Masterarbeit

Predicting SSH keys in Open SSH Memory dumps

A report by

Rascoussier, Florian Guillaume Pierre

Prüfer
Prof. Dr. Michael Granitzer
Christofer Fellicious
Prof. Dr. Pierre-Edouard Portier

September 4, 2023

Abstract

As the digital landscape evolves, cybersecurity has become an indispensable focus of IT systems. Its ever-escalating challenges have amplified the importance of digital forensics, particularly in the analysis of heap dumps from main memory. In this context, the Secure Shell protocol (SSH) designed for encrypted communications, serves as both a safeguard and a potential veil for malicious activities. This research project focuses on predicting SSH keys in OpenSSH memory dumps, aiming to enhance protective measures against illicit access and enable the development of advanced security frameworks or tools like honeypots.

This Masterarbeit is situated within the broader SmartVMI project, a collaborative research initiative with the objective to advance artificial intelligence-based mechanisms for attack detection and digital forensics. Specifically, this work seeks to build upon existing research on key prediction in OpenSSH heap dumps. Utilizing machine learning algorithms, the study aims to refine feature extraction techniques and explore innovative methods for effective key detection prediction. The objective is to accurately predict the presence and location of SSH keys within memory dumps. This work builds upon, and aims to enhance, the foundations laid by SSHkex [1] and SmartKex [2], enriching both the methodology and the results of the original research while exploring the untapped potential of newly proposed approaches.

This report encapsulates the progress of a year-long Master's thesis research project executed between October 2022 and October 2023. Conducted within the framework of the PhDTrack program between the University of Passau and INSA Lyon, the research has been supervised by Christofer Fellicious and Prof. Dr. Michael Granitzer from the University of Passau, as well as Prof. Dr. Pierre-Edouard Portier from INSA Lyon. It offers an in-depth discussion on the current state-of-the-art in key prediction for OpenSSH memory dumps, research questions, experimental setups, programs development as well as discussing potential future directions.

Acknowledgements

A special acknowledgment goes to Christofer Fellicious, my engaged supervisor at the University of Passau, for his guidance, support and feedback during the Masterarbