16/15 13:18:38] seaf-daemon.c(504): starting seafie client 4.2.8." and "[07/22/15 10:38:53] repo-mgr.c(2868): All events are processed for repo cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479." (respectively).



```
d8ea787ec871d2d5739bafdb1da191604065c - Notepad
File Edit Format View Help
{"enc_version": 2, "encrypted": "true", "description": "Added \\Enron3111.zip\\ and 5 more files.\n", "root_id": "9efba3dedcedfc44021c8745b3f17225a980c54", "repo_category": null, "commit_id": "78d8ea787ec871d2d5739bafdbf1da191604065c", "key": "b6b0b6fbf4466f68bbfd1abebe7dba2c174fd82a8b55cc8bf362793db6ac41b5746a7e8b912fc33b98f8697aaf188788", "repo_name": "UbuntuClientEncrypted", "parent_id": "14acbf231d316e0ac82486512ffe14ccfebcb5d6", "repo_id": "33c7aa59-d0a9-4102-9cd9-1e2351b4c9d0", "creator_name": "alice7070@gmail.com", "version": 1, "second_parent_id": null, "creator": "1cc3ce7c980efdf7f65019bb231e89692c4db819f", "ctime": 1437567874, "repo_desc": "UbuntuClientEncrypted", "magic": "1f5cb2a265cf714f17ab9d9f2358a649645488a749503651318d54abce7480d6", "no_local_history": 1}
```

Fig. 4 Analysis of the commit files

Examination of the Android client located copies of the dirent caches of the libraries in the 'DirenCache' table of data.db database. The 'content' table column held a list of property information relating to the sync files and directories in the libraries, such as the file/directory names, read write permissions, last modified times, types ('file' or 'folder'), file/directory IDs, and sizes in the 'content' table column in Javascript Object Notation (JSON) format; each record was identified by the repo ID. Figure 5 shows an example of the 'content' table column recovered in our research. The dirent caches could be located in the iOS client in the 'ZDIRECTORY' table of the /private/var/mobile/Containers/

<UUID>/seafolder/seafolder_pro.sqlite database (where UUID stands for Universally Unique Identifier), within the ‘ZCONTENT’ table column.

```
[{
    "name": "Enron3111.docx",
    "permission": "rw",
    "mtime": 1418488303,
    "type": "file",
    "id": "93b6592fbff2eaa6969cedde18377f1b67ad4450",
    "size": 78080
}, {
    "name": "Enron3111.jpg",
    "permission": "rw",
    "mtime": 1418488442,
    "type": "file",
    "id": "2cc06e3170b1f295b73933a67295e228395b971e",
    "size": 287937
},
...
}]
```

Fig. 5 An example of the ‘content’ table column

Within the ‘ZDOWNLOADEDFILE’ table of the seafolder_pro.sqlite database there maintained a list of filenames for the files downloaded to the iOS client, alongside the library and file IDs. Further analysis of the ‘ZSEAFCACHEOBJ’ table located caches of conversations in the ‘ZCONTENT’ table column, which included the email addresses and names for the correspondents, as well as the conversation texts and timestamps of the last conversation threads with the correspondents in JSON format (see Fig. 6 for details); the entry of which could be differentiated from the ‘ZKEY’ table column that referenced ‘CONTACTS’. Moreover, we also recovered caches of the HTTP response bodies for the “list message” web API call in the ZSEAFCACHEOBJ table, which could be discerned from the ‘ZKEY’ table column holding the entry “/api2/user/messages/<Correspondent’s Email Address>/”. The entry held a list of the conversation threads associated with a correspondent, each thread created an opening and closing bracket in the ‘msgs’ entry to house the file attachment information, email address of the user which initiated the conversation thread, conversation texts, and timestamp information as detailed in Fig. 7.

Another database of interest with the Seafile iOS client was the /private/var/mobile/Applications/<UUID> for the SeafileiOSapp/Library/Caches/com.seafolder.seafolder_P-ro/Cache.db. Similarly to the structure of the Safari browser’s Cache.db database, the cached items were stored in the ‘receiver_data’ table column of the ‘cfurl_cac-

Fig. 6 Further analysis of the ‘ZSEAFCACHEOBJ’ table located caches of conversations in the ‘ZCONTENT’

```
{
    "contacts": [
        {
            "msgnum": 0,
            "mtime": 1437568672,
            "lastmsg": "hello alice",
            "email": "fbcchelper@gmail.com",
            "name": "fbcchelper"
        },
        {
            "umsgnum": 0,
            "replynum": 0,
            "groups": [],
            "gmsgnum": 0,
            "newreplies": []
        }
    ],
    "to_email": "fbcchelper@gmail.com",
    "next_page": -1,
    "msgs": [
        {
            "attachments": [],
            "timestamp": 1437568755,
            "from_email": "alice70707@gmail.com",
            "msgid": 5,
            "msg": "Hope you enjoyed them..",
            "nickname": "alice70707"
        },
        {
            "attachments": [],
            "timestamp": 1437568739,
            "from_email": "fbcchelper@gmail.com",
            "msgid": 4,
            "msg": "thankn you for the marvelous stuffs, alice!",
            "nickname": "fbcchelper"
        },
        {
            ...
        }
    ]
}
```

Fig. 7 Recovered caches of the HTTP response bodies for the “list message” web API

he_receiver_data’ table, while the corresponding URLs and timestamps were stored in the ‘cfurl_cache_response’ table, in the ‘request_key’ and ‘time_stamp’ table columns respectively. By issuing the SQL query “SELECT cfurl_cache_receiver

`_data.receiver_data,cfurl_cache_response.request_key,cfurl_cache_response.time_stamp` FROM `cfurl_cache_receiver_data,cfurl_cache_response` WHERE `cfurl_cache_receiver_data.entry_ID=cfurl_cache_response.entry_ID`”, it was possible to recover caches of the HTTP requests/responses for the web/non-web API calls from the Cache.db database, alongside the URLs and associated timestamps as illustrated in Fig. 8.

RecNo	receiver_data	request_key	time_stamp
Click here to define a filter			
1	http://192.168.1.170/api2/repos/		2015-07-22 12:12:42
2	http://192.168.1.170/api2/account/info/		2015-07-22 12:12:42
3	http://192.168.1.170/api2/groupandcontacts/		2015-07-22 12:12:42
4	http://192.168.1.170/api2/html/events/		2015-07-22 12:12:52
5	http://192.168.1.170/media/css/mobile.css?t=1369028461		2015-07-22 12:12:52
6	http://192.168.1.170/media/js/jq.min.js?t=1369028460		2015-07-22 12:12:52
7	http://192.168.1.170/api2/avatars/user/alice7@gmail.com/resized/80/		2015-07-22 12:13:02
8	http://192.168.1.170/media/avatars/7/7ba0ddcf2b4084e4d357cf2fb08c68/resized/80/fcabcced6616ee7675c73632feba9a.png		2015-07-22 12:13:02
9	http://192.168.1.170/api2/repos/f378e650-ab41-7d0-b63c-60ec4dcd175/dr/?p=/		2015-07-22 12:17:54
10	http://192.168.1.170/api2/repos/f378e650-ab41-7d0-b63c-60ec4dcd175/upload-link/		2015-07-22 12:18:53
11	http://192.168.1.170/api2/repos/f378e650-ab41-7d0-b63c-60ec4dcd175/dr/?p=/seafile-tutorial.doc		2015-07-22 12:20:00
12	http://192.168.1.170/seafhttp/files/3/bf/114-0e30-4f99-9cab-5a6de7d1c005/seafile-tutorial.doc		2015-07-22 12:20:02
13	http://192.168.1.170/api2/starredfiles/repo_id=f378e650-ab41-7d0-b63c-60ec4dcd175p=/seafile-tutorial.doc		2015-07-22 12:20:36
14	http://192.168.1.170/api2/repos/ffac3dff-f9f6-498d-bac4-6feacec805738/dr/?p=/		2015-07-22 12:26:53
15	http://192.168.1.170/api2/repos/ffac3dff-f9f6-498d-bac4-6feacec805738/file/?p=/Enron3111.jpg		2015-07-22 12:27:02
16	http://192.168.1.170/seafhttp/files/1/bb/d9/a-0d8f-4c3c-bec1-d591/f30c180/Enron3111.jpg		2015-07-22 12:27:02
17	http://192.168.1.170/api2/repos/ffac3dff-f9f6-498d-bac4-6feacec805738/file/?p=/Enron3111.trf		2015-07-22 12:27:02
18	http://192.168.1.170/seafhttp/files/7/52/2d5-aef4-fb0-899a-8085-4feafe230/Enron3111.trf		2015-07-22 12:27:02
19	http://192.168.1.170/api2/repos/ffac3dff-f9f6-498d-bac4-6feacec805738/file/?p=/Enron3111.bt		2015-07-22 12:27:02
20	http://192.168.1.170/seafhttp/files/f72692ef-83cf-4c7c-8784-be6a8a7c621b/Enron3111.txt		2015-07-22 12:27:02
21	http://192.168.1.170/api2/repos/ffac3dff-f9f6-498d-bac4-6feacec805738/file/?p=/Enron3111.zip		2015-07-22 12:27:12
22	http://192.168.1.170/seafhttp/files/f6d54589-192a-4e6a-a6c0-7e0fbfe5fc/Enron3111.zip		2015-07-22 12:27:12
23	http://192.168.1.170/api2/repos/ffac3dff-f9f6-498d-bac4-6feacec805738/file/?p=/Enron3111.docx		2015-07-22 12:27:12
24	http://192.168.1.170/seafhttp/files/0/b4/f3/e8/d-38-46-19/b7e/d095fde347f/Enron3111.docx		2015-07-22 12:27:12
25	http://192.168.1.170/api2/repos/ffac3dff-f9f6-498d-bac4-6feacec805738/upload-link/		2015-07-22 12:27:31
26	http://192.168.1.170/api2/repos/ffac3dff-f9f6-498d-bac4-6feacec805738/dr/?p=/&oid=32c617e5b6c751feff7ecdb2fd981b976929b705		2015-07-22 12:31:27
27	http://192.168.1.170/api2/repos/2/b36023/cb44-4cc0-a049-c154ada8a1370/dr/?p=/		2015-07-22 12:32:26
28	http://192.168.1.170/api2/repos/2/b36023/cb44-4cc0-a049-c154ada8a1370/dr/?p=/&oid=c5f076c0ba7fbc03f7eec2bc235e6beb758e4f1		2015-07-22 12:32:32
29	http://192.168.1.170/api2/repos/2/b36023/cb44-4cc0-a049-c154ada8a1370/upload-link/		2015-07-22 12:32:52
30	http://192.168.1.170/api2/usermsgs/fbcchelper@gmail.com/		2015-07-22 12:38:02
31	http://192.168.1.170/api2/avatars/user/fbcchelper@gmail.com/resized/80/		2015-07-22 12:38:02

Fig. 8 The possibility to recover caches of the HTTP requests/responses for the web/non-web API calls

In particular, when the file download/viewing was taken place, we observed the URLs “`http://<IP Address or Domain Name>/seafhttp/files/<AccessToke n>/<Filename>`” and “`http://<IP Address or Domain Name>/api2/repos/<Repo ID>/file/?p=<Filename>`”). Meanwhile, the URL “`http://<IP Address or Domain Name>/api2/usermsgs/<Correspondent’s Email Address>`” was seen in regard to the conversation. Details of the web API calls can be found at http://manual.seafile.com/develop/web_api.html#list-libraries.

Further analysis of the Cache.db and seafile_pro.sqlite databases recovered copies of the HTTP response bodies for the list libraries web API call [102] from the ‘receiver_data’ table column (of the ‘cfurl_cache_receiver_data’ table) and ‘ZCONTENT’ table column (of the ZSEAFCACHEOBJ table) of the databases respectively, with the ‘ZKEY’ table column referencing the value ‘REPOS’ in the latter (see Fig. 9). A closer examination of the response bodies indicated the permission and encryption information, last modified times, owners’ email

addresses, as well as repo IDs, names, sizes, and other relevant details in JSON format as detailed in Fig. 10.

Z_PK	Z_ENT	Z_OPT	ZTIMESTAMP	ZCONTENT	ZKEY	ZURL	ZUSERNAME
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	3	4	459259955.324943	[{"permission": "rw", "encrypted": true, "enc_version": 2, "mtime_relative": "<time datetime=\"2015-07-22T20:33:33\" is=\"relative-time\" title=\"Wed, 22 Jul 2015 20:33:33 +0800\">9 minutes ago</time>", "mtime": 1437568784, "owner": "alice70707@gmail.com", "root": "9efba5dedcedfc44021c8745b3f17225a980c54", "id": "33c7aa59-d0a9-4102-9cd9-1e2351b4c9d0", "size": 474825, "magic": "1f5ceb7a265cf714f17ab9d9f2358a649645488a749503651318d54abce7480d6", "name": "UbuntuClientEncrypted", "type": "repo", "virtual": false, "random_key": "b6b0b6fb4466f68bfda1bebe7dba2c174fd82a8b55cc8bf362793db6ac41b5746a7e8b912fcc33b98f8697aaf188788", "desc": "UbuntuClientEncrypted", "size_formatted": "463.7 KB"}]	REPOS	http://192.168.1.170	alice70707@gmail.com
2	2	3	459259956.432345	[{"contacts": [{"respnoun": 0, "ntime": 1437566872, "lastmsg": "Hello Alice", "email": "fbchelper@gmail.com", "name": "fbchelper"}, {"uns..."]}, {"to_email": "fbchelper@gmail.com", "next_page": "-1", "msg...": [{"attachments": [], "timestamp": 1437568755, "from_email": "Alice...": "fbchelper@gmail.com/"}]}]	CONTACTS	http://192.168.1.170	alice70707@gmail.com
3	3	7	459261475.483739		/api2/usermsgs/	http://192.168.1.170	alice70707@gmail.com/

Fig. 9 Further analysis of the Cache.db and seafile_pro.sqlite databases

In addition to the client application generated artefacts mentioned above, we were able to determine the load count, last run time, and other associated timestamps for the Windows client application in the CCNET.EXE(pf and SEAFILE-APPLET.EXE(pf prefetch files. Further inspections of the Windows desktop client's directory listings located shortcuts for %ProgramFiles% Seafile bin seafile-applet.exe at %ProgramData% Microsoft Windows Start Menu Programs Seafile Start Seafile .lnk and %Desktop%/Seafile.lnk. The link files held the original file paths, file sizes, as well as timestamp records that reflected the install and last accessed times. Analysis of the Windows registry revealed the install date alongside the client application's name, version, install path, and other relevant information in the HKEY_LOCAL_MACHINE SOFTWARE

```
[{
    "permission": "rw",
    "encrypted": true,
    "enc_version": 2,
    "mtime_relative": "<time datetime=\"2015-07-22T20:24:34\" is=\"relative-time\" title=\"Wed, 22 Jul 2015 20:24:34 +0800\">9 minutes ago</time>",
    "mtime": 1437568784,
    "owner": "alice70707@gmail.com",
    "root": "9efba5dedcedfc44021c8745b3f17225a980c54",
    "id": "33c7aa59-d0a9-4102-9cd9-1e2351b4c9d0",
    "size": 474825,
    "magic": "1f5ceb7a265cf714f17ab9d9f2358a649645488a749503651318d54abce7480d6",
    "name": "UbuntuClientEncrypted",
    "type": "repo",
    "virtual": false,
    "random_key": "b6b0b6fb4466f68bfda1bebe7dba2c174fd82a8b55cc8bf362793db6ac41b5746a7e8b912fcc33b98f8697aaf188788",
    "desc": "UbuntuClientEncrypted",
    "size_formatted": "463.7 KB"
},
...
}]
```

Fig. 10 A closer examination of the response bodies

Microsoft Windows CurrentVersion Installer UserData <SID> Products <GUID> InstallProperties and HKEY_LOCAL_MACHINE SOFTWARE

Wow6432Node Microsoft Windows CurrentVersion Uninstall {<GUID>} (where 'SID' and 'GUID' stand for Security Identifier and Globally Unique Identifier) registry branches.

A search for the term ‘Seafile’ indicated the timestamp for the client application’s installation and uninstallation in the Windows’ Application.evtx event log; Fig. 11 shows an example of the uninstall event. A search for the same term in the Ubuntu OS’s system and dpkg logs only revealed the client application’s installation time, from the entries “Jul 22 04:29:17 ubuntu AptDaemon.Worker: INFO: Installing local package file: /home/suspectpc/Desktop/linux seafile client_4.2.5_amd64.deb” and “2015-07-22 04:33:35 status installed seafile:amd64 4.2.5” respectively. Analysis of the Ubuntu OS’s Zeitgeist log revealed the current (in the ‘subj_url’ and ‘subj_current_uri’ table columns) and original (in the ‘subj_origin_uri’ table column; if any) directory paths for the libraries/sync files and client application in the ‘event_view’ table of the ./local/share/zeitgeist/activity.sqlite database. The artefacts could prove useful when seeking to determine the directory paths in cases when the files have been moved. The log file also revealed the types of actions made to the directories/files of relevance (e.g., Access, Create, Move, Leave, and Delete; each action created an entry in the database) in the ‘interpretation’ table column, alongside the timestamps associated with the actions. Analysing the Gnome’s /home/<User Profile>/local/share/recently-used.xbel log located caches of the directory paths as well as the last added, modified, and visited times for the client application and sync files in the ‘bookmark visited’ property, indicative that the files were recently accessed using applications running in the Gnome desktop [103]. The ‘bookmark visited’ property also held the names of the applications used to open the files, and a count of the number of times the applications were used to access the files (in the ‘bookmark:applications’ property). Figure 12 shows an example of the recently-used.xbel log entry for the Ubuntu client application.

Application Number of events: 544					
Level	Date and Time	Source	Event ID	Task C	^
Information	7/16/2015 1:17:34 PM	System-Restore	8300	None	
Information	7/16/2015 1:17:31 PM	MsilInstaller	1033	None	
Information	7/16/2015 1:17:31 PM	MsilInstaller	11707	None	
Information	7/16/2015 1:17:27 PM	RestartManager	10001	None	
Information	7/16/2015 1:17:27 PM	MsilInstaller	1042	None	
Information	7/16/2015 1:17:25 PM	RestartManager	10000	None	
Information	7/16/2015 1:17:21 PM	System Restore	8194	None	
Information	7/16/2015 1:17:07 PM	MsilInstaller	1040	None	
Information	7/16/2015 1:16:17 PM	Windows Error Reporting	1001	None	
Error	7/16/2015 1:16:11 PM	Application Error	1000	(100)	▼

Event 1033, MsilInstaller	X
<input checked="" type="button"/> General <input type="button"/> Details	
Windows Installer installed the product. Product Name: Seafile 4.2.8. Product Version: 4.2.8. Product Language: 1033. Manufacturer: HaiWenHuZhi Ltd.. Installation success or error status: 0.	
Log Name: Application	

Fig. 11 An example of the uninstall event

```

<?xml version="1.0" encoding="UTF-8"?>
<xbel xmlns:mime="http://www.freedesktop.org/standards/shared-mime-info" xmlns:bookmark="http://www.freedesktop.org/standards/desktop-bookmarks">
  <xbel visited="2015-07-22T11:26:05.882584Z" modified="2015-07-22T11:26:05Z" added="2015-07-22T11:26:05Z" href="file:///home/suspectpc/Desktop/linux%20seafile%20client_4.2.5_.and64.deb">
    <info>
      <metadata owner="http://freedesktop.org">
        <mime:mime-type type="application/x-deb"/>
        <bookmark:applications>
          <bookmark:application modified="2015-07-22T11:26:05Z" count="1" exec="/usr/bin/software-center %u" name="nautilus"/>
        </bookmark:applications>
      </metadata>
    </info>
  </bookmark>
</xbel>

```

Fig. 12 An example of the recently-used.xbel log entry for the Ubuntu client application

When logging in and out using the Seafile web application, we observed the URLs “<http://<IP address or domain name>/accounts/login/>” and “<http://<IP address or domain name>/accounts/logout/>” in the web browsing history. Accessing a library produced the URL “<http://<IP address or domain name>/#mylibs/lib/<Repo ID>>”, while viewing and downloading a file created the URLs “<http://<IP address or domain name>/lib/<RepoID>/file/<Filename>>” and “<http://<IP address or domain name>/repo/<RepoID>/<AccessToken>/download/?p=/<Filename>>”, respectively. When the conversations took place, we were able to determine the correspondent’s email address from the URL “<http://<IP address or domain name>/user/<Correspondent's Email Address>/msgs/>”. The web browsing history also held the last access timestamps of the URLs.

6.5 Data Storage

The libraries downloaded were located in %Users% <UserProfile> Seafile, /home/<User Profile>/Seafile/, /Users/<User Profile>/Seafile/, and /home/Seafile/<User'sEmailAddress>(<Server'sURL>) of the Windows, Ubuntu, Mac OS, and Android clients respectively, by default. The libraries remained even after being unsynced from the client devices. Although the iOS client did not download the libraries like as the client and Android clients, we could recover copies of the viewed and uploaded files in /private/var/mobile/Containers/<UUID>/seafile/objects/ and /private/var/mobile/Containers/<UUID>/seafile/uploads/, respectively. Further inspection of the former determined that were no were no filename and extension information associated with the downloaded files, and hence a manual analysis of the file header is necessary to identify the file types. Additionally, it was observed that the clear texts copies of the synced files could be recovered from the ‘receiver_data’ table column of the ‘cfurl_cache_receiver_data’ table (of the Cache.db database).

Within %seafile-data%/avatars/ there were copies of the avatars used by the user(s) logged in from the desktop clients. In the iOS device, the ‘avatars’ directory could be located at /private/var/mobile/Containers/<UUID>/seafile/avatars. Analysis of the thumbnail cache only located thumbnail images for the synced files in %AppData% Local Microsoft Windows Explorer /home/suspectpc/.cache/

`thumbnails/large/`, and `/home/Seafile/cache/thumbnail/` directory of the Windows, Ubuntu, and Android clients (respectively), indicating thumbnail caches as a potential evidence source for the sync files. Figure 13 illustrates that the thumbnail cache of which could be differentiated from the ‘`jtEXtThumb::URI file:`’ that referenced the library’s directory path in the Ubuntu client.



Fig. 13 Analysis of the thumbnail cache

6.6 Network Analysis

Unsurprisingly, analysis of the network traffic revealed the IP addresses of the clients and servers as well as timestamp information associated with the cloud transactions. We also recovered copies of the clear text HTTP headers for the web API/non-web-API calls for user actions such as logging in and off, conversations, and uploading, downloading, deleting, sharing and viewing libraries and files. A closer inspection of the HTTP request and response bodies revealed the login credentials, user and correspondents’ email addresses and account names, conversation texts, share links (if any), and other library properties in JSON and Hyper-text Markup Language (HTML) formats. Table 5 shows examples of the HTTP request/response bodies of forensic interest. The remaining of the web API calls are documented in the Seafile manual (http://manual.seafile.com/develop/web_api.html#api-basics) and hence, are not discussed in our research. Reconstruction of the network traffic resulted in the recovery of the complete web application root directory, which included copies of the user’s avatar, transferred libraries/files and caches of conversations from the HTTP traffic intact.

6.7 Memory Analysis

Examinations of the running processes using the ‘`pslist`’ function of Volatility indicated the process names, process identifiers (PID), parent process identifiers (PPID), as well as process initiation and termination times. The process names of which could be differentiated from ‘`ccnet`’, ‘`seaf-daemon`’, and ‘`seafle-applet`’. Examinations of the memory dumps using the ‘`netscan`’ or ‘`netstat`’ function of Volatility recovered the network information associated with the processes such

Table 5 HTTP request and response bodies of forensic interest

Description	Request type	Example of request body	Example of response body
Obtaining authorisation token during the client application login.	POST /api/2/auth-token/ HTTP/1.1	username=alice70707%40gmail.com password=alicepassword&platform_version= &client_version=4.2.8&&device_id=9fcf97e60 09f5caa0b39-cdf38475ba52542c51d&&device_ name=Windows%20Client&&platform=windo ws [{"token": "763306a14b8019efc120b5f98e 6a31fd8e889137"}]	
Logging in using the web application.	POST /accounts/login/ HTTP/1.1	crfmiddlewaretoken=Ka6G0si5Ncp621 90AZKf4Ub0dvnEjAA&username=inter netexplorer@gmail.com&password=inter netexplorer&next=%2F&remember_me= on	Nil
Obtaining the commit file of a library.	GET /seafhttp/repo/Repo ID/commit/Commit ID HTTP/1.1	Nil	Clear text content of the commit file
Obtaining the commit file of a library.	PUT /seafhttp/repo/Repo ID/commit/Commit ID HTTP/1.1	Clear text content of the commit file	Nil
Downloading a file.	GET /repo/Repo ID/download/?p=Filename HTTP/1.1	Nil	Nil
Viewing a file.	GET/seafhttp/files/22f2d 3c2-ed49-4e78-80ff-1307 24aa67fa/FilenameHTTP /1.1	Nil	Viewed file
Obtaining a file share link.	GET/share/ajax/get-down load-link?repo_id=Repo D&p=Filename&type=f &_=1456107943936	Nil	Nil
Generating a public file share link.	POST/share/ajax/get-down load-link/HTTP/1.1	use_passwd=0&repo_id=7e5e6398-3fec-4 ad1-838b-984cb8fe2a49&p=%2FEmron31 11.pdf&type=f	{"token": "f279b486a4", "download_link": "http://192.168.1.170/f279b486a4/"} }

Private file sharing.	POST /share/link/send/ HTTP/1.1 email=internetexplorer%40gmail.com&extra_=msg=sharedfileforyou&file_shared_link=http%3A%2F%2F192.168.1.170%2F%2F192.168.1.	Nil
Creating a private share link.	POST /share/ajax/private-share-file/ HTTP/1.1	<pre>{ "name": "Entron3111.pdf&file_shared_type=f", "repo_id": "7e5e66398-3fec-4ad1-838b-984cb8fe2a49&path=%2FEntron3111.pdf&emails=fbchelper@gmail.com" }</pre>
Creating a library.	POST /api2/repos/?from=web HTTP/1.1	<pre>{ "name": "IEClientEncrypted", "encrypted": true, "passwd1": "encrypeddrive", "passwd2": "encrypeddrive", "d2": "encrypeddrive", "owner": { "id": null, "desc": "...", "mtime": 0, "ctime_relative": "...", "owner": "...", "nickname": "..." } }</pre>
		<pre>{ "repo_size": 0, "repo_size_formatted": "0 bytes", "repo_id": "5172fdcc6-7ea2-4c83-a498-bc02bf05364", "magic": "4cd40342a11930988aaffcb376a85a504e9d5ff6014c406f01677aocc267620", "encrypted": 1, "repo_desc": "...", "random_key": "9a47bcef101069d3cc80ea a5725b322cebcc9f44f7956f5a606a2587 bebf8bb0fc3401ef1ee59680e5f6dad497ccb1", "relay_id": "cc4b17c1308276069e5eef70a7f1760704629137", "enc_version": 2, "mtime_relative": "...", "<time datetime='2015-07-22T16:44:38' is='relative-time' title='Wed, 22 Jul 2015 16:44:38 +0800'>1 second ago</time>", "relay_addr": "192.168.20.180", "token": "...", "repo_version": 1, "relay_port": "10001", "mime": "1437554678, \"email\": \"internetexplorer@gmail.com\", \"repo_name\": \"IEClientEncrypted\"" }</pre>

(continued)

Table 5 (continued)

Description	Request type	Example of request body	Example of response body
Decrypting a library.	POST /repo/set_password/HTTP/1.1	repo_id=5172fdc6-7ea2-4e83-a498-bc02bf0b3e4&password=encrypteddrive&username=internetexplorer%40gmail.com	{"success": true}
Uploading a file to a library.	POST /seafhttp/upload-aj/8d8011c5-da2f4a35-8221-fea4!6c81abcHTTP/1.1	Content-Disposition: form-data; name="parent_dir"/Content-Disposition: form-data; name="file"; filename="Enron3111.docx"	[{"name": "Enron3111.docx", "id": "8a24ac6aa6d7d6d1171c5119b3a20b28ba7353d", "size": 78080}]
Listing the libraries in a group.	GET/api2/groups/<GroupID>/repos/?from=web HTTP/1.1	File content...	{"repos": [{"owner": "alice70707", "permission": "rw", "encrypted": false, "intime_relative": "<time", "datetime": "2015-07-21T01:41:42", "is": "relative-time", "title": "Tue, 21 Jul 2015 01:41:42 +0800", "gt": "1 day ago</time>", "mimeType": "1437414102, "owner": "alice70707@gmail.com", "id": "1deefc98-1319-449e-a005-dbad8b2added", "size": 0, "name": "GroupOne Library", "shareFromMe": false, "desc": "", "sizeFormatted": "0 bytes"}], "is_staff": false, ...}

Creating a library in a group.	POST /api2/groups/<Group ID>/repos/ HTTP/1.1	<pre>{ "name": "InternetExplorerGroupLibrary", "encrypted": false, "passwd1": "...", "passwd2": "...", "passwd": "...", "permisson": "rw", "id": null, "desc": "...", "mtime": "0", "owner": "...", "owner_relative": "...", "owner_nickname": "..." }</pre>	<pre>{ "owner.nickname": "internetexplorer", "rw": "encrypted": false, "mtime.relative": "time", "datetime": "2015-07-22T17:31:59", "is.relative.time": "Wed, 22 Jul 2015 17:31:59 +0800", "ago.time": "1 second ago", "intime": "1437557519", "owner": "internetexplorer@gmail.com", "id": "1f439351-5f83-4c90-950a-ab294b6231f", "size": 0, "name": "InternetExplorerGroupLibrary", "share.from.me": true, "dese": "", "size.formatted": "0 bytes" }</pre>	
Sending a group message.	POST /group/<Group ID>/discussion/add/ HTTP/1.1	<pre>message=InternetExplorer+has+just+added+a+new+library!</pre>	<pre>{ "msg.id": 1, "msg.con": "n<p>msg-con <InternetExplorer has just added a new library!</p> n n" }</pre>	
Listing the messages in a group.	GET /group/<Group ID>/discuss/ HTTP/1.1	Nil	HTML coding for the group discussion window	
Sending a private message.	POST/massage/message_send?from=userHTTP1.1	mass_msg=hello+miss+alice&mass_email=alice70707%40gmail.com	Complete web document for a conversation thread, including the conversation texts, user and correspondent's email address and account name, and the corresponding timestamp information.	
Obtaining list of messages associated with a correspondent.	GET/user/user' semail/messages/HTTP P/1.1	Nil	Complete web document for a conversation windows, including the conversation texts, user and correspondent's email address and account name, and the corresponding timestamp information.	

as the host and server's IP addresses, port numbers, socket states, and protocols (see Fig. 14), providing an alternative method for recovery of the network information.

Offset(?)	Proto	Local Address	Foreign Address	Status	Pid	Owner	Created
0x7b603d10	TCPv4	127.0.0.1:49469	127.0.0.1:49468	CLOSED	2596	seaf-daemon.exe	
0x7b603f80	TCPv4	192.168.220.165:49449	192.168.220.180:80	ESTABLISHED	2312	seafile-applet	
0x7b703f70	UDPv4	0.0.0.0:0	=*:	LISTENING	2528	cernet.exe	2015-07-22 02:33:53 UTC+0000
0x7b723590	UDPv4	0.0.0.0:0	=*:	LISTENING	2528	cernet.exe	2015-07-22 02:33:53 UTC+0000
0x7c4b20e0	TCPv4	127.0.0.1:3419	0.0.0.0:0	LISTENING	2528	cernet.exe	
0x7c4b2a470	TCPv4	127.0.0.1:49497	127.0.0.1:49196	ESTABLISHED	2528	cernet.exe	
0x7c4b2d70	TCPv4	127.0.0.1:49190	127.0.0.1:49191	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7d3920	TCPv4	127.0.0.1:49196	127.0.0.1:49197	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7d4010	TCPv4	127.0.0.1:3419	127.0.0.1:49171	ESTABLISHED	2528	cernet.exe	
0x7b7d4010	TCPv4	127.0.0.1:3419	127.0.0.1:49171	ESTABLISHED	2312	seafile-applet	
0x7b7d4ec0	TCPv4	127.0.0.1:49170	127.0.0.1:3419	ESTABLISHED	2312	seafile-applet	
0x7b7d7fa70	TCPv4	127.0.0.1:3419	127.0.0.1:49169	ESTABLISHED	2528	cernet.exe	
0x7b7d8030	TCPv4	127.0.0.1:49169	127.0.0.1:49170	ESTABLISHED	2312	seafile-applet	
0x7b7d81010	TCPv4	127.0.0.1:49194	127.0.0.1:49195	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7d83650	TCPv4	127.0.0.1:3419	127.0.0.1:49179	ESTABLISHED	2528	cernet.exe	
0x7b7d83650	TCPv4	127.0.0.1:3419	127.0.0.1:49179	ESTABLISHED	2312	seafile-applet	
0x7b7d91010	TCPv4	127.0.0.1:49188	127.0.0.1:49189	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7d97d10	TCPv4	127.0.0.1:49193	127.0.0.1:49194	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7da7760	TCPv4	127.0.0.1:3419	127.0.0.1:49185	ESTABLISHED	2528	cernet.exe	
0x7b7db1010	TCPv4	127.0.0.1:3419	127.0.0.1:49179	ESTABLISHED	2312	seafile-applet	
0x7b7db3000	TCPv4	127.0.0.1:49179	127.0.0.1:3419	ESTABLISHED	2312	seafile-applet	
0x7b7db1960	TCPv4	127.0.0.1:49185	127.0.0.1:3419	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7db770	TCPv4	127.0.0.1:49189	127.0.0.1:49188	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7db7f00	TCPv4	127.0.0.1:3419	127.0.0.1:49182	ESTABLISHED	2528	cernet.exe	
0x7b7dc2010	TCPv4	127.0.0.1:3419	127.0.0.1:49181	ESTABLISHED	2528	cernet.exe	
0x7b7dc2010	TCPv4	127.0.0.1:49181	127.0.0.1:49184	ESTABLISHED	2528	cernet.exe	
0x7b7dc2010	TCPv4	127.0.0.1:49192	127.0.0.1:49193	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7de0470	TCPv4	127.0.0.1:49193	127.0.0.1:49192	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7de1ae0	TCPv4	127.0.0.1:49191	127.0.0.1:49190	ESTABLISHED	2596	seaf-daemon.exe	
0x7b7de1e0	TCPv4	127.0.0.1:49190	127.0.0.1:49160	CLOSED	2596	seaf-daemon.exe	
0x7c029260	TCPv4	127.0.0.1:49181	127.0.0.1:3419	ESTABLISHED	2596	seaf-daemon.exe	
0x7c045560	TCPv4	127.0.0.1:49175	127.0.0.1:3419	ESTABLISHED	2312	seafile-applet	
0x7c045560	TCPv4	127.0.0.1:49179	127.0.0.1:49168	ESTABLISHED	2312	seafile-applet	
0x7c049040	TCPv4	127.0.0.1:49184	127.0.0.1:49183	ESTABLISHED	2596	seaf-daemon.exe	
0x7c189d10	TCPv4	192.168.220.165:49320	192.168.220.180:80	ESTABLISHED	2596	seaf-daemon.exe	
0x7c189d10	TCPv4	192.168.220.165:49320	192.168.220.180:80	ESTABLISHED	2312	seafile-applet	
0x7c1b1e010	TCPv4	127.0.0.1:3419	127.0.0.1:49170	ESTABLISHED	2528	cernet.exe	
0x7c1ca070	TCPv4	127.0.0.1:49468	127.0.0.1:49469	CLOSED	2596	seaf-daemon.exe	
0x7c1ca070	TCPv4	127.0.0.1:49468	127.0.0.1:49469	ESTABLISHED	2312	seafile-applet	
0x7c317690	TCPv4	127.0.0.1:49182	127.0.0.1:3419	ESTABLISHED	2596	seaf-daemon.exe	
0x7c9bdc10	TCPv4	127.0.0.1:49166	127.0.0.1:49167	ESTABLISHED	2528	cernet.exe	
0x7cc7c9c0	TCPv4	127.0.0.1:49167	127.0.0.1:49166	ESTABLISHED	2528	cernet.exe	
0x7cc98940	TCPv4	192.168.220.165:49308	192.168.220.180:80	ESTABLISHED	2390	seaf-daemon.exe	

Fig. 14 Examinations of the running processes using the ‘pslist’

A search for the user's email address revealed the login password following the text ‘password=’ in plain text (see Fig. 15). The text appeared to be the remnants from the HTTP request body for the web API call for authorisation token creation. We also recovered copies of the commit files, databases (e.g., repo.db and accounts.db), log files, and other HTTP request/response bodies containing the email address. By searching for the library names, we recovered the cleartext encryption passwords prefixed with the text ‘passwd=’, which seems to be remnants of the HTTP request body for the library creation's web API call (see Fig. 16). A Yarascan search determined that the texts were associated with the ‘seafile-applet’ process, confirming that the texts were remnants from the client applications. Figure 17 shows an example of the Yarascan output for the login password. Data carving of the memory dump only recovered copies of the sample files, repo.db and accounts.db database files, and icon images used by the client application. As for the web application, we determined that it was possible to recover copies of the web caches from the memory space of the web browsers' processes in plain text.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
31568140	D0	F4	63	02	00	00	00	00	D0	DD	79	02	00	00	00	00	Đôc.....Đýy.....
31568150	50	8B	54	00	00	00	00	00	00	00	00	00	00	00	00	00	P<T.....
31568160	00	00	00	00	00	00	00	00	70	1D	63	02	00	00	00	00p.c.....
31568170	00	00	00	00	00	00	00	00	09	00	01	00	00	00	00	00
31568180	00	00	00	00	00	00	00	00	11	01	00	00	00	00	00	00
31568190	02	00	00	00	E0	00	00	00	BB	00	00	00	02	7F	00	00à...>....
315681A0	A8	E1	7D	02	00	00	00	00	70	61	73	73	77	6F	72	64	"á)...password
315681B0	3D	61	6C	69	63	65	70	61	73	73	77	6F	72	64	26	64	=alicepassword&d
315681C0	65	76	69	63	65	5F	6E	61	6D	65	3D	55	62	75	6E	74	vie_name=Ubuntu
315681D0	75	25	32	30	63	6C	69	65	6E	74	26	63	6C	69	65	6E	u%20client&client_
315681E0	74	5F	76	65	72	73	69	6F	6E	3D	34	2E	32	2E	35	26	_version=4.2.5&
315681F0	70	6C	61	74	66	6F	72	6D	3D	6C	69	6E	75	78	26	75	platform=linux&user
31568200	73	65	72	6E	61	6D	65	3D	61	6C	69	63	65	37	30	37	name=alice70707%40gmail.com&p
31568210	30	37	25	34	30	67	6D	61	69	6C	2E	63	6F	6D	26	70	latform_version=
31568220	6C	61	74	66	6F	72	6D	5F	76	65	72	73	69	6F	6E	3D	&device_id=1cc3c
31568230	26	64	65	76	69	63	65	5F	69	64	3D	31	63	63	33	63	e7c980efd7f650199bb231e89692c4db8
31568240	65	37	63	39	38	30	65	66	64	37	66	36	35	30	31	39	19f.lobal_eBusine
31568250	62	62	32	33	31	65	38	39	36	39	32	63	34	64	62	38	19f.lobal_eBusine
31568260	31	39	66	00	6F	62	61	6C	5F	65	42	75	73	69	6E	65	19f.lobal_eBusine

Fig. 15 A search for the user's email address

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
30138450	00	00	00	00	00	00	00	00	48	14	4C	00	80	60	00	00H.L.€'..
30138460	4C	14	4C	00	80	60	00	00	00	00	00	00	00	00	00	00	L.L.€'.....
30138470	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30138480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30138490	AD	BE	FA	B1	B8	22	DD	BA	90	BF	2D	00	80	60	00	00	..%út,"Ý°.č-€'..
301384A0	00	00	00	00	00	00	00	00	00	00	00	00	00	60	20	00	00
301384B0	00	00	00	00	00	00	00	00	00	00	01	00	00	00	01	00`..
301384C0	00	00	00	00	00	00	00	00	B8	14	4C	00	80	60	00	00	,L.€'..
301384D0	BC	14	4C	00	80	60	00	00	00	00	00	00	00	00	00	00	4.L.€'..
301384E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
301384F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30138500	AD	BE	FA	B1	B8	22	DD	BA	70	77	2C	00	80	60	00	00	..%út,"Ý°pw..€'..
30138510	18	00	00	00	00	00	00	00	70	61	73	73	77	64	3D	65passwde=
30138520	6E	63	72	79	70	74	65	64	64	72	69	76	65	26	64	65	ncrypteddrive&de
30138530	73	63	3D	4D	61	63	43	6C	69	65	6E	74	53	79	6E	63	sc=MacClientSync
30138540	45	6E	63	72	79	70	74	65	64	26	6E	61	6D	65	3D	4D	Encrypted&name=Ma
30138550	61	63	43	6C	69	65	6E	74	53	79	6E	63	45	6E	63	72	cClientSyncEncr
30138560	79	70	74	65	64	00	00	00	00	00	00	00	00	00	00	00	ypted.....

Fig. 16 Searching for the library names

```
Task: seafire-applet pid 11626 rule r1 addr 0x27de1b1
0x027de1b1 61 6c 69 63 65 70 61 73 73 77 6f 72 64 26 64 65
0x027de1c1 76 69 63 65 5f 6e 61 6d 65 3d 55 62 75 6e 74 75
0x027de1d1 25 32 30 63 6c 69 65 66 74 26 63 6c 69 65 6e 74
0x027de1e1 5f 76 65 72 73 69 6f 6e 3d 34 2e 32 2e 35 26 70
0x027de1f1 6c 61 74 66 6f 72 6d 3d 6c 69 6e 75 78 26 75 73
0x027de201 65 72 6e 61 6d 65 3d 61 6c 69 63 65 37 30 37 30
0x027de211 37 25 34 30 67 6d 61 69 6c 2e 63 6f 6d 26 70 6c
0x027de221 61 74 66 6f 72 6d 5f 76 65 72 73 69 6f 6e 3d 26
0x027de231 64 65 76 69 63 65 5f 69 64 3d 31 63 63 33 63 65
0x027de241 37 63 39 38 30 65 66 64 37 66 36 35 30 31 39 62
0x027de251 62 32 33 31 65 38 39 36 39 32 63 34 64 62 38 31
0x027de261 39 66 00 6f 62 61 6c 5f 65 42 75 73 69 6e 65 73
0x027de271 73 5f 43 41 2e 70 65 6d 00 ff ff ff ff ff ff 0a 8b
0x027de281 08 0a 00 00 00 00 00 db 54 2f 07 97 48 c9 26 30
0x027de291 00 0a 46 69 72 6d 61 21 01 00 00 00 00 00 00 00 70
0x027de2a1 52 e5 95 02 7f 00 00 e0 e0 7d 02 00 00 00 00 00 10 R.....}....
```

Fig. 17 An example of the Yarascan output for the login password

6.8 Server Forensics (*Divide and Conquer*)

In this section, we identify, preserve, collect, and analyse the artefacts from the Seafile server.

6.8.1 Evidence Source Identification

Recalling the synchronisation and file management as well as authentication and encryption metadata recovered from the Seafile clients, the server's name and logical address could be located in the network captures and commit, ccnet.config, latest_account.xml, group.com.seafile.seafilePro.plist and seafile.log files. For the purposes of this research, we assume that the physical servers that are hosting the Seafile instances are located within the jurisdiction of the Law Enforcement Agency (LEA).

6.8.2 Collection and Preservation of Evidence

The servers were disconnected from the network to prevent further adulteration of the evidence sources. To facilitate the research purpose, we made a bit-for-bit image of the seized servers (in raw dd format) and converted the image to .VMDK format using raw2vmdk. This enabled the forensic image to be booted using a VMWare player, and subsequently provided access to the Seafile server's MySQL database as well as files uploaded to the data repositories in the server environment (hence preserve and collect).

Additionally, it was determined that the artefacts could be collected remotely from the Seafile servers via the web API calls, using scripting options such as curl [103, 104], Python script [105], PHP [106], and Javascripts. This process involves the following steps:

1. Obtain the administrator credentials from the administrator or the client devices.
2. Create an authorisation token for the web API calls using the 'obtain auth' API call e.g., "curl -d "username=<Email Address>&password=<Password>" http(s)://<Server's domain name or IP address>/api2/auth-token".
3. Execute the web API calls of preference with the authorisation token created in step 2, such as:
 - Listing the libraries and the properties: "curl -H 'Authorization: Token <Authorisationtoken>' -H 'Accept: application/json; indent=4' http(s)://<Server's domain name or IP address>/api2/repos".
 - Retrieving the library metadata: "curl -G -H 'Authorization: Token <Authorisationtoken>' -H 'Accept: application/json; indent=4' http(s)://<Server's domain name or IP address>/api2/repos/<Repo ID>/".

- Retrieving the library history: “curl -H ‘Authorization: Token <Authorization token>’ -H ‘Accept: application/json; indent=4’ http(s)://<Server’s domain name or IP address>/api2/repos/<Repo ID>/history/”.
- Downloading a library or sub-library: “curl -H ‘Authorization: Token <Authorization token>’ -H ‘Accept: application/json; charset=utf-8; indent=4’ http(s)://<Server’s domain name or IP address>/api2/repos/<Repo ID>/dir/?p=/<Reponame>”.
- Retrieving the group messages: “curl -H ‘Authorization: Token <Authorization token>’ http(s)://<Server’s domain name or IP address>/api2/groupmsgs/<Message number>/”.
- Downloading a file: “curl -v -H ‘Authorization: Token <Authorization token>’ -H ‘Accept: application/json; charset=utf-8; indent=4’ http(s)://<Server’s domain name or IP address>/api2/repos/<Repo ID>/file/?p=/<Filename>&reuse=1””.

Notice that the ID, filename, and URL for the library or sync file can be located using the list library web API call [103, 104] and in the repo.db, ccnet.conf, and commit files on the desktop clients; data.db database on the Android client; seafolder_pro.sqlite and Cache.db databases on the iOS client. The data collected should then be preserved into a logical container (hence, collect and preserve).

6.8.3 Evidence Analysis and Reconstruction

There is no default installation directory for the server’s data directory, and hence the location may vary according to implementations. Analysis of the /%ccnet%/ccnet.conf file (in the data directory) revealed the server’s username, name, peer ID, and URL in the ‘General’ property. Meanwhile, the ‘Database’ property provided details about the DBMS in use, including the DBMS type, database name for the ccnet server, and the cleartext username and password for the database. The similar database property information could be located for the daemon, file as well as Seahub servers at /%seafolder-data%/seafolder.conf and /%<Data Root Directory>%/seahub_settings.py, along with the secret key and directory path for the file server root in the latter. The database property information could then be used to map the MySQL databases for further administrative and file management metadata as shown in Tables 6, 7, and 8. Similarly to the desktop clients, examination of the /%ccnet%/misc/<Peer ID>file observed the public key of the server, alongside the peer ID and server’s name. The /%ccnet%/mykey.peer file held the private key for the non-HTTP syncing protocol in base64-encoded format.

Cloud Logging and Authentication Data

The cloud transaction and authentication records were predominantly located in the ccnet-db, seafile-db, and seahub-db databases; Tables 9, 10, and 11 show the tables of forensic interest from the databases. Inspection of the Seahub.access.log in %Nginx-<version>%/ (the data directory of the web server used in our research) located records of the HTTP requests (for both the API and non-API calls) from the web and mobile applications, in the format <Client's IP address [Local Access Time]><“Request”><Status><Body Bytes Sent><“HTTP Referrer”><“User Agent String”>[107], Table 12 shows the logs entries of forensic interest. Additionally, a description of the file syncing history (similarly to that found in the client devices) could be recovered for the libraries in the commit files in /%seafile-data%/storage/commits/.

Table 6 Tables of forensic interest to administrative and file management metadata from ccnet-db

Table	Relevance
groupuser	Holds the usernames (user_name) associated with the groups, and indicates whether a user is an administrator (is_staff).
organization	Holds details about the organisations such as the organisation names (org_name), web URLs (url_prefix), email address of the users who created the organisation (creator), and the creation times (ctime); each organisation is identified by a unique organisation ID (org_id).
orggroup	Holds a list of group IDs (group_id) associated with the organisations.
orguser	Keeps track of the user information in relation to the organisation, including the email addresses (email) and whether a user is an administrator (is_staff).

Table 7 Tables of forensic interest to administrative and file management metadata from seafile-db

Table	Relevance
branch	Holds the repo (repo_id) and commit IDs (commit_id) for the libraries.
repogroup	Holds the library sharing properties in the groups such as the repo IDs (repo_id), email addresses of the library creators (user_name), and the permission information (permission).
repoowner	Holds the repo IDs and owners' email addresses (owner_id) for the libraries.
garbageRepos	Holds a list of repo IDs for the deleted libraries.
repousertoken	Holds the user email addresses (email) and magic tokens (token) associated with the libraries.
sharedrepo	Holds the repo IDs (repo_id), senders' emails (from_email), recipients' emails (to_email), and permission information (permission) for the shared libraries; each share is uniquely identified by a share ID (id).

Table 8 Tables of forensic interest to administrative and file management metadata from seahub-db

Table	Relevance
contacts_contact	Maintains a list of contact history, such as the email addresses of the users who initiated the contacts (user_email) and the correspondents' email addresses (contact_email) and contact names (contact_name); each contact is identified by a unique contact ID (id).
profile_detailedprofile	Stores the user names (user), department names (department), and phone numbers (telephone) associated with the users.

Table 9 Tables of forensic interest to cloud logging and authentication data from ccnet-db

Table	Relevance
group	Contains the groups' property information such as the unique group IDs (group_id), names (group_name), email addresses of the group creators (creator_name), and creation times (timestamp).
emailuser	Stores the login email addresses (email), password hashes (passwd), and account creation timestamps in Unix epoch format (ctime) of the users, and details such as whether the users are an admin (is_staff) and active (is_active). The password hashes are stored in the form of 'PBKDF2SHA256\$iterations\$salt\$hash'; indicating the hash algorithm in use, number of iterations of the hash algorithm, random salt number, and final hash generated from the password [95] (93), providing a basis for recovering the login password using the brute force method.
ldapusers	Stores the user IDs (id), email addresses (email), and password hashes (password) for the Lightweight Directory Access Protocol (LDAP) (if any), alongside details such as whether the users are an admin (is_staff) and active (is_active).

Table 10 Tables of forensic interest to cloud logging and authentication data from seafire-db

Table	Relevance
repottrash	Holds the repo IDs (repo_id), library names (repo_name), owners' email addresses (owner_id), folder sizes (size), and deletion times (del_time) associated with the deleted libraries.
repotokenpeerinfo	Holds the magic tokens (token), peer IDs (peer_id), IP addresses (peer_ip), device names (peer_name), and sync times (sync_time) associated with users. The practitioners can map the magic tokens in this table with those in the 'repousertoken' table to determine the libraries associated with the users.

Data Repositories

The data repositories were located in the ‘seafire-data’ sub-directory (of the data directory). When Seafire was deployed on the Ubuntu server, it was possible to

Table 11 Tables of forensic interest to cloud logging and authentication data from seafolder-db

Table	Relevance
group_groupmessage	Holds the group IDs (group_id), senders' emails (from_email), conversation texts (message), as well as timestamps (timestamp) associated with the group conversation threads; each thread is uniquely identified by a group message ID (id).
group_groupmessageattachment	Holds the attachment information associated with the group messages, such as the repo IDs (repo_id), attachment types (attach_type), and repository paths (path). The group messages are identified by the group message IDs of the 'group_groupmessage' table.
group_groupmessagereply	Holds the properties of messages replied to the group(s), such as senders' email addresses (from_email), conversation texts (message), and the message timestamps (timestamp). The group messages identified by the group message IDs of the 'group_groupmessage' table.
message_usermessage	Holds the properties of private messages including the conversation texts (message), email addresses of the senders (from_email) and recipients (to_email), timestamps (timestamp), and whether the messages have been read (ifread), and the deletion timestamps (sender_deleted_at or recipient_deleted_at); each message is identified by a unique message ID (message_id).
share_fileshare	Holds the library/file sharing properties such as the sharers' usernames (username), repo IDs (repo_id), repository paths (path), share tokens (token), share times (ctime), view counts (view_cnt), password hashes (password; if any), and share link expiry dates (expire_date). The data type could be differentiated from the 's_type' table column, where the value 'd' indicates a directory and 'f' indicates a file. Each share is identified by a unique share ID (id).
share_orgfileshare	Holds the share IDs (file_share_id) associated with the organisation IDs (org_id). The share IDs are correlated with the 'id' table column of the 'share_fileshare' table.
share_privatefiledirshare	Holds the private library/file sharing properties such as the IDs of the senders' (from_user) and recipients (to_user), repo IDs (repo_id), repository paths (path), magic tokens (token), permission information (permission), each of which is identified by a unique private share ID (id). The data type could be differentiated from the 's_type' table column, where the value 'd' indicates a directory and 'f' indicates a file.
message_usermsgattachment	Holds the private share IDs (priv_file_dir_share_id) associated with the message IDs (user_msg_id). The private share IDs are correlated with the 'id' table column of the 'share_privatefileshare' table, while the message IDs are identified by the 'message_id' table column of the 'message_usermessage' table.
share_anonymousshare	Holds the anonymous library/file sharing properties, such as the creators' email addresses (repo_owner) and repo IDs (repo_id) for the share libraries, sharers' email addresses (anonymous_email), and the share tokens (token).

(continued)

Table 11 (continued)

Table	Relevance
notification_usernotifications	Holds the notification history, including property information such as the recipients' e-mail addresses (to_user), message types (msg_type; 'group_msg' indicates group message while 'user_message' indicates private message), the associated timestamp information, and whether the notifications have been seen (see). This table also holds details about the user actions in the 'detail' table column in JSON format, such as: <ul style="list-style-type: none"> • Group messaging: "msg_from": "<Sender's email address>", "group_id": <Group ID>, "message": "<Conversation texts from a group message thread>". • Private messaging: "message": "<Conversation texts from a private message thread>", "msg_from": "<Sender's email address>". • Library sharing: {"repo_id": "<Repo ID>", "share_from": "<Sender's email address>"}.
base_userlastlogin	Maintains a list of email addresses (username) and login times associated with the web application's logins.

Table 12 Tables of forensic interest to cloud logging and authentication data from seafile-db

User action	Examples of log entries
Logging in using Google Chrome browser.	192.168.220.165 - - [22/Jul/2015:18:31:25 +0800] "GET /accounts/login/?next=/ HTTP/1.1" 200 11861 "-" "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.99 Safari/537.36"
File download using Mozilla Firefox browser.	192.168.220.165 - - [22/Jul/2015:18:55:11 +0800] "GET /repo/61c957cd-fca0-4044-ae70-3457e879 abdc/499bc9c809ce602a23f85bded5e3544c9e8f42b3/download/?p=%2FEnron3111.txt HTTP/1.1" 302 5 " http://192.168.220.180/ " "Mozilla/5.0 (Windows NT 6.2; WOW64; rv:13.0) Gecko/20100101 Firefox/13.0"
File download using the Seafile iOS client.	192.168.1.12 - - [22/Jul/2015:05:27:03 -0700] "GET /api2/repos/ffac3dff-f9f6-498d-bac4-6feaec805738/file/?p=/Enron3111.txt HTTP/1.1" 200 99 "-" "SeafilePro/1.9.1 CFNetwork/672.1.15 Darwin/14.0.0"
Library/sub-library download using Internet Explorer browser.	192.168.220.165 - - [22/Jul/2015:16:44:47 +0800] "GET /ajax/lib/5172fdc6-7ea2-4c83-a498-bc02bfb0b364/dir/?p=%2F HTTP/1.1" 403 71 " http://192.168.220.180/#my-libs/lib/5172fdc6-7ea2-4c83-a498-bc02bfb0b364 " "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
Deleting a file using Internet explorer browser.	192.168.220.165 - - [22/Jul/2015:16:52:16 +0800] "GET /ajax/repo/5172fdc6-7ea2-4c83-a498-bc02bfb0b364/file/delete/?parent_dir=%2F&name=Enron3111.txt HTTP/1.1" 200 28 " http://192.168.220.180/#my-libs/lib/5172fdc6-7ea2-4c83-a498-bc02bfb0b364 " "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
Deleting an account using Google Chrome browser.	192.168.220.165 - - [22/Jul/2015:13:41:31 +0800] "GET /profile/delete/ HTTP/1.1" 302 5 " http://192.168.220.180/profile/ " "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.99 Safari/537.36"

gain ‘read-only’ access to the unencrypted libraries using the ‘seaf-fuse’ command “./seaf-fuse.sh start -o uid=<UID> ‘<Mount Path>’” in the ‘seafile-server-latest’ sub-directory. By running the ‘seaf-fsck’ command “./seaf-fsck.sh –export ‘<Export Path>’” or “./seaf-fsck.sh –export top_export_path <Library ID>”, we were able to export the unencrypted libraries from the repositories. However, only the ‘seaf-fsck’ command provided access to the deleted libraries and files, but the findings may be subject to the retention period specified in the history settings [108].

7 Discussion

A timeline of the data from the various sources was outlined to demonstrate the cloud transactions from the clients and the servers, including the application installation and accessing times as well as sync directory and file creation and modification times. As the scope of the complete timeline analysis is too broad to present fully in this paper, we only outline the timeline for the Enron3111.txt sample file on the Seafile Windows client and Internet Explorer browser (see Table 13).

Our case study demonstrated that a practitioner investigating Seafile private cloud service can review the server manual (<http://manual.seafile.com>) as a starting point for the investigation. Our examinations of the client devices determined that focus should be placed on the repo.db, ccnet.conf, data.db, accounts.db, mykey.peer, latest_account.xml, Seafile Client.config, com.seafile.Seafile Client.plist, group.com

Table 13 Example of timeline for Seafile research

Source	Key date	Event type	Comment
Autopsy file list	2014-12-14 00:33:11	Last modified	Enron3111.txt
Application.evtx	2015-07-16 13:17:31	Installed	Seafile desktop client version 4.2.8.
SEAFILE-APPLET.EXE-686E37B6(pf)	2015-07-16 13:17:41	First run	First run of the desktop client.
seafile.log	2015-07-16 13:18:38	First access	“[07/16/15 13:18:38] seaf-daemon.c(504): starting seafile client 4.2.8.”
CCNET.EXE-866BD80B(pf)	2015-07-16 13:18:47	First access	First access of the desktop client.
SEAF-DAEMON.EXE-1649E662(pf)	2015-07-16 13:18:48	First access	First access of the desktop client.
Autopsy file list	2015-07-18 18:56:38	Created	C: WindowsClientNotEncrypted Enron3111.txt.
Autopsy file list	2015-07-18 19:00:15	Created	C: WindowsClientEncrypted Enron3111.txt.

(continued)

Table 13 (continued)

Source	Key date	Event type	Comment
Autopsy file list	2015-07-22 00:49:12	Created	C: WindowsClientEncrypted WindowsUploadedFolderEncrypted Enron3111.txt.
CCNET.EXE-866BD80B.pf	2015-07-22 10:33:52	Last access	Last access of the desktop client.
SEAF-DAEMON.EXE-1649E662.pf	2015-07-22 10:33:54	Last access	Last access of the desktop client.
seafile.log	2015-07-22 10:33:57	Last access	[07/22/15 10:33:57] seaf-daemon.c(504): starting seafile client 4.2.8
Autopsy file list	2015-07-22 10:38:50	Initialised library	Created commit file C: Users/anonymous/Seafile/seafile-data/storage/commits/cb1c9eb1-b32f-4fdf-8103-74ad8a1bf479 for library C: WindowsClientNotEncrypted.
Autopsy file list	2015-07-22 11:27:41	Initialised library	Created commit file C: Users/anonymous/Seafile/seafile-data/storage/commits/53fd4e70-3b81-408a-9224-d17fa4efa934 for library C: WindowsClientEncrypted.
Autopsy file list	2015-07-22 11:31:32	Downloaded	C: Users anonymous Seafile MacClientNotEncrypted Enron3111.txt.
Autopsy file list	2015-07-22 11:32:18	Downloaded	C: Users anonymous Seafile MacClientEncrypted Enron3111.txt.
Autopsy file list	2015-07-22 11:32:18	Downloaded	C: Users anonymous Seafile MacClientEncrypted MacUploadedFolderEncrypted Enron3111.txt
Application.evtx	2015-07-22 11:45:27	Uninstalled	Seafile desktop client version 4.2.8.
SEAFILE-APPLET.EXE-686E37B6.pf	2015-07-22 11:45:57	Last run	Last run of the desktop client.
Seahub.access.log	2015-07-22 16:42:23	Accessed	Logging in the web application using Internet Explorer browser "192.168.220.165 - - [22/Jul/2015:16:42:23 +0800] "GET / HTTP/1.1" 200 57978 " http://192.168.220.180/accounts/login/ " "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko""
Autopsy file list	2015-07-22 16:44:23	Initialised library	Created commit file C: seafile-server seafile-data storage commits 5172fdc6-7ea2-4c83-a498-bc02fb0b364 for library 'IEClientEncrypted'.

(continued)

Table 13 (continued)

Source	Key date	Event type	Comment
Commit file '82 df490a8cde25b30b8eb9e1a9c3f80dbba7afbf'	2015-07-22 16:45:19	Uploaded	dded Enron3111.txt to library 'IEClientEncrypted' {"repo_id": "5172fdc6-7ea2-4c83-a498-bc02fb0b364", "parent_id": "1c63aea6dfc8e67c1633c85 8643d2e12db32f5bb", "second_parent_id": null, "repo_category": null, "commit_id": "82df490a8cde25b30b8eb9 e1a9c3f80dbba7afbf", "root_id": "a3215116c149e20dc5ec6d 34f3f62b1f1e3d2bf0", "creator_name": "internetexplorer@gmail.com", "creator": "00", "description": "Added \"Enron3111.txt\".", "ctime": 1437554719, "no_local_history": 1, "encrypted": true, "repo_name": "IEClientEncrypted", "repo_desc": "", "enc_version": 2, "magic": "4cda04342a119309 b88affcb376a85a504e9d5ff6014c406f01677a6cc 267620", "key": "9a47bcf101069d3ce80eaa572 5b322cbebcc9f44f7956f5a6065a2587bebfc8bb0 bfc3401ef1ee59686e5f6dad97ccb1", "version": 1}.
Autopsy file list	2015-07-22 16:46:02	Initialised library	Created commit file C: seafolder-seafolder-data storage commits 71fd040-89d-bdbc-d3e2-0f8c4d 5ebe355d5d949-7ea2-4c83-a498-bc02fb0b364 for library 'IEClientNotEncrypted'.
Commit file 'ad 0f379011a651c1ad6a6a19631e402ecf4d5b56'	2015-07-22 16:46:34	Uploaded	Added Enron3111.txt to library 'IEClientNotEncrypted' {"repo_id": "a6b20769-99a9-4402-b7d0-6d0b55740f5c", "parent_id": "58e36f98bb9a1c513e98e796c7c7bfdd6ec27ccb", "second_parent_id": null, "repo_category": null, "commit_id": "ad0f379011a651c1ad6a6a1 9631e402ecf4d5b56", "root_id": "2b3a68dc670 0acc28d856bd96b4b49376a125d4e", "repo_name": "IEClientNotEncrypted", "no_local_history": 1, "creator_name": "internetexplorer@gmail.com", "creator": "00", "description": "Added \"Enron3111.txt\".", "ctime": 1437554794, "repo_desc": "", "version": 1}.
BrowsingHistory View for Internet Explorer	2015-07-22 16:51:08	Downloaded	Downloaded Enron3111.txt from library 'IEClientEncrypted' " http://192.168.220.180/repo/5172fdc6-7ea2-4c83-a498-bc02fb0b364/e11d7d0be7d0a81fed7dbc4545cf7294c7f314c1/download/?p=%2FEnron3111.txt ".

(continued)

Table 13 (continued)

Source	Key date	Event type	Comment
Seahub.access.log	2015-07-22 16:51:08	Downloaded	Downloaded Enron.3111.txt from library 'IEClientEncrypted' using Internet Explorer "192.168.220.165 - - [22/Jul/2015:16:51:08 +0800] \"GET/repo/5172fdc6-7ea2-4c83-a498-bc02bfb0b364/e11d7d0be7d0a81fed7dbc4545cf7294c7f314c1/download/?p=%2FEnron3111.txtHTTP/1.1\" 302 5 " http://192.168.220.180/ " "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"".
Autopsy file list	2015-07-22 16:51:23	Created	Created C: Users anonymous Downloads Enron3111.txt.
Seahub.access.log	2015-07-22 16:52:16	Last visited	Deleted Enron.3111.txt from repo 'IEClientEncrypted' using Internet Explorer "192.168.220.165 - - [22/Jul/2015:16:52:16 +0800] \"GET/ajax/repo/5172fdc6-7ea2-4c83-a498-bc02bfb0b364/file/delete/?parent_dir=%2F&name=Enron3111.txtHTTP/1.1\" 200 28 " http://192.168.220.180/#my-libs/lib/5172fdc6-7ea2-4c83-a498-bc02bfb0b364 " "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"".

.seafile.seafilePro.plist files for the synchronisation and file management as well as authentication and encryption metadata. These include the credential information, URLs to the servers and data repositories, as well as the directory paths, email addresses and repo and peer IDs associated with the libraries, which are necessary for identifying the data storage and mapping the cloud transaction records for timelining the user actions. Our examinations of the mobile clients determined that the mobile applications maintained a different data directory structure in comparison with the desktop applications, and only caches of the viewed files and HTTP requests could be recovered.

Our inspection of the MD5 values for the downloaded files observed the same values with the original copies in all cases, establishing the integrity of the files downloaded using the client applications. With regard to the file deletion, it was determined that the files deleted (locally and remotely from the corresponding nodes) could be recovered locally from the unallocated space. However, only the files deleted locally could be recovered from the Trash or Recycle Bin directory. As for the web application, the deleted files/libraries could be restored from the trash function using an administrator's account, but the findings may be subject to the retention period specified in the history settings. Uninstallation of the client applications removed the 'seafile-data' and 'ccnet' directories from the desktop clients, but not for the download directories. However, uninstallation of the mobile

client applications removed the data directories completely from the mobile clients. Deleting the Seafile accounts using the client applications resulted in the same outcomes as with the case of the uninstallation, but deleting the Seafile accounts using the web application removed the accounts completely from the server (including the sync libraries).

Analysis of the network traffic produced unique identifiable information such as the IP addresses, URLs, and timestamps for the cloud transactions. We also managed to reconstruct the complete web application root directories for the downloaded/viewed files, web resources (icon images, HTML and JSON files for the webpages etc.) as well as script files for the HTTP requests and responses for the web API/non-API calls, but it is noteworthy that the artefacts may vary according to different network protocol implementations. For example, if the HTTPS protocol is used, the HTTP requests will be fully encrypted in the network traffic and in such environment, the practitioner may rely on the web browsing history and caches from web browsers as potential alternative sources of network artefacts [68].

Our examinations of the physical memory captures indicated that the memory dumps can provide potential for alternative methods for recovering the login credentials, encryption passwords, client/web application caches, logs, and HTTP request/response bodies for the web API/non-web-API calls. In future investigations, the artefacts could be identified by terms unique to the log and database files, web API/non-web-API methods, and other keywords of relevance as identified in our research. We speculate that the credential information appeared in the memory dump for two reasons: the client application cached the credential information in the memory for creating the authorisation token for the web API calls; texts written to the memory during the payload creation for the web API calls of relevance. The memory dump also provided an alternative method for recovering the running process and network information using the ‘netscan’ or ‘netstat’ function of Volatility. The presence of the artefacts in the memory dump also means that the artefacts can be potentially located in the swap files as a result of inactive memory pages being swapped out of the memory to the hard disk during the system’s normal operation [69–71, 77]. Therefore, the swap files should not be overlooked. Nevertheless, a practitioner must keep in mind that memory changes frequently according to users’ activities and will be wiped as soon as the system is shut down. Hence, obtaining a memory snapshot of a compromised system as quickly as possible increases the likelihood of preserving the artefacts before being overwritten in memory.

Our forensic examination of the Seafile server revealed that the administrative and file management metadata as well as cloud logging and authentication data could be recovered to support evidence found from the client devices. There are also residual artefacts specific to the server. For example, the libraries deleted from the web applications could only be recovered from the server using the ‘seaf-fsck’ script. However, the script is not supported by the Seafile Windows server application, and hence a migration of the data directory to a Linux server may

be necessary [109]. Moreover, the Seafile server provides the only way (to our knowledge) to detect the IP addresses and device names of the synchronisation points associated with a library (e.g., from the web application log or mapping the magic tokens in the ‘repousertoken’ and ‘repotokenpeerinfo’ table columns of the ‘seafile-db’ database), which are essential for tracking a suspect’s location and search warrant issuance. Hence, the importance of collecting evidence from the Seafile server should not be overlooked. A summary of the findings from the Seafile private cloud storage service is in Table 14.

Table 14 HTTP request and response bodies of forensic interest

Category	Artefact category	Sources of information
Client	Sync and file management metadata	<ul style="list-style-type: none"> • /%seafolder-data%/repo.dband/%ccnet%/ccnet.c onfonthedesktopclients • /data/data/com.seafile.seadroid2/databases/databasedatabaseontheAndroidclient
	Authentication and encryption metadata	<ul style="list-style-type: none"> • /%seafolder-data%/accounts.db,%ccnet%/mykey.peer,%ccnet%/misc/<peerID> on the desktop clients • /data/data/com.seafile.seadroid2/databases/accounts.db and /data/data/com.seafile.seadroid2/shared_prefs/latest_account.xml on the Android client • ./config/Seafile/SeafileClient.config on the Ubuntu client • /Users/<UserProfile>/Library/Preferences/com.seafile.Seafile Client.plist on the Mac OS client • /private/var/mobile/Applications/<UUIDforthe SeafileiOSApp>/Library/Preferences/group.com.seafile.seafilePro.plist on the iOS client. • When the login credentials and encryption password were saved in the web browsers, it was possible to recover the credential information or password using Nirsoft Web Browser Passview

(continued)

Table 14 (continued)

Category	Artefact category	Sources of information
	Cloud transaction logs	<ul style="list-style-type: none"> • <code>/%seafile-data%/accounts.db,/%seafile-data%/storage/commits/<RepoID>/<first two characters of the commit ID>/<remaining of the commit ID>, /%seafile-data%/index/<Repo ID>, and /%ccnet%/logs/seafile.log</code> on the desktop clients • <code>/data/data/com.seafile.seadroid2/databases/accounts.db</code> and <code>/data/data/com.seafile.seadroid2/databases/data.db</code> on the Android client • <code>/Users/<UserProfile>/Library/Preferences/com.apple.spotlight.plist</code> on the Mac OS client. • <code>/private/var/mobile/Containers/<UUID>/seafile/seafile_pro.sqlite</code> and <code>/private/var/mobile/Applications/<UUIDfortheSeafileiOSApp>/Library/Caches/com.seafile.seafilePro/Cache.db</code> on the iOS client • Application.evtx, CCNET.EXE(pf, SEAFILE-APPLET.EXE(pf, Seafile.lnk, logs.lnk, ccnet.lnk, seafile.lnk, and HKEY_LOCAL_MACHINE SOFTWARE Microsoft Windows CurrentVersion Installer UserData <SID> Products <GUID> InstallProperties and HKEY_LOCAL_MACHINE SOFTWARE Wow6432Node Microsoft Windows CurrentVersion Uninstall {<GUID>} registry on the Windows client • The Zeitgeist, recently-used.xbel, and system and dpkg logs of the Ubuntu client • Web browsing history and caches
	Data Storage	<ul style="list-style-type: none"> • <code>%Users% <User Profile> Seafile</code> on the Windows client. • <code>/home/<User Profile>/Seafile/</code> on the Ubuntu client. • <code>/Users/<User Profile>/Seafile/</code> on the Mac OS client. • <code>/home/Seafile/<User'sEmailAddress>(<Server'sURL>)</code> on the Android client. • <code>/private/var/mobile/Containers/<UUID>/seafile/objects/</code> and <code>/private/var/mobile/Containers/<UUID>/seafile/uploads/</code> on the iOS client. • <code>/%seafile-data%/avatars</code> • Copies of the synced PDF and images from the thumbnail cache

(continued)

Table 14 (continued)

Category	Artefact category	Sources of information
	Network Captures	<ul style="list-style-type: none"> IP addresses of the clients and servers The timestamp information associated with the cloud transactions Clear text HTTP requests and full URLs to the server's directories (in the 'Referer' header field) for the web API/non-web-API calls Copies of the complete server directories for the HTTP requests
	Volatile memory dumps	<ul style="list-style-type: none"> The process names could be differentiated from 'ccnet', 'seaf-daemon', and 'seaf-applet' Portion of the HTTP request body for the authorisation token and library creation in plain text Copies of the commit files, databases (e.g., repo.db and accounts.db), and log files in plain text Copies of the sample files and icon images used by the client and web applications (via data carving)
Server	Administrative and file management metadata	<ul style="list-style-type: none"> /%ccnet%/ccnet.conf /%seafle-data%/seafle.conf /%<DataRootDirectory>%/seahub_settings.py /%ccnet%/misc/<PeerID> /%ccnet%/mykey.peer ccnet-db, seafle-db, and seahub-db MySQL databases (DBMS and database names may vary according to implementations)
	Cloud logging and authentication data	<ul style="list-style-type: none"> ccnet-db and seahub-db MySQL databases Seahub.access.log in the web server's data directory Commit files in /%seafle-data%/storage/commits/
	Data repositories	<ul style="list-style-type: none"> 'seafle-data' directory When Seaf file was deployed on the Ubuntu server, it was possible to reconstruct the synced files from the unencrypted repositories using the 'seaf-fsck.sh' and 'Seaf-fuse.sh' scripts

(continued)

8 Conclusion and Future Work

Although private cloud might appear attractive to small and medium enterprise users, it is not forensic challenge-free. In this paper, we proposed a conceptual forensics framework for private cloud storage investigation. We demonstrated the utility of our proposed framework in the investigation of the popular open-source private cloud storage service Seafile. Our investigations revealed that a wide range of evidential data could be recovered from the (desktop and mobile) client devices and network traffic after a user has used the Seafile client applications, such as artefacts relating to the installation, uninstallation, log in and log off, account management, conversations, and file synchronisation and sharing. The recovered terrestrial artefacts can inform the and expand the investigation scope (e.g. identifying the relevant cloud hosting instance, necessary for evidence tracking and provenance determination). The proposed framework will provide forensic practitioners with an efficient process to investigate private cloud services.

References

1. James Baldwin, Omar M. K. Alhawi, Simone Shaughnessy, Alex Akinbi, and Ali Dehghantanha. Emerging from the cloud: A bibliometric analysis of cloud forensics studies. In *Advances in Information Security*, pages 311–331. Springer International Publishing, 2018. doi: 10.1007/978-3-319-73951-9_16. URL https://doi.org/10.1007/978-3-319-73951-9_16.
2. Yee-Yang Teing, Ali Dehghantanha, and Kim-Kwang Raymond Choo. CloudMe forensics: A case of big data forensic investigation. *Concurrency and Computation: Practice and Experience*, 30(5):e4277, jul 2017. doi: 10.1002/cpe.4277. URL <https://doi.org/10.1002/cpe.4277>.
3. Kim-Kwang Raymond Choo. Organised crime groups in cyberspace: a typology. *Trends in Organized Crime*, 11(3):270–295, Sep 2008. ISSN 1936-4830. doi: 10.1007/s12117-008-9038-9. URL <https://doi.org/10.1007/s12117-008-9038-9>.
4. Kim-Kwang Raymond Choo. Cloud computing: Challenges and future directions. <http://www.aic.gov.au/publications/currentJan> 2018. (Accessed on 01/02/2018).
5. Nurul Hidayah Ab Rahman and Kim-Kwang Raymond Choo. A survey of information security incident handling in the cloud. *Computers & Security*, 49:45–69, mar 2015. doi: 10.1016/j.cose.2014.11.006. URL <https://doi.org/10.1016/j.cose.2014.11.006>.
6. Yee-Yang Teing, Ali Dehghantanha, Kim-Kwang Raymond Choo, and Laurence T Yang. Forensic investigation of p2p cloud storage services and backbone for IoT networks: BitTorrent sync as a case study. *Computers & Electrical Engineering*, 58:350–363, feb 2017. doi: 10.1016/j.compeleceng.2016.08.020. URL <https://doi.org/10.1016/j.compeleceng.2016.08.020>.
7. Opeyemi Osanaiye, Haibin Cai, Kim-Kwang Raymond Choo, Ali Dehghantanha, Zheng Xu, and Mqhele Dlodlo. Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2016 (1), may 2016. doi: 10.1186/s13638-016-0623-3. URL <https://doi.org/10.1186/s13638-016-0623-3>.
8. Alan Duke. 5 things to know about the celebrity nude photo hacking scandal, 2014. <http://edition.cnn.com/2014/09/02/showbiz/hacked-nude-photos-five-things/>, Jan 2018. (Accessed on 01/02/2018).

9. Opeyemi Osanaiye, Haibin Cai, Kim-Kwang Raymond Choo, Ali Dehghantanha, Zheng Xu, and Mqhele Dlodlo. Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2016(1), may 2016. doi: 10.1186/s13638-016-0623-3. URL <https://doi.org/10.1186/s13638-016-0623-3>.
10. Opeyemi Osanaiye, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework. *Journal of Network and Computer Applications*, 67:147–165, may 2016. doi: 10.1016/j.jnca.2016.01.001. URL <https://doi.org/10.1016/j.jnca.2016.01.001>.
11. Stuart Schechter David Molnar. Self hosting vs. cloud hosting: Accounting for the security impact of hosting in the cloud. <http://research.microsoft.com/apps/pubs/default.aspx?id=132318>, Jun 2010. (Accessed on 01/02/2018).
12. Amarnath Jasti, Payal Shah, Rajeev Nagaraj, and Ravi Pendse. Security in multi-tenancy cloud. In *44th Annual 2010 IEEE International Carnahan Conference on Security Technology*. IEEE, oct 2010. doi: 10.1109/cbst.2010.5678682. URL <https://doi.org/10.1109/cbst.2010.5678682>.
13. R. Ko and R. Choo. *The Cloud Security Ecosystem: Technical, Legal, Business and Management Issues*. Elsevier Science, 2015. ISBN 9780128017807. URL <https://books.google.com/books?id=meycBAAAQBAJ>.
14. Olga Kharif Joseph Galante and Pavel Alpeev. Sony network breach shows amazon cloud's appeal for hackers. <https://www.bloomberg.com/news/articles/2011-05-15/sony-attack-shows-amazon-s-cloud-service-lures-hackers-at-pennies-an-hour>, May 2011. (Accessed on 01/02/2018).
15. Yee-Yang Teing, Ali Dehghantanha, Kim-Kwang Raymond Choo, Tooska Dargahi, and Mauro Conti. Forensic investigation of cooperative storage cloud service: Symform as a case study. *Journal of Forensic Sciences*, 62(3):641–654, nov 2016. doi: 10.1111/1556-4029.13271. URL <https://doi.org/10.1111/1556-4029.13271>.
16. Farid Daryabar, Ali Dehghantanha, and Kim-Kwang Raymond Choo. Cloud storage forensics: MEGA as a case study. *Australian Journal of Forensic Sciences*, 49(3):344–357, apr 2016. doi: 10.1080/00450618.2016.1153714. URL <https://doi.org/10.1080/00450618.2016.1153714>.
17. William R Claycomb and Alex Nicoll. Insider threats to cloud computing: Directions for new research challenges. In *2012 IEEE 36th Annual Computer Software and Applications Conference*. IEEE, jul 2012. doi: 10.1109/compsac.2012.113. URL <https://doi.org/10.1109/compsac.2012.113>.
18. T. Dargahi, A. Dehghantanha, and M. Conti. Investigating storage as a service cloud platform. In *Contemporary Digital Forensic Investigations of Cloud and Mobile Applications*, pages 185–204. Elsevier, 2017. doi: 10.1016/b978-0-12-805303-4.00012-5. URL <https://doi.org/10.1016/b978-0-12-805303-4.00012-5>.
19. Ali Dehghantanha and Katrin Franke. Privacy-respecting digital investigation. In *2014 Twelfth Annual International Conference on Privacy, Security and Trust*. IEEE, jul 2014. doi: 10.1109/pst.2014.6890932. URL <https://doi.org/10.1109/pst.2014.6890932>.
20. Kim-Kwang Raymond Choo and Rick Sarre. Balancing privacy with legitimate surveillance and lawful data access. *IEEE Cloud Computing*, 2(4):8–13, jul 2015. doi: 10.1109/mcc.2015.84. URL <https://doi.org/10.1109/mcc.2015.84>.
21. Surya Nepal, Rajiv Ranjan, and Kim-Kwang Raymond Choo. Trustworthy processing of healthcare big data in hybrid clouds. *IEEE Cloud Computing*, 2(2):78–84, mar 2015. doi: 10.1109/mcc.2015.36. URL <https://doi.org/10.1109/mcc.2015.36>.
22. Barton Gellman, Ashkan Soltani, and Andrea Peterson. How we know the nsa had access to internal google and yahoo cloud data. https://www.washingtonpost.com/news/the-switch/wp/2013/11/04/how-we-know-the-nsa-had-access-to-internal-google-and-yahoo-cloud-data/?noredirect=on&utm_term=.6933b51f8781, Nov 2013. (Accessed on 01/02/2018).

23. Rong Jiang, Rongxing Lu, and Kim-Kwang Raymond Choo. Achieving high performance and privacy-preserving query over encrypted multidimensional big metering data. *Future Generation Computer Systems*, 78:392–401, jan 2018. doi: 10.1016/j.future.2016.05.005. URL <https://doi.org/10.1016/j.future.2016.05.005>.
24. Ximeng Liu, Robert H. Deng, Kim-Kwang Raymond Choo, and Jian Weng. An efficient privacy-preserving outsourced calculation toolkit with multiple keys. *IEEE Transactions on Information Forensics and Security*, 11(11):2401–2414, nov 2016. doi: 10.1109/tifs.2016.2573770. URL <https://doi.org/10.1109/tifs.2016.2573770>.
25. Ximeng Liu, Kim-Kwang Raymond Choo, Robert H. Deng, Rongxing Lu, and Jian Weng. Efficient and privacy-preserving outsourced calculation of rational numbers. *IEEE Transactions on Dependable and Secure Computing*, 15(1):27–39, jan 2018. doi: 10.1109/tdsc.2016.2536601. URL <https://doi.org/10.1109/tdsc.2016.2536601>.
26. Luciana Duranti and Corinne Rogers. Trust in digital records: An increasingly cloudy legal area. *Computer Law & Security Review*, 28(5):522–531, oct 2012. doi: 10.1016/j.clsr.2012.07.009. URL <https://doi.org/10.1016/j.clsr.2012.07.009>.
27. S.H. Mohtasebi, A. Dehghantanha, and K.-K.R. Choo. Cloud storage forensics. In *Contemporary Digital Forensic Investigations of Cloud and Mobile Applications*, pages 205–246. Elsevier, 2017. doi: 10.1016/b978-0-12-805303-4.00013-7. URL <https://doi.org/10.1016/b978-0-12-805303-4.00013-7>.
28. A. Dehghantanha and T. Dargahi. Residual cloud forensics. In *Contemporary Digital Forensic Investigations of Cloud and Mobile Applications*, pages 247–283. Elsevier, 2017. doi: 10.1016/b978-0-12-805303-4.00014-9. URL <https://doi.org/10.1016/b978-0-12-805303-4.00014-9>.
29. Ben Martini and Kim-Kwang Raymond Choo. Distributed filesystem forensics: XtreemFS as a case study. *Digital Investigation*, 11(4):295–313, dec 2014. doi: 10.1016/j.diin.2014.08.002. URL <https://doi.org/10.1016/j.diin.2014.08.002>.
30. Mohammad Shariati, Ali Dehghantanha, Ben Martini, and Kim-Kwang Raymond Choo. Ubuntu one investigation. In *The Cloud Security Ecosystem*, pages 429–446. Elsevier, 2015. doi: 10.1016/b978-0-12-801595-7.00019-7. URL <https://doi.org/10.1016/b978-0-12-801595-7.00019-7>.
31. Asou Aminnezhad, Ali Dehghantanha, Mohd Taufik Abdullah, and Mohsen Damshenas. Cloud forensics issues and opportunities. *International Journal of Information Processing and Management*, 4(4):76–85, jun 2013. doi: 10.4156/ijipm.vol4.issue4.9. URL <https://doi.org/10.4156/ijipm.vol4.issue4.9>.
32. Farid Daryabar and Ali Dehghantanha. A review on impacts of cloud computing and digital forensics. *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, 2(2), 2014.
33. M. Amine Chelihi, A. Elutilo, I. Ahmed, C. Papadopoulos, and A. Dehghantanha. An android cloud storage apps forensic taxonomy. In *Contemporary Digital Forensic Investigations of Cloud and Mobile Applications*, pages 285–305. Elsevier, 2017. doi: 10.1016/b978-0-12-805303-4.00015-0. URL <https://doi.org/10.1016/b978-0-12-805303-4.00015-0>.
34. Farid Daryabar, Ali Dehghantanha, Brett Eterovic-Soric, and Kim-Kwang Raymond Choo. Forensic investigation of OneDrive, box, GoogleDrive and dropbox applications on android and iOS devices. *Australian Journal of Forensic Sciences*, 48(6):615–642, mar 2016. doi: 10.1080/00450618.2015.1110620. URL <https://doi.org/10.1080/00450618.2015.1110620>.
35. Data model · seafie server manual. https://manual.seafie.com/develop/data_model.html, . (Accessed on 07/06/2018).
36. Fuse extension · seafie server manual. <https://manual.seafie.com/extension/fuse.html>, . (Accessed on 07/06/2018).
37. Seafie fsck · seafie server manual. https://manual.seafie.com/maintain/seafie_fsck.html, . (Accessed on 07/06/2018).
38. Farhood Norouzizadeh Dezfooli, Ali Dehghantanha, Brett Eterovic-Soric, and Kim-Kwang Raymond Choo. Investigating social networking applications on smart-

- phones detecting facebook, twitter, LinkedIn and google+ artefacts on android and iOS platforms. *Australian Journal of Forensic Sciences*, 48(4):469–488, aug 2015. doi: 10.1080/00450618.2015.1066854. URL <https://doi.org/10.1080/00450618.2015.1066854>.
39. Yee-Yang Teing, Ali Dehghantanha, Kim-Kwang Raymond Choo, Zaiton Muda, Mohd Taufik Abdullah, and Wee-Chiat Chai. A closer look at syncany windows and ubuntu clients' residual artefacts. In *Lecture Notes in Computer Science*, pages 342–357. Springer International Publishing, 2016. doi: 10.1007/978-3-319-49145-5_34. URL https://doi.org/10.1007/978-3-319-49145-5_34.
40. Mohammad Shariati, Ali Dehghantanha, and Kim-Kwang Raymond Choo. Sugarsync forensic analysis. *Australian Journal of Forensic Sciences*, 48(1):95–117, apr 2015. doi: 10.1080/00450618.2015.1021379. URL <https://doi.org/10.1080/00450618.2015.1021379>.
41. Ben Blakeley, Chris Cooney, Ali Dehghantanha, and Rob Aspin. Cloud storage forensic: hubiC as a case-study. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, nov 2015. doi: 10.1109/cloudcom.2015.24. URL <https://doi.org/10.1109/cloudcom.2015.24>.
42. M. Petraityte, A. Dehghantanha, and G. Epiphaniou. Mobile phone forensics. In *Contemporary Digital Forensic Investigations of Cloud and Mobile Applications*, pages 79–89. Elsevier, 2017. doi: 10.1016/b978-0-12-805303-4.00006-x. URL <https://doi.org/10.1016/b978-0-12-805303-4.00006-x>.
43. Ali Dehghantanha and Katrin Franke. Privacy-respecting digital investigation. In *2014 Twelfth Annual International Conference on Privacy, Security and Trust*. IEEE, jul 2014. doi: 10.1109/pst.2014.6890932. URL <https://doi.org/10.1109/pst.2014.6890932>.
44. James Gill, Ihechi Okere, Hamed HaddadPajouh, and Ali Dehghantanha. Mobile forensics: A bibliometric analysis. In *Advances in Information Security*, pages 297–310. Springer International Publishing, 2018. doi: 10.1007/978-3-319-73951-9_15. URL https://doi.org/10.1007/978-3-319-73951-9_15.
45. Mohsen Damshenas, Ali Dehghantanha, Ramlan Mahmoud, and Solahuddin bin Shamsuddin. Forensics investigation challenges in cloud computing environments. In *Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*. IEEE, jun 2012. doi: 10.1109/cybersec.2012.6246092. URL <https://doi.org/10.1109/cybersec.2012.6246092>.
46. Darren Quick, Ben Martini, and Kim-Kwang Raymond Choo. *Cloud Storage Forensics*. Elsevier, 2014. doi: 10.1016/c2013-0-09718-6. URL <https://doi.org/10.1016/c2013-0-09718-6>.
47. Ben Martini and Kim-Kwang Raymond Choo. Cloud forensic technical challenges and solutions: A snapshot. *IEEE Cloud Computing*, 1(4):20–25, nov 2014. doi: 10.1109/mcc.2014.69. URL <https://doi.org/10.1109/mcc.2014.69>.
48. Keyun Ruan, Joe Carthy, Tahar Kechadi, and Mark Crosbie. Cloud forensics. In *Advances in Digital Forensics VII*, pages 35–46. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-24212-0_3. URL https://doi.org/10.1007/978-3-642-24212-0_3.
49. Dominik Birk and Christoph Wegener. Technical issues of forensic investigations in cloud computing environments. In *2011 Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*. IEEE, may 2011. doi: 10.1109/sadfe.2011.17. URL <https://doi.org/10.1109/sadfe.2011.17>.
50. Stephen O'Shaughnessy and Anthony Keane. Impact of cloud computing on digital forensic investigations. In *Advances in Digital Forensics IX*, pages 291–303. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-41148-9_20. URL https://doi.org/10.1007/978-3-642-41148-9_20.
51. Christopher Hooper, Ben Martini, and Kim-Kwang Raymond Choo. Cloud computing and its implications for cybercrime investigations in australia. *Computer Law & Security Review*, 29(2):152–163, apr 2013. doi: 10.1016/j.clsr.2013.01.006. URL <https://doi.org/10.1016/j.clsr.2013.01.006>.

52. Stavros Simou, Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. Cloud forensics: Identifying the major issues and challenges. In Matthias Jarke, John Mylopoulos, Christoph Quix, Colette Rolland, Yannis Manolopoulos, Haralampos Mouratidis, and Jennifer Horkoff, editors, *Advanced Information Systems Engineering*, pages 271–284, Cham, 2014. Springer International Publishing. ISBN 978-3-319-07881-6.
53. Mohsen Damshenas, Ali Dehghantanha, and Ramlan Mahmoud. A survey on digital forensics trends. *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, 3:209–234, 2014.
54. Darren Quick and Kim-Kwang Raymond Choo. Impacts of increasing volume of digital forensic data: A survey and future research challenges. *Digital Investigation*, 11(4):273–294, dec 2014. doi: 10.1016/j.diin.2014.09.002. URL <https://doi.org/10.1016/j.diin.2014.09.002>.
55. Darren Quick and Kim-Kwang Raymond Choo. Data reduction and data mining framework for digital forensic evidence: Storage, intelligence, review and archive. *Trends & Issues in Crime and Criminal Justice*, 480:1–11, 2014.
56. Lingjun Zhao, Lajiao Chen, Rajiv Ranjan, Kim-Kwang Raymond Choo, and Jijun He. Geographical information system parallelization for spatial big data processing: a review. *Cluster Computing*, 19(1):139–152, Mar 2016. ISSN 1573-7543. doi: 10.1007/s10586-015-0512-2. URL <https://doi.org/10.1007/s10586-015-0512-2>.
57. Darren Quick and Kim-Kwang Raymond Choo. Big forensic data reduction: digital forensic images and electronic evidence. *Cluster Computing*, 19(2):723–740, Jun 2016. ISSN 1573-7543. doi: 10.1007/s10586-016-0553-1. URL <https://doi.org/10.1007/s10586-016-0553-1>.
58. Yanjiang Yang, Haiyan Zhu, Haibing Lu, Jian Weng, Youcheng Zhang, and Kim-Kwang Raymond Choo. Cloud based data sharing with fine-grained proxy re-encryption. *Pervasive and Mobile Computing*, 28:122–134, jun 2016. doi: 10.1016/j.pmcj.2015.06.017. URL <https://doi.org/10.1016/j.pmcj.2015.06.017>.
59. Ben Martini and Kim-Kwang Raymond Choo. An integrated conceptual digital forensic framework for cloud computing. *Digital Investigation*, 9(2):71–80, nov 2012. doi: 10.1016/j.diin.2012.07.001. URL <https://doi.org/10.1016/j.diin.2012.07.001>.
60. Raffael Marty. Cloud application logging for forensics. In *Proceedings of the 2011 ACM Symposium on Applied Computing - SAC 11*. ACM Press, 2011. doi: 10.1145/1982185.1982226. URL <https://doi.org/10.1145/1982185.1982226>.
61. Shams Zawoad and Ragib Hasan. Cloud forensics: A meta-study of challenges, approaches, and open problems. *CoRR*, abs/1302.6312, 2013. URL <http://arxiv.org/abs/1302.6312>.
62. Josiah Dykstra and Alan T. Sherman. Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform. *Digital Investigation*, 10:S87–S95, aug 2013. doi: 10.1016/j.diin.2013.06.010. URL <https://doi.org/10.1016/j.diin.2013.06.010>.
63. Tobias Gebhardt and Hans P. Reiser. Network forensics for cloud computing. In *Distributed Applications and Interoperable Systems*, pages 29–42. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-38541-4_3. URL https://doi.org/10.1007/978-3-642-38541-4_3.
64. Shams Zawoad, Amit Kumar Dutta, and Ragib Hasan. SecLaaS. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security - ASIA CCS 13*. ACM Press, 2013. doi: 10.1145/2484313.2484342. URL <https://doi.org/10.1145/2484313.2484342>.
65. Ben Martini and Kim-Kwang Raymond Choo. Remote programmatic vCloud forensics: A six-step collection process and a proof of concept. In *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, sep 2014. doi: 10.1109/trustcom.2014.124. URL <https://doi.org/10.1109/trustcom.2014.124>.
66. Josiah Dykstra and Alan T. Sherman. Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques. *Digital Investigation*, 9:S90–S98, aug 2012. doi: 10.1016/j.diin.2012.05.001. URL <https://doi.org/10.1016/j.diin.2012.05.001>.

67. Neha Thethi and Anthony Keane. Digital forensics investigations in the cloud. In *2014 IEEE International Advance Computing Conference (IACC)*. IEEE, feb 2014. doi: 10.1109/iadcc.2014.6779543. URL <https://doi.org/10.1109/iadcc.2014.6779543>.
68. Hyunji Chung, Jungheum Park, Sangjin Lee, and Cheulhoon Kang. Digital forensic investigation of cloud storage services. *Digital Investigation*, 9(2):81–95, nov 2012. doi: 10.1016/j.diin.2012.05.015. URL <https://doi.org/10.1016/j.diin.2012.05.015>.
69. Darren Quick and Kim-Kwang Raymond Choo. Dropbox analysis: Data remnants on user machines. *Digital Investigation*, 10(1):3–18, jun 2013. doi: 10.1016/j.diin.2013.02.003. URL <https://doi.org/10.1016/j.diin.2013.02.003>.
70. Darren Quick and Kim-Kwang Raymond Choo. Google drive: Forensic analysis of data remnants. *Journal of Network and Computer Applications*, 40:179–193, apr 2014. doi: 10.1016/j.jnca.2013.09.016. URL <https://doi.org/10.1016/j.jnca.2013.09.016>.
71. Darren Quick and Kim-Kwang Raymond Choo. Digital droplets: Microsoft SkyDrive forensic data remnants. *Future Generation Computer Systems*, 29(6):1378–1394, aug 2013. doi: 10.1016/j.future.2013.02.001. URL <https://doi.org/10.1016/j.future.2013.02.001>.
72. Jason S. Hale. Amazon cloud drive forensic analysis. *Digital Investigation*, 10(3):259–265, oct 2013. doi: 10.1016/j.diin.2013.04.006. URL <https://doi.org/10.1016/j.diin.2013.04.006>.
73. Jason Farina, Mark Scanlon, and M-Tahar Kechadi. BitTorrent sync: First impressions and digital forensic implications. *Digital Investigation*, 11:S77–S86, may 2014. doi: 10.1016/j.diin.2014.03.010. URL <https://doi.org/10.1016/j.diin.2014.03.010>.
74. Ben Martini and Kim-Kwang Raymond Choo. Cloud storage forensics: ownCloud as a case study. *Digital Investigation*, 10(4):287–299, dec 2013. doi: 10.1016/j.diin.2013.08.005. URL <https://doi.org/10.1016/j.diin.2013.08.005>.
75. Ben Martini, Quang Do, and Kim-Kwang Raymond Choo. Mobile cloud forensics. In *The Cloud Security Ecosystem*, pages 309–345. Elsevier, 2015. doi: 10.1016/b978-0-12-801595-7.00015-x. URL <https://doi.org/10.1016/b978-0-12-801595-7.00015-x>.
76. Farhood Norouzizadeh Dezfooli, Ali Dehghantanha, Ramlan Mahmoud, Nor Fazlida Binti Mohd Sani, and Solahuddin bin Shamsuddin. Volatile memory acquisition using backup for forensic investigation. In *Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*. IEEE, jun 2012. doi: 10.1109/cybersec.2012.6246108. URL <https://doi.org/10.1109/cybersec.2012.6246108>.
77. Teing Yee Yang, Ali Dehghantanha, Kim-Kwang Raymond Choo, and Zaiton Muda. Windows instant messaging app forensics: Facebook and skype as case studies. *PLOS ONE*, 11(3):e0150300, mar 2016. doi: 10.1371/journal.pone.0150300. URL <https://doi.org/10.1371/journal.pone.0150300>.
78. Darren Quick and Kim-Kwang Raymond Choo. Forensic collection of cloud storage data: Does the act of collection result in changes to the data or its metadata? *Digital Investigation*, 10(3):266–277, oct 2013. doi: 10.1016/j.diin.2013.07.001. URL <https://doi.org/10.1016/j.diin.2013.07.001>.
79. Ben Martini, Quang Do, and Kim-Kwang Raymond Choo. Mobile cloud forensics: An analysis of seven popular android apps. *CoRR*, abs/1506.05533, 2015. URL <http://arxiv.org/abs/1506.05533>.
80. Karen Kent, Suzanne Chevalier, Timothy Grance, and Hung Dang. Sp 800-86. guide to integrating forensic techniques into incident response. Technical report, NIST, Gaithersburg, MD, United States, 2006.
81. Rodney McKemmish. What is forensic computing?, 1999. URL <https://aic.gov.au/file/5923/download?token=KpelECKP>.
82. Mark Scanlon, Jason Farina, and M-Tahar Kechadi. BitTorrent sync: Network investigation methodology. In *2014 Ninth International Conference on Availability, Reliability and Security*. IEEE, sep 2014. doi: 10.1109/ares.2014.11. URL <https://doi.org/10.1109/ares.2014.11>.
83. Mark Scanlon, Jason Farina, Nhien-An Le-Khac, and M. Tahar Kechadi. Leveraging decentralization to extend the digital evidence acquisition window: Case study on bittorrent sync. *CoRR*, abs/1409.8486, 2014. URL <http://arxiv.org/abs/1409.8486>.

84. Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. A forensically sound adversary model for mobile devices. *PLOS ONE*, 10(9):e0138449, sep 2015. doi: 10.1371/journal.pone.0138449. URL <https://doi.org/10.1371/journal.pone.0138449>.
85. Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. Is the data on your wearable device secure? an android wear smartwatch case study. *Software: Practice and Experience*, 47(3):391–403, may 2016. doi: 10.1002/spe.2414. URL <https://doi.org/10.1002/spe.2414>.
86. Nurul Hidayah Ab Rahman, Niken Dwi Wahyu Cahyani, and Kim-Kwang Raymond Choo. Cloud incident handling and forensic-by-design: cloud storage as a case study. *Concurrency and Computation: Practice and Experience*, 29(14):e3868, may 2016. doi: 10.1002/cpe.3868. URL <https://doi.org/10.1002/cpe.3868>.
87. Brian Carrier. Open source digital forensics tools: The legal argument, 2002.
88. Home. <https://accessdata.com/>. (Accessed on 07/06/2018).
89. Niken Dwi Wahyu Cahyani, Ben Martini, and Kim-Kwang Raymond Choo. Using multimedia presentations to enhance the judiciary’s technical understanding of digital forensic concepts: An indonesian case study. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, jan 2016. doi: 10.1109/hicss.2016.695. URL <https://doi.org/10.1109/hicss.2016.695>.
90. Niken Dwi Wahyu Cahyani, Ben Martini, Kim-Kwang Raymond Choo, and AKBP Muhammad Nuh Al-Azhar. Forensic data acquisition from cloud-of-things devices: windows smartphones as a case study. *Concurrency and Computation: Practice and Experience*, 29(14):e3855, may 2016. doi: 10.1002/cpe.3855. URL <https://doi.org/10.1002/cpe.3855>.
91. Niken Dwi Wahyu Cahyani, Ben Martini, and Kim-Kwang Raymond Choo. Using multimedia presentations to enhance the judiciary’s technical understanding of digital forensic concepts: An indonesian case study. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, jan 2016. doi: 10.1109/hicss.2016.695. URL <https://doi.org/10.1109/hicss.2016.695>.
92. Niken Dwi Wahyu Cahyani, Ben Martini, and Kim-Kwang Raymond Choo. Effectiveness of multimedia presentations in improving understanding of technical terminologies and concepts: a pilot study. *Australian Journal of Forensic Sciences*, 49(1):106–122, jan 2016. doi: 10.1080/00450618.2015.1128968. URL <https://doi.org/10.1080/00450618.2015.1128968>.
93. Seafire - open source file sync and share software. <https://www.seafire.com/en/home/>. (Accessed on 07/06/2018).
94. Introduction · seafire server manual. <https://manual.seafire.com/>. (Accessed on 07/06/2018).
95. Security features · seafire server manual. https://manual.seafire.com/security/security_features.html. (Accessed on 07/06/2018).
96. Seafire components · seafire server manual. <https://manual.seafire.com/overview/components.html>. (Accessed on 07/06/2018).
97. Backup and recovery · seafire server manual. https://manual.seafire.com/maintain/backup_recovery.html. (Accessed on 07/06/2018).
98. Dongsong Zhang and Boonlit Adipat. Challenges, methodologies, and issues in the usability testing of mobile applications. *International Journal of Human-Computer Interaction*, 18(3):293–308, jul 2005. doi: 10.1207/s15327590ijhc1803_3. URL https://doi.org/10.1207/s15327590ijhc1803_3.
99. Does ccnet/mykey.peer need to be backed up? · issue #1044 · haiwen/seafire. <https://github.com/haiwen/seafire/issues/1044>. (Accessed on 07/06/2018).
100. Webrowserpassview - recover lost passwords stored in your web browser. http://www.nirsoft.net/utils/web_browser_password.html. (Accessed on 07/06/2018).
101. Sync algorithm · seafire server manual. https://manual.seafire.com/develop-sync_algorithm.html. (Accessed on 07/06/2018).
102. Web api · seafire server manual. https://manual.seafire.com/develop-web_api.html#get-library-history. (Accessed on 07/06/2018).
103. Web api · seafire server manual. https://manual.seafire.com/develop-web_api.html. (Accessed on 07/06/2018).

104. Web api v2.1 · seafile server manual. https://manual.seafile.com/develop/web_api_v2.1.html. (Accessed on 07/06/2018).
105. Python api · seafile server manual. https://manual.seafile.com/develop/python_api.html. (Accessed on 07/06/2018).
106. rene-s/seafile-php-sdk: This is a php package for accessing seafile web api. <https://github.com/rene-s/Seafile-PHP-SDK>. (Accessed on 07/06/2018).
107. Nginx docs — configuring logging. <https://docs.nginx.com/nginx/admin-guide/monitoring/logging/>. (Accessed on 07/06/2018).
108. Help center - seafile. https://www.seafile.com/en/help/history_setting/. (Accessed on 07/06/2018).
109. Migrate seafile from windows to linux · seafile server manual. https://manual.seafile.com/deploy_windows/migrate_from_win_to_linux.html. (Accessed on 07/06/2018).

Distributed Filesystem Forensics: Ceph as a Case Study



Krzysztof Nagrabski, Michael Hopkins, Milda Petraityte, Ali Dehghantanha, Reza M. Parizi, Gregory Epiphanou, and Mohammad Hammoudeh

Abstract Cloud computing is becoming increasingly popular mainly because it offers more affordable technology and software solutions to start-ups and small and medium enterprises (SMEs). Depending on the business requirements there are various Cloud solution providers and services, yet because of this it becomes increasingly difficult for a digital investigator to collect and analyse all the relevant data when there is a need. Due to the complexity and increasing amounts of data, forensic investigation of Cloud is turning into a very complex and laborious endeavour. Ceph is a filesystem that provides a very high availability and data self-healing features, which ensure that data is always accessible without getting damaged or lost. Because of such features, Ceph is becoming a favourite file system for many cloud service providers. Hence, understanding the remnants of malicious

K. Nagrabski · M. Hopkins · M. Petraityte

School of Computing, Science, and Engineering, University of Salford, Manchester, UK

e-mail: k.j.nagrabski@edu.salford.ac.uk; michael.hopkins@project-clarity.com;

Milda.Petraityte@kpmg.co.uk

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

e-mail: ali@cybersciencelab.org

R. M. Parizi

Department of Software Engineering and Game Development, Kennesaw State University, Marietta, GA, USA

e-mail: rparizi1@kennesaw.edu

G. Epiphanou

Wolverhampton Cyber Research Institute (WCRI), School of Mathematics and Computer Science, University of Wolverhampton, Wolverhampton, UK

e-mail: G.Epiphanou@wlv.ac.uk

M. Hammoudeh

School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester, UK

e-mail: M.Hammoudeh@mmu.ac.uk

users activities is become a priority in Ceph file system. In this paper, we are presenting residual evidences of users' activities on Ceph file system on Linux Ubuntu 12.4 operating system and discuss the forensics relevance and importance of detected evidences. This research follows a well-known cloud forensics framework in collection, preservation and analysis of CephFS remnants on both client and server sides.

Keywords Ceph · Cloud forensics · Cloud storage · Investigative framework · Metadata · Data analysis

1 Introduction

The emergence of the Cloud technology and its services offers an effective solution for many companies [1]. The amount of data that organisations deal with increases rapidly. It is becoming more and more complicated to keep up to speed with various technological solutions for organisations that want to focus on developing their business instead of investing in technology. Therefore, Cloud services are likely to become the next thriving technology business of the twenty-first century [2, 3].

However, because of its many functions and broad capabilities, Cloud may become an attractive tool for cybercriminals who may use any of its services to launch attacks over legitimate businesses [4, 5]. Various individuals as well as organised crime groups are also likely to breach Cloud services used by legitimate businesses in attempt to steal valuable data or disrupt their services [6, 7].

The flexibility and agile environment of the Cloud as well as the previously mentioned criminal activity introduced some problems for forensic investigators, these can be summarised as:

1. *Cloud is not forensics-friendly.* Since it is a relatively new feature it does not have built-in forensics capabilities [8]. The data may be stored in various locations and accessed by multiple devices and users that forensic investigators may have no access to [9].
2. *Cloud forensics is fundamentally different from other forensic investigations.* The tools, approach and training required is very different in other cases [10]. Frameworks for cyber investigations are essential and it is imperative to use them at all times while carrying out an investigation, yet Cloud forensics is a new concept and the amount of various Cloud services is rapidly growing [2, 6].

From a forensics investigator point of view, investigation of Infrastructure-as-a-Service (IaaS) will prove to be most beneficial as it includes Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) investigation as well [11]. Ceph is presented as “a distributed object store and file system designed to provide excellent performance, reliability and scalability” [12]. Moreover, it provides object, block and file system storage in a single platform and is based on a Linux kernel. Ceph is idea for managing large amounts of data that can be easily retrieved. Among many

of its advantages is its ability to self-manage and self-heal, which means that data will never be lost or damaged. Unified storage results in no silos of data and only one system must be scaled meaning that the system capacity is easier to manage. Moreover, Ceph is open source and extremely expandable. It consists of three main components: DIRectory service (DIR) or the client, Object Storage Device (OSD), and a metadata server.

The main goal of this research is to pursue the following objectives:

1. Examine what kind of configuration files, volatile and non-volatile data can be retrieved from DIR, OSD and metadata server.
2. Propose processes for the collection of electronic evidence from Ceph.
3. Validate the applicability of the proposed model by B. Martini and R. Choo [2] to a forensic investigation of the Ceph distributed file system.

The rest of the paper is organised as follows. Section 2 overviews the related work on previous researches and analysis of relevant existing issues in IaaS forensics. Section 3 describes the investigative framework that is devised for the investigation of Ceph file system. The environment and experimental setup are explained in Sect. 4. Section 5 reports and analyses the results and findings. Lastly, Sect. 6 gives the conclusion and provides recommendation for future research.

2 Related Work

The concept of Cloud forensics is to assist the forensic community to create suitable approaches and frameworks to analyse various Cloud connected services, data stored on the Cloud and devices connected to the cloud infrastructure [13]. Attempts to cover various fields and techniques in Cloud forensics have contributed to the body of knowledge in the field [14]. Cloud offers increasingly more services and there is a growing number of devices connected to the Internet. It is safe to predict that with the dawning era of the Internet of Things (IoT) and the Internet of Everything (IoE) there will be even more devices and tools that generate big data [15, 16], which further complicates forensics challenges [17, 18]. The increasing use of Cloud services to host all these data inevitably presents digital investigators with a growing amount of data for a forensic investigation [19]. This is likely to become the main problem of services that are specifically meant for processing or storage of big data, and Ceph is exactly this kind of service [20]. Moreover, emergence of new vectors of attack such as Distributed Denial of Services [10], Ransomware [21], malware [15, 22, 23] and Trojans [24] further complicated cloud forensics activities.

Out of all the researches published so far, only a handful of studies [2, 25–27] currently exist in the area of distributed filesystem forensics. It is evident in the literature that the majority of existing research focus on client side analysis [28–30]. One explanation may be the difficulty in getting cloud server logs for examining historical user actions [31]. Service providers do not want to release this information for their client's protection as logs may provide information across multiple tenants [32].

Some researchers have already suggested the concepts of digital Forensics-as-a-Service (FaaS), where forensic investigation would be carried out by certain software, ridding an investigator of time consuming processing of terabytes and petabytes of information [6, 33]. No doubt, it will be beneficial for the necessary data crunching and forensic data storage, however it is hard to predict if the currently available software will satisfy the requirements of a forensics investigation where the environment is rapidly changing [34].

Creating frameworks and methodologies for the investigation of Cloud services and devices available will require significant efforts in comparison to the speed that the new technology is created and introduced to the consumer market. No doubt, a unified approach and framework is something that forensic investigators should be looking for [35] as it will not only cover all possible areas where valuable information could be found, but also help to make sure that data is collected in the order of volatility and can be presented in the court of law in a credible manner [36]. There have been attempts to create high level framework for Cloud forensics, specifically a more detailed framework has been introduced by B. Martini and R. Choo [2] as part of their investigation of XtreemeFS. This research aims to validate their framework by carrying out an empirical investigation of CephFS according to their methodology, this way supporting the emergence of a broadly accepted cloud forensics investigation framework.

3 Cloud Forensics Framework

The methodology used in the work follows the four Stage Digital Forensics Framework defined by Ben Martini and Raymond Choo, based upon prior work by Rodney McKemmish and NIST [2, 37–39]. The four iterative stages of this framework being:

1. *Evidence Source Identification and Preservation.* The primary objective of this stage is to identify sources of evidence. The main sources of information will be the client system, either stored locally on physical devices or on server nodes, which would form part of the back-end service infrastructure.
2. *Collection.* This stage will involve collection of data identified in stage one above, from locations such as memory, virtual and physical disks. The disparate deployment model means that this phase can be more complex with a distributed file system than it would be with other models (such as client server).
3. *Examination and Analysis.* The examination phase will glean an understanding of the component parts of the file system and their inter-relationships. Analysis will provide the intelligence required to achieve reconstruction of the data collected in stage two, which should facilitate forensically sound recreation of the original user data.

4. *Reporting and Presentation.* Whilst not intended for legal scrutiny, we plan to document all processes tools and applications used, together with any known limitations.

For identifying and collecting evidence from the Ceph file system, we follow the method proposed by Martini and Choo [2], which comprises the following stages:

3.1 Stage 1: The Location, Collection and Examination of DIR

In a distributed file system, Directory Services typically fulfil a registry role for all volumes and services included within the system, which necessitates communication with all other parts of the service. This makes it a logical source of forensic information about the logical and physical components of the file system. In this stage, we will be gathering metadata from the Ceph file systems Directory Service(s) to gain an initial understanding of the operation and the location of the file system nodes, which will involve:

1. Identifying the location(s) of the directory services
2. Preserving and collecting data from these locations
3. Analysis of the Directory service metadata data to identify
 - (a) The file system components being utilised by Ceph such as stored data, metadata and clusters architecture
 - (b) The locations of these file system components
4. Post-analysis documentation to support any subsequent analysis, reporting and enquiry related to provenance and chain of custody. In this stage, details for each of the component nodes, such as IP addresses and unique id's and node type, are recorded.

It is essential to note that prior to commencing the investigation, we did not know whether the Directory Services for Ceph are stored within a centralised location, such as dedicated directory server(s), or distributed across multiple peer to peer nodes within the system.

3.2 Stage 2: The Location, Collection and Examination of Metadata Cluster

Within distributed file systems, Metadata (all types of data other than file content), are commonly stored in a Metadata Replica Catalogue (MRC) [1]. The MRC is therefore likely to contain metadata related to the volumes including filenames, ownership and permissions, access and modification information.

Guided by our analysis of Directory Services, we will then have an understanding of the Metadata available within the Ceph environment, and they will be in a position to identify potential sources of interest for collection. In order to minimise noise and keep this stage of the investigation to manageable proportions, the identification and collection process will be focused upon persons or artefacts of interest; identified from within the test data set used for the investigation process. This may involve consideration of permissions and ownership data and we intend to pay particular regard to temporal data. The steps involved in this stage include:

1. Locating any server(s) storing metadata
2. Collection of the identified metadata from these locations
3. Analysis the collected metadata to identify items (files, directories or volumes) of interest to forensic investigation

3.3 Stage 3: The Location, Collection and Examination of OSD

The Object Storage Devices within a distributed file system store the file content data provided by clients. This last stage is one of data collection and re-assembly of data within the Object Storage Devices, guided by what we have learned from our analysis of the Directory Services and Metadata Storage, with a view to understanding the extent and content of the data originally stored by the user. Given the nature of distributed file systems, which typically store replicated and fragmented files across disparate locations to maintain availability, this stage is likely to include:

1. The location, selection and collection of those specific devices or logical datasets containing data of interest to our investigation
2. Potential reassembly of any fragmented data sets into the original source files

We intend to pay particular attention to forensic preservation techniques, the analysis of temporal data and log collection. Consideration will also be given on whether to make logical collection from client nodes or physical collection from the underlying data repositories, which is likely to require file re-assembly to recreate the original user data.

4 Environment and Experimental Setup

4.1 Ceph Overview

Ceph is an open source, scalable, software-defined storage system [40] in 2006, as an alternative to existing data storage projects. Its main goals are to deliver object,

block, and file storage in one unified system, which is highly reliable and easy to manage. Ceph is characterised by its high scalability without a single point of failure and high fault tolerance, due to the fact that all data are replicated.

4.2 *Ceph Filesystem*

The Ceph file system, is also known as CephFS and it is a POSIX-compliant file system that uses the Ceph storage cluster to store data. CephFS provides increased performance and reliability as its store data and metadata separately.

Ceph file system library (libcephfs) works on top of the RADOS library (librados), which is the Ceph storage cluster protocol, and is common for file, block, and object storage. CephFS requires at least one Ceph metadata server (MDS), which is responsible for storage of metadata to be configured on any cluster node.

It is worth of highlighting the fact that CephFS is the only component of the Ceph storage system, which is not currently production-ready.

4.3 *Ceph Storage Cluster*

The Ceph Storage Cluster is the main part of each Ceph deployment and is based on RADOS (Reliable Autonomic Distributed Object Store). Ceph Storage Clusters consist of two types of daemons: a Ceph OSD Daemon (OSD) and a Ceph Monitor (MON). There is no limit to the number Ceph OSD's, but it is recommended to have at least one Ceph Monitor and two Ceph OSD Daemons for data replication. Ceph OSD Daemon is responsible for storing user data. It also handles data replication, recovery, backfilling and rebalancing. It also provides monitoring information to Ceph Monitors by checking other Ceph OSD Daemons for a heartbeat. Ceph Monitor maintains maps of the cluster state, including the monitor map, the OSD map, the Placement Group (PG) map, and the CRUSH map. Ceph maintains a history of each state change in the Ceph Monitors, Ceph OSD Daemons and PGs.

4.4 *Environment Configuration*

The experiment environment consists of number of virtual machines providing 2 OSD's, 1 Monitor and MDS node, 1 Administration node for cluster and 1 Client block device. Individual nodes has been configured and tested in accordance with Ceph values. Table 1 presents the environment configuration used in our forensic research.

Table 1 Environment configuration

Ceph node configuration	Environment
Operating system	Linux Ubuntu 12.4
Ceph distribution	Giant

Table 2 Utilised forensics tools

Used software	Version	Function
HxD Hexeditor	1.7.7.0	Helped to see text in hex
CapLoader	1.2	Used for carving out information from memory files
Wireshark	1.12.4	Used for network traffic analysis
Vmware Workstation	11	Used to collect the disk images
OSFMount	1.5	Used to mount the images as physical drives on the computers.
Ext2 Volume Manager	0.53	Helped to read Linux drive on Windows machines
OSForensics	3.1	Used for majority of DIR, Metadata and OSF investigations
Autopsy	3.1.2	Used for investigation of DIR and the way data is distributed across the filesystem
Meld	3.12.3	Used for GUI comparison of disks

Table 2 presents the software used in the extraction, conversion and analysis of VMDK, VMEM and PCAP files.

5 Findings and Analysis

Following the methodology, the forensic investigation was performed on five nodes. Six snapshots were taken for each node having performed several actions:

1. Node set up
2. Data copied to cluster
3. Data copied from Filesystem to a local folder
4. Data moved to cluster
5. Data moved from Filesystem to a local folder
6. Ceph removed

A VMware snapshot of each virtual machine was taken following completion of every activity. In order to examine the drives, we then recreated the 30 disk drives (VMDKs) for each node as they would have been at the point of snapshot creation (using VMware's 11 utility). For each of the 30 drives, we then generated hashes to identify where changes had occurred on each respective. A comparison of these hashes using a signature comparison highlighted modified, deleted and new files between any two virtual machine states. The drives were then mounted in read only mode and examined using hex editor, file comparison tools and forensic frameworks.

The examination revealed information about:

1. The configuration of our Ceph cluster
2. The parts of the local drives affected during normal Ceph operation
3. The locations in which user data is stored within a Ceph cluster
4. The ability to read and recover data from within the Object Storage Devices
5. The Object Storage Devices themselves
6. The location of Metadata

This procedure provided a better insight into how Ceph stores data and how it moves around the Filesystem when a variety of actions are performed there. We believe that these various actions should cause the system to behave in a certain way and leave traces of system information that could be valuable for a forensic investigator. It is also a way to identify whether the suggested framework can be applied to every file system or is there any variation and to what extent. Framework recommends paying attention to the three kinds of data, while conducting the investigation that could be of interest across all the parts of a file system, including, volatile environment metadata, non-volatile environment data and configuration files.

5.1 Directory Services

Directory services were detected by performing a search of original document's titles on raw data across all nodes (Fig. 1) using OSForensics software that allowed

The screenshot shows the OSForensics Raw Disk Viewer interface. The main window title is "Raw Disk Viewer". In the top bar, there is a "Disk" dropdown set to "node1-cl2-000011-0: [Image File]" and a "Config..." button. Below the title bar, there is a "Search" button and a search pattern input field containing "63097.txt".

The search options panel on the left includes radio buttons for "Hex" and "Text", and checkboxes for "ASCII", "UTF-8", and "Unicode", along with "Match case", "Wild character (?)", and "Regular expressions". The results options panel shows "Max results: 1000".

The "Search Results:" table has columns: Byte Offset, Context, Encoding, Sector, Partition, LCN, File, and Object Type. The results show several entries, with the last one highlighted in blue:

Byte Offset	Context	Encoding	Sector	Partition	LCN	File	Object Type
0x20A3CD0C0	Ig- 6 63097.txt Ig- 4s	ASCII	17112680	0	2139085	Inode 8	File
0x20A4F10D4	Ig- 4s 63097.txt	ASCII	17115016	0	2139377	Inode 8	File
0x4028070C0	Ig- 6 63097.txt Ig- 4s	ASCII	33636408	0	4204551	node1-cl2-000011-0:\home\ceph\venron\	Directory
0x4028090D4	Kg- 4s 63097.txt	ASCII	33636416	0	4204552	node1-cl2-000011-0:\home\ceph\venron\1\	Directory

Below the search results table, there is a large preview pane showing the raw hex and ASCII data of the file "63097.txt". The preview pane has scroll bars and a status bar at the bottom.

Fig. 1 File title search within raw data on *OSForensics*

a quick detection of the original data. Moreover, it showed that different snapshots bring up different results and this way it was possible to see what happens to the data as it is copied, moved or deleted. This will be covered in more detail while explaining the findings about OSD.

The framework suggests that in DIR we should be looking to find IP addresses that accessed the data after the installation of the file system, the Unique User Identifiers (UUID) and the Service Registry and Configuration files. It was possible to identify the IP addresses of the nodes that were used for the configuration of Ceph.

With CapLoader, we were able to see the IP addresses as well as ports and hosts used within Ceph cluster. Moreover, we were able to recover data associated with the copy, move and delete operations such as the name of the file and its content and this will be detailed in the last section on OSD.

The results with *Autopsy* showed that there are multiple retrievable IP addresses, email addresses and URLs remaining on Ceph. The tool also shows the image snapshots these addresses remained on or their correlation with the actions performed on the original data. This is particularly interesting because in our case the snapshot –000015.vmdk was the snapshot after files have been removed from Ceph and all data should have been removed.

In addition to this, the original data have been discovered on one node, while the email addresses were found in each disk. This gave us an understanding that the contents of the data in the original files will not be stored in one location and therefore are spread across multiple nodes. As a consequence, we made an assumption that this was what happened with all of the data, as well as metadata, and we tried to prove or reject our assumption as we went along investigating Ceph.

We have also collected several .VMEM memory files from each node of Ceph cluster and a client machine. We have used HxDHexeditor 1.7.7.0 to investigate captured memory files, where we conducted keyword searches to locate Cephcredentials, the dataset and keywords like “Ceph”, “OSD”, “MDS”. When searching for Ceph username and password, we have managed to locate a username and user details, but no password (Fig. 2).

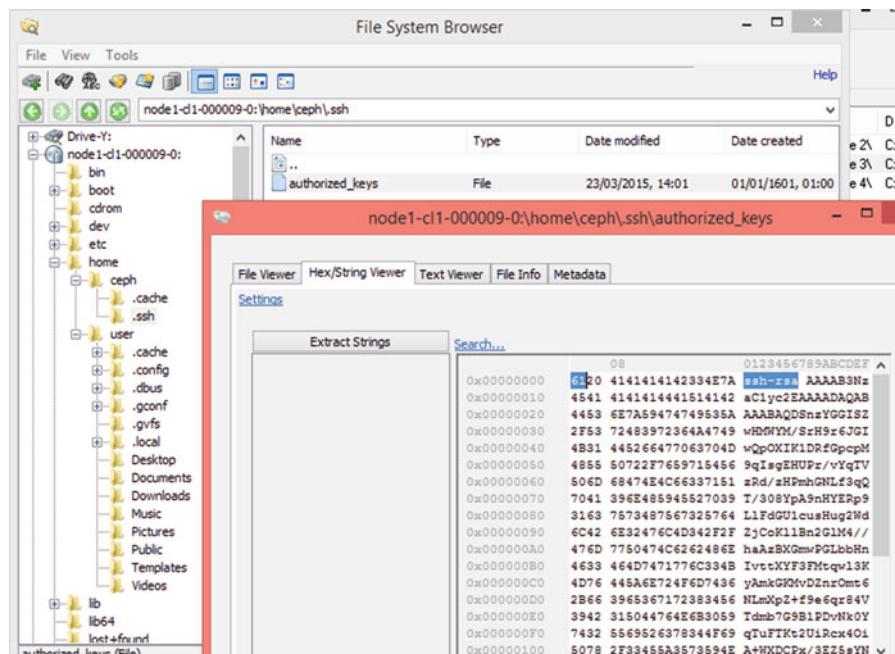
Martini and Choo [2] also suggest that the service registry data which is of interest resides within one main directory. However this is not the case with Ceph. Having analysed the signatures of each image and compared them manually, we discovered that there are two main directories together with their subdirectories where most of the changes have been identified within all of the nodes, i.e., */etc* and */var*.

As researchers have identified, */var* may contain some valuable logging data and should be analysed. In the case of XtreemFS it did not provide any valuable information. However, in the case of Ceph it contains the metadata as well as OSD details and this will be discussed in the next section. */etc* normally contains configuration files for programs that run on Linux. We were also able to determine that network connection files and the details in these two directories are replicated across all of the nodes.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
1533D610	08	01	67	00	05	61	7B	73	76	7D	00	00	00	00	00	00	
1533D620	05	01	75	00	25	00	00	00	06	01	73	00	05	00	00	00	
1533D630	3A	31	2E	32	36	00	00	00	07	01	73	00	05	00	00	00	
1533D640	3A	31	2E	31	38	00	00	00	E8	02	00	00	00	00	00	00	
1533D650	03	00	00	00	55	69	64	00	01	74	00	00	00	00	00	00	
1533D660	E9	03	00	00	00	00	00	00	08	00	00	00	55	73	65	72	
1533D670	4E	61	6D	65	00	01	73	00	04	00	00	00	63	65	70	68	
1533D680	00	00	00	00	00	00	00	00	08	00	00	00	52	65	61	60	
1533D690	4E	61	6D	65	00	01	73	00	00	00	00	00	00	00	00	00	
1533D6A0	0B	00	00	00	41	63	63	6F	75	6E	74	54	79	70	65	00	
1533D6B0	01	69	00	00	00	00	00	00	0D	00	00	00	48	6F	6D	65	
1533D6C0	44	69	72	65	63	74	6F	72	79	00	01	73	00	00	00	00	
1533D6D0	0A	00	00	00	2F	68	6F	6D	65	2F	63	65	70	68	00	00	
1533D6E0	05	00	00	00	00	53	68	65	6C	00	01	73	00	00	00	00	
1533D6F0	00	00	00	00	00	00	00	00	00	05	00	00	00	45	6D	61	69
1533D700	6C	00	01	73	00	00	00	00	00	00	00	00	00	00	00	00	
1533D710	08	00	00	00	4C	61	6E	67	75	61	67	65	00	01	73	00	
1533D720	05	00	00	00	65	6E	5F	55	53	00	00	00	00	00	00	00	
1533D730	0D	00	00	00	46	6F	72	6D	61	74	73	4C	6F	63	61	6C	

Fig. 2 Ceph user credentials

Moreover, there was a number of hidden files starting with “.” created across other directories that contain data of interest, such as `/home/ceph` and `/home/user` (Fig. 3). The data on the first one is backed up on two nodes and on those two nodes the folder structure is similar to the latter one. The nodes that do not have the hidden files in `/home/ceph` only contain the SSH key. Where the directory contains the hidden files it also contains SSH config file that displays hosts and user names (Fig. 4).

**Fig. 3** The folder structure and SSH key on OSForensics

	00	08	0123456789ABCDEF
0x00000000	486F7374206E6F64	65310A0A2020486F	Host node1.. Ho
0x00000010	73746E616D65206E	6F6465310A0A2020	stname node1..
0x00000020	5573657220636570	680A0A486F737420	User ceph..Host
0x00000030	6E6F6465320A0A20	20486F73746E616D	node2.. Hostnam
0x00000040	65206E6F6465320A	0A20205573657220	e node2.. User
0x00000050	636570680A0A486F	7374206E6F646533	ceph..Host node3
0x00000060	0A0A2020486F7374	6E616D65206E6F64	.. Hostname nod
0x00000070	65330A0A20205573	657220636570680A	e3.. User ceph.
0x00000080	0A486F7374206E6F	6465340A0A202048	.Host node4.. H
0x00000090	6F73746E616D6520	6E6F6465340A0A20	ostname node4..
0x000000A0	2055736572206365	70680A0A486F7374	User ceph..Host
0x000000B0	2061646D696E2D6E	6F64650A0A202048	admin-node.. H
0x000000C0	6F73746E616D6520	61646D696E2D6E6F	ostname admin-no
0x000000D0	64650A0A20205573	657220636570680A	de.. User ceph.

Fig. 4 Hosts and user names displayed in SSH config file

As it was mentioned above, */etc* contains configuration files. We were able to discover the system configuration details in these files (Figs. 5 and 6).

```

0304EFDO 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
0304FEO 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
0304EFF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
0304F000 5B 67 6C 6F 62 61 6C 5D 0A 66 73 69 64 20 3D 20 [global].fsid =
0304F010 36 38 34 37 33 66 61 31 2D 62 37 30 64 2D 34 38 68473fa1-b70d-48
0304F020 37 30 2D 61 33 39 30 2D 39 61 66 64 39 62 3b 66 70-a390-9afdf9bf
0304F030 66 33 34 38 0A 6D 6F 6E 5F 69 6E 69 74 69 61 6C f348.mon_initial
0304F040 5F 6D 65 6D 62 65 72 73 20 3D 20 6E 6F 64 65 31 _members = node1
0304F050 0A 6D 6F 6E 5F 68 6F 73 74 20 3D 20 31 37 32 2E .mon_host = 172.
0304F060 31 36 2E 32 30 39 2E 33 31 0A 61 75 74 68 5F 63 16.209.31.auth_c
0304F070 6C 75 73 74 65 72 5F 72 65 71 75 69 72 65 64 20 luster_required
0304F080 3D 20 63 65 70 68 78 0A 61 75 74 68 5F 73 65 72 = cephx.auth_ser
0304F090 76 69 63 65 5F 72 65 71 75 69 72 65 64 20 3D 20 vice_required =
0304FOAO 63 65 70 68 78 0A 61 75 74 68 5F 63 6C 69 65 6E cephx.auth_clien
0304FOBO 74 5F 72 65 71 75 69 72 65 64 20 3D 20 63 65 70 t_required = cep
0304FOCO 68 78 0A 66 69 6C 65 73 74 6F 72 65 5F 78 61 74 hx.filestore_xat
0304FODO 74 72 5F 75 73 65 5F 6F 6D 61 70 20 3D 20 74 72 tr_use_omap = tr
0304FOEO 75 65 0A 6F 73 64 5F 70 6F 6F 6C 5F 64 65 66 61 ue.osd_pool_defa
0304FOFO 75 6C 74 5F 73 69 7A 65 20 3D 20 32 0A 6F 73 64 ult_size = 2.osd
0304F100 5F 70 6F 6F 6C 5F 64 65 66 61 75 6C 74 5F 6D 69 _pool_default_mi
0304F110 6E 5F 73 69 7A 65 20 3D 20 32 0A 6F 73 64 5F 6A n_size = 2.osd_j
0304F120 6F 75 72 6E 61 6C 5F 73 69 7A 65 20 3D 20 34 30 ournal_size = 40
0304F130 30 30 0A 6F 73 64 5F 70 6F 6F 6C 5F 64 65 66 61 00.osd_pool_defa
0304F140 75 6C 74 5F 70 67 5F 6E 75 6D 20 3D 20 31 32 38 ult_pg_num = 128
0304F150 0A 70 75 62 6C 69 63 5F 6E 65 74 77 6F 72 6B 20 .public_network
0304F160 3D 20 31 37 32 2E 31 36 2E 32 30 39 2E 30 2F 31 = 172.16.209.0/1
0304F170 36 0A 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6..... .
0304F180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
0304F190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .

```

Fig. 5 Ceph config file content

00912A10	08 00 00 00 EC 7F 00 00 F0 47 16 04 00 00 00 001...8G.....
00912A20	60 3C 73 72 EC 7F 00 00 66 00 00 00 00 00 00 00	`<sri...f.....
00912A30	98 45 16 04 00 00 00 00 F1 5A 50 72 EC 7F 00 00	"E.....ñZPri...
00912A40	00 00 00 00 02 00 00 00 80 0C 00 00 00 00 00 00€.....
00912A50	63 65 70 68 20 76 30 32 37 48 16 04 00 00 00 00	ceph v027H.....
00912A60	00 00 00 00 C2 08 00 00 02 00 1A 90 AC 10 D1 21ñ.....-ñ!
00912A70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00912A80	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00912A90	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Fig. 6 Ceph version

5.2 Metadata Cluster

The forensics framework of interest identifies one location where metadata information could be stored on a file system and gives a variety of data to look for. However, Ceph seems to be sharing several folders within `/var` directory with OSD. It required a manual process of looking through the directories and files to be able to discover metadata as it seems to be spread across several folders. However, this snippet of data is available on all nodes probably because of the nature of the system itself as the data are called by monitors on each node and they must know where it is located.

These directories contained monitor log data as shown in Fig. 7 as well as log data of when a monitor was called and by which node (Fig. 8):

For a long time we were unable to discover the database where metadata is stored because the location was not that obvious: `/var/lib/ceph/mon/ceph-node3/store.db/`.

The screenshot shows a file system browser interface with two main panes. The left pane displays the directory structure under `/var`, specifically the `node1-c1-d1-000009-0:/var` path. The right pane shows the contents of the `ceph-mon.log` file. The log file contains the following entries:

```

2015-03-28 22:24:55.698934 7149990028c0 0 ceph version 0.87.1 (283c2e7fcfa2c57799f534744d7d54bf83ea1395e), process ceph-mon, pid 19539
2015-03-28 22:24:56.102855 7149990028c0 0 mon.nodes does not exist in monmap, will attempt to join an existing cluster
2015-03-28 22:24:56.103341 7149990028c0 0 using public_addr 172.16.209.33:6789/0
2015-03-28 22:24:56.103341 7149990028c0 0 starting w/ nodeid -1 at 172.16.209.33:6789/0, mon data /var/lib/ceph/mon/ceph-node3 fsid 684793fa1-b7d0-4309-a559-189843749899
2015-03-28 22:24:56.188094 7149990028c0 1 mon-node3=1(logging) e0 initial_members_node1, filtering seed monmap
2015-03-28 22:24:56.204555 7149990028c0 0 -- 172.16.209.33:6789/0 >> 0.0.0.0:0/0 pipe(0x47edc0) sd:12 st:1 psc:0 csc:0 l:0 c:0x4791b80).fault
2015-03-28 22:24:56.492349 7149990028c0 0 log_channel(audit) log [DB6] : from:'admin socket' entity:'admin socket' cmd='mon_status' args='[]'; finished
2015-03-28 22:24:56.532123 71499129700 1 mon-node3=1(synchronizing).paxosservice(oepap 1..57) refresh upgraded, format 0 -> 1
2015-03-28 22:24:56.532183 71499129700 1 mon-node3=1(synchronizing).px v0 on_upgrade discarding in-core PGMap
2015-03-28 22:24:56.547695 71499129700 0 mon-node3=1(synchronizing).ads el print_main

```

Fig. 7 Folders that contain metadata with monitor log data

Fig. 8 Monitor election in metadata logs

One of the folders contained information on permission levels (Fig. 9).

Fig. 9 Permission levels on Ceph

As it was mentioned before, the network information is stored in the `/etc/networkmanager` which is outside of the `/var` directory. There is also VPN information available (Fig. 10) and full network details as well (Fig. 11).

String Address	String Value
0x00000000	SB6504E20436FEE
0x00000010	0A6E616D653D7070
0x00000020	63653D6F72672E66
0x00000030	6F702E4E6574776F
0x00000040	722E7074700A70
0x00000050	7573722F6C69622F
0x00000060	616E616765722FEE
0x00000070	6572669365650A0A
0x00000080	617574682D646961
0x00000090	2F6C69622F426574
0x000000A0	6765722F6E6D2D70
0x000000B0	2D6469616C6F670A
0x000000C0	65733D2F7573722F
0x000000D0	6F726B4D616E6167
0x000000E0	2D707074702D7072
0x000000F0	0A737570706F7274
0x00000100	616C2D75692DDE6F
0x00000110	64653D7472756560A
0x00000120	al-ui-mode=true.

Fig. 10 VPN connection information with PPTP protocol listed

String Address	String Value
0x00000000	SB3830322D332D65
0x00000010	0A6475706C65783D
0x00000020	2D61646472657373
0x00000030	3D30303A35303A35
0x00000040	363A33343A383363A
0x00000050	34300AOA5B636FEE
0x00000060	6E656374696F6E5D
0x00000070	0A69643D57697265
0x00000080	0E642063F6E6E563
0x00000090	74656F726E65745D
0x000000A0	[002-3-ethernet]
0x000000B0	.duplex=full.mac
0x000000C0	.address=00:50:5
0x000000D0	.connection=1.u
0x000000E0	.uid=6092ccb8c-dc3
0x000000F0	.f=4-fed-bbdf-cbeb
0x00000100	.type=80
0x00000110	.2-3-ethernet.tim
0x00000120	.estamp=142711037
0x00000130	.1..[ipv6].method
0x00000140	.auto..[ipv4].me
0x00000150	.thod=manual.dns=
0x00000160	.172.16.209.2;.ad
0x00000170	.dress1=172.16.20
0x00000180	.9.34/16,172.16.2
0x00000190	.09.2.

Fig. 11 Network information on /etc

Wireshark monitoring showed that the search on “Ceph” resulted in a information relating to the TCP connection banner protocol used by Ceph. This action appears after the protocol handshake confirming that the server and client nodes are both taking Ceph. It is likely that Ceph uses an authentication procedure when accessing data as SSH keys have been discovered in the */etc* directory. It is possible that the system is similar to Kerberos as it uses encryption for authentication (Fig. 12).

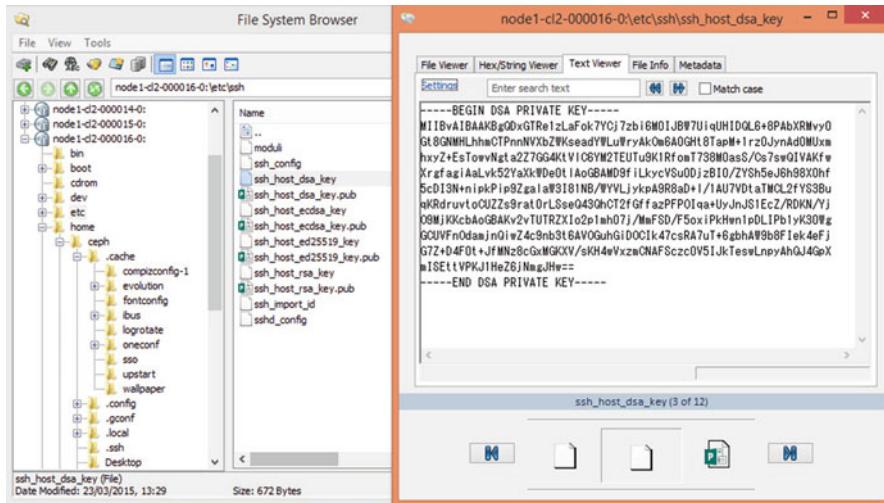


Fig. 12 Ceph authentication private key

5.3 Object Storage Device

The proposed framework was not very helpful in trying to identify how Ceph stores its data or where it is actually located. The framework rather addresses what is applicable for XtreemFS, but Ceph seems to be behaving in a slightly different way. The way Ceph distributes the data and replicates it, does not seem to fit into any type of RAID levels. It stripes the data like RAID 0, yet the data do not take physical space and is not shared or distributed on any other disk. Data objects are stored on one disk, yet the system registers each file in so called journals using CRUSH algorithm. Objects are assigned to individual OSDs using CRUSH, which is defined by a CRUSH map and determined at the point of installation.

The distributed nature of the CRUSH algorithm is designed to remove a single point of failure and is distributed across the nodes within the cluster. CRUSH is predictable meaning that, given certain inputs (a map and a placement group), any

node with access to the CRUSH algorithm can calculate the names and locations of the individual objects, which comprise a file loaded to the Ceph File system by a given user(s). To locate an object within the Ceph filesystem, CRUSH requires a placement group and an Object Storage Device cluster map. This eliminates the need for the distribution of location based metadata across nodes within the cluster and the requirement for a central single metadata repository, which would form a bottleneck.

Following the copy of data to the cluster, the search within raw data for message ID (10028279.1075849274084 from file 124367.txt) reveals two locations on the drive where user data is stored:

1. In a “free block”. In this example within sector 1086152, which is most likely to be the Ceph object.
2. Within a journal file stored in `var\local\osd\journal`.

Selecting and opening a Ceph object from such search displays the message in plaintext (Fig. 13). However, the original document is not stored on the node where such search was performed and could be replicated with other nodes as well as during the network traffic analysis using Wireshark (Fig. 14) or Autopsy (Fig. 15). Autopsy also reveals more information about data removal, if any. Figure 15 shows that the same message has been registered on four snapshots when data has been moved around, yet it was also found on a snapshot –000015, which was taken after data has been removed from the filesystem. This suggests that data is actually not fully removed unless Ceph is removed as well.

Byte Offset	Content	Encoding	Sector	Partition	LCN	File	Object Type
0x0000000000000000	age ID: 10028279.1075849274084	ASCII	1086152	0	1086152		Free block
0xABC76000	age ID: 10028279.1075849274084	ASCII	5530544	0	691318	F:\var\local\osd\journal\124367.txt	File

Fig. 13 Message visible in plain text if stored in a block/Ceph object

Searches on both of the profiles created (Ceph or user as credentials) did not return any results. This was expected as credentials are encrypted. Close

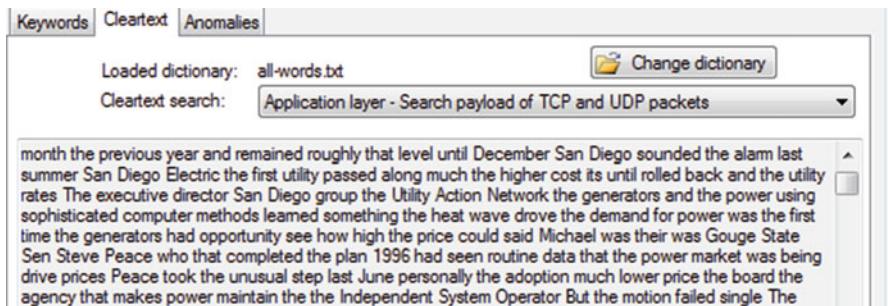


Fig. 14 Plain text data via *Network Miner*

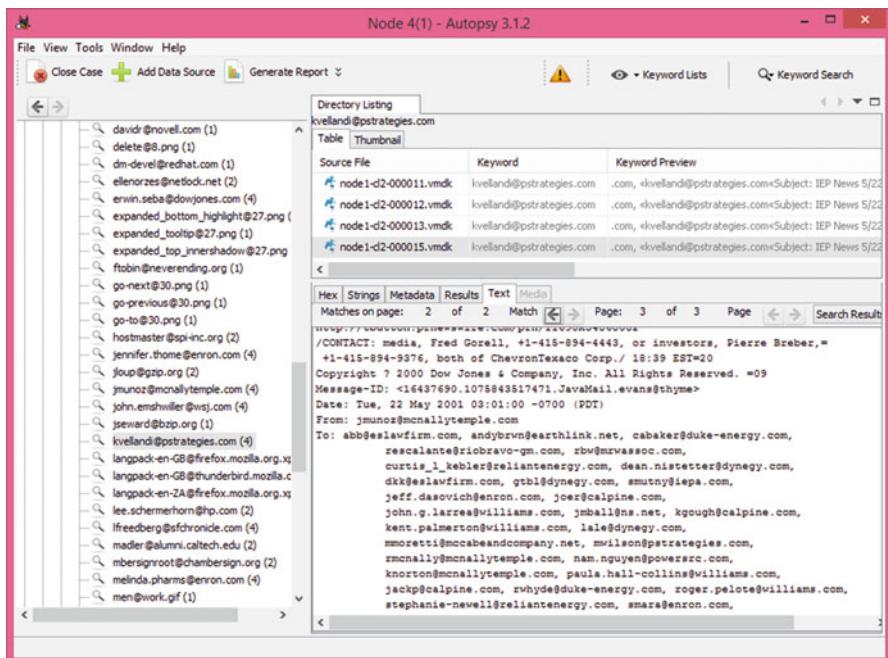


Fig. 15 Message displayed in plaintext on *Autopsy*

to the message there is an ID of a RADOS object within which the data is stored (see Fig. 16). It appears as a certain stamp or header according to which monitors can look-up and identify the data that is there and extract it when needed.

Search for RADOS block ID identified several types of entries, mainly within the journal file that is located in the `\var\local\osd\current\omap`, which seems to be

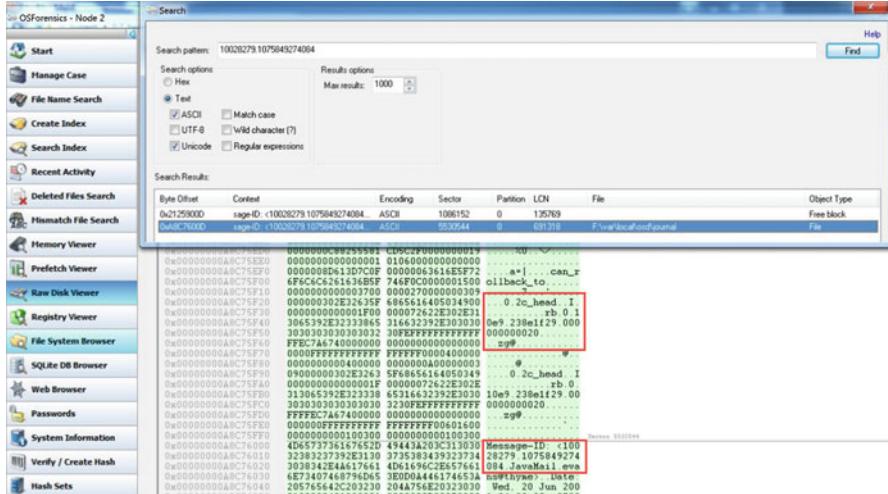


Fig. 16 The message and RADOS header is next to each other

a directory for Monitor map. It was previously discussed that Ceph Storage Cluster consists of two main components, namely, the OSD and the Ceph Monitor, located in the same `/var` directory. Having investigated them it seems clear that this is where OSD and Monitor files are stored (Fig. 17). A file from within MON folder also shows the monitor requests in action (Fig. 18).

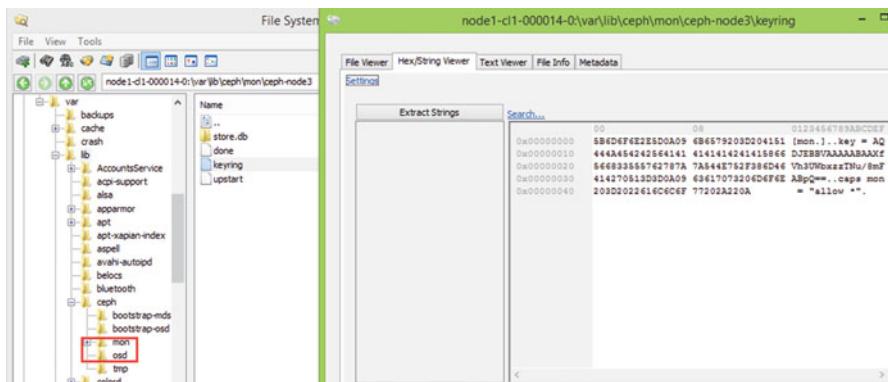


Fig. 17 OSD and Monitor directories with keyring file open

The free block entries contain reference to what seems to be the user ID and Epoch (0.2c_epoch). The Epoch string (in image 34 “0.2c_epoch”) refers to an OSD

The screenshot shows a window titled "node1-cl1-000014-0:\var\lib\ceph\mon\ceph-node3\store.db\000043.idb". The interface includes tabs for "File Viewer", "Hex/String Viewer", "Text Viewer", "File Info", and "Metadata". Below these tabs is a "Settings" link. The main area is divided into two columns: "Extract Strings" and "Search...". The "Search..." column contains a search bar and a dropdown menu. The "Extract Strings" column displays a large list of memory dump entries. Each entry consists of a hex address (e.g., 0x000000D0, 0x000000E0, etc.) followed by a string of characters and numbers. The strings include various Ceph metadata such as monmap information, object IDs, and file system details. A vertical scrollbar is visible on the right side of the list.

Fig. 18 Monitor data

map. The first part of the RADOS object ID refers to the RBD (the Rados Block Device) and rollback references.

6 Conclusion and Recommendations for Further Research

The extensive labour in trying to collect large amount of data produced by ten simple text files is not only unfriendly for a forensic investigator, but it also leaves no doubt about the need of a Could forensics framework as well as an automated tool that would help to process all this data without the need for a lot of manual process. However, at the same time the problem remains regarding the behaviour of different file systems. Just like Ceph and XtremeFSone, file system may be different from another in that they try to offer better product and better service using innovative and improved solutions. This way one particular Cloud forensics solution can be applicable to a particular system, but fail while using another.

The tools used for this investigation were helpful in identifying relevant information, however, there should be more automation in place to avoid manual search for

documents. They may be helpful in identifying any further gaps in the framework as the automated tool would discover data no matter where it is recommended to look for it. This could be one of the further research areas.

Also, there is still much more to discover about Ceph, because the way it operates generally is very effective and convenient for any business that needs high availability. However, the data that it leaves behind could potentially be an information security concern if the information on Ceph Cluster is sensitive. It would be good to find out how could this be improved.

There could be more research done on forensic tools and how useful they are for investigating a Cloud platform. How much of storage and processing power does a forensic investigator need in order to be able to use them, and most importantly how long would all this take. Obviously, a bigger amount of data would produce far greater amount of data for forensic analysis and it could take a very long time to carefully investigate it or for a forensic investigator to build their own tools.

It would be interesting to see if there are any similar file systems like Ceph or is it one of a kind and what way is the market going towards. Would it be useful to build tools and create frameworks for such file systems or are they quickly passing and evolving solutions that we should not dwell on? Such and similar research could no doubt be useful for forensic investigators working on such projects as well as for anyone considering or choosing their Cloud solutions. Ceph seems to be providing an out of the box solution that could find its place where a very high availability and integrity of data is a key. However, just like any other Cloud service, it would serve very poorly in ensuring confidentiality and security of sensitive data. Moreover, it is interesting to find out how developed guidelines of this research can be applied when investigating real-world attacks in cloud computing environments such as Malware [41] and Botnet [42] attacks, exploit-kits [43], crypto-ransomware [44], or even exploiting network infrastructure devices such as SDN routers [45].

References

1. J. Baldwin, O. M. K. Alhawi, S. Shaughnessy, A. Akinbi, and A. Dehghantanha, *Emerging from the cloud: A bibliometric analysis of cloud forensics studies*, vol. 70. 2018.
2. B. Martini and K.-K. R. Choo, “Distributed filesystem forensics: XtreemFS as a case study,” *Digit. Investig.*, vol. 11, no. 4, pp. 295–313, Dec. 2014.
3. K. Ruan, J. Carthy, T. Kechadi, and I. Baggili, “Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results,” *Digit. Investig.*, vol. 10, no. 1, pp. 34–43, Jun. 2013.
4. E. Casey, “Cloud computing and digital forensics,” *Digit. Investig.*, vol. 9, no. 2, pp. 69–70, 2012.
5. F. Daryabar, A. Dehghantanha, N. I. Udzir, N. Fazlida, S. Shamsuddin, and F. Norouzizadeh, “A Survey on Cloud Computing and Digital Forensics,” *J. Next Gener. Inf. Technol.*, vol. 4, no. 6, pp. 62–74, 2013.
6. J. Dykstra and A. T. Sherman, “Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques,” *Digit. Investig.*, vol. 9, pp. S90–S98, Aug. 2012.

7. A. Aminnezhad, A. Dehghantanha, M. T. Abdullah, and M. Damshenas, "Cloud Forensics Issues and Opportunities," *Int. J. Inf. Process. Manag.*, vol. 4, no. 4, 2013.
8. Y.-Y. Teing, A. Dehghantanha, and K.-K. R. Choo, "CloudMe forensics: A case of big data forensic investigation," *Concurr. Comput.*, 2017.
9. Y.-Y. Teing, D. Ali, K. Choo, M. T. Abdullah, and Z. Muda, "Greening Cloud-Enabled Big Data Storage Forensics: Syncany as a Case Study," *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2017.
10. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing," *EURASIP J. Wirel. Commun. Netw.*, vol. 2016, no. 1, p. 130, May 2016.
11. Y.-Y. Teing, D. Ali, K.-K. R. Choo, M. Conti, and T. Dargahi, "Forensic Investigation of Cooperative Storage Cloud Service: Symform as a Case Study," *J. Forensics Sci.*, vol. [In Press], 2016.
12. "Ceph Homepage - Ceph." [Online]. Available: <https://ceph.com/>. [Accessed: 14-Feb-2018].
13. F. Daryabar, A. Dehghantanha, and K.-K. R. Choo, "Cloud storage forensics: MEGA as a case study," *Aust. J. Forensic Sci.*, pp. 1–14, Apr. 2016.
14. F. Daryabar, A. Dehghantanha, B. Eterovic-Soric, and K.-K. R. Choo, "Forensic investigation of OneDrive, Box, GoogleDrive and Dropbox applications on Android and iOS devices," *Aust. J. Forensic Sci.*, pp. 1–28, Mar. 2016.
15. H. Haddadpajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A Deep Recurrent Neural Network Based Approach for Internet of Things Malware Threat Hunting," *Futur. Gener. Comput. Syst.*, 2018.
16. E. Oriwoh, D. Jazani, G. Epiphaniou, and P. Sant, "Internet of Things Forensics: Challenges and Approaches," in *Proceedings of the 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2013, pp. 608–615.
17. S. Watson and A. Dehghantanha, "Digital forensics: the missing piece of the Internet of Things promise," *Comput. Fraud Secur.*, vol. 2016, no. 6, pp. 5–8, Jun. 2016.
18. M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things Security and Forensics: Challenges and Opportunities," *Futur. Gener. Comput. Syst.*, Jul. 2017.
19. D. Quick and K.-K. R. Choo, "Impacts of increasing volume of digital forensic data: {A} survey and future research challenges," *Digit. Investig.*, vol. 11, no. 4, pp. 273–294, Dec. 2014.
20. S. H. Mohtasebi, A. Dehghantanha, and K.-K. R. Choo, *Cloud Storage Forensics: Analysis of Data Remnants on SpiderOak, JustCloud, and pCloud*. 2016.
21. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence," *IEEE Trans. Emerg. Top. Comput.*, 2017.
22. A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning," *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2018.
23. H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "Intelligent OS X malware threat detection with code inspection," *J. Comput. Virol. Hacking Tech.*, 2017.
24. D. Kiwia, A. Dehghantanha, K.-K. R. Choo, and J. Slaughter, "A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence," *J. Comput. Sci.*, Nov. 2017.
25. D. Birk and C. Wegener, "Technical Issues of Forensic Investigations in Cloud Computing Environments," in *2011 IEEE Sixth International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, 2011, pp. 1–10.
26. Y.-Y. Teing, A. Dehghantanha, K.-K. R. Choo, T. Dargahi, and M. Conti, "Forensic Investigation of Cooperative Storage Cloud Service: Symform as a Case Study," *J. Forensics Sci.*, vol. 62, no. 3, 2017.
27. Y.-Y. Teing, A. Dehghantanha, K.-K. R. Choo, and L. T. Yang, "Forensic investigation of P2P cloud storage services and backbone for IoT networks: BitTorrent Sync as a case study," *Comput. Electr. Eng.*, vol. 22, no. 6, pp. 1–14, 2016.

28. A. Dehghanianha and T. Dargahi, *Residual Cloud Forensics: CloudMe and 360Yunpan as Case Studies*. 2016.
29. M. Shariati, A. Dehghanianha, B. Martini, and K.-K. R. Choo, “Chapter 19 - Ubuntu One investigation: Detecting evidences on client machines,” in *The Cloud Security Ecosystem*, R. K.-K. R. Choo, Ed. Boston: Syngress, 2015, pp. 429–446.
30. Y.-Y. Teing, D. Ali, K.-K. R. Choo, M. Zaiton, M. T. Abdullah, and W.-C. Chai, “A Closer Look at Syncany Windows and Ubuntu Clients’ Residual Artefacts,” in *Proceedings of 9th International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage (SpacCS 2016)*.
31. M. Shariati, A. Dehghanianha, and K.-K. R. Choo, “SugarSync forensic analysis,” *Aust. J. Forensic Sci.*, vol. 48, no. 1, pp. 95–117, Apr. 2015.
32. B. Blakeley, C. Cooney, A. Dehghanianha, and R. Aspin, “Cloud Storage Forensic: hubiC as a Case-Study,” in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2015, pp. 536–541.
33. R. B. van Baar, H. M. a van Beek, and E. J. van Eijk, “Digital Forensics as a Service: A game changer,” *Digit. Investig.*, vol. 11, pp. S54–S62, 2014.
34. T. Dargahi, A. Dehghanianha, and M. Conti, *Investigating Storage as a Service Cloud Platform: PCloud as a Case Study*. 2016.
35. M. Petraityte, A. Dehghanianha, and G. Epiphanou, “A Model for Android and iOS Applications Risk Calculation: CVSS Analysis and Enhancement Using Case-Control Studies,” Springer, Cham, 2018, pp. 219–237.
36. H. Haughey, G. Epiphanou, H. Al-Khateeb, and A. Dehghanianha, *Adaptive traffic fingerprinting for darknet threat intelligence*, vol. 70. 2018.
37. “NIST SP 800-86, Guide to Integrating Forensic Techniques into Incident Response.”
38. B. Martini and K.-K. R. Choo, “An integrated conceptual digital forensic framework for cloud computing,” *Digit. Investig.*, vol. 9, no. 2, pp. 71–80, Nov. 2012.
39. R. McKemmish, *What is forensic computing?* Canberra: Australian Institute of Criminology, 1999.
40. “Intro to Ceph — Ceph Documentation.” [Online]. Available: <http://docs.ceph.com/docs/master/start/intro/>. [Accessed: 14-Feb-2018].
41. N. Milosevic, A. Dehghanianha, and K.-K. R. Choo, “Machine learning aided Android malware classification,” *Comput. Electr. Eng.*, vol. 61, 2017.
42. S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghanianha, and R. Khayami, “BoTShark: A Deep Learning Approach for Botnet Traffic Detection,” Springer, Cham, 2018, pp. 137–153.
43. M. Hopkins and A. Dehghanianha, “Exploit Kits: The production line of the Cybercrime economy?,” in *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, 2015, pp. 23–27.
44. J. Baldwin and A. Dehghanianha, *Leveraging support vector machine for opcode density based detection of crypto-ransomware*, vol. 70. 2018.
45. M. K. Pandya, S. Homayoun, and A. Dehghanianha, *Forensics investigation of openflow-based SDN platforms*, vol. 70. 2018.

Forensic Investigation of Cross Platform Massively Multiplayer Online Games: Minecraft as a Case Study



**Paul J. Taylor, Henry Mwiki, Ali Dehghantanha, Alex Akinbi,
Kim-Kwang Raymond Choo, Mohammad Hammoudeh, and Reza M. Parizi**

Abstract Minecraft, a Massively Multiplayer Online Game (MMOG), has reportedly millions of players from different age groups worldwide. With Minecraft being so popular, particularly with younger audiences, it is no surprise that the interactive nature of Minecraft has facilitated the commission of criminal activities such as denial of service attacks against gamers, cyberbullying, swatting, sexual communication, and online child grooming. In this research, we simulate the scenario of a typical Minecraft setting, using a Linux Ubuntu 16.04.3 machine, acting as the MMOG server, and client devices running Minecraft. Then, we forensically examine both server and client devices to reveal the type and extent of evidential artefacts that can be extracted.

P. J. Taylor · H. Mwiki

School of Computing, Science and Engineering, University of Salford, Manchester, UK
e-mail: Paul.Taylor_Titan@titan.police.uk

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada
e-mail: ali@cybersciencelab.org

A. Akinbi

School of Computer Science, Liverpool John Moores University, Liverpool, UK
e-mail: o.a.akinbi@ljmu.ac.uk

K.-K. R. Choo

Department of Information Systems and Cyber Security, The University of Texas at San Antonio,
San Antonio, TX, USA
e-mail: raymond.choo@fulbrightmail.org

M. Hammoudeh

School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University,
Manchester, UK
e-mail: M.Hammoudeh@mmu.ac.uk

R. M. Parizi

Department of Software Engineering and Game Development, Kennesaw State University,
Marietta, GA, USA
e-mail: rparizi1@kennesaw.edu

Keywords Forensic science · Massively multiplayer online games (MMOG) · Minecraft forensics · MMOG forensics · Game forensics · Online games forensics

1 Introduction

These days, cyber threats are affecting all aspects of our digital life [1]. Ransomware [2] that are targeting our health infrastructure [3], malware that are infecting our cloud computing [4] and big data analytic platforms [5], Trojans that are disrupting our financial services [6], exploit-kits which are targeting users [7], and malicious programs that are infecting our mobile phones [8] and laptops [9] are just a few examples of attack tools targeting our digital infrastructure and services every day. Cyber criminals are leveraging software and hardware vulnerabilities [10] in any new technologies from Peer2Peer (P2P) cloud services [11] to the Internet of Things (IoT) [12, 13] and autonomous vehicular networks [14] to launch Denial of Services (DoS) attacks [15] and Botnets [16] and drop Ransomware [17] and other malicious programs [18]; and Massively Multiplayer Online Games (MMOG) are not an exception. MMGs allow millions of individual users to interact together in one gaming environment. Their popularity with the gaming community lies with the ability to collaborate, challenge and communicate with other like-minded individuals from all over the world. However, there is also the potential that the gaming platform can be abused by criminals to commit a variety of computer misuse such as cyberbullying, swatting and sexual offences, including against children and young people [19, 20]. For instance, similar to other online venues, gaming platforms can be used as recruiting grounds for child sex tourism [21].

Minecraft is one such game, which is the focus of our research due to its popularity and cross platform nature. Specifically, Minecraft is a sandbox game that allows users to construct buildings and play in a multiplayer mode to compete for resources. Minecraft was originally developed by Stockholm-based company Mojang, which was acquired by Microsoft in 2014. Investment from Microsoft allowed Minecraft to proliferate across multiple platforms including Windows, Mac OS X, iOS, Android, Xbox and Playstation, and has reportedly 55 million monthly players. The cross platform nature of Minecraft was further bolstered in September 2017 with the release of the “Better Together Update”, which allowed players on Xbox, Windows 10, Virtual Reality and mobile devices to play together either in small groups or in massive online games that allow up to millions of players.

MMOG’s like Minecraft attracted the attention of online offenders, who may seek to exploit the interactive nature of the game. For example, they are able to communicate with users from different jurisdictional areas, e.g., different countries or continents, and have the ability to maintain a level of anonymity that may hamper the efforts of cross-border law enforcement investigations [22].

It is not uncommon for offenders to target other users for purely malicious reasons, an act referred to in the community as ‘griefing’ [23], e.g., steal or destroy

objects built by other players, or ‘swatting’ [20, 24] when the swatter calls an emergency service (e.g. 911 or 999) to report a fictitious serious crime involving another user (victim) that results in law enforcement actions be undertaken against the user. Distributed Denial of Service (DDoS) attacks can also be launched against the server hosting the game as well as against other user(s). In the latter, perpetrators can obtain the IP address of a user/victim through other online conversations [25]. It has even been suggested that the use of ‘booters’ to facilitate DDoS attacks against Minecraft users is part of the very culture of the game [26]. Of most concern is the abuse of the chat facility by adult offenders in order to communicate with children and young people in a sexually explicit manner, inciting them to commit sexual acts or even grooming them with the aim of arranging a meeting in real life [27].

There are mechanisms in place to assist in the prevention of online offending and protection of young gamers. One such development is the introduction of ‘Realms’ by Microsoft, which restricts multiplayer games to a maximum of 10 invite-only players. There is also Minecraft specific advice focussed at parents to assist in education and crime prevention, such as that provided by the National Society for the Prevention of Cruelty to Children (NSPCC). When offences are committed, it is important to be cognisant of the evidential opportunities available on victim devices and the servers they use for online multiplayer games. Digital forensic investigation of victim and suspect devices is often critical in such cases and there is a need to know what data is available on the various Minecraft platforms, and where to find such data.

1.1 ***Problem Statement***

This research focuses on identifying, collecting and analysing digital artefacts in a forensically sound manner on Windows and Linux operating systems, two widely used platforms in gaming. Our research aims to identify evidential artefacts on client devices that may be in the hands of victims, witnesses or offenders. We also aim to determine what evidential products are held on a Linux based Minecraft server should law enforcement identify such a server being either the target of computer misuse offences (e.g. under the Computer Misuse Act 1990 in UK; 18 U.S. Code § 1030, 18 U.S. Code § 2701, 18 U.S. Code § 2251, etc. in the U.S.) or a repository for evidence of offending (e.g. under 18 U.S.C. § 2252A- certain activities relating to material constituting or containing child pornography in the U.S.).

The initial scoping exercise led the researchers to believe that the official Minecraft server, as provided by Microsoft and used in this research, would incorporate the ‘Better Together’ update advertised by Microsoft. This would have allowed Minecraft users on all device platforms to connect to the same server; however, we discovered that cross-platform access is dependent on a number of factors. The Minecraft server provided by Microsoft allows for MMOGs to be conducted over the Internet. However, this is restricted to desktop environments including Windows, Mac and Linux. The reason for this is that the Better Together

update released in September 2017 became the primary version on iOS, Android, Console and Windows 10. The new version allows for massively multiplayer gaming, but through a small selection of Microsoft vetted partner server providers. Becoming a Microsoft partner for the purposes of hosting such a server is beyond the scope of this research and for that reason we will not explore evidential opportunities on iOS and Android devices. Thus, the server software that is made available by Microsoft and used in this research is actually referred to as *Minecraft: Java Edition* and only compatible client software available on Desktop will be able to connect to this server.

The Android Google PlayStore does have an application that allows connection to a Java Edition server. This application is called *Boardwalk* that “[a]llows you to run the PC version of Minecraft on your device”, but it also warns of unreliability. Although we could get Boardwalk to open, it did not progress past the initial splash screen and there was no option to connect to custom servers. There is also no comparative application on the Apple App Store, at the time of this research.

Research has been conducted by other scholars with Minecraft installations on Xbox, PlayStation 4 gaming consoles [28] and on a Windows Minecraft server [29]. However, there is no study focusing on the forensic analysis of Minecraft installed on a Linux Ubuntu server, which is the recommended Operating System for deployments on Microsoft Azure cloud platforms and a popular choice in user forums [30]. The outcome of this research will be to expand upon previous findings, with a particular focus on live memory and traffic data artefacts.

Similar to the approach undertaken by Quick and Choo [31, 32] in this research we attempt to answer the following questions:

1. What artefacts of evidential interest are retained on the Linux Minecraft server after installation of the Minecraft server software and the running of the game with multiple cross platform users?
2. What evidence is there on the server and client devices of communication between users running Minecraft?
3. What information can be seen from server and client logs that can assist in the identification and profiling of a user?
4. What can be seen from network traffic between clients and the server?

The rest of the paper is structured as follows. Section 2 contains a review of the current literature on the topic of digital forensics and MMOG’s. Section 3 presents the framework used to guide the forensic investigation, followed by the experimental setup. Section 4 covers the evidential collection phase and seeks to answer the first research question. Section 5 continues with analysing the data found on the hard disk of the client. Section 6 looks at the memory, which will address the second and third questions. Section 7 looks into what can be seen the network traffic with the aim of addressing the fourth research questions. Finally, Sect. 8 provides a conclusion to the paper and outlines the potential for future studies.

2 Literature Review

Since their introduction to the gaming community, MMOGs gained significant fanfare [33]. Specifically, Minecraft has increased in popularity, including among autistic children [34] who seek to meet like-minded friends in their own age group. However, it is known that this gamming platform can be abused by child sexual and other offenders. The Leahy Center for Digital Investigation [33], for example, recognized such a risk and examined chat function logs from several MMOGs. They discovered that chat logs were stored in plain text and different vendors took entirely different approaches to the storage of conventions, which highlights the need to conduct further research with other MMOG's, such as Minecraft.

Minecraft began its existence as a Java based application and then evolved to incorporate a Windows 10 edition due to Microsoft's compatibility issues relating to Java applications [35]. Players of Minecraft explore a virtual world and interact with other characters that spawn in and out of existence as gameplay progresses, and users across the Internet connect to multiplayer game hosting servers. There is no one clear objective for all users. Players can enter a game and choose to explore the virtual world around them, create buildings and structures, or play a survival game whereby bots and other characters have the aim of destroying one another.

Due to the fact that MMOG's can be a platform for online offending, research continues into the real-time identification of offenders through behavioural analysis [36]. In our research, we look at the evidence available in the aftermath of a crime being committed. In their study, Ki, Woo and Kim [37] suggested that individual actions, such as malicious social interactions and behaviors, as well as automated bots are used in MMOGs for such offending.

The personal information and virtual properties of the people who subscribe to and participate in such games draws a lot of attention from attackers [38]. Other criminal activities include compromising of user accounts. A popular reason for account compromise is to achieve easy profits and in extreme scenarios this can even lead to gold farming, which is the stock piling of virtual game assets for eventual profitable sale in the real world [38].

With the rise of MMOGs exploitation in cybercrime, forensic investigators are facing several challenges during the collection of evidence that is essential to prove that the crime did really happen [39]. In MMOG forensics, there are situations with which forensic tools can be fooled by an attacker, whereby they remove data traces so as to make sure that they cannot be easily be tracked [40]. It has been suggested that if users are engaging in criminal activity and there is no in-depth knowledge of where or how to find evidence, only the most apparent incidents of misconduct can be discerned, prevented or eradicated [29].

Existing research conducted on Xbox and PlayStation 4 platforms running Minecraft, showed that established tools alone cannot be relied upon for the retrieval of all relevant artefacts [28]. Similarly, existing research on a Windows Minecraft server and client, highlighted the availability of chat logs but identified that further

work was required in relation to the analysis of network traffic [29]. Our research aims to fill this gap by exploring evidential opportunities on a Linux server and in the cross-platform network traffic.

Forensic investigation of a cross-platform MMOG like Minecraft is challenging as players use different platforms. The footprints detected by forensic analysis from these different platforms can be used in criminal proceedings, in particular where the investigation is needed to fulfill further rigid restrictions dictated by the strict procedures agreed in court [40].

Aside from a Linux server and a Windows client, our work will also look at two mobile device clients. Rajendran and Gopalan [41] argued the very important challenges in mobile device forensics are variations from design to design, model to model, time to time, manufacturer to manufacturer, and the adaptation of the technology [41].

It is well-known that users of MMOGs utilize third-party applications to communicate with other gamers whilst in play. Sound research has been conducted into the evidential retrieval of data from popular VoIP applications on mobile platforms [42] so it is now necessary to also explore the in-built chat facilities between various operating system platforms.

Open source intelligence following a report of crime initiated or facilitated on an MMOG may lead investigators to a repository of information, suspect machine or even assist in the identification of victims through attribution of their devices. The use of this intelligence could be combined with application specific forensic methods, as outlined in this paper, to counter the challenge of the growing volume of disparate data with consideration for the future use of a Digital Forensic Data Reduction Framework [43].

3 Research Methodology

3.1 *Forensic Framework*

In this section, we will outline the digital forensic framework and the experimental setup relied upon to conduct our research. It is necessary for the digital forensic method to adhere to the well-established guidelines and policies published internationally.

The UK's Association of Chief Police Officers (ACPO) Good Practice Guide for Digital Evidence [44] contains four overarching principles; prevent the original data from being changed, where data has to be accessed this should be done by competent examiners, they must maintain accurate, reproducible records and the decisions being made must be the decision of the person in charge of the investigation. These principles are complemented with the practical advice available from the US National Institute for Justice's Guide for First Responders [45], which outlines guidance for handling a wide variety of digital media at crime scenes. Most recently, the UK's Forensic Science Regulator has mandated that anyone reporting

scientific or technical work to the courts must comply with the Forensic Code of Conduct [46]. This stipulates that public, police and commercial digital forensic providers have to be accredited to International Organisation for Standardisation (ISO) and International Electrotechnical Commission (IEC) standard 17025. The National Institute of Science and Technology (NIST) suggested an outline for a digital forensic process of data collection, examination, analysis and reporting [47]. Efforts continue to be made internationally to find consensus in the approach taken by digital forensic examiners [48] and consideration was given to the above policies and guidance when conducting our research.

We chose to utilise the framework set out by Martini and Choo [49] as shown in Fig. 1. This shares similarities with other established frameworks, but allows for iterations between the examination and analysis step through to the evidence identification and preservation step, which assists in the preservation of evidence found in cloud platforms.

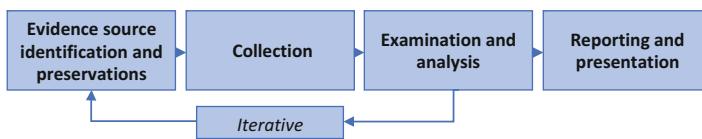


Fig. 1 Digital forensics framework of Martini and Choo

3.1.1 Phase 1: Evidence Source Identification and Preservation

Initially, we identified the platform-dependent hardware to be subjected to analysis. The Minecraft server and Windows client were contained within a Virtual Machine (VM) and the files of interest were VMware Virtual Disk files (VMDK) and Virtual Memory files (VMEM). These were extracted and a bit-for-bit working copy was produced of each.

3.1.2 Phase 2: Collection

In this phase, we collected the files that contains items of interest for forensic analysis, keyword searching and network traffic data. MD5 and SHA1 hash values corresponding to the files of interest were calculated and retained for subsequent comparison to ensure that working file copies were true to the original.

3.1.3 Phase 3: Analysis

In this phase, we examined the data contained within the forensic captures of the hard disks and memory from the Linux server and Windows, iOS and Android

clients. Then, we examined the traffic data that was captured during live gameplay. These sets of data were subjected to keyword searching for known usernames, passwords and extracts of conversation exchanged during gameplay. Log files stored on the machines were analysed and carved to establish the existence of evidential artefacts.

3.1.4 Phase 4: Reporting and Presentation

For this final phase, we recommended a short-form template report that could be used as an evidential exhibit that could be produced with accompanying witness testimony in court.

3.2 *Experimental Setup*

For our research, we simulated the scenario of an individual running a Minecraft server on their Linux Ubuntu 16.04.3 Server machine and hosting a multiplayer game. The other gamers were connected to the server from a Windows 7 Enterprise Service Pack 1 PC. We hosted the Linux server and Windows client in their own VMs on the same network and used the snapshot tool of the VMware software to create 8 snapshots representing different environments based on the real-life use of the server and client. Base VM snapshots were used as a control to compare against snapshots taken after some activity. Table 1 provides a list of the different VM snapshots captured and the description of the activity taken prior to the snapshot.

The Windows version used was a free developer edition from Microsoft. A Windows 7 Enterprise Service Pack 1 VM file was downloaded from <https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/>.

A network adaptor was added and set to allow a bridged connection from the VM. A snapshot was taken in this state (1.0). VM snapshots were used as opposed to physical hardware for practical time considerations and convenience. This was aligned to the points made by Teing et al. [50] and Quick and Choo [31] in relation to the ease in which a machine state can be reverted to a restore point for repeated experimental activity and to keep the file space to a minimum for collection of evidence and analytical efficiency.

During each snapshot, VMware creates multiple .vmdk files. To combine these into one file to allow for forensic acquisition we used *vmware-diskmanager* with the following command:

```
-r "Virtual Disk-000001.vmdk" -t 0 [new file name].vmdk
```

Live RAM captures were represented by the corresponding VMEM files generated by VMware upon the initialisation of a snapshot. In order to simulate a real-life situation, the snapshots, and hence the RAM captures, were taken following a particular activity and whilst the machine was powered on. Guidance to UK police

Table 1 Configurations of virtual machines on Linux (server) and Windows (client)

VM snapshot	Description
Base VM1.0, 2.0	The Base VM snapshots were prepared for the following operating systems: <ul style="list-style-type: none"> • Windows 7 Enterprise Service Pack 1 (Build 7601) with 1GB RAM (1.0) • Linux Ubuntu 16.04.3 Server with 2GB RAM and 20GB hard disk (2.0)
Install VM1.1, 2.1	Minecraft software was installed on the operating systems and further snapshots were taken <ul style="list-style-type: none"> • On the Windows machine; Minecraft version 1.12.2 (1.1) • On the Linux machine; Minecraft Server version 1.12.2 (2.1)
Activity VM1.2, 2.2	The Minecraft server was initiated on the Linux machine and the Minecraft client on the Windows machine then connected over the network Snapshots were taken on the Windows machine (1.2) and the Linux machine (2.2)
Uninstall VM1.3, 2.3	Further snapshots were created following the standard uninstallation of Minecraft from the operating systems. Standard documented uninstall procedures were conducted on the Windows (1.3) and Linux (2.3) machines

officers is to isolate the power immediately from a machine rather than initiate a shutdown procedure [44] and hence snapshots from live machines were most representative of the scenario (Fig. 2).

**Fig. 2** VM snapshots created on Linux and Windows machine running Minecraft server and client respectively

In order to prepare the installation snapshot on Windows (1.1), Minecraft was downloaded from the official site <https://minecraft.net/en-us/download/> and installed to the default directory. A snapshot was taken at this stage and we attempted to ‘Play’, however it crashed due to a graphics adaptor issue. It was necessary to install VMware tools and to disable Windows Update to prevent changes to the system outside of the Minecraft installation. Following the installation of VMware tools the system had to be restarted. A further snapshot was taken. We then conducted a test to see if Minecraft could be played and it could so the latter snapshot was taken as the install snapshot.

The activity taken to create the ‘Activity VM’ snapshots involved connecting to the server, starting a new game and communicating through the chat facility. It was decided that no particular avatar interaction was required due to previous research by Alstad, Duncan, Detlor et al. [51] finding that network activity levels between active and idling players were similar.

Table 2 Research tools and their usage

Tool	Usage
VMware Fusion version 8.5.8 (Mac)	To create the Linux server and Windows client and allow for the capture of system snapshots
FTK Imager version 3.2.0 (Windows)	To acquire forensic images for .VMDK and .VMEM files from the Linux and Windows machine snapshots
Autopsy version 4.5.0 (Windows)	To view and analyse directories, files, Windows registry data, page and swap files
HxD version 1.7.7.0 (Windows)	To keyword search live memory captures and carve relevant data
Volatility version 2.6 (Linux)	For automated carving of evidential artefacts from captured live memory
Wireshark version 2.4.2 (Mac)	To capture network traffic passing through the virtual adaptor on the host machine
Network Miner version 2.2 (Mac)	To parse out relevant information from the network traffic capture files
Windows Event Viewer (Windows)	To view relevant events recorded in the Windows system

The capturing of network traffic on desktop operating systems is relatively straightforward when compared to the difficulties encountered with capturing traffic from mobile devices [43]. We adapted our approach and chose to capture the network traffic by running Wireshark 2.4.2 in the host environment, with the option set to only capture traffic passing through the virtual network adaptor. Table 2 lists the tools used for the collection and analysis phases of our research.

In order to capture the traffic exclusively between the Windows client and the Linux server we followed the below steps:

1. Open Wireshark on the host machine and capture all traffic on the default network adaptor.
2. Test Wireshark with ping commands between machines.
3. Apply a filter for the two relevant IP addresses; (*ip.addr==192.168.0.29 and ip.addr==192.168.0.82*)
4. Export the displayed data to a new file *capture1.pcap* (to remove background noise from the host machine).

4 Server Analysis

Due to the sheer volume of the involved data, we decided to collect and analyse the data which would only contain potential information of interest for Minecraft forensics. Examples of the data collected are chat log artefacts, artefacts related to users, artefacts related to the game server and client machine and network artefacts [29]. These data can be found in files stored in different location, such as Minecraft log files, Window system files and unallocated partitions. For control purposes, the examination of base snapshots were confirmed to have no Minecraft artefacts and as such were suitable for comparison against subsequent snapshots.

4.1 Analysis

In this section we are presenting the findings from comparing the base level VM snapshots with that of snapshots following specified activity on the server and client devices.

4.2 Linux

The captured image files were examined for changes between the base Linux server and the server following installation of the Minecraft software and game activity. The */etc/hostname* directory contains the *hostname* file, which shows that the server had been titled ‘minecraft-server’. The *passwd* file within */etc/* shows the user account, which for this research was ‘taylor’.

The Date Modified attribute had been updated for 3 sub-directories within */var/lib*:

var/lib/update-notifier: The file named *updates-available* residing within this directory had been changed to reflect an increase of 2 updatable packages and 6 security updates, however, nothing identifying Minecraft was found.

var/lib/lxd: The contents of this directory remained empty and there had been no change other than the Date Modified attribute being updated to reflect the time of installation.

var/lib/snappyd: The file named *state.json* file contained a date that reflected the time of last access, under a heading ‘last-refresh’.

The *var/tmp* directories all displayed an updated Date Modified attribute and a new directory had been created; */hsperfdata_taylor*. Contained within was a file named *1363* that contained file paths for Minecraft’s Java installation. The main path referred to was */usr/lib/jvm/java-8-openjdk-amd64/jre/lib/* which contains the Java installation files and copyright notices.

There were no changes to the */opt* directory, but significant logs available within */var/log*. The *syslog* file recorded the Minecraft server’s IP address of 192.168.0.23 as per the DHCP requests made following initialisation. There were also records following each initialisation indicating that the system time had been synchronised over the internet with the time server at <ntp.ubuntu.com>.

The *auth.log* file detailed the time and dates of successful logins to the server under the root user, taylor.

There were significant items of interest contained with the user directory */home/taylor*. The file *.bash_history* listed all the recent commands input by the server operator, including the download command necessary to obtain the Minecraft software; *wget https://s3.amazonaws.com/Minecraft.Download/versions/1.12.2/minecraft_server.1.12.2.jar*

The downloaded software *minecraft_server.1.12.2.jar* was saved to the same directory with a Date Modified value not representative of any user action and likely the date and time it was uploaded to the Amazon server by developers.

Following installation of the software, a number of files were created in the same directory. Of note is the file *server.properties*, which allows the server operator to configure a number of settings prior to launching a game. The contents of *server.properties* is in plain text and useful information like time stamp, port number and ‘Message of the Day’ can be seen and compared to defaults, which are shown in Fig. 3; the values changed from default are displayed in bold and underlined. Difficulty was set to 0 to prevent spawned characters from attacking the user and preventing the necessary activity required to conduct the research. Online-mode was set to 0 to prevent the server from attempting to verify connecting users with Microsoft, which was necessary due to running the applications on a LAN.

```
#Minecraft server properties
#Sat Dec 02 20:47:15 GMT 2017|
max-tick-time=60000
generator-settings=
force-gamemode=false
allow-nether=true
gamemode=0
enable-query=false
player-idle-timeout=0
difficulty=0
spawn-monsters=true
op-permission-level=4
pvp=true
snooper-enabled=true
level-type=DEFAULT
hardcore=false
enable-command-block=false
max-players=20
network-compression-threshold=256
resource-pack-sha1=
max-world-size=29999984
server-port=25565
server-ip=
spawn-npcs=true
allow-flight=false
level-name=world
view-distance=10
resource-pack=
spawn-animals=true
white-list=false
generate-structures=true
online-mode=false
max-build-height=256
level-seed=
prevent-proxy-connections=false
use-native-transport=true
motd=Hello, this is a test message of the day for all to see.
enable-rcon=false
```

Fig. 3 Contents of *server.properties*

The file *usercache.json* provides a list of previously connected users. The user name is recorded, in our case ‘taylor_salford’. A UUID is assigned and an expiry date is set for 1 month after their initial joining of the server.

Within */home/taylor/logs* there were a number of files of the name format *yyyy-mm-dd-[sequential number from 1].log.gz*. Each of these files contained complete outputs of the server user’s screen, including timestamps for when commands were typed after the initialisation of the Minecraft server software. The most recent file contained within the */home/taylor/logs* directory was *latest.log*, the contents of which are shown in Fig. 4.

```
[20:47:14] [Server thread/INFO]: Loading properties
[20:47:14] [Server thread/INFO]: Default game type: SURVIVAL
[20:47:14] [Server thread/INFO]: Generating keypair
[20:47:15] [Server thread/INFO]: Starting Minecraft server on *:25565
[20:47:15] [Server thread/INFO]: Using epoll channel type
[20:47:15] [Server thread/WARN]: *** SERVER IS RUNNING IN OFFLINE/INSECURE MODE!
[20:47:15] [Server thread/WARN]: The server will make no attempt to authenticate usernames. Beware.
[20:47:15] [Server thread/WARN]: While this makes the game possible to play without internet access,
it also opens up the ability for hackers to connect with any username they choose.
[20:47:15] [Server thread/WARN]: To change this, set "online-mode" to "true" in the
server.properties file.
[20:47:15] [Server thread/INFO]: Preparing level "world"
[20:47:15] [Server thread/INFO]: Loaded 488 advancements
[20:47:15] [Server thread/INFO]: Preparing start region for level 0
[20:47:16] [Server thread/INFO]: Preparing spawn area: 1%
[20:47:17] [Server thread/INFO]: Preparing spawn area: 33%
[20:47:18] [Server thread/INFO]: Preparing spawn area: 87%
[20:47:19] [Server thread/INFO]: Done (3.745s)! For help, type "help" or "?"
[20:48:27] [Server thread/INFO]: taylor_salford[/192.168.0.82:49804] logged in with entity id 413 at
(-134.25112558991043, 59.0, 219.43146141990562)
[20:48:27] [Server thread/INFO]: taylor_salford joined the game
[20:50:12] [Server thread/INFO]: <taylor_salford> Hi, my name is testuser1 and I am playing this
Minecraft game on Windows. The time on my system is 20:50
[20:53:30] [Server thread/INFO]: <taylor_salford> Hi, this is testuser1 again and the time on my
system now is 20:53
[20:53:39] [Server thread/INFO]: taylor_salford lost connection: Disconnected
[20:53:39] [Server thread/INFO]: taylor_salford left the game
```

Fig. 4 Contents of */home/taylor/logs/latest.log*

Figure 4 shows that the contents of *latest.log* replicate the screen output made to the Minecraft server operator. The following information could be useful to investigators:

- “Starting Minecraft server on *:25565” This is the port that the server will be available on.
- “Server is running in offline/insecure mode!” This is a change from the default option and allows users to connect to the server with any username they chose; there will be no verification of the username with Microsoft. The feature can be enabled to assist with blacklisting of offending users.
- The entry at time [20:48:27] shows username taylor_salford connecting from 192.168.0.82, which is in fact the Windows client used in the research.
- Any chat typed out by the connected user taylor_salford is output to the screen on the server. Our test message shows that the time on the Windows client machine was in sync with that of the Linux server.

Following the uninstallation process, evidence was still available to suggest the machine was running the Minecraft server software.

The *.bash_history* that resides in */home/taylor* shows the command used to uninstall Minecraft; rm –vr ./*

Further to this, hex data exists for the directory */home/taylor* and indicates that the software was once present on the machine, as shown in Fig. 5.

000	F2 01 0C 00 0C 00 01 02-2E 00 00 00 01 00 0A 00	ò.....
010	0C 00 02 02 2E 2E 00 00-99 13 0C 00 14 00 0C 01
020	2E 62 61 73 68 5F 6C 6F-67 6F 75 74 9C 13 0C 00	.bash_logout.....
030	10 00 08 01 2E 70 72 6F-66 69 6C 65 F4 16 0C 00profileò..
040	10 00 07 01 2E 62 61 73-68 72 63 2D 3A 17 0C 00bashrc-:..
050	10 00 06 02 2E 63 61 63-68 65 0C 00 43 17 0C 00cache..C..
060	80 00 19 01 2E 73 75 64-6F 5F 61 73 5F 61 64 6Dsudo_as_adm
070	69 6E 5F 73 75 63 63 65-73 73 66 75 6C 16 0C 00	in_successful...
080	CB 18 0C 00 30 00 1B 01-6D 69 6E 65 63 72 61 66	È...0...minecraf
090	74 5F 73 65 72 76 65 72-2E 31 2E 31 32 2E 32 2E	t_server.1.12.2.
0a0	6A 61 72 65 E5 18 0C 00-0C 00 04 02 6C 6F 67 73	jare[.....logs
0b0	E7 18 0C 00 2C 00 11 01-73 65 72 76 65 72 2E 70	ç..., ..server.p
0c0	72 6F 70 65 72 74 69 65-73 16 0C 00 E9 18 0C 00	roperties..é...
0d0	10 00 08 01 65 75 6C 61-2E 74 78 74 E6 18 0C 00eula.txtæ..
0e0	B0 00 05 02 2E 6E 61 6E-6F 6B 50 7A 69 6C 70 73	°....nanokPzilps
0f0	E8 18 0C 00 18 00 0D 01-65 75 6C 61 2E 74 78 74	è.....eula.txt

Fig. 5 Data contained within */home/taylor* showing Minecraft server software

5 Client Analysis

Similar to other MMOGs, e.g., Second Life and World of Warcraft, the main starting point of Minecraft investigation on the Windows client machine is in:

Users/[username]/AppData/Roaming/.minecraft

The files of particular interest within this directory are:

Users/[username]/AppData/Roaming/.minecraft/logs/latest.log

Users/[username]/AppData/Roaming/.minecraft/launcher_profiles.json

Users/[username]/AppData/Roaming/.minecraft/servers.dat

We began by identifying artefacts related to the user. First of all we identified the user setting file, which is:

Users/[username]/AppData/Roaming/.minecraft/logs/2017-11-27-1.log.gz/2017-11-27-1.log

```
[12:47:51] [Client thread/INFO]: Setting user: taylor_salford
```

Fig. 6 Contents of 2017-11-27-1.log

We found out that the name of the file itself reflected the date the user settings were made. The file also provides timestamp of the user setting, as can be seen in Fig. 6.

User details are important to reveal identity of the person who was playing the game. In our experiment we were able to find the information related to user in this file:

Users/[username]/AppData/Roaming/.minecraft/launcher_profiles.json

This file provided details of the user such as Universal Unique Identifier which is used to identify the player account (prefixed with ‘profiles:’), email address of the player (prefixed with ‘username:’), display name of the player in the game (prefixed with ‘displayName:’), user account number (prefixed with ‘selectedUser:’), access token (prefixed with ‘accessToken:’), analytics token (prefixed with ‘analyticsToken:’) and client token (prefixed with ‘clientToken:’). Figure 7 shows the contents of this file.

```
"authenticationDatabase": { "73a2b743f4069a379a79ca3cfafc24ac": {  
"accessToken": "ebca4d0fd8d14868a53b6518b340efe7", "username":  
"p.taylor12@edu.salford.ac.uk", "properties": [ { "name":  
"preferredLanguage", "value": "en-us" } ], "profiles": {  
"dfffc6cf658a40cd858167c969b05390": { "displayName": "taylor_salford" } } }  
, "selectedUser": { "account": "73a2b743f4069a379a79ca3cfafc24ac",  
"profile": "dfffc6cf658a40cd858167c969b05390" }, "analyticsToken":  
"6694a3927fd953990ad66f1dd0589cd1", "analyticsFailcount": 0, "clientToken":  
"423b3863a0f390d91c7aaaf08cc2d79d7" }
```

Fig. 7 Contents of launcher_profiles.json file

Chat logs were stored in:

Users/[username]/AppData/Roaming/.minecraft/logs/latest.log

We were able to obtain the full chat logs that were not encrypted and were saved as plain text so they were easily accessible. Apart from chat logs, *latest.log* file also provided details such as an IP address to which the Windows client connected, port number and the display name of the player. This information is fixed and will only change when user opens another account according to Minecraft [29]. Timestamps of activity were present, but dependent on the local machine time zone settings, when the player started and stopped the game. Figure 8 shows the discrepancy in time between the player’s local and machine time.

```
[12:48:22] [Client thread/INFO]: Connecting to 192.168.0.29, 25565
[12:48:28] [Client thread/INFO]: Loaded 16 advancements
[12:50:12] [Client thread/INFO]: [CHAT] <taylor_salford> Hi, my name is testuser1 and I am
playing this Minecraft game on Windows. The time on my system is 20:50
[12:53:30] [Client thread/INFO]: [CHAT] <taylor_salford> Hi, this is testuser1 again and t
he time on my system now is 20:53
[12:59:37] [Client thread/INFO]: Stopping!
```

Fig. 8 Contents of latest.log file

Logs were also recorded in *latest.log* when attempts were made to login to the game using the same accounts details. An error would be output as there is a duplicate login and the player had already logged in from another location. Duplicate login error can help the investigator know if the account was compromised and someone else used the account to log into the game. Figure 9 shows an example one the message output to the log file.

```
multiplayer.disconnect.duplicate_login=You logged in from another location
```

Fig. 9 Duplicate username log

Artefacts related to server were identified on the Windows client. The file *Users/[username]/AppData/Roaming/.minecraft/servers.dat* contained information about the server, the information is stored in plain text and easily readable. This file provides information such as server name and IP address of the Minecraft server.

Like any other application, Minecraft can crash and we discovered that the file *Users/[username]/AppData/Roaming/.minecraft/crash-reports* provided details of the crash, which includes the source of the crash and associated timestamp.

After uninstalling Minecraft software on the Windows machine we could see that the folder *Users/[username]/AppData/Roaming/.minecraft/* remained on the hard drive, however it was empty. We determined that if the Recycle Bin was not empty then the artefacts could still be recovered from the Recycle.Bin.

Log files are important artefacts to extract as they allow for operating system events to be determined. In our experiment, we did log analysis by searching for the term Minecraft and this involved going through the entries identifying events that are relevant to Minecraft.

We were able to locate event entries referencing the installation and un-installation of the Minecraft application. This provided timestamps in Windows event files, e.g., Application.evtx, as shown in Fig. 10.

Application Number of events: 1,900					
Level	Date and Time	Source	Event ID	Task Categ...	
Information	11/27/2017 1:04:25 P...	LoadPerf	1001	None	
Information	11/27/2017 1:03:45 P...	MsiInstaller	1033	None	
Information	11/27/2017 1:03:45 P...	MsiInstaller	11707	None	

Event 1033, MsiInstaller

General	Details
	Windows Installer installed the product. Product Name: Minecraft. Product Version: 1.0.3.0. Product Language: 1033. Manufacturer: Mojang. Installation success or error status: 0.
Log Name:	Application
Source:	MsiInstaller
Event	1033
Level:	Information
User:	IE11WIN7\IEUser
OpCode:	
More Information:	Event Log Online

Fig. 10 Windows event log entry for Minecraft installation

6 Memory Analysis

The Volatility tool was utilised in order to analyse the live memory captures from the Windows machine following Minecraft activity. There were difficulties with adding a custom profile for the Linux machine in Volatility; Linux Ubuntu 16.04.3 Server does not have a pre-configured profile available for download and so the memory carving was completed manually.

On the Windows client the *netscan* Volatility plugin failed to identify an IP address for the Linux server.

The *pscan* Volatility plugin was used to compare processes in memory on the Windows machine before and after the game is started and ended. It could be seen that the process named *MinecraftLaunc* was present in memory from the moment of boot up. Following the software being launched and a game being played, it could be seen that a process called *javaw.exe* had been spawned from *MinecraftLaunc* and both had exited upon the software being closed.

The *memdump* plugin was run against the *MinecraftLaunc* process causing a 254.7 MB .dmp file to be isolated, which contained the contents of all the memory concerned with the running process. Figure 11 shows the data available in the memory of the dumped process. The IP address and name of the Linux server was recorded, as was the server owner's message of the day and the last communication sent from the Windows client over the Minecraft chat facility.

230312208	000001800	000000100	40000000	180000000	0A0000000	00077365	72766572	server ip 192.168.0.29 name Minecraft Server CyG v e r s . d a t servers ip 192.168.0.29 name Minecraft Server CyG
230312236	730A00000	000108000	026970000	0C313932	2E302E32	39080004		
230312264	6E616D65	00104D69	6E656372	61667420	53657276	65720000	FFFFFFFFFF	
230312292	82794711	FFFFFFFFFF	82794711	76006500	72007300	2E006400	61007400	
230312320	5F007400	6D0070000	800000000	580000000	00001800	00000100	400000000	
230312348	180000000	0A0000009	00077365	72766572	730A00000	000108000	026970000	
230312376	0C313932	2E302E32	39080004	6E616D65	00104D69	6E656372		
230312404	61667420	53657276	65720000	FFFFFFFFFF	82794711	000000000	000000000	
230312432	000000000	000000000	000000000	000000000	000000000	000000000	000000000	
35057708	48BB5018	00FB8220	00004500	02424869	2C207468	69732069	73207465	H P C E B H i , t h i s i s t e s t u s e r 1 a g a i n a n d t h e t i m e o n m y s y s t e m n o w i s 2 0 : 5 3
35057736	73747573	65723120	61676169	6E206166	64207468	65207469	6D65206F	
35057764	6E206D79	20737973	74656020	6E6F7720	69732032	303A3533	000000000	
35057792	000000000	000000000	000000000	000000000	000000000	000000000	000000000	
37175376	6F6E223A	7B227465	7874223A	22486565C	6C6F2C20	74686973	20697320	on": {"text": "Hello, this is a test message of the day for all to see."}, "players": {"max": 20, "online": 0}, "version": {"name": "1.12.2"}, "protocol": 340]} , „ÙÉ ØU-cth“{ ØCÜ
37175404	61207465	73742060	65737361	6765206E	66207468	65206461	7920666F	
37175432	7220616C	6C20746F	20736565	2E227D2C	22706C61	79657273	223A7B22	
37175460	60617822	3A32302C	226F6E6C	696E6522	3A307D2C	22766572	73696F6E	
37175488	223A7B22	6E616D65	223A2231	2E31322E	32222C22	70726F74	6F636F6F	
37175516	223A3334	307D7D00	000000C0	FA9CE600	AFF40063	A068FD7B	18CD28F4	
37175544	BD171A43	5C0B4D05	75EF8521	6E0575D0	D650715F	C62D2BB2	FFB2DB4A	Ω C \ M u ð Õ n u - + P q _ Δ - + S ^ e j

Fig. 11 Data stored in memory of MinecraftLaun process

The live memory capture from the Windows machine was manually searched for known information, to establish what was retained in memory during active game play. The IP address and name of the server was recorded in memory in several locations. Additionally, the following data provides time stamps for the connection to the server, the IP address and port number of the server and all chat sent from the client machine.

```
[12:48:22] [Client thread/INFO]: Connecting to 192.168.0.29, 25565
[12:48:28] [Client thread/INFO]: Loaded 16 advancements
[12:50:12] [Client thread/INFO]: [CHAT] <taylor_salford> Hi, my name is testuser1 and I am playing this Minecraft game on Windows. The time on my system is 20:50
[12:53:30] [Client thread/INFO]: [CHAT] <taylor_salford> Hi, this is testuser1 again and the time on my system now is 20:53
```

The same chat information was stored in one other location in memory and could be found with a slightly adapted keyword search of “info [CHAT]”. The Linux live memory was analysed and further useful information was identified. If the local server IP address was not already known, then it could be searched for by performing a keyword search for “DHCPoffer” and reading the timestamp and IP address displayed.

The Windows client IP address was only stored in the Linux memory on one occasion, which was in an area of strings that replicated the contents of the log file shown in Fig. 4. Similarly, this area of memory provided the clearest and most chronologically sound evidence of the chat sent from the Windows client. Artefacts of chat were found elsewhere in memory, however they were often cut short or mixed together in incoherent ways, as is shown in Fig. 12.

The full contents of the *server.properties* file was stored in memory, which included the message of the day. Figure 12 shows that the UUID assigned by the

1209126912	00000000 00000000 00000000 00000000 0FEB027B 22747261 6E736C61 7465223A	i {"translate": "chat.type.text", "with": [{"insertion": "taylor_salford", "clickEvent": {"action": "suggest_command", "value": "/msg taylor_salford"}, "hoverEvent": {"action": "show_entity", "value": {"text": {"name": "\taylor_salford", "id": "\d5f50175-aeb3-34dc-aaef-678a184dd7fc"}, "text": "taylor_salford"}}, "text": "Hi, thi s is testuser1 again and the tim e on my system now is 20:53"]}] w s. The time on my system is 20:5 0"]}_boat_minecraft:recipes/ro ot_entered_water_has_the_recipe entered_water_has_the_recipe
1209126944	22636861 74E27479 70652E74 65787422 2C227769 7468223A 5B782269 6E736572	
1209126976	74696F6E 223A2274 61796C6F 725F7361 6C666F72 64222C22 636C9963 6B457665	
1209127008	6E7423A 7B226163 7496F6F6 223A2275 75676765 73745F63 6F6D0606 6E64222C	
1209127040	22766167 7565223A 222F6073 67207461 796C6F72 5F736167 666F7264 2022702C	
1209127072	22686F76 656724576 656E7422 3A782261 6374696F 6E223A22 73686F77 5F656E74	
1209127104	69747922 2C227661 6C756522 3A782274 65787422 3A22786E 6160653A 5C227461	
1209127136	796C6F72 5F73616C 666F7264 5C222C69 643A5C22 64356665 30313735 2D616562	
1209127168	33203344 64632061 6166322D 36373861 31383464 64376663 5C227022 707D2C22	
1209127200	74657874 223A2274 61796C6F 725F7361 6C666F72 6422702C 2248692C 20746869	
1209127232	73206973 20746573 74757365 72312061 6761696E 20616E64 20746869	
1209127264	65206F6E 20607920 73797374 6560206E 6F772069 73203230 3A353322 5D700077	
1209127296	732E2054 68652074 69606520 6F6E2060 79207379 73746560 20697320 32303A35	
1209127328	30225070 005F626F 61749116 60969E65 63726166 743A7265 63697065 732F726F	
1209127360	6F740000 0D656E74 65726564 5F77617A 65720E68 61735F74 68655F72 65636970	
1209127392	65010208 656E7465 7265645F 77617465 720E6861 735F7468 655F7265 63697065	

Fig. 12 Remnants of chat from Windows client stored in Linux memory

server was stored in memory alongside the connected username. The username and password for logging into the server existed in memory adjacent to the bash command `ifconfig`, as shown in Fig. 13.

1974720692	00000000 00000000 7461796C 6F720D70 61756C0D 6966636F 6E666967	taylor paul ifconfig
1974720720	001B5B41 1B5B411B 5B411B5B 42001B5B 411B5B41 1B5B411B 5B411B5B	[A [A [B [B [A [A [A [A [A
1974720748	411B5B41 1B5B411B 5B411B5B 411B5B41 1B5B411B 5B411B5B 411B5B41	A [A [A [A [A [A [A [A [A [A
1974720776	1B5B411B 5B411B5B 411B5B41 1B5B411B 5B411B5B 411B5B41 00000000	[A [A [A [A [A [A [A [A [A
1974720804	00000000 00000000 00000000 00000000 00000000 00000000	

Fig. 13 Server username and password

Although the server username and password do not reside in memory alongside any data that could easily be located to identify this information, the username is stored many times in memory following the string, “/home/”.

7 Network Analysis

A Minecraft game begins when the server operator logs in, starts up the Minecraft software and waits for incoming connections. A client can then choose to connect to either a recommended Microsoft server or manually enter an IP address and connect to a private server (desktop Java Edition only). Upon connection, a welcome message is displayed to the client in the form of a ‘message of the day’ and the user has a character/avatar immediately spawned into an expansive landscape ready for gameplay.

Upon successful connection, the Windows client immediately passes its Minecraft username to the server and the UUID is passed back in return. Then, a rapid series of packets is passed back and forth, which contain seemingly encoded data. Over the period of activity of approximately 5 min a significant volume of traffic was observed between the Linux server and Windows client. The average flow of data was 21 KB/s.

The open ports on the connected machines could be determined using the Wireshark *Statistics > Endpoints* feature. Figure 14 shows that the Linux machine (192.168.0.29) communicated through port 25565 (as set in the *server.properties* file) yet the Windows machine (192.168.0.82) incrementally changed the port number, albeit the majority of communication was on port 49804. The Windows port number was not pre-determined prior to running the software.

Fig. 14 Ports utilised by server and client

Address	Port	Packets	Bytes
192.168.0.29	25565	72,153	7133 k
192.168.0.82	49803	7	414
192.168.0.82	49804	72,132	7132 k
192.168.0.82	49806	14	1009

The contents of the conversation generated by the Windows client utilising the chat feature was sent via TCP in plaintext, as can be seen in Fig. 15.

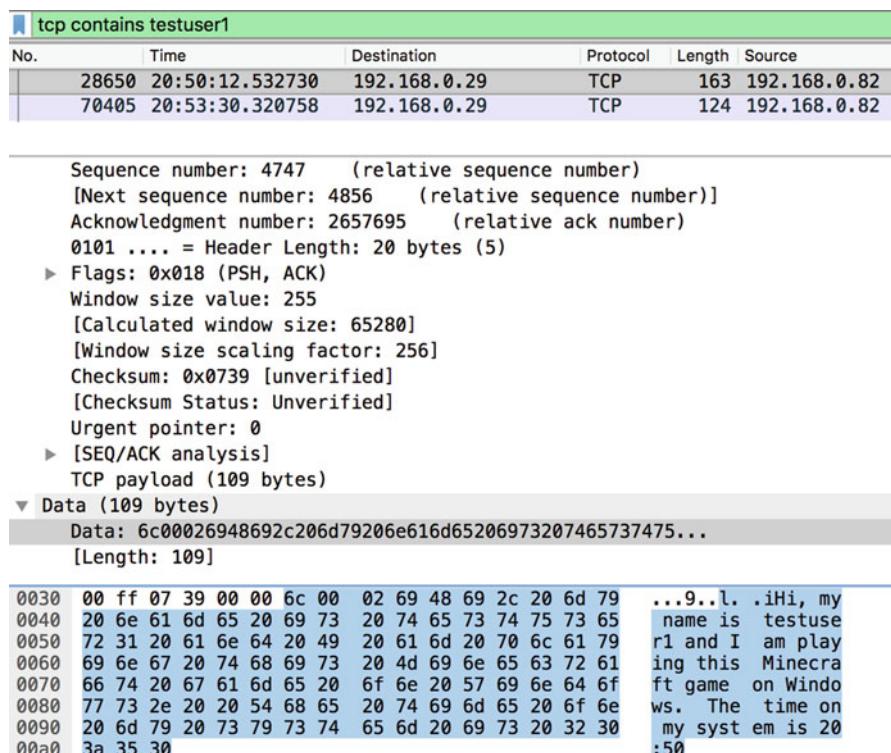


Fig. 15 Plaintext communication sent from Windows client

Similarly the server operator's message of the day was sent in plaintext, which can be located by searching for the identifiers that enclose the message. Figure 16 shows the message in its entirety and a 'tcp contains' search for { "description" } will allow for an examiner to locate the message of the day in captured traffic.

No.	Time	Destination	Protocol	Length	Source
72165	20:53:44.336014	192.168.0.82	TCP	217	192.168.0.29
<hr/>					
0000	00 0c 29 c0 86 c0 dc a9	04 8e 67 62 08 00 45 00	..)..... .gb..E.		
0010	00 cb 3d 8c 40 00 40 06	7a e1 c0 a8 00 1d c0 a8	..=@. z.....		
0020	00 52 63 dd c2 8e 33 c5	0c e6 06 29 d8 a0 50 18	.Rc...3.).P.		
0030	00 e5 72 a5 00 00 a1 01	00 9e 01 7b 22 64 65 73	..r..... ...{"des		
0040	63 72 69 70 74 69 6f 6e	22 3a 7b 22 74 65 78 74	cription ": {"text		
0050	22 3a 22 48 65 6c 6c 6f	2c 20 74 68 69 73 20 69	": "Hello , this i		
0060	73 20 61 20 74 65 73 74	20 6d 65 73 73 61 67 65	s a test message		
0070	20 6f 66 20 74 68 65 20	64 61 79 20 66 6f 72 20	of the day for		
0080	61 6c 6c 20 74 6f 20 73	65 65 2e 22 7d 2c 22 70	all to s ee."}, "p		
0090	6c 61 79 65 72 73 22 3a	7b 22 6d 61 78 22 3a 32	layers": { "max": 2		
00a0	30 2c 22 6f 6e 6c 69 6e	65 22 3a 30 7d 2c 22 76	0, "online": 0} }, "v		
00b0	65 72 73 69 6f 6e 22 3a	7b 22 6e 61 6d 65 22 3a	ersion": { "name":		
00c0	22 31 2e 31 32 2e 32 22	2c 22 70 72 6f 74 6f 63	"1.12.2" , "protoc		
00d0	6f 6c 22 3a 33 34 30 7d	7d	ol": 340 } }		

Fig. 16 Plaintext message of the day

TCP packets carrying information about the actual gameplay are not in a readable form and they appear to be encoded. Times of player idling patterns can be seen in the TCP stream, which include a large number of full stops and the number 6. An example of concatenated TCP packets sent from the Windows client to the Linux server during idling is:

```
.....&.....&.....&.....&.....&.....&.....&.....&.....&
.....&.....&.....&.....&.....&.....&.....&.....&.....&
.....&.....&.....&.....&.....&.....&.....&.....&.....&
.....&.....&.....&.....&.....&.....&.....&.....&.....&
&.....6..` ..&.....&.....&.....&.....&.....&.....&
..6..f..&.....&.....&.....&.....&.....&.....&.....&
....'[....[....B....6..A.'.....<.....6.....'[....'
4.j.N.....6.....
```

There is no evidence of the use of encryption for any of the games identifiers, user, communications or operations.

8 Conclusion

The research was able to determine that communications, player identifiers and IP addresses were all available on both the Linux server and the Windows client. There are several scenarios that could present themselves and allow for this research to apply in a live investigation:

- The Minecraft server is identified as being administered by an offender; evidence will exist on the machine of communication to connected users, which could include victims and associates, and identifying usernames and IP addresses that will allow for tracing and safeguarding of victims.
- The Minecraft server is a repository of evidence; any offending clients will have their username, IP address, port number and communications stored on the server to be made available to law enforcement.
- The Windows Minecraft client is operated by a victim of some crime; communications sent and received with others will be available. As will identifying information about other users and importantly the server operator, who will be able to provide more extensive data regarding the offending connected user.

One of the configurable *server.properties* settings is the port number. As Minecraft is a popular platform with many opportunities to create custom gaming opportunities, server hosting is popular in the community of gamers. Consideration should be given to server hosting from a home address, whereby port forwarded must be configured on a typical home broadband internet access point. Therefore, evidence of Minecraft server hosting could be located within the home broadband access point settings.

The memory dump of the Windows client provided the IP address and name of the Linux server. The message of the day is stored in plain text in the memory of the Windows client and in several areas of memory in the Linux server. The entire server properties configuration settings are not only found within the */home/[user]* folder, but also in their entirety along with timestamp in memory.

The Linux server assigns a UUID to each user, provides the user with a lease of this number of 1 month and stores this data in *usercache.json* file within */home/[user]*. The data also resides in Linux memory after the game has ceased and the file on disk has been erased. When investigations concern allegations of impersonation of a Minecraft gamer, consideration should be given to the importance of such UUIDs and it should be noted that the expiry time will always be one calendar month ahead of the date the user connected to the server.

The password to login to the Linux server and begin typing the commands necessary to run the server software was available in plaintext in memory alongside the user name. Even if the username for the server was not known it could easily be identified by conducting a search for “*/home/*” which quickly identified the server user account.

By acquiring the above results, the objectives of this research were met as follows:

1. Artefacts of evidential interest were retained on the Linux Minecraft server after installation of the software and activity during gameplay with a Windows client. Client IP addresses, chat logs, player data and server credentials were all available in unencrypted form.
2. Evidence of communication between client devices and the server was identified on all connected devices; communications over Minecraft’s chat feature were stored with timestamps on disk and in live memory.

3. In order to assist in identifying and profiling users, IP addresses were available along with UUIDs that remained the same for 1 month and the Windows client stored the e-mail address associated to the registered Minecraft user.
4. The network traffic revealed communications, the ‘message of the day’ IP addresses, port numbers, user names and operational commands, albeit in a seemingly encoded format.

Consideration for future work must be given to mobile platforms, with appreciation for the fact that unrooted devices can only make use of the official Minecraft client application and can only connect to verified Microsoft servers. There are third party applications available, such as PocketMine, which allow users of mobile devices to run a modified and unofficial version of Minecraft in order to connect to multiplayer servers that do not have to be subjected to verification by Microsoft. Research extended into these areas would enlighten investigators and give an appreciation for the popularity of unofficial and customisable Minecraft server and client platforms.

Acknowledgement We would like to thank the editor and anonymous reviewers for their constructive comments. The information and views set out in this paper are those of the author(s) and do not necessarily reflect the official opinion of institutes they are working at.

References

1. L. Christopher, K.-K. R. Choo, and A. Dehghantanha, *Honeypots for Employee Information Security Awareness and Education Training: A Conceptual EASY Training Model*. 2016.
2. A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, “Detecting crypto-ransomware in IoT networks based on energy consumption footprint,” *J. Ambient Intell. Humaniz. Comput.*, pp. 1–12, Aug. 2017.
3. S. Walker-Roberts, M. Hammoudeh, and A. Dehghantanha, “A Systematic Review of the Availability and Efficacy of Countermeasures to Internal Threats in Healthcare Critical Infrastructure,” *IEEE Access*, 2018.
4. Y.-Y. Teing, D. Ali, K. Choo, M. T. Abdullah, and Z. Muda, “Greening Cloud-Enabled Big Data Storage Forensics: Syncany as a Case Study,” *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2017.
5. Y.-Y. Teing, A. Dehghantanha, and K.-K. R. Choo, “CloudMe forensics: A case of big data forensic investigation,” *Concurr. Comput.*, 2017.
6. D. Kiwia, A. Dehghantanha, K.-K. R. Choo, and J. Slaughter, “A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence,” *J. Comput. Sci.*, Nov. 2017.
7. M. Hopkins and A. Dehghantanha, “Exploit Kits: The production line of the Cybercrime economy?,” in *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, 2015, pp. 23–27.
8. A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, “Machine Learning Aided Static Malware Analysis: A Survey and Tutorial,” 2018, pp. 7–45.
9. H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, “Intelligent OS X malware threat detection with code inspection,” *J. Comput. Virol. Hacking Tech.*, 2017.
10. M. Petraityte, A. Dehghantanha, and G. Epiphanou, *A model for android and iOS applications risk calculation: CVSS analysis and enhancement using case-control studies*, vol. 70. 2018.

11. Y.-Y. Teing, A. Dehghantanha, K.-K. R. Choo, and L. T. Yang, "Forensic investigation of P2P cloud storage services and backbone for IoT networks: BitTorrent Sync as a case study," *Comput. Electr. Eng.*, vol. 58, pp. 350–363, Feb. 2017.
12. H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting," *Futur. Gener. Comput. Syst.*, 2018.
13. M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things security and forensics: Challenges and opportunities," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan. 2018.
14. G. Epiphaniou, P. Karadimas, D. K. Ben Ismail, H. Al-Khateeb, A. Dehghantanha, and K.-K. R. Choo, "Non-Reciprocity Compensation Combined with Turbo Codes for Secret Key Generation in Vehicular Ad Hoc Social IoT Networks," *IEEE Internet Things J.*, pp. 1–1, 2017.
15. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing," *EURASIP J. Wirel. Commun. Netw.*, vol. 2016, no. 1, p. 130, May 2016.
16. S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha, and R. Khayami, "BoTShark: A Deep Learning Approach for Botnet Traffic Detection," 2018, pp. 137–153.
17. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence," *{IEEE} Trans. Emerg. Top. Comput.*, p. 1, 2017.
18. A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning," *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2018.
19. K. R. Choo, "Online child grooming: a literature review on the misuse of social networking sites for grooming children for sexual offences," *Aust. Inst. Criminol.*, p. 132, 2009.
20. L.-K. Bernstein, "Investigating and Prosecuting Swatting Crimes," *United States Atty. Bull.*, vol. 64, no. 3, pp. 51–56, 2016.
21. A. Carpinteri, B. Bang, K. Klimley, R. A. Black, and V. B. Van Hasselt, "Commercial Sexual Exploitation of Children: an Assessment of Offender Characteristics," *J. Police Crim. Psychol.*, Aug. 2017.
22. H. Hillman, C. Hooper, and K.-K. R. Choo, "Online child exploitation: Challenges and future research directions," *Comput. Law Secur. Rev.*, vol. 30, no. 6, pp. 687–698, Dec. 2014.
23. L. Achternbosch, C. Miller, C. Turville, and P. Vamplew, "Griefers versus the Grieved - what motivates them to play Massively Multiplayer Online Role-Playing Games ?," *Comput. Games J. Ltd.*, vol. 3, no. 1, 2014.
24. E. M. Jaffe, "Article : Swatting : the New Cyberbullying Frontier After Elonis V. United States," *Drake Law Rev.*, pp. 455–483, 2016.
25. A. Choo and A. May, "Maintaining Long Distance Togetherness Synchronous Communication with Minecraft and Skype," in *Games Innovation Conference (IGIC), 2013 IEEE International*, 2013.
26. A. Noroozian, M. Korczyński, C. H. Gañan, D. Makita, K. Yoshioka, and M. van Eeten, "Who Gets the Boot? Analyzing Victimization by DDoS-as-a-Service," in *Research in Attacks, Intrusions, and Defenses: 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings*, F. Monroe, M. Dacier, G. Blanc, and J. Garcia-Alfarro, Eds. Cham: Springer International Publishing, 2016, pp. 368–389.
27. J. Taylor, "Online investigations: protection for child victims by raising awareness," *ERA Forum*, vol. 16, no. 3, pp. 349–358, 2015.
28. S. Khanji, R. Jabir, F. Iqbal, and A. Marrington, "Forensic analysis of xbox one and playstation 4 gaming consoles," *8th IEEE Int. Work. Inf. Forensics Secur. WIFS 2016*, 2017.
29. M. Cheah, L. Wyndham-Birch, and B. Bird, "What artifacts of evidentiary value can be found when investigating multi-user virtual environments." 2015.
30. "What OS for server? - Server Administration - Server Support - Support - Minecraft Forum - Minecraft Forum." .

31. D. Quick and K.-K. R. Choo, "Google Drive: Forensic analysis of data remnants," *J. Netw. Comput. Appl.*, vol. 40, pp. 179–193, Apr. 2014.
32. D. Quick and K.-K. R. Choo, "Dropbox analysis: Data remnants on user machines," *Digit. Investig.*, vol. 10, no. 1, pp. 3–18, Jun. 2013.
33. L. C. for D. Investigation, "1/21/2016 175," *Leahy Center for Digital Investigation*, no. 802. 2016.
34. A. Rutkin, "Your place or Minecraft?," *New Sci.*, vol. 230, no. 3071, pp. 22–23, 2016.
35. "Mojang - Minecon 2015 - Day Two - Twitch." .
36. Z. Zhang, H. Anada, J. Kawamoto, and K. Sakurai, "Detection of illegal players in massively multiplayer online role playing game by classification algorithms," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, vol. 2015–April, pp. 406–413, 2015.
37. Y. Ki, J. Woo, and H. K. Kim, "Identifying Spreaders of Malicious Behaviors in Online Games," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 315–316.
38. J. Oh, Z. H. Borbora, and J. Srivastava, "Automatic Detection of Compromised Accounts in MMORPGs," in *2012 International Conference on Social Informatics*, 2012, pp. 222–227.
39. A. S. V Nair and B. A. S. Ajeena, "A Log Based Strategy for Fingerprinting and Forensic Investigation of Online Cyber Crimes," in *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, 2014, p. 7:1–7:5.
40. M. Barni and B. Tondi, "Threat Models and Games for Adversarial Multimedia Forensics," in *Proceedings of the 2nd International Workshop on Multimedia Forensics and Security*, 2017, pp. 11–15.
41. S. Rajendran and N. P. Gopalan, "Mobile Forensic Investigation (MFI) Life Cycle Process for Digital Data Discovery (DDD)," in *Proceedings of the International Conference on Soft Computing Systems: ICSCS 2015, Volume 2*, L. P. Suresh and B. K. Panigrahi, Eds. New Delhi: Springer India, 2016, pp. 393–403.
42. T. Dargahi, A. Dehghantanha, and M. Conti, "Chapter 2 - Forensics Analysis of Android Mobile VoIP Apps," in *Contemporary Digital Forensic Investigations of Cloud and Mobile Applications*, K.-K. R. Choo and A. Dehghantanha, Eds. Syngress, 2017, pp. 7–20.
43. K. K. R. Choo and A. Dehghantanha, *Contemporary Forensic Investigation of Cloud and Mobile Applications*. 2017.
44. ACPO, "ACPO Good Practice Guide for Digital Evidence," Association of Chief Police Officers, 2012.
45. N. C. J. U.S. Department of Justice, "Electronic Crime Scene Investigation: A Guide for First Responders," *NIJ Res. Rep.*, no. NCJ 187736, p. 96, 2001.
46. F. S. Regulator, "Codes of Practice and Conduct Issue 4," 2017.
47. K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to Integrating Forensic Techniques into Incident Response," *The National Institute of Standards and Technology*, 2006.
48. A. Antwi-Boasiako and H. Venter, "A Model for Digital Evidence Admissibility Assessment," in *Advances in Digital Forensics XIII: 13th IFIP WG 11.9 International Conference, Orlando, FL, USA, January 30 - February 1, 2017, Revised Selected Papers*, G. Peterson and S. Shenoi, Eds. Cham: Springer International Publishing, 2017, pp. 23–38.
49. B. Martini and K.-K. R. Choo, "An integrated conceptual digital forensic framework for cloud computing," *Digit. Investig.*, vol. 9, no. 2, pp. 71–80, Nov. 2012.
50. Y.-Y. Teing, D. Ali, K.-K. R. Choo, M. Conti, and T. Dargahi, "Forensic Investigation of Cooperative Storage Cloud Service: Symform as a Case Study," *J. Forensics Sci.*, vol. [In Press], 2016.
51. T. Alstad *et al.*, "Minecraft computer game performance analysis and network traffic emulation by a custom bot," *Proc. 2015 Sci. Inf. Conf. SAI 2015*, pp. 227–236, 2015.

Big Data Forensics: Hadoop Distributed File Systems as a Case Study



Mohammed Asim, Dean Richard McKinnel, Ali Dehghantanha,
Reza M. Parizi, Mohammad Hammoudeh, and Gregory Epiphaniou

Abstract Big Data has fast become one of the most adopted computer paradigms within computer science and is considered an equally challenging paradigm for forensics investigators. The Hadoop Distributed File System (HDFS) is one of the most favourable big data platforms within the market, providing an unparalleled service with regards to parallel processing and data analytics. However, HDFS is not without its risks, having been reportedly targeted by cyber criminals as a means of stealing and exfiltrating confidential data. Using HDFS as a case study, we aim to detect remnants of malicious users' activities within the HDFS environment. Our examination involves a thorough analysis of different areas of the HDFS environment, including a range of log files and disk images. Our experimental environment was comprised of a total of four virtual machines, all running Ubuntu. This HDFS research provides a thorough understanding of the types of forensically relevant artefacts that are likely to be found during a forensic investigation.

M. Asim · D. R. McKinnel

Department of Computer Science, University of Salford, Manchester, UK

e-mail: m.asim1@edu.salford.ac.uk; d.r.mckinnel@edu.salford.ac.uk

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

e-mail: ali@cybersciencelab.org

R. M. Parizi

Department of Software Engineering and Game Development, Kennesaw State University, Marietta, GA, USA

e-mail: rparizi@kennesaw.edu

M. Hammoudeh

School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester, UK

e-mail: M.Hammoudeh@mmu.ac.uk

G. Epiphaniou

Wolverhampton Cyber Research Institute (WCRI), School of Mathematics and Computer Science, University of Wolverhampton, Wolverhampton, UK

e-mail: G.Epiphaniou@wlv.ac.uk

Keywords HDFS · Digital forensics · Hadoop · Big data · Distributed file systems

1 Introduction

Human dependence on technology has become an inadvertent side-effect of the transition into the digital age, which has subsequently resulted in an exponential increase in data production [1]. This growth of heterogeneous data has led to a relatively new computing paradigm evolving in recent years, this paradigm aptly being named Big Data [2]. Günther et al. [3] explicitly evaluated the current literature associated with the importance of Big Data, highlighting the value of Big Data for organisations as a means of building data-driven business models. Consequently, popularity surrounding Big Data environments has led to its widespread adoption, with many different types of industry utilising the environments for their own means. Companies like the Bank of America, Deutsche Bank and other large financial conglomerates are known users of Hadoop, utilising data analytics to fight against fraudulent activity and present customers with tailored packages based on transactions [4, 5]. The Hadoop ecosystem is developing significant popularity, largely due to Hadoop's efficiency in parallel data-processing and low fault tolerance [6]. Yu and Guo [7] emphasise the extent of Hadoop adoption when they make reference to the large percentage (79%) of investment in Hadoop by companies attending the Strata Data conference. Demand for Big Data analytics has become so vast that major business intelligence companies like that of IBM and Oracle have developed their own Big Data platforms as a means of allowing companies without extensive infrastructure to take advantage of Big Data analytics [8]. In addition to these elastic Big Data environments, Vorapongkitipun and Nupairoj [9] state that the Hadoop Distributed File System is extremely lightweight and can therefore be hosted using commodity hardware, thus making it more accessible to the average user for storing smaller file types in a distributed manner.

There is a cornucopia of forensic problems associated with the adoption of Big Data environments from both the perspective of companies and individuals alike [10]. Tahir and Iqbal [1] specify that Big Data is one of the top issues facing forensic investigators in coming years due to the complexity and potential size of Big Data environments. As taxonomized by Fu et al. [11], criminals with access to a company's confidential Hadoop environment can perform data leakage attacks from different platform layers within the Hadoop ecosystem. The authors also extend the problem of permissions to the Hadoop audit logs, specifying that attackers can make changes to these logs as a means of anti-forensics. Moreover, Big Data storage have been targeted by a range of attack vectors from Ransomware [12–14] and Trojans [15] to Distributed Denial of Service (DDoS) [16] attacks.

Malicious actors may also utilise Hadoop and the HDFS environment as a means of performing a criminal activity as they have done previously in cloud platforms such as Mega [17] and SugarSync [18]. Almulla et al. [19] illustrate how criminals

can use such settings as a means of hiding and distributed illicit content with generalised ease. This consensus is also shared by Tabona and Blyth [20], in which the authors indicate that complex technologies like that of Hadoop have become readily available to criminals, subsequently resulting in crimes with digital trails that are not so easily discovered. Gao and Li [21] emphasise this complexity by identifying the difficulty in forensic extraction.

As well as problems attributed to the criminals themselves, Guarino [22] outlines problems associated with the sheer size of an HDFS environment. The potential area for analysis in an HDFS environment can be substantial, therefore making it extremely difficult to interpret information of forensics significance efficiently. This enormity can be an inherent problem during the data collection phase of forensic analysis, as the gigantean size of the dataset can correlate positively with the potential for errors during collection. Subsequently, errors in the collection can propagate in the latter stages of the investigation, therefore invalidating the integrity of the overall investigation [23].

If the HDFS environment is not already difficult to forensically interpret, the proposal of a secure deletion method by Agrawal et al. [24] may further complicate the forensic practices associated with Hadoop and the distributed file system.

As outlined above, there is a requirement for the analysis of HDFS forensically as a means of determining all the forensically viable information that may be extracted from an environment. The potential for anti-forensic techniques creates a need for more forensic analysis to take place to ensure that forensic examiners are always one step ahead of criminals. In its infancy, Big Data is a paradigm that relatively unexplored regarding forensics. Therefore there is an evident need for in-depth digital forensic analysis of the environment.

In this paper, we seek to answer the followings:

1. Does the Hadoop Distributed File System offer any forensic insight into data deletion or modification, including the change of file metadata?
2. Can the memory and file systems of the Nodes within the HDFS cluster be forensically analysed and compared as a means of gaining forensic insight by obtaining forensic artefacts?
3. Can forensically significant artefacts be obtained from auxiliary modules associated with HDFS, i.e. web interfaces?
4. Does any network activity occur in HDFS between the respective nodes that may attribute valuable forensic information?

It is expected that the findings of this research will help the forensic communities further understanding of the forensic significant components of the Hadoop Distributed File System.

This paper is structured as follows; Sect. 2 begins by giving an overview of current related research on Big Data forensics with an emphasis on Hadoop, focusing primarily on HDFS. This overview will allow the forensic investigation performed in this paper to become contextualised, emphasising the importance of research into this field. Following the literature review, Sect. 3 outlines the research

methodology that will be employed in this paper, ensuring that the investigation coincides with accepted forensic frameworks. Section 4 encompasses the main experimental setup and collection of the testing data. Section 5 presents the experimental results and analysis, discussing the findings of the investigation in this research. Finally, Sect. 6 concludes the investigation and extrapolates the findings to develop ideas about future work.

2 Related Work

Despite the relatively new adoption of Big Data analytics, several researchers have begun to forensically uncover HDFS and other distributed file system environments as a means of broadening the forensic community's insight into these systems. Majority of Big Data forensics efforts were based on procedures developed previously for cloud investigation [25] such as those used for analysing end-point devices [26, 27], cloud storage platforms [28, 29], and network traffic data [30, 31]. Forensic guidelines have also been developed as a means of building acceptable frameworks for the use of distributed file system forensics [32].

Martini and Choo [33] use XtreemFS as a big data forensics case study with an emphasis on performing an in-depth forensic experiment. The authors focus on the delineation of certain digital artefacts within the system itself, in conjunction with highlighting the issues regarding the collection of forensic data from XtreemFS.

Thanekar et al. [34] have taken a holistic approach to Hadoop forensics by analysing the conventional log files and identifying the different files generated within Hadoop. The authors made use of accepted forensic tools, e.g. *Autopsy* to recover the various files from the Hadoop environment.

Leimich et al. [35] utilise a more calculated method of forensics by performing triage of the metadata stored within the RAM of the Master NameNode to provide cluster reconnaissance, which can later be used at targeted data retrieval. This research focusses predominately on establishing an accepted methodology for use by forensic investigators with regards to RAM of the Master NameNode of a multi-cluster Hadoop environment. Despite the promising results gained by the investigators in data retrieval of the DataNodes, the research itself does not provide a means of identifying user actions within the environment itself.

A range of forensic frameworks for significant data analysis has been proposed in contrast to convention methods of forensic analysis and frameworks. Gao et al. [36] suggest a new framework for forensically analysing the Hadoop environment named *Haddle*. The purpose of this framework is to aid in the reconstruction of the crime scene, identifying the stolen data and more importantly, the user that perpetrated the action within Hadoop. Like the work mentioned above, the authors use Haddle and other tools to create custom logs to identify users performing specific actions, thus identifying the culprit.

Dinesh et al. [37] propose a similar incident-response methodology is utilising logs within the Hadoop environment. The analysis of these logs aids in determining the timeline at which an attack occurred as subsequent comparisons can be made with previous logs that encompass the current state of the systems at that time.

The use of an HDFS specific file carving framework is proposed by Alshammari et al. [38], in which the authors identify the difficulties of file carving in bespoke distributed file systems that do not utilise conventional methods of storing data. This research sets out specific guidelines for the retrieval of HDFS data without corruption, summarising that to retrieve data from HDFS using file carving that the raw data needs to be treated not unlike that found in conventional file systems with only the initial consolidation being an issue.

As mentioned the current state of Big Data forensics is relatively unexplored due to the infancy of the area [39]. However, it is evident that researchers are moving into this area of forensics as a means of understanding all aspects of new systems relevant to Big Data with an emphasis on distributed systems [40]. It is plain to see from the current research that areas of the Hadoop infrastructure are yet to be explored on a comprehensive level, as most researchers seem to stick to what they can see as oppose to what they can't. A general trend in Hadoop and other file system forensics seems to be the analysis of log files. Despite the inarguable richness of forensic data contained with system logs, the majority of papers mentioned above hold scepticism over the heavy reliance on these files in forensics. Where mentioned in the papers above, logs are deemed as a fragile source of forensic information, as these files can be easily tampered with or deleted. Inspired by this fact and for the further advancing the Big Data forensic research, there is a need to explore the Hadoop Distributed File System to retrieve forensic artefacts that may appear in the infrastructure. The emphasis should be on determining such aspects like the perpetrating user of action or time associated with any of actions with the Hadoop environment. Our work proposed in this research took the first step to address this need in the community.

3 Methodology

Digital forensic frameworks play a significant role during a forensics investigation. The previous research presents many reasons why a digital forensics investigation would face challenges and failures. As mentioned by Kohn et al. [41], the main concern is due to lack of upfront planning subsequently avoiding best security practices. To ensure best security practices, an investigator must adhere to security guidelines and standards, and this can be achieved by following a suitable framework.

As outlined by the National Institute of Standards, digital forensics is divided into four stages; collection, examination, analysis, and reporting. Alex and Kishore [42] have also followed a very similar structure to the NIST with their stages including identification, collection, organisation and presentation.

In this paper, we have adopted the cloud investigations framework recommended by Martini and Choo [43]. This framework contains elements that are very similar to research performed by Alex and Kishore [42] and the NIST, with a few distinct differences. The main difference being the presence of an iterative cycle that allows the investigator to go back and forth from stages, thus being able to re-examine and make changes much earlier in the investigation. This is advantageous to the proposed experiment, as it offers flexibility and an opportunity to explore additional technologies that may prove more efficient in obtaining forensically relevant results. Figure 1 below illustrates the four stages of the cloud investigations framework. Each stage has been explained regarding the research carried out within this paper.

1. *Evidence source identification and preservation:* In this stage, we identified the main components of the environment within HDFS. A forensic copy of the VMEM, VMDK and FSImages (secondary NameNode checkpoint) were generated for each of the respective nodes at each interval of action. Also, network captures were carried out during each action as a means of analysing the network transactions. We verified the integrity of each of the data files (including the dataset) by calculating the hash value associated with them. Windows PowerShell was selected to verify the integrity of each of the files collected using the command *Get-FileHash -Algorithm MD5*.
2. *Collection:* During this stage, we collected a varied array of different evidence from the files outlined in the previous section. Each piece of granular evidence retrieved from the main evidence files was attributed its hash.
3. *Examination and Analysis:* The purpose of this stage was to examine and analyse the data collected. Due to the composition of the metadata of the NameNode and data-blocks in HDFS string searching was the effective tool used for analysis in both the VMEM captures, network captures and the VMDK captures. The string searching involved finding appropriate information with regards to aspects such as block IDs, dataset file name or text within the sample dataset. A secondary stage of the analysis is to analyse log information within the NameNode and FSImage information supplied by the secondary NameNode, as a means of discovering more information that can validate findings within the environment.
4. *Reporting and presentation:* This stage will contain all the evidence found during the investigation carried out within this paper, including a legal document that can be used in a real-life court environment.

4 Experimental setup

Our forensic experiment involved a multi-node cluster setup using Hadoop 2.7.4 based on Ubuntu-16.04.3 LTS. As shown in Fig. 1 the cluster consisted of four nodes; a Master NameNode, a secondary NameNode and two DataNodes. Snapshots of each of the machine states were taken following the completion of one of the base functions within the Hadoop distributed file system. The actions being analysed

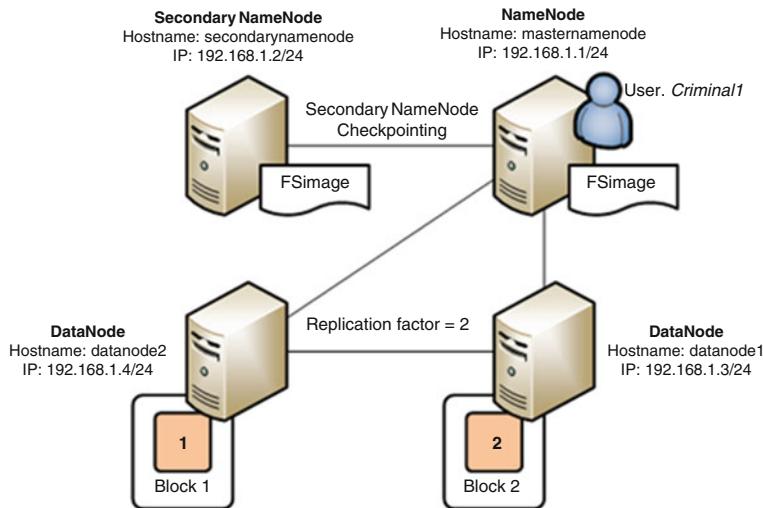


Fig. 1 HDFS cluster showing the FSImage checkpoint communication between the NameNode and Secondary NameNode, as well as the replication of the dataset over the two DataNodes

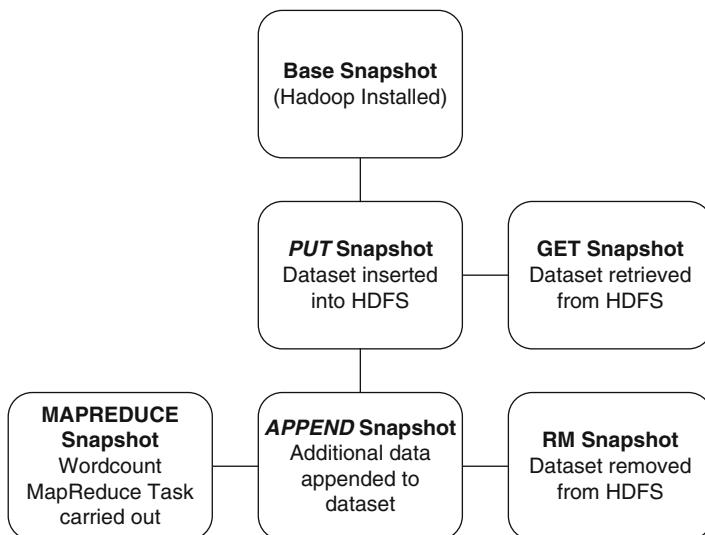


Fig. 2 The snapshots taken within each stage of the HDFS forensic analysis. All systems will be snapshotted during each interval if action

within HDFS were the PUT, GET, APPEND, REMOVE and the transactional processes of a simple MapReduce job. A base snapshot was taken before any data was inserted into HDFS and no actions have been performed. This base snapshot

offered a basis for comparison with other snapshots. Figure 2 illustrates the Snapshot hierarchy of all snapshots taken within the analysis.

The cluster itself was housed within a virtual environment using VMware Workstation as a means of saving time in analysis, as to create a physical Hadoop cluster with all the intermediate components involved would be an overly time-consuming exercise. Each VM was also given its minimum running requirements and disk space to save time during the analysis of the VMDKs. As stated by Rathbone [44], HDFS is designed only to take a small range of file types for use in its analysis; these include; plain text storage such as CSV or TSV files, Sequence Files, Avro or Parquet. To reduce complications, the use of a CSV dataset was chosen from U.S. Department of Health that contained information of Chronic Disease Indictors; this would aid in string searching later.

It should be noted that in a real-life scenario a commercial Hadoop cluster would be unable to be powered down to obtain forensic copies of the environment, therefore live data acquisition would need to be utilised to obtain the information obtained within this experiment. To reduce the potential for error in data acquisition, the experiment made use of the VMEM and VMDK files within the virtual machine directories. This allowed real-time acquisition of the virtual machine states while and after the HDFS processes were performed. In the instances of the FSImages, the inbuilt tools of HDFS were utilised to examine these files. These consisted of both the Offline Image Viewer and Offline Edits Viewer, which gave an overview of operations performed, as well as granular information about each operation. FTK Imager was used to examine the VMDK file systems of each of the related snapshots with an emphasis on recovering certain log or configuration files. Finally, Wireshark was used during each action to compile communications occurring between each of the respective nodes. Table 1 outlines the tools utilised within this experiment, with Table 2 describing the specifications of each snapshot.

Table 1 Tools used during analysis

Tool (Incl. Version)	Usage
Access Data FTK Imager 3.4.3.3	To analyse the file system structures and files of VMDK files recovered from the virtual machine snapshots
HxD (Hexeditor) 1.7.7.0	To analyse the VMEM files recovered from the virtual machine snapshots using string searches of the Hex data
Offline Image Viewer (Hadoop Tool)	To analyse the secondary NameNode FSImages for each of the snapshots
Offline Edits Viewer (Hadoop Tool)	To analyse the secondary NameNode edit logs for each of the snapshots
Wireshark 2.4.2	To analyse network capture files
VMware Workstation Pro 12.5.0	To host the virtual machines

Table 2 Snapshot specification

Snapshot	Details
Base snapshot	A base snapshot with Hadoop installed, no data within cluster
PUT snapshot	Copy of the base snapshot with data inserted into the cluster using the PUT command in HDFS
GET snapshot	Copy of the PUT Snapshot with the HDFS data retrieved using the GET command in HDFS
APPEND snapshot	Copy of the PUT Snapshot with the <i>jellyfish.mp4</i> file appended to the original dataset
REMOVE snapshot	Copy of APPEND Snapshot with the data removed from HDFS
MAPREDUCE snapshot	Copy of PUT Snapshot with MapReduce configured

5 Results and Discussion

The following section is structured using each of the prominent actions performed in HDFS, showing the step-by-step analysis of the environment relevant to these actions.

5.1 Forensic Analysis of HDFS PUT

The HDFS *PUT* command (*hdfs dfs -put*) allows files within the local Linux file system to be distributed between the DataNodes of HDFS respective to their size and the replication factor configured within the core-site.xml file of the Hadoop configuration files. To begin the forensic analysis, it was imperative

The screenshot shows the Hadoop web interface with two main sections: "NameNode Journal Status" and "NameNode Storage".

NameNode Journal Status:

- Current transaction ID: 44
- Journal Manager: FileJournalManager(root+[/usr/local/hadoop_tmp/hdfs/namenode](#))
- State: EditLogFileOutputStream([/usr/local/hadoop_tmp/hdfs/namenode/current/edits_inprogress_0000000000000000044](#))

NameNode Storage:

Storage Directory	Type	State
/usr/local/hadoop_tmp/hdfs/namenode	IMAGE_AND_EDITS	Active

Fig. 3 Hadoop web interface showing location of NameNode Edit Log Files

that the location of the NameNode FSImage files and edit logs were located. Fortuitous for the investigation, the main page of the Hadoop web interface (<http://localhost:50070/dfshealth.html#tab-overview>) presented the exact location of these files along with other pertinent information such as the transactional ID associated with the current Edit in progress (see Fig. 3). The location configured was within the Master NameNode's (192.168.1.1) file directory at `/root/usr/local/hadoop_tmp/hdfs/namenode/`.

Upon browsing to this location within the VMDK image file using FTK Imager, the files `edits_0000000000000000xx-0000000000000000xx`, `fsimage_0000000000000000xx` and `VERSION` were found to exist. These files play a significant role in the forensic analysis of the environment as when analysed correctly they offer an abundance of information relative to the environment. This information includes certain forensically significant information such as timestamps (accessed time etc.), transaction owner as well as the names of files blocks etc. Using the inbuilt Hadoop tools *Offline Image Viewer (OIV)* and *Offline Edits Viewer (OEV)* the FSImage can be consolidated into both text and XML files that show actions within Hadoop, as well as the metadata associated with these actions. Whilst on the live Master NameNode, the command “`hdfs oiv -i /usr/local/hadoop_tmp/hdfs/namenode/ fsimage_0000000000000000xx -o /Desktop/oivfput.txt`” was utilised to interpret the current FSImage.

```
<INodeSection><lastInodeId>16386</lastInodeId><inode><id>16385</id><type>DIRECTORY</type><name></name><ctime>1513782878676</ctime><permission>c:mrw-r--r--</permission><crlnl>supergroup:rwxr-xr-x</crlnl><nsquota>9223372036854775807</nsquota><dsquota>1</dsquota></inode><inode><id>16386</id><type>FILE</type><name>U.S. Chronic Disease Indicators_CDI_.csv</name><replication>2</replication><ctime>1513782878571</ctime><atime>1513782858757</atime><preferredBlockSize>134217728</preferredBlockSize><permission>c:mrw-r--r--</permission><blocks><block><id>1073741825</id><genstamp>1001</genstamp><numBytes>123570042</numBytes></block></blocks></inode>
```

Fig. 4 Output of the Offline Image Viewer text file outlining a range of information regarding the dataset and associated blocks

This command consolidates both the FSImage logs to produce information about the current NameNode metadata. As seen in Fig. 4, the information regarding files within HDFS is logged with all their associated data in the FSImage. After the *PUT* action had been finished the NameNode stores information about the location of this data. This is done in both the HDFS block and the relative Inode location data regarding the Linux file system itself. Forensically significant aspects of this data are present in both the modification time and the access time, as well as the permissions for this data. Following the acquisition of this data, the edit logs within the NameNode were also analysed using the Offline Edits Viewer, which like the Offline Image Viewer consolidates the edit logs with regards to the transactional processes that have occurred within the time between the last checkpoint of the NameNode metadata and the current checkpoint. The command “`hdfs oev -i /usr/local/hadoop_tmp/hdfs/namenode/ edits_0000000000000000xx-0000000000000000xx -o /Desktop/baseedit.txt -p stats`” was used to create an

```

VERSION : -63
OP_ADD      ( 0): 1
OP_RENAME_OLD ( 1): 1
OP_DELETE    ( 2): null
OP_MKDIR     ( 3): null

```

Fig. 5 Output of the Offline Edits Viewer showing the transaction summary for the checkpoint

overview of the transactional processes that had occurred. Proceeding the *PUT* command, the transactional process added was the *OP_ADD* operation (see Fig. 5). As a means of further investigation, the Offline Edits Viewer was used to gain granular information regarding this transaction. Using the “*hdfs oev -i edits_0000000000000000xx-0000000000000000xx -o /Desktop/baseedits2.xml -v*” created an XML file that contained more verbosity regarding the operations in the checkpoint.

The output file created as a result of this command includes a large amount of granular information surrounding the *OP_ADD* operation. As seen in Fig. 6, the Inode number for the data within the Linux file system is noted first, thus supplying the investigator a means of retrieving the information based on location in such a scenario where no other information regarding the data is to be found. The name of the file inserted into HDFS is also located in this XML file as well as the replication number, thus showing that the data has repeatedly been striped within HDFS allowing a higher chance of retrieval from the data-blocks themselves. Despite a creation time timestamp not being present, under the consideration that the modification time and accessed time are the same relative to the *OP_ADD* operation, it is safe to assume that this epoch is the time of creation. A critical piece of information regarding the operation is the owner of that operation. In this case, it is evident that *criminal1* acted inserting the sample file into HDFS.

```

<OPCODE>OP_ADD</OPCODE>
<DATA>
  <TXID>18</TXID>
  <LENGTH>0</LENGTH>
  <INODEID>16386</INODEID>
  <PATH>/U.S._Chronic_Disease_Indicators__CDI_.csv._COPYING_</PATH>
  <REPLICATION>2</REPLICATION>
  <MTIME>1513782858757</MTIME>
  <ATIME>1513782858757</ATIME>
  <BLOCKSIZE>134217728</BLOCKSIZE>
  <CLIENT_NAME>DFSClient_NONMAPREDUCE_1292467381_1</CLIENT_NAME>
  <CLIENT_MACHINE>192.168.1.1</CLIENT_MACHINE>
  <OVERWRITE>true</OVERWRITE>
  - <PERMISSION_STATUS>
    <USERNAME>criminal1</USERNAME>
    <GROUPNAME>supergroup</GROUPNAME>
    <MODE>420</MODE>

```

Fig. 6 Granular output of the Offline Edits Viewer showing detailed information of HDFS *PUT* transaction

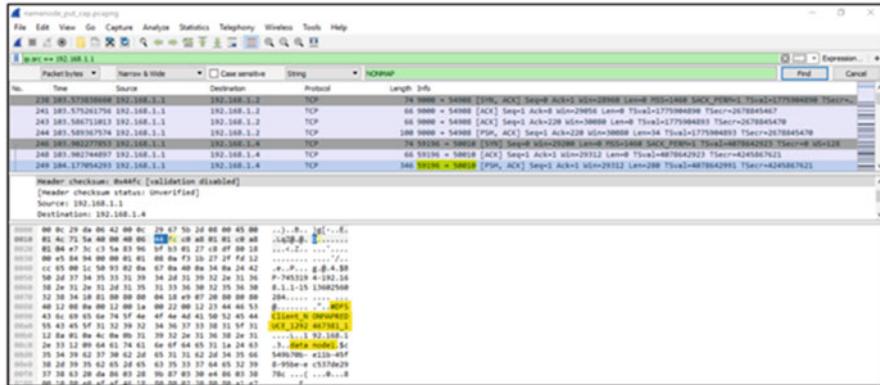


Fig. 7 Network capture occurring between the NameNode and DataNode

In addition to the information pertaining to the creation time, location and names of the file, other forensic information can be gleaned from the Edits Viewer output that coincides with additional forensic information obtained during the investigation. Information in the `<CLIENT_NAME>DFSClient_NONMAPREDUCE_1292 467381_I</CLIENT_NAME>` section of the edits output is also seen in the network communications between the NameNode and the individual DataNodes that are presented with the data (see Fig. 7). This initial communication occurs over ports 59196 on the NameNode to ports 50010 on the DataNode, which according to Zeyliger [45] correspond with the default Hadoop ports for each of these transactions. This packet is closely followed by the transfer of the file information from the NameNode to the DataNodes over the same ports.

Assessing the remaining operations with the edits log showed that block allocations occurred directly after a file being added so that HDFS can place the information on the respective blocks within the DataNodes. As the CSV dataset was relatively small in comparison to the size of datasets that would occur on a commercial cluster, only one block ID was allocated to the DataNodes. As seen in Fig. 8, Block ID 1073741825 with a GENSTAMP of 1001 was generated for the information.

```

<OPCODE>OP_ADD_BLOCK</OPCODE>
- <DATA>
  - <TXID>21</TXID>
  - <PATH>/U.S._Chronic_Disease_Indicators_CDT_.csv._COPYING_</PATH>
  - <BLOCK>
    <BLOCK_ID>1073741825</BLOCK_ID>
    <NUM_BYTES>0</NUM_BYTES>
    <GENSTAMP>1001</GENSTAMP>
  </BLOCK>

```

Fig. 8 Offline Edits Viewer output showing the block allocation ID associated with the file being added

Following the discovery of the block identification number, attention was turned to the VMDK files of the DataNodes, to determine if this information could be retrieved from the Linux File Directory that sits below the logical HDFS file directory. Upon analysing the VMDK of the NameNode, it was determined that the Hadoop configuration files were located in the `/usr/local/hadoop-2.7.4/etc/hadoop/` (see Fig. 9). Within this folder was the `hdfs-site.xml` file which contained key information about key configuration items within the Hadoop infrastructure. It was found that the DataNode data directory path was located at `/usr/local/hadoop_tmp/hdfs/datanode` on the DataNode (see Fig. 10). The DFS replication value was also located in this file, therefore showing that some level of replication had been configured, validating the previous findings of replication occurring.

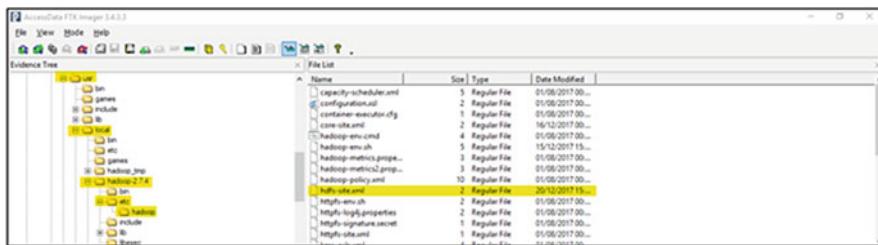


Fig. 9 File structure of NameNode showing the location of the Hadoop configuration files

```
<property>
    <name>dfs.replication</name>
    <value>2</value>
</property>
<property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
</property>
</configuration>
```

30.1

Bot

Fig. 10 Contents of the `hdfs-site.xml` file showing the DataNode data directory

Further analysis of the DataNode image using FTK Imager revealed that browsing to the directory specified in the `hdfs-site.xml` showed the block files associated with the file as ascertained earlier in the investigation. These block files contained the raw information for the CSV file along with metadata relevant to each of the block files. Additional forensic information was procured through the file directory itself, as the node ID (*BP-7456194*) itself was found within the file path of the DataNode data directory (see Fig. 11).

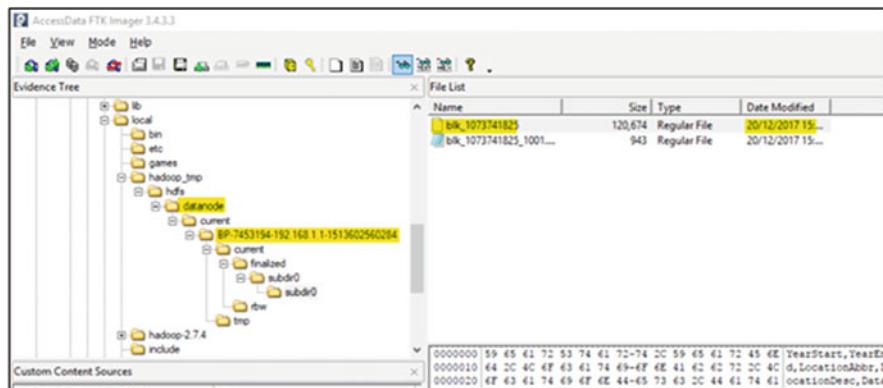


Fig. 11 The DataNode file structure with the block files present

During the PUT process, volatile memory was assessed while the put command was undertaken in HDFS. Many processes were running before, during and after the process. However, most of these processes were discredited forensically as they were processes that did not relate to HDFS itself. The only process that showed any relevant activity during the PUT action was that of PID 4644 which was labelled as a java process. Unfortunately, this did not supply any additional information upon further analysis of the PID itself, but the user was listed showing that criminal1 had been performing actions malicious or not (see Fig. 12).

```
top - 05:16:00 up 12 min, 1 user, load average: 0.27, 0.57, 0.51
Tasks: 250 total, 1 running, 255 sleeping, 0 stopped, 0 zombie
%Cpu(s): 16.1 us, 4.4 sy, 0.0 ni, 79.4 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st
KiB Mem : 4025436 total, 1594292 free, 1340128 used, 1091016 buff/cache
KiB Swap: 2094076 total, 2094076 free, 0 used. 2364348 avail Mem

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  4644 criminal+  20   0 2149428  78576 22380 S 65.4  2.0  0:01.97 java
 1010 root       20   0 478636 102392 34764 S  5.3  2.5  0:22.45 Xorg
1630 criminal+  20   0 1259656 98116 65064 S  4.3  2.4  0:35.82 compiz
2214 criminal+  20   0 669324 39148 28264 S  4.0  1.0  0:09.04 gnome-terminal-
1441 criminal+  20   0 278472  6448 5212 S  3.0  0.2  0:03.89 ibus-daemon
1474 criminal+  20   0 484584 29832 24848 S  1.3  0.7  0:01.15 ibus-ui-gtk3
1502 criminal+  20   0 195460  5408 5060 S  1.0  0.1  0:01.17 ibus-engine-sim
 2806 criminal+  20   0 2949208 274164 23148 S  0.7  6.8  0:23.86 java
```

Fig. 12 Output of Linux top command following real time analysis of memory processes

Following the active memory analysis, the raw data within the memory file was analysed with the goal of retrieving any additional information. No specific information regarding the data blocks or data itself was found. However, the recent bash commands performed in the terminal of the NameNode were stored showing that criminal1 had performed a PUT command with the file name of the dataset (Fig. 13).

Fig. 13 Raw volatile memory file analysed in HxD

As a means of validating the information above, the Hadoop log files were analysed to determine whether the changes in the HDFS had been logged. As previously found while analysing the file structure of the NameNode using FTK Imager, the log files were in the /usr/local/Hadoop-2.7.4/logs (see Fig. 14). The sizeable log file of the “Hadoop-criminal1-namenode-masternamenode.log” was of particular interest, as this file seemed to log the majority of actions performed in Hadoop.

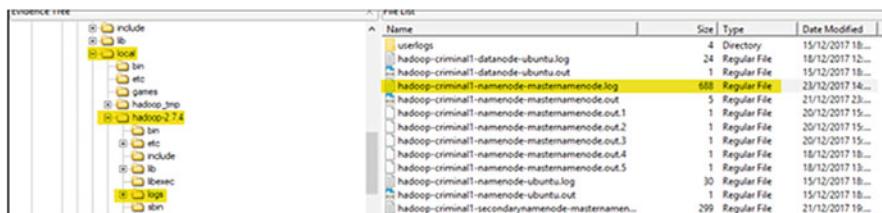


Fig. 14 NameNode VDMK file within FTK Imager showing the location of log files

Upon obtaining this log file, string searches were performed using the data retrieved above, i.e. the block ID, the name of the file and additional information that may be logged within the log file that coincided with the timestamps of the insertion of the dataset. The log file was preferably analysed within a Linux environment as the *grep* command could be used to retrieve the information referencing the details above. Following the use of the command “*grep* “Chronic” *Hadoop-criminal1-namenode-masternamenode.log*”, multiple entries were found with regards to the allocation of blocks associated with the dataset and the times that these operations occurred at. Another key piece of information that occurs is the DFSClient code that coincides with previous information about this operation (Fig. 15).

```
[root@hadoop-crime1 ~]# cat /var/log/hadoop-crime1-namenode-masternamenode.log | grep 'PUT'
18/07/20 09:45:26 INFO org.apache.hadoop.hdfs.StateChange: BLOCK_VILLAGERS_VIA_187374085_00010000000000000000xx-1 truncatedBlockCount=1, primaryNodeIndex=1, replication=1
18/07/20 09:45:26 INFO org.apache.hadoop.hdfs.StateChange: BLOCK_VILLAGERS_VIA_187374085_00010000000000000000xx-2 truncatedBlockCount=1, primaryNodeIndex=1, replication=1
18/07/20 09:45:26 INFO org.apache.hadoop.hdfs.StateChange: BLOCK_VILLAGERS_VIA_187374085_00010000000000000000xx-3 truncatedBlockCount=1, primaryNodeIndex=1, replication=1
18/07/20 09:45:26 INFO org.apache.hadoop.hdfs.StateChange: DIF_COMPLETEFILE: /U.S._Chronic Disease Indicators_CDI_.csv_COPYTIME_is closed by DFSClient_NONMAPREDUCE_777645396
18/07/20 09:45:26 INFO org.apache.hadoop.hdfs.StateChange: FOCK started by crime1nn1 with SPLITIFY From /192.168.1.1 For path /U.S._Chronic Disease Indicators_.csv
18/07/20 09:45:26 INFO org.apache.hadoop.hdfs.StateChange: FOCK started by crime1nn1 with SPLITIFY From /192.168.1.1 For path /U.S._Chronic Disease Indicators_.csv
18/07/20 09:45:26 INFO org.apache.hadoop.hdfs.StateChange: FOCK started by crime1nn1 with SPLITIFY From /192.168.1.1 For path /U.S._Chronic Disease Indicators_.csv
```

Fig. 15 grep output of the Hadoop-crime1-namenode-masternamenode.log file following the PUT command

5.2 Forensic Analysis of HDFS GET

Proceeding the use of the PUT command within HDFS, data stored within HDFS can retrieve to the local file system on the NameNode using the GET command (*hdfs does -get*). As with the PUT command, both the Hadoop tools were used to analyse the current FSimages and edits that had occurred during the GET command on the snapshots. Again, using the Offline Image Viewer command “`hdfs oev -i /usr/local/hadoop_tmp/hdfs/namenode/fsimage_0000000000000000xx-o /Desktop/getoivfs.xml`” on the current FSimage, a preview of the current changes was presented. The image viewer output in this scenario showed an unusual absence of information regarding the retrieval of the information from HDFS. However a notable piece of information was the change in the access time associated with the file which was listed in the image output (see Fig. 16).

```
<inode>
<id>16386</id>
<type>FILE</type>
<name>U.S._Chronic Disease_Indicators_CDI_.csv</name>
<replication>2</replication>
<mtime>1513782878571</mtime>
<atime>1513899234122</atime>
<preferredBlockSize>134217728</preferredBlockSize>
<permission>crime1nn1:supergroup:rW-r--</permission>
```

Fig. 16 Output of the Offline Image Viewer showing a difference in access time

The edited viewer was used as a means of determining whether more information was present regarding the GET command. The command “`hdfs oev -i /usr/local/hadoop_tmp/hdfs/namenode/edits_0000000000000000xx-0000000000000000xx -o /Desktop/getoevfs.txt -p stats`” was used on the current edits log to give a summary of operations that had occurred since the last image. Unlike the PUT command, however, the GET command did not list any specific operation codes relating to the operation of GET. The operation codes that were listed after this command were in fact only the OP_START_LOG_SEGMENT and the OP_END_LOG_SEGMENT, which unfortunately occur for every operation within HDFS (see Fig. 17).

To investigate further, the verbosity of the command was added to break down the operations further and see if additional information could be gained from the edits log (“`hdfs oev -I edits_0000000000000000xx-0000000000000000xx -o`

OP_CANCEL_DELEGATION_TOKEN	(20): null
OP_UPDATE_MASTER_KEY	(21): null
OP_REASSIGNLEASE	(22): null
OP_END_LOG_SEGMENT	(23): 1
OP_START_LOG_SEGMENT	(24): 1
OP_UPDATE_BLOCKS	(25): null

Fig. 17 Overview of the Offline Edits Viewer

/Desktop/getoeyfs2.txt -v”). Although the detailed output did present more details regarding the operations themselves, the only notably forensic information on the edits log was the filename of the file that had been accessed by the changed access time and unchanged modification time (see Fig. 18).

```
<RECORD>
<OPCODE>OP_START_LOG_SEGMENT</OPCODE>
<DATA>
<TXID>44</TXID>
</DATA>
</RECORD>
<RECORD>
<OPCODE>OP_TIMES</OPCODE>
<DATA>
<TXID>45</TXID>
<LENGTH>0</LENGTH>
<PATH>/U.S._Chronic_Disease_Indicators__CDI_.csv</PATH>
<MTIME>-1</MTIME>
<ATIME>1513899234122</ATIME>
</DATA>
</RECORD>
<RECORD>
<OPCODE>OP_END_LOG_SEGMENT</OPCODE>
```

Fig. 18 Edits log showing the change in access time for the dataset

As no information of great value could be yielded from the FSimage and Edit logs attention was turned to the log files within the name of Hadoop itself. Upon actively performing the GET command to see if the logs were updated on either the NameNode or the DataNodes, using the grep command with the filename as an argument, it was found that the file was copied with a DFS Client ID associated with the copy (see Fig. 18). In addition to the analysis of the log files, a static approach of file system reconnaissance was used to recover information regarding the downloaded file. Using the Linux command “sudo find. -name “*.csv” -type f -time 0 > modified.txt” from the root directory listed any files that had been modified (creation time is deemed the same as modified time) in the last 24 h. Upon running this command, it was found that the download.csv file (including its directory) that had been created as a result of the Get command appeared in the output of the find command. To verify the creation time of this file the Linux command “stat download.csv” was used as a means to determining the creation time. However,

the birth field was absent in the stat output for this file (see Fig. 19) which may be a result of the absence of APIs within the Kernel. The ext4 filesystem within Linux does, in fact, maintain the file system creation time. After finding the Inode ID from the stat command, using the command “`sudo debugfs -R ‘stat <1052122>’ /dev/sda1`”, the creation time was revealed to be shortly after the epoch located in the edits log (see Fig. 20).

```
criminal@masternamenode:~$ stat ~/Desktop/download.csv
  File: '/home/criminal/Desktop/download.csv'
  Size: 123570042    Blocks: 241360   IO Block: 4096   regular file
Device: 801h/2049d  Inode: 1052122      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/criminal)  Gid: ( 1000/criminal)
Access: 2017-12-29 23:34:25.196688048 -0800
Modify: 2017-12-29 23:34:32.806988241 -0800
Change: 2017-12-29 23:34:33.079076476 -0800
 Birth: -
criminal@masternamenode:/$
```

Fig. 19 Terminal output for the statistics of the newly created download.csv

```
Inode: 1052122  Type: regular  Mode: 0644  Flags: 0x800000
Generation: 3098657449  Version: 0x00000000:00000001
User: 1000  Group: 1000  Size: 123570042
File ACL: 0  Directory ACL: 0
Links: 1  Blockcount: 241360
Fragment: Address: 0  Number: 0  Size: 0
  ctime: 0x5a474189:12da1430 -- Fri Dec 29 23:34:33 2017
  atime: 0x5a474181:2ee4e2c0 -- Fri Dec 29 23:34:25 2017
  mtime: 0x5a474188:c066a744 -- Fri Dec 29 23:34:32 2017
  crtime: 0x5a474181:2ee4e2c0 -- Fri Dec 29 23:34:25 2017
Size of extra inode fields: 32
```

Fig. 20 Terminal output from the debugfs command showing the creation time of the file relative to the Inode

After analysing the network file proceeding this interval of action, much like the PUT command, a DFSClient_NONMAPREDUCE entry was found, occurring between NameNode (192.168.1.1) and the DataNode (192.168.1.4), followed by an influx of data relating to the dataset itself (see Fig. 21). Despite this DFSClient ID being present, no other instance of it was found in the log files, FSimage or edit logs. Upon inspection of the VMEM file, this was also the case.

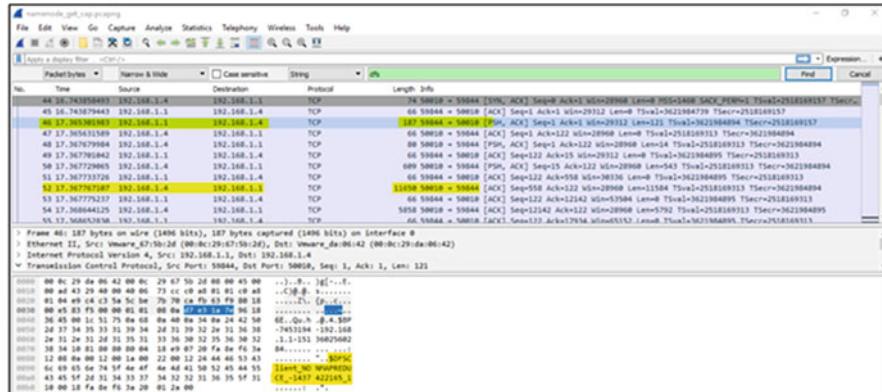


Fig. 21 Presence of the DFSClient ID in the network capture followed by data transfer

5.3 Forensic Analysis of HDFS APPEND

As HDFS does not allow the modification of files once they are populated within the filesystem. The use of the APPEND (hdfs dfs -append) command is therefore needed to add additional information to a file already within HDFS, lest the user must delete the entire file and repopulate the environment with the modified version of the file. In this instance, a sample MP4 file was appended to the original dataset to simulate a criminal trying to hide information within another HDFS based file.

```

<inode>
  <id>16386</id>
  <type>FILE</type>
  <name>U.S. Chronic Disease_Indicators__CDI_.csv</name>
  <replication>2</replication>
  <mtime>1513951731976</mtime>
  <atime>1513782858757</atime>
  <preferredBlockSize>134217728</preferredBlockSize>
  <permission>criminally:supergroup:rw-r--r--</permission>
  - <blocks>
    - <block>
      <id>1073741825</id>
      <genstamp>1002</genstamp>
      <numBytes>134217728</numBytes>
    </block>
    - <block>
      <id>1073741826</id>
      <genstamp>1003</genstamp>
      <numBytes>134217728</numBytes>
    </block>
    - <block>
      <id>1073741827</id>
      <genstamp>1004</genstamp>
      <numBytes>134217728</numBytes>
    </block>
  - <block>

```

Fig. 22 Output of the Offline Image Viewer proceeding an append operation

Again, drawing on the detailed logging capabilities of the FSimage and edit logs, the HDFS OIV and OEV commands were utilised to analyse the logs following the use of the APPEND command.

The Offline Image Editor harboured some interesting information regarding the dataset. Additional blocks were allocated to the filename of the dataset with each block being allocated a subsequent block ID following the block ID of the initial block allocation ID. Naturally, the accessed time was changed (see Fig. 22). The results of the Offline Edits Viewer overview output also gave interesting information proceeding the APPEND. Naturally, an OP_APPEND operation was logged in the edits overview with the multiple block ID allocations and block additions being cited in the overview (see Fig. 23).

OP_DISALLOW_SNAPSHOT	(30): null
OP_SET_GENSTAMP_V2	(31): 5
OP_ALLOCATE_BLOCK_ID	(32): 4
OP_ADD_BLOCK	(33): 4
OP_APPEND	(47): 1
OP_SET_QUOTA_BY_STORAGETYPE	(48): null

Fig. 23 Output of the Offline Image Viewer proceeding an append operation

A large quantity of activity was logged in the detailed edits log following the APPEND operation. The OP_APPEND operation holds a large amount of key information regarding the file being appended, such as the name of the file and whether a new block is needed to accommodate for the new appended information. Assuming this is the case the first block has enough space to store some of the

```
<OPCODE>OP_APPEND</OPCODE>
- <DATA>
  <TXID>45</TXID>
  <PATH>/U.S._Chronic_Disease_Indicators__CDI_.csv</PATH>
  <CLIENT_NAME>DFSClient_NONMAPREDUCE_-856343220_1</CLIENT_NAME>
  <CLIENT_MACHINE>192.168.1.1</CLIENT_MACHINE>
  <NEWBLOCK>false</NEWBLOCK>
  <RPC_CLIENTID>82b55a99-ce7e-47a0-8ddd-10ecf3282eef</RPC_CLIENTID>
  <RPC_CALLID>1</RPC_CALLID>
</DATA>
</RECORD>
+ <RECORD></RECORD>
- <RECORD>
  <OPCODE>OP_UPDATE_BLOCKS</OPCODE>
  - <DATA>
    <TXID>47</TXID>
    <PATH>/U.S._Chronic_Disease_Indicators__CDI_.csv</PATH>
    - <BLOCK>
      <BLOCK_ID>1073741825</BLOCK_ID>
      <NUM_BYTES>123570042</NUM_BYTES>
      <GENSTAMP>1002</GENSTAMP>
    </BLOCK>
    <RPC_CLIENTID>82b55a99-ce7e-47a0-8ddd-10ecf3282eef</RPC_CLIENTID>
    <RPC_CALLID>4</RPC_CALLID>
  </DATA>
  </RECORD>
- <RECORD>
  <OPCODE>OP_ALLOCATE_BLOCK_ID</OPCODE>
  - <DATA>
    <TXID>48</TXID>
    <BLOCK_ID>1073741826</BLOCK_ID>
  </DATA>
```

Fig. 24 Block allocations and updates contained within the detailed edits log

newly added information, this block could be utilised once more to retrieve the newly appended information for analysis. As seen in Fig. 24, following the append function, once HDFS realises that the first block is filled to the specified block size of HDFS new blocks are allocated new block IDs to accommodate for the additional information.

The DFSClient ID was once again used as a search string parameter in Wireshark to determine the transfer of data within HDFS to the newly appended blocks. The DFSClient ID matched that of the Client ID in the network traffic transfer and was shortly followed by the transfer of data as the file was appended (see Fig. 25).

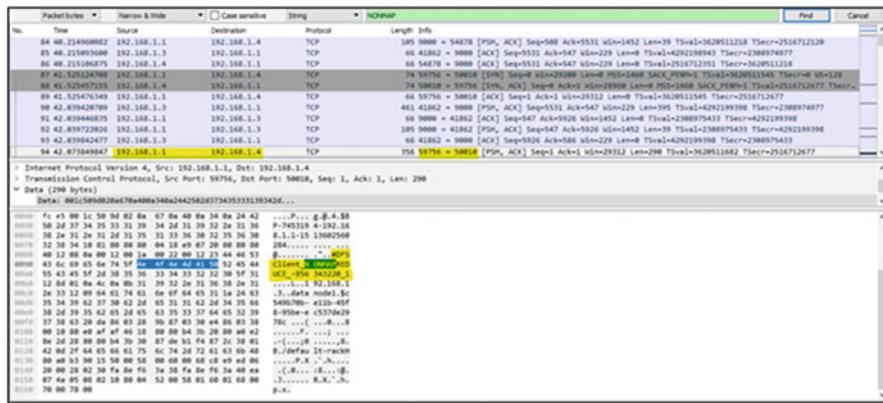


Fig. 25 DFSClient ID present followed by data transfer of APPEND operation

The new designated block size change is logged in the hadoop-criminal1-namenode-masternamenode.log log file referencing the block ID and additional information such as the new length of the file and the nodes that this file is stored on (see Fig. 26).

```
hadoop@namenode:/usr/local/hadoop-2.7.4/logs$ grep '056343220' hadoop-criminal1-namenode-masternamenode.log
2017-12-22 06:44:17.10.733 INFO org.apache.hadoop.hdfs.namenode.FDNameSystem$DataStreamer[putBlock_1073741826_0001]: newGID=1002, newLength=123570842,
newNodes=[192.168.1.3:50010, 192.168.1.3:50010], clientDFSClient StateChange: DIR* completefile: /V.S_Chronic_Disease_Indicators_CDI..csv is closed by DFSClient_N
ONMAPREDUCE 056343220 1
hadoop@namenode:/usr/local/hadoop-2.7.4/logs$
```

Fig. 26 grep output of the NameNode log file, using DFSClient ID as a search parameter

An analysis of the block file associated with the appended dataset using HXD Editor was undertaken as a means of determining how the file was appended. As seen in Fig. 27 the file signature of the MP4 format was still recoverable in the appended CSV file. That illustrates that despite being appended as a different file format, HDFS did not convert the file to a respective format type. It simply adds the MP4 file as additional text data to the dataset, therefore making the MP4 file recoverable with file scraping methods.

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
075D8770	4C 2C 4F 56 52 2C 2C 2C 2C 0A 00 00 00 18 66 74
075D8780	79 70 6D 70 34 32 00 00 00 69 73 6F 6D 6D 70
075D8790	34 32 00 16 39 6A 6D 6F 76 00 00 00 6C 6D 76
075D87A0	68 64 00 00 00 00 D4 E4 3A 18 D4 E4 3A 18 00 01
075D87B0	5F 90 0F 91 CF 58 00 01 00 00 01 00 00 00 00 00
075D87C0	00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00

Fig. 27 MP4 file signature located in the dataset file block proceeding the append

top - 05:16:00 up 12 min, 1 user, load average: 0.27, 0.57, 0.51											
Tasks: 256 total, 1 running, 255 sleeping, 0 stopped, 0 zombie											
%Cpu(s): 16.1 us, 4.4 sy, 0.0 ni, 79.4 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st											
KiB Mem : 4025436 total, 1594292 free, 1340128 used, 1091016 buff/cache											
KiB Swap: 2094076 total, 2094076 free, 0 used. 2364348 avail Mem											
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4644	crimina+	20	0	2149428	78576	22380	S	65.4	2.0	0:01.97	java
1010	root	20	0	478636	102392	34764	S	5.3	2.5	0:22.45	Xorg
1630	crimina+	20	0	1259656	98116	65064	S	4.3	2.4	0:35.82	compiz
2214	crimina+	20	0	669324	39148	28264	S	4.0	1.0	0:09.04	gnome-terminal
1441	crimina+	20	0	278472	6448	5212	S	3.0	0.2	0:03.89	ibus-daemon
1474	crimina+	20	0	484584	29832	24848	S	1.3	0.7	0:01.15	ibus-ui-gtk3
1502	crimina+	20	0	195460	5488	5060	S	1.0	0.1	0:01.17	ibus-engine-sim
2806	crimina+	20	0	2949208	274164	23148	S	0.7	6.8	0:23.86	java

Fig. 28 Output of the “top” command in Linux during the APPEND process

The volatile memory of the NameNode was also assessed during and following the append function to determine if any forensic artefacts were present in memory. During the active assessment, it was found that many different processes were running during the append actions within HDFS. These processes in addition to the *ibus-daemon* and *ibus-ui-gtk3*, however, it is not clear if these processes were associated with the actions in HDFS. During the APPEND command, the PID 4644 (java) rose to a high CPU usage until the command was finished. This java execution is to be expected under the consideration that HDFS has programmed in Java. Unfortunately, it does not give any granular details regarding the action itself but only denotes the criminal user at the time (see Fig. 28).

Static analysis of the memory during the time of the append function was undertaken to determine if any forensic artefacts were encapsulated with the raw memory file itself. Following on from the discovery of the DFSClient ID found within the log files (above), the Client ID was also listed within the volatile memory itself. Additional information was found within strings relating to a data streamer that pointed towards the file name of the dataset followed by the blocks that the dataset was hosted upon. Bash commands were also stored within the raw file, incriminating the criminal1 user by outlining the commands that had been facilitated by that user in HDFS (see Fig. 29).

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	
005B1100	41 00 00 20 7F 00 00 00 44 00 61 00 74 00 61 00	A.Data.
005B1110	53 00 74 00 72 00 65 00 61 00 6D 00 65 00 72 00	S.t.r.e.a.m.e.r.
005B1120	20 00 66 00 6F 00 72 00 20 00 66 00 69 00 6C 00	.f.o.r. .f.i.l.
005B1130	65 00 20 00 2F 00 55 00 2E 00 53 00 2E 00 5F 00	e. ./U.S....
005B1140	43 00 68 00 72 00 6F 00 6E 00 69 00 63 00 5F 00	C.h.r.o.n.i.c._
005B1150	44 00 69 00 73 00 65 00 61 00 73 00 65 00 5F 00	D.i.s.e.a.s.e..
005B1160	49 00 6E 00 64 00 69 00 63 00 61 00 74 00 6F 00	I.n.d.i.c.a.t.o.
005B1170	72 00 73 00 5F 00 5F 00 43 00 44 00 49 00 5F 00	r.s._..C.D.I. ..
005B1180	2E 00 63 00 73 00 76 00 20 00 62 00 6C 00 6F 00	..c.s.v. .b.l.o.
005B1190	63 00 6B 00 20 00 42 00 50 00 2D 00 37 00 34 00	c.k. .B.P.-.7.4.
005B11A0	35 00 33 00 31 00 39 00 34 00 2D 00 31 00 39 00	5.3.1.9.4.-.1.9.
005B11B0	32 00 2E 00 31 00 36 00 38 00 2E 00 31 00 2E 00	2...1.6.8...1...
005B11C0	31 00 2D 00 31 00 35 00 31 00 33 00 36 00 30 00	1.-.1.5.1.3.6.0.
005B11D0	32 00 35 00 36 00 30 00 32 00 38 00 34 00 3A 00	2.5.6.0.2.8.4.:
005B11E0	62 00 6C 00 6B 00 5F 00 31 00 30 00 37 00 33 00	b.l.k._..1.0.7.3.
005B11F0	37 00 34 00 31 00 38 00 32 00 39 00 5F 00 31 00	7.4.1.8.2.9._.1.
005B1200	30 00 30 00 38 00 00 00 C3 06 FB EA 00 00 00 00	0.0.8...A.üe....

Fig. 29 Data stream information regarding the dataset and the blocks associated with it

5.4 Forensic Analysis of HDFS REMOVE

Deleting information from an environment is often an area of focus for forensic investigations. In HDFS the command hdfs dfs -rm is utilised to remove data files from the filesystem. The OEV and the OIV were once again used on the FSImage and edit logs to determine the operation codes that were associated with the removal of data from HDFS. As seen in Fig. 30, the OIV output contained information regarding a new directory name.Trash with a modification (assumed to be creation time) time that coincided with the deletion of the data within HDFS.

```
-<inode>
<id>16389</id>
<type>DIRECTORY</type>
<name>.Trash</name>
<mtime>1513956422419</mtime>
<permission>criminal:supergroup:rwx-----</permission>
<nsquota>-1</nsquota>
<dsquota>-1</dsquota>
</inode>
```

Fig. 30 Offline Image Viewer output showing the creation of the .Trash directory

The outputs within the Edit logs were analysed with the assumption that a delete operation would be found within the overview and detailed outputs. This was not found to be the case, however, as the OP_DELETE counter showed that not had been performed. The only new operational instances were that of the OP_MKDIR and OP_RENAME operations (in Fig. 31).

VERSION	:	-63
OP_ADD	(0):	null
OP_RENAME_OLD	(1):	null
OP_DELETE	(2):	null
OP_MKDIR	(3):	4
OP_RENAME	(15):	1
OP_CONCAT_DELETE	(16):	null

Fig. 31 Overview of Offline Edits Viewer output showing the absence of a delete operation

```
<OPCODE>OP_RENAME</OPCODE>
-<DATA>
  <TXID>67</TXID>
  <LENGTH>0</LENGTH>
  <SRC>/U.S_Chronic_Disease_Indicators_CDI_.csv</SRC>
-<DST>
  /user/criminal1/.Trash/Current/U.S_Chronic_Disease_Indicators_CDI_.csv
</DST>
<TIMESTAMP>1514038442456</TIMESTAMP>
<OPTIONS>TO_TRASH</OPTIONS>
<RPC_CLIENTID>e8c0123b-f54f-4daa-bda6-90760d9f8485</RPC_CLIENTID>
<RPC_CALLID>6</RPC_CALLID>
```

Fig. 32 Offline edits viewer output showing the rename of the dataset with the addition of the TO_TRASH option

```
<OPCODE>OP_MKDIR</OPCODE>
-<DATA>
  <TXID>66</TXID>
  <LENGTH>0</LENGTH>
  <INODEID>16390</INODEID>
  <PATH>/user/criminal1/.Trash/Current</PATH>
<TIMESTAMP>1514038439778</TIMESTAMP>
```

Fig. 33 Offline edits viewer showing the creation of the Trash directory

This absence of a deletion operation was intriguing. Thus the increase in verbosity for the Offline Edits Viewer was required. Both Figs. 32 and 33 show the output of the increased detail offered by the Offline Edits Viewer, with the operation codes for both the directory creation and the renaming of the dataset to a filename that encompassed the newly created .Trash directory. The OPTIONS parameter in the OP_RENAME operation is marked TO_TRASH, thus meaning that when a file is deleted using the *rm* parameter in HDFS. It must therefore only be renamed, subsequently moving it to the trash directory temporarily. As HDFS is a logical and stateless file system, the actual transit of data from one directory to the next does not occur, but the renaming of data moves the data accordingly.

To validate the creation of this new directory and to determine if the files could still be recovered, HDFS was explored proceeding the deletion of the dataset. The deleted dataset was found within the newly created *Trash* directory and could be retrieved with the GET command in HDFS (see Fig. 34).

```
criminal1@masternamenode:/usr/local/hadoop/etc/hadoop
crimina...@masternamenode:/usr/local/hadoop/etc/hadoop$ hdfs dfs -ls /user/criminal1/.Trash/Current
17/12/22 06:46:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
...
2 criminal supergroup 564882092 2017-12-22 06:08 /user/criminal1/.Trash/Current/U.S._Chronic_Disease_Indicators__CDI_.csv
crimina...@masternamenode:/usr/local/hadoop/etc/hadoop$
```

Fig. 34 Newly created .Trash directory within HDFS containing the deleted dataset

When analysing the network traffic in Wireshark during the deletion, no traffic of note was recorded. This concluded that only the NameNode was aware of the deletion and that with the DataNodes unaware of the deletion this meant that the data saved on the respective blocks were stored within the same place as before and was only logically moved to trash in the logical file structure of HDFS.

During the removal of data within HDFS the processes in volatile memory were analysed with a means of determining any artefacts within the memory that may demonstrate that a user is performing a specific action in HDFS. Unfortunately, like the previous two memory analyses, no processes of note (disregarding the java process) were found. The static memory analysis in HxD only recovered the bash commands relating to the removal of the data but did show the trash interval time, thus showing that the data was still recoverable (Fig. 35).

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	
01DA4F10	6F 64 65 1B 5B 30 30 6D 3A 1B 5B 30 31 3B 33 34	ode.[00m:.[01:34
01DA4F20	6D 7E 1B 5B 30 30 6D 24 20 68 64 66 73 20 64 66	m-.[00mG hdfs di
01DA4F30	73 20 2D 72 6D 72 20 2F 08 1B 5B 4B 08 1B 5B 4B	[] -rmr /..[K..[K
01DA4F40	08 1B 5B 4B 20 2F 55 2E 53 2E 5F 43 68 72 6F 6E	..[K /U.S._Chron
01DA4F50	69 63 5F 44 69 73 65 61 73 65 5F 49 6E 64 69 63	ic_Disease_Indic
01DA4F60	61 74 6F 72 73 5F 5F 43 44 49 5F 2E 20 0D 63 73	ators_CDI_.cs
01DA4F70	76 0D 0A 31 37 2F 31 32 2F 33 30 20 30 38 3A 32	y..17/12/30 08:2
01DA4F80	38 3A 31 31 20 57 41 52 4E 20 75 74 69 6C 2E 4E	8:11 WARN util.N
01DA4F90	61 74 69 76 65 43 6F 64 65 4C 6F 61 64 65 72 3A	ativeCodeLoader:
01DA4FA0	20 55 6E 61 62 6C 65 20 74 6F 20 6C 6F 61 64 20	Unable to load
01DA4FB0	6E 61 74 69 76 65 2D 68 61 64 6F 6F 70 20 6C 69	native-hadoop li
01DA4FC0	62 72 61 72 79 20 66 6F 72 20 79 6F 75 72 20 70	brary for your p
01DA4FD0	6C 61 74 66 6F 72 6D 2E 2E 20 75 73 69 6E 67	atform... using
01DA4FE0	20 62 75 69 6C 74 69 6E 2D 6A 61 76 61 20 63 6C	builtin-java cl
01DA4FF0	61 73 73 65 73 20 77 68 65 72 65 20 61 70 70 6C	asses where appl
01DA5000	69 63 61 62 6C 65 0D 0A 31 37 2F 31 32 2F 33 30	icable..17/12/30
01DA5010	20 30 38 3A 32 38 3A 31 32 20 49 4E 46 4F 20 66	08:28:12 INFO f
01DA5020	73 2E 54 72 61 73 68 50 6F 6C 69 63 79 44 65 66	s.TrashPolicyDef
01DA5030	61 75 6C 74 3A 20 4E 61 6D 65 6E 6F 64 65 20 74	ault: Namenode t
01DA5040	72 61 73 68 20 63 6F 6E 66 69 67 75 72 61 74 69	rash configurati
01DA5050	6F 6E 3A 20 44 65 6C 65 74 69 6F 6E 20 69 00 00	on: Deletion i..

Fig. 35 RAW memory file showing the trash policy default time

5.5 Forensic Analysis of a Basic MapReduce Task

One of the fundamental actions performed in Hadoop is that of data reduction and collation using HDFS and MapReduce. The Apache Hadoop MapReduce tutorial Wordcount v1.0 was used to simulate the use of MapReduce within the environment [46]. Following the MapReduce tutorial, the word selected for the wordcount was “Chronic” and was tested against the original dataset inserted into Hadoop. Following the MapReduce operation, the Hadoop infrastructure was assessed to try and unearth any artefacts that had been left as a result of the MapReduce execution within the cluster. It was found that the NameNode log file “*yarn-criminal1-resourcemanager-masternamenode.log*” was updated during the time of the MapReduce operation. Figure 36 shows the user *criminal1* successfully submitting the job to HDFS to carry out the MapReduce job, found by using grep with the expression *criminal1*.

```
2017-12-22 10:54:13.112 INFO org.apache.hadoop.yarn.server.resourcemanager.ClientRMService Application with id 1 submitted by user criminal1
2017-12-22 10:54:13.117 INFO org.apache.hadoop.yarn.server.resourcemanager.RMAuditLogger USER-PRINCIPAL
Application=Request=ClientRMService RESULT=SUCCESS APPID=application_1513976150448_0001
2017-12-22 10:54:13.124 INFO org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.ParentQueue: Application added - appid: application_151
2017-12-22 10:54:13.125 INFO org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler: Accepted application application_151
2017-12-22 10:54:13.126 INFO org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.LeafQueue: Application application_1513976150448_0001 from user: criminal1 activated in queue: default
2017-12-22 10:54:13.250 INFO org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.LeafQueue: Application application_1513976150448_0001 from user: criminal1 activated in queue: default
```

Fig. 36 grep output of the *yarn-criminal1-resourcemanager-masternamenode.log* file showing successful submission by the *criminal1* user

The network capture of the MapReduce again showed DFSClient IDs, as shown in Fig. 34 above. The DFSClient ID was used as a *grep* search parameter on the log files as a means of determining if the MapReduce task was logged in any additional places. It was found that information regarding the MapReduce task was found in the generic NameNode log file “*hadoop-criminal1-namenode-masternamenode.log*”. This log file revealed that each job created by a user in MapReduce had a temporary directory made for itself in HDFS, and could, therefore, be retrieved in the HDFS

```
crim1@masternamenode:~/var/local/hadoop-2.7.4/logs$ grep "DFSClient|MAPREDUCE|job_1513976150448_0001" hadoop-criminal1-namenode-masternamenode.log
2017-12-22 10:54:13.118 INFO org.apache.hadoop.hdfs.StateChange: DFSC completedFile: /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513980535224_0001/job.xml is closed by DFSClient_N
2017-12-22 10:54:13.251 INFO org.apache.hadoop.hdfs.StateChange: DFSC completedFile: /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513980535224_0001/job-split is closed by DFSClient_N
INFODFSC completedFile: /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513980535224_0001/job-splitmetafile is closed by DFSClent_N
2017-12-22 10:54:13.379 INFO org.apache.hadoop.hdfs.StateChange: DFSC completedFile: /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513980535224_0001/job.xml is closed by DFSClent_N
2017-12-22 10:54:13.384 INFO org.apache.hadoop.hdfs.StateChange: DFSC completedFile: /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513980535224_0001/job-splitmetafile is closed by DFSClent_N
```

Fig. 37 Output of the *grep* command against the *masternamenode.log* file

```
crim1@masternamenode:~/Desktop$ hdfs dfs -ls /tmp/hadoop-yarn/staging/criminal1/_staging
17/12/22 11:19:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-Java classes
Found 9 items
drwx----- . criminal1 supergroup 0 2017-12-22 11:06 /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513885535224_0002
drwx----- . criminal1 supergroup 0 2017-12-22 11:58 /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513885535224_0003
drwx----- . criminal1 supergroup 0 2017-12-22 12:04 /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513973014039_0001
drwx----- . criminal1 supergroup 0 2017-12-22 12:23 /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513974672553_0002
drwx----- . criminal1 supergroup 0 2017-12-22 12:23 /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513974672553_0003
drwx----- . criminal1 supergroup 0 2017-12-22 12:49 /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513975780825_0001
drwx----- . criminal1 supergroup 0 2017-12-22 12:56 /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513976150448_0001
drwx----- . criminal1 supergroup _ 0 2017-12-22 13:12 /tmp/hadoop-yarn/staging/criminal1/_staging/job_1513976150448_0002
```

Fig. 38 Temporary directories created for each Hadoop job within HDFS

file structure, the directory is contained within the `/tmp/` folder of HDFS. Each of these job directories contained additional information files regarding the jobs (see Figs. 37 and 38).

Using the HDFS GET command the directory contents of `/tmp/hadoop-yarn/staging/criminal1/.staging/job_1513885535224_0002` was retrieved for analysis. The directory contained an XML as seen below in Fig. 39 and the JAR file containing the compiled java code associated with the MapReduce job above. The XML file contained configuration information for each of the settings respective of the job, one of the parameters within this file was the MapReduce job username, marked in this scenario as *criminal1*.

```
<name>mapreduce.job.user.name</name>
<value>criminal1</value>
<source>programmatically</source>
</property>
-<property>
- <name>
```

Fig. 39 Job username of the MapReduce job contained in the job.xml file of the job directory

The network capture of the Master NameNode confirmed the DFS Client ID that was encountered during the analysis of the logs (see Fig. 40).

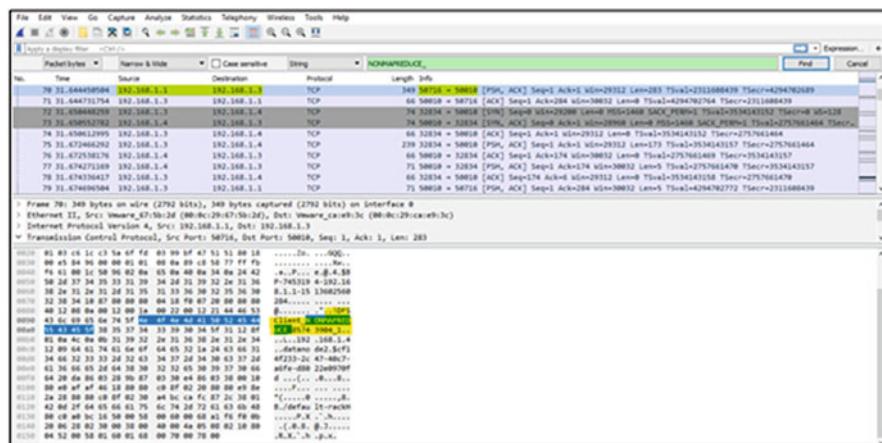


Fig. 40 Network capture during the MapReduce Task

Auspiciously for forensic examiners, all tasks relating to MapReduce are accessible via a web interface on the NameNode, configured at port 8088. Users can see all tasks including current tasks and completed tasks, as well as the task owner and the times associated with the start and end of each task. Figure 41 illustrates the web interface.

The screenshot shows the Hadoop web interface with the title "ACCEPTED Applications". On the left, there's a sidebar with "Cluster Metrics" and "Scheduler Metrics" sections, and a main area for "Applications". The "Applications" section has columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, Tracking UI, and Blacklisted Nodes. Two entries are listed:

ID	User	Name	Application Type	Queue	Start Time	Finish Time	State	Final Status	Progress	Tracking UI	Blacklisted Nodes
application_1513976150488_0002	criminal1	worksource	MAPREDUCE	default	13-12-28 08:00:00 2017	N/A	ACCEPTED	UNDEFINED		applicationMaster 0	
application_1513976150488_0003	criminal1	wordcount	MAPREDUCE	default	13-12-22 13:56:13 2017	N/A	ACCEPTED	UNDEFINED		applicationMaster 0	

At the bottom, it says "Showing 1 to 2 of 2 entries".

Fig. 41 Hadoop web interface showing the currently scheduled jobs

6 Conclusion and Future Work

HDFS is a large and complex infrastructure with a web of different configuration files, logs and systems that can be analysed to gain information on the actions occurring within. The purpose of this paper was to forensically explore the HDFS environment and uncover forensic artefacts that could aid in a forensic investigation. During this investigation, each key HDFS command was executed and monitored to understand which artefacts were produced following the execution of each command. The Linux file systems, volatile system memory and network files were all analysed at each interval of action.

The inbuilt HDFS commands for both the Offline Image Viewer and Offline Edits Viewer proved invaluable in forensic data retrieval during the investigation. Each tool provided the examiners with a range of information featured in each FSimage and edited log, which could be extrapolated to other areas of the investigation. The granularity of each of the tool outputs allowed for a comprehensive insight into the activities of the cluster during each action. Subsequently, it is our recommendation that effective forensic analysis of the HDFS environment can occur, checkpointing of the Master NameNode on the secondary NameNode must occur at a high frequency to effectively track the activities of HDFS.

In addition to the FSimages and edit logs provided by checkpointing, the comprehensive nature of the Hadoop log files themselves provided a plethora of information on any actions made with Hadoop. It is due to this and the collation of the previously mentioned FSimages that more information could be found. The generic `hadoop-criminal-namenode-masternamenode.log` file logged almost every action within the HDFS cluster, showing that this file has great importance to forensic investigators, and interpreting the output of this log file (and others) is vital when forensically analysing the Hadoop environment.

One of the key objectives of this investigation was to assess all aspects of the HDFS including processes within volatile memory. Unfortunately, the findings surrounding this area of the investigation did not reap any forensically significant information as the processes associated with HDFS is run as abstract processes of

Java, these processes were associated with the malicious user, but this information alone does not offer any insight into the actions being taken.

Clarification was also sought regarding the communication occurring between the Nodes and whether network traffic produced forensic artefacts that could prove useful in determining activity. Although some information was gained from the network traffic that corresponded with other findings, the traffic offered only a small amount of information that could be used to validate communications between the NameNode and DataNodes. It should also be noted that in some HDFS installations, communication between each of the nodes is often encrypted as a means of security and would, therefore, offer less information.

Assumptions were made with some of the actions that seemed logical, the removal of data being once instance in which one would assume some notation of deletion to occur, however, this produced results that did not coincide with the assumptions made. It is because of assumptions like this that forensically analysing environments like that of HDFS and other distributed file systems are required so that the forensically community can build a repertoire of knowledge with regards to these systems.

As this investigation focused predominately on HDFS, more works needs to be done to understand the entirety of the Hadoop ecosystem with special attention being paid to the types of artefacts that are generated as a result of MapReduce. The exploration of the Eclipse IDE with the MapReduce plugin may also prove valuable in gathering forensic insight into HDFS forensics. Additional research may be needed in HDFS environments that utilise Kerberos for authentication, as more forensic artefacts may be attributed to a Kerberos base cluster. Moreover, forensics investigation of evidence generated in HDFS through connecting it to emerging IoT and IIoT environments is another interesting future research [47, 48]. Analysing malicious programs are specifically written to infect HDFS platforms to exfiltrate data or steal private information within cyber-attack campaigns is another future work of this study [15]. Finally, as machine learning based forensics investigation techniques are proving their value in supporting forensics investigators and malware analyst activities it is important to build AI-based techniques to collect, preserve and analyse potential evidence from distributed file system platforms [49–51].

Acknowledgement We would like to thank the editor and anonymous reviewers for their constructive comments. The views and opinions expressed in this article are those of the authors and not the organisation with whom the authors are or have been associated with or supported by.

References

1. S. Tahir and W. Iqbal, “Big Data-An evolving concern for forensic investigators,” in *2015 1st International Conference on Anti-Cybercrime, ICACC 2015*, 2015.
2. W. Yang, G. Wang, K.-K. R. Choo, and S. Chen, “HEPart: A balanced hypergraph partitioning algorithm for big data applications,” *Futur. Gener. Comput. Syst.*, Jan. 2018.

3. W. A. Günther, M. H. Rezazade Mehrizi, M. Huysman, and F. Feldberg, “Debating big data: A literature review on realizing value from big data,” *J. Strateg. Inf. Syst.*, 2017.
4. T. H. Davenport and J. Dyche, “Big Data in Big Companies,” *Int. Inst. Anal.*, no. May, pp. 1–31, 2013.
5. B. Fang and P. Zhang, “Big data in finance,” in *Big Data Concepts, Theories, and Applications*, 2016, pp. 391–412.
6. S. Sharma, U. S. Tim, J. Wong, S. Gadia, and S. Sharma, “A Brief Review on Leading Big Data Models,” *Data Sci. J.*, vol. 13, no. December, pp. 138–157, 2014.
7. S. Yu and S. Guo, *Big Data Concepts, Theories, and Applications*, 1st ed. 20. Cham: Springer International Publishing, 2016.
8. X. Wu, X. Zhu, G. Q. Wu, and W. Ding, “Data mining with big data,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, 2014.
9. C. Vorapongkitipun and N. Nupairoj, “Improving performance of small-file accessing in Hadoop,” in *2014 11th Int. Joint Conf. on Computer Science and Software Engineering: “Human Factors in Computer Science and Software Engineering” - e-Science and High Performance Computing: eHPC, JCSSE 2014*, 2014, pp. 200–205.
10. Y. Y. Teing, A. Dehghantanha, and K. K. R. Choo, “CloudMe forensics: A case of big data forensic investigation,” *Concurrency Computation*, 2017.
11. X. Fu, Y. Gao, B. Luo, X. Du, and M. Guizani, “Security Threats to Hadoop: Data Leakage Attacks and Investigation,” *IEEE Netw.*, vol. 31, no. 2, pp. 67–71, 2017.
12. A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, “Detecting crypto-ransomware in IoT networks based on energy consumption footprint,” *J. Ambient Intell. Humaniz. Comput.*, pp. 1–12, Aug. 2017.
13. J. Baldwin and A. Dehghantanha, *Leveraging support vector machine for opcode density based detection of crypto-ransomware*, vol. 70. 2018.
14. A. D. James Baldwin, Omar Alhwai, *Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection*. Cyber Threat Intelligence- Springer Book, 2017.
15. D. Kiwia, A. Dehghantanha, K.-K. R. Choo, and J. Slaughter, “A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence,” *J. Comput. Sci.*, Nov. 2017.
16. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, “Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing,” *Eurasip J. Wirel. Commun. Netw.*, vol. 2016, no. 1, 2016.
17. F. Daryabar, A. Dehghantanha, and K.-K. R. Choo, “Cloud storage forensics: MEGA as a case study,” *Aust. J. Forensic Sci.*, pp. 1–14, Apr. 2016.
18. M. Shariati, A. Dehghantanha, and K.-K. R. Choo, “SugarSync forensic analysis,” *Aust. J. Forensic Sci.*, vol. 48, no. 1, pp. 95–117, Apr. 2015.
19. S. Almulla, Y. Iraqi, and A. Jones, “Cloud forensics: A research perspective,” in *2013 9th International Conference on Innovations in Information Technology, IIT 2013*, 2013, pp. 66–71.
20. O. Tabona and A. Blyth, “A forensic cloud environment to address the big data challenge in digital forensics,” in *2016 SAI Computing Conference (SAI)*, 2016, pp. 579–584.
21. Y. Gao and B. Li, “A forensic method for efficient file extraction in HDFS based on three-level mapping,” *Wuhan Univ. J. Nat. Sci.*, vol. 22, no. 2, pp. 114–126, 2017.
22. A. Guarino, “Digital Forensics as a Big Data Challenge,” in *ISSE 2013 Securing Electronic Business Processes*, 2013, pp. 197–203.
23. S. Zawoad and R. Hasan, “Digital Forensics in the Age of Big Data: Challenges, Approaches, and Opportunities,” in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, 2015, pp. 1320–1325.
24. B. Agrawal, R. Hansen, C. Rong, and T. Wiktorowski, “SD-HDFS: Secure deletion in hadoop distributed file system,” in *Proceedings - 2016 IEEE International Congress on Big Data, BigData Congress 2016*, 2016, pp. 181–189.

25. J. Baldwin, O. M. K. Alhawi, S. Shaughnessy, A. Akinbi, and A. Dehghantanha, “Emerging from the Cloud: A Bibliometric Analysis of Cloud Forensics Studies,” Springer, Cham, 2018, pp. 311–331.
26. F. Daryabar, A. Dehghantanha, B. Eterovic-Soric, and K.-K. R. Choo, “Forensic investigation of OneDrive, Box, GoogleDrive and Dropbox applications on Android and iOS devices,” *Aust. J. Forensic Sci.*, pp. 1–28, Mar. 2016.
27. F. Norouzizadeh Dezfouli, A. Dehghantanha, B. Eterovic-Soric, and K.-K. R. Choo, “Investigating Social Networking applications on smartphones detecting Facebook, Twitter, LinkedIn and Google+ artefacts on Android and iOS platforms,” *Aust. J. Forensic Sci.*, pp. 1–20, Aug. 2015.
28. S. H. Mohtasebi, A. Dehghantanha, and K.-K. R. Choo, *Cloud Storage Forensics: Analysis of Data Remnants on SpiderOak, JustCloud, and pCloud*. 2016.
29. A. Dehghantanha and T. Dargahi, *Residual Cloud Forensics: CloudMe and 360Yunpan as Case Studies*. 2016.
30. M. N. Yusoff, A. Dehghantanha, and R. Mahmod, *Network Traffic Forensics on Firefox Mobile OS: Facebook, Twitter, and Telegram as Case Studies*. 2016.
31. H. Haughey, G. Epiphanou, H. Al-Khateeb, and A. Dehghantanha, *Adaptive traffic fingerprinting for darknet threat intelligence*, vol. 70. 2018.
32. Y.-Y. Teing, D. Ali, K. Choo, M. T. Abdullah, and Z. Muda, “Greening Cloud-Enabled Big Data Storage Forensics: Syncany as a Case Study,” *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2017.
33. B. Martini and K. K. R. Choo, “Distributed filesystem forensics: XtreemFS as a case study,” *Digit. Investig.*, vol. 11, no. 4, pp. 295–313, 2014.
34. S. A. Thanekar, K. Subrahmanyam, and A. B. Bagwan, “A study on digital forensics in hadoop,” *Int. J. Control Theory Appl.*, vol. 9, no. 18, pp. 8927–8933, 2016.
35. P. Leimich, J. Harrison, and W. J. Buchanan, “A RAM triage methodology for Hadoop HDFS forensics,” *Digit. Investig.*, vol. 18, pp. 96–109, 2016.
36. Y. Gao, X. Fu, B. Luo, X. Du, and M. Guizani, “Haddle: A framework for investigating data leakage attacks in hadoop,” in *2015 IEEE Global Communications Conference, GLOBECOM 2015*, 2015.
37. S. Dinesh, S. Rao, and K. Chandrasekaran, “Traceback: A Forensic Tool for Distributed Systems,” *Proc. 3rd Int. Conf. Adv. Comput. Netw. Informatics*, pp. 17–27, 2016.
38. E. Alshammari, G. Al-Naymat, and A. Hadi, “A New Technique for File Carving on Hadoop Ecosystem,” in *The International Conference on new Trends in Computing Sciences (ICTCS'2017), At Jordan-Amman*, 2017.
39. Y.-Y. Teing, A. Dehghantanha, K.-K. R. Choo, T. Dargahi, and M. Conti, “Forensic Investigation of Cooperative Storage Cloud Service: Symform as a Case Study,” *J. Forensic Sci.*, vol. 62, no. 3, pp. 641–654, May 2017.
40. Y. Y. Teing, A. Dehghantanha, K. K. R. Choo, and L. T. Yang, “Forensic investigation of P2P cloud storage services and backbone for IoT networks: BitTorrent Sync as a case study,” *Comput. Electr. Eng.*, vol. 58, pp. 350–363, 2017.
41. M. Kohn, J. H. P. Eloff, and M. S. Olivier, “Framework for a Digital Forensic Investigation,” *Communications*, no. March, pp. 1–7, 2006.
42. M. E. Alex and R. Kishore, “Forensics framework for cloud computing,” *Comput. Electr. Eng.*, vol. 60, pp. 193–205, 2017.
43. B. Martini and K. K. R. Choo, “An integrated conceptual digital forensic framework for cloud computing,” *Digit. Investig.*, vol. 9, no. 2, pp. 71–80, 2012.
44. M. Rathbone, “A Beginner’s Guide to Hadoop Storage Formats (or File Formats).”
45. P. Zeyliger, “Hadoop Default Ports Quick Reference – Cloudera Engineering Blog.”
46. Apache Hadoop, “Apache Hadoop 2.9.0 – MapReduce Tutorial.”
47. M. Conti, A. Dehghantanha, K. Franke, and S. Watson, “Internet of Things security and forensics: Challenges and opportunities,” *Futur. Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan. 2018.

48. S. Watson and A. Dehghantanha, "Digital forensics: the missing piece of the Internet of Things promise," *Comput. Fraud Secur.*, vol. 2016, no. 6, pp. 5–8, Jun. 2016.
49. N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided Android malware classification," *Comput. Electr. Eng.*
50. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence," *IEEE Trans. Emerg. Top. Comput.*, pp. 1–1, 2017.
51. H. H. Pajouh, A. Dehghantanha, R. Khayami, and K. K. R. Choo, "Intelligent OS X malware threat detection with code inspection," *Journal of Computer Virology and Hacking Techniques*, pp. 1–11, 2017.

Internet of Things Camera Identification Algorithm Based on Sensor Pattern Noise Using Color Filter Array and Wavelet Transform



**Kimia Bolouri, Amin Azmoodeh, Ali Dehghantanha,
and Mohammad Firouzmand**

Abstract The Internet of Things (IoT) is cutting-edge technology of recent decade and has influenced all aspects of our modern life. Its significance and wide-range applications necessitate imposing security and forensics techniques on IoT to obtain more reliability. Digital cameras are the noteworthy part of IoT that play a vital role in the variety of usages and this entails proposing forensic solutions to protect IoT and mitigate misapplication.

Identifying source camera of an image is an imperative subject in digital forensics. Noise characteristics of image, extraction of Sensor Pattern Noise (SPN) and its correlation with Photo Response Non-Uniformity (PRNU) has been employed in the majority of previously proposed methods. In this paper, a feature extraction method based on PRNU is proposed which provides features for classification with Support Vector Machine (SVM). The proposed method endeavours to separate more powerful signals which is linked to camera sensor pattern noise by identifying color filter array pattern. To overcome the computational complexity, the proposed method is boosted by utilizing wavelet transform plus reducing dimensions of the image by selecting the most important components of noise. Our experiments demonstrate that the proposed method outperforms in terms of accuracy and runtime.

K. Bolouri

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada
e-mail: bolouri@shirazfava.ir

A. Azmoodeh

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

Department of Computer Science & Engineering, Shiraz University, Shiraz, Iran

e-mail: azmoodeh@cse.shirazu.ac.ir

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada
e-mail: ali@cybersciencelab.org

M. Firouzmand

Iranian Research Organization for Science and Technology (IROST), Tehran, Iran
e-mail: firouzmand@irost.org

Keywords Internet of Things · Forensic · Camera identification · Photo response non-uniformity (PRNU) · Sensor pattern noise (SPN) · Color filter array (CFA) · Wavelet transform · Support vector machine (SVM)

1 Introduction and Background

The Internet of Things (IoT) typically consists a wide-range of Internet-connected smart nodes that sense, store, transfer and process heterogenous collected data [1, 2]. IoT devices are influencing various aspects of life ranging from health [3] and agriculture [4] to energy management [5] and military domains [6]. Its pervasiveness, usefulness and significance have made IoT an appealing target for cyber-criminals [7].

Millions of smart nodes are communicating sensitive and private data through the IoT networks that necessitate protecting IoT against malicious attacks [8]. Above and beyond traditional malware, crypto-ransomware attacks are becoming a challenge for securing IoT infrastructure [9, 10]. IoT-based Botnets (Botnet of Things) and DDoS attacks using an IoT infrastructure are becoming major cyber threat [11, 12]. Therefore, it is important to propose efficient and accurate methods for forensics investigation of IoT networks [13, 14]. Digital cameras are inseparable and substantial subset of IoT nodes that play key role in a wide-range of applications and should possess forensic mechanism to boost its safety and admissibility [15].

Nowadays digital imaging instruments are expanding rapidly and replacing their analog counterparts. Digital images are penetrating all aspects of our life ranging from photography and pharmaceutical pictures to court evidences and the military services [16, 17]. In many applications, validity of images and identification of origin camera are crucial to judge about the images as an evidence. Particularly, electronic image identification techniques are vital in many judiciary processes. For instance, recognizing the source tools could disclose the mystery of felonies. Furthermore, recognizing the source of image is beneficial to every digital forensic approach related to images and videos [18].

To identify source camera of a taken image, it is necessary to have knowledge about what processing actions are performed on the real scene and how it has influenced final digital outcome. Majority of identification techniques have been proposed in this regard. A sizeable proportion of these methods leverage sensor pattern noise because it remains in image as a fingerprint of digital imaging sensor which is highly connected with the type of camera [19–22]. Lukas et al. [19] proposed a method which identified type of camera using noise correlation relating to Photo Response Non Uniformity (PRNU). In their method, it was important to collect a large-scale dataset of images relating to specified images for averaging them among the residual noise and obtaining Sensor Pattern Noise (SPN). Moreover, they analysed error rates change using JPEG compression and gamma correction.

Li [23] opined that stronger signal component in an SPN is the less trustworthy component and attenuated it. Then, by assigning weighting factors inversely to

the magnitude of the SPN components identified the source of images. Kang et al. [24] examined and compared source camera identification performance and introduced camera reference phase SPN and removed image's contamination and therefore reached more precise classification. They also presented by theoretical and practical experiments that their proposed method could achieve higher ROC outcome compared with [19].

Quiring and Kirchner [25] studied sensor noise in an adversarial environment and focused on sensor fingerprint that is fragile to lossy JPEG compression and proposed a method to accurately detect manipulated images by attackers. Applying camera identification on user authorization process, Valsesia et al. [26] leveraged high-frequency components of the photo-response nonuniformity and extracted it from raw images. They also presented an innovative scheme for efficient transmission and server-side verification. Sameer et al. [27] introduced a two-level classification mechanism using Convolutional Neural Networks to detect authentic and counter-forgery images and then type of undergone attack.

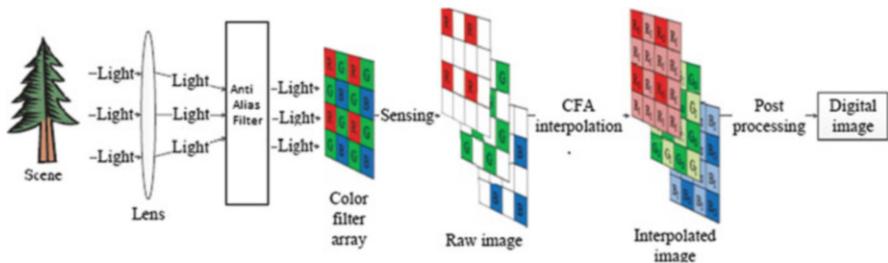


Fig. 1 Stages of imaging in digital camera [22]

The remainder of this paper is as follows. Section 2 reviews the structure of digital camera and Sect. 3 explains the proposed method and Sect. 4 evaluates the performance of it. Section 5 completes this paper by discussing and concluding about the achievements of this paper.

2 Digital Camera

In this section, we briefly explain processing stages which occur in a particular digital camera and investigate to find features that facilitate recognizing source camera of images. Imaging stages in digital camera are illustrated in Fig. 1.

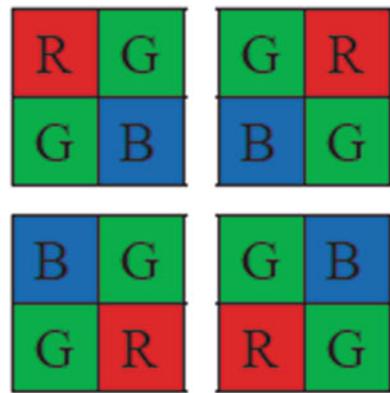
Light enters a set of lenses from the scene and is passed through anti-aliasing filters. Then, it is accessible to color filter array and in each pixel, one of the components of Red (R), Green (G) and Blue (B) is taken into account because this sensor is only able to pass one color in each pixel. Over here, an interpolation process is applied to estimate intensity of two other colors for each pixel using neighborhoods. In the

next stage, the sequence of image processing operations like color correction, white color balance, gamma correction, image improvement, JPEG compaction etc., have been performed. Afterwards, the image is saved as a particular image file. Some specifications in this process can be applied for camera type identification such as sensor pattern noise (SPN), camera dependency response function, resampling, color filter array interpolation, JPEG compression and lens aberrations. In this study, we utilize a combination of camera pattern noise and color filter array for recognizing source camera of an image.

2.1 Camera Noise

The noise which is available in images originates from different sources. In order to source camera identification, we explore all types of noise which are caused by camera and has the minor dependency to surrounding environment or the subjects which are imaged.

Fig. 2 Bayer pattern



Based on source of noise, they are categorized as temporal, spatial and a combination of both. The noise which appears in the reconstructed image and is fixed in a definite location of image, indicates Fixed Pattern Noise (FPN). Basically, since it has been fixed from spatial point of view, it can be removed in dark points using signal processing methods. The principle component of FPN is current non-uniformity of the dark points in a charge-coupled device (CCD) sensor. This can be due to photo contact time or high temperature. Complementary metal-oxide semiconductor (CMOS) is the main sources of FPN current non-uniformity of dark points [28]. The considerable point in FPN is that spatial pattern of these changes remains fixed. Since FPN is added to all frames or images produced with a sensor and is independent of light, it can be deleted from the image by reducing a dark frame. The source which is similar to FPN in terms of characteristics is PRNU. In fact, this is the same difference in response to light in pixel when light enters

the sensor. One of the reasons is non-uniformity of size of the active areas which throughout light photons are absorbed. PRNU is the result of silicon material heterogeneity and some other effects which have been created during production process.

2.2 *Color Filter Array and Bayer Pattern*

As described in camera imaging stages, light enters Color Filter Array (CFA) sensors after passing lens and anti-visual filters. This array is two-dimensional and monochromic. In other words, sensors pass only one color in each pixel (Fig. 1). In order to obtain three color channels, an interpolation function should be applied and two subsequent colors for each pixel should be identified. Color pattern is different due to various types of CFA in different cameras. Among these patterns, Bayer Pattern [29] is leveraged frequently and includes four dimensional matrix which is repeated during CFA. This pattern includes two green pixels, one blue pixel and one red pixel and has four different models dependent on pixel location (Fig. 2). Considering that human vision is more sensitive to green color, most pixels are allocated to green color in this pattern.

It is obvious that the signal which is located in CFA is more powerful and since it is directly received from the sensor is an appropriate option to be used for extraction of features and type detection of digital camera. Therefore, this pattern can be applied as a fingerprint to identify source of image. In fact, it can be recognized that the sampled image in each pixel has been entered directly from sensor of camera or has been interpolated.

3 Proposed Method

Considering the fact that different cameras possess various noise characteristics due to their sensor specifications, disparate patterns should be obtained from the images. To recognize images' source, the proposed method includes two main building block namely Feature Extraction and PRNU Pattern Classification.

3.1 *Feature Extraction*

PRNU noise is extracted by de-noising filter that is rooted in wavelet transform. De-noising filter is explained in [30]. This filter of image extracts Gaussian noise with a given variance (which enters the filter as an input). Based on this assumption, image and noise are additive composition of non-static Gaussian signal which have a given variance in wavelet range. We obtain residual noise of image by subtracting de-noised image from original image (Eq. (1)).

$$PRNU = I - I_{denoised} \quad (1)$$

$I_{denoised}$ is obtained with wavelet de-noising filter[30] and consequently, we are able to compute high frequency components of image which have been lost in the de-noised image as image noise pattern. It is to be noted that complexities and edges of image are also effective on the obtained pattern. In this method, PRNU and image dimensions are same. In the subsequent section three different feature extraction methods are described. Section 3.1.1 explains basic PRNU feature extraction method and Sects. 3.1.2 and 3.1.3 belong to the proposed method.

3.1.1 Basic PRNU

This feature extraction method uses pattern noise of the entire image similar to Lukas et al. [19] algorithm. The difference is that image vector extraction method and vectors classification with SVM classifier have been used instead of using averaging method and calculate camera sensor noise pattern and correlation rate. Moreover, this algorithm employ PCA for dimension reduction. Two main stages to extract desired feature are as follows:

1. Extracting pattern noise across the entire image using Eq. (1).
2. Using PCA for dimensionally reduction and eliminating useless components of noise

Considering what we explained in Sect. 2.2, all pixels of the image have not been received directly from sensor of camera and some of them are the result of interpolation based on color filter array pattern. Therefore, noise pattern obtained with this method (Stage 1) includes color interpolation errors. In fact, most digital camera identification algorithms which act based on pattern noise, use one of the color channels.

3.1.2 PRNU Extraction Considering Color Channels

In order to reduce color interpolation error and promote pattern noise, in [22] colors analysis in pattern noise and extraction of Color-Decoupled Photo Response Non-Uniformity are used. For this purpose, it has analyzed image into four sub-images for each color channel and extracted PRNU pattern noise using Eq. (1). To compare the proposed algorithm with this algorithm, we use CD-PRNU instead of extracting PRNU in the first stage of feature extraction. The proposed algorithm considers all color channels in extraction of pattern noise as well as CD-PRNU extraction algorithm. It is notable that CFA pattern has not been certainly recognized in CD-PRNU extraction algorithm and the extracted pattern noise is a combination of pattern noise of all three RGB color channels. Instead, CFA pattern is identified in the proposed algorithm using Choi et al. method [31]. In this regard, we only select the pixels from pattern noise which are related to CFA.

A method for identifying color filer array (CFA) which uses Intermediate Value Counting is presented in [31] and has been formed based on the primary hypothesis that CFA interpolation fills empty pixels using neighbor pixels. As described in

Sect. 2.2, there are different pixel patterns for coloring green, red and blue channels. For each channel, a special neighborhood is defined and intermediate value is counted. CFA pattern is estimated by information of these intermediate values in three channels. Finally, one of the four Bayer patterns (RGGB, GRBG, BGGR and GBRG) is attributed to any camera.

In the proposed algorithm, we first determine CFA pattern using intermediate value counting method and then in PRNU pattern, select pixels which are related to CFA. It is assumed that PRNU matrix is a noise pattern extracted from image (Eq. (1)). For an image of $M \times N$ with three RGB color channels, PRNU matrix will be $M \times N$ with three color channels. To select pixels among pattern noise, we present Eqs. (2)–(5).

$$\begin{aligned} CFA_{RGGB}(2x - 1, 2y - 1) &= PRNU_R(2x - 1, 2y - 1) \\ CFA_{RGGB}(2x - 1, 2y) &= PRNU_G(2x - 1, 2y) \\ CFA_{RGGB}(2x, 2y - 1) &= PRNU_G(2x, 2y - 1) \\ CFA_{RGGB}(2x, 2y) &= PRNU_B(2x, 2y) \end{aligned} \quad (2)$$

$$\begin{aligned} CFA_{GRBG}(2x - 1, 2y) &= PRNU_R(2x - 1, 2y) \\ CFA_{GRBG}(2x - 1, 2y - 1) &= PRNU_G(2x - 1, 2y - 1) \\ CFA_{GRBG}(2x, 2y) &= PRNU_G(2x, 2y) \\ CFA_{GRBG}(2x, 2y - 1) &= PRNU_B(2x, 2y - 1) \end{aligned} \quad (3)$$

$$\begin{aligned} CFA_{BGGR}(2x, 2y) &= PRNU_R(2x, 2y) \\ CFA_{BGGR}(2x - 1, 2y) &= PRNU_G(2x - 1, 2y) \\ CFA_{BGGR}(2x, 2y - 1) &= PRNU_G(2x, 2y - 1) \\ CFA_{BGGR}(2x - 1, 2y) &= PRNU_B(2x - 1, 2y - 1) \end{aligned} \quad (4)$$

$$\begin{aligned} CFA_{GBRG}(2x - 1, 2y - 1) &= PRNU_R(2x - 1, 2y - 1) \\ CFA_{GBRG}(2x - 1, 2y) &= PRNU_G(2x - 1, 2y) \\ CFA_{GBRG}(2x, 2y - 1) &= PRNU_G(2x, 2y - 1) \\ CFA_{GBRG}(2x, 2y) &= PRNU_B(2x, 2y) \end{aligned} \quad (5)$$

Where $x \in [1, 2, \dots, M]$ and $y \in [1, 2, \dots, N]$ and $PRNU_R, PRNU_G, PRNU_B$ are PRNU matrix in green, red and blue channels respectively. CFA matrix is a two-dimensional $M \times N$ matrix which includes values of the pixels directly received from color filter array sensors. In fact, CFA matrix has been obtained from combination of PRNU color channels. Image of this combination for RGGB pattern is shown in Fig. 3. Finally, PCA is applied on calculated CFA matrix to reduce dimension.

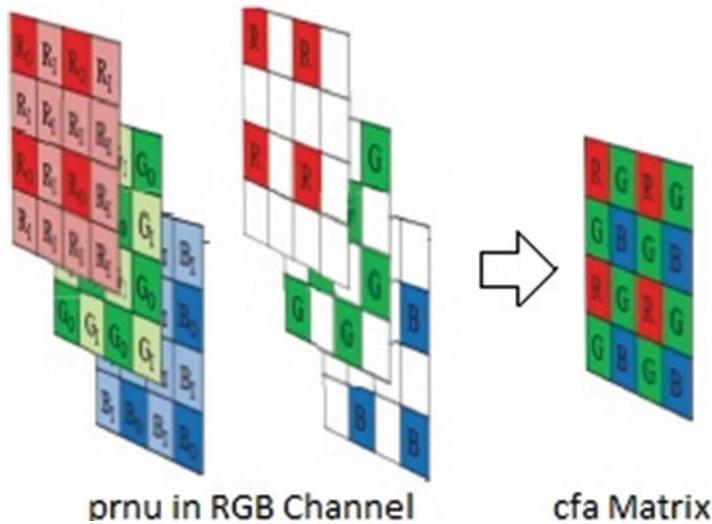


Fig. 3 Obtaining CFA matrix from PRNU



Fig. 4 Wavelet transforms bands in a hypothetical image

3.1.3 Reduction of Image Dimensions Using Wavelet Transform

Although PCA is applied at the final stage of methods in Sects. 3.1.2 and 3.1.1, high resolution of digital images and consequently processing time remains another problem to solve. In order to mitigate this issue, we leverage Multi-Resolution Theory and Discrete Wavelet Transform [32]. Suppose $M \times N$ image as input of algorithm, we will have four sub bands (LL: image estimation, LH: horizontal, HL: vertical and HH: diametrical) as output. Effect of wavelet transform on a test image and bands resulting from it are shown in Fig. 4. Since our goal is to retain pattern noise and reduce dimensions of the image, we use HH which includes the most important parts of pattern noise and perform all stages of the proposed algorithm on HH band of wavelet transform. In this regard, the dimensions of image is 4^l -times smaller (l is the level of wavelet transform). It is evident that higher wavelet transform level (the higher value of l), leads to lower dimensions of

image and higher speed of image processing. However, exceeding in l decrease the detection identification performance. Figure 5 illustrates how to analyze image in HH band.

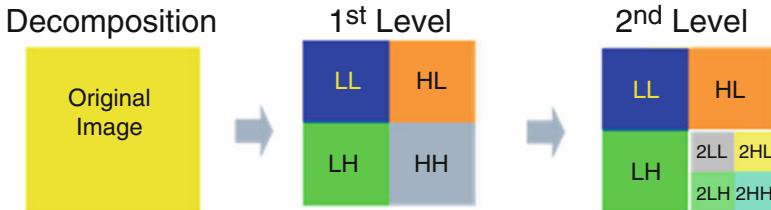


Fig. 5 Wavelet transform bands to level 2

Table 1 Images dataset information

Label	Camera model	Sensor	CFA pattern	Image size
C1	Canon EOS400D	CMOS	RGGB	3888*2592
C2	Kodak EasyShare CX7530	CCD	GBRG	2560*1920
C3	HP PhotoSmart E327	CCD	RGGB	2560*1920
C4	Panasonic DMC-FZ20	CCD	RGGB	3648*2738
C5	Canon PowerShot A400	CCD	GBRG	1536*2048
C6	Panasonic Lumix DMC-FS5	CCD	GRBG	3648*2736

3.2 Noise Classification

Support Vector Machines (SVM) [33] are a set of classification methods relating to supervised learning which analyze data and identify the patterns. To utilize SVM, it is necessary that each of the samples be represented as a vector of real numbers. In our experiments, input part is a vector of features which is extracted from PRNU noise, CFA pattern and its wavelet transform. Furthermore, PCA is applied to reducing dimensions of data and selecting more beneficial elements of data [34]. Output of SVM classifier is a label predicting camera type.

4 Experiments

4.1 Dataset and Settings

Dataset used in this research include images of 6 digital cameras shown in Table 1. each dataset includes 130 images. Photographing conditions are equal in all images. All pictures were taken on a tripod without flash, automatic focus,

without magnitude and the best compression quality and also other options have been automatically set on the default values. In addition, attempt has been made to have images with different subjects and non-uniform edges and complexities so as to simulate real-world condition. To assess the validity of our performance evaluation, 10-fold cross-validation is applied. System specification to run experiments is Microsoft Windows7, CPU Intel Corei3 3.20GHz and 4Gb of RAM.

4.2 Results

At the first stage, features explained in Sect. 3.1 are extracted for PRNU [19], CD-PRNU [22] and the proposed algorithm. In the next stage, a SVM classifier is trained for each algorithm based on 10-fold cross-validation algorithm. Accuracy is the performance evaluation metric which is the number of samples that a classifier correctly detects, divided by the number of all samples (Eq. (6)).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

Table 2 Accuracy obtained from the proposed algorithm and comparison with previous methods

	PRNU [19]	CD-PRNU [22]	Proposed method
C1	91%	89.00%	99.50%
C2	83.00%	83.33%	91.83%
C3	84%	90.00%	97.17%
C4	91.83%	94.67%	100%
C5	90%	86.83%	99.50%
C6	94.67%	91.67%	99.83%
Average	89%	89.25%	97.97%

Table 3 Results of wavelet transform application and image dimensions reduction

	$l = 1$	$l = 2$	$l = 3$
C1	99.33%	100%	100%
C2	85.83%	84.17%	83.50%
C3	86.67%	83.67%	83.17%
C4	97%	89%	82.83%
C5	98.17%	84.50%	84.83%
C6	99.17%	100%	100%

Results of these three algorithms are shown in Table 2. As can be seen from Table 2, CD-PRNU feature extraction were negligibly accurate compared with PRNU while accuracy has been considerably improved in the proposed algorithm.

In order to evaluate effect of applying wavelet transform on runtime, we apply three levels of it ($l = 1, 2, 3$) on the images to reduce processing time and then use HH part of wavelet transform instead of the image. Implementation results are shown in Table 3 and Fig. 6 compares runtime of different methods. It is obvious although higher wavelet transform level leads to lower accuracy in identification, the runtime speed increases significantly due to reduction of the image size in higher levels of wavelet transform.

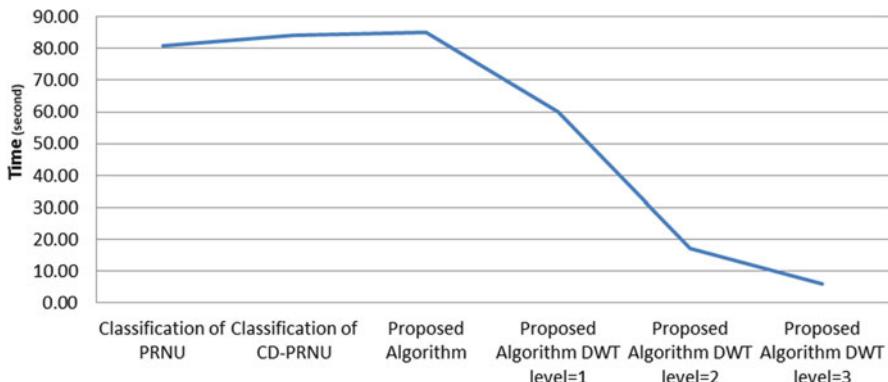


Fig. 6 Algorithm comparison in terms of time

5 Conclusion

Nowadays, the Internet of Things penetrating all aspect of our life and enormous digital images are taken in small intervals by IoT nodes for different applications. This soar in usage and importance of IoT necessitates proposing applicable method for forensic issues. Camera identification refers to a category of methods that endeavour to recognize source camera based on images taken by it.

In this paper, an algorithm has been proposed for extracting features so as to digital camera identification. The proposed algorithm has been evaluated by a dataset including images taken by six different digital cameras. To increase accuracy, we proposed a new feature extraction method to generate CFA pattern. Furthermore, to select the principle components of pattern noise and to reduce dimensions of pattern noise, we leveraged HH part of wavelet transform and results demonstrate upsurge of runtime speed. The proposed method outperform 97.07% of accuracy which are higher than the average result obtained in the rival methods, 89% and 89.25% from PRNU [19] and CD-PRNU [22] respectively. To outdo other methods in terms of runtime, we contributed that speed considerably will increases using HH component of wavelet transform while average accuracy remains approximately above 90%.

References

1. M. Conti, A. Dehghantanha, K. Franke, S. Watson, Internet of things security and forensics: Challenges and opportunities, *Future Generation Computer Systems* 78 (2018) 544–546.
2. S. Watson, A. Dehghantanha, Digital forensics: the missing piece of the internet of things promise, *Computer Fraud & Security* 2016 (2016) 5–8.
3. S. Walker-Roberts, M. Hammoudeh, A. Dehghantanha, A systematic review of the availability and efficacy of countermeasures to internal threats in healthcare critical infrastructure, *IEEE Access* 6 (2018) 25167–25177.
4. S. Zhongfu, Y. S. Du Keming, Development trend of internet of things and perspective of its application in agriculture [j], *Agriculture Network Information* 5 (2010) 21.
5. A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities, *IEEE Internet of Things journal* 1 (2014) 22–32.
6. A. Azmoodeh, A. Dehghantanha, K. K. R. Choo, Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning, *IEEE Transactions on Sustainable Computing* (2018) 1–1.
7. G. Epiphaniou, P. Karadimas, D. K. B. Ismail, H. Al-Khateeb, A. Dehghantanha, K. K. R. Choo, Non-reciprocity compensation combined with turbo codes for secret key generation in vehicular ad hoc social IoT networks, *IEEE Internet of Things Journal* (2017) 1–1.
8. H. HaddadPajouh, A. Dehghantanha, R. Khayami, K.-K. R. Choo, A deep recurrent neural network based approach for internet of things malware threat hunting, *Future Generation Computer Systems* 85 (2018) 88–96.
9. A. Azmoodeh, A. Dehghantanha, M. Conti, K.-K. R. Choo, Detecting crypto-ransomware in IoT networks based on energy consumption footprint, *Journal of Ambient Intelligence and Humanized Computing* (2018) 1–12.
10. H. H. Pajouh, R. Javidan, R. Khayami, D. Ali, K. K. R. Choo, A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks, *IEEE Transactions on Emerging Topics in Computing* (2016) 1–1.
11. S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha, R. Khayami, Botshark: A deep learning approach for botnet traffic detection, *Cyber Threat Intelligence* (2018) 137–153.
12. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, M. Dlodlo, Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing, *EURASIP Journal on Wireless Communications and Networking* 2016 (2016) 130.
13. R. Hegarty, D. J. Lamb, A. Attwood, Digital evidence challenges in the internet of things., in: INC, pp. 163–172.
14. C. Esposito, A. Castiglione, F. Pop, K. K. R. Choo, Challenges of connecting edge and cloud computing: A security and forensic perspective, *IEEE Cloud Computing* 4 (2017) 13–17.
15. A. Ariffin, K.-K. R. Choo, Z. Yunos, Forensic readiness: A case study on digital CCTV systems antiforensics, in: *Contemporary Digital Forensic Investigations of Cloud and Mobile Applications*, Elsevier, 2017, pp. 147–162.
16. J. Nakamura, *Image sensors and signal processing for digital still cameras*, CRC press, 2017.
17. J. Fridrich, Digital image forensics, *IEEE Signal Processing Magazine* 26 (2009) 26–37.
18. T. V. Lan, K. S. Chong, S. Emmanuel, M. S. Kankanhalli, A survey on digital camera image forensic methods, in: *2007 IEEE International Conference on Multimedia and Expo*, pp. 16–19.
19. J. Lukas, J. Fridrich, M. Goljan, Digital camera identification from sensor pattern noise, *IEEE Transactions on Information Forensics and Security* 1 (2006) 205–214.
20. C.-T. Li, Source camera linking using enhanced sensor pattern noise extracted from images (2009).
21. K. Matsushita, H. Kitazawa, An improved camera identification method based on the texture complexity and the image restoration, in: *Proceedings of the 2009 International Conference on Hybrid Information Technology*, ACM, pp. 171–175.

22. C.-T. Li, Y. Li, Digital camera identification using colour-decoupled photo response non-uniformity noise pattern, in: Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, IEEE, pp. 3052–3055.
23. C. T. Li, Source camera identification using enhanced sensor pattern noise, *IEEE Transactions on Information Forensics and Security* 5 (2010) 280–287.
24. X. Kang, Y. Li, Z. Qu, J. Huang, Enhancing source camera identification performance with a camera reference phase sensor pattern noise, *IEEE Transactions on Information Forensics and Security* 7 (2012) 393–402.
25. E. Quiring, M. Kirchner, Fragile sensor fingerprint camera identification, in: 2015 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 1–6.
26. D. Valsesia, G. Coluccia, T. Bianchi, E. Magli, User authentication via PRNU-based physical unclonable functions, *IEEE Transactions on Information Forensics and Security* 12 (2017) 1941–1956.
27. V. U. Sameer, R. Naskar, N. Musthyala, K. Kokkalla, Deep learning based counter-forensic image classification for camera model identification, in: C. Kraetzer, Y.-Q. Shi, J. Dittmann, H. J. Kim (Eds.), *Digital Forensics and Watermarking*, Springer International Publishing, Cham, 2017, pp. 52–64.
28. H. M. X. L. Abbas El Gamal, Boyd A. Fowler, Modeling and estimation of FPN components in CMOS image sensors, volume 3301, pp. 3301 – 3301 – 10.
29. A. Bosco, M. Mancuso, Noise filter for Bayer pattern image data, 2008. US Patent 7,369,165.
30. M. K. Mihcak, I. Kozintsev, K. Ramchandran, Spatially adaptive statistical modeling of wavelet image coefficients and its application to denoising, in: *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 6, IEEE, pp. 3253–3256.
31. C.-H. Choi, J.-H. Choi, H.-K. Lee, CFA pattern identification of digital cameras using intermediate value counting, in: *Proceedings of the Thirteenth ACM Multimedia Workshop on Multimedia and Security, MM&Sec '11*, ACM, New York, NY, USA, 2011, pp. 21–26.
32. S. G. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, *IEEE transactions on pattern analysis and machine intelligence* 11 (1989) 674–693.
33. M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, *IEEE Intelligent Systems and their applications* 13 (1998) 18–28.
34. S. Wold, K. Esbensen, P. Geladi, Principal component analysis, *Chemometrics and intelligent laboratory systems* 2 (1987) 37–52.

Protecting IoT and ICS Platforms Against Advanced Persistent Threat Actors: Analysis of APT1, Silent Chollima and Molerats



Samuel Grooby, Tooska Dargahi, and Ali Dehghantanha

Abstract One of the greatest threats to cyber security is the relatively recent increase in intrusion campaigns conducted by well trained, well-funded and patient adversaries. These groups are known as advanced persistent threats and they are a growing concern for governments and industries around the world. APTs may be backed by terrorist organisations, hacktivists or even nation state actors, conducting covert cyber-warfare against other countries. Due to the advanced capabilities of these groups, a non-targeted, catch-all defence strategy is unlikely to be successful. Instead, potential targets of APTs must be able to research and analyse previous attacks by the groups in order to tailor a cyber defence triage process based on the attacker's modus operandi. In this paper we attempt to do just that using Diamond Model and kill chain analysis to craft a course of action matrix for three example APT groups.

Keywords Big data · Advanced persistent threat · APT · Diamond model · Cyber kill chain

1 Introduction

Advanced Persistent Threat (APT) groups are a recent and innovative type of adversary facing organisations and nation states today [1]. These groups boast sophisticated malware [2, 3] and substantial resources. The term APT was coined in 2006 by United States Air Force Analysts as a placeholder for the classified identity of attackers [4]. The National Institute of Standards and Technology defines an APT

S. Grooby · T. Dargahi

Department of Computer Science, School of Computing, Science and Engineering, University of Salford, Manchester, UK

e-mail: S.A.Egunjobi@edu.salford.ac.uk; T.Dargahi@Salford.ac.uk

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

e-mail: ali@cybersciencelab.org

as “*An adversary that possesses sophisticated levels of expertise and significant resources which allow it to create opportunities to achieve its objectives by using multiple attack vectors (e.g. cyber, physical, and deception)*” [5]. APT attacks strategies have been adopted by hacktivists [6], terrorists [7] and criminals looking for financial gain [8, 9]. The rise in prevalence of these groups is spurring on the APT security solution market which is expected to grow significantly [10].

A key distinction between APT groups and garden variety attackers is the amount of resources available to each group [11]. APT groups may be backed by nation states and have salaried employees [12] meaning that attacks can continue for as long as the target has value to the sponsor [13]. These groups have the goal of attacking the integrity or confidentiality of an information asset of a specific target [14, 15] and have the ability to use customised tools and develop zero-day exploits to achieve this [16–18]. The use of zero-day exploits implies a high level of focus in the attacker [19], as they are committed enough to allocate a portion of their resources to buy a zero-day [20] or to conduct the research needed to develop one [21].

In this paper, we are mainly focusing on three APT campaigns which were mainly targeting IoT and ICS devices namely APT1 which is a Chinese group which has targeted organisations from industries in English speaking countries which the Chinese government has identified as strategic to their growth [22]; Silent Chollima which is a North Korean group responsible for the 2014 Sony Pictures Entertainment cyberattack [23]; and Molerats which is a middle-eastern group attacking targets in Palestine and the surrounding countries [24]. Knowing the common avenues of attack for each APT we can attempt to fashion a proper defensive procedure [25].

Currently there is a lack of literature on defence mechanisms tailored for individual APT groups [26]. A cyber defence triage process (CDTP) was developed for industrial controls systems in [27]. The authors provided a method to identify and defend critical ICS equipment from cyber-attacks by combining the Diamond Model of Intrusion Analysis, the Mandiant Attack Lifecycle, and the CARVER Matrix. This paper focuses on the ICS as a potential target of any threat group instead of tailoring a triage process to one specific group. In [14] a blacklist based intrusion detection system (IDS) is proposed in order to improve upon the shortcomings of current IDSs when it comes to defence against APTs. In [28], traffic from large network environments was analysed in order to formulate a method to identify APT hosts. In [29] a simulation of a waterholing attack is conducted and a policy is proposed to mitigate the impacts of these attacks by APTs. All these papers propose defensive strategies, however none focus on defence against specific APT groups, instead using common characteristics of APTs to fashion a solution.

In this paper we develop a cyber defence triage process for three APT groups targeting IoT and ICS environments, namely APT1, Silent Chollima and Molerats. In particular, in our CDTP we apply well-recognized attack attribution models, namely the Diamond Model (for details refer to Sect. 3.5) [30], the Lockheed Martin intrusion kill chain (for details refer to Sect. 3.4) [31] and the Course of Action (CoA) Matrix (for details refer to Sect. 3.7) [31]. Our CDTP will be personalised to counter each group’s tactics, techniques and procedures (TTPs). This research will be valuable to organisations which match the criteria of past targets of these

groups as hopefully by applying our CDTP they might be able to find solutions for mitigating the impact of future attacks.

The rest of the paper is structured as follows: Sect. 2 gives an overview of the three APT groups that we are analysing. In Sect. 3 we perform Diamond and Course of Action Matrix analysis of our chosen APT actors. In Sect. 4 we conclude the paper, discussing our findings.

2 Overview

In this section we will give an overview of APT groups of this study. This will include the time span of each group's activity, their methods and how their different attacks have been linked together.

2.1 APT1

APT1 is one of the most prolific APT groups currently active. The campaign has been attributed to China's People's Liberation Army (although this is denied by the Government of China) and their primary focus appears to be to steal information assets such as intellectual property from organisations in the English-speaking world.

As reported by Mandiant [22], the first attack attributed to APT1 was in 2006, however the compile times of some malware samples date back to 2004. A number of 141 separate victims of compromise by the APT1 group have been identified. These victims come from 15 different countries and 20 different industries, 115 of which were based in the United States of America. The distribution of attacks between industries is more even, suggesting that no single industry is the primary target of the group.

APT1 most often uses spear phishing to first compromise a target. After downloading a malicious file from a personalised email, the target machine will open a backdoor to the group's command and control (C2) servers. From there the attacker will scout the target network for the information it is looking for, escalating their privilege as needed, before sending the information assets back to China compressed in .RAR files.

Analysis by Mandiant [22] has concluded that APT1 is likely a branch of the People's Liberation Army designated as Unit 61398. The Unit hire staff who are qualified in computer and network security as well as the English language. Two of the four large Shanghai networks APT1 activity has been traced to belong to the Pudong New Area where Unit 61398's headquarters are located. These findings point to Unit 61398 and APT1 being the same organisation. In addition to this, the industries of APT1 targets correspond to areas which the Government of China has identified as strategic to their growth in their 12th Five Year Plan.

It is estimated that Unit 61398 is made up of hundreds or potentially thousands of employees, based upon the building space they have been provided with [32]. In 2007 a new building was opened for Unit 61398 which could potentially accommodate 2000 workers. This is not the largest building the group has at their disposal. An investigation by the FBI has led to members of this group being charged with 31 counts, including economic espionage. This is the first time criminal charges have been filed against member of a known state hacking group.

2.2 *Molerats*

Molerats, also known as “Gaza Hackers Team” is an APT group which originally attacked Israeli and Palestinian targets. The methods used by the group have become more sophisticated over time, and their range of targets has increased to organisations in neighbouring countries as well as USA and Europe, however most attacks remain in the Middle East.

The Fagerland report in 2012 [33] investigated spear phishing emails sent to the Israeli government. Following further incidents of infection more Israeli targets were found as well as Palestinian ones, each with lures adapted to their target. It was found that some of the bait documents attached to the emails sent to Palestinian targets were created in 2011, earlier than those sent to Israeli ones (2012). This suggests that some events may have caused the original focus of the group to change. Waves of attacks in 2014 [34] and operation DustySky in 2016 [35] have expanded the number of countries targeted by the group.

In several early attacks the group has used spear phishing with lures copied from news headlines to deliver self-extracting archive files. The archives contained media such as documents, images or videos as well as an off the shelf backdoor trojan called XtremeRat. The 2016 DustySky campaign did not use spear phishing in its emails, however they were still targeted albeit not at an individual level. The DustySky malware NeD Worm was first reported in the DustySky campaign suggesting that it was developed specially. The change in methodology of the group from using off the shelf malware to custom malware shows that the resources available to Molerats may have increased since their inception.

The exact affiliation and motivation of the Gaza hackers group is still unknown; however, a range of attacks has been attributed to a single group. The original Palestinian and Israeli attacks were linked by their C2 infrastructure, some malware from the different incidents had also been signed by the same digital certificate. IP addresses found to have been used by “Gaza cybergang” were also used in operation DustySky [35]. One malware sample from DustySky was uploaded to

VirusTotal. The compilation time and upload time were found to be only seconds apart suggesting it was uploaded by its creator to check if it could be detected. The upload originated in Gaza, and so it is likely that the group is based there.

2.3 *Silent Chollima*

Silent Chollima is an APT group with ties to North Korea, being named for the mythical Chollima, which is an important symbol there. Its attacks have included espionage and data wiping in South Korea, and the Sony Pictures leak.

A report by Sherstobitoff, Liba and Walter into a 2013 attack on financial services and media firms in South Korea found that a campaign against South Korean targets had been ongoing since 2010. This campaign was named Operation Troy [36]. The initial compromise was caused by a watering-hole attack, with later compromises coming from spear-phishing attacks.

Another campaign against South Korean think tanks was reported upon in 2013 [37] called the Kimsuky operation. Spear phishing was used to deliver a DLL file to targets in this attack. At first instalment the malware makes itself autorun upon system start up. The malware provides several reconnaissance functions including keylogging, document theft and remote-control access.

In 2014 Sony Pictures Entertainment was hacked by the group in response to the production of a comedy film about North Korea. When Sony refused to comply with the group's demands for them to cancel release of the film, the hackers leaked confidential data including employee personal information and intellectual property and then wiped their computer infrastructure. A report by the FBI attributed the attack to North Korea, citing undisclosed sources [23]. The group responsible is also believed to be the same as the perpetrators of Operation Troy, as the same malware and C2 infrastructure was used in both.

3 APT Group Analysis

In this section we use the Diamond Model of intrusion analysis [30] and the Lockheed Martin intrusion kill chain [31] to analyse publicised intrusions conducted by our APT groups. We then use this information to create a Course of Action (CoA) Matrix [31], which will give potential targets of these groups a roadmap in how to defend themselves. The analysis will include finding indicators at each stage of the intrusion kill chain, creating activity threads for each intrusion, defining an adversary process based on the intrusions for each group and finally providing a CoA Matrix based on these adversary processes.

3.1 APT1 Intrusion Attempts

Here we outline the intrusion attempts made by APT1 which we will use for our analysis.

3.1.1 Intrusion Attempt 1

In a 2011 article by Hoglund, a campaign of attacks into commercial and government systems is outlined [38]. In the attacks the adversary sends spear-phishing emails containing a lot of information about the targets. The lures are so convincing that Hoglund believes the attackers have access to emails stolen from the targets in their reconnaissance. The email automatically downloads a dropper malware onto the victim machine, this is then used to download a second more versatile piece of malware, a Remote Access Trojan (RAT), before the dropper is uninstalled. To communicate with the RAT the adversary uses a compromised third-party webserver. The RAT will periodically check comments on the compromised website where the attackers place hidden instructions for the malware. The comments are base64 encoded and are decoded by the malware and used as a configuration file. The config file has the ability to specify which server addresses to check for further instruction among other options specified by headers. These headers can be used to detect the malware:

- [ListenMode]
- [MServer]
- [BServer]
- [Day]
- [Start Time]
- [End Time]
- [Interval]
- [MWeb]
- [BWeb]
- [MWebTrans]
- [BWebTrans]
- [FakeDomain]
- [Proxy]
- [Connect]
- [Update]
- [UpdateWeb]

3.1.2 Intrusion Attempt 2

In 2012 a honeynet was set up by Trend Micro consisting of several honeypot machines across the globe [39]. The purpose of this experiment was to gather data on targeted attacks into SCADA systems. In December 2012 a highly targeted attack on

a honeypot located in the US was detected. The honeypot was disguised as a water pressure station including a website which displayed an email which could plausibly belong to the city government. A phishing email was sent to the address with a file named “CITYREQUEST.doc” attached. This suggests the attacker was attempting to exploit human complacency, hoping that an employee of the local government would download and open the file thinking it was a legitimate request coming from a citizen.

Clicking on the file opens a word document with one line of text. The document then automatically closes whilst a dialog box opens with the title “tomb-keeper@126.com” and an unreadable string of characters as the message. Clicking “OK” on the box sends messages to C2 servers in China and US letting the adversary know the payload has been opened, it also produces the files ai.exe and gh.exe. gh.exe is a tool for dumping password hashes found on the network to a text file, ai.exe was found to be HACKSFASE malware as it contained the string “tthack-fas@#\$”. Three hours after CITYREQUEST.doc was first opened, ai.exe began exfiltrating the Security Accounts Manager (SAM) database, VPN configuration files, hostname, IP address, and location to the adversary. Through the C2 network the attackers conducted more reconnaissance looking for communication patterns and disabled the firewall and antivirus software.

3.1.3 Intrusion Attempt 3

A report by Symantech details intrusions into several aerospace and defence industry organisations in 2013 [40]. Payloads were delivered to high-ranking staff within the companies via spear-phishing emails with an industry outlook report as a lure. The sender email was one that could plausibly belong to the author of the report. The attached pdf uses an exploit in the Adobe Flash Player CVE-2011-0611 ‘SWF’ File Remote Memory Corruption Vulnerability to drop malware on the victim’s machine disguised as an instance of svchost.exe. This malware then drops a malicious ntshrui.dll. Both files are the malware Backdoor.Barkiofork which has the ability to exfiltrate data from the victim via a C2 server at osamu.update.ikwb.com.

3.2 Molerats *Intrusion Attempts*

Here we outline the intrusion attempts made by the APT group known as Molerats which we will use for our analysis.

3.2.1 Intrusion Attempt 1

Snore Fagerland authors a 2012 report into a series of attacks on Palestinian and Israeli targets [33]. The initial attack against the Israeli government consisted of

a sent spear-phishing email claiming to be the IDF chief of staff. The email had a news story lure and a malicious attachment. The attachment is a self extracting archive file which has been disguised as a Word document by extending the filename with “-” characters. The archive contains several files including a the benign news story barrage.doc and an instance of the XtremeRAT backdoor malware renamed Word.exe. The malware has a false digital signature supposedly from Microsoft (03 e1 e1 aa a5 bc a1 9f ba 8c 42 05 8b 4a bf 28) however it will not validate properly. Upon execution XtremeRAT connects to external hosts which resolve to addresses in the US. Most addresses change regularly, however the host lokia.mine.nu at address 69.80.107.129 has remained since at least the date of the report. The malware has the ability to exfiltrate surveillance data such as screenshots, webcam footage and microphone recordings.

3.2.2 Intrusion Attempt 2

An article on FireEye recounts attacks against US and Middle Eastern targets in June-July 2013 [41]. The adversary sent spear-phishing emails to the targets either with a malicious RAR file attached or containing links to the file hosted on third party sites. The archive contains a .scr file, however the attachment has a long filename including “.doc” which obfuscates the true file extension. This dropper malware then downloads an instance of PoisonIVY malware. The malware is signed with a false Microsoft certificate just as the XtremeRAT malware was. The configuration for the malware revealed a plaintext password: “!@#GooD#@!”, which can be used as an identifier. The malware exfiltrates keylogs, screenshots and webcam footage via the C2 network previously used in intrusion attempt 1 over port 443.

3.2.3 Intrusion Attempt 3

A report by ClearSky details the DustySky campaign which started in May 2015 [35]. Among the targets of this campaign were embassies, aerospace and defence companies; financial institutions; journalists and software developers. Each attack of the campaign began with a phishing email being sent to the target. The emails contain new article lures and claim to be sent by members of the Israel Defense Forces. Some of the emails were sent from the IP addresses 185.12.187.105 and 31.223.186.71, belonging to an ISP based in Gaza. They also either have a RAR or ZIP archive attached, or they contain a link to one. The link can be described with the regular expression “\([A-Za-z]+\.\php\)?(:id|token1|token2|C)=\([A-Za-z0-9\+=%]*=\{0,2\}\&\?\){4}” Upon clicking the link, the User-Agent string of the

target is checked. If the user is on a Windows Machine then the malicious archive is downloaded, else the user is redirected to a fake email account login page in order to steal the user's credentials. In some cases, the attackers used these credentials to log in to the email address in order to conduct more reconnaissance. The downloaded archive file contains executables disguised as word documents, opening them presents the expected content and runs a malicious macro, infecting the host machine with the DustySky malware under the name Plugin.exe. The malware contains two hardcoded C2 domains which it will beacon out to with a GET request to TEST.php or index.php. After it receives a response it will send information about the infected machine such as username, operating system, antivirus. The parameters of the message can be described with the regular expression “ $\backslash\{[A-Za-z]\{2,5\}\}\backslash.php\backslash?(\:(Pn|fr|GR|com|ID|o|ho|av|v)=\{[A-Za-z0-9V=+]\}*\{0,2\}&\?)\{5,9\}$ ”. It may use the following paths when communicating with the C2 server:

- Update.php
- conn.php
- geoiploc.php
- news.htm
- pass.php
- passho.php
- passyah.php

3.3 Silent Chollima Intrusion Attempts

Here we outline the intrusion attempts made by APT group Silent Chollima which we will use for our analysis.

3.3.1 Intrusion Attempt 1

In 2013 a malware campaign was conducted against South-Korean think tanks [37]. Spear-phishing attacks were used to deliver a dropper which would proceed to download other malware. If the malware finds itself on a Windows 7 machine it uses the Win7Elevate exploit to inject malicious code into explorer.exe. It then saves a surveillance malware with a hardcoded name e.g. “~DFE8B437DD7C417A6D.TMP”. The malware disables AhnLab firewalls should it find one, this is a brand of firewall commonly used in South Korea. System information is collected and saved in the hardcoded path “C:Program FilesCommon

FileSystemOle DBoledvbs.inc.” Finally the malware exfiltrates keylogs, .hwp documents and directory listings through a Bulgarian public email server(mail.bg) using hardcoded credentials. The following accounts were used in the campaign:

- beautifl@mail.bg
- ennemyman@mail.bg
- fasionman@mail.bg
- happylove@mail.bg
- lovest000@mail.bg
- monneyman@mail.bg
- sportsman@mail.bg
- veryhappy@mail.bg

3.3.2 Intrusion Attempt 2

On March 20th 2013 an intrusion into South Korean media and financial firms was conducted [36]. A spear-phishing email was opened by a user which downloaded a remote access trojan malware. The RAT creates a hidden instance of Internet Explorer and injects itself into the IE process, modifying the registry to allow for remote connections to its C2 server. It then uses an internal server to distribute a dropper malware which then installs a third malware named “[29]” on machines in the network. This malware has the ability to wipe the Master Boot Record and it was found to contain the strings “principes” and “hastati”. Once sufficient data has been exfiltrated, the adversary sends a command for the third malware to wipe the MBR of the machines it is installed upon, effectively destroying the machines.

3.3.3 Intrusion Attempt 3

The 2014 Sony hack resulted in the leaking of a huge amount of the company’s insider information and intellectual property [42]. A person alleging to be a member of the group responsible for the attack claimed that a lack of physical security was the cause of the attack [43]. System administrator credentials were used to drop malware on the network [44]. When executed the malware installs itself as a Windows service and attempts to connect to a C2 network through one of three hardcoded IP addresses [45]. It then removes the machines capability of sending email by shutting down the Microsoft Exchange Information Store service and dismounting Exchange’s databases. The infected machine will then be suspended for 2 h before rebooting with a blank hard drive [45].

3.4 Kill Chain Analysis

The intrusion kill chain [31] is a way of modelling the different stages of an adversary's intrusion into the network of their target in order to meet an objective [46]. The adversary must complete an intrusion event at each phase of the kill chain in order to advance to the next phase, meaning that if they are stopped at any phase before their objective is completed then the defender has been successful. The phases of the kill chain are as follows [31]:

- Reconnaissance—Conducting reconnaissance requires learning as much information as possible about the target network as well as employees in order to find possible vulnerabilities and other data which will be useful in later stages of the intrusion. Useful tools for this are advanced Google searches of the public parts of the target network and social media platforms.
- Weaponization—Using data learned in the previous stage to craft a payload for delivery to the target. The payload must be able to exploit a vulnerability in the target's system to gain access.
- Delivery—Exposing the target network to the payload often requiring a level of social engineering. Common methods of delivery include phishing attacks, spear-phishing attacks and water-holing attacks.
- Exploitation—Using vulnerabilities in operation systems or user programs or even user incompetence to execute malicious code on the target system.
- Installation—Installation of malicious software on the target environment giving the adversary a foothold from which the adversary can execute the capabilities of their payload. Adversaries may be able to take measures to ensure their malware remains on the system after discovery and attempted removal at this phase.
- Command and Control (C2)—A channel of communication between the installed malware and the adversary which allows them access to the target system as if it was their own machine. Malware may be given specific commands at this stage.
- Actions on Objective—Here the adversary completes their objectives i.e. the compromise of the confidentiality, integrity or availability of the target's information assets. Examples of this include data exfiltration, wiping of hard drives, conducting surveillance or editing sensitive information. With a rising amount of cyber-physical systems in the world objectives may include controlling the behaviour of mechanisms which have an effect on the physical world [27].

3.4.1 APT1 Intrusion Indicators

The indicators for each stage of the intrusion kill chain for the APT1 is depicted in Table 1.

Table 1 Indicators at each stage of the intrusion kill chain for APT1 intrusion attempts

Phase	Intrusion Attempt 1	Intrusion Attempt 2	Intrusion Attempt 3
Reconnaissance	Recipient list Stolen email information	Google dorks for web facing industrial control systems Recipient email address	Lure file: “Global A&D outlook 2012.pdf”
Weaponization			
Delivery	Email subject Email body Sender email address Sender IP address	Email subject Email body Sender email address Sender IP address CITYREQUEST.doc attachment	Email subject: “2012 Global aerospace and defense industry outlook” Email body Sender email address Sender IP address Global A&D outlook 2012.pdf attachment
Exploitation		tombkeeper@126.com Unreadable text string	Adobe Flash Player CVE-2011-0611 ‘SWF’ File Remote Memory Corruption Vulnerability
Installation	Dropper malware Remote Access Trojan	Ai.exe Gh.exe “tthackfas@#”	svchost.exe ntshru.dll Backdoor.Barkiofork
C2	Exploited middleman webserver IP Base64 encoded instructions Configuration file headers: [ListenMode] [MServer] [BServer] Etc. Port 80 or 443 connection	C2 server IP address	osamu.update.ikwb.com
Action on objectives	N/A	N/A	N/A

3.4.2 Molerats Intrusion Indicators

The indicators for each stage of the intrusion kill chain for the Molerats APT group is depicted in Table 2.

3.4.3 Silent Chollima Intrusion Indicators

The indicators for each stage of the intrusion kill chain for the Silent Chollima APT group is depicted in Table 3.

Table 2 Indicators at each stage of the intrusion kill chain for Molerats intrusion attempts

Phase	Intrusion Attempt 1	Intrusion Attempt 2	Intrusion Attempt 3
Reconnaissance	Benign file: barrage.doc	Lure file: Hamas shoot down Israeli F-16 fighter jet by modern weapon in Gaza sea.doc	Recipient list
Weaponization			
Delivery	Email subject Email body False IDF chief of staff sender address Sender IP address	Email subject Email body Link to third party file hosting site Sender email address Sender IP address	Email subject Email body contains link described by “V[A-Za-z]+.php?(?id token token2 C)=[A-Za-z0-9%+=%]*=(0,2)&?)\{4” Sender header mentions Israeli Defense Force Sender IP address: 185.12.187.105, 31.223.186.71
Exploitation	Hidden .scr extension		
Installation	Word.exe Fake Microsoft digital signature: 03 e1 e1 aa a5 bc a1 9f ba 8c 42 05 8b 4a bf 28	Malware password: “@#GooD#@!” Fake Microsoft digital signature: 06 50 11 a5 bc bf 83 c0 93 28 16 5e 7e 85 27 75	Plugin.exe
C2	69.80.107.129 lokia.mine.nu		Communication parameters described by: “V[A-Za-z][2-5]\.php?(?:(Pn fr GR com ID o ho av v)= A-Za-z0-9%+=%)*=(0,2)&?)\{5,9” Paths used: <ul style="list-style-type: none">• Update.php<ul style="list-style-type: none">• comm.php• geoiploc.php• news.htm• pass.php• passho.php• passyah.php [GET request]
Action on objectives	N/A	N/A	N/A

Table 3 Indicators at each stage of the intrusion kill chain for Silent Chollima intrusion attempts

Phase	Intrusion Attempt 1	Intrusion Attempt 2	Intrusion Attempt 3
Reconnaissance	Recipient list	Recipient list	
Weaponization			
Delivery	Email subject Email body Sender email address Sender IP address	Email subject Email body Sender email address Sender IP address	Stolen credentials
Exploitation	Win7Elevate exploit		
Installation	Hardcoded malware name: “~DFE8B437DD7C417A6D.TMP” Hardcoded path for system info: C:\Program Files\Common Files\System\Ole\Boledvb.s.inc.	“principes” “hastati” AgentBase.exe	
C2	Bulgarian public email server accounts: beautiful@mail.bg ennemyman@mail.bg fashionman@mail.bg happylove@mail.bg lovest000@mail.bg monneyman@mail.bg sportsman@mail.bg veryhappy@mail.bg Hardcoded credentials		Hardcoded IP addresses
Action on objectives	N/A	N/A	N/A

3.5 Activity Thread Analysis

We can model individual events that occur during an intrusion with the *Diamond Model of Intrusion Analysis* [30]. In this model a single intrusion event has four core features, each given a corresponding value for how confident the analyst is that the feature is correct. These attributes are as follows:

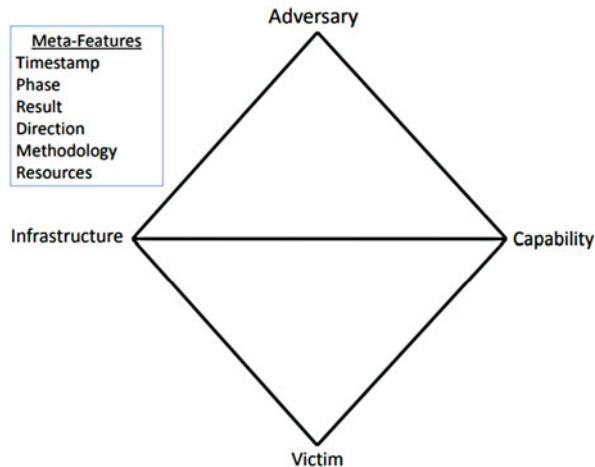
- Adversary—the entity responsible for the intrusion. This can be further split into the Adversary Operator who actually executes the event and the Adversary Customer who sanctions the event. These two can be one in the same however.
- Capability—the tools and methods used in the event. For example, the ability to command and control malware on the victim machine is a capability of the adversary.
- Infrastructure—resources that the adversary needs to execute the event, this can include IP addresses used to issue command and control, or email addresses used to conduct spear-phishing attacks. Infrastructure can either be wholly under the control of the adversary (type 1) or be controlled by an intermediary (type 2) in an attempt to obfuscate the source of the intrusion.

- Victim—the target of the intrusion event, either the systems attacked or the owner of the systems may be referred to as the victim of the event.

In order to be considered an intrusion event an adversary must have used a capability over an infrastructure against a victim in order to produce a result, however not all features need to be known by the analyst to model the event. Each core feature is represented graphically as a vertex of a diamond shape giving the model its name.

In addition to the core features, extra meta-features can be recorded about the even such as timestamps, phase (referring to the phase of attack in a chain of events), result, direction, methodology and resources (Fig. 1).

Fig. 1 Diagram of the Diamond model of intrusion analysis taken from [30]



The model allows for the analytic technique of pivoting, where a feature is analysed and used to find more information relating to the same or other features. For example, a malware sample is reverse engineered, and a command and control domain is found, this uses a capability of the adversary to find more data on the infrastructure used in the event [47]. Another strength of the diamond model is its ability to be combined with other intrusion models to produce a more comprehensive presentation of an attack.

By combining the Diamond Model of intrusion analysis with the intrusion kill chain we have a way of modelling the entire chain of events in an attack which results in an adversary achieving their objective. Placing each intrusion event in a phase of the kill chain provides a two-dimensional representation of the attack which shows the cause and effect relationship between events. This results in a directed phase ordered graph in which Diamond events are linked by arcs which represent the relationship between the events. The graph is split into vertical threads providing the intrusion events of an attack on a single victim, however events in separate threads may be linked by arcs.

Each graph is accompanied by two tables which annotate the events and arcs respectively. Each event can be described as actual, if evidence for the occurrence of the event can be provided, or hypothetical if the analyst has reason to believe it occurred. Arcs are given a confidence value for how likely the analyst believes it is that one event led to another, and a AND/OR value depending on if there are more than one arcs to the next event. Arcs also are described as hypothetical or actual depending on whether one or both of the events they describe the relationship of is hypothetical.

3.5.1 APT1 Intrusion Activity Threads

In this section Fig. 2 shows the activity threads we have created for each APT1 intrusion event accompanied by the event description and arc description tables (Tables 4 and 5).

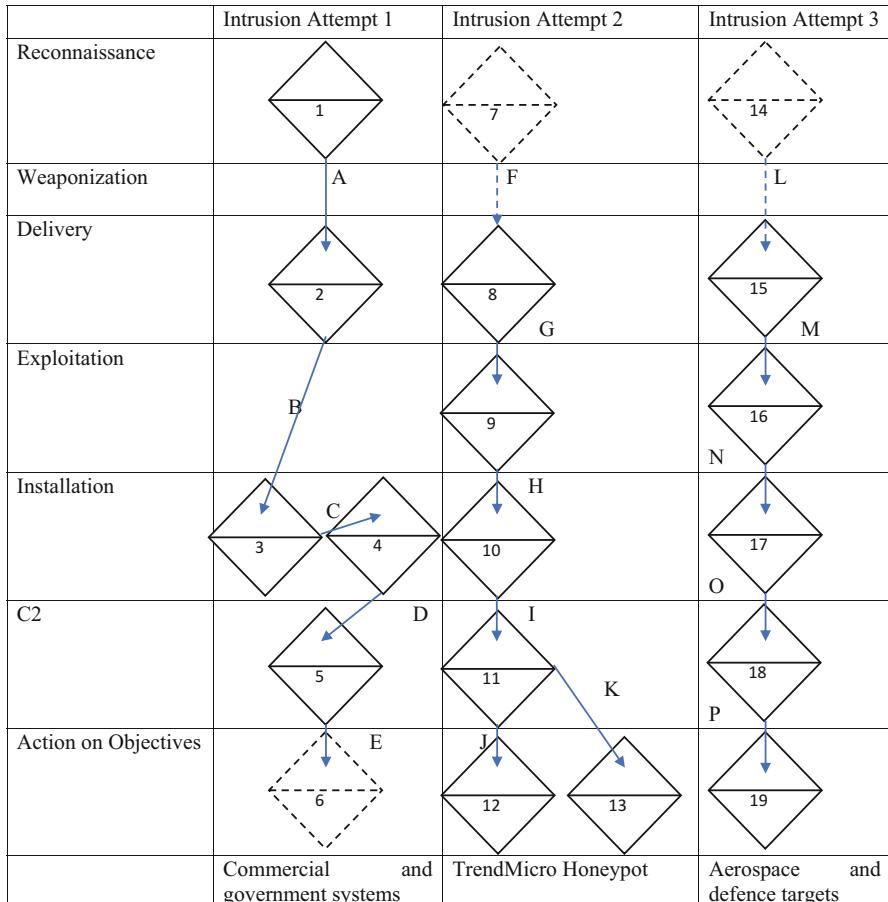


Fig. 2 Activity threads for each APT1 intrusion

Table 4 Activity thread event descriptions for Fig. 2

Event	Hypothesis/actual	Description
1	Actual	Adversary learns about their target through stolen emails
2	Actual	Using knowledge gained from stolen emails convincing spear-phishing emails are constructed and sent to the target
3	Actual	Backdoor program is installed onto victim network
4	Actual	Using the backdoor the adversary installs a RAT and uninstalls the original backdoor
5	Actual	Adversary places a hidden comment on a middleman webpage which the RAT will connect to periodically. The RAT deciphers the comment and receives instructions from it
6	Hypothesis	Data is exfiltrated from the network
7	Hypothesis	Adversary visits the honeypot website searching for email addresses before finding one disguised as a real city government email address
8	Actual	Adversary sends spear-phishing email with the attachment “CITYREQUEST.doc” to the discovered address
9	Actual	The attachment is opened revealing a decoy word document which automatically closes, a dialogue box then opens with the title “tombkeeper@126.com” and an unreadable message. Clicking ok executes the malicious code
10	Actual	The files gh.exe and ai.exe are dropped on the machine, gh.exe is a password hash dump file and ai.exe is the “HACKSFASE” malware. Antivirus software and the local firewall are disabled
11	Actual	Contact is made to the C2 server to signal installation, C2 begins to send commands to the malware
12	Actual	Data is exfiltrated to the C2 server including hostname, location, IP address, Security Accounts Manager database, VPN config files
13	Actual	At the order of the C2 server the machine sends pings and traceroutes to default gateways and adjacent networks, as well as ARP commands to look for communication patterns
14	Hypothetical	Adversary uses a Google search to find targets with important roles in organisations
15	Actual	A spear-phishing attack is used with a report on industry outlook as lure. A malicious pdf file is attached
16	Actual	The PDF attachment attempts to exploit Adobe Flash Player vulnerability CVE-2011-0611
17	Actual	A backdoor.barkiofork malware named svchost.exe file is dropped on the machine
18	Actual	Contacts the command-and-control (C&C) server at osamu.update.ikwb.com
19	Actual	Exfiltrates system information

Table 5 Activity thread arc descriptions for Fig. 2

Arc	Confidence	And/or	Hypothesis/actual	Provides
A	High	And	Actual	Provides adversary with information necessary to create an effective spear-phishing lure
B	High	And	Actual	[None]
C	High	And	Actual	[None]
D	High	And	Actual	[None]
E	High	And	Actual	[None]
F	Low	And	Hypothesis	Provides the email address used as spear-phishing target
G	High	And	Actual	[None]
H	High	And	Actual	[None]
I	High	And	Actual	Provides persistence mechanism so continued observation can be conducted on the network
J	High	And	Actual	[None]
K	High	And	Actual	[None]
L	Medium	And	Hypothesis	Provides targets for spear-phishing email
M	High	And	Actual	[None]
N	High	And	Actual	[None]
O	High	And	Actual	[None]
P	High	And	Actual	[None]

3.5.2 Molerats Intrusion Activity Threads

In this section we present the activity threads we have created for each Molerats intrusion event accompanied by the event description and arc description tables (Tables 6 and 7).

3.5.3 Silent Chollima Intrusion Activity Threads

In this section we present the activity threads we have created for each Silent Chollima intrusion event accompanied by the event description and arc description tables (Tables 8 and 9).

3.6 Adversary Process

An adversary process is a sub-graph of an activity thread [30]. By analysing multiple activity threads from an adversary, we can identify intrusion events which reoccur, in order to devise a kind of signature for the adversary. This adversary process graph can potentially be used to attribute intrusions of unknown source to the adversary

Table 6 Activity thread event descriptions for Fig. 3

Event	Hypothesis/actual	Description
1	Actual	Spear-phishing email purporting to come from IDF chief of staff including a malicious attachment
2	Actual	The user opens the file which they believe to be a word document, however the attachment has a long filename including “.doc” which obfuscates the true file extension which is a “.scr” archive. The archive contains an XtremeRAT malware sample
3	Actual	XtremeRAT malware connects to external hosts which resolve to IP addresses in the US
4	Hypothesis	Malware exfiltrates surveillance data such as screenshots, webcam footage, microphone recordings
5	Actual	Spear-phishing email sent to target with malicious attachment
6	Actual	Spear-phishing email sent to target containing link to third party website which hosts malicious file
7	Actual	The user opens the file which they believe to be a word document, however the attachment has a long filename including “.doc” which obfuscates the true file extension which is a “.scr” archive. The malware drops an instance of PosionIVY malware
8	Actual	Malware communicates with C2 network previously used in event 3 over port 443
9	Hypothesis	Malware exfiltrates surveillance data such as screenshots, webcam footage, microphone recordings
10	Hypothesis	Adversary uses advanced Google search to find email addresses of targets
11	Actual	Phishing email is sent to target with non-personalised news story lure, email contains archive file attachment
12	Actual	Phishing email contains malicious link. If the user is on a Windows machine then the link will download the malicious files, if not then a false email log in page will load to harvest credentials
13	Actual	Credentials are used to log into the account where more reconnaissance can be done
14	Actual	User opens the archive file which contains executables disguised as word documents. Opening the executable file presents the word document and runs a macro that installs the malware
15	Actual	The malware sends a GET request to one of two hardcoded domains and waits for the server to reply with an OK
16	Actual	If the first server does not respond in time the malware will send the GET request to the second server
17	Actual	Malware exfiltrates screenshots, active processes, files containing certain keywords and passwords stored in browsers to the C2 server

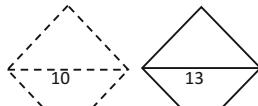
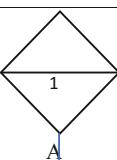
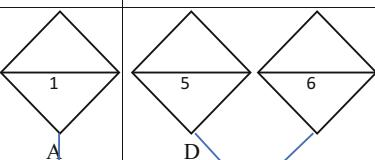
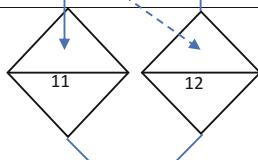
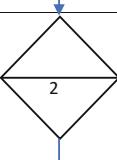
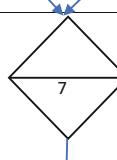
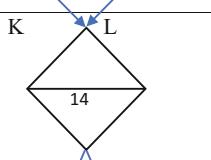
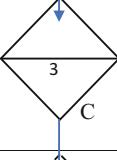
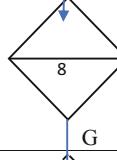
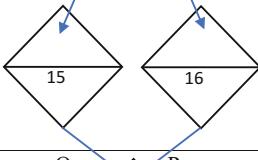
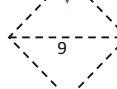
	Intrusion Attempt 1	Intrusion Attempt 2	Intrusion Attempt 3
Reconnaissance			
Weaponization			
Delivery	 A	 D E	 H I J
Exploitation			 K L
Installation	B	F	M N
C2	 C	 G	 O P
Action on Objectives			 Q
	Israeli government 2012	Middle Eastern and US targets June-July 2013	DustySky 2015

Fig. 3 Activity threads for each Molerats intrusion

in question, although this must be accompanied with more concrete evidence as it does not account for copycat attacks. An organisation or individual who believes they are a potential target for the adversary can use the adversary process as a tool to influence which security measures they apply.

Table 7 Activity thread arc descriptions for Fig. 3

Arc	Confidence	And/or	Hypothesis/actual	Provides
A		And	Actual	[None]
B		And	Actual	[None]
C		And	Hypothesis	[None]
D		And	Actual	[None]
E		And	Actual	[None]
F		And	Actual	[None]
G		And	Hypothesis	[None]
H	Low	Or	Hypothesis	Provides email addresses for phishing attack
I	Low	Or	Hypothesis	Provides email addresses for phishing attack
J	High	Or	Actual	Provides email credentials for further reconnaissance
K	High	And	Actual	[None]
L	High	And	Actual	[None]
M	High	Or	Actual	[None]
N	High	Or	Actual	[None]
O	High	And	Actual	[None]
P	High	And	Actual	[None]

3.6.1 APT1

Based off the activity threads of their intrusion attempts we have created an adversary process for APT1 as shown in Table 10.

3.6.2 Molerats

Based off the activity threads of their intrusion attempts we have created an adversary process for Molerats as shown in Table 11.

3.6.3 Silent Chollima

Based off the activity threads of their intrusion attempts we have created an adversary process for Silent Chollima as shown in Table 12.

3.7 Course of Action Matrix

A *Course of Action (COA) Matrix* is an extension of the kill chain in which it is combined with several different methods of defending against an intrusion [30]. These methods are as follows:

Table 8 Activity thread event descriptions for Fig. 4

Event	Hypothesis/actual	Description
1	Hypothesis	Adversary gains physical access to victim headquarters and obtains sysadmin credentials
2	Actual	Adversary uses sysadmin credentials to drop malware on the network
3	Hypothesis	Spear-phishing email is sent to user
4	Hypothesis	User accesses water-holed website which delivers malware
5	Actual	Dropper installs itself as a Windows service on victim machine
6	Actual	Malware attempts to connect to c2 network through one of three hardcoded IP addresses
7	Actual	Malware shuts down Microsoft Exchange Information Store services and makes email inaccessible
8	Actual	Windows is suspended for 2 h and hard drive is wiped
9	Actual	Spear-phishing email is sent to user
10	Actual	Remote access trojan creates a hidden instance of Internet explorer and injects itself into the process
11	Actual	Remote access trojan modifies the registry to allow for remote connections
12	Actual	The malware searches drives for files containing military specific keywords and exfiltrates them to the attacker's server
13	Actual	Remote access trojan uses internal server to distribute dropper trojan across the network. Dropper trojan installs the wiper malware
14	Actual	Wiper malware wipes the machine's master boot record making them unbootable
15	Actual	Spear-phishing email is sent to user
16	Actual	Malware uses win7elevate to inject malicious code into explorer.exe
17	Actual	Trojan dropper delivers encrypted library. Malware achieves persistence between system reboots with DriverManage, WebService and WebClientManager functions
18	Actual	Attackers communicate with the malware through a Bulgarian public email server using credentials hardcoded in the malware
19	Actual	Malware exfiltrates keystrokes, .hwp documents and directory listings

- Detect—Find evidence of the intrusion so that a defensive response can be made.
- Deny—Block the adversary's ability to successfully complete an intrusion event.
- Disrupt—Interfere with intrusion events mitigating negative effects.

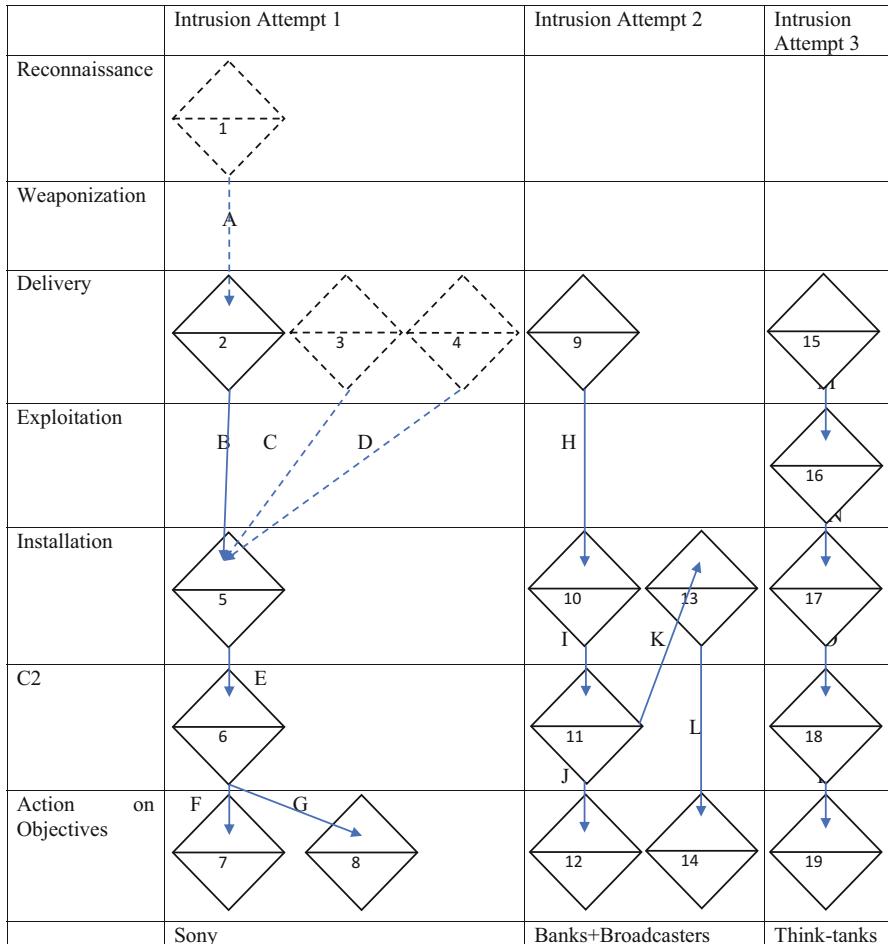


Fig. 4 Activity threads for each Silent Chollima intrusion

- Degrade—Reducing the efficiency of the attacker slowing them down.
- Deceive—Present the adversary with misinformation in order to alter their future actions
- Destroy—Counterattack against the adversary's infrastructure

Each element of the matrix defines an action the defender can take to either detect, deny, disrupt, degrade, deceive or destroy the attempted intrusion at each phase of the kill chain.

Table 9 Activity thread arc descriptions for Fig. 4

Arc	Confidence	And/or	Hypothesis/actual	Provides
A	Low	And	Hypothesis	Provides sysadmin credentials
B	Medium	And	Actual	[None]
C	Low	And	Hypothesis	[None]
D	Low	And	Hypothesis	[None]
E	High	And	Actual	[None]
F	High	And	Actual	[None]
G	High	And	Actual	[None]
H	High	And	Actual	[None]
I	High	And	Actual	[None]
J	High	And	Actual	[None]
K	Medium	And	Actual	[None]
L	High	And	Actual	[None]
M	High	And	Actual	[None]
N	High	And	Actual	
O	High	And	Actual	
P	High	And	Actual	

Table 10 Adversary process for APT1

		Process features
Reconnaissance		
Weaponization		
Delivery		The adversary delivers the payload using a spear-phishing attack [Derived from events 2, 8 and 15]
Exploitation		
Installation		Backdoor malware is dropped onto target system [Derived from events 3, 10, 17]
C2		
Action on objectives		Sensitive data is exfiltrated from the target environment [Derived from events 12, 19]

3.7.1 APT1 COA Matrix

Here we show several courses of action that can be taken against APT1 at each stage of their adversary process (Table 13).

Table 11 Adversary process for Molerats

	Process features
Reconnaissance	
Weaponization	
Delivery	 Email with malicious attachment is sent to target (either spear-phishing or phishing attack) [Derived from events 1, 5, 11]
Exploitation	  User opens malicious file which is disguised as a word document [Derived from events 2, 7, 14]
Installation	
C2	 Malware connects to C2 infrastructure (in the case of events 3 and 8 the same infrastructure was used) [Derived from events 3, 8, 15]
Action on objectives	

Table 12 Adversary process for Silent Chollima

	Process features
Reconnaissance	
Weaponization	
Delivery	
Exploitation	
Installation	 Backdoor is installed on victim machine [Derived from events 5, 10, 17]
C2	  Malware communicated with C2 network via hardcoded address [Derived from events 6, 18]
Action on objectives	  Victim hard-drives are wiped after data is exfiltrated [Derived from events 8, 12, 14, 19]

3.7.2 Molerats COA Matrix

Here we show several courses of action that can be taken against Molerats at each stage of their adversary process (Table 14).

3.7.3 Silent Chollima COA Matrix

Here we show several courses of action that can be taken against Silent Chollima at each stage of their adversary process (Table 15).

Table 13 Course of action matrix derived from the adversary process

	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance						
Weaponization						
Delivery	Verify sender email before opening attachment Email security software			Respond saying the email will be opened later	Open email on honeypot machine	
Exploitation				Allow installation privileges only on administrator accounts	Allow installation on honeypot machine	
Installation	Antivirus scanner software					
C2	Network capture analysis	Encrypt data at storage level, store private key separately Remove internet connection upon detection	Firewall	Data masking techniques	Allow adversary to exfiltrate false data from Honeypot machine	Hide a malware instance in the exfiltrated data (likely illegal)
Action on objectives						

Table 14 Course of action matrix derived from the adversary process

	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance						
Weaponization						
Delivery	Educate users on how to recognise an email is coming from a malicious source Network Intrusion Detection Software	Verify sender email before opening attachment Email security software		Respond saying the email will be opened later	Open email on honeypot machine	
Exploitation	Educate users on the importance of double checking file extensions before opening Host intrusion detection software	Do not open files from unknown sources Make sure machines on the network are full updated	Data execution prevention software		Communicate with adversaries through honeypot machine	
Installation						
C2	Network intrusion detection software Network capture analysis	IP whitelist	Network intrusion prevention software			
Action on objectives						

Table 15 Course of action matrix derived from the adversary process

	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance						
Weaponization						
Delivery						
Exploitation						
Installation	Antivirus scanner software			Allow installation privileges only on administrator accounts	Allow installation on honeypot machine	
C2	Network intrusion detection software Network capture analysis	IP whitelist	Network intrusion prevention software		Communicate with adversaries through honeypot machine	
Action on objectives	Network capture analysis	Encrypt data at storage level, store private key separately Remove internet connection upon detection	Firewall Back up sensitive data in off-network location	Data masking techniques	Allow adversary to exfiltrate false data from Honeypot machine	Hide a malware instance in the exfiltrated data (likely illegal)

4 Conclusion and Future Works

The aim of this paper was to create a cyber defence triage process for three APT groups targeting IoT and ICS infrastructure namely APT1, Molerats, and Silent Chollima. In order to do this we made use of the Diamond Model of Intrusion Analysis and the Lockheed Martin intrusion kill chain, both of which were used in the construction a cyber defence triage process for industrial control systems. After detailing several intrusion attempts by the three groups, we used the intrusion kill chain to display indicators at different stages of the attacks. We then used activity threads, a function of the diamond model, to display individual events and the links between them in each attack. Based on these activity threads we looked for common elements and converted these into an adversary process for each APT group. Finally, we used these common elements to devise a course of action matrix for each group, which will aid potential targets in the defence of their networks.

In future this analysis could be expanded to cover more intrusion attempts conducted by the APT groups as well as applying the methodology to other APT groups. It would be interesting to analyse APT groups developing and targeting cloud platforms [48], big data platforms [49], handheld and mobile devices [50]. Moreover, understanding how new technologies such as machine learning [51] and block-chain can be used to model APT groups activities ranging from Ransomware [52] and general Botnet [53] and DDoS [54] attacks to specially created customised cyber attack tools, would be another future work of this research.

References

1. H. Haughey, G. Epiphanou, H. Al-Khateeb, and A. Dehghantanha, *Adaptive traffic fingerprinting for darknet threat intelligence*, vol. 70. 2018.
2. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, “Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence,” *{IEEE} Trans. Emerg. Top. Comput.*, p. 1, 2017.
3. D. Kiwia, A. Dehghantanha, K.-K. R. Choo, and J. Slaughter, “A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence,” *J. Comput. Sci.*, Nov. 2017.
4. B. E. Binde, R. McRee, and T. J. O’Connor, “Assessing Outbound Traffic to Uncover Advanced Persistent Threat,” 2011.
5. NIST, “Glossary of Key Information Security Terms,” 2013.
6. N. Villeneuve, N. Moran, M. Scott, and T. Haq, “OPERATION SAFFRON ROSE,” 2013.
7. S. E. Goodman, J. C. Kirk, and M. H. Kirk, “Cyberspace as a medium for terrorists,” *Technol. Forecast. Soc. Change*, vol. 74, no. 2, pp. 193–210, 2007.
8. A. Earls, “APTs New waves of advanced persistent threats are vastly improved and smarter than ever.,” *ebook SC Magazine*, 2015.
9. H. Haddad Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo, “A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks,” *IEEE Trans. Emerg. Top. Comput.*, pp. 1–1, 2016.
10. G. M. Insights, “Advanced Persistent Threats (APT) Market Size, Industry Outlook, Regional Analysis (U.S., Canada, Germany, France, UK, Italy, Russia, China, Japan, India, Thailand, Indonesia, Malaysia, Australia, Brazil, Mexico, Saudi Arabia, UAE, South Africa), Applicat,” 2017.

11. M. Hopkins and A. Dehghanianha, "Exploit Kits: The production line of the Cybercrime economy?," in *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, 2015, pp. 23–27.
12. A. Azmoodeh, A. Dehghanianha, and K.-K. R. Choo, "Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning," *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2018.
13. R. Brewer, "Advanced persistent threats: Minimising the damage," *Netw. Secur.*, vol. 2014, no. 4, pp. 5–9, 2014.
14. I. Friedberg, F. Skopik, G. Settanni, and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection," *Comput. Secur.*, vol. 48, pp. 35–57, 2015.
15. H. HaddadPajouh, A. Dehghanianha, R. Khayami, and K.-K. R. Choo, "A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting," *Futur. Gener. Comput. Syst.*, 2018.
16. J. Chen, C. Su, K.-H. Yeh, and M. Yung, "Special Issue on Advanced Persistent Threat," *Futur. Gener. Comput. Syst.*, vol. 79, pp. 243–246, 2018.
17. H. H. Pajouh, A. Dehghanianha, R. Khayami, and K.-K. R. Choo, "Intelligent OS X malware threat detection with code inspection," *J. Comput. Virol. Hacking Tech.*, 2017.
18. C. Tankard, "Advanced Persistent threats and how to monitor and deter them," *Netw. Secur.*, vol. 2011, no. 8, pp. 16–19, Aug. 2011.
19. A. Azmoodeh, A. Dehghanianha, M. Conti, and K.-K. R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *J. Ambient Intell. Humaniz. Comput.*, pp. 1–12, Aug. 2017.
20. A. Greenberg, "The Zero-Day Salesmen," *Forbes*, vol. 189, no. 6, pp. 40–44, 2012.
21. M. Petraityte, A. Dehghanianha, and G. Epiphanou, "A Model for Android and iOS Applications Risk Calculation: CVSS Analysis and Enhancement Using Case-Control Studies," 2018, pp. 219–237.
22. Mandiant, "APT1: Exposing One of China's Cyber Espionage Units," 2014.
23. F. N. P. Office, "Update on Sony Investigation," 2017.
24. B. Parys, "MoleRats: there's more to the naked eye," *PWC Blogs*, 2016.
25. M. Conti, A. Dehghanianha, K. Franke, and S. Watson, "Internet of Things Security and Forensics: Challenges and Opportunities," *Futur. Gener. Comput. Syst.*, Jul. 2017.
26. J. Baldwin, O. M. K. Alhwai, S. Shaughnessy, A. Akinbi, and A. Dehghanianha, "Emerging from the Cloud: A Bibliometric Analysis of Cloud Forensics Studies," 2018, pp. 311–331.
27. A. Cook, H. Janicke, R. Smith, and L. Maglaras, "The industrial control system cyber defence triage process," *Comput. Secur.*, vol. 70, pp. 467–481, Sep. 2017.
28. M. Marchetti, F. Pierazzi, M. Colajanni, and A. Guido, "Analysis of high volumes of network traffic for Advanced Persistent Threat detection," *Comput. Networks*, vol. 109, pp. 127–141, Nov. 2016.
29. K. A. Ismail, M. M. Singh, N. Mustaffa, P. Keikhosroki, and Z. Zulkefli, "Security Strategies for Hindering Watering Hole Cyber Crime Attack," *Procedia Comput. Sci.*, vol. 124, pp. 656–663, 2017.
30. S. Caltagirone, A. Pendegast, and C. Betz, "The Diamond Model of Intrusion Analysis," *Threat Connect*, vol. 298, no. 0704, pp. 1–61, 2013.
31. E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains."
32. U.S. Department of Justice, "U.S. Charges Five Chinese Military Hackers For Cyber Espionage Against U.S. Corporations And A Labor Organization For Commercial Advantage," 2014.
33. S. Fagerland, "Systematic cyber attacks against Israeli and Palestinian targets going on for a year," 2012.
34. T. Dahms, "Molerats, Here for Spring!," 2014.
35. ClearSky, "Operation DustySky," 2016.
36. R. Sherstobitoff and I. Liba, "Dissecting Operation Troy: Cyberespionage in South Korea," 2013.

37. D. Tarakanov, “The ‘Kimsuky’ Operation: A North Korean APT?,” 2013.
38. Fast Horizon, “Inside an APT Covert Communications Channel,” 2011.
39. K. Wilhoit, “The SCADA That Didn’t Cry Wolf,” 2013.
40. S. Narang, “Backdoor.Barkiofork Targets Aerospace and Defense Industry,” *Symantec Official Blog*, 2013.
41. N. M. Nart Villeneuve, Thoufique Haq, “Operation Molerats: Middle East Cyber Attacks Using Poison Ivy,” *FireEye*, 2013.
42. RBS, “A Breakdown and Analysis of the December, 2014 Sony Hack,” *RiskBasedSecurity*, 2014.
43. J. Bort, “How The Hackers Broke Into Sony And Why It Could Happen To Any Company,” *Business Insider UK*, 2014.
44. P. Brown, J. Scuitto, E. Perez, E. Bradner, and J. Acosta, “Investigators think hackers stole Sony passwords,” *CNN Politics*, 2014.
45. S. Gallagher, “Inside the ‘wiper’ malware that brought Sony Pictures to its knees [Update],” *ars Technica*, 2014.
46. S. Walker-Roberts, M. Hammoudeh, and A. Dehghantanha, “A Systematic Review of the Availability and Efficacy of Countermeasures to Internal Threats in Healthcare Critical Infrastructure,” *IEEE Access*, pp. 1–1, 2018.
47. A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, *Machine learning aided static malware analysis: A survey and tutorial*, vol. 70. 2018.
48. Y.-Y. Teing, A. Dehghantanha, K. Choo, M. T. Abdullah, and Z. Muda, “Greening Cloud-Enabled Big Data Storage Forensics: Syncany as a Case Study,” *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2017.
49. Y.-Y. Teing, A. Dehghantanha, and K.-K. R. Choo, “CloudMe forensics: A case of big data forensic investigation,” *Concurr. Comput.*, 2017.
50. N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, “Machine learning aided Android malware classification,” *Comput. Electr. Eng.*, vol. 61, 2017.
51. O. M. K. Alhwai, J. Baldwin, and A. Dehghantanha, *Leveraging machine learning techniques for windows ransomware network traffic detection*, vol. 70. 2018.
52. J. Baldwin and A. Dehghantanha, *Leveraging support vector machine for opcode density based detection of crypto-ransomware*, vol. 70. 2018.
53. S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha, and R. Khayami, “BoTShark: A Deep Learning Approach for Botnet Traffic Detection,” 2018, pp. 137–153.
54. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, “Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2016, no. 1, p. 130, May 2016.

Analysis of APT Actors Targeting IoT and Big Data Systems: Shell_Crew, NetTraveler, ProjectSauron, CopyKittens, Volatile Cedar and Transparent Tribe as a Case Study



Paul J. Taylor, Tooska Dargahi, and Ali Dehghantanha

Abstract Advanced Persistent Threats (APTs) can repeatedly threaten individuals, organisations and national targets, utilising varying tactics and methods to achieve their objectives. This study looks at six such threat groups, namely Shell_Crew, NetTraveler, ProjectSauron, CopyKittens, Volatile Cedar and Transparent Tribe, examines the methods used by each to traverse the cyber kill chain and highlights the array of capabilities that could be employed by adversary targets. Consideration for mitigation and active defence was then made with a view to preventing the effectiveness of the malicious campaigns. The study found that despite the complex nature of some adversaries, often straightforward methods could be employed at various levels in a networked environment to detract from the ability presented by some of the known threats.

Keywords Advanced persistent threat · APT · CKC · Cyber kill chain · IoT · Big data

P. J. Taylor

School of Computing, Science and Engineering, University of Salford, Manchester, UK
e-mail: Paul.Taylor_Titan@titan.police.uk

T. Dargahi

Department of Computer Science, School of Computing, Science and Engineering, University of Salford, Manchester, UK
e-mail: T.Dargahi@Salford.ac.uk

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada
e-mail: ali@cybersciencelab.org

1 Introduction

An Advanced Persistent Threat (APT) is one that persistently targets an individual, organisation, business or government in order to exploit the victim network and steal data [1]. A single APT can utilise more than one particular malware and will always be specific and targeted in its adversarial methods [2]. Often the capabilities of such groups are sufficient to achieve set objectives against a variety of hardened targets [3, 4]. The adversarial methods can evolve and react in ways that defeat some defensive systems [5] and as such there is no ‘silver bullet’ that will allow a target to prevent themselves from becoming a victim when targeted by such a motivated, capable and persistent attacker [6].

The Cyber Kill Chain (CKC) [7] is a process map that describes a series of steps, or phases, that a malicious actor, or APT, must work through to fully execute their intended action against a target. For the purposes of this study we make use of the widely used kill chain model described by Lockheed Martin Corporation [7]. Using the CKC will allow for mapping of adversarial actions by the APT at each step along the process, from initial reconnaissance against the target through to the final objectives being met [8]. In addition, the breaking down of the phases throughout which an adversary has to traverse allows for consideration of mitigation and prevention of attacks at multiple stages of the attack [9] and it helps to build forensics mechanisms for investigating APT actors [10].

The CKC can be mapped out in a two dimensional table against actionable defence tactics, tools and techniques in a course of action matrix [7]. The matrix will consider how an attack could be detected, denied, disrupted, degraded, deceived and destroyed at each step along the kill chain using available target capabilities.

Those who are responsible within an organisation for cyber security defence have a responsibility to not only protect the network but also mitigate as best they can against future threats [11]. In a landscape that includes advanced, capable adversaries targeting highly valued assets the risk of compromise is increased; with increased risk of compromise combined with the potential for compromise of valuable and often sensitive data, defenders must employ all available tactics to prevent the worse from happening [12].

Private and public researchers and cyber security product vendors produce report regularly on identified attacks, which provides useful information that can feed into existing threat intelligence models within an organisation [13]. When faced with multiple adversaries and the distinct potential for harm along with a vast amount of data feeding into threat intelligence systems, defenders need a way to focus on the most pertinent data that can assist and bolster cyber defensive capabilities in a timely and efficient way [14].

This study focuses on six known and reported APT groups and seeks to map out the methods used by these groups to attack their targets and provide advice as to how to defend against the same. The techniques used in this study combine elements of the CKC [7] and the Diamond Model [15] in a way that could be easily transferable to help defenders map out the activities of these cyber adversaries and highlight the

best response to such threats. The course of actions matrices that are produced in this study or by those adopting these methods may allow for timely consideration of available tactical options to defend a network. These may assist in evaluating the most appropriate use of resources in an organisation such as staff, facilities and finance.

The APTs analysed in this study are briefly described as follows:

Shell_Crew

Shell_Crew is also known by the names Deep Panda, WebMasters, KungFu Kittens, SportsFan, PinkPanther and Black Vine [16]. The RSA Incident Response team published a full analysis report in 2014 [17], and the threat actors were alleged to originate from China following a data breach at Anthem Insurance Company.

The APT exploited victims via web shells and vulnerabilities in web applications, making alterations to Dynamic Link Libraries (DLLs) for persistence. It utilised commonly available tools such as *Mimkatz* to operate.

The FBI reported on the Anthem breach and provided a list of tools used to aid in detection. The threat actors made use of Virtual Private Networks (VPNs) and compromised servers to obfuscate the traffic origins of the malware.

NetTraveler

Although not openly affiliated to China, Kaspersky Labs are said to have reported that members of the NetTraveler group spoke Chinese [16]. The main published researchers into this group are Kaspersky Labs, who published a report in 2011 through their Global Research and Analysis Teams [18].

NetTraveler infects its victims following execution of a Microsoft Office attachment in a spear phishing e-mail, which then allows attackers to steal private information and keylogs.

ProjectSauron

Symantec reported on ProjectSauron in 2016 [19], which they described as attacking specific targets in countries such as Russia, China and Sweden. This APT mostly used a backdoor tool named *Remsec*, which was capable of allowing adversaries to gain full access to target networks upon successful exploitation. Where required, additional functionality was delivered through the use of custom Lua modules.

CopyKittens

The CopyKittens APT were described by ClearSky Cyber Security in 2015 [20] due to their usage of other publicly available code to produce their ever customisable Remote Access Trojan (RAT) tools. The threat was targeted towards Israeli officials and constantly developed to maintain stealth and overcome the threat of detection from security products.

Volatile Cedar

This APT group was mostly reported on by Check Point Software Technologies in 2015 [21], which described the use of a RAT named *Explosive*. The malware evolved after detection incidents, maintained periods of silence following infection and showed no signs of being used for financial gain. Check Point report that

the group could have been operating for up to 3 years prior to their report and attribute some of the success to the unique way in which the malware spread and even jumped between machines via offline USB devices. Attribution was levelled towards Lebanon and a further research paper [22] indicated that the majority of victim Internet Service Providers (ISPs) also resided in Lebanon.

Transparent Tribe

This campaign involved a targeted attack against military personnel and Indian diplomats within Saudi and Kazakhstan embassies, as reported at the time by Huss in 2016 [23]. The group utilised phishing e-mails to download the malware onto the target device in a two-step process. Further work reported by FireEye [24] suggested that the group originated from Pakistan and used the same shellcode for Microsoft Word exploitation against a number of different attacks.

This paper aims to dissect these six APTs and map their activities across the Cyber Kill Chain, with the result of providing some form of defence mechanism. This will be presented in the form of a course of action matrix with accompanying recommendations.

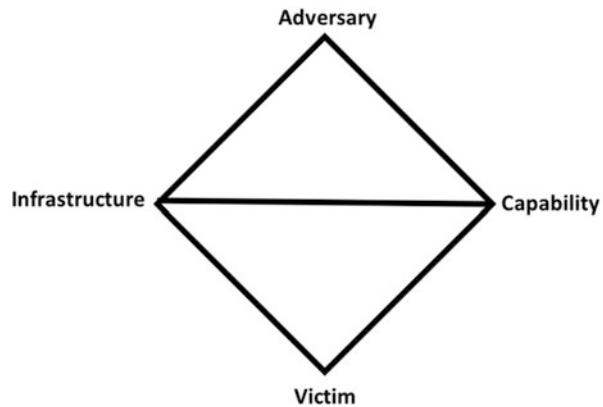
The remainder of this paper is structured as follows; Sect. 2 discusses how the attack behaviour will be presented for each of the six APT groups. Section 3 provides information in relation to the course of action matrix. Section 4 shows the attack activity graphs for all of the APT's and Diamond Model Analysis. Section 5 provides the course of action matrix for each, and finally Sect. 6 concludes this study.

2 Attack Behaviour Modelling

There are several options to choose from when deciding how to model attack behaviour and the methods chosen for this study are the Diamond Model [15] and the Cyber Kill Chain (CKC) [7]. The Diamond Model provides a relatively simplistic way of representing an attack by presenting four parts of the whole picture and the CKC is a well established technique for mapping out cyber threats and has been used in other areas such as by the US Military. The two modelling styles can be overlaid in order to present a more comprehensive visual representation of the activity taking place by the malicious attackers, as was demonstrated by Cook et al. [25].

The Diamond Model focusses on not only the motivations of an attacker for targeting a particular victim, but also the steps taken to succeed in the attack. The four vertices of the diamond are adversary, victim, capability and infrastructure, as shown in Fig. 1. The flow of an attack can be expressed in a Diamond Model by analysing the relationship between each component; what is known about the adversary's identity, location, system and how does this relate to their infrastructure and capability? What is known about the victim's position in an organisation, how they were targeted and what action they took and how does that relate to their infrastructure and capability?

Fig. 1 Components of a Diamond model



The Cyber Kill Chain [7] consists of the following phases:
Pre-compromise

1. **Reconnaissance**—Research undertaken by the attacker in order to gather information about the victim organisation, systems and people. Prevention of risk related material being published or made available in public can prevent the attacker from acquiring harmful levels of information.
2. **Weaponisation**—Preparation of the malicious code ready for deployment by the attackers. Groups of attackers can be identified at this stage by their use of specific code embedding techniques, thus providing detection opportunities.
3. **Delivery**—Sending the malicious payload to the victim either through internet services or via the transfer of data from hardware media devices.

Compromise

1. **Exploitation**—The execution of the payload against the victim through the use of a known or zero day vulnerability.
2. **Installation**—The attacker installs a backdoor to allow for repeated re-entry to the target system and gains persistence.

Post-compromise

1. **Command and Control**—Persistence is gained by the attacker to allow for access to the system.
2. **Actions on Intent**—The attacker sets out on a course of conduct to achieve their goal of theft of data, encryption of files or destruction of files.

3 Mapping Actionable Intelligence and Defensive Capabilities

The identified attack vectors along the kill chain for each APT were identified and consideration was then given to attack mitigation and prevention. For each APT, the stages of the kill chain were mapped out against indicators of compromise, tactics, techniques and procedures that would facilitate a sufficient response by the victim in the following ways:

1. **Detect**—The ability to discover the threat through analysis of known malicious activity and comparison against empirical data available in logs.
2. **Deny**—The deployment of tools and agents to prevent the compromise.
3. **Disrupt**—The application of techniques to hinder the efforts of the malicious actor and interfere with their desired purpose.
4. **Degradate**—The reduction in speed of the attackers due to some action taken by the victim to slow the effectiveness of any planned activity.
5. **Deceive**—The purposeful use of deception to mislead the attackers and cause them to make incorrect decisions at their detriment.
6. **Destroy**—The offensive response by the victim against the attacker.

The mapping out of defensive factors against the kill chain allowed for the production of a kill chain course of action matrix for each APT. The produced set of course of action matrices highlights the defensive actions that can be taken and provides a model for actionable intelligence as described in [7].

4 Attack Activity Graphs

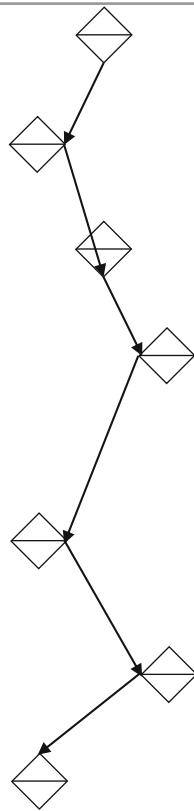
The six APT groups examined in this study are mapped against the Cyber Kill Chain with accompanying Diamond Model analysis shown below as attack activity graphs in Tables 1, 2, 3, 4, 5, and 6.

5 Course of Action Matrices

5.1 *Shell_Crew*

Table 7 shows the course of action matrix for the *Shell_Crew* APT. Due to the dependency of the adversary on known web application vulnerabilities there are opportunities to prevent a successful attack by focussing on detection and denial tactics [27]. Should an attack take place a productive incident response could feed intelligence back in to the detection systems.

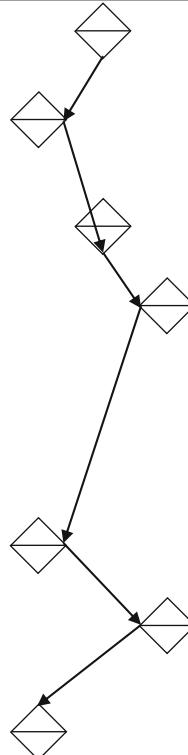
Table 1 Attack activity graph for Shell_Crew APT

Shell_Crew [17]		Diamond analysis
CKC PHASE	Event	
Reconnaissance	Scan for web application vulnerabilities, for example Adobe ColdFusion directory traversal vulnerability (CVE-2010-2861)	
Weaponisation	[v2c] Prepare or obtain currently existing exploit of known web application vulnerability identified from victim reconnaissance	
Delivery	[c2v] Perform capable exploit against known web application vulnerability and acquire admin credentials	
Exploitation	[v2i] Utilise credentials obtained through exploit (such as <i>password.properties</i> file from ColdFusion webserver) to create scheduled tasks to download further malicious tools including web shells and Trojans and use the victim's own infrastructure against them	
Installation	[i2c] Install malicious tools in multiple locations to maintain persistence, for example inject malicious DLL file into <i>lsass.exe</i> process for harvesting credentials	
C2	[c2i] Implement Remote Desktop Protocol to allow for backdoor access and create HTTP proxy on port 666	
Actions—Intent	[i2a] Gather trade secrets, password hashes and other enterprise data and send to target adversary address in a password protected Zip file over the internet	

5.2 NetTraveler

Table 8 shows the course of action matrix for the NetTraveler APT. Known vulnerabilities are exploited through the successful delivery of spear phishing e-mails, which means that a strong defence must include good patching policies and where possible, staff training and increased awareness of the threat of phishing e-mails.

Table 2 Attack activity graph for NetTraveler APT

NetTraveler [18]		Diamond analysis
CKC PHASE	Event	
Reconnaissance	Determine the most appropriate construction for a spear-phishing e-mail	
Weaponisation	[v2c] Prepare Office documents for deployment via spear-phishing e-mails with known exploits CVE-2012-0158 and CVE-2010-3333	
Delivery	[c2v] Send spear-phishing e-mail to target identified in the reconnaissance phase	
Exploitation	[v2i] Depending on exploit used, drop malicious files in the following directories: >%temp% >%windir% >%AppData%\Adobe >%windir%\System >%windir%\System32	
Installation	[i2c] Execution in most cases of netmgr.exe and winlogon.exe files	
C2	[c2i] Maintain command and control through approximately 40 maliciously registered domain names	
Actions—Intent	[i2a] Encode and exfiltrate data of previously specified file type	

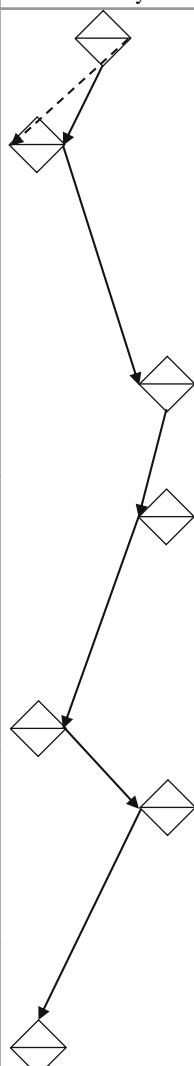
5.3 ProjectSauron

Table 9 shows the course of action matrix for the ProjectSauron APT. This advanced APT exploit the very thing that most network administrators rely upon to provide security; patching. This creates difficulties for detection so in order to have the best defensive strategy, care must be taken in validating the correct files required for software updates through a file hash comparison to published, non-malicious updates.

5.4 CopyKittens

Table 10 shows the course of action matrix for the CopyKittens APT. Specifically relating to this APT, there is a reliance on the *Matryoshka* Remote Access Trojan

Table 3 Attack activity graph for ProjectSauron APT

ProjectSauron [19, 26]		Diamond analysis
CKC PHASE	Event	
Reconnaissance	Identify system administrators and the software they use in order to identify the target software updates to be modified	
Weaponisation	<p>[v2c] Modification of existing software update scripts to incorporate a small, malicious payload with a downloader for the malware</p> <p>[c2i] Air Gap Attacks: USB memory devices are formatted with a special partition containing a virtual filesystem, which won't be recognised by common operating systems such as Windows</p>	
Delivery	[c2i] The downloader is delivered to the victim machine along with the software update the victim chose to download from the internet or install from a USB memory device	
Exploitation	<p>[v2i] The ProjectSauron payload is downloaded from a hard-coded IP address following execution of the initial payload under an administrator account.</p> <p>The air gap attacks require zero day exploits for the payload to traverse to the target machine</p>	
Installation	[i2c] The malware searches for encrypted network communication software information	
C2	[c2i] Direct communication with C2 occurs in addition to more unique methods involving DNS tunnelling and e-mail	
	Direct communication to IP addresses and domains is target specific and domain reuse and selection of service providers is diverse to obfuscate indicators of compromise	
Actions—Intent	<p>[i2a] System information and network configuration data is exfiltrated through small DNS packets</p> <p>Base64 encoded data is also sent via an e-mail attachment through a hard-coded e-mail server with benign subject and body</p>	

having to run in memory, with a dependence on simple registry keys to execute this process upon reboot. Regular monitoring of significant registry changes could assist in the detection and disruption of the malware.

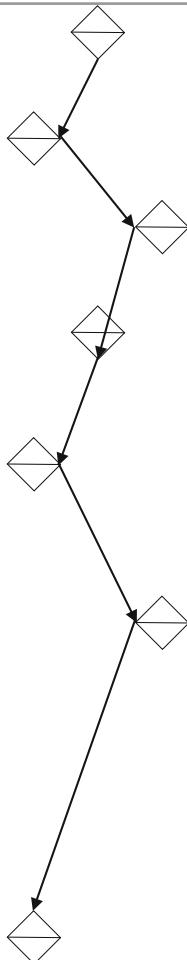
Table 4 Attack activity graph for CopyKittens APT

CopyKittens [20]		Diamond analysis
CKC PHASE	Event	
Reconnaissance	Targets are carefully researched to acquire information about an individual and his position in the government agency	
Weaponisation	[v2c] Spear phishing e-mails are prepared with subject and body that would likely be of interest to individual victims. The e-mails contain documents with malicious OLE objects	
Delivery	[c2v] The victim is enticed to run a macro within the document or execute a .scr file extension to initialise the infection process	
Exploitation	[v2i] An image file is downloaded from a remote server to signal to the adversary that the target has been reached. The malware runs checks to determine if the target machine is in a virtualised environment	
Installation	[i2v] “Matryoshka” malware is downloaded and run in memory to avoid anti-virus detection, which would be possible if stored on the disk. The loader is saved to a common Windows registry key to allow for the Matryoshka trojan to execute and run from memory again following reboot	
C2	[v2i] Communication with C&C is performed via DNS packets	
Actions—Intent	[i2a] Data including passwords, screenshots and keylogs are exfiltrated	

5.5 Volatile Cedar

Table 11 shows the course of action matrix for the Volatile Cedar APT. The attack begins with the identification and subsequent exploitation of vulnerabilities within victim web servers. If the attacks cannot be thwarted at this stage then adaptive network zone defence can be utilised to protect the more critical areas of a network; the malware is advanced at evading detection from anti-virus and the tiered structure of the C&C servers also presents difficulties for detection in the latter stages of an attack.

Table 5 Attack activity graph for Volatile Cedar APT

Volatile Cedar [21, 22]		Diamond analysis
CKC PHASE	Event	
Reconnaissance	Selection of targets is very specific and focusses on public facing web servers. Scans for known vulnerabilities	
Weaponisation	[v2c] Known vulnerability of web server is exploited with web shell	
Delivery	[c2i] Web shell allows for the downloading of the malware known as Explosive, which is key to the attack	
Exploitation	[i2v] Commands are then executed through the Explosive Trojan from an array of C&C servers, initiating key logging, screen recording and data capturing processes	
Installation	[v2c] AV detections, process memory usage and the use of a separate DLL for API activities all combine to provide the Explosive Trojan with an increased ability to avoid detection and maintain persistence	
C2	[c2i] An array of command and control servers are used and for large data exfiltration, SSH is utilised Radio silence is maintained at pre-determined times according to the working hours of the target in order to avoid detection The C&C servers are diverse and utilise a tiered system, making use of attackers own servers in addition to rented and compromised servers	
Actions—Intent	[i2a] Data exfiltration and further infection of USB devices to further propagation	

5.6 Transparent Tribe

Table 12 shows the course of action matrix for the Transparent Tribe APT. The specificity of the spear phishing e-mails likely results in a greater success rate for the adversary. To counter this efficient and targeted threat intelligence could be used to mitigate the information available in the public space. As ever with spear-phishing e-mails that successfully make it to an individual's inbox, the security of the network could rely on one individual's ability to detect and report a suspicious e-mail; training is essential in such cases to reduce the available attack surface.

Table 6 Attack activity graph for TransparentTribe APT

Transparent Tribe [23]		Diamond analysis
CKC PHASE	Event	
Reconnaissance	Very selective targeting of Indian embassies and military The adversary also posts blogs with stories targeting the interests of Indian military personnel	
Weaponisation	[c2v] Phishing e-mails and documents hosted in blogs are prepared for sending with attachment “ <i>Harrasment (sic) Case Shakantula.doc</i> ” containing exploit CVE-2012-0158	
Delivery	[v2c] Phishing e-mail is sent and the exploit is run by the pre-selected victim, which launches a downloader	
Exploitation	[c2i] The downloader is part of the malware MSIL/Crimson, which downloads a remote access trojan to the victim machine	
Installation	[i2c] The MSIL/Crimson Trojan is executed and allows for up to 40 commands to be run by the attacker, including the ability to download further malware to maintain persistence	
C2	[c2i] A variety of C&C servers are utilised depending on the variant of MSIL/Crimson	
Actions—Intent	[i2a] Data exfiltration takes place upon adversarial instruction via the trojan	

Table 7 Course of action matrix for Shell_Crew APT

CKC PHASE	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	NIDS	Patch				
Weaponisation	Threat intelligence					
Delivery	AV	AV				
Exploitation	HIDS	Application whitelisting				
Installation	Audit Log	Privilege separation	Inline AV			
C2	NIDS		Router ACLs	Queuing		
Actions—Intent	Log monitoring	Secure password			Honeypot	Incident response

Table 8 Course of action matrix for NetTraveler APT

CKC PHASE	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance						
Weaponisation	Threat intelligence	Patch				
Delivery			Staff training			
Exploitation	HIDS					
Installation	HIDS	Privilege separation Application whitelisting	Endpoint malware protection			
C2	NIDS				DNS redirect	
Actions—Intent		Firewall ACL				Adaptive zone defence

Table 9 Course of action matrix for ProjectSauron APT

CKC PHASE	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	Threat intelligence					
Weaponisation		File hash comparison				
Delivery						Adaptive zone defence
Exploitation		Firewall ACL Privilege separation				
Installation	NIDS	Encryption	Endpoint malware protection			
C2	Log monitoring			Queuing		DNS sinkholes
Actions—Intent	Log monitoring	Encryption				DNS sinkholes

6 Conclusion and Future Works

This paper demonstrates that there can be significant variability between the capabilities of different adversarial and persistent threat actors. The thorough research available in publications and reports in conjunction with ongoing reporting of infections assists the security community in determining exactly how these groups operate. By carefully considering activities against the CKC, the malicious processes can be mapped out for the aim of mitigation of risk through course of

Table 10 Course of action matrix for CopyKittens APT

CKC PHASE	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	Threat intelligence					
Weaponisation				Limit available open source Intel		
Delivery		Application whitelisting	Staff training			
Exploitation	NIDS	Firewall ACL				
Installation	Log Monitoring		Inline AV			
C2	NIDS				DNS sinkholes	
Actions—Intent		Encryption				

Table 11 Course of action matrix for Volatile Cedar APT

CKC PHASE	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	NIDS					
Weaponisation		Threat intelligence				
Delivery		Patch			Honeypot	
Exploitation	Log monitoring		Inline AV			
Installation		Patch	Inline AV			Adaptive zone defence
C2	NIDS			Quality of service		
Actions—Intent		Encryption				

action matrices. The Diamond Model allows a process by which each stage of the CKC can be considered from the perspective of the victim, adversary and their respective capabilities and infrastructure, further supporting the analytical success in deconstructing the activities of advanced persistent threats.

The course of action matrices in this study can provide a map for defending a network that could be vulnerable to such attacks. The commonalities between the matrices, particularly in the *detect* and *deny* stages indicate that even when cyber threat intelligence fails to identify a specific adversary, a potential victim can use the tools and tactics set out in the matrices to adopt a generalised defensive plan.

Certainly, the more that can be done to detect an adversary throughout any stage of the CKC, the more opportunities there are to deny the adversary what is sought by them to complete their objectives [28]. If detection and denial of an adversary's

Table 12 Course of action matrix for transparent tribe APT

CKC PHASE	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	Threat intelligence				Honey pot	
Weaponisation		Patch				
Delivery			Staff training			
Exploitation	HIDS					
Installation	NIDS		Inline AV			
C2				Quality of service		
Actions—Intent		Encryption				

activities is not capable or overlooked then the success of that adversary is far more likely, as can be seen in the matrices by the reduced opportunities to disrupt, degrade, deceive and contain the threat [4].

Future works may include analysis of APT groups based on their tools of attacks i.e. if they are relying on classical malware [29], ransomware [30], or Botnets [31]. Moreover, it is interesting to offer forensics taxonomies and investigation guidelines such as those suggested by [32, 33] for different APT groups.

References

1. M. Hopkins and A. Dehghantanha, “Exploit Kits: The production line of the Cybercrime economy?,” in *2015 2nd International Conference on Information Security and Cyber Forensics, InfoSec 2015*, 2016.
2. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, “Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence,” *IEEE Trans. Emerg. Top. Comput.*, 2017.
3. S. Walker-Roberts, M. Hammoudeh, and A. Dehghantanha, “A Systematic Review of the Availability and Efficacy of Countermeasures to Internal Threats in Healthcare Critical Infrastructure,” *IEEE Access*, 2018.
4. H. Haddad Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo, “A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks,” *IEEE Trans. Emerg. Top. Comput.*, pp. 1–1, 2016.
5. N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, “Machine learning aided Android malware classification,” *Comput. Electr. Eng.*, vol. 61, 2017.
6. A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, “Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning,” *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2018.
7. E. M. Hutchins, M. J. Cloppert, and R. M. Amin, “Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains.”
8. D. Kiwia, A. Dehghantanha, K.-K. R. Choo, and J. Slaughter, “A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence,” *J. Comput. Sci.*, Nov. 2017.

9. H. Haddadpajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, “A Deep Recurrent Neural Network Based Approach for Internet of Things Malware Threat Hunting,” *Futur. Gener. Comput. Syst.*, 2018.
10. S. Watson and A. Dehghantanha, “Digital forensics: the missing piece of the Internet of Things promise,” *Comput. Fraud Secur.*, vol. 2016, no. 6, 2016.
11. M. Conti, A. Dehghantanha, K. Franke, and S. Watson, “Internet of Things Security and Forensics: Challenges and Opportunities,” *Futur. Gener. Comput. Syst.*, Jul. 2017.
12. H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, “Intelligent OS X malware threat detection with code inspection,” *J. Comput. Virol. Hacking Tech.*, 2017.
13. M. Petraityte, A. Dehghantanha, and G. Epiphanou, “A Model for Android and iOS Applications Risk Calculation: CVSS Analysis and Enhancement Using Case-Control Studies,” 2018, pp. 219–237.
14. H. Haughey, G. Epiphanou, H. Al-Khateeb, and A. Dehghantanha, *Adaptive traffic fingerprinting for darknet threat intelligence*, vol. 70. 2018.
15. S. Caltagirone, A. Pendegast, and C. Betz, “The Diamond Model of Intrusion Analysis,” *Threat Connect.*, vol. 298, no. 0704, pp. 1–61, 2013.
16. A. Lemay, J. Calvet, F. Menet, and J. M. Fernandez, “Survey of publicly available reports on advanced persistent threat actors,” *Comput. Secur.*, vol. 72, pp. 26–59, Jan. 2018.
17. EMC/RSA, “RSA Incident Response - Emerging Threat Profile: Shell Crew,” no. January, pp. 1–42, 2014.
18. Kaspersky, “The NetTraveler (aka ‘Travnet’),” 2004.
19. S. Response and S. Page, “Security Response Backdoor . Remsec indicators of compromise,” pp. 1–13, 2016.
20. Clearsky, “CopyKittens Attack Group,” *Minerva Labs LTD Clear. Cyber Secur.*, no. Nov, pp. 1–23, 2015.
21. T. Intelligence, “Volatile cedar,” 2015.
22. B. K. Baumgartner, “Cedar DGA Infrastructure Statistics ;,” pp. 2–6, 2015.
23. D. Huss, “Operation Transparent Tribe - Threat Insight,” 2016.
24. Y. H. Chang and Singh Sudeep, “APT Group Sends Spear Phishing Emails to Indian Government Officials « APT Group Sends Spear Phishing Emails to Indian Government Officials | FireEye Inc,” *FireEye*, 2016.
25. A. Cook, H. Janicke, R. Smith, and L. Maglaras, “The industrial control system cyber defence triage process,” *Comput. Secur.*, vol. 70, pp. 467–481, Sep. 2017.
26. Global Research and Analysis Team, “The ProjectSauron APT,” *Kaspersky Lab*, vol. 02, pp. 1–23, 2016.
27. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, “Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing,” *Eurasip J. Wirel. Commun. Netw.*, vol. 2016, no. 1, 2016.
28. A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, “Detecting crypto-ransomware in IoT networks based on energy consumption footprint,” *J. Ambient Intell. Humaniz. Comput.*, pp. 1–12, Aug. 2017.
29. A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, *Machine learning aided static malware analysis: A survey and tutorial*, vol. 70. 2018.
30. O. M. K. Alhawi, J. Baldwin, and A. Dehghantanha, “Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection,” 2018, pp. 93–106.
31. S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha, and R. Khayami, “BoTShark: A Deep Learning Approach for Botnet Traffic Detection,” Springer, Cham, 2018, pp. 137–153.
32. J. Gill, I. Okere, H. HaddadPajouh, and A. Dehghantanha, *Mobile forensics: A bibliometric analysis*, vol. 70. 2018.
33. A. A. James Baldwin, Omar Alhawi, Simone Shaughnessy and A. Dehghantanha, *Emerging from The Cloud: A Bibliometric Analysis of Cloud Forensics Studies*. Cyber Threat Intelligence- Springer Book, 2017.

A Cyber Kill Chain Based Analysis of Remote Access Trojans



Reyhaneh HosseiniNejad, Hamed HaddadPajouh, Ali DehghanTanhah, and Reza M. Parizi

Abstract Computer networks and industrial systems are always under cyber threat and attack. Existing vulnerabilities in different parts of systems have given cyber attackers the opportunity to think about attacking, damaging or hindering the working process of important infrastructures of the country. Figuring out these threats and weak points which are used by malwares like Trojans, considering the evolution of used techniques for preventing identification and ways to identify, is a big challenge. Having a destructive hierarchy can help identification and risk mitigation strategies. In this paper, we have analyzed a hierarchy based on characteristics of remote-controlled malwares using 477 Trojans collected from real-world samples, using different methods of assessment. The carried out analysis used one of the popular models for identifying cyber threats named Cyber Kill Chain. We proposed a hierarchy based on dataset sample in different stage of malware lifecycle.

Keywords Cyber kill chain · Trojan · RAT · Threat intelligence · Threat hunting

1 Introduction

With the ever expanding new technologies and internet finding its way in every aspect of people's lives, the stage is set for infiltration and contamination of different systems with different intentions [1]. Malwares are computer programs with the

R. HosseiniNejad
Pishtazan Higher Education Institute, Shiraz, Iran

H. HaddadPajouh · A. DehghanTanhah (✉)
Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada
e-mail: hamed@cybersciencelab.org; ali@cybersciencelab.org

R. M. Parizi
Department of Software Engineering and Game Development, Kennesaw State University,
Marietta, GA, USA

goal of infecting computer systems [2]. Not only malwares attack and get access to system resources, information, damage the service quality, and operations, but also by stealing personal information illegally [3]. Malware may impose huge psychological impacts, beside the financial losses as well [4]. In the past two decades' malware, detection issue became as one of the hottest challenges in cyber security research domain. Developing malwares with is increasing exponentially and there needs to be more emphasis on confronting this issue and more accurate and efficient methods should be proposed [5]. These kinds of software have always been and will always be talked about in the computer and the network sciences [6]. Malware has always been one of the main problems of system and network security managers [7]. The most important defense against malwares are anti malware and anti-virus software [8]. When it comes to spying malwares those which hide themselves, in a way that even anti malwares are not be able to detect them [9]. Those malwares transmit vital information from the victim to base [10]. These days' malwares are a lot more active and are a lot smarter. Thus, some anti-malware [11] programs may not be able to identify and clean them first [12]. Although advancements in the anti-malware technologies have made the risk of that a lot less, still the absence of software to analyses and identify the behavior and activities can be felt [13].

Malwares in general are software with malicious behavior, created with different goals like collecting vital information, accessing personal computers, and in some situations destroying systems.

Developing malwares with the goal of destruction, is growing every day [14]. All the heuristic methods against malware point that there should be some action taking place against this matter and more accurate and efficient methods for identifying and preventing their infiltration should be suggested [15].

Figure 1 shows a summary of malware growth over the last few years [16, 17].

Malwares will access resources and information is on a system, damage service quality and activities of the system, and by stealing personal information they will impose huge psychological impacts, beside the financial losses for instances the average of \$197 will be lost in each cyber-attack [18, 19] to people [4].

According to [11] basically identifying malwares is a complex, delicate and extremely hard task. Malware detection will the is one of the hottest issue in cyber security in last decades [19].

Malware types are usually categorized into Backdoor, Rootkit, Dropper, Key Logger, Virus, Ransomware family [7], and etc. [20].

In addition, malwares can more clearly the effects on infrastructures of computer networks, technologies and different services like Operating systems, File sharing [21], Online social networks [22], Wireless networks [23], virtual systems, internet sites and email.

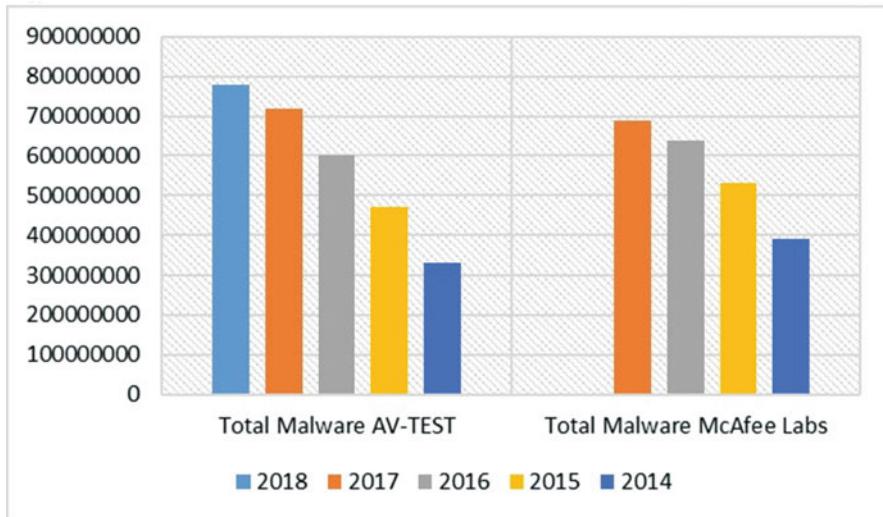


Fig. 1 Total number of unique samples include in McAfee Labs and AV-TEST's malware repository

1.1 *Remote Controlled Trojans and the Goal*

Remote Access Trojans (RAT) mean Remote Administrative Tool. RAT gives the damaging user the ability to remote control the system. RAT is used for stealing confidential information from target organization [24] and generally includes services that are listening to specific TCP/UDP ports on the victim system. Also the client plays the role of the intermediary between server and the attacker [25]. This type of tool is one of the most malfunction programs. Using RAT the hacker will enter unnoticed and will be able to control the whole system remotely and do anything like sending video files. These type of malwares are damaging fragmented codes, which are already included in legal programs and will access the system by patching games and email attachments [25]. Using RAT attacker will be able to install key logger and other programs on the victim's system and remotely infect the system and get the control off the victim's system. This management includes these matters:

- Stealing files
- searching through the personal files of the victim's system
- installing key logger in order to capture and monitor victims keyboard
- File management (download/upload/execute/etc.)
- managing and changing the registry (query/add/delete/modify)

- shell control (generally using command line)
- using victim's system for DDOS attacks
- using network bandwidth
- monitoring, stopping or starting tasks in task management
- moving and clicking the mouse accidentally
- disabling security features of the system

RATs can create processes on some ports of TCP/UDP as much as they like, shift the traffic from one system to another in order to do it Distributed Denial of Service (DDoS) attack [25]. There are three ways to attach email, USB drive, drive-by download [24, 25]. RATs generally start their process from their command server. There exist two patterns in relation to RAT: one of them is from the attacker's service receiver. Service receiver is out of target network; thus, traffic cannot easily pass through the firewall. The other one is starch from victim server. In this pattern, traffic can even pass through a proxy server [26]. Trojans open a backdoor in the system for hackers to steal information or change the system structure. Also rootkits include a series of organized tools with the goal of attacking and a specific operating system. These days' rootkits exist for most of the operating systems including Microsoft operating systems.

1.2 *Trojan Operating Platforms*

First RAT in Windows was discovered in 1998 [20]. RAT targets more than 75% of main files of Windows platform [27]; but still we are witness to the increasing number of Trojans for Linux, Mac OS X, and Android [27]. In this research, our Focus is on Windows RAT.

According to Trend Micro [28] remote controlled Trojan has a Cross platform dad can be installed on any system with Java, including windows, Mac OS X [3], Linux, iOS [27] and Android [29] and infect them. Thus anti Trojan solutions should cover most of the managed or unmanaged operating systems including personal computers, Macs and mobile Systems.

2 Related Work

Creating a hierarchy is an important tool for organizing knowledge and understanding existing problems and creating a space solution. For example, Lindorfer et al. in 2011 [30] proposed a hierarchy for damaging behavior of malwares based on identifying environmental variables. They faced issues with destructive codes and identifying tool environment used for escaping analysis. Karim et al. in 2014 [31] suggested a hierarchy for mobile phone Botnet. In the matter of mobile phone attacks. Ugarte et al. in 2015 [32] suggested a hierarchy for destructive packages

based on measuring execution time complexity and Code comparing method. Their solution was based on some hypotheses. Thus the lack of reference detail complex and lack of analyzing tool for package behavior shows that packaging issue has been put aside by research community. Gupta et al. [33] suggested the hierarchy of different Phishing attacks and how to defend against them. They were faced with limitations like the need for fast access and the need for high identification rate. While fast access solutions are suffering from low identification rate. Nawir In 2016 [34] provided the hierarchy of different security attacks in Internet of Things (IoT) [8] networks for helping researchers and developers to plan suitable security measures in advancing IoT. Alan Cook in 2017 [35] provided an integrated process named ICS_CDTP¹ with an effective triage from attack vectors and opposition capabilities for possible goals. This approach cannot predict the exact attack route. Also the attacker behavior can change based on defensive measures. Kiwia et al. in 2017 [18] suggested the first hierarchy based on Cyber Kill Chain (CKC) based on characteristics of banking Trojans which can be used to get more information about risk mitigation and identification strategies. But it is necessary to expand this hierarchy to cover all the other families.

3 Proposed Methodology

The proposed methodology in this paper for detecting RATs threats is based on threat hunting models. This approach will be useful for identifying the RATs threats in every stage of their lifecycles. In this paper Cyber Kill Chain threat hunting model is chosen to confront RATs threats in enterprises.

3.1 *Searching for Threats (Threat Hunting Models)*

Some cyber threats are known as Advanced Persistent Threat (APT), not only because of the abilities of the attacker, but also because of their ability in creating and sustaining long-term operations against a target. Security experts will not only wait for reacting to warning or compromise indicators. They are actively looking for threats in order to prevent them or mitigate the damage [36]. One of the shortcomings in common models for cyber-attack analyzation is that enterprises have no plan for the time an attacker has already infiltrated the network, this is while experts believe defensive operations should take place with precautions regarding damages. Two smart models for cyber threats which are widely used in the industry or CKC [18] and Diamond Model [35] which helps identifying infiltration and predicting in order to identify and understand how enemy fights [36]. CKC is a

¹Industrial control system cyber defense triage process.

systematic process considered for targeting and engaging enemy for creating effects and includes seven steps that are shown in Fig. 2 for [18, 37]. In each step security architectures should understand the tactics, methods and processes used by attackers in order to base on them be able to plan their methods and defensive processes.

Diamond model is in a structure for analogy that will interpret Atomic infiltration and will describe four main characteristics of the antagonist. Attackers, by using capabilities given to them by an infrastructure, in order to target the Victim, will create a result. These characteristics can be considered as knots that are connected together on the edges and the edges are the thing that describes them as shown in Fig. 3 [35].

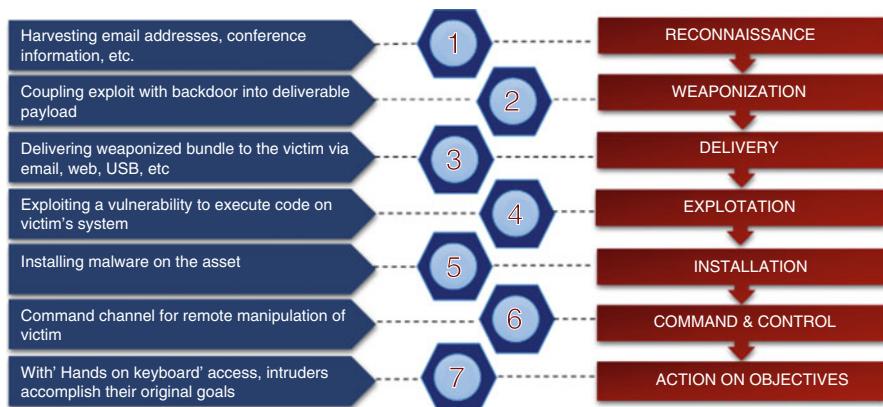


Fig. 2 Lockheed Martin CKC steps

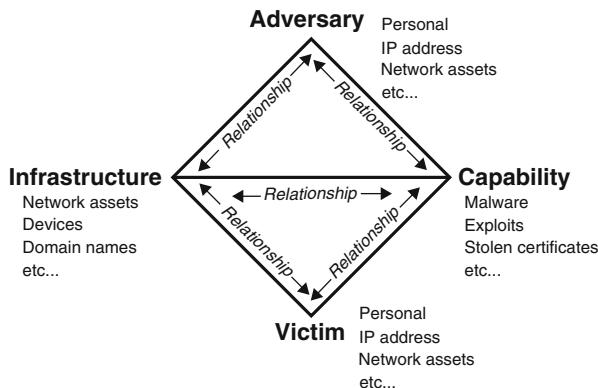


Fig. 3 Diamond model for detecting threats

Cyber attackers use different campaigns in order to get access to their target system or network. A campaign includes enough information, complete operation against organization or target network and their systems. In this paper, we intend to create a model for Remote Access Trojans (RAT). As it can be seen our proposal methodology is presented in Fig. 4. Considering that most of RATs are after infecting executable files of Microsoft Windows operating system. We have collected 477 real world RAT files for our experiments. In order to discover the behavior of malwares two methods based on a Static analysis and Dynamic analysis is used. These two techniques allow us to reach a complete and accurate understanding of details, dangers and intention of malware software.

The static method enables us to detect the malwares without executing it. This also has less computational overhead and will check all the executing routes [38]. On the other hand, we used dynamic analysis and executing the malware in cuckoo sandbox [39]. Dynamic methods try to execute malware, and by monitoring the behavior of malware and its reactions they we analyzed it. After death using the results, we tried to excavate suitable properties in order to put the malwares in a hierarchy. Some of the features listed in Appendix 1 are provided. Some of the notable ones are Desired Access, Port, Protocol, Model and API summoning which has a good knowledge of the behavior and purpose of executable files. Then based on collective properties and by using CKC model, we proposed a hierarchy based on the properties of remote controlled malwares which have been discussed in Sect. 3.3. This hierarchy of features (Appendix 2) will provide a step-by-step understanding of cyber-attacks and it will be useful for security expert and risk mitigation and identification strategy designers [18]. Identifying malware behavior is effective in preventing attacks on network, organizations and Systems.

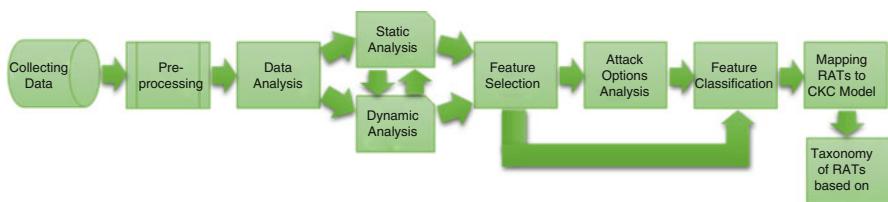


Fig. 4 Proposed RAT analysis methodology

- 4-2 Extracted feature from static analysis File version
- business name
- product name
- internal name of the file (name of the file during compiling)
- product version
- name of the file during transportation to system
- showing executed file sections including information for each section

- Name
- virtual address
- occupied space on virtual memory
- size of the raw data
- Entropy
- Showing (IAT) including
 - list of the used DLLs
 - list of the functions used for each DLL
- the ability to search any function on MSDN database
- showing the list of available Majors on file (string)

Despite of so many successes in this branch, these bunch of methods face big challenges in static analysis method, complexity of use on new and unseen malwares, because new malwares use anti problem solving mechanisms and defensive techniques to prevent code branches from being analyze [40]. The problem with static analysis based methods is that we don't always have access to code structure of malware [41]. There is also the chance of malwares encoding their code structure, which in general static analysis methods are not useful in these situations. Dynamic analysis methods are used to solve these problems.

3.2 Extracted Feature from Static Analysis

After executing the Sandbox, malicious software will probably create files and registry keys in order to keep themselves in the system, start new processes and contact control and command servers [27].

These are the results

- The ability to show processes created by malware
- The resolved domain name during analysis process (resolve is the activity that will translate the domain name into an IP address)
- network connection is established
- showing the list of summoned APIs by each process with these details:
 - API use case (accessing the network, accessing the system file, registry, services, integration, etc.)
 - API summoning time
 - API name
 - Input values and API arguments
 - output values of API
 - number of repetition
- showing changes in system file (adding new file, changing file, deleting the file)
- showing the changes in the windows registry
- showing created and used Mutexes

In addition, the results of analyzing network connection section

- Destination hosts address
- downloading the packet capture file for all exchanged packets during analysis

In addition to monitoring the aforementioned processes, such a system should also be able to collect accurate system information like the structure of current network [27].

- Name of the resolved domains
- list of HTTP(S) connections
- list of ICMP connections
- list of UDP connections
- list of TCP connections
- list of DNS connections

Other information obtained from analysis by Cuckoo Sandbox can be found in the file attributes including those listed in Table 1.

Table 1 An example of Trojan features

File Name	A7m3d-Rat V1.0.exe	Adwind RAT v3.0.jar	AceRAT-Client.exe
Size	1.1 MB	2.8 MB	440.0 KB
Type	PE32 executable (GUI) Intel 80386, for MS Windows	Java archive data (JAR)	PE32 executable (GUI) Intel 80386, for MS Windows
MD5	4451fc1ccdfa5134c5cb19 1366951972	b83082c7899204065f4b2d 79506c8773	67aaaf784a80fcdb899459 77e43a45ea5
CRC32	7BA16B81	A11E19D9	AC0F25C8
ssdeep	None	None	None
Yara	vmdetect - Possibly employs anti-virtualization techniques Advapi_Hash_API - Looks for advapi API functions RijnDael_AES - RijnDael AES	shellcode - Matched shellcode byte patterns	SEH_vba - (no description)
Screenshots	160	96	88
Identified by AntiVirus	42	30	48

- file name
- file size
- file type
- hashes including Sha512, Sha256, Sha1, Md5

- CRC file
- Hash SSDeep
- The ability to compare Yara signatures in order to identify known malwares without the need for analysis
- showing a screenshots from virtual machine during the execution of malware file
- the name of the identified malwares using anti viruses and their known signatures

3.3 Proposed Mapping Paradigm for RATs in the CKC Model

When Trojans are installed on a system not only it will weaken the information against threats, but also will give the attacker the opportunity for an attack. Generally speaking, an attack includes seven major steps [18, 20]. Our suggested hierarchy is based on these seven steps:

- **Reconnaissance:** identifying cyber-attack includes planning and researching about target using data collecting tools.
- **Weaponization:** by using Trojan making software, a package infected with Trojan will be created. This level is the biggest step for increasing successfulness of an attack and decreasing chances for identifying the attack and limiting security researcher's ability to identify and analysis the threat [18]. In general, identifying malicious codes can be put into two categories: Host-based methods and Network-based methods, as shown in Table 2 [20]. Remote controlled Trojans use these techniques.

Table 2 Five methods for detecting RATs

Categories	Subclass	Applied to RATs Detection
Host-based	Signature	Static, detect binary
	Behavior	Sandbox, dynamic detect
Network-based	Signature	Static, detect application layer
	Anomaly	Statistics of traffic flow
	Protocol	Nonstandard protocol

– Host-based evasion

Host-based Intrusion Detection System, which are based on host, is a type of infiltration identifying system that will monitor and analysis the inside of

an accounting system. For example, Packets that are being transmitted on the network mediator of that system. Host based intrusion detection system (HIDS) monitors all the dynamic situations and behaviors of the computer system. This will also indicate which programs will have access to which resources. However, attackers will use different techniques and solutions to hide themselves and install on host without being noticed [18].

Embedding malicious code

In this method, malicious codes of Trojan will be presented in the shape of an attractive file like office software [33]. When user executes this file at first children will be installed in the background without the knowledge of the user and after that the main program will be installed. Attacker is able to compress any binary code (DOS or windows) using a special tool. This tool will extract the file when being executed. This will allow children to remain undetected and most of anti-viruses are not able to identify digital signatures inside files [18, 42].

Rootkit

A compilation of software that will take control of the system. In this situation user will not notice the existence of the rootkit and hacker will have access to all the settings. Rootkits are a lot like Trojans and backdoors but they are a lot harder to be identified because they will supersede the operating system files and even sometimes they will supersede the windows kernel [3, 33].

Backdoor

Most of the time attacker will create a hidden exits so that they will be able to send, received or control the system [18, 33]. In other words, attacker will be able to trick normal security mechanisms, give himself administrator access to the system, and easily steal information and files.

– Network-based evasion

identifying and noticing illegal intrusions before it's becomes critical is on the shoulder of network based intrusion detection system (NIDS) which will investigate packets and active communication protocols [18]. Considering that NIDS is not limited to one system, it has a higher spread and it will do identification process in a dispersed manner. Thus, these systems lose their capability when facing encrypted packets or fast networks.

Attackers are able to hide among the noise of Internet traffic and expected operations [43]. Remote controlled malwares also use different techniques to do the same thing.

Port: using shared ports and sometimes a specific port; in this technique, Trojans will use common ports to do their destructive task [37]. This method will decrease the probability of identifying network ports as destructive communication.

Common Protocol: using shared ports; in this technique, Trojans will generally use protocols like HTTP(S), UDP, and TCP.

- **Delivery:** attacker will use a method to contact defender (target) and will send its destructive weight to the target environment. In general this level can be done in different ways:
 - Downloading: in this mechanism, user (with or without awareness) by using the active content like JavaScript, which is compatible with browser, will download destructive program [44]. Malicious code is embedded in an object and will force the victim to download it [18].
 - Destructive websites: victim will enter a website, it will execute a destructive program on the victim system, and the Trojan will enter.
 - Email: it is possible that the sent email have the destructive file and by clicking on it the Trojan will enter the system [18].
 - Using weaknesses in software: Trojans use weaknesses in software like browsers and enter the victim system
 - By using diskettes

In this stage, Trojans will find a way to deliver destructive weight to target space. Delivery options include special HTTP server created for Trojan diffusion using normal web pages, opening email attachments, executing destructive programs or games and will trick the user to download, repackage and finally execute such programs [25]. Remote controlled Trojans usually use three methods:

- **Email attachment:** most organizations use email communication and most of them have PDF documents or macros in Office software and these attachments have destructive codes in the shape of Macro or JavaScript [18].
- **Web links:** In this method, victim will enter the site. Site will execute a program on the system and Trojan will enter the system.
- **Downloader:** victim will download a program. Trojan is hidden inside the program and this way will enter the system.
- **Exploitation:** after delivering malicious software, by executing destructive program or by using system weakness the attacker will target the system. A successful infiltration may lead to stealing personal information, injecting malicious codes into applications, manipulating functions, etc. [18, 44]. After delivering destructive weight to the victim space, Trojans will enter target system with the goal of exploiting weaknesses or executing destructive programs.

- **Web injection:** Trojans with the ability of web injection can little by little change a web page on victims system [18]. One of web injection characteristics is as follows.

User land Rootkit: In this technique, Trojans will communicate using APIs in order to inject their code during web browser [45] start-up then use WININET.DLL in Internet Explorer loading and then by using high level communication functions like HttpQueryInfo· HttpSendRequest· InternetReadFile· InternetGetConnectState· InternetReadFileEX· InternetQueryOption etc. and by manipulating functions in API Hooking they will change user situation. Thus, a trojan can track information before they become encoded by HTTP [18, 45].

- **Protection:** granted access to a security account can be assigned in any way. One object will always inherit its granted access that is assigned to its main object, this will make security management easier but accessing, and manipulating it by attacker will be easier as well.
- **Desired access:** considering that objects that are related to files can be stored, accessing them is controlled by accessing model and accessing all secured objects in windows is controlled this way. Attacker will define a security interpreter for a file or directory when summoning Create File· Create Directory· CreateDirectoryEx. They will change access rules including Delete· Read Control Owner, Write and Synchronous.
- **Installation:** In this stage, attacker tries to access more systems and put more nodes in danger [18]. Before installation, RAT servers can be modified through structure packages provided by RAT. This modification includes default setting for TCP/UDP used by RAT server, describing auto start methods, encoding algorithms, and determining primary passwords [25].
RAT servers during installation, may introduce themselves to other legitimate programs as host, and anytime a guest requests for host, they will be summoned [25]. Attackers usually use methods that will make their discovery hard [18].
- **DLL loading**
Windows allows applications to load their DLLs if they specify the destination. Attackers exploit this and load their DLLs in a higher rank than others load. These way applications will load the destructive DLLs [18, 46].
- **Command and Control:** Controlling and coordinating the operations is done with command and control and information will be sent to C2 servers.

Table 3 User common behaviors in campus traffic flows

User behavior	Remarks	Protocol	Examples
Resource down/upload	Data transport	TCP	FTP, Thunder
Multimedia	Online game, video and more	UDP, TCP	MMS, RTSP
Website visit	Ajax	TCP, UDP	HTTP, HTTPS, DNS
Cloud storage	sync	TCP	Dropbox, Sky Drive
Remote access	Connect to special server for work	TCP	TeamViewer, SSH
P2P-like interaction	Peer-to-peer	UDP, TCP	QICQ, Skyp, BitTorrent
Mail	Mail servers, transfer agents	TCP	POP3, SMTP

This mechanism will connect malicious software to C&C server. Destructives will be registered in a C2 domain to upload or receive commands [3, 8]. Data traffic includes all the things that are moving in the network. Based on behavioral patterns and their effects, data traffic is divided into these classes: resource download, multimedia, web site visit, hyper personal storage, remote access, P2P interactions and email [20]. There exist two types of RAT connections, Direct Connection and Reverse Connection; in direct connection there will be a direct connection between attacker host and infected host and in reversed connection infected host is connected to attacker host recursively [23].

– IP address

Attacker as a domain name can connect to each IP address. Because of the rules, getting a domain name is really complicated [18].

– DNS, TCP, UDP, HTTP(S)

Attacker will use different protocols and will receive different domain names that will be visited by victim and will try to host his Trojans as indicated in Table 3, on that same domain names. Usually, attackers will create many C2 routes to make sure if one is identified, connection will remain [43].

– Action Objective

After doing all of these, attacker will follow his plan which may include disturbance for system, information speed, moving around network, installing and executing additional features and executing them.

– Dropped file

Trojans can use Backdoors to put different file types on victims system to do destructive things. Now, according to the seven basic steps mentioned, our proposed classification is shown in Fig. 5.

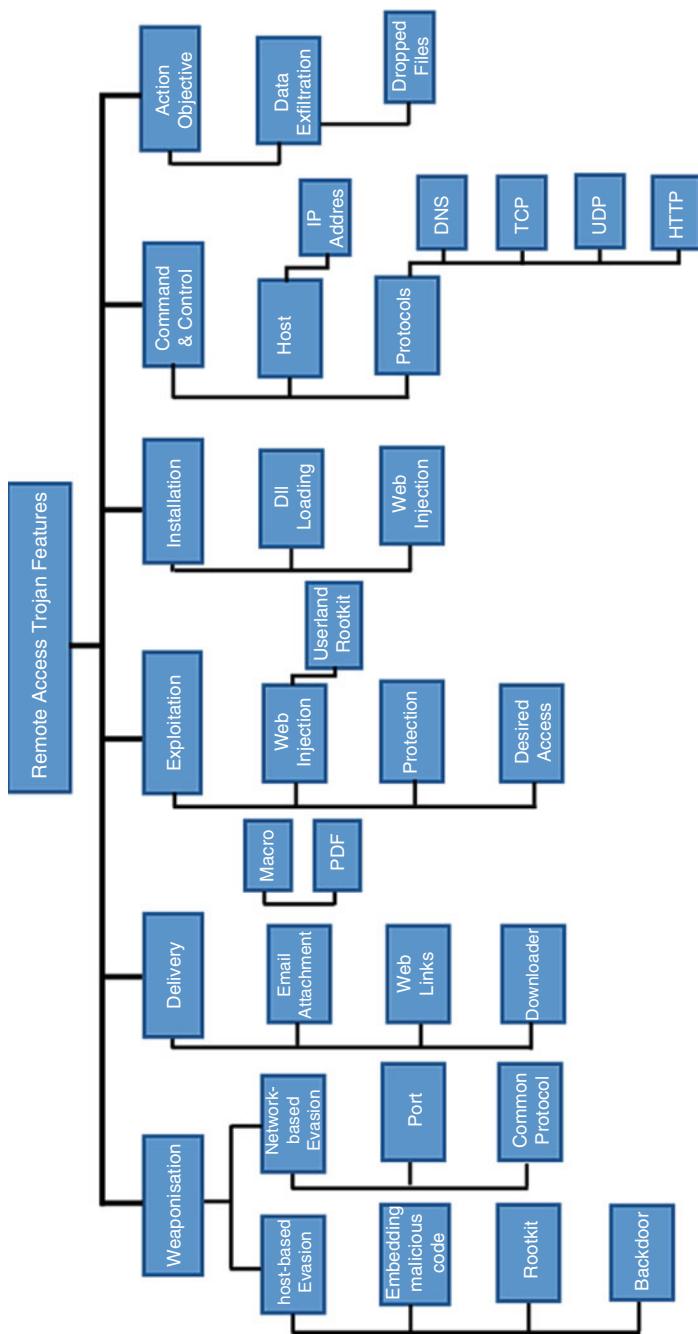


Fig. 5 Taxonomy of RAT features based on CKC

4 Conclusion

New age malwares like RATs are a challenge to discover. The structure of this type of malwares, by using different mystifying techniques, will change after each distribution, but their main function remains the same and this makes identifying them hard and complex. Malware developers put jumping codes in malwares so that in each intrusion, their signature would be different and this will prevent them from being discovered by anti-viruses and Anti-malwares software. Changes in variety, structure encoder, or including different codes is the reason for failure in discovering malwares and other cases the reason may be high calculation overflow, identification accuracy, and poor performance of method. In this paper, we proposed a novel mapping mechanism to detect RATs by their static and dynamic features based on CKC threat hunting model. Our proposed model will help to identify and classify RATs by their current actions based on their features and map into CKC working area.

Appendix 1: Some Types of Extracted Features

Name	API	Desired access	share_access	Section	Import	allocation_type	Protection	Status	Return	Port	Protocol	Socket	region_size	Flag
RAT1				.text .data .idata .rls	kernel32.dll				111					0
	GetAdaptersAddresses								1	0				
	OpenDirectoryObject								1	37				
	GetTempPathW													
	AllocateVirtualMemory					MEM_COMMIT MEM_RESERVE	READWRITE, NOACCESS	1	0					1259760
	ProtectVirtualMemory						READWRITE, READONLY, READ	1	0					
	FreeVirtualMemory								1	0				
	OpenProcess								1	0				
	CreateThread								1	232				4
	ReadProcessMemory								1	1				
RAT2				.text .data .idata .rls	kernel32.dll									
	GetDiskFreeSpaceExW								1	1				
	OpenFile					FILE_SHARE_READ FILE_SHARE_WRITE FILE_SHARE_DELETE			1	0				

(continued)

WSASStartup						1	0	564		6	564
socket						1	0	122		564	
bind						1	0			564	
listen						1	0			564	
LdrGetDIHandle						1	0				
getsockname						1	0	64208			
sendio						1	41	124		0	
shutdown						1	0			584	
closesocket						1	0			520	
CreateSection						1	0				
MapViewOfSection						1	1				
					READONLY	1	1				

Appendix 2: Remote Trojan Taxonomy

	CreateFile	(FILE_READ_ATTRIBUTES) SYNCHRONIZE GENERIC_WRITE)	FILE_SHARE_READ FILE_SHARE_WRITE)		1	0		
	NtOpenFile	SYNCHRONIZE	FILE_SHARE_READ FILE_SHARE_WRITE FILE_SHARE_DELETE					
	WSAStartup				1	0		
	socket				1	564		
	bind				1	0	122	564
	listen				1	0		564
	LdrGetDllHandle				1	0		
	getsockname				1	0	64208	
	sendto				1	41	124	0
	shutdown				1	0		584
	closesocket				1	0		520
	CreateSection				1	0		
	MapViewOfSection			READONLY 1				
RAT3		.text .sdata .rsrc .reloc	mscoree.dll					
	GetFileAttributesW				4294967295			
	FindFirstFileExW				1	6532512		

	CreateFile	FILE_READ_ATTRIBUTES READ_CONTROL SYNCHRONIZE	FILE_SHARE_READ FILE_SHARE_WRITE FILE_SHARE_DELETE FILE_SHARE_READ				1 0
	GetSystemDirectoryW					1 19	
	GetSystemWindowsDirectoryW					1 10	
	OpenDirectoryObject					1 0	
	GetFileAttributesExW					1 8212	
	ReadFile					1 0	4095
	WriteFile					1 0	
	SetFilePointer					1 73	
	DeleteFileW					1 1	
	OpenFile	READ_CONTROL	(FILE_SHARE_READ FILE_SHARE_WRITE FILE_SHARE_DELETE)			1 0	
	SetEndOfFile					1 1	
	InternetCrackUrlA					1 1	268435456
	getaddrinfo					1 0	

(continued)

InternetSet OptionA	INTERNET_OPTION_CONNECT_TIMEOUT [CONTROL_SEND TIMEOUTCONTROL_RECEIVE_TIMEOUT] MAX_CONNS_PER_SERVERERROR_MASK REQUEST_PRIORITY[CONNECTED_STATE]				1			
InternetQuery OptionA	INTERNET_OPTION_REQUEST_FLAGS				18			
InternetGetConnectedState				1	1			
getaddrinfo				1	0			
socket				1	1300	17	1300	
bind				1	0		1300	
getsockname				1	0	60951		
connect				1	0	80	60951	1300 1372
InternetOpenW				1	0			268435456
HttpOpenRequestW				1	13369356			4194304
GetAdaptersAddresses				111				0
getaddrinfo				11001				
InternetQueryOptionA	INTERNET_OPTION_CONNECTED_STATE			1	1			
send				1	1		1300	
select				1	1		1,2	
setsockopt				1	0		1372	

	GetSystemInfo						
	CreateMutant	SUPERVISOR_PRIVILEGE STANDARD_RIGHTS_ALL STANDARD_RIGHTS_REQUIRED DELETE READ CONTROL WRITE DAC WRITE_OWNER SYNCHRONIZE					
	OpenKey	MAXIMUM_ALLOWED					
	OpenKeyEx	READ_CONTROL					
	LdrLoadDll					0	
	LdrGetProcedureAddress					1	
	NtQueryKey					1	
	EnumerateKey					1	
	Query ValueKey					1	
	NtClose					1	
	LdrGetDllHandle					1	
RAT4		.text .sedata .idata .rsrc				mscoree.dll, MSVCRT.dll, IPHLPAPI.dll, PSAPI.dll, KERNEL32.dll, USER32.dll, ADVAPI32.dll, SHELL32.dll	

Weaponisation		Delivery				Exploitation				Installation				Command and control				Action objective		
		Network-based Evasion		Email attachments		Web links		Userland rootkit		Desired DLL loading		Share access		Protocols		IP address		TCP UDP HTTPS(s)		Data exfiltration
	Host-based evasion	Embedding malicious code	Rootkit	Backdoor	Port	protocols	Macro PDF	Downloader	Protection	access	DLL	Share	access	Protocols	TCP	UDP	HTTPS(s)	Dropped files		
RAT1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	14	1	
RAT2	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	15	1	*	
RAT3	*	*	*	*	*	*	*	*	*	*	*	*	*	4	3	9	25	14	*	
RAT4	*	*	*	*	*	*	*	*	*	*	*	*	*	1	1	0	14	1	*	
RAT5	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	18	1	*	
RAT6	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	14	1	*	
RAT7	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	18	1	*	
RAT8	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	13	1	*	
RAT9	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	12	1	*	
RAT10	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	12	1	*	
RAT11	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	16	1	*	
RAT12	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	10	1	*	
RAT13	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	15	1	*	
RAT14	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	14	1	*	
RAT15	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	19	1	*	
RAT16	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	17	1	*	
RAT17	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	12	1	*	
RAT18	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	12	1	*	
RAT19	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	20	1	*	
RAT20	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	1	14	1	*	
RAT21	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	2	15	2	*	

*Indicates availability of selected property in a threat actor

References

1. S. Walker-Roberts, M. Hammoudeh, and A. Dehghantanha, “A Systematic Review of the Availability and Efficacy of Countermeasures to Internal Threats in Healthcare Critical Infrastructure,” *IEEE Access*, vol. 6, pp. 25167–25177, 2018.
2. M. Conti, A. Dehghantanha, K. Franke, and S. Watson, “Internet of Things security and forensics: Challenges and opportunities,” *Futur. Gener. Comput. Syst.*, vol. 78, pp. 544–546, 2018.
3. H. H. Pajouh, A. Dehghantanha, R. Khayami, and K. K. R. Choo, “Intelligent OS X malware threat detection with code inspection,” *J. Comput. Virol. Hacking Tech.*, pp. 1–11, 2017.
4. L. Chen, T. Li, M. Abdulhayoglu, and Y. Ye, “Intelligent malware detection based on file relation graphs,” in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, 2015, pp. 85–92.
5. A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, “Detecting crypto-ransomware in IoT networks based on energy consumption footprint,” *J. Ambient Intell. Humaniz. Comput.*, vol. 0, no. 0, p. 0, 2017.
6. H. Haddad Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo, “A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks,” *IEEE Trans. Emerg. Top. Comput.*, vol. 6750, no. c, pp. 1–1, 2016.
7. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, “Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence,” *IEEE Trans. Emerg. Top. Comput.*, vol. 6750, no. c, pp. 1–11, 2017.
8. H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K. K. R. Choo, “A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting,” *Futur. Gener. Comput. Syst.*, vol. 85, pp. 88–96, 2018.
9. A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, “Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning,” *IEEE Trans. Sustain. Comput.*, vol. 3782, no. c, pp. 1–1, 2018.
10. M. Damshenas, A. Dehghantanha, and R. Mahmoud, “A Survey on Malware propagation, analysis and detection,” *Int. J. Cyber-Security Digit. Forensics*, vol. 2, no. 4, pp. 10–29, 2013.
11. “kaspersky.” [Online]. Available: https://kasperskycontenthub.com/securelist/files/2016/11/KL_Predictions_2017.pdf.
12. A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, *Machine learning aided static malware analysis: A survey and tutorial*, vol. 70, 2018.
13. J. Baldwin and A. Dehghantanha, “for Opcode Density Based Detection of Crypto-Ransomware,” 2018.
14. M. Hopkins and A. Dehghantanha, “Exploit Kits: The production line of the Cybercrime economy?,” *2015 2nd Int. Conf. Inf. Secur. Cyber Forensics, InfoSec 2015*, pp. 23–27, 2016.
15. A. Khalilian, A. Baraani, “An Investigation and Comparison of Metamorphic Virus Detection and Current Challenges.,” *Biannu. J. Monadi Cybersp. Secur.*, 2014.
16. “AV-TEST,” 2018. [Online]. Available: <https://www.av-test.org/en/statistics/malware/>.
17. McAfee, “McAfee Labs Threat Report,” no. December, p. 50, 2016.
18. D. Kiwia, A. Dehghantanha, K.-K. R. Choo, and J. Slaughter, “A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence,” *J. Comput. Sci.*, Nov. 2017.
19. G. Canfora, F. Mercaldo, C. A. Visaggio, and P. Di Notte, “Metamorphic Malware Detection Using Code Metrics,” *Inf. Secur. J. A Glob. Perspect.*, vol. 23, no. 3, pp. 57–67, May 2014.
20. S. Wu, S. Liu, W. Lin, X. Zhao, and S. Chen, “Detecting Remote Access Trojans through External Control at Area Network Borders,” *Proc. - 2017 ACM/IEEE Symp. Archit. Netw. Commun. Syst. ANCS 2017*, pp. 131–141, 2017.
21. S. Shin, J. Jung, and H. Balakrishnan, “Malware prevalence in the KaZaA file-sharing network,” in *Proceedings of the 6th ACM SIGCOMM on Internet measurement - IMC '06*, 2006, no. May, p. 333.

22. S. Mohtasebi and A. Dehghantanha, "A Mitigation Approach to the Malwares Threats of Social Network Services," *Multimed. Inf. Netw. Secur.*, pp. 448–449, 2009.
23. X. M. Wang, Z. B. He, X. Q. Zhao, C. Lin, Y. Pan, and Z. P. Cai, "Reaction-diffusion modeling of malware propagation in mobile wireless sensor networks," *Sci. China Inf. Sci.*, vol. 56, no. 9, pp. 1–18, 2013.
24. D. Jiang and K. Omote, "A RAT detection method based on network behavior of the communication's early stage," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E99A, no. 1, pp. 145–153, 2016.
25. M. N. Kondalwar and C. J. Shelke, "Remote Administrative Trojan/Tool (RAT)," *Int. J. Comput. Sci. Mob. Comput.*, vol. 3333, no. 3, pp. 482–487, 2014.
26. D. Jiang and K. Omote, "An approach to detect remote access trojan in the early stage of communication," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, vol. 2015–April, pp. 706–713, 2015.
27. U. Losche, M. Morgenstern, and H. Pilz, "Platform Independent Malware Analysis Framework," *Proc. - 9th Int. Conf. IT Secur. Incid. Manag. IT Forensics, IMF 2015*, pp. 109–113, 2015.
28. "scmagazine." [Online]. Available: <https://www.scmagazine.com/cross-platform-rat-alienspy-targets-mac-os-x-windows-and-android-users/article/535974/2/>.
29. A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Mobile malware detection through analysis of deviations in application network behavior," *Comput. Secur.*, vol. 43, pp. 1–18, Jun. 2014.
30. M. Lindorfer, C. Kolbitsch, and P. M. Comparetti, "Detecting environment-sensitive malware," in *International Workshop on Recent Advances in Intrusion Detection*, 2011, vol. 2011, pp. 338–357.
31. A. Karim, S. Adeel, A. Shah, and R. Salleh, "New Perspectives in Information Systems and Technologies, Volume 2," vol. 276, pp. 153–164, 2014.
32. X. Ugarte-Pedrero, D. Balzarotti, I. Santos, and P. G. Bringas, "SoK: Deep packer inspection: A longitudinal study of the complexity of run-time packers," *Proc. - IEEE Symp. Secur. Priv.*, vol. 2015–July, pp. 659–673, 2015.
33. B. B. Gupta, A. Tewari, A. K. Jain, and D. P. Agrawal, "Fighting against phishing attacks: state of the art and future challenges," *Neural Comput. Appl.*, vol. 28, no. 12, pp. 3629–3654, Dec. 2017.
34. M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Internet of Things (IoT): Taxonomy of security attacks," *2016 3rd Int. Conf. Electron. Des.*, pp. 321–326, 2016.
35. A. Cook, H. Janicke, R. Smith, and L. Maglaras, "The industrial control system cyber defence triage process," *Comput. Secur.*, vol. 70, pp. 467–481, 2017.
36. T. Who and E. T. Hunting, "Interested in learning SANS Institute InfoSec Reading Room The Who, What, Where, When, Why and How of."
37. T. Yadav and A. M. Rao, "Technical aspects of cyber kill chain," in *International Symposium on Security in Computing and Communication*, 2015, pp. 438–452.
38. S. Attaluri, "Detecting Metamorphic Viruses Using Profile Hidden Markov Models," no. December, 2007.
39. B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep Learning for Classification of Malware System Call Sequences."
40. T. Yadav and P. Szor, "The art of computer virus research and defense," *Choice Rev. Online*, vol. 43, no. 03, pp. 43–1613–43–1613, Nov. 2005.
41. M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Comput. Surv.*, vol. 44, no. 2, pp. 1–42, 2012.
42. F. Daryabar, A. Dehghantanha, and N. I. Udzir, "Investigation of bypassing malware defences and malware detections," in *Information Assurance and Security (IAS), 2011 7th International Conference on*, 2011, pp. 173–178.
43. M. Assante and R. Lee, "Interested in learning SANS Institute InfoSec Reading Room System Cyber Kill Chain," 2015.

44. S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, “A Taxonomy of botnet behavior, detection, and defense,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 2, pp. 898–924, 2014.
45. A. Buescher, F. Leder, and T. Siebert, “Banksafe Information Stealer Detection Inside the Web Browser,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6961 LNCS, Springer, 2011, pp. 262–280.
46. A. Stewart, “DLL Side-Loading: A Thorn in the Side of the Anti-Virus (AV) Industry,” *FireEye, Inc*, 2014.

Evaluation and Application of Two Fuzzing Approaches for Security Testing of IoT Applications



Omar M. K. Alhawi, Alex Akinbi, and Ali Dehghantanha

Abstract The proliferation of Internet of Things (IoT) embedded with vulnerable software has raised serious doubts about security of IoT devices and networks. Enhancing fuzzing performance and efficiency to enable testing these software samples is a challenge. Fuzzing is an automated technique widely used to provide software quality assurance during testing to find flaws and bugs by providing random or invalid inputs to a computer software. However, the technique could take significant amount of time and effort to complete during the test phase of the software development lifecycle. Reducing the time required to fuzz a software will improve efficiency and productivity during the software testing phase to enable detailed analysis and fixing of bugs or flaws found in the computer program. There are a number of factors that influence the fuzzing technique, such as quality of test cases or invalid inputs used during the test and how these samples were collected or created. In this paper, we introduce a technique to leverage from the different crashes discovered from two fuzzing approaches to improve fuzzers by concentrating on utilised test cases. The code coverage is used as an efficiency metric to measure the test case on the tested software and to assess the quality of a given input. Different sample features were created and analysed to identify the most effective and efficient feature used as input for the fuzzer program to test the target software.

Keywords Fuzzing · Fuzzing analysis · Software testing · Software quality assurance · IoT · Test case construction

O. M. K. Alhawi

Department of Computer Science, University of Salford, Manchester, UK

e-mail: o.alhawi@edu.salford.ac.uk

A. Akinbi

School of Computer Science, Liverpool John Moores University, Liverpool, UK

e-mail: o.a.akinbi@ljmu.ac.uk

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

e-mail: ali@cybersciencelab.org

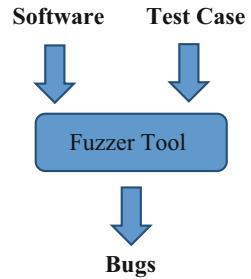
1 Introduction

These days, cyber attacks are targeting almost all our digital infrastructure and services from banking [1] and cloud computing [2, 3] and big data analytics platforms [4] to Windows [5] and Mac [6] machines and mobile devices [7]. Emerging Distributed Denial of Services attack [8], Malware [9] and Ransomware [10, 11] attacks and Botnets [12] along with ever increasing software and hardware vulnerabilities [13] made security a significant barrier in wider deployment of new technologies including Internet of Thing (IoT). In many cases, especially in the Internet of Things (IoT) domain bugs and security flaws are found after software have been released [14], which make them vulnerable to exploits that can compromise security of devices running such vulnerable applications [15]. IoT applications vulnerabilities may expose the entire network to attacker [16] and cause a significant damage [17]. A range of techniques from utilisation of machine learning [18] and deep learning [19] to the development of novel key generation methods [20] and intrusion detection mechanisms [21] were suggested for securing IoT environments. However, still software defects such as buffer overflows, logic flaws, inconsistent error handling are the major source of security issues in IoT [22]. Software quality assurance plays a key role, especially in IoT environment where a device might not be updated for considerably long time [23] or devices are deployed in the safety critical environments [24]. Testing processes have evolved over time and even though there have been significant improvements in methods, techniques and tools, there are difficulties to keep up with the increasingly rapid evolution of software engineering and trends of development paradigms [25].

Fuzzing is a technique used to find bugs in a specific software. By feeding the targeted software some random and invalid input, we can monitor and observe the software for behavioural anomaly, such as a crash or a denial of service. We can also record the input data used to cause this crash, in order to investigate the exact location of the data responsible for the application malfunctioning within the program stack. This process is often repeated with different data types each time to discover flaws in the software.

Fuzzing has leapt to prominence as a quick and cost-effective method to reveal bugs and security defects before software deployment [26]. This makes it ideal in the discovery of any potential bugs in the program before it is released. Fuzzing is an automated process and requires minimum supervision which makes it ideal to test software without having access to the source code or knowledge of the high-level programming language used to write the application. The test case made up of randomly generated inputs supplied to a fuzzer to fuzz the target software and find bugs as shown in Fig. 1.

Two fuzzers namely FileFuzz and Peachfuzz representing the two main fuzzing approaches are used in this study to determine the most effective and how fuzzing process can be improved for a better efficiency. While analysing fuzzing tools, key parts of these tools will be given more attention in terms of trying to enhance fuzzing

Fig. 1 Fuzzing

efficiency. The information and the results achieved from this project will be in terms of no previous knowledge required about the target internal structure as on mutational fuzzing, or with a previous research about the target structure as in the generational fuzzing approach. This approach helps to test software embedded in IoT application where no knowledge of its internal structure is common as well as the underlying operating system.

The rest of the paper is organised as follows: Sect. 2 highlights various terminologies, fuzzers and the fuzzing strategies. In Sect. 3, we describe the research methodology, along with the required fuzzer programs used in this study. Section 4 presents the experiment and discusses the results. Summary of the study and future works are discussed in Sect. 5.

2 Background

This section describes the basic terminologies, tools, techniques and strategies of using fuzzing and how it impacts the quality of the software. It also gives various benefits of fuzzing a software periodically.

2.1 Fuzzing

Fuzzing is a software testing technique, which basically consist of finding security flaws by injecting randomly generated input in an automated manner. A fuzzer program can be used to fuzz applications running on computer systems or devices which requires user input to be supplied to detect software bugs in order to improve the security of the software or exploit the system running it [27]. It is an automated or semi-automated technique aimed at finding security flaws in a software by providing unexpected and manipulated data to a target and monitor the impact on the target for any irregular behaviour.

Due the automated processes of deploying fuzzers to test running applications without knowledge or access to the underlying source code, it is one of the preferred method of black box testing for penetration testers. The technique is often used by malicious hackers to test applications for vulnerabilities and then write exploits for the identified vulnerabilities.

There are two categories of fuzzers: Mutation-based fuzzers, which takes a valid data input from an existing data sample by the user and mutate it in some types to build a test case,¹ but does update the corresponding cyclic redundancy check. The second is the generation-based fuzzers, which create random data input from the scratch from data model, but generate a correct corresponding in cyclic redundancy check [27, 28]. A further discussion on the techniques and strategies utilised by both categories is described in the following subsection.

2.2 *Fuzzing Techniques and Strategies*

Fuzzing can be used on different types of applications such as database, multimedia, word processing, simulation software and application suites.

Mutation-based fuzzer, also known as dumb fuzzing or protocol-unaware fuzzing, describes a technique used when a valid test case which contains invalid input is used to fuzz a target. The test case of this method is changed and mutated to contain anomalies without proper understanding of the input data or the target software format. A fuzzer program such as “File Fuzz” automates this process by taking a valid template from the user and mutates it to create test cases required to cause the target software to malfunction.

Generation-based fuzzer, also known as intelligent fuzzing or protocol-aware fuzzing, requires previous knowledge of the target, as the test case should be built from the scratch based on the format. For instance, targeting a program which requires a request for comments (RFC) through user input, we need to search for some knowledge of the software by looking through the software documentation or by reverse engineering the software to have an understanding of the software structure and format. Then the journey starts for building the test case which includes adding anomalies to the information extracted. Adjusting test cases is very difficult to do manually. However, a few fuzzing tools exists will make this process easier. Both methods will be discussed further in this paper.

All the methods covered so far counts as black-box fuzzing. According to Microsoft Forefront Edge Security team [29], black-box fuzzing requires sending malformed data without actual verification of which code paths were hit and which were not. In black-box testing, it is not important to have knowledge about the

¹Test case should be similar to a real valid data, but it must have problem on it “anomalies”. E.g. To fuzz Microsoft office, a test case should be a word document or excel sheet (data sample), so, the mutated version generated of similar package called test case.

target source code, which gives it a strength and wider usage among developers and attackers. However, the testing efficiency will not meet developer expectations when testing a software with complicated logics and because of the minimum knowledge about the target. In addition, black-box test provides low code coverage and can miss security bugs [30].

White-box techniques or smart fuzzing, different names but shares the same principle of extending the scope from unit testing as on black-box test to a whole-software testing searching for negative paths, where it can cover millions of instruction, execution traces and lines of code [29, 30]. Symbolic execution used during white box test [31]. Where symbolic values are used instead of normal data, this method allows you to know how the software is going to behave when it fed by infinite inputs and to explore the unfeasible spaces of input if a random test was performed. A new command is given to mutate the inputs used when the branch reached satisfiability modulo theories SMT² solver. If the problem was solved, the software runs again with a new input. The mechanism performed at this technique focus on discovering new paths to lead to whole-software security testing. However, white-box fuzz requires high effort, skills set and expensive.

2.3 Seed Selection

One of the most important aspect during fuzzing is the valid samples³ used to fuzz a target. Basically, fuzzers required a large number of samples to be able to cover more paths on our target. Depending to the target, these samples could be larger than anticipated and may require lot of time and computer processing power to complete. This issue could hinder the opportunity to discover more vulnerabilities.

Therefore, it is highly recommended to find a method to minimize the selections seeds required to fuzz a program as the program normally repeat running the same code blocks each time [32]. By removing the samples that use the same paths or have the same coverage on the target code, without reducing the coverage quality on our target code. Thus, if this process is achieved, the time and effort required for fuzzing process will be reduced.

Choosing the best seed files can greatly increase the number of bugs found during fuzzing. In order to minimize the samples required to cover most of the target blocks, the first step is to generate the best samples set and the code coverage for each sample used in our target will be saved and extracted using one of the instrumentation tools. Secondly, the information extracted will be compared to other coverage achieved from different sample. The best ones according to its coverage

²SMT solver is a tool adding equality reasoning, arithmetic, fixed-size bit-vectors, arrays, quantifiers and decide the satisfiability of formulas in these theories [39].

³Samples which are accepted as an input to our research target.

will be chosen until covering the majority of our target paths. The other samples will be excluded from our data set.

This phase can be done using different techniques. However, in this paper, an automated python script called (`minset.py`⁴) which makes use of the Microsoft Research Phoenix Project to find the basic blocks in an application was used. It is part of the generational fuzzer tool (Peach) libraries and has the ability to perform code coverage of a target application as it consumes a set of sample files and select the set of files that generates the most coverage of the target application. Feeding the chosen script, the directory of original samples and the path to the software where these sample supposed to be tested on, showed the result contains fewer samples but with the same efficiency as the original samples and will be saved in a new folder.

The script was seen to be the most suitable to work with generational fuzzers. This type of fuzzers required high volume of samples to be downloaded or created manually simply, because lack of knowledge of quality of these samples. Therefore, it takes more time and effort if these samples were not filtered appropriately, which may not be commensurate with the time allocated for this project. However, the process is different for Mutational-fuzzing approach, where fuzzers from this category required one valid sample only “template” to generate the seeds/test cases from. In this case, the fuzzer add, delete or replace part of the template inputs to generate the seeds required during fuzzing. The number of the seeds generated will be according to the selected scope as it is shown in Fig. 11.

2.4 *Instrumentation*

There are different types of fuzzers and as such they require different types of monitoring methods according to the software being tested [33]. The main goal for using instrumentation tools during fuzzing, is because of its ability to take control of the software being fuzzed at a runtime, as it capable to modify or delete any instruction before executed and monitoring the results. In which it will assist us to further analysis our research target behaviour, very similar to Burp Suit tool when we send a specific session to Repeater to work on a specific process. This phase can be broadly done in the following ways:

Dynamic instrumentation: Which is a technique applies when an emulated process is required to be modified dynamically while program is running, by executing the new instrument⁵ code just before the target code being analyse in two separate processes. The power of using dynamic instrumentation relies on its ability for manipulating each instruction in an existing binary before execution [34]. Figure 2 describe the basic operation for dynamic instrumentation framework.

⁴Tool description can be found here: <http://community.peachfuzzer.com/v2/Tools.html>.

⁵“instrument” used to refer to the extra piece of code added to a software. While sometimes it used to refer to the code doing the instrumentation itself.

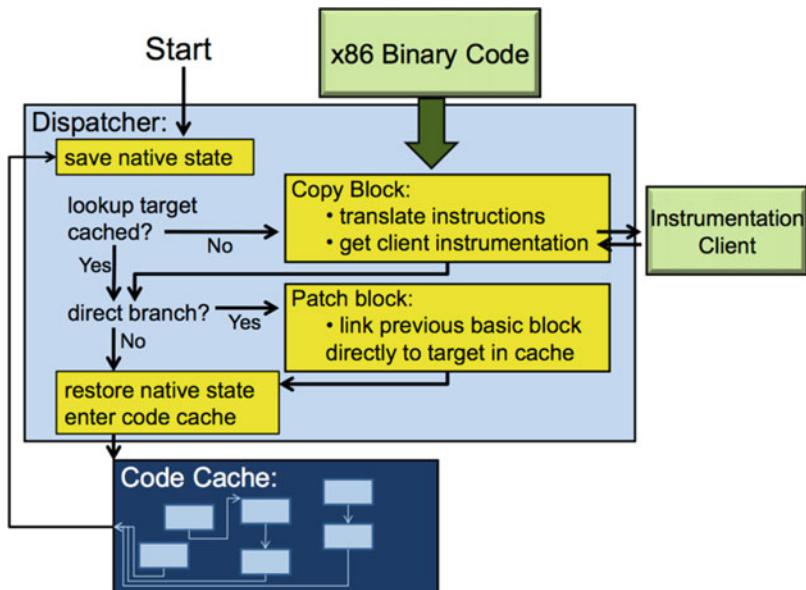


Fig. 2 Dynamic instrumentation framework

Static Instrumentation: This is an approach for adding instrument code to the software before running it. Essentially the probes are baked in to your binary. There are two ways this can be done. One is to have the compiler put the probes in. Visual C++ takes this approach to implement Profile Guided Optimizations (PGO). The other method takes the binary which has been emitted by a compiler, crack and discover all the methods, instrument and modified the binary like it never been touched.

3 Methodology

This section describes two categories of fuzzers representing two fuzzing approaches used in this study in order to improve speed, accuracy and efficiency during fuzzing with less user interaction. In addition, several fuzzing steps and supporting tools will be analysed and explained in detail as the results in the next section will be based on them.

3.1 Metrics

To measure the tracking of different fuzzing tools on our target, as it is not included on the fuzzer itself. It was compulsory to use different tools/methods to cover this area. Which can be broadly done in the following ways:

Code coverage: also known as fuzzer tracking is an important metric used to measure the efficiency during the fuzzing process. [29], has defined code coverage as a technology which allows inspection of which code paths were executed during fuzz testing. Which is important during the verification of test effectiveness and to improve sample coverage. Code coverage gathered the lines of the software code which was instrumented and make a comparison of what of these codes was reached by a fuzzer and which was not, and give a percentage according to this performing. According to this information, code coverage will be used as a method to make a comparison between the two fuzzers used in this project.

The tool described Sect. 2.4 will be used mainly to measure the code coverage/traces on my target. In addition, to the branch purpose of reducing the selecting set samples.

Time: Using code coverage as a metric has nothing to provide to the end customer about the test quality [35]. It is therefore required another measurement guidance while working on the research objectives. The time required to perform a certain process will be used as metric, because it will show how long the process will take, which will be used to differentiate between the two fuzzers, to record the optimal time each of them consumed for a specific fuzzing process.

This metric will be performed on the mutational fuzzer by looking into the time before and after fuzzing is finished. However, the generational fuzzer will record the time itself.

3.2 Fuzzing Methods

In this section, the two fuzzing approaches (mutational and generational) are explained. In addition, the two fuzzers and the supporting tools will be described in detail, to make a valid comparison between the two main approaches/tools used to fuzz my research target program.

3.2.1 Mutational Fuzzing

Dumb or mutational fuzzing, is one of the fuzzing strategies which required a test case (template) at first phase before start fuzzing. The test cases on dump fuzzing

will be built by manipulating a set of valid inputs. The second phase will mutate these inputs and build a set of seed test cases automatically, where each of them will be used on fuzzing. Our research target will run the new mutated data, and start monitoring any crashes occurs. In other words, mutational fuzzing can be illustrated into three different phases as shown in Fig. 3 below.

FileFuzz

A well-known mutational-fuzzing tool written by Michael Sutton was chosen called FileFuzz. This fuzzer was chosen because it works on the three mutational-fuzzing phases separately, as this allows proper test and analysis of the fuzzing process, most importantly when generating the test cases and using it. In addition, the fuzzer has a graphic user interface which provides a user-friendly interface to monitor the fuzzing progress in order to observe the memory location, registry value and the types of the error that occur during the crash.

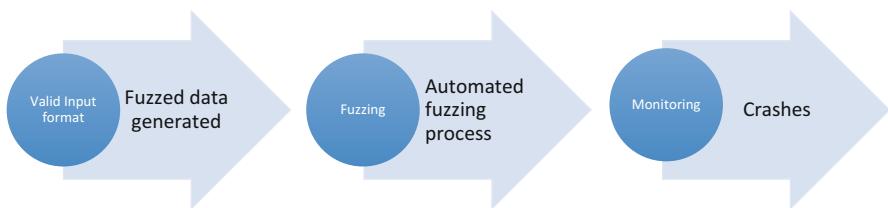


Fig. 3 Mutational fuzzing phases

3.2.2 Generational Fuzzing

The generational/smart fuzzer, required previous knowledge about the target structure, the input format of the target or the file definition of the target. Therefore, previous research should be made to understand the file definition and instead of randomly generating test case like the procedures followed at mutational fuzzing, the researcher should compose the syntax that illustrate how the target works. The fuzzer takes this information, analyse it, generate smart test cases and send the new fuzzed data to our target to fuzz. Because this technique requires prior knowledge, the results will vary based on the mechanism used in this approach. Generational fuzzing can be divided into five phases, as shown at Fig. 4 below.

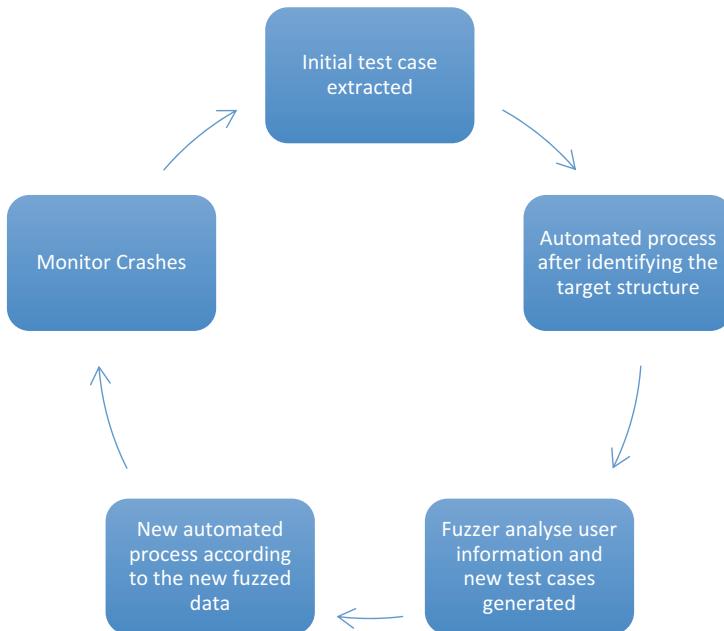


Fig. 4 Generational fuzzing phases

Peach Fuzzing Framework

Peachfuzz is one of the industry's most advanced security testing platforms written in python [36]. Peach can be used to fuzz almost everything starting from Programs, ActiveX, Network application, SQL ... Etc. This tool provides a quick service to generate a fuzzer depend on the target's format method. The smart platform requires Peach Pit Packs that contained a definition of the target specifications to meet our target requirements before fuzzing starts. This pack store in a file where Peach calls the required Pit based on the target the user selects to fuzz, by defining the relation between the data format and the data structure. The outcome of this definition will be used by Peach to add anomalies to the collected data in a smart way.

3.3 *Experiment Tools*

Dynamic Binary Instrumentation The power of using dynamic instrumentation during fuzzing is its ability to add/delete from the code while it is running. The tool chosen for this research was PIN3.2 [37], to record and deeply monitor the activities while fuzzing the target “Sumatra PDF”. The tool was chosen because not only is it produced from Intel but it is a portable version and supports almost all the platforms including Linux.

Although debuggers require to be paused for tracing, dynamic instrumentation and provides the flexibility to trace without interruption, Pintools provides a rich application programming interface (API) and its ability to instrument per block or per function. The test bed environment in this study was done using a Windows Operating System. It required using Microsoft Visual Studio to support running the tool and the files attached to it.

! exploitable 1.6.0 Fuzzing our target it is likely to crash several times at one or different locations, this will be logged at one package by the fuzzer for further analysis. However, it is possible to use another automatic debugging tool for more advance properties such as: prioritising the crashes based on its severity to distinguish between different crashes. In this study, we used !exploitable, a windows debugging extension tool attached with Peach fuzzer. This tool was first published by Microsoft after 14 months of fuzzing effort when Windows Vista released in 2009 [38]. This tool is able to prioritise and assigns the vulnerabilities, according to its exploitability such as: exploitable, probably not exploitable or unknown.

!exploitable debugger depends on crash logs with the ability to analyse any crashes that occur. Peach fuzzer uses this tool to classify the discovered vulnerabilities and differentiate amongst them based on predefined rules.

Interactive Disassembler IDA Pro 5.0 A tool which was used to dissemble the target code, produce and record disassembled database (.idb). It contains all the structures and functions of the target software application, where the tester can return to it during the investigation process. In addition, the output from this tool can be used for more analysis through different tools such as: BinDiff which supports exporting IDA output files, was used to discover the instructions executed while running different mutated samples on the target “Sumatra PDF” application. The main goal of this tool is to reverse engineer the target application source code. All the target functions and variable names will be uncovered with the ability to display the result in different methods, which will make it easy to link between functions, subroutines and source code.

3.4 Test Environment

The practical work for this project was conducted on both Windows 7 and XP installed on two different Virtual Machines, to prevent any problems coming from the main host (Mac OS X). Both VMs were prepared by installing all the required applications that support running both fuzzer tools such as python2.7, win32ap, .net framework 4.5, Microsoft Visual C++ 2015 and visual basic 2010. In addition, ASLR was disabled on both virtual machines to analyse the executions process without platform restrictions. Both of the VMs were issued the following configuration (Table 1):

Table 1 VMs configuration

System configuration	Value
OS	Windows XP, Windows 7
CPU	2 Core each
Ram	4 GB for each
Virtual hard disk capacity	40 GB

The host specifications as shown at the table below (Table 2):

Table 2 Host specification

System configuration	Value
OS	MacOS 10.12.6
CPU	2.9 GHz Intel Core i7
RAM	16 GB 2133 MHz LPDDR3
Hard disk capacity	500 GB

During fuzzing, both VMs are required to work continuously for long periods until you get satisfactory results, the laptop was configured not to sleep, the fan increased to 4500 rpm as the laptop temperature raised to higher 70 °C and the screen will not turn off, as this may cause interruption and waste the time already spent on fuzzing specially during mutational-fuzzing approach when FileFuzz tool is used. In addition, both VMs have shared folder with the main host, to share files and facilitate the coordination among them.

Sumarta PDF The target software in this study is Sumarta PDF. It is an open source pdf viewer which can run both on Windows and Linux platforms. The graphical user interface on our target is required, as this will add some complexity to be attached with. In addition, the target required a PDF file format as an input, which is not difficult to collect. The target size for the latest Sumarta version is 5.2 MB and below 2 MB for earlier version which will add simplicity to reach code blocks during our fuzz testing. The earlier version Sumatra1.5.1 is used in this study as target should not be an up to date version (mature), as this will result in less bugs and a robust software. My target consists of one executable file only, where no libraries are required to execute it.

4 Experiments and Results

4.1 Generating a Minimum Set of the Chosen Templates

Both fuzzers are required PDF file format as input stream because of our target specifications. Where it should contain different features to enhance the possibility

of reaching each block of our target. Therefore, it is required to download pdf files from different resources such as: www.malwr.com/analysis/search “type:pdf” and from Google by using this command “test +.pdf”. 2000 PDF initial samples were downloaded for my generational fuzzer “Peachfuzz” where it will be under construction for further investigation to generate a minimum set of these samples. Furthermore, one sample is required for FileFuzz which representing the mutational-fuzzing approach, which will be selected from the new dataset generated from the script described in Fig. 5. According to the mutational strategy the fuzzer takes the mutation template as an input and producing a fuzzed buffer to be sent to the tested software [29].

It is important to minimize the collected dataset, to save days of work by excluding the samples which have the same effect on the target code, without reducing the quality of the original dataset. The script used for this purpose is “minset.py” which will be used to reduce the 2000 PDF samples downloaded but with the same coverage on Sumatra PDF features. The python script was able to analyse and determine the coverage for each sample on our target code, excludes the samples with similar/duplicate coverage and generate a new dataset. By using this script, the researcher, saved weeks of testing all the collected samples. The new dataset generated contain 132 PDF files instead of the original 2000 ones after more than 2 days of non-stop operation on 4 Cores/10GB RAM. In other words, 80% of the original dataset is useless. The tool commands and part of the process while reducing the 2000 PDF samples can be seen in Fig. 6.

```
c:\peach\tools\minset>python minset.py
] Peach Minset Finder v0.9
] Copyright (c) Michael Eddington
Usage:
minset.py [-k] -s samples -m minset command.exe args %s
minset.py [-k] -s samples -t traces command.exe args %s
minset.py -s samples -t traces -m minset

Note:
"%s" will be replaced by sample filename.
```

Fig. 5 Minset script commands

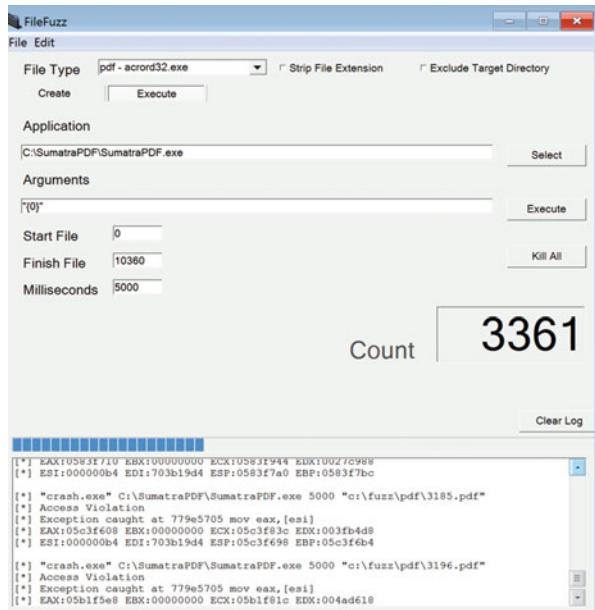
```
C:\Users\Administrator\Desktop\peach-3.1.124-win-x86-debug>peachminset -s pdf -m minset -t traces C:\SumatraPDF\SumatraPDF.exe %s
] Peach 3 -- Minset
] Copyright (c) Deja vu Security
[*] Running both trace and coverage analysis
[*] Running trace analysis on 2000 samples...
[0:00:00.2401212] (1:2000) Coverage trace of pdf\ 2sample.pdf... Completed
[0:00:02.7267827] (2:2000) Coverage trace of pdf\ 3sample.pdf... Completed
[0:1:29.1362308] (3:2000) Coverage trace of pdf\ 4sample.pdf...■
```

Fig. 6 Screenshot while generating the dataset

4.2 Fuzzing Tests

FileFuzz Strategy The beauty of using this strategy, is that no prior knowledge about our target is required and all requirements for this test is based on the file format of the target software which is a pdf. A chosen PDF format was selected and fed to the fuzzer, because of the input stream our target support. The pdf file (template) selected was used as an initial test case, where FileFuzz was later used to generate test cases with different code coverage. The selected file was split by FileFuzz to 10,360 PDF files. The time limit was increased slightly from the default 2 s value to 5 s, which gave the fuzzer adequate time to test each of the generated samples against the target. If the software did not crash, the fuzzer will stop the current test sample and proceed to the next one. Figure 7 shows the fuzzing process.

Fig. 7 FileFuzz fuzzing process



Peachfuzz Strategy From the new dataset generated using the python script, the script was then attached to Peach. Knowledge of the internal structure of the target software was carried out using IDA Pro and PIN. Peach required pit file (.xml) to begin the fuzzing process. The pit file was generated and modified according the Sumatra PDF functions and the input stream supported was linked to the minimum data set.

Peach engine generated the data model according to the definition of each PDF sample as one binary large object (blob). Each of the data model section worked as a precursor to allow Peach recognise the target structure before manipulating the

test cases by adding invalid inputs. Peach uses different elements for this purpose which can be found inside Peach folders. The fuzzing process required a significant processing time compared to the FileFuzz strategy to cover all the target's functions. The process was terminated for each sample after 1500 iterations in order to gain a reasonable coverage of the target's features. Figures 8 and 9 show Peach command line during the fuzzing process.

```
c:\Users\Administrator\Desktop\peach-3.1.124-win-x86-debug>peach.exe -h
[[ Peach v3.1.124.0
Copyright (c) Michael Eddington
This is the Peach Runtime. The Peach Runtime is one of the many ways
to use Peach. Right now, Current functionality is limited to development,
but already exposes several abilities to the end-user such as performing
simple fuzzer runs and performing parsing tests of Peach XML files.

Please submit any bugs to https://forums.peachfuzzer.com.

Syntax:
  peach -a channel
  peach -c peach_xml_file [test_name]
  peach [-skipto #] peach_xml_file [test_name]
  peach -p 100 [-skipto #] peach_xml_file [test_name]
  peach -r range 100-200 peach_xml_file [test_name]
  peach -t peach_xml_file
  -1                                         Perform a single iteration
```

Fig. 8 Peachfuzz commands

```
[74,103440,13:30:49.041] Performing iteration
[*] Fuzzing: TheDataModel.DataElement_0
[*] Mutator: BlobBitFlipperMutator
Peach.Core.MutationStrategies.Sequential.ApplyMutation: Fuzzing: TheDataModel.DataElement_0
Peach.Core.MutationStrategies.Sequential.ApplyMutation: Mutator: BlobBitFlipperMutator
Peach.Core.Dom.Action.ActionType.output
Peach.Core.Dom.Action.ActionType.open()
Peach.Core.Publishers.FilePublisher output(39762 bytes)
Peach.Core.Dom.Action.ActionType.close
Peach.Core.Publishers.FilePublisher close()
Peach.Core.Agent.AgentManager Message: Action.Call => ScoobySnacks
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid _StopDebugger
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid Cpu is idle, stopping process.
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid _StopDebugger
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid _DetectedFault()
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid DetectedFault() - No fault detected

[75,103440,13:30:56.052] Performing iteration
[*] Fuzzing: TheDataModel.DataElement_0
[*] Mutator: BlobBWORDSliderMutator
Peach.Core.MutationStrategies.Sequential.ApplyMutation: Fuzzing: TheDataModel.DataElement_0
Peach.Core.MutationStrategies.Sequential.ApplyMutation: Mutator: BlobBWORDSliderMutator
Peach.Core.Dom.Action.ActionType.output
Peach.Core.Publishers.FilePublisher open()
Peach.Core.Publishers.FilePublisher output(39762 bytes)
Peach.Core.Dom.Action.ActionType.close
Peach.Core.Publishers.FilePublisher close()
Peach.Core.Agent.Action.ActionType.Call
Peach.Core.Agent.AgentManager Message: Action.Call => ScoobySnacks
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid _StopDebugger
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid _WaitForExit() failed: Cannot process request because the process (2964) has exited.
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid _WaitForExit()
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid _DetectedFault()
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid DetectedFault - Using system debugger, caught exception
Peach.Core.Agent.AgentManager Fault detected: Collecting monitor data.
Peach.Core.Agent.AgentManager Fault detected: Collecting monitor data.
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid _FinishDebugger
Peach.Core.Agent.Monitors.WindowsDebuggerHybrid _StopDebugger
Peach.Core.Engine runTest: detected fault on iteration 75

-- Caught fault at iteration 75, trying to reproduce --
Peach.Core.Loggers.FileLogger Found core fault [Exception: 0xc0000005]
Peach.Core.Loggers.FileLogger Saving action: [Exception: 0xc0000005]
Peach.Core.Loggers.FileLogger Saving fault: [Exception: 0xc0000005]
Peach.Core.Engine runTest: Attempting to reproduce fault.
Peach.Core.Engine runTest: replaying iteration 75
```

Fig. 9 Peach fuzzing process

4.2.1 PIN Extract Tracer

Both of Pin and Peachfuzz should be run separately. However, the result relies on what instructions given to the PIN tool while the target is running the mutated sample through Peach. PIN tool gathered the code coverage information, by recording all the functions and addresses used when and after the target executed. All the samples produced errors were processed individually to record the execution

traces, by running each of the samples against the target using PIN tool to discover all the instructions and functions the sample touched at Sumatra PDF, as shown in the Fig. 10. The result of this process was saved for further analysis to find more about the exact location and the functions which caused the crashes. It would be important to analyse how the new code from the mutated sample which caused the crash differs from the original sample as shown in Figs. 11 and 13 respectively.

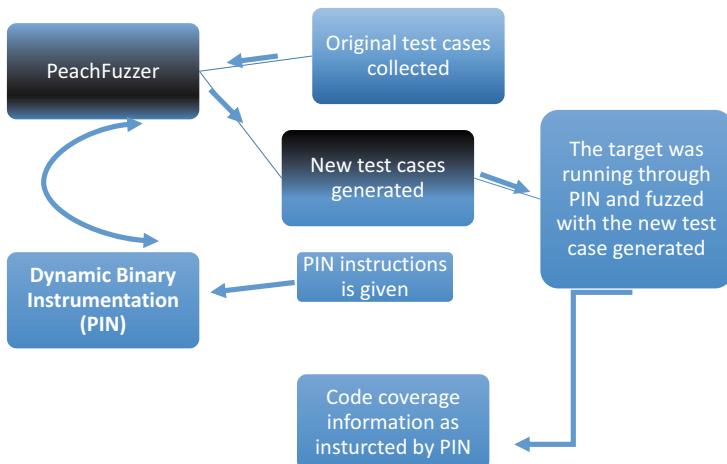


Fig. 10 Getting code coverage information

4.3 Test Results

Faults and exceptions were seen from the results using both fuzzers. Each of these crashes were produced due to the mutated samples supported by Peach fuzzer. On the other hand, the FileFuzz was tested against the Sumatra PDF software in an uncontrolled environment. As mentioned earlier the result from FileFuzz was recorded in the fuzzer main interface as it showed at Fig. 7. The FileFuzz results is summarised in the chart below based on the number of faults and the exceptions (Chart 1).

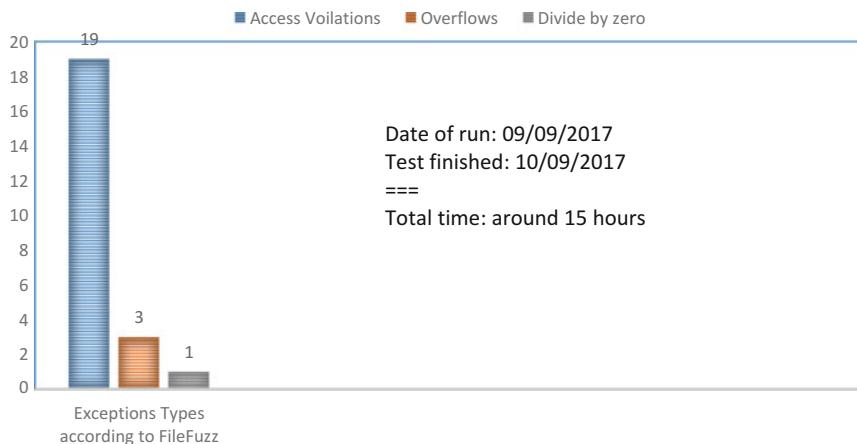


Chart 1 FileFuzz fuzzing results

Peachfuzz crashes were categorized as described before by using the built-in debugger (!exploitable), which store and classified each of the faults at logs folder.⁶ The Chart below provides a summary of these results based on !exploitable classifications.

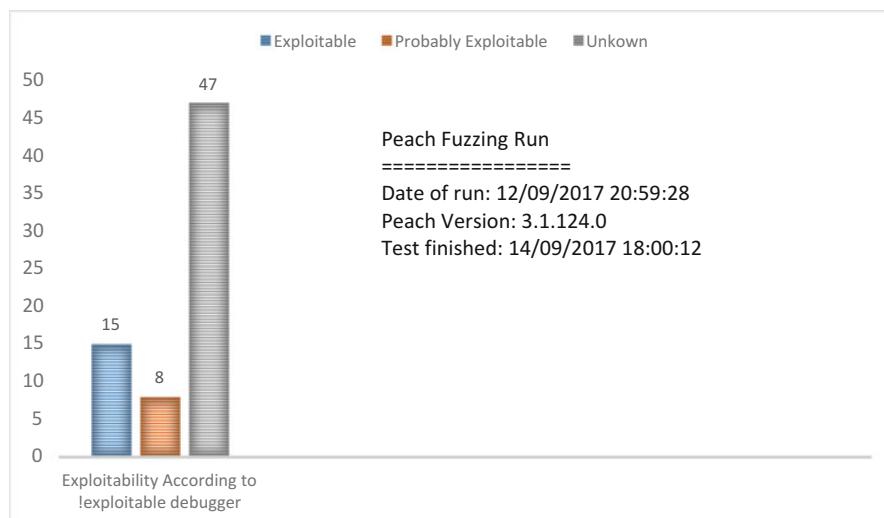


Chart 2 Peach fuzzing results

⁶C:\Users\Administrator\Desktop\peach-3.1.124-win-x86-debug\Logs\omar.xml_20171209205928\Faults.

4.4 Analysis of Both Fuzzing Results

This section demonstrates FileFuzz exceptions analysis and Peach fault analysis using different crashes presented in Sect. 4.3. By illustrating how the data input used during fuzzing affect the fuzzing results.

4.4.1 Crash Analysis

From both of the charts presented, it was clear how insufficient FileFuzz test was. The temporary method FileFuzz used to record the exceptions on the fuzzer main interface led to loss of results already been displayed in the event of any failure and therefore, all the samples had to be tested again. In addition, the total exceptions FileFuzz produced was less than Peach fuzzer. However, it is very important to mention that the result presented from FileFuzz was tested once against our target Sumarta PDF.

Therefore, this type of fuzzing requires different rounds and different samples to achieve a better and consistent result. Furthermore, to have a valid comparison between both fuzzers this test was performed once on both fuzzers.

The result FileFuzz produced was analysed and the results showed few faults share the same address. For instance, address 779e5705 was reported at four different faults triggered by four different samples. Using Pin tool, these samples were tested again and analysed based on the instructions and functions each sample produced. In addition, the four samples were analysed using a hex editor to display the samples inputs. Figure 11 shows that each of the sample have almost the same data. However, the fuzzers classify them as four unique faults which share the same address “779e5705”, ESI “000000b4” and EDI “703b19d4” but with different ESP, ECX, EDX and EAX for each of the faults discovered.

	3196.pdf x	3084.pdf	3160.pdf	3185.pdf	
	Edit As: Hex	Run Script	Run Template		
0	25	50	44	46	0123456789ABCDEF
1	2D	31	2E	33	PDF-1.3.%ääÖ..
2	0A	25	E2	E3	
3			CF	D3	00 00
4				E	F
5					
6					
7					
8					
9					
A					
B					
C					
D					
E					
F					
0000h:	25	50	44	46	
0010h:	00	00	00	00	
0020h:	00	00	00	00	
0030h:	00	00	00	00	

Fig. 11 Display the raw hex bytes for 3196.pdf, 3084.pdf, 3185.pdf and 3160.pdf samples

After analysing the new results generated from PIN tool which displayed all the instructions and functions executed when the target ran by each of the samples mentioned at Fig. 10, by searching for the function address where crash occurs. Four different instructions were founded which caused four unique access violation exceptions, but within the same function “779e5705”.

To summarise and simplify the information presented, simply if we think of how SQL injection discovered as an example; when the attacker adds a malicious select

query at the end of the website link “mutated samples”. If the website interacts with the malicious code “instructions”, meaning we can use different malicious queries “different mutated samples”, because the website is vulnerable “exceptions discovered” and therefore, different malicious code can be used which can be exploited afterwards. The same principles apply to the research target Sumatra PDF, when different instructions disclosed different faults, but within the same function because the function is vulnerable, as a result of the mutation option “Scope” selected when the samples was created at the first step. Figure 12 shows the mutation option selected when the seed samples were generated from my original test case (template).



Fig. 12 Mutation option selected to generate seed samples

The changes made to the main template are outlined in Fig. 12 for one of the samples which caused an access violation exception “3196.pdf” which was captured on the screenshot in Fig. 7. The following Figure refer to the main template selected to fuzz on my target “fuzz.pdf” “to investigate the location FileFuzz manipulate and was responsible to reveal that exception. From the Figs. 10 and 12, it can be noted that “.2” value highlighted below has been deleted from our 3196.pdf sample, which was responsible to cause crash on my research target (Fig. 13).

3196.pdf																fuzz.pdf x	
		Edit As:		Hex		Run Script		Run Template									
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF	
0000h:	25	50	44	46	2D	31	2E	33	0A	25	E2	E3	CF	D3	0A	32	%PDF-1.3.%aa10.2
0010h:	20	30	20	6F	62	6A	0A	3C	3C	0A	2F	43	72	65	61	74	0 obj.<<./Creat
0020h:	69	6F	6E	44	61	74	65	20	28	44	3A	32	30	31	30	30	ionDate (D:20100
0030h:	34	30	31	30	39	35	39	34	37	2D	30	37	27	30	30	27	401095947-07'00'
0040h:	29	0A	2F	4D	6F	64	44	61	74	65	20	28	44	3A	32	30	./ModDate (D:20
0050h:	31	30	30	34	30	31	30	39	35	39	34	37	2D	30	37	27	100401095947-07'

Fig. 13 Display the raw hex bytes for fuzz.pdf

The mutational-fuzzing required a template to start fuzzing, which meant that the fuzzer will not produce codes not exists at the mother sample. For instance, PDF file format used ten types of chunks but the main template which was selected used only five of them, FileFuzz cannot produce the other five missing chunks and therefore, only the available chunks are going to be modified.

Furthermore, the Peach shows a better result. In addition, to the method Peach used to record the vulnerabilities where a new folder created with the date and the time of the test, where “status.txt” file created and listed all the faults founded and in which iterations. Another subfolder “faults” created and listed each of the faults founded in different subfolders according to !exploitable debugger classifications for these faults, as summarised in Chart 2. This shows that generational-fuzzing approach, which was represented by Peach is better than FileFuzz in discovering and recording software flaws. The screenshot below for the mentioned status.txt file, shows the first fault details, which was highlighted in Fig. 9, and how Peach organised the discovered vulnerabilities (Fig. 14).

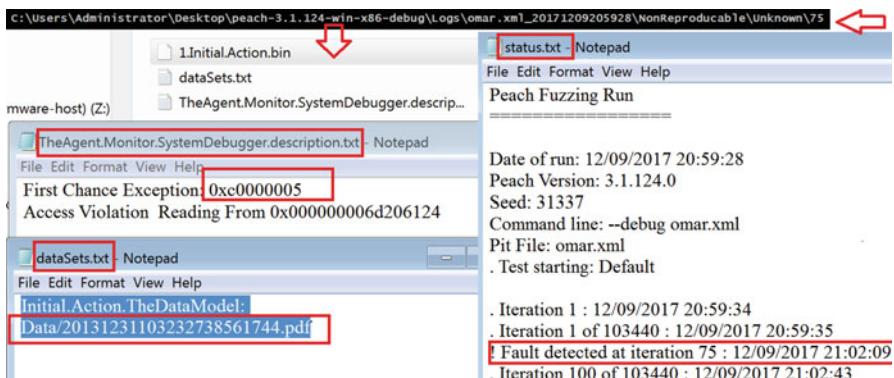


Fig. 14 First fault Peach detected

The case is not different while investigating Peach results, as explained before, the difference between mutational and generational approaches is based on the approach used to generate the samples required for fuzzing. However, Peach required some knowledge about the target functions and the test cases required to be downloaded, which led to a higher code coverage because of the sample varieties, which has been proven by the number of vulnerabilities discovered compared to the FileFuzz fuzzer.

The same procedures were followed again, to analyse several faults produced by Peach using PIN tool, Hex editor and Minset script to determine the coverage for each sample as in the Fig. 15 below. Comparison between both results shows that some mutated samples created the same traces as in normal sample.

```
C:\Users\Administrator\Desktop\peach-3.1.124-win-x86-debug>PeachMinset -k -s C:\fuzz\pdf\Faults_samples -t traces C:\SumatraPDF\SumatraPDF.exe args %s
} Peach 3 -- Minset
} Copyright (c) Deja vu Security
[*] Running trace analysis on 3 samples...
[00:00:00.2510736] (1:3) Coverage trace of C:\fuzz\pdf\Faults_samples\guide_c_noss.pdf... Completed
[00:00:09.6647788] (2:3) Coverage trace of C:\fuzz\pdf\Faults_samples\guidec_noss.pdf... Completed
[00:00:19.1319668] (3:3) coverage trace of C:\fuzz\pdf\Faults_samples\n_cv_2per_ree.pdf... Completed
[00:00:28.9709200] Finished
```

Fig. 15 Find the coverage for three of the samples produced crashes

This is because of the different smart Mutators Peach support when anomalies added to the test cases, in a way that it did not affect the majority of the samples input. In contrast to FileFuzz, which generate samples according to one line only for each of the samples, as it showed in Fig. 11, which justifies the large number 10,360 of the samples that have been created.

From both fuzzer results, it can be seen that the code coverage changed based on the test cases contents, which reflects on the target features and the number of faults discovered eventually.

4.4.2 Reconstruction of Test Cases

Based on the understanding which has been concluded from previous analyses and results which illustrate a positive correlation between code coverage and detection of vulnerabilities. At this section, five different PDF samples will be created, to analyse how different features can reflect on my fuzzing results.

Discover Different Contents Coverage

This task will start by analysing the coverage of a blank PDF sample which was generated, by running the sample and each further sample against Sumatra PDF using Pin tool and additional python script.⁷ The result will be generated in a text file according to the instructions each sample used during execution. The blank sample coverage will be used as a metric to compare it coverage with other created samples to extract and measure the differences easily.

First, all the instructions attached with Sumatra PDF will be generated again using Pin tool with additional support from the script mentioned above, to differentiate between the main functions/instructions and the new modules/instructions caused by the five new samples created. Figure 16 below shows the command used for this purpose. Five coverage.log file was created and contained all the target instructions.

⁷<https://github.com/Cr4sh/Code-coverage-analysis-tools> ... Code coverage analysis tools for PIN.

```
> execute_pin.bat "c:\SumatraPDF\SumatraPDF.exe"
```

Fig. 16 Sumatra PDF instructions recorded using PIN tool

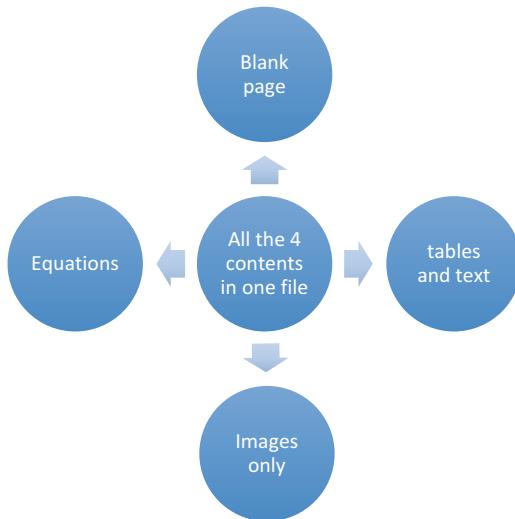
The second process, will be achieved when Immunity debugger attached to my target, run the blank PDF sample through Sumatra interface, then thanks to dynamic instrumentation PIN tool which will record all the executed functions in a text file. The following commands used to extract function calls information which related only to our target module, and to sort the result by the number of functions calls to figure later the best coverage among the different features created (Fig. 17).

```
> python coverage_parse.py Coverage.log --dump-routines --modules "SumatraPDF" --order-by-calls --outfile C:\Users\Administrator\Desktop\generated samples\samplecoverage.txt
```

Fig. 17 PIN tool commands

The second phase will be repeated for each of the samples created. The following figure provided a summary of the features for five different samples I have created using Microsoft word and saved as PDF.

Fig. 18 Sample features to be analysed



When all the created samples executed through Sumatra PDF application and their coverage extracted. The next step will be a comparison between each of the created samples shown in Fig. 18 against the empty PDF sample coverage which

was selected as a metric for this task, to discover how different features influence the code coverage.

Coverage Results and Analysis

Based on our target structure which endeavours of displaying PDF sample content on the application main interface. According to this fact, the sample with different features in theory will result in a better coverage as this will require more functions and resources to work in parallel to display such contents.

The following script used to analyse the five coverage log files, which will convert the coverage for all the created samples to call tree profile format, to extract and differentiate between each sample coverage. This process takes several minutes and the result will be created at “callgrind.out” file. To figure out the best feature which will trigger more function calls on my research target (Fig. 19).

```
> python coverage_to_callgraph.py Coverage.log *
```

Fig. 19 Analysing the five samples coverage

The coverage results for each of the created features produced after conducting all the steps mentioned above were it compared to the blank sample coverage as a metric and summarised into a simple Chart below.

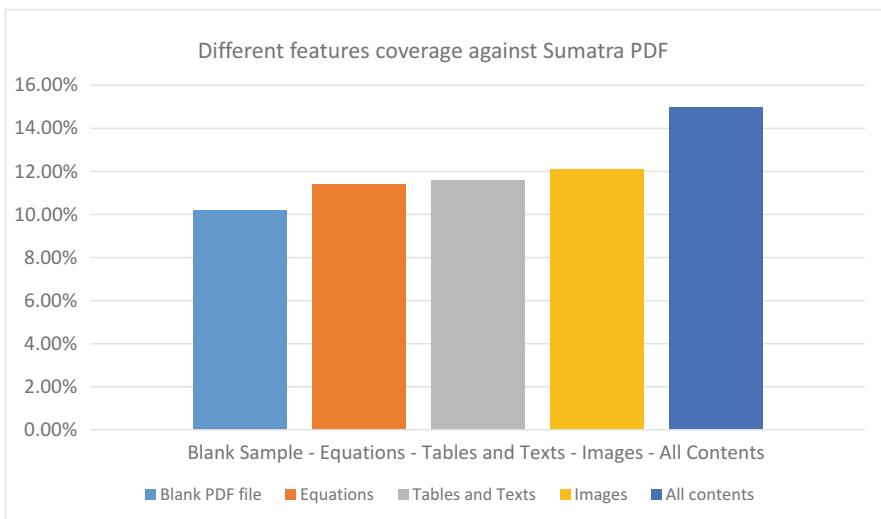


Chart 3 Different features coverage

Based on the results presented in the Chart 3, it can be confirmed that the more different content occupies at our sample, a better coverage will be granted. The PDF sample with no content used as a metric to compare other features with, which covers 10.2% of our target code. When a sample contained images only, which has increased the coverage by 1.9%, this mean that the highest coverage will be no more than $10.2 + 1.9 = 12.1\%$, of all the created samples, no matter which fuzzer was used. Despite that the result was not as expected, but the idea was proven. However, this could be illustrated as the samples were created from one page only. Therefore, the covered areas will be better if several pages were created with more complicated data. In addition, to our research target complexity structure (GUI) which contained from different functions that might not be easy to extract coverage from, comparing to a program with a console. What draws attention in the results is how the coverage changed based on the quality of the content which vary from one feature to another. This was proven when all the contents were built in one sample which shows the best code coverage for 15% among all other samples.

To achieve a better fuzzing result after conducting multiple experiments in this research, the samples should be reconstructed and the fuzzing should be performed according to the new enhanced dataset. However, in case of using FileFuzz, different created samples should be chosen after measuring the code coverage for each and the highest will be selected. In addition, different rounds of fuzzing are required to cover as much as possible at our target functions. The research results obtained can be the basis for further researches. Especially in the process of creating sophisticated and complex samples and to create advance fuzzer with the ability to concentrate the code modifications at specific locations inside the provided sample, such idea can be applied using Peach platform, as it allows users to create their own fuzzer.

5 Conclusion and Future Work

This study demonstrated an attempt to improve fuzzing software for anomalies and how show how test cases can play a crucial role for this purpose. This provides a scope for future development of fuzzers which can be used to fuzz applications running on emerging platforms such as embedded systems running real time operating systems (RTOS) built on Windows Operating Systems. For instance, the Windows IOT, formerly Windows Embedded takes on core features of the Windows preloaded on hardware devices. In this study, different features were created and analysed for two types of fuzzers. The first task was to create five new samples with different contents, then analyse the code coverage for each of these samples. Finally, we discovered the best feature which provides the highest coverage. The result presented in Chart 3, “Different features coverage”, shows clearly that for any feature added to test cases, it would be better than a sample with no contents. Results from the combined features yielded the highest coverage compared to other samples. However, before fuzzing is started, it is necessary to work on samples according to a clear understanding of the target’s functionalities. For instance, the

display contents of the target Sumatra PDF viewer are different to other document readers which reads the text from file input or console program. Therefore, the target and the sample format are important and extra consideration should be taken during the fuzz test, as this reduces the test time and the effort by inserting the correct mutated samples directly to the target functions.

Although performing generational-fuzzing approach using Peach, improves the effectiveness of mutational-fuzzing approach, it requires much effort, time and processing power. Although, FileFuzz generated the samples with one click, Peach required the samples to be download, and the time spent to complete the test is considerable higher than FileFuzz as demonstrated in Charts 1 and 2, “test Results”. The test cases generated by FileFuzz produce low code coverage and therefore less effectiveness when compared to Peach samples. In addition, FileFuzz requires one sample to begin fuzzing, while the other 10,360 samples created partially from the mother sample, as shown in the sample presented in Fig. 7. Peach required different samples to be downloaded with different features from different resources, which may not lead to realistic comparison. However, with consideration in the number of crashes discovered and the number of samples tested, it shows that some prior knowledge about the target application’s structure is required and the sample coverage which makes Peach fuzzer take precedence on this comparison. The approach could be implemented to enhance testing software embedded in IoT devices.

This work serves as a baseline for future research in which researchers could extend and develop datasets to include other samples and enhance the sample features by adding additional attributes. There are several areas of research that could be undertaken, for example: by testing the target based on a database of specific vulnerabilities built in different test cases and contained simple execution instructions.

References

1. D. Kiwia, A. Dehghantanha, K.-K. R. Choo, and J. Slaughter, “A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence,” *J. Comput. Sci.*, Nov. 2017.
2. Y.-Y. Teing, D. Ali, K. Choo, M. T. Abdullah, and Z. Muda, “Greening Cloud-Enabled Big Data Storage Forensics: Syncany as a Case Study,” *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2017.
3. Y.-Y. Teing, A. Dehghantanha, K.-K. R. Choo, and L. T. Yang, “Forensic investigation of P2P cloud storage services and backbone for IoT networks: BitTorrent Sync as a case study,” *Comput. Electr. Eng.*, vol. 58, pp. 350–363, Feb. 2017.
4. Y.-Y. Teing, A. Dehghantanha, and K.-K. R. Choo, “CloudMe forensics: A case of big data forensic investigation,” *Concurr. Comput. Pract. Exp.*, p. e4277, Jul. 2017.
5. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, “Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence,” *IEEE Trans. Emerg. Top. Comput.*, 2017.

6. H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, “Intelligent OS X malware threat detection with code inspection,” *J. Comput. Virol. Hacking Tech.*, 2017.
7. F. Norouzizadeh Dezfooli, A. Dehghantanha, B. Eterovic-Soric, and K.-K. R. Choo, “Investigating Social Networking applications on smartphones detecting Facebook, Twitter, LinkedIn and Google+ artefacts on Android and iOS platforms,” *Aust. J. Forensic Sci.*, pp. 1–20, Aug. 2015.
8. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, “Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing,” *Eurasip J. Wirel. Commun. Netw.*, vol. 2016, no. 1, 2016.
9. A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, “Machine Learning Aided Static Malware Analysis: A Survey and Tutorial,” 2018, pp. 7–45.
10. J. Baldwin and A. Dehghantanha, *Leveraging support vector machine for opcode density based detection of crypto-ransomware*, vol. 70. 2018.
11. O. M. K. Alhawi, J. Baldwin, and A. Dehghantanha, *Leveraging machine learning techniques for windows ransomware network traffic detection*, vol. 70. 2018.
12. S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha, and R. Khayami, “BoTShark: A Deep Learning Approach for Botnet Traffic Detection,” Springer, Cham, 2018, pp. 137–153.
13. M. Petraityte, A. Dehghantanha, and G. Epiphanou, “A Model for Android and iOS Applications Risk Calculation: CVSS Analysis and Enhancement Using Case-Control Studies,” Springer, Cham, 2018, pp. 219–237.
14. M. Conti, A. Dehghantanha, K. Franke, and S. Watson, “Internet of Things security and forensics: Challenges and opportunities,” *Futur. Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan. 2018.
15. M. Hopkins and A. Dehghantanha, “Exploit Kits: The production line of the Cybercrime economy?,” in *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, 2015, pp. 23–27.
16. H. Haddadpajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, “A Deep Recurrent Neural Network Based Approach for Internet of Things Malware Threat Hunting,” *Futur. Gener. Comput. Syst.*, 2018.
17. S. Watson and A. Dehghantanha, “Digital forensics: the missing piece of the Internet of Things promise,” *Comput. Fraud Secur.*, vol. 2016, no. 6, pp. 5–8, Jun. 2016.
18. A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, “Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning,” *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2018.
19. N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, “Machine learning aided Android malware classification,” *Comput. Electr. Eng.*, vol. 61, 2017.
20. G. Epiphanou, P. Karadimas, D. K. B. Ismail, H. Al-Khateeb, A. Dehghantanha, and K. R. Choo, “Non-Reciprocity Compensation Combined with Turbo Codes for Secret Key Generation in Vehicular Ad Hoc Social IoT Networks,” *IEEE Internet Things J.*, 2017.
21. H. Haddad Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo, “A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks,” *IEEE Trans. Emerg. Top. Comput.*, pp. 1–1, 2016.
22. G. Megraw, “Software security.” *IEEE Secur. Priv. Mag.*, vol. 2, no. 2, pp. 80–83, Mar. 2004.
23. A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, “Detecting crypto-ransomware in IoT networks based on energy consumption footprint,” *J. Ambient Intell. Humaniz. Comput.*, pp. 1–12, Aug. 2017.
24. S. Walker-Roberts, M. Hammoudeh, and A. Dehghantanha, “A Systematic Review of the Availability and Efficacy of Countermeasures to Internal Threats in Healthcare Critical Infrastructure,” *IEEE Access*, 2018.
25. A. Causevic, D. Sundmark, and S. Punnekatt, “An Industrial Survey on Contemporary Aspects of Software Testing,” in *2010 Third International Conference on Software Testing, Verification and Validation*, 2010, pp. 393–401.
26. P. Godefroid, M. Y. Levin, and D. Molnar, “Automated Whitebox Fuzz Testing,” 2008.

27. & A. G. Michael Sutton, Adam Greene, “Fuzzing: Brute Force Vulnerability Discovery - Google Books.” 2007.
28. Charlie Miller and Zachary N.J. Peterson, “Analysis of Mutation and Generation-Based Fuzzing,” 2007.
29. John Neystadt, “Automated Penetration Testing with White-Box Fuzzing.” 2008.
30. P. Godefroid, M. Y. Levin, and D. Molnar, “SAGE: Whitebox Fuzzing for Security Testing SAGE has had a remarkable impact at Microsoft. THE HIGH COST OF SECURITY BUGS A Sample JPG Image,” 2012.
31. V. W. Cadar Cristian, Godefroid Patrice, Khurshid Sarfraz, Corina S. Pasareanu, Sen Koushik, Tillmann Nikolai, “Symbolic Execution for Software Testing in Practice – Preliminary Assessment,” 2011.
32. A. Rebert, J. Foote, J. Org, D. Warren, and D. Brumley, “Optimizing Seed Selection for Fuzzing.”
33. Microsoft, “Introduction to Instrumentation and Tracing.” 2013.
34. P. Feiner, A. D. Brown, and A. Goel, “Comprehensive Kernel Instrumentation via Dynamic Binary Translation,” 2012.
35. A. Takanen, J. DeMott, and C. Miller, “Fuzzing for Software Security Testing and Quality Assurance (Artech House Information Security and Privacy),” 2008.
36. PeachTech, “Peach Fuzzer: Discover unknown vulnerabilities.”
37. R. Luk, Chi-Keung, Cohn, Robert, Muth, G. Patil, Harish, Klauser, Artur, Lowney, and K. Vijay, Steven Wallace, Reddi, Janapa, Hazelwood, “Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation,” 2005.
38. J. S. Dave Weinstein, “The History of the !exploitable Crash Analyzer – Security Research & Defense.” 2009.
39. L. de Moura and N. Bjørner, “Z3: An Efficient SMT Solver,” 2008, pp. 337–340.

Bibliometric Analysis on the Rise of Cloud Security



Lim Sze Thiam, Tooska Dargahi, and Ali Dehghantanha

Abstract Cloud storage systems are becoming a gold mine for cyber attackers as they storage a lot of private and confidential information. Therefore, a lot of research is directed toward securing cloud platforms. Majority of cloud security related research or studies are emphasising on looking into prevention, detection, and mitigation methods against the threat in cloud computing. Although there are many research studies in this area, there are still no evidence of any bibliometric analysis on the cloud security context. This paper provides a bibliometric analysis of research development in cloud security from 2010 to 2018. A dataset of 656 academic papers are used for analysis in term of most productive countries, institutions, journals, authors, research areas, keywords, and also highly cited articles. We found that countries of China, India and United States are the top contributor of the findings. Trends in number of publication in the area of cloud security increases and the number of related keywords increases as well.

Keywords Cloud computing · Security · Bibliometric analysis · Cloud security

L. S. Thiam

School of Computing, Science and Engineering, University of Salford, Manchester, UK
e-mail: L.SzeThiam@edu.salford.ac.uk

T. Dargahi

Department of Computer Science, School of Computing, Science and Engineering, University of Salford, Manchester, UK
e-mail: T.Dargahi@salford.ac.uk

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada
e-mail: ali@cybersciencelab.org

1 Introduction

Cloud computing has been precipitously emerged as a proficient concept of approach for service and distribution [1]. The National Institute of Standards and Technology (NIST) defines cloud computing as a delivery model for feasible network access to a public, configurable group of computing data resources with effortless interaction with the service provider or management [2]. The three main cloud service models are: infrastructure as a service (IaaS), software as a service (SaaS), and platform as a service (PaaS) [3]. In IaaS virtualization technology is used to allocate computing resources, capabilities and also basic storages for implementing services [4]. SaaS, models provide a platform for developing software application over the cloud [5]. Lastly, Platform as a service (PaaS) is a model that offers freedom to build or develop any application and runs on the provider's high-end infrastructure [6].

Through its unique advantages, cloud computing facilitates a vital paradigm change in the deployment and delivery of computing services. Organizations and users can escape from devoting huge amount of money and resources when buying and handling software and hardware as well as maintenance [7]. Due to these circumstances it is prominent to have issues interconnected with the security aspects of cloud computing as it becomes more vulnerable to security threats such as unwanted breaches, classic issues which is data's confidentiality, integrity and also availability, Denial of Service (DoS) and so on [8]. Thus, as the popularity for cloud computing increases, the security issues and challenges affecting it are not far behind [9].

Although there are existing prevention such as antivirus, firewall technology on every end point, blocking vacant services, ports, Intrusion Detection System (IDSs) and so on to overcome known security issues but there are always some threats that cloud computing cannot avoid to occur due to its methods and characteristics of its process [10]. Securing cloud computing cannot be measured as a single step course of action but as a continuous process which indicates an obligatory to analyse security of the computing as a necessary practice [11]. There are many research papers related to cloud security and forensics i.e. [12–14] but most of them are addressed in single attribute for instance, infrastructure issues, data issues, and auditing.

Bibliometric is a useful tool for assessing research trends in various science fields [15]. This approach has been applied to evaluate the scientific outcomes or research relationships of authors, journals, countries, institutes and also to categorise and enumerate international cooperation. While the orthodox bibliometric approach mainly focus on citation and content analysis, the newly established bibliometric analysis appraises the academic outputs of authors, countries, institutes and pinpoints the chronological evolution of research patterns [16]. In this research paper, a bibliometric analysis is conducted on the rise of cloud security.

The remainder of this paper is structured as follows, a methodology section that illustrates the method used to conduct the study and a section that analyses and also

visualize the outcome of the studies and lastly a final section of the summary of the research which hopefully might provide an impending bench mark for future research.

2 Methodology

Bibliometric Analysis is used in this research paper is a quantitative method which is for displaying trends and growths in the production of science and technology [17]. The information provided by the bibliometric analysis is upon the publication of scientific research and also determine how effective the impact of researchers, organization or institutes is [18]. There are a few studies of innovation and technological rely on this methodology as part of their analysis, for example, analysis of agro-geoinformatics for the period 1991–2015 to evaluate different perspective and trends [16], analysis of fuzzy decision making where decision is made during complex or uncertain environment [19], study on the trends on XBRL for the period of 2000–2014 where it is a new concept of XML and financial statements [17], study on the rise of malware which is about a malicious software that threatens internet users[18].

For the development of this paper, we have chosen to obtain information from Web of Science database as our core data collection. This database also include various databases such as IEEE explorer, ACM, and Elsevier and so on. Web of Science (WoS) was chosen as it hosts almost all research fields and the information contains more than 50 million vary in category like journals, books, conferences and articles [20]. The quality of publications indexed by this database are widely recognized as is it used among institutes around the world.

The research process begins with using the keyword “cloud computing security” and “cloud security” as search terms for all publications which contains related words in abstracts and title. Furthermore, the search was limited between the period from 2010 and 2018. Reason being that the popularity of the topic cloud computing emerged during the early 2000s therefore we considered the period of 2010 up to 2018, it is because that the quality of the data before 2010 was not very good. As a result, the total number of publications related to cloud security found for the past 8 years are 753 in various format such as articles, journal, books, and conferences. Non-related publications for instances Korean Journal Database –KCI, SciELO Citation Index, Derwent Innovations Index and other non-English publications are excluded and a total of 656 articles remains for analysis purpose. The results of the search obtained for the publication was assessed within the preferred search parameters. The criteria used for analysing are as below:

- Number of publications per year
- Number of publications by country
- Number of publications by author
- Number of publication by highest cited articles
- Number of publication by research area.

Lastly, Cite Space software is used to visualize all the results from the analysis. The software is available online for no cost and comprised of wonderful features and also support numerous types of bibliographic for analysis. Figure 1 below shows the flow chart of the research process.

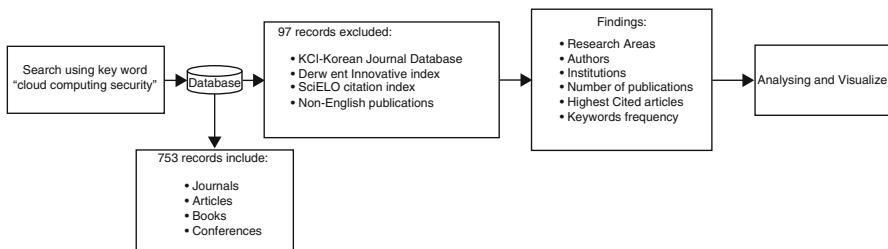


Fig. 1 Flow chart

3 Findings

This section deliberates about the findings which related to the topic cloud security. The findings is segregated into five sub topics which are research areas, highly cited articles, authors, number of publications, keywords frequency and institutions. These findings are essential for evaluating publishing ranks through bibliometric data. On top of that, it is also capable to sort out quality research which may assists in producing more knowledge thus to confirm that the interest in cloud security is more comprehensive.

3.1 Number of Publications

Figure 2 illustrates the number of publications and citations over the last 8 years. In 2017, the number of publications have dropped slightly. This is probably due to the long duration of journals in acknowledging articles for publications hence the number of publications of the alleged year is affected. It is most likely that the number of publications will increase in trend for the year of 2018. Besides that, the increase of publications are capable of increasing the number of citations.

Citation evaluation is utilised to weigh the rate of journals records mined from the citation index. It is treated as an evaluation of researcher's performance based on the patterns of citation. Moreover, shared references allow a researcher to gain more information from other researchers on the topic researched. Looking at the

publications over the past 8 years indicate that the number of citations are greatly influenced by the quantity of publications.

The amount of citations gathered by published articles is averagely around 460 per year during the period of 2010 till 2018. Furthermore, from 2015 to 2018 shows a progressive propagation with two similar pinnacle taking place in 2016 and 2017. There is an increase of 30.76% citations from 2015 to 2017. Perhaps during 2015, there was rise of studies steered to resolve the cloud's security issue. The increment of citation is also due to scholars citing other scholars' papers as references. This trend elucidates a progressive results until 2017, while 2018 is still in progress.

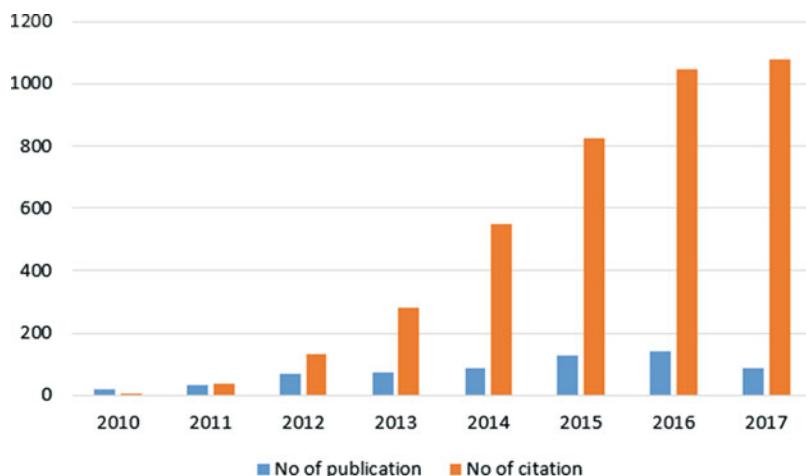


Fig. 2 Bar chart for number of publications and citation per year

3.2 Research Areas

This section is about the amount of publications comprising a variety of research areas. Research areas involve a sole ordered and versatile ordered whilst they intend to acquire a logical perception of an explicit research areas and how they contest other area in diverse parts of various industries. The occurrence of whichever research area is likely for displaying the publication's development over time. The WoS database holds an index of numerous divisions containing more than 100 scientific research areas such as Computer Science, Communication, Mathematics, and so on.

Table 1 points out that the widely held published articles are from Computer Science and Engineering research area. Incidentally, the two areas are a combination which regard as a source in developing new approaches and expertise that beneficial

to all community. Machine learning, software engineering, data structures and algorithms, cyber security are examples of sub areas of Computer Science and Engineering. The leading article including the most citation in the Computer Science's area was "A survey on security on security issues in service delivery models of cloud computing". As topics on cloud security involves virtualization, access controls, multi layered security, and etcetera as an approach, thus it is noticeable that there is a surge in publications generated under the Computer Science area.

Table 2 demonstrates the records of publications based on the geographical area, it appears that most publications came from the country of China and India. On the other hand, not all articles published contains the full address, hence the part for country is incapable to attain the full results. Altogether, this findings illustrates that the rate of recognition for the country China and India is good enough as their scholars have the capability to mould the articles based on the journal emphases, structures and areas.

Table 1 Cloud security research areas

Research areas	Publication	%
Computer Science	777	77.86
Engineering	385	35.58
Telecommunication	181	18.14
Automation Control Systems	45	4.51
Business Economics	20	2.0
Science Technology	17	1.7
Education research	15	1.5
Materials Science	12	1.2
Robotics	11	1.1
Operation Research Management Science	9	0.9
Energy Fuels	6	0.6
Mathematics	6	0.6
Social Science	6	0.6
Information Science	5	0.5
Environmental Science	4	0.4
Mathematical Computational Biology	4	0.4
Radiology Nuclear Medicine	4	0.4
Transportation	4	0.4
Biotechnology Applied Microbiology	3	0.3
Imaging Science Photographic Technology	3	0.3
Instruments Instrumentation	3	0.3
Biochemistry Molecular Biology	2	0.2
Food Science	2	0.2
Optics	2	0.2
Pharmacology Pharmacy	2	0.2

Table 2 Cloud research according to country

Countries/regions	Publications	%
China	182	27.7
India	166	25.3
USA	68	10.3
England	35	5.35
South Korea	30	4.57
Morocco	18	2.74
Malaysia	17	2.59
Saudi Arabia	16	2.43
Pakistan	14	2.13
Brazil	13	1.98
Australia	12	1.82
France	10	1.52
Iran	10	1.52
Japan	10	1.52
Spain	10	1.52
Taiwan	10	1.52
Canada	8	1.22
Italy	8	1.22
Germany	7	1.06
Tunisia	7	1.06
Greece	6	0.91
Turkey	5	0.76
Bangladesh	5	0.76
Egypt	4	0.61
Finland	4	0.61
Jordan	4	0.61
Lebanon	4	0.61
Austria	4	0.61
Israel	3	0.45
Kuwait	3	0.45
New Zealand	3	0.45
Russia	3	0.45
Singapore	3	0.45

3.3 Institutions

This section is to distinguish the number of publications corresponding to the institutions. The purpose of this section is to determine the quality of research and also to pinpoint which institutions are active by assessing institutions corresponding to the paper [21] conducted by a group of researcher from the University of Granada. Table 3 shows the list of institutions that done research associated with cloud

security. It also shows that the institutions are from various country such as China, United States, Malaysia, Australia, Finland, India and also Hong Kong.

It appears that the most number of publications came from China. India holds the second most number of publications and followed by United States. Apparently, the University of Xidian, China possesses the most publications. It looks like most of the renowned institutions are also from China and possibly that the rate of publication is significantly faster than other countries. Last but not least, having reliable facilities allow researchers to produce good quality research and in turn grants the institutions in China the most active article publisher.

Table 3 Number of publications according to institutions

Institutions	Publications	Country	%
XiDian University	112	China	2.59
Chinese Acad Science	85	China	1.96
Beijing University Telecommunication	59	China	1.36
King Saud Uni	55	Saudi Arabia	1.27
Nanjing University Science	48	China	1.11
Guangzhou University	45	China	1.04
University Elect Science China	45	China	1.04
Vit University	43	India	0.99
Wuhan University	39	China	0.90
Fujian Normal University	37	China	0.85
Nanjing University Telecommunication	29	China	0.67
Natl Inst Technology	28	India	0.64
Beihang University	27	China	0.62
Huazhong University	27	China	0.62
Texas University San Antonio	25	United States	0.57
Anna University	23	India	0.53
Deakin University	23	Australia	0.53
Natl University Defence Technology	23	India	0.53
Shandong University	23	China	0.53
University Malaya	23	Malaysia	0.53
Aalto University	22	India	0.51
City University Hong Kong	22	Hong Kong	0.51
Penn state University	22	United States	0.51
Shanghai Jiao Tong University	22	China	0.51
Thapar University	22	India	0.51

3.4 Authors

This section is regarding the number of publications by authors from various country. The goal of this section is to show which country's authorship is the most active. Table 4 illustrates the list of the most productive authors according to countries. As shown in Table 4, most of the authors are from China. Besides, there are quite a number of contribution from India, Saudi Arabia and United States. However, authors from Malaysia, Hong Kong, and Finland have lesser publications produced. So it implies that authors from India Saudi Arabia and United states are fairly active.

Table 4 shows that Li J from China and Chen XF also from China are the two top authors that contribute the most publications while the rest of the publications are less than 0.56%. “Timer: secure and reliable cloud storage against data re-outsourcing” [22] has won the best paper awards during the 10th International Conference on Information Security Practice and Experience(ISPEC 2014) which are related to the topic of cloud security. In addition, both of the authors also have outstanding honours and awards as a scholar which benefit them in publishing articles.

Table 4 Number of publications according to authors

Authors	Publications	Country	%
Li J	49	China	0.91
Chen XF	32	China	0.59
Li H	30	China	0.56
Zhang Y	30	China	0.56
Ma JF	25	China	0.46
Choo KKR	22	United States	0.41
Shen J	22	China	0.41
Wang J	22	China	0.41
Liu Q	21	China	0.39
Khan MK	20	Saudi Arabia	0.37
Buyya R	19	India	0.35
Essaaidi M	19	Saudi Arabia	0.35
Huang XY	19	China	0.35
Park JH	17	United States	0.31
Zbak M	17	India	0.31
Wang C	17	China	0.31
Chang V	17	China	0.31
Gani A	16	Malaysia	0.29
Sing S	16	India	0.29
Khan SU	16	India	0.29
Salah K	15	India	0.28
Gai KK	15	United States	0.28

3.5 Highly Cited Articles

The number of citations obtained by the journals will be discussed in this section. Furthermore, the quality of research carried out and the effect it has on alike topics. Top twenty five of the most cited articles are listed as shown in Table 5. The information on Table 5 includes the titles of the articles, total citation, source which is the published journals, and the publication year. As stated in the conception that when the articles are inside the database for a long time, the number of citations will pile up. This is shown as where the top four most cited articles were published in 6–8 years ago.

Looking at the list of published articles, the highest cited article is “A survey on security issues in service delivery models of cloud computing”. Briefly, the article describes about the security issues on the three delivery models of cloud computing which are also the main functions. Furthermore, summary of current security solutions are discuss here as well. Not only that, the authors are intending to develop a storage framework that may focused on providing data security and accessing based on meta-data. From this, we can see that not only this article can be served as a quick reference on current knowledge of security in cloud computing but also more information on intending solutions. Thus, this shows that articles that are highly cited are no average articles but deemed as good quality articles whereby other researchers or scholars recognized other author’s works. Other than that, articles which contain appealing topics are also able to contribute in increasing of citations particularly when the issues become popular.

3.6 Keywords Frequency

This segment will discuss about the category of keywords that researchers used regularly. This part is valuable as it allows articles to be identified including prior and also present issues of journals’ research. Around 1990, Web of Science started to make available of keywords and a theme that describes the article [23]. The keywords and titles are able to use for distinguishing research gaps and also to study the research trends. Table 6 shows the record of distinct keywords and titles obtained from all the articles.

Based on Table 6, it seems that the majority of titles and keywords are virtualization, encryption and so on. It shows that these titles and keywords are often used by researchers, for instance the title of articles “secure virtualization for cloud computing” and “cloud security defence” contains the term “cloud computing” and “cloud security”.

Table 5 Top 25 highest cited articles

Titles	Source	Publication year	Total citations
A survey on security issues in service delivery model of cloud computing	Network and computer applications	2011	735
Addressing cloud computing security issues	Grid computing and science	2012	479
Security and Privacy Challenges in Cloud Computing Environments	Security and privacy	2010	333
Enabling Public Auditability and Data Dynamics for storage security in cloud computing	Parallel and distributed system	2011	303
Security challenges for public cloud	Internet computing	2012	196
Security and Privacy for storage and computation in cloud computing	Information sciences	2014	170
Security and Privacy in cloud computing	Communications surveys and tutorials	2013	131
Security in Cloud computing: opportunities and challenges	Information sciences	2015	129
Beyond lightning: A survey on security challenges in cloud computing	Computers and electrical engineering	2013	99
Security issues in cloud environments: a survey	Information security	2014	97
Identifying security risk of cloud computing	Government information quarterly	2010	96
Cloud computing adoption framework: a security framework	Network and computer applications	2016	81
Cloud security defence against Http Dos	Intelligent transportation systems	2011	81
Security Challenges in Vehicular cloud computing	Systems and software	2013	80
Cloud computing security scientific challenges and solutions	Supercomputing	2013	78
A survey on security issues and solutions at different layer of cloud computing	Computing	2013	78
Locking the sky: a survey on IaaS cloud security	Grid computing and science	2011	71
CyberGuarder a virtualization security assurance	Network and computer applications	2012	55
A combined approach to ensure data security in cloud computing	Computer and system science	2012	54
A security framework across distributed cloud data centres	Education	2014	45
Cloud based virtual laboratory for network security education	CEIS 2011	2014	44
Surveying and analysing security, privacy and trust issues in cloud environments	Service computing	2011	43
Towards achieving data security within the cloud computing framework	Telecommunication policy	2016	41
Privacy and security issues in cloud computing	Communications	2013	39
New approaches to security and availability for cloud data	Computer and system science	2013	39

Table 6 Keywords frequency

Keywords	Frequency
Cloud Security	467
Cloud Computing	325
Security	132
Cloud Computing Security	82
Information Security	29
Data Security	15
Mobile Cloud Computing	14
Virtualization	293
Privacy	10
Private Cloud	10
Framework	8
Hypervisor	6
Encryption	110
Authentication	98
Data Privacy	76
Cryptography	71
Deployment Model	64
Trust	62
Integrity	51
Service	49
Storage Security	45

Figure 3 is provided for more detailed analysis. The keywords are extracted from the contents of the articles and the word map is generated using a software called “Cite Space”. It also comprises by clusters which express the growth of research related to cloud security. There are five main clusters which includes cloud computing, security, cloud security, data security and privacy. Each of the cluster is then further break down into their own sub clusters for example, data security as the main cluster, its sub clusters includes threats, data protection, data encryption and so on. These clustering or categories implies that many researchers had carried out on researching topics such as type of cloud security issues, infrastructures, detection tools and techniques. In the next section, more analysis is done based on the clustering.

4 Discussion

Cloud computing has emerged as an advanced paradigm which consists of all the fundamental elements of computing for instance, network communications, management systems, end user’s machine and so on [24]. However, such attractive platform prone to security issues and is a prime target for cyber criminals [25]. Cloud data and infrastructure have to be safeguarded against attacks throughout all

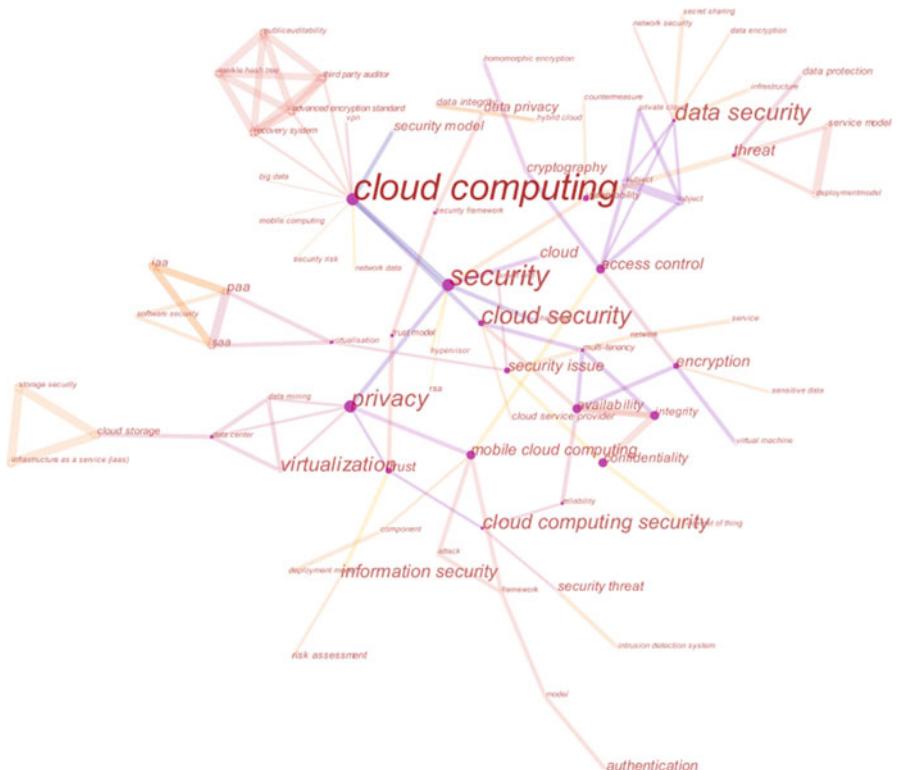


Fig. 3 Map of keywords frequency

the components [26]. Cloud security does not only face challenge on delivering better security but also need to work out with lower cost and moderating performance issues.

This paper helps to provide precautionary methods in protecting the cloud services, data, and infrastructure. The information regarding system security should be implemented in the cloud architecture to attain better security. Furthermore, policy updates should be done regularly to match up the fast changing cloud infrastructure. Besides that, sharing statistics of attacks publically can help other cloud security professional to protect themselves. Lastly, cloud computing's backbone, virtualization is still an open target as it is still contains many flaws such as VM rollback attacks, data leakage, service and account hijacking. For that reason, it is significant to improve the design which includes ample of security processes.

Existing body of research is mainly focused on the performance evaluation and analysis of cloud services in terms of cost effectiveness. However, other traits such as quality of service has been greatly overlooked [27]. Moreover, cloud computing is lacking of tools that can guarantee user data protection when the service contract is ended and proper deletion of all data [28]. This leads to some issues regarding

user data privacy and confidentiality [29]. There are other challenges related to cloud infrastructure when users access the cloud through mobile platforms [30]. Mobile platforms are more prone to threats because they have weak security on the application's development part, an example is discussed in the “access category section” [31]. Around twenty percent of android applications allows third party applications to retrieve data, managing its contents without user knowledge [32]. Therefore, mobile security issues and threats such as malware attacks, could easily become a cloud security problem as well [33]. Moreover, integration of Internet of Things (IoT) devices and data into cloud computing significantly increases attack surface and further complicates cloud security activities [34, 35].

5 Conclusion

Cloud computing has emerged as one of the fast growing technologies. It offers infrastructures or services which are convenience and not costly however such attractive offer comes with risk as well. The risk for this futuristic technologies are non-other than security issues, threats and vulnerabilities. Not like conventional solution whereby the source of threats only includes in the internal and external of the networks. The bibliometric analysis is used in this paper for analysing the trend of cloud security from 2010 to 2018. Research areas, number of publications, highly cited articles, frequency of keywords, institutions and author are the six criteria described and presented in this study. All the six criteria aided in disclosing the developments related to cloud security. There is an average of 15% annual increment of number of publications relating to cloud security. Thus, the findings also indicate the development of cloud security also increases the citation and article published.

Furthermore, the findings are also being analysed according to the institutions and it is found that the institutions from China contributes the highest production of publication relating to cloud security. Other than that, this paper also shows the findings for authors who are the most active for publication contribution. It is observed that the top contributors are Li J and Chen XF from China. Following, includes map of frequency used keywords for illustrating the developments direction for cloud security research. “Cloud Computing”, “security”, “privacy”, “virtualization” are some of the keywords used in cloud security research.

References

1. Y.-Y. Teing, A. Dehghantanha, K. Choo, M. T. Abdullah, and Z. Muda, “Greening Cloud-Enabled Big Data Storage Forensics: Syncany as a Case Study,” *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2017.
2. Z. P. Gariba and J. A. Van Der Poll, “Security Failure Trends of Cloud Computing,” *2017 IEEE 3rd Int. Conf. Collab. Internet Comput.*, no. Vm, pp. 247–256, 2017.

3. F. Daryabar, A. Dehghantanha, and K.-K. R. Choo, "Cloud storage forensics: MEGA as a case study," *Aust. J. Forensic Sci.*, pp. 1–14, Apr. 2016.
4. M. Sharifiati, A. Dehghantanha, and K.-K. R. Choo, "SugarSync forensic analysis," *Aust. J. Forensic Sci.*, vol. 48, no. 1, pp. 95–117, Apr. 2015.
5. Y.-Y. Teing, D. Ali, K.-K. R. Choo, and M. Yang, "Forensic investigation of P2P cloud storage services and backbone for IoT networks: BitTorrent Sync as a case study," *Comput. Electr. Eng.*, no. 2016, 2016.
6. F. Daryabar, A. Dehghantanha, B. Eterovic-Soric, and K.-K. R. Choo, "Forensic investigation of OneDrive, Box, GoogleDrive and Dropbox applications on Android and iOS devices," *Aust. J. Forensic Sci.*, vol. 48, no. 6, pp. 615–642, Nov. 2016.
7. Y.-Y. Teing, A. Dehghantanha, and K.-K. R. Choo, "CloudMe forensics: A case of big data forensic investigation," *Concurr. Comput. Pract. Exp.*, p. e4277, Jul. 2017.
8. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing," *Eurasip J. Wirel. Commun. Netw.*, vol. 2016, no. 1, 2016.
9. Y.-Y. Teing, A. Dehghantanha, K.-K. R. Choo, T. Dargahi, and M. Conti, "Forensic Investigation of Cooperative Storage Cloud Service: Symform as a Case Study," *J. Forensic Sci.*, vol. 62, no. 3, 2017.
10. A. A. James Baldwin, Omar Alhawi, Simone Shaughnessy and A. Dehghantanha, *Emerging from The Cloud: A Bibliometric Analysis of Cloud Forensics Studies*. Cyber Threat Intelligence- Springer Book, 2017.
11. T. Dargahi, A. Dehghantanha, and M. Conti, *Investigating Storage as a Service Cloud Platform: PCloud as a Case Study*. 2016.
12. S. H. Mohtasebi, A. Dehghantanha, and K.-K. R. Choo, *Cloud Storage Forensics: Analysis of Data Remnants on SpiderOak, JustCloud, and pCloud*. 2016.
13. A. Dehghantanha and T. Dargahi, *Residual Cloud Forensics: CloudMe and 360Yunpan as Case Studies*. 2016.
14. M. Amine Chelihi, A. Elutilo, I. Ahmed, C. Papadopoulos, and A. Dehghantanha, *An Android Cloud Storage Apps Forensic Taxonomy*. 2016.
15. J. Gill, I. Okere, H. HaddadPajouh, and A. Dehghantanha, *Mobile forensics: A bibliometric analysis*, vol. 70. 2018.
16. H. L. Dqj et al., "Global Research Trends in Agro-Geoinformatics during 1991-2015: A Bibliometric Analysis," *2017 6th Int. Conf. Agro-Geoinformatics*, 2017.
17. M. El Ansary and M. Oubrich, "State of the art and trends of the research on XBRL bibliometric analysis from 2000-2014," *Colloq. Inf. Sci. Technol. Cist*, pp. 243–250, 2017.
18. M. F. A. Razak, N. B. Anuar, R. Salleh, and A. Firdaus, "The rise of 'malware': Bibliometric analysis of malware study," *J. Netw. Comput. Appl.*, vol. 75, pp. 58–76, 2016.
19. F. Blanco-Mesa, J. M. M. Lindahl, and A. M. Gil-Lafuente, "A bibliometric analysis of fuzzy decision making research," *2016 Annu. Conf. North Am. Fuzzy Inf. Process. Soc.*, pp. 1–4, 2016.
20. J. M. Merigo, F. Blanco-Mesa, A. M. Gil-Lafuente, and R. R. Yager, "A bibliometric analysis of the first thirty years of the International Journal of Intelligent Systems," *2016 IEEE Symp. Ser. Comput. Intell.*, pp. 1–6, 2016.
21. G. Buela-Casal, O. Gutiérrez-Martínez, M. P. Bermúdez-Sánchez, and O. Vadillo-Muñoz, "Comparative study of international academic rankings of universities," *Scientometrics*, vol. 71, no. 3, pp. 349–365, 2007.
22. T. Jiang, X. Chen, J. Li, D. S. Wong, J. Ma, and J. K. Liu, "Towards secure and reliable cloud storage against data re-outsourcing," *Futur. Gener. Comput. Syst.*, vol. 52, pp. 86–94, 2015.
23. J. Sun, M. H. Wang, and Y. S. Ho, "A historical review and bibliometric analysis of research on estuary pollution," *Mar. Pollut. Bull.*, vol. 64, no. 1, pp. 13–21, 2012.
24. B. Blakeley, C. Cooney, A. Dehghantanha, and R. Aspin, "Cloud Storage Forensic: hubiC as a Case-Study," in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2015, pp. 536–541.

25. M. Hopkins and A. Dehghanianha, “Exploit Kits: The production line of the Cybercrime economy?,” in *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, 2015, pp. 23–27.
26. Y.-Y. Teing, A. Dehghanianha, K.-K. R. Choo, Z. Muda, M. T. Abdullah, and W.-C. Chai, “A Closer Look at Syncany Windows and Ubuntu Clients’ Residual Artefacts,” in *Security, Privacy and Anonymity in Computation, Communication and Storage*, G. Wang, I. Ray, J. M. A. Calero, and S. M. Thampi, Eds. Springer International Publishing, 2016, pp. 342–357.
27. H. Lv and Y. Hu, “Analysis and research about cloud computing security protect policy,” *Proc. - 2011 Int. Conf. Intell. Sci. Inf. Eng. ISIE 2011*, pp. 214–216, 2011.
28. M. Shariati, A. Dehghanianha, B. Martini, and K.-K. R. Choo, “Chapter 19 - Ubuntu One investigation: Detecting evidences on client machines,” in *The Cloud Security Ecosystem*, R. K.-K. R. Choo, Ed. Boston: Syngress, 2015, pp. 429–446.
29. A. Dehghanianha and K. Franke, “Privacy-respecting digital investigation,” in *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, 2014, pp. 129–138.
30. T. Y. Yang, A. Dehghanianha, K. R. Choo, and Z. Muda, “Windows Instant Messaging App Forensics: Facebook and Skype as case studies,” *PLoS One*, Feb. 2016.
31. F. Norouzizadeh Dezfooli, A. Dehghanianha, B. Eterovic-Soric, and K.-K. R. Choo, “Investigating Social Networking applications on smartphones detecting Facebook, Twitter, LinkedIn and Google+ artefacts on Android and iOS platforms,” *Aust. J. Forensic Sci.*, pp. 1–20, Aug. 2015.
32. M. Damshenas, A. Dehghanianha, K.-K. R. Choo, and R. Mahmud, “M0Droid: An Android Behavioral-Based Malware Detection Model,” *J. Inf. Priv. Secur.*, Sep. 2015.
33. N. Milosevic, A. Dehghanianha, and K.-K. R. Choo, “Machine learning aided Android malware classification,” *Comput. Electr. Eng.*, vol. 61, 2017.
34. M. Conti, A. Dehghanianha, K. Franke, and S. Watson, “Internet of Things security and forensics: Challenges and opportunities,” *Futur. Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan. 2018.
35. S. Watson and A. Dehghanianha, “Digital forensics: the missing piece of the Internet of Things promise,” *Comput. Fraud Secur.*, vol. 2016, no. 6, pp. 5–8, Jun. 2016.

A Bibliometric Analysis of Botnet Detection Techniques



Shehu Amina, Raul Vera, Tooska Dargahi, and Ali Dehghantanha

Abstract Botnets are rising as a platform for many unlawful cyber activities such as Distributed Denial of Service (DDoS) attacks, malware dissemination, phishing, click fraud, and so on. As of late, detecting botnet has been an intriguing research topic in relation to cybercrime analysis and cyber-threat prevention. This paper is an analysis of publications related to botnet detection techniques. We analyse 194 botnet related papers published between 2009 and 2018 in the ISI Web of Science database. Seven (7) criteria have been used for this analysis to detect highly-cited articles, most impactful journals, current research areas, most active researchers and institutions in the field. It was noted that the average number of publications related to botnet detection have been reduced recently, which could be because of overwhelming existing literature in the field. Asia is the most active and most productive continent in botnet research and computer science is the research area with most publications related to botnet detection as expected.

Keywords Bibliometric analysis · Botnet detection · Distributed Denial of Service · DDoS · Malware

1 Introduction

A botnet is a group of devices called bots interconnected over the Internet that have been compromised via malicious software [1, 2]. These devices can be computers, mobile devices [3], and even Internet of things (IoT) devices [4], which are remotely

S. Amina · R. Vera

School of Computing, Science and Engineering, University of Salford, Manchester, UK

T. Dargahi

Department of Computer Science, School of Computing, Science and Engineering, University of Salford, Manchester, UK

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada

e-mail: ali@cybersciencelab.org

accessible by the intruder [5]. Among the many reasons botnets are created for, the most representative ones are launching distributed denial of service (DDOS) attacks [6], phishing and ransomware distribution [7], identity theft, or using the powerful computational resources of the network on a wide range of malicious distributed tasks [8]. Although botnets have been on the rise for about two decades [9], they still represent one of the biggest challenges that security researchers and analysts must face [10]. The economic lost caused by the breach of digital networks rise as much as several millions of dollars every year [11].

Several types of research on botnets detection techniques have been developed based on different botnets characteristics, including their communication protocols (HTTP, P2P, IRC, etc.) [12], target platforms [13] and size of the botnet [14, 15]. These approaches have provided cybersecurity analysts with resources like network tracing and packet capturing tools for detecting and mitigating botnet attacks [16]. However, even though these techniques have helped to prevent several attacks, many researchers on the field have continued being carried out due to the non-stoppable increment of botnets attacks, originating especially from IoT devices [17, 18].

The study of botnets is a domain of analysis and investigation of botnet features to come up with new approaches to prevent, detect and react to botnet attacks [19]. In this respect, it is worth mentioning that the achievements obtained from research in this domain are significant as can be seen for example in studies such as [11] botnet detection using big data analytics framework or [20, 21] which explain machine learning approaches for identifying a botnet in a network with dynamic adaptation. Nevertheless, despite the considerable number of articles being published on this subject, as far as we know there is not yet a bibliometric article that reports on the research impacts and trends of such investigations.

Bibliometric methods carry out bibliographic overviews of scientific makings or selections of publications widely cited on patents, thesis, books, journal articles and conference papers to supply a quantitative analysis of publications written on a specific domain [22, 23]. This method has the potential to discover emergent and highest-impact subfields, as well as following the trend over extended periods of time within many areas.

This paper aims to present an exhaustive assessment of botnet detection research practices published from 2009 to 2018 indexed by Web of Science (WoS) with the purpose of demonstrating the growth of botnet detection research. To do this, we have analyzed research topics and publication patterns on botnet detection having in mind the following research questions: (a) What is the trending in publications about botnet detection study? (b) How does this trend help to identify the future course of botnet detection study?

This paper is organized as follows. Section 2 presents the research methodology. Section 3 exposes the findings and gives information on botnet detection studies. Section 4 discuss challenges and future trend of botnet detection study. Section 5 gives the conclusion to the research.

2 Methodology

Bibliometrics is the utilization of quantitative analysis and figures to publications, for instance, journal articles and their running with reference checks. Quantitative appraisal of publications and reference data is presently utilized as a part of all countries around the world [24]. Bibliometrics is used as a piece of research execution assessment, especially in school and government labs, and by policymakers, administrators, information experts and researchers themselves [9]. It includes the estimation of information not necessarily for the content but the amount of information on the particular area of research [11] and it can be divided into two parts by the researcher which include publication analysis and general instructions [25]. Publication analysis talks about the assessment of publication, for example, references, impact factor and publishers. This method has been used for many publications while for general instructions, the researcher briefs readers about how to seek great articles by utilizing search engines to maintain a strategic distance from possible errors in the search process [11].

For successful writing of this research paper, a few techniques were utilized to get the publications and it starts by utilizing “Botnet Detection Techniques” as the fundamental keyword. The keyword is critical because it offers the exact data for the research topic and does not drift from the topic [26]. We used both automatic and manual pursuit technique to break down the recovered publications. A scan for related publications ordered in the Web of Science database confines the search to the previous 10 years i.e. between 2009 and 2018. Thus, we distinguished a total of 341 publications from which are conference papers, book reviews, and articles.

To evacuate inconsequential distributions, 147 publications were removed consisting of non -English Publications, low impact journals, and low productivity articles. Because of this exclusion, 194 articles were secured for analysis and research purpose. The analysis was led considering the below criteria

1. Authors,
2. Highly cited articles,
3. Research areas,
4. High efficiency/Productivity,
5. Keyword recurrence,
6. High Impact journals
7. Institutions.

At last, to visualize the outcomes, we embraced the Microsoft Visio, Microsoft Word visuals feature, VOS Viewer, and web of science analysis feature as they are free and contain astounding highlights that help different sorts of visuals. Figure 1 below is provided for demonstration.

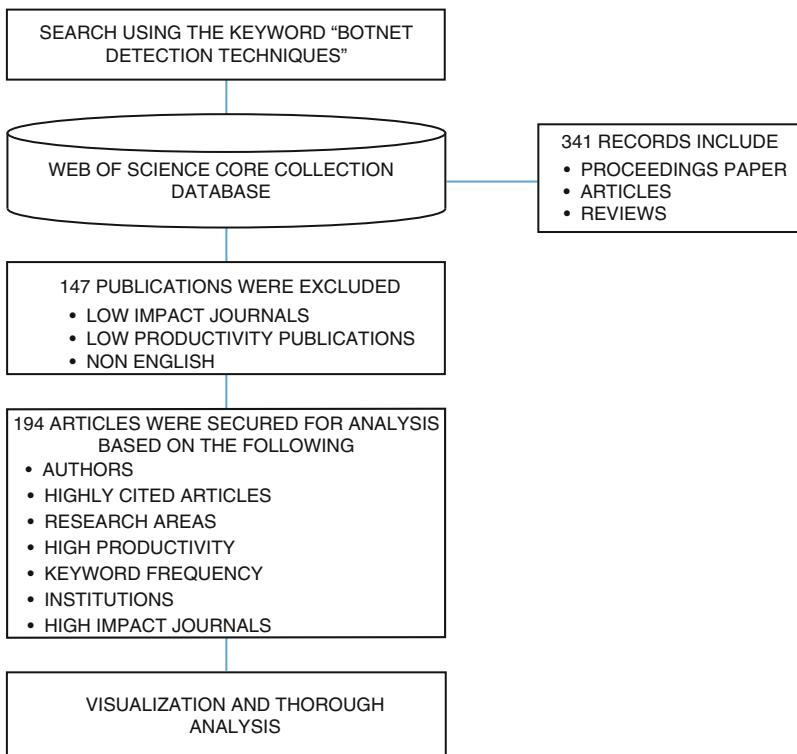


Fig. 1 Flowchart showing the analysis of the publication

There are numerous databases used to list articles and other publications, for example, IEEE Explore, Springer, Elsevier's Scopus, Google Scholar, Science Direct, Association for Computing Machinery (ACM). The three (3) principle Bibliometrics data source for searching literature and other texts are Google Scholar, WoS and Elsevier's Scopus. These information sources are normally used to class journals as far as their productivity and the total references got to show the journals influence or impact.

In this paper, Web of Science (WoS) was used as the primary search engine. The WoS database was the first bibliometric analysis tool and other search engines were indexed in WoS. The WoS Database is a product of "Thomson Reuters Institute of Scientific Information" and it contains more than 12,000 titles of diaries from multidisciplinary regions. The database offers effective searching and browsing options by enabling many options to help filter out results. Furthermore, the WoS database is additionally able to sort the publications based on parameters like date of publication, highest cited, newest publications, usage count, authors name, and relevance. In addition, refining the outcomes in the ISI Web of Science database likewise empowered certain outcomes to be excluded by type of documents, years,

continents, countries authors and institution. Added to that is its capacity to give vital data like quartile, impact factor and citation counts, this made the research easier and more conducive.

3 Findings

This section talks about the finding of the topics that are related to botnet detection. This section is classified into seven sub-points which are productivity, research areas, institutions, writers, impact factor, highly-cited articles and keyword frequency. These discoveries are vital because they demonstrate the publishing rates with bibliometric information. In Addition, it is able to disentangle leading edge research that supports the production of knowledge and to guarantee that the interest in botnet is more extensive than it sounds.

An analysis was performed by analysing the activities in seven (7) main continents which are Asia, North America, Europe, Australia, Africa and South America. Broadly speaking, this analysis showed that Asia leads the research area with the highest number of publications of 47.421% followed by North America with approximately 33%. The table below represents the analysis (Table 1).

Figure 2 and Table 2 illustrate the amount of three classes of publications, which are, conference proceedings paper, journal articles, and review papers. The proceedings, has the highest proportion of publications with a total of 70.09%. Articles have a total of 27.94% of publications while reviews have the lowest percentage of 1.96%.

Figure 2 indicates that since 2015, distribution of proceeding papers increased until 2017 then drastically reduced while articles and reviews are slightly increasing. This is likely because of a perceived reduction of botnet attacks. The publication peak year was 2015. This may have affected the publications to concentrate on more critical issues, thereby influencing the number of distributions in the next years. Following this, the increment of conference papers and articles can increase citations.

Table 1 Distribution of research based on continent

Continent	Publications (%)
Asia	47.421
North America	32.99
Europe	20.10
Australia	2.577
Africa	2.062
South America	1.031

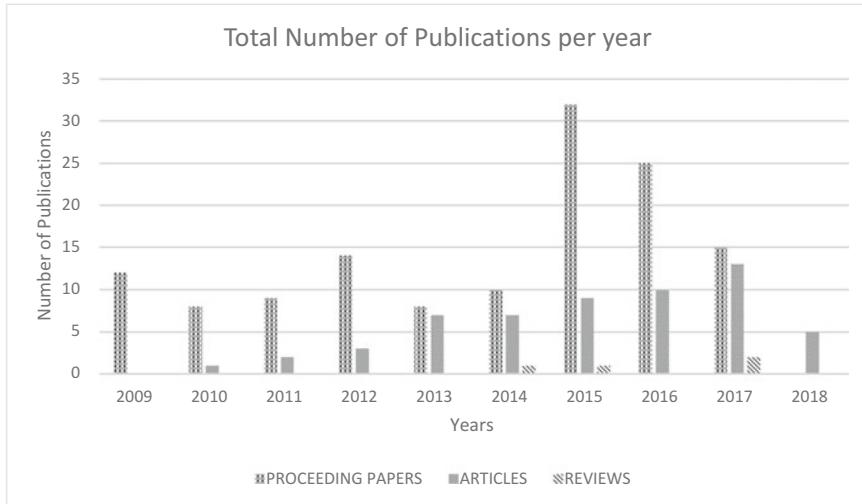


Fig. 2 Total number of publications per year

Table 2 Yearly number of publications

	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
Proceedings	12	8	9	14	8	10	32	25	15	0
Articles	0	1	2	3	7	7	9	10	13	5
Reviews	0	0	0	0	0	1	1	0	2	0

Citation analysis is utilized to evaluate the recurrence of the publications considering information extracted from the citation index. This is used to assess researchers' performance in view of citation patterns, particularly for academic purposes. It additionally gives information about researchers to other researchers utilizing similar references while likewise giving an all-encompassing perspective of the topic under research.

Figure 3 shows citation distribution of publications within the past 10 years. It demonstrates that the quantity of publications impacts the number of citations. The quantity of citations obtained by an article that has been published increases after the articles remains in the database for a long period of time. The earlier an article is published the more it is cited.

The normal number of citations gathered by distributed articles is around 78.30% yearly amid the time of 2009–2018. The quantity of yearly references demonstrates a positive multiplication with three peak periods in 2015, 2016 and 2017. There has been an increase since 2012. There was a great increase of 28.57% from 2014 to 2015. 16.67% from 2015 to 2016 and 22.72% from 2016 to 2017 and drastic decrease of 81.81% till date. The references have been expanding from 2015 to 2017 compared to how it was from 2009 to 2013. This is potentially because, in 2013, there was a rise in researchers directing studies to comprehend and solve botnet

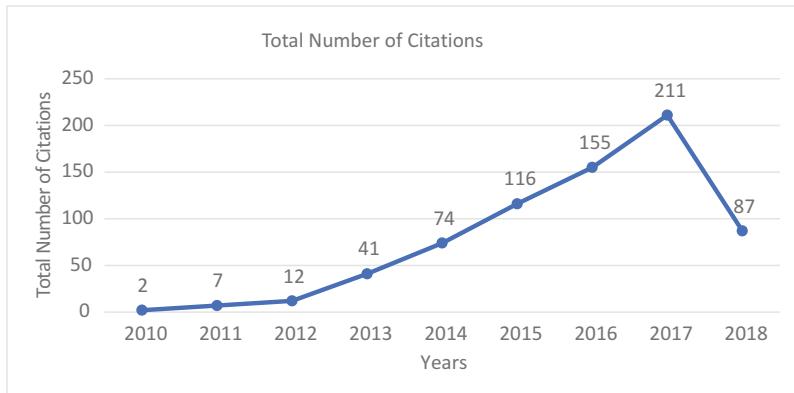


Fig. 3 Total number of citations

attacks. Researchers referring to other researchers works had additionally brought about an expansion in the citation. This pattern represents a positive outcome until 2017.

3.1 *Productivity of Continents*

This section talks about productivity of different continents. Efficiency in publications refers to the recurrence or number of productions collected. It is utilized as a tool to gauge the number of articles that were published among several continents. A research of productivity growth of articles enables the researcher to remain concentrated and focused on the production of articles as well as strengthen the research components. A vital part to focus on the analysis of productivity is the increments of the productivity proficiency of the publications. Additionally, it aids in evaluating countries and continents which have delivered most publications. Table 3 records the number of publications from 2009 up to date as indicated by continents. Asia has the highest number of published articles (92) and North America has the second higher number of publications with the United States being generally exceptional with a total of 64. This is followed by Europe and along these lines, Australia, Africa, and South America which relatively directed lesser research on botnets. Information demonstrates that the United States added to 24.7% of the whole distribution of articles in North America. Followed by Canada with 8.25%. Interestingly, in Asia, India filled in as the nation with the major contribution to research related to detecting botnets. This is followed by China and South Korea.

Table 3 Productivity of countries and continents

List of continents	Number of articles	Percentage (%)
Asia	92	47.421
India	29	14.948
China	19	9.794
Taiwan	8	4.124
Iran	7	3.608
Japan	5	2.577
Saudi Arabia	5	2.577
South Korea	10	5.155
Jordan	3	1.546
Pakistan	3	1.546
Vietnam	3	1.546
North America	64	32.99
Canada	16	8.247
USA	48	24.742
Europe	39	20.10
Ukraine	7	3.608
France	5	2.577
England	9	4.639
Italy	4	2.062
Spain	4	2.062
Czech Republic	3	1.546
Poland	3	1.546
Austria	2	1.031
Germany	2	1.031
Australia	5	2.577
Australia	5	2.577
Africa	4	2.062
South Africa	4	2.062
South America	2	1.031
Brazil	2	1.031

3.2 Research Areas

This section examines various productions including certain research areas. Research areas aim at building up a scientific understanding of research areas and how these challenge different regions in various segments of different industries. Research areas can be used to quantify the performance of a research depending on publication and reference rates. The performance of any research area demonstrates the pattern of the production in an interval. The Web of Science database contains a list of different disciplines. Some of these research areas include Engineering, Computer Science, Telecommunications, Automation Control Systems, Materials Science, Science and technology topics and others.

Table 4 Research areas with the highest publications

Research areas	Publication	Publication (%)
Computer Science	173	81.991
Engineering	97	45.97
Telecommunications	64	30.332
Automation Control Systems	9	4.265
Materials Science	4	1.896
Science Technology Other Topics	3	1.422
Business Economics	2	0.948
Energy Fuels	2	0.948
Physics	2	0.948
Chemistry	1	0.474
Environmental Sciences Ecology	1	0.474
Government Law	1	0.474
Information Science Library Science	1	0.474
Mathematics	1	0.474
Operations Research Management Science	1	0.474
Robotics	1	0.474

As shown in the Table 4 majority of articles go under Computer Science and Engineering. Computer Science and Engineering is an integration that used to build up new techniques and innovations that are valuable to academics and the general population at large.

Programming, Website design, algorithms, artificial intelligence, machine learning, Big data analysis and processing, Data mining, Computer and network security are a portion of the sub-regions that go under the term of Computer Science and Engineering. Of the articles distributed, the best article with the most astounding reference in Computer Science zone was “[Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers](#)” Since records on detecting botnets use computing and network security, penetration testing, programming, and machine learning.

Table 4 indicates distribution of papers in the computing field. It also demonstrates that most productions came generally from North America and Asia.

3.3 Institutions

This section means to distinguish which of the institutions are the most active and furthermore determine their size of research by comparing number of publications. Table 5 lists institutions with botnet related research in North America, Asia, and Europe. It demonstrates that institutions from North America have the most astounding number of distributions. This is trailed by Asia which has the second highest

Table 5 Ranking of institutions based on relevant publications

Institutions	Publications	Publications (%)	Country
Khmelnitsky National University	7	3.608	Ukraine
Korea University	6	3.093	South Korea
Dalhousie University	5	2.577	Canada
PSG College Technology	5	2.577	India
University of Malaya	5	2.577	Malaysia
University of New Brunswick	5	2.577	Canada
University of Texas System	5	2.577	United States
King Saud University	4	2.062	Saudi Arabia
Texas A&M University College Station	4	2.062	United States
Texas A&M University System	4	2.062	United States
Carnegie Mellon University	3	1.546	United States
Keene State College	3	1.546	United States
Los Alamos National Laboratory	3	1.546	United States
Southeast University China	3	1.546	China
Tuyhoa Industrial College	3	1.546	China
United States Department of Energy Doe	3	1.546	United States
University Sains Malaysia	3	1.546	Malaysia
Universiti Technology Malaysia	3	1.546	Malaysia
University of Electronic Science Technology of China	3	1.546	China
University Of North Carolina	3	1.546	United States
University of Pretoria	3	1.546	South Africa
The University of Texas At San Antonio USA	3	1.546	United States
University System Of New Hampshire	3	1.546	United States
UTP University of Science Technology	3	1.546	Poland
Al Balqa Applied University	2	1.031	Jordan

and then Europe which has the third highest number of publications. It appears that Khmelnitsky National University has the highest number of publications which are all conference papers.

The proceeding five (5) institutions are from Asia and North America: Korea University, Dalhousie University, PSG Coll Technology, University of Malaya and University of New Brunswick. These institutions are situated in China and the United States respectively. It appears that the speed of distribution in China is considerably higher than alternate nations in Asia.

3.4 Authors

This section is meant to recognize who is most active author in the field of botnet analysis. Table 6 lists the authors based on the number of contributions and

Table 6 Ranking of authors based on relevant publications

Author	Publications	Publication (%)	Country
Lysenko Sergii	7	3.608	Poland
Savenko Oleg	7	3.608	Poland
Kryshchuk Andrii	6	3.093	Poland
Anitha R	5	2.577	India
Lee Heejo	5	2.577	China
Pomorova Oksana	5	2.577	United States
Zincir-Heywood A Nur	5	2.577	France
Bobrovnikova Kira	4	2.062	United States
Haddadi Fariba	4	2.062	China
Ghorbani Ali A.	3	1.546	United States
Karim Ahmad	3	1.546	China
Kozik Rafal	3	1.546	England
Kumar Sanjiv	3	1.546	India
Lee Jehyun	3	1.546	China
Lu Wei	3	1.546	China
Yan Guanhua	3	1.546	China
Abadi Mahdi	2	1.031	Iraq
Almomani Ammar	2	1.031	Jordan
Alothman Basil	2	1051	England
Alzahrani Abdullah J	2	1.031	Saudi Arabia
Anuar Nor Badrul	2	1.031	Malaysia
Bin Muhyah Fahad T	2	1.031	South Korea
Butler Patrick	2	1.031	Spain
Chen Chia-Mei	2	1.031	Malaysia
Cheng Guang	2	1.031	China

their country of residence. Most authors are from China and the United States. Additionally, it is noticed that authors from Europe and Africa delivered fewer contributions. As seen in Table 6, Lysenko and Savenko from Poland are the topmost authors with most joint articles.

The three authors with the highest number of publications are from Asia and are said to have worked together for the most cited publication. The most cited of the top three authors is a conference paper “[A Technique for the Botnet Detection Based on DNS-Traffic Analysis](#)” which was cited six times. Lysenko and Savenko have worked together for all their seven publications. Kryshchuk Andrii is the author with the second highest botnet related publications. Anitha R has had five publications with “[Botnet detection via mining of traffic flow characteristics](#)” being the most cited by nine times. Figure 4 demonstrates the total and average number of citations and h-Index received by the top four authors.

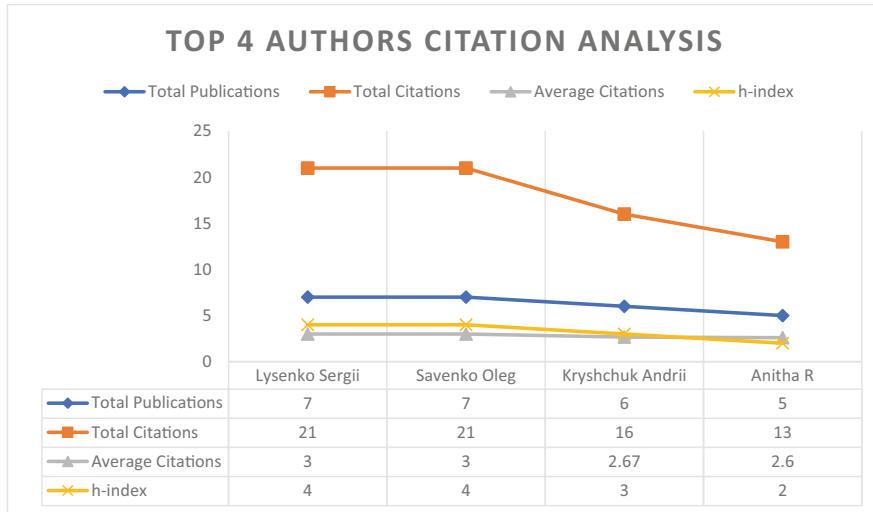


Fig. 4 Citation count for the top four most productive authors

3.5 Highly Cited Articles

This section ponders on the quality of research done and the impact it has on related fields. Table 7 shows 25 of the most referred articles in botnet research. These articles significantly contribute around 29.3% to the general publications. Moreover, the main four most cited articles were distributed between 2009 and 2014, demonstrating consistency with the idea that the longer the articles have been in the database, the higher the number of times it has been cited. The research areas which add to the productions are Telecommunications, Engineering, Automation Control Systems, Materials Science and different Topics, and Automation and Control Systems with Computer Science being the highest of the articles distributed. The most cited publication is “A Survey of Botnet and Botnet Detection”. The paper discusses botnets and ways of detecting botnet attacks extensively. The articles comprehensively discuss botnet and ways of detecting them. The article shows the extent of research. It inferred that exceedingly cited articles are not common articles but rather are quality research articles in which the researcher recognized other author’s discoveries, strategies, thoughts, and impact in specific fields.

3.6 Impact Factor

This section discuss about impact factor of journals that published most botnet related papers. This is essential since it determines the most leading articles in

Table 7 Top 25 highly cited articles

Title	Total citations	Published journal	Publication year	Research area
A Survey of Botnet and Botnet Detection	62	2009 Third International Conference on Emerging Security Information, Systems, and Technologies	2009	Computer Science
Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers	50	IEEE Transactions on Information Forensics and Security	2013	Computer Science
Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization	34	2013 IEEE Symposium on Security and Privacy (sp)	2013	Computer Science
Identifying botnets by capturing group activities in DNS traffic	34	Computer Networks	2012	Computer Science
Detecting P2P Botnets through Network Behaviour Analysis and Machine Learning	30	2011 Ninth Annual International Conference on Privacy, Security, and Trust	2011	Computer Science
A fuzzy pattern-based filtering algorithm for botnet detection	29	Computer Networks	2011	Computer Science
A Taxonomy of Botnet Behaviour, Detection, and Défense	17	IEEE Communications Surveys and Tutorials	2014	Computer Science
DNS for Massive-Scale Command and Control	17	IEEE Transactions on Dependable and Secure Computing	2013	Computer Science
Mobile Botnet Detection Using Network Forensics	15	Future Internet-FIS 2010	2010	Computer Science
DFBotKiller: Domain-flux botnet detection based on the history of group activities and failures in DNS traffic	13	Digital Investigation	2015	Computer Science
Botnet detection techniques: review, future trends, and issues	12	Journal of Zhejiang University-Science C-Computers & Electronics	2014	Computer Science
DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale Net Flow Analysis	12	28th Annual Computer Security Applications Conference (acsac 2012)	2012	Computer Science
RTECA: Real-time episode correlation algorithm for multi-step attack scenarios detection	11	Computers & Security	2015	Computer Science

(continued)

Table 7 (continued)

Title	Total citations	Published journal	Publication year	Research area
A botnet-based command and control approach relying on swarm intelligence	10	Journal of Network and Computer Applications	2014	Computer Science
Peer to Peer Botnet Detection Based on Flow Intervals	10	Information Security and Privacy Research	2012	Computer Science
Active Botnet Probing to Identify Obscure Command and Control Channels	10	25th Annual Computer Security Applications Conference	2009	Computer Science
PsyBoG: A scalable botnet detection method for large-scale DNS traffic	8	Computer Networks	2016	Computer Science
Botnet detection via mining of traffic flow characteristics	8	Computers & Electrical Engineering	2016	Engineering
Peri-Watchdog: Hunting for hidden botnets in the periphery of online social networks	8	Computer Networks	2013	Computer Science
On the detection and identification of botnets	8	Computers & Security	2010	Computer Science

publication and the highest citations acquired. From this data, researchers can fortify their work by distributing in great quality journals.

Table 8 shows that the highest number of publications belongs to articles under IEEE communications surveys and tutorials followed by IEEE transactions on information forensics and security and then IEEE transactions on cybernetics. IEEE communications surveys and tutorials got 8647 references throughout the years followed by IEEE transactions on information forensics and security with 5646 and IEEE transactions on cybernetics with 5553 references. However, twelve (12) journal titles were likewise documented in Table 8 to mention they are available in the database.

3.7 Keyword Frequency

This section presents an analysis of the kind of keywords frequently used by researchers by looking at trends in the highest occurrence of keywords in our dataset. This discussion is essential as it empowers articles to be associated and therefore detected in present and past topics of journal papers. These keywords can be utilized to investigate and identify research patterns and gaps.

Web of science produces full records by searching titles, abstract, keywords and author information. The data presented in this section was extracted from a

Table 8 Productivity of journals

Journal title	Impact factor	Quartile	Publications	Publications (%)
China Communications	0.903	Q4	735	1.22
Computer Communications	3.338	Q1	4865	8.08
Computer Journal	0.711	Q4	3214	5.34
Applied Sciences-Basel	1.679	Q3	591	0.98
Computer Networks	2.516	Q2	9028	15
Computers & Electrical Engineering	1.57	Q3	1756	2.92
Computers & Security	2.849	Q2	2.489	4.14
Concurrency and Computation-Practice & Experience	1.133	Q3	2057	3.42
Digital Investigation	1.774	Q3	948	1.58
IBM Journal of Research and Development	1.083	Q3	3350	5.57
IEEE-Inst Electrical Electronics Engineers Inc	3.244	Q1	1899	3.16
IEEE Communications Surveys and Tutorials	17.188	Q1	8654	14.38
IEEE Internet of Things Journal	7.596	Q1	938	1.56
IEEE Network	7.23	Q1	3014	5.01
IEEE Transactions on Cybernetics	7.384	Q1	5553	9.23
IEEE Transactions on Dependable and Secure Computing	2.926	Q1	1477	2.45
IEEE Transactions on Information Forensics and Security	4.332	Q1	5646	9.38
Information Systems and e-Business Management	1.723	Q3	342	0.56
International Journal of Distributed Sensor Networks	1.239	Q3	3055	5.01
International Journal of Information Security	1.915	Q2	564	0.94

sum of 965 keywords and 602 titles obtained from 57 publications for the period between 2009 until 2018. As mentioned before, we looked at trends in the highest occurring keywords in our dataset. Table 9 shows that the most applicable titles and keywords are “botnet detection” and “method”, as these are being used consistently by the authors for writing. This infers that most researchers had used these titles and keyword in their research. It is not surprising that “Botnet detection” is the highest occurring title because after searching, most results discuss botnet detection in several ways, some discuss the approaches, techniques and even tools. It is also totally understandable that the highest keyword is “method” since detecting botnets

Table 9 Highest occurring keywords and titles

Keyword	Frequency	Title	Frequency
Method	99	Botnet Detection	211
Traffic	62	Network Traffic	153
Activity	61	Detection System	117
Host	61	Android Botnet	110
Model	55	Evasion Technique	97
Service	43	P2p botnet	91
Server	40	Application	80
Botmaster	29	User	73
DDOS Attack	28	Implementation	65
Device	16	Dataset	41

is a process and so this word is used for any article discussing botnet detection. It is imperatively important to discuss the methods of detecting them. Different researchers have used different methods of botnet detection. However, one of the most common methods that the publications discuss is capturing network traffic and monitoring its behavior. This justifies “traffic” and “Network traffic” as the second most used keyword and title. Keywords with “traffic” include “DNS Traffic”, “Botnet traffic” and “Network traffic”.

Information provided in Fig. 4 shows the word map derived from analyzing the content of the articles. This outlines them into five groups, which are highlighted in five colors including red, blue, yellow, green and purple. As shown in Fig. 4, most of the researchers conducted their research on certain topics like Botnet detection methods, Botnet behavior, DNS Traffic analysis, Botnet detection tools and Botnet communication protocols (Fig. 5).

4 Research Trends

In this section, we discuss our findings, research trends and the overall outcome of this research. The section focuses mainly on the production of research publications analyzed along the period of study of this paper, from 2009 to June 2018.

Figure 6 illustrates the fluctuation experienced by botnet detection research publications from 2009 to date. It is evident that in 2009 there was a considerable amount of publications before it slightly dropped in 2010 and increase again a little in 2011, not as much as it was in 2009. In general, botnet detection research has been unpredictable and kept fluctuating until 2015 when it increased to its peak time. However, since 2015, the number of research papers have been decreasing.

Figure 7 shows a more detailed analysis by considering yearly publications for each continent. There has been an increase in publications from the year 2011 to 2015 with Asia having an exceptionally great number of articles but a sudden decrease from 2015 to 2016 which points a reduction in attacks compared to its peak time in 2015. Annual increase in publications urges researchers to expand their

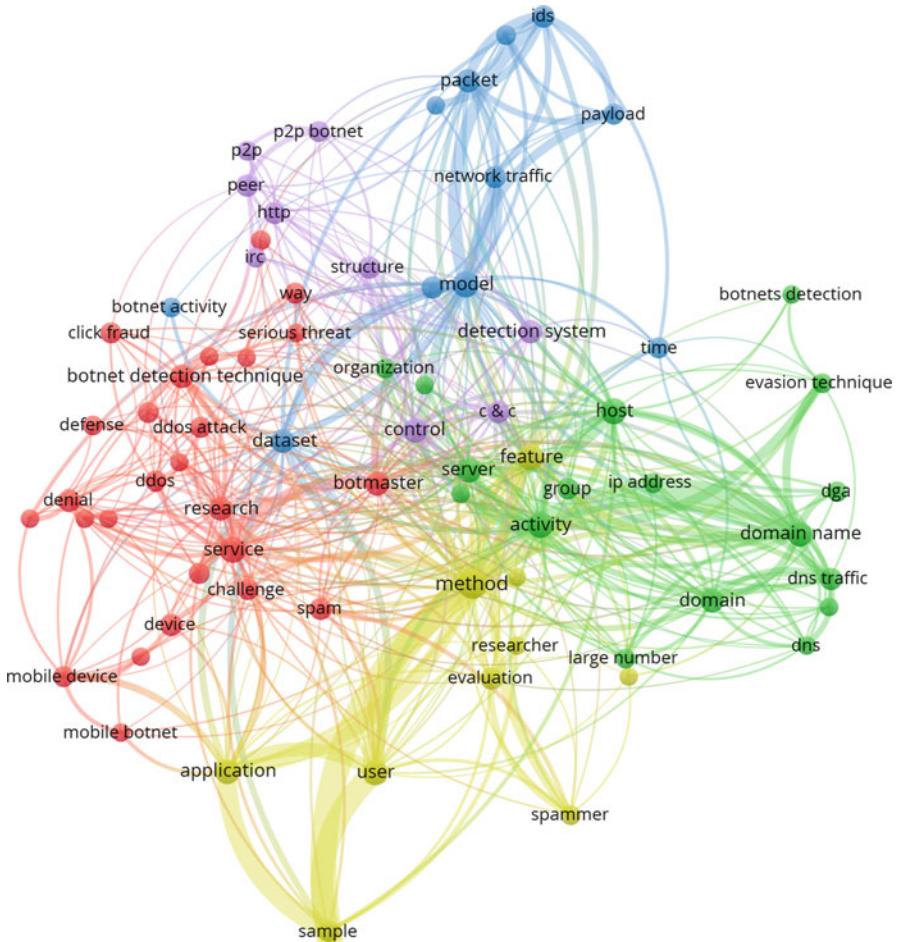


Fig. 5 Keywords cluster diagram

research. The decrease, on the other hand, may indicate success or efficiency in the newly researched detection techniques. However, the relaxation in the research may have led to more attacks because it indicates another rise in the year 2017 before the unusual decrease in 2018. This, in general, portrays positive energy in Asia for botnet related research.

Continents like Europe, North America, Australia, and Africa have performed not so well. Europe has been very unpredictable, there was a decrease from 2009 to 2010 and kept increasing slowly until 2012 when sharply decreased until 2014 before a massive growth in 2015 that remained until 2017, however it has been falling till date. The figure above summarizes the findings.

It is evident from the results above that 2015 was the peak year for botnet attacks and the year which botnet research had trended the most. Continents like North

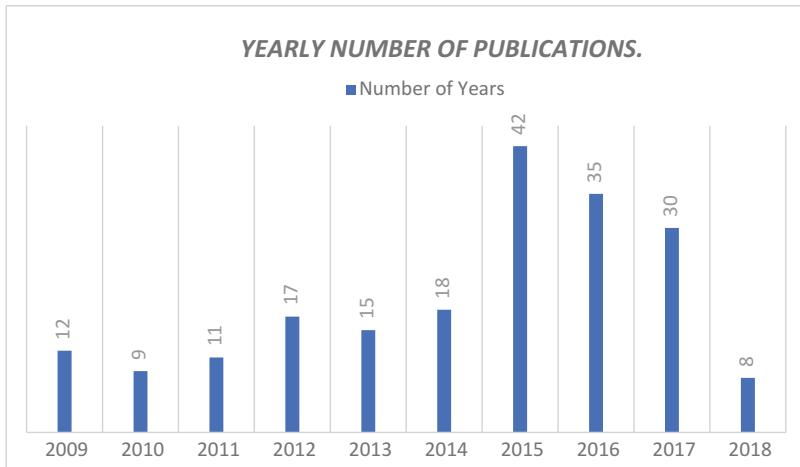


Fig. 6 Yearly number of publications

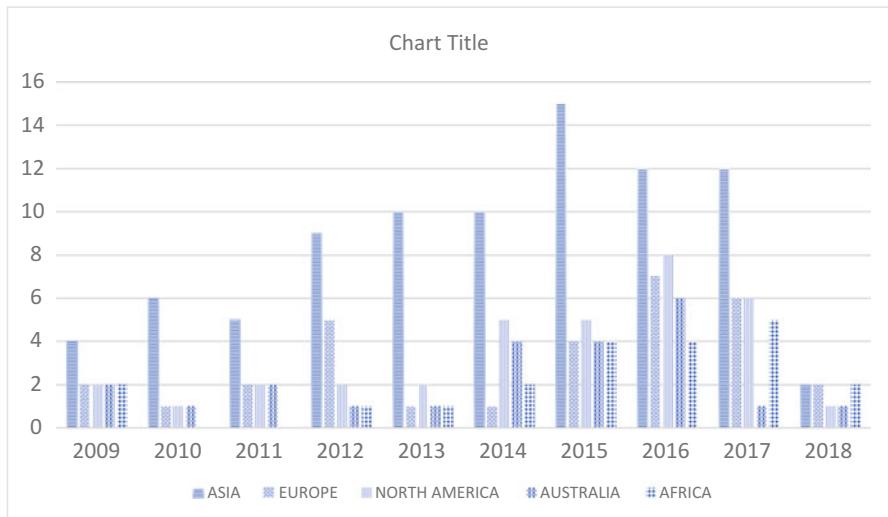


Fig. 7 Yearly number of publications based on continents

America, Australia, and Africa did not have a relevant amount of publication until 2014. This also indicates that there was a trigger which led to a sudden interest in botnet detection research.

Following the analysis, Fig. 8 presents the trends in publications from the research areas point of view. From the previous section is known that most of the articles about botnet detection are published under the areas of Computer Science and Engineering.

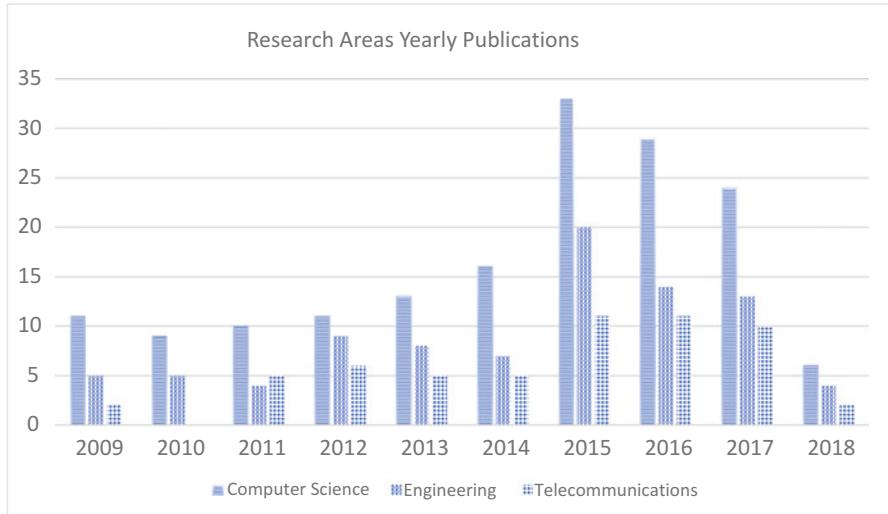


Fig. 8 Yearly number of publications based on research area

Research publications in the area of Computer Science has experienced a continuous growth since 2010. Between 2010 and 2014, the number of publications grew slowly in the aforementioned areas from approximately 14 to 28 articles published. Nonetheless, in 2015 reached a remarkable peak of 64 publications, more than twice the amount of the previous year, showing consistency with the preceding graphs. After that, the growth rate showed a steady decline in bibliography production during the next 2 years, and a sharp drop in growth rate in 2018, although the amount for this last year is not definitive yet. This decreasing behavior of the last 3 years is a general trend in the rest of research areas, Engineering and Telecommunications, although for these two on the previous years to 2015 their amount of publications shows a more fluctuating behavior, which makes difficult to predict future trends.

In conclusion, the author believes botnet research is vital because the trend of publication clearly shows a need for ongoing research to tackle botnet detection techniques. However, the projection in number of publications in a near future is hard to estimate since it is related to attack events that can take place in the future worldwide cybernetic landscape, without mentioning possible advances in technology that are not being contemplated in this work and which may affect the way botnets operate.

5 Conclusion and Future Works

In this work, we used bibliometric analysis to examine botnet detection techniques during the period from 2009 to 2018, which allowed us to expose global tendencies related to bibliography production of botnet detection techniques. In this

investigation, we offered seven (7) analysis criteria including productivity, research areas, institutions, authors, impact journals, highly-cited articles and keyword frequency. These criteria led us to note that Asia has the highest number of published articles, followed by North America where the United States stands out with an exceptional number of published articles.

The research area in which most of the articles about botnet detection are published is Computer Science and Engineering. Therefore, the research area with the highest cited publications is Computer Science. In this regard, it was also demonstrated that most productions come generally from North America and Asia.

On the other hand, the institutions that have directed most of the research related to botnets are mainly located, in descendent order of publications, in North America, Asia, and Europe. In the first two continents, most of the authors are from the United States and China respectively. Other countries in Asia like India and Malaysia, along with China, are very active in publishing articles as well.

Regarding the impact factor, the highest number of publications belongs to articles of IEEE Communications Surveys and Tutorials journal followed by IEEE Transactions on Information Forensics and Security and finally IEEE Transactions on Cybernetics. It was pointed out that the publication venue is critical in determining whether a newly published paper will be highly cited or not. It is worth to mention that the year of publication matters regarding the impact of the article, the earlier an article is published the more it is cited.

Lastly, during the analysis of keyword frequency, it was determined that the most applicable titles and keywords for the subject of this paper are respectively “botnet detection” and “method”. These are being used consistently by authors for writing and they have a direct influence on describing the trends and research directions for future study in botnet detection related research.

References

1. S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha, and R. Khayami, “BoTShark: A Deep Learning Approach for Botnet Traffic Detection,” 2018, pp. 137–153.
2. M. Hopkins and A. Dehghantanha, “Exploit Kits: The production line of the Cybercrime economy?,” in *2015 2nd International Conference on Information Security and Cyber Forensics, InfoSec 2015*, 2016.
3. M. Damshenas, A. Dehghantanha, K.-K. R. Choo, and R. Mahmud, “M0Droid: An Android Behavioral-Based Malware Detection Model,” *J. Inf. Priv. Secur.*, Sep. 2015.
4. A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, “Detecting crypto-ransomware in {IoT} networks based on energy consumption footprint,” *J. Ambient Intell. Humaniz. Comput.*, Aug. 2017.
5. A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, *Machine learning aided static malware analysis: A survey and tutorial*, vol. 70. 2018.
6. O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, “Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing,” *Eurasip J. Wirel. Commun. Netw.*, vol. 2016, no. 1, 2016.

7. J. Baldwin and A. Dehghantanha, *Leveraging support vector machine for opcode density based detection of crypto-ransomware*, vol. 70. 2018.
8. D. Kiwia, A. Dehghantanha, K. K. R. Choo, and J. Slaughter, “A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence,” *J. Comput. Sci.*, 2017.
9. D. Zhao *et al.*, “Botnet detection based on traffic behavior analysis and flow intervals,” *Comput. Secur.*, vol. 39, no. PARTA, pp. 2–16, 2013.
10. H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, “A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting,” *Futur. Gener. Comput. Syst.*, 2018.
11. K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, “Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests,” *Inf. Sci. (Ny.)*, vol. 278, pp. 488–497, 2014.
12. O. M. K. Alhwai, J. Baldwin, and A. Dehghantanha, *Leveraging machine learning techniques for windows ransomware network traffic detection*, vol. 70. 2018.
13. H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, “Intelligent OS X malware threat detection with code inspection,” *J. Comput. Virol. Hacking Tech.*, 2017.
14. A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, “Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning,” *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2018.
15. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, “Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence,” *IEEE Trans. Emerg. Top. Comput.*, pp. 1–1, 2017.
16. H. Haughey, G. Epiphanou, H. Al-Khateeb, and A. Dehghantanha, “Adaptive Traffic Fingerprinting for Darknet Threat Intelligence,” 2018, pp. 193–217.
17. M. Conti, A. Dehghantanha, K. Franke, and S. Watson, “Internet of Things security and forensics: Challenges and opportunities,” *Futur. Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan. 2018.
18. H. Haddad Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo, “A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks,” *IEEE Trans. Emerg. Top. Comput.*, pp. 1–1, 2016.
19. N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, “Machine learning aided Android malware classification,” *Comput. Electr. Eng.*, vol. 61, 2017.
20. S. Ranjan, J. Robinson, and F. Chen, “Machine Learning Based Botnet Detection Using Real-Time Connectivity Graph Based Traffic Features,” 2015.
21. S. Ranjan and F. Chen, “Machine Learning Based Botnet Detection With Dynamic Adaptation,” 2006.
22. J. Baldwin, O. M. K. Alhwai, S. Shaughnessy, A. Akinbi, and A. Dehghantanha, *Emerging from the cloud: A bibliometric analysis of cloud forensics studies*, vol. 70. 2018.
23. J. Gill, I. Okere, H. HaddadPajouh, and A. Dehghantanha, *Mobile forensics: A bibliometric analysis*, vol. 70. 2018.
24. I. Ghafir, V. Prenosil, and M. Hammoudeh, “Botnet Command and Control Traffic Detection Challenges : A Correlation-based Solution,” no. April, pp. 1–5, 2017.
25. G. Kirubavathi and R. Anitha, “Botnet detection via mining of traffic flow characteristics,” *Comput. Electr. Eng.*, vol. 50, pp. 91–101, 2016.
26. J. A. Jerkins, “Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code,” *2017 IEEE 7th Annu. Comput. Commun. Work. Conf. CCWC 2017*, 2017.

Security in Online Games: Current Implementations and Challenges



**Reza M. Parizi, Ali Dehghantanha, Kim-Kwang Raymond Choo,
Mohammad Hammoudeh, and Gregory Epiphanou**

Abstract Security in online gaming is a growing target for hackers due to the amount of money involved in online gaming. Components are used in the security measures implemented for these games, but no single security component is 100% effective. Our research aims to investigate methods of game security and the components used in them, as well as the hacks used to exploit and circumvent online gaming security mechanisms. As a result, our study arrives to some interesting points, and outlines a number of recommendations and potential research directions. This, in turn, can pave the way for facilitating and empowering future research in this domain to assist game engineers and testers in security management.

Keywords Online game · Security · Game data · Data privacy · Hacking

R. M. Parizi

Department of Software Engineering and Game Development, Kennesaw State University, Marietta, GA, USA

e-mail: rparizi1@kennesaw.edu

A. Dehghantanha (✉)

Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON, Canada
e-mail: ali@cybersciencelab.org

K.-K. R. Choo

Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX, USA

e-mail: raymond.choo@utsa.edu

M. Hammoudeh

School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester, UK

e-mail: M.Hammoudeh@mmu.ac.uk

G. Epiphanou

Wolverhampton Cyber Research Institute (WCRI), School of Mathematics and Computer Science, University of Wolverhampton, Wolverhampton, UK

e-mail: G.Epiphanou@wlv.ac.uk

1 Introduction

This paper discusses current security implementations in online gaming, how various security measures are circumvented in these games, and what countermeasures developers can take to counteract these lapses in security. With 150 million American consumers and accruing nearly 32 billion USD in revenue in 2016 alone [1], video games make up one of the largest sectors of the entertainment industry. As such, it is also home to the exchange of sensitive personal data of millions of users. This makes the video gaming industry a prime target for hackers who wish to exploit this sensitive data with cheats, hacks and various attacks, often for monetary gain. Often times in online multiplayer games many user accounts and the in-game items associated with them, have a real monetary value that other players would be willing to pay for. With this valuable data at risk, developers must include a multitude of components in their games, some specifically with security in mind. No single security measure or component is fully effective, so it is necessary to implement multiple types of security components to protect users and user data [2]. With the various ways that hackers could attempt to infiltrate a gaming service, developers should identify these methods of infiltration in order to properly counteract them.

When it comes to circumventing how a game is intended to be played, there are many ways to do so. Cheats, hacks, and outright attacks on the game that is being played can all lead to an unfair advantage for the person engaging in this activity, with consequences beyond just winning that particular encounter. When a person cheats, hacks, or attacks the game, they not only compromise the integrity of the game, but also may be engaging in illegal activity, stealing resources, money, or even identities from other users. Though very similar to one another, cheats, hacks, and attacks are not all the same all the time. Each has a different method of circumventing the game and its security, whether by modifying the game, skipping steps designed to keep intruders out, and other methods intended to keep players inside the expected play area.

Attacks in online games are defined as malicious attempts to harm other players, gain economic rewards, steal information, or otherwise harm the play experience for others [3]. This is a very direct way to gain from other players, especially by stealing information and gaining economically from stolen accounts, passwords, and even credit cards or bank accounts. However, there are more than one way to achieve this goal of attacking an online game and its players. There are many ways to gain access to an account owned by another player and the information held within. Probably the most prominent way to gain access to another account is through social engineering. Through clever use of words and fake messages, an attacker can make the user hand over the credentials to their own account, including usernames, emails, passwords, and even bank information [4]. Using this information, the attacker can lock the original user out of the account, and then can sell the account, the account information, or steal from the user. Similar results can be produced if a game or its service does not have sufficient security surrounding its login information. Unencrypted messages sent from the user's game client and the game server can be

intercepted and the information within stolen. This can lead to the account being compromised with no wrongdoing by the user, giving a false sense of security and allowing the attack to occur unnoticed.

Additionally, an attack could occur when a fake game server or client is set up and distributed to gain the login credentials of unsuspecting users. This is especially dangerous on games without any authentication beyond username and password, as it relies solely on the integrity and confidentiality of the login system. Another attack on logins without authentication are dictionary and brute force attacks, which don't steal login information but crack it by way of guessing. Passwords are often easy to remember, which means that a program may be able to generate it using a set of parameters such as the requirements for the password being cracked. Over the course of millions of permutations of letters and numbers, the password is guessed correctly and the attacker is granted unlimited access to the account. Malicious software such as Trojan horses and keyloggers can also give an attacker access to another user's account. The virus gets installed on the target machine and then secretly records and sends the information being entered to the attacker, compromising the account. These are some of the main ways an account can be compromised or outright stolen, but are not the only types of attacks on online games.

Distributed Denial of Service (DDoS) is an attack on the game itself, preventing anyone but the attacker from accessing the game and the information the game is using [4]. This can include login information like is mentioned above, or it can contain resources in game that are sought after and traded for using real world currency. The attacker can gain unrestricted access to these resources and gain stature and money as a result of that unrestricted access.

Cheating is another form of gaining an advantage over other players to gain resources in game. Cheating can be defined as any advantage a player gains over another that is not according to the game rules, according to the entity that is operating the game [4]. This is different in online games than in single player games in that in a single player game, other players are not being cheated out of a proper gaming experience. Often, a game developer will put "cheats" in a single player game that modify the gaming experience for players that have completed the game or want to play it differently. These are sometimes in the form of codes that can be entered, or sometimes switches in a menu, but they often disable recording statistics that get published to leaderboards so that the experience for other players is not warped.

Cheating in online games takes on many forms, and have varying levels of technical execution. One way that players can cheat is collusion in game [4]. Groups of players can abuse in game mechanics to fix outcomes in their favor, agreeing to certain behavior that ultimately benefits them. Boosting is a form of collusion that benefits players by artificially inflating player levels to reap in game rewards. Players can sell accounts that have been boosted or pay to have their accounts boosted so they don't have to work to reach that level. Players agree to alternate wins in matches, allowing each player to gain experience. In some games, there is a penalty

for losing matches, and in that case, boosters will use multiple accounts to boost one, working in a pattern that benefits the account being worked on.

Vote kicking is another form of collusion in online gaming that can be abused to prevent playing with players that are deemed undesirable, such as players who won't give up rewards to the majority group [4]. Vote kicking is intended to prevent players from playing with players in a group environment who are detrimental to the group, uncooperative, and otherwise toxic to the playing experience. Cheaters can abuse this by gaining a majority and making demands of players joining the group to play. If the demands are not met, however arbitrary and unrelated to the game the demands may be, the new player will be kicked and a brand new player will be introduced to the group.

Other ways of cheating without needing technical knowledge or modification is abusing in game mechanics and bugs in the code. As mentioned before, losing an online match can have negative consequences such as lowering player level or a lower level of rewards for the player. Sometimes an undesirable outcome can be avoided by simply turning off the device and preventing the match from being completed if it is clear that the player is going to lose [4]. This cheats the system by operating outside the way that is intended, and cheats the winning player by denying them the reward that they earned by winning the match.

Bugs and loopholes are another way of cheating without modification if the player is abusing them to get around completing the game the right way. If a bug is present that allows access to an area before it is unlocked, it allows players to bypass chunks of the game to complete it faster or obtain items that cannot be obtained easily [3, 4]. These items can be sold to other players through third party marketplaces for real world money, instead of being bought from the game provider.

Modifying the game experience through hardware or software is another way that players can cheat. Often online games are too large to be completely hosted on game servers, so they rely on the game client to handle gameplay and state of the player [4]. This trust by the server with the client can be abused to make the player seem different than they are, tricking other players into encounters that they are not prepared for. This can also be used to modify the game boundaries to allow the player to skip areas, access forbidden areas and items, and allow more immediate rewards than playing the game without the cheats [3]. The hardware on the client side can also be modified to benefit the player as well. Graphics cards can be altered to change the textures of certain in game elements, such as making the wall or boundary of a level appear to be clear. Also known as a "wall hack", this can create an unfair advantage in games where players are looking for something, or in online shooters where players can escape an encounter by running around obstacles and hiding [3, 4].

Another way to cheat is through the use of Artificial Intelligence (AI) to aid the user. These AI are used to gain an advantage by helping the player to accomplish tasks faster and more efficiently than they could by themselves [4]. Also called "bots", they encompass a wide range of assists, ranging from helping players playing shooters to never miss an opponent to automating the game character completely to increase the player level without doing any work. These bots allow

players to obtain an unfair advantage over honest players, often to the detriment of the players that are just playing the game [3].

Earlier it was discussed that Distributed Denial of Service (DDoS) was a way for an attacker to gain unrestricted access to a game and its resources. This concept exists on a smaller scale as Denial of Service (DoS) that exists in peer to peer games. A cheater can target an individual for any reason and cause their game client to become unusable to the point that the other player cannot play the game any more. This creates an environment with fewer competitors and more to gain for the cheater, while disallowing another player to enjoy the game and gain the rewards themselves [4].

DoS gets close to the concept of hacking in online games, which can be defined as remotely breaking into another player's computer and compromising their account or game client to gain information [4]. This is also very similar to attacks, but the methods are more specific. Server hacks can be used to provide an uneven playing field for players that do not know that they are playing in an altered environment. Hackers can reconfigure the server of a game to redistribute rewards, steal from players, or make themselves invincible in game. Private servers are also hacks, as they reuse the code from the server to run their own playing environment. This can lead to a warped playing experience in favor of the owner of the server, and the connections to the private server may be hijacked and the information stolen [4].

Client hacks are another way to hack another player. Attaching false logins to a game client distribution can redirect the information to a malicious server, which will store the information to be used illegally at a later time. Malicious software can also be distributed along with the fake game client to gain access to the player's computer and the files on it. Trojan Horses and keyloggers can glean passwords, but some viruses can be installed to make the game completely unplayable for the infected user. This will lock the player out of the game, allowing the hacker to step in and take the account [4].

2 Comparing Hacks in Online Games

The aforementioned methods of hacking or exploiting online gaming share a multitude of similarities as well as differences. If a pattern of offense can be recognized then we may be able to proceed with a strategy to prevent future attacks of similar nature.

Social engineering attacks utilize cunning and deceptive tactics in order to benefit oneself. A social engineer, more commonly known as a scammer, will commonly pose as an authoritative identity and request for account information. The person may also pose as somebody that is trying to help the victim and say that they can install certain "cheats" or get them virtual treasures. The victim will be lied to and persuaded into giving account information and sometimes personal information as well as financial information [4]. Similarly, a Trojan horse or a key logger is used to deceive a victim. Trojan horses and key loggers are planted on the victim's

computer typically by a malicious email or a false link. The person who accidentally downloaded the virus is then, unknowingly, sending data back to the hacker [4]. Both of these methods are used to gain private information about accounts. Smaller cases would be losing an account or virtual treasure. In some worse cases, a hacker can steal information about a credit card or other personal information. Social engineering only requires a person to be deceptive and cunning with their words while planting a key logger requires knowledge of how to operate the software.

Use of Artificial Intelligence, commonly referred to as “bots” by the online community, is popular amongst online gamers [3]. Programs can perform mouse and key tasks that are required by the player. The bots can perform a variety of tasks from helping a gamer win in a first person shooter to enabling a player to gather a vast amount of resources in certain role playing games. Another way of gathering experience or treasures illegally is from in-game collusion, or “boosting” [4]. Through boosting players can rapidly earn experience, in-game rewards, and complete challenges. Boosting requires the use of other players or even multiple accounts whereas botting only requires the software necessary to run the bot. Boosters can be easily identified in some games such as online first person shooters. The pair of boosters will have one player performing extremely well while the other player is performing poorly. Identifying a person using an Artificial Intelligence program is a trickier obstacle to overcome. Typically somebody that is running an AI will be performing the same task for an unusually long period of time [3]. Online games usually have a feature that allows a player to report another player if they feel that they are abusing any of these instances. During online play, a player may see or experience another player abusing certain game mechanics or loopholes [4]. Some forms of exploitation may be more obvious than others. One of the most obvious examples would be exploiting a bug or a loophole. A player may find that an action or a certain set of actions will result in an unexpected result. This result may come in the form of extra currency, treasure or possibly a competitive edge. Another hack that would give a user a competitive edge would be a wall hack [3]. Both bug exploitations and wall hacks abuse the fact that the game has a certain way to bypass expected play parameters.

3 Comparison Between Countermeasures for Cheats, Hacks, and Attacks

With all the methods of cheating, attacking, and hacking, online game developers have developed ways to counteract these ways of circumventing the security of the game. This practice can take on active and passive roles at different stages of the process. One of the most prevalent ways of countering these attackers and hackers is by using two factor authentication. Used by many gaming companies and services such as Valve, which uses Steam Guard, and Blizzard, who uses Blizzard Authentication, it creates a barrier to people who seek to steal accounts. Two factor authentication works by generating a code on a trusted device that must be entered

within a certain time frame when logging in. The user enters their credentials, then is prompted for this code. If the code is not entered in time a new one must be entered. If the code cannot be correctly entered, access to the account will not be granted. This prevents users other than the owner of the account from accessing the account by adding an extra layer of security.

Another form of security is by monitoring the account itself and its behaviors. If an account is attempting a login from an unknown device or location, the authenticator service such as the one used by Blizzard, called the [Battle.net](#) Dial-In Authenticator [4], will request additional information from the user before they can access the account. This can come in the form of a phone call, text message, or email asking if this is a legitimate login attempt. If it is, then the user can indicate it and then continue playing.

Client monitoring is another way of securing a game, but instead of protecting logins it detects players who are cheating in game. One way of detecting cheats is by having the client itself look for programs or processes running on the client machine that modify the game in any way. World of Warcraft's Warden is such a device, and it detects bots, hacks, and other cheats running on the host computer [3, 4]. Other things that can be monitored are player location in game, what items a player has, and their movements in game. If a player is in an area that is off limits to all players, or is off limits to players of that level, the monitor can flag the account and remove that player. This prevents cheaters from accessing parts of the game that other players cannot access, and preserving the environment created by the developers.

Movement is also a metric that can be tracked to identify cheaters [5, 6]. If a player is using a “wall hack”, their movement will vary from that of an honest player. Paths taken to objectives, items, or other players will differ when the player can see through or even teleport through walls, as will the sequence of keys being pressed. Key detection can prevent automated bots from taking over servers, as bots do not require key presses after being started [5]. If no keys are pressed within a certain time period but the player is still moving and playing the game, it indicates that the player is a bot and is removed from the game and flagged for cheating [5]. Machine learning can be utilized to automate the process of detecting these cheats and more efficiently remove those who are using them [3]. On large scale games this can lead to a better playing experience for those who are playing the game honestly.

Valve has created an environment for honest players in their online multiplayer games called Valve Anti Cheat (VAC) servers, which do not allow players who have been flagged for cheating to play on that server [7]. How Valve detects the cheaters is not known outside of the company, but the players who cheat are not banned immediately after being detected. The cheats happen in waves, presumably to further confirm that a player is cheating before banning them. VAC servers and bans are on a game by game basis and being banned in one game will not prevent the player from playing on VAC secured servers in another game [7].

Patches are a more reactionary method of securing games, but is an effective way to fix security issues and prevent cheaters from playing. Patches can be applied to a game server if it is a game wide issue or a game client if it is an individually based

problem to fix the issues being addressed [4]. These can fix small bugs, loopholes in game procedures, or outright security holes in the game to prevent these flaws from being abused by cheaters and hackers. However, patches can also introduce new problems that may need to be addressed, as the fix for one problem may cause problems to arise in other areas of the game [4]. Cheaters and hackers can find these new problems, so patches must be tested for security before being released to the public.

Many of these countermeasures have similarities that allow them to be used together or independently of one another. Passive methods such as two factor authentication and account monitoring are easy ways to detect and prevent accounts from being compromised while still allowing the player to access their account when doubt is cast on the login attempt. Patches and servers such as the VAC Servers are more active ways of preventing cheaters from playing due to developer involvement in the process of security. Every method has one thing in common, however. No method is 100% accurate, and every method mentioned above must balance the risk of falsely accusing an honest user who is just very good at the game of cheating. Honest players being banned would not be an acceptable outcome, so the developers must be careful. Additionally, every security measure in online games requires extra effort by either the user or the developer, and too many layers can cause access time to slow down.

Two factor and dial-in authentication can cause access time for the user to slow, and if it does not work quickly can lead to frustration for the end user, especially if it is slower than other forms of security such as account monitoring and patches. Patches, while faster for the end user once implemented, can be slow to distribute and incomplete if rushed. When done properly it is a more permanent solution to security problems than account monitoring or secured servers, which can be slower even still to prevent cheaters from playing. Monitoring and secure servers take time to gather information before determining that a cheater is playing and removing them, leading to a longer exposure for honest players.

4 Prominent Gaming Hacks

As there are so many potential security concerns that can be exploited with video games, it comes as no surprise that there have been several large scale hacks in recent years in the gaming industry. It will go to show that no matter a company's size or name, all involved can be at risk of a potential attack. Attacks in this sense are not always for an attacker's personal gain, some in fact are performed just for the 'fun' of it or just to point out security shortcomings. The industry has received such attention from malicious users in recent years because of video games rise in popularity throughout the world. Due to this increased attention, it is imperative for individuals to prepare themselves for potential attacks by educating themselves on previous security breaches.

One of the largest of all time, the 2011 Sony Playstation Network hack, was an attack that affected over 77 million people. This security breach initially occurred between April 16th and April 19th of 2011 and exposed a substantial amount of user's personal information. The information exposed in the breach included names, birthdates, addresses with city/state/zip, email addresses, gender, phone numbers, usernames, and hashed passwords [8]. The initial breach occurred over the specified days, but did not affect the performance of the network. After Sony had discovered the breach on April 19th, the internal security team took down the Playstation Network in order to discover the extent of the security breach and create new security measures. This takedown lasted for 23 days. Some individuals involved in the security industry participating in the Sony network had discovered several months before the hack that network game servers were run on Apache web servers. This is not a problem except for that the game servers were running on very old versions of the Apache software, and were unpatched with no firewalls against potential security threats [8]. These findings had been reported to Sony, but either the company did not take the warnings seriously or there was not enough time to have implemented the updates. The actual method used to breach the network has not been officially released, but it is commonly believed the hackers were able to get access to Sony's internal company network by modifying a Sony Playstation 3 unit into believing it was a developer console. This modification allegedly allowed these units to access the internal Sony network unrestricted. While the network was down, Sony engaged multiple expert information security firms and performed an extensive audit of the system. Since then Sony has implemented a slew of new security features that are intended to provide greater protection of user's personal information [9]. Sony went even as far as to add more automated software monitoring, increased levels of data protection and encryption, increased the ability to detect software intrusions within the network, and the implementation of additional firewalls. In order to try and win back some of the service's previous users, Sony implemented a "Welcome Back" appreciation program. To show their commitment to the protection of user's personal data, the company provided complimentary offerings to assist users in enrolling in identity theft protection [9]. Sony also provided users with premium content such as free games and free 30 days Playstation plus membership.

Another recent hack, albeit not as harmful, was an attack that affected almost 24,000 Japanese members of Nintendo's Club Nintendo service. Over the course of several days, June 9 to July 4 of 2013, Nintendo was able to identify 15.46 million false login attempts, only 23,926 of which were actually successful. This hack was successful in gaining access to user's name, home address, phone numbers, and email addresses associated with the corresponding club account [10]. In response to the hack, Nintendo said that it suspended the accounts used in the unauthorized access, and is notifying those affected that their information may have been illegally accessed. Nintendo has not released the methods used to exploit their system, but they have made it clear that further security measures were implemented to prevent this kind of attack in the future. While this attack was not as potentially harmful, it goes on to show a continuation of hacking attempts on large-scale gaming services.

In the same year, another prominent gaming hack took place on another large gaming company: Ubisoft. The company made it known to its users that a discovery had been made that one of their websites was used to gain unauthorized access to the company's account database. This information accessed included customer's usernames, email addresses, and encrypted passwords. Instead of it being an outright hack, the company's online network was accessed with stolen credentials [11]. This means that there was not necessarily a fallback with the network's online security as much as their physical security, whether that be on fault of a particular employee or the company as a whole. Ubisoft made it very clear that user's payment information was not compromised, as that information is not stored with Ubisoft [11]. Also, Ubisoft wanted their users to know that although the user's encrypted passwords were accessed, the passwords are not stored as plain text internally. This means that the passwords accessed were obfuscated and cannot be reversed, although they can be cracked if the password was weak. Following up with the unauthorized access, Ubisoft said that they began to explore all available means to expand and strengthen their security measures to prevent something like this from happening in the future.

Just a few years after its first large hack, the Playstation Network along with Microsoft's Xbox live were both attacked on Christmas day 2014. On this day as many as 160 million people were affected and could not play the new games they had acquired only hours previously [12]. A group that called itself Lizard Squad claimed responsibility for attacking both gaming networks. This group used a DDoS (Distributed Denial of Service) attack to take down both gaming services by using fake access requests. This attack was especially harmful on the newer generation gaming consoles, as they usually require online authentication to prevent gaming piracy. Without the Playstation Network, and Xbox live service, gamers were unable to access content for hours. Even users that don't use their consoles for games, but as home entertainment systems, were affected. Since the services were merely taken down, not hacked, no information was stolen from the services. The motive for the attack is unclear, but Lizard Squad threatened to attack both networks at christmas describing themselves as the "next-generation Grinch" [12]. This points to the group performing the attack for no particular reason other than the enjoyment of seeing others unhappy, an act often called trolling. This means that companies may not even suspect themselves as potential targets because of the information they carry, an attacker may perform their malicious act with no particular motive.

An attack of a different type occurred in early 2014 with the popular online game company Big Fish Games. In this case several pieces of user's information was stolen including: customer name, address, payment card number/expiry date/CCV2 code [13]. This information was not acquired by directly hacking the company's internal databases or gaining access to their network. Instead, all of this information was acquired after malware was installed on the billing and payment pages of the Big Fish Games website [13]. This malware intercepted all the customer's payment information as given before. Following the malware's discovery, Big Fish Games took the necessary steps to remove said malware, and prevent it and any other unwanted program from being installed in the future. The company is cooperating with law enforcement and also reported the information theft to credit reporting

agencies so that the proper action may be taken with the customer's card accounts. Furthermore, to help potentially affected customers, a 1-year membership to identity protection services was provided [13].

There were also two notable hacks of a different nature performed on the popular video game Pokemon GO. The first attack was claimed by a group calling themselves PoodleCorp, in which a DDoS attack was performed preventing gamers from entering the gaming platform [14]. Again since the gaming service was taken down instead of hacked, no user's personal information was stolen. One startling fact is that PoodleCorp claimed that this attack was "just a lil test, we will do something on a larger scale soon" [14]. The group also went on to say about targeting the game: "We take the servers offline because it is popular right now and nobody can stop us . . . We do it because we can, nobody can stop us and we just like to cause chaos" [14]. This goes to show that the group had no goal in mind other than to disrupt a popular service to reveal their power and gain notoriety. There was another hack aimed to take out Pokemon GO, but this one without the malicious intent in mind. A group calling itself OurMine said that they launched their assault to encourage the workers behind the game to enhance their internal security. OurMine also launched a DDoS attack on the Pokemon Go game servers. The group went on to say that they would not cease the attack until representatives from the game company contacted members of the hacker group to learn how to protect the game. "We don't want other hackers attack their servers, so we should protect their servers," [14] OurMine said. These two different types of attacks highlight the diversity in goals of attackers to online gaming services. These online hacks were completed by two very different hacker groups, with very different motivations. Another type of hack was performed on the same game, but with a very different approach and goal in mind. The game being as popular as it was, had to be rolled out in several days across several regions in order for the Pokemon GO team to ensure quality of service for its users. This involved the game being released to different geographical locations at different times [14]. Someone unknown tried to exploit this fact to gain access to thousands of mobile devices. An alternate version of the Pokemon Go app for the Android phone was released, this one with malware used to gain unauthorized control of the device that installed the app [14]. All those users that participated in getting the app early, through unauthorized means, exposed themselves to the potential of getting their phones remotely accessed, information stolen, and even payment information used. These attacks in particular bring to light some possible ulterior motives for hackers to attack online gaming services. Hacks on these websites are not always as straightforward as stealing customer's personal and card information.

5 Types of Components Used for Security CU Online Gaming

With cheating, unauthorized access, and piracy now becoming more commonplace in online gaming, game developers have implemented measures to mitigate and

prevent these issues through a variety of components. These components are the result of developers having to adapt to the ever growing and changing problems associated with outdated procedures and services in online game protection. The components that are most commonplace in online gaming security deal with protecting servers and user account data.

A popular means of protecting this data is using services that incorporate a cryptographic system. A cryptographic system [15], as defined by IBM, functions with the use of two parts: an algorithm that en-ciphers and deciphers data, and an encryption key [16]. The encryption key is used by the algorithm to determine an output which is through a relationship with the key and selected data. A Canadian software security company by the name of Clockware software came into business in the early 2000s in order to provide a host of cryptographic services for many game developers [16]. This distribution of an encryption service can be good and bad since game developers would have a form of protection with their product though the frequent use of the same service through multiple games would be easier for criminals to break into. Other companies such as Valve develop their own cryptographic systems which are at less risk of having recognizable assets by hackers.

A more important service that game developers must consider is having secure networks that have minimal risks and a less likelihood of being jeopardized. Video game companies often use network components such as Cisco (Certified Network Professional) which comes with fully functional IP LAN networks with a WAN to connect multiple facilities [8]. Networks provided by distribution companies in of themselves don't offer that much protection and video game developers would need a dedicated lease line to connect safer servers. The lease lines however tend to be very expensive and without proper encryption, dedicated lease lines would be more vulnerable [8]. Server services may come into high risk in situations such a company acquisition. When Zenimax acquired id Software and Arkane studios, great care had to be taken when exchanging servers because any leaked data during the process may compromise entire projects. Server exchange is considered to be one of the highest risks faced in gaming security [8].

One of the most notable measures of combating criminal access through servers and gaming products is with DRM. Digital Rights Management or DRM is a component service used to control how a user accesses a product in order to combat piracy. The Steam platform provides DRM for most of its available games and as a result users are only allowed to have one copy of a game per account that has to be verified before installation [17]. DRM services tend to be vast in their variety among gaming services in order to limit rampant piracy that would otherwise be able to crack a DRM service. There are third party DRM used on top of what is provided with distribution services such as Steam though their use tend to be a higher risk of being hacked. Denuvo is a DRM service that has been known to be cracked by pirates within hours of a game's release and afterwards the game would be illegally sold as DRM free [17]. DRM has also become a topic of contention among gamers since the service is seen as a limitation on consumer ownership and ends up being counterproductive by encouraging piracy.

In order to protect user accounts in online gaming, components are needed to control who has access to an account and how access to an account is obtained. A common authentication component for a user account would be for a user to have a username and password to grant access in their account but this is not enough to stop hackers from illegally accessing their data. To mitigate this, gaming companies have added extra measures in order to secure their users. The Valve has an authenticator called Steam Guard that uses 2FA to give better protection to their users. 2FA (2 Factor Authentication) is an extra step for users to provide information that they would only be able to have knowledge of [18]. This can be in the form of a special code sent to a user's phone number or their personal email. Though the system is not perfect, it has made the case of Steam users encountering account breaches less common.

6 What Current Measures Are Taken to Enforce Components That Are Used in Online Games?

Despite the many methods of components available to combat hacking and piracy, gaming companies have come across cases where current measures taken are not good enough. Newer and more effective components that will fight illegal activity on gaming company products and platforms are needed.

Blizzard Entertainment frequently deals with cheating via hacking on their Battle.Net system (which launches many Blizzard game such as World of Warcraft and StarCraft 2), then leading Blizzard to take various steps to increase user protection. In June of 2008, Blizzard introduced a new layer of authentication under their Blizzard Authenticator system [4]. Originally meant for World of Warcraft accounts, the authenticator adds an extra measure which is to have the user enter a confirmation code sent to them on top of a username and password [4]. The confirmation code also randomly regenerated during different intervals so that there is a lack of consistency for hackers. In February 2010, there was a breach in Battle.Net and this was due to a number of oversights in the authenticator. What happened was that hackers had placed a file in a user's system that intercepted data that was accessed through the authenticator. The hackers would get the valid code while the user in return was given an invalid code for the Blizzard authenticator [4]. As a result, Blizzard updated their system by making every code that can be possibly generated only valid for a single use and for a user to unlink their account, they must submit two consecutive codes.

Better authentication in gaming programs and services greatly increases safety though there are some hindrances such as the assumption that a user isn't careless and wouldn't provide easily accessible information in the authentication system. The guarantee that users are absolutely safe also contributes to negligence to security. Gaming companies should also encourage frequent updates on security systems with up to date scripts [4].

A way Blizzard has handled this negligence is by adding a monitoring system to their games that are launched through Battle.Net. This monitoring system is an embedded program called Warden which scans the computer code of a game process as well as DDLs running in network space [4]. Five of Blizzard's games on Battle.Net use the Warden monitoring service with the intent of better protecting their users. Even though Warden meets the legal standards regarding their EULA that includes a terms of use which can be agreed to by a user, Warden is often discussed as a system that pushed the ethical boundaries of the producer and user relationship [4]. Ultimately Blizzard may have the power to exploit user data which can lead from making improvements to their product and to targeting users with advertisements [4]. Gaming companies can have a limit to how far they go about protecting their users and ethical lines may be blurred to a point that can be counterintuitive to combating online hacking.

One of the most effective ways to prevent hacking in many game systems is to have specialized components that take greater strides with security. White Box Cryptography is a type of specialized encryption that makes it far more difficult to be reverse engineered by hackers. CryptoExperts, a distribution company of cryptographic components for games and apps, delineates white box cryptography as an unintelligible system where the encryption keys cannot be detected. In the encryption system, the keys can only be accessed through a generated white box secure program that implements a cryptographic algorithm [19]. The generated program cannot be read outside data that are inputs and outputs which makes finding the encryption key highly difficult. CryptoExperts also note that white box cryptography is meant to fight against cryptanalysis techniques [19]. These techniques are built into program algorithms that hackers use to analyze data. Clockware's also provides white box cryptography as a specialized service for Canadian game developers. Life for hackers and online pirates has become more difficult with the advent of better and more specialized services available under gaming companies.

7 Considerations to Take When Designing Security Components

The goal of Component-Based Software Development is reusability of software, or the ability to take a piece of software from one project and later implement it in another or multiple projects. While the idea behind it is logically sound, implementation is no easy feat. The choice between in-house and third-party components is the first of many things to take into consideration.

When a developer implements third-party components there is no guarantee that any one component will fill all the developers' (and their clients') needs. Plus, not all third-party components have the same licensing agreements. While some are available as a one-time purchase, others may be pay-per-use or even time-expiring

which is counter-intuitive if the component is security-based. There are also not many repositories to search for components, making it hard to even find one to implement, and even harder to find that is certified to work on the environment being used [20]. Because of their often black-box approach, security-based components are also tough to prove that they are completely secure.

One gaming related example is the rumors surrounding the DRM component, Denuvo. Users have reported that their machine's storage devices started failing after playing games that implemented Denuvo. Supposedly, the DRM service writes a digital signature to SSDs and HDDs at incredibly high speeds in order to verify whether or not the copy of the game is valid. This ended up wearing out their storage devices, and ultimately causing users' machines to fail. This rumor, while completely unsubstantiated as it was never proven to be true, ruined the reputation of games that implemented Denuvo and provide a glimpse at what could potentially happen when implementing third-party components [21].

Another, more proven example would be sendmail, a Unix-email server that was said to have posed security vulnerabilities due to a flaw in its design [22].

All of this, combined with the fact that a component's cohesion with a project is not guaranteed, may drive developers away from implementing third-party security components.

However, developing an in-house component is also not a clear solution as it is often time consuming, and developing security for reuse is even tougher as security standards are constantly changing in the software industry [23]. A security component built today may not necessarily still be secure even 5 years down the line due to depreciation of code or vulnerabilities found in the component itself, ruining its reusability.

8 Recommendations and Conclusion

When including components in a piece of software, especially one that millions of people may use, special security measures must be included. A developer must not only consider one component when creating a security policy, instead several components should be considered and the interactions between all of them. Furthermore, there should not be just one security policy in place for a given task. All security policies should have multiple redundancies to ensure the integrity of the system and prevent there from being a “weakest link” scenario in the overall component composition. In the event that one policy is broken and failing, another backup policy should exist to take its place to keep the system going. One excellent security component in particular is Whitebox cryptography. With the use of this component, developers can ensure that the sensitive data that their software handles is not able to be decrypted and used for malicious purposes.

Even so, cryptography alone is not enough to make a game secure. More security components and policies should be implemented to ensure the security of their game and its clients. Developers may want to implement a system similar to

Blizzard's Warden software which monitors user's actions to prevent any misuse of the software. If any illegitimate actions are detected, the developers can more closely monitor the activity of the account and protect other players that this account is interacting with. Two factor authentication is an effective way to prevent account fraud from taking place, and it keeps accounts in the hands of the original owners. It's harder to gain access to an account protected this way and gives some recourse to the account owner. Anti cheat servers keep online gaming experiences clean and allow players to play as the developers intended, and banning the cheater makes the player pool more predictable. Using these types of security components together can form a level of security that could be considered a baseline of security, allowing players to be able to trust that their accounts, personal information, and most importantly financial information is safe from cheaters, attackers, and hackers. Developers should always ensure that at least some form of these securities are involved in their online gaming environment for the protection of the players.

Developers must make sure that their security components allow the player to play the game, however. Too much involvement for the player can cause a frustrating experience and cause the player to stop playing the game, while too little involvement can cause a lack of security that results in loss of accounts, data, and ultimately money.

However, the biggest thing that a developer can do is to ensure that their software is safe, is to make sure thorough testing takes place, preferably automated testing [24]. Without heavy testing, and careful considerations on how a piece of software is used, an entire software system can be compromised or provide a bad user experience. Worse, a lack of testing can often lead to the security being provided to players being compromised, leading to personal and financial information being stolen and misused. The most important part of security testing is making sure that all components being used work independently, but that they also work together as a unit to provide the best security and user experience to the players playing the game.

The results of the research conducted allows for a better understanding of how online games implement security and how components are used to make those games more secure. The video game industry has become one of the most targeted industries due to the abundance of potential risks in securing user data and gaming exploits. Hackers come up with various methods of obtaining information and exploiting in-game benefits. Without proper protection and attention to security, valuable information is prone to being stolen by criminals through hacking, cheating, and attacking. Since important data could be compromised through illegal intervention, developers have taken security measures to their games and services through various types of components. To strengthen security features for their games, developers cannot just rely on simple and common methods of component implementation, instead they should focus on a variety of up-to-date components and specialized security policies. To conclude, security in online games has been a growing target for criminals since its infancy and developers must adapt to the changes in gaming technology and the methods criminals use to access online game data.

References

1. ESA. (2017). Two-Thirds of American Households Regularly Play Video Games. Retrieved from <http://www.theesa.com/article/two-thirds-american-households-regularly-play-video-games/>
2. Kiwia, D., Dehghanianha, A., Choo, K.-K. R., & Slaughter, J. (2017). A cyber kill chain based taxonomy of banking Trojans for evolutionary computational intelligence. *Journal of Computational Science*. <https://doi.org/10.1016/j.jocs.2017.10.020>
3. Maguluri, N. S. N. (2017). *Multi-Class Classification of Textual Data: Detection and Mitigation of Cheating in Massively Multiplayer Online Role Playing Games*. (Thesis), Wright State University, Browse all Theses and Dissertations.
4. Summeren, R. v. (2011). *Security in Online Gaming*. (Thesis), Radboud University Nijmegen.
5. Chang, H., Park, J. H., & Kang, H. (2008). *The Security System Design in Online Game for u Entertainment*. Paper presented at the 22nd International Conference on Advanced Information Networking and Applications
6. Xiang Zuo, Gandy, C., Skvoretz, J., & Iamnitchi, A. (2016). *Bad Apples Spoil the Fun: Quantifying Cheating in Online Gaming*. Paper presented at the Tenth International AAAI Conference on Web and Social Media
7. Liu, D., Gao, X., Zhang, M., Wang, H., & Stavrou, A. (2017). *Detecting Passive Cheats in Online Games via Performance-Skillfulness Inconsistency*. Paper presented at the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).
8. Mohr, S., & Rahman, S. S. (2011). IT Security Issues Within The Video Game Industry. *International Journal of Computer Science & Information Technology*, 3(5).
9. Seybold, P. (2011). Press Release: Some PlayStation Network and Qriocity Services to be Available This Week. Retrieved from <https://blog.us.playstation.com/2011/04/30/press-release-some-playstation-network-and-qriocity-services-to-be-available-this-week/>
10. Kyodo. (2013). Customer data exposed in huge breach of Nintendo member website. Retrieved from <https://www.japantimes.co.jp/news/2013/07/06/business/corporate-business/japan-customer-data-exposed-in-massive-breach-of-nintendo-member-website/>
11. Steinman, G. (2013). Security update for all Ubisoft account holders. Retrieved from <http://blog.ubi.com/en-GB/security-update-for-all-ubisoft-account-holders/>
12. Kiss, J. (2014). Xbox live and Playstation attack: Christmas ruined for millions of gamers. Retrieved from <https://www.theguardian.com/technology/2014/dec/26/xbox-live-and-psn-attack-christmas-ruined-for-millions-of-gamers>
13. Hurlock-Jones, I. (2015). Multi-state Notification Letter. Retrieved from https://oag.ca.gov/system/files/BFG%20-%20MULTI-STATE%20NOTIFICATION%20LETTER_Proof_1.pdf
14. Paganini, P. (2016). Attacking Gaming Systems: A Dangerous Trend. Retrieved from <http://resources.infosecinstitute.com/attacking-gaming-systems-a-dangerous-trend/>
15. IBM. (2014). The basic elements of a cryptographic system. Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.csf500/csf5za206.htm
16. Gorder, P. F. (2004). Balancing video-game piracy issues. *IEEE Security & Privacy*, 2(1), 17.
17. Mendez, J. (2017). How Steam Employs DRM & What That Means For Your Game. Retrieved from <https://blackshellmedia.com/2017/06/28/steam-employs-drm-means-game/>
18. STEAM. (2017). Account Security Recommendations. Retrieved from https://support.steampowered.com/kb_article.php?ref=1266-OAFV-8478
19. Baignères, T., Finiasz, M., & Lepoint, T. (2013). White-Box Cryptography Retrieved from <https://www.cryptoexperts.com/technologies/white-box/>
20. Ghosh, A. K., & McGraw, G. (1998). *An Approach for Certifying Security in Software Components*. Paper presented at the 21st National Information Systems Security Conference.
21. Arif, F. (2014). Denuvo: SSD Rumors are Wrong – No System is Infallible, Striving to Beat Pirates By Being One Step Ahead. Retrieved from <https://wccftech.com/denuvo-ssd-rumors-wrong-system-infallible-striving-beat-pirates-step/>

22. Crnkovic, I. (2003). *Component-based software engineering - new challenges in software development*. Paper presented at the 25th International Conference on Information Technology Interfaces, 2003. ITI 2003.
23. Vitharana, P. (2003). Risks and challenges of component-based software development. *Communications of the ACM*, 46(8), 67-72.
24. Parizi, R. M., Ghani, A. A. A., Lee, S. P., & Khan, S. U. R. (2017). RAMBUTANS: automatic AOP-specific test generation tool. *International Journal on Software Tools for Technology Transfer*, 19(6), 743-761.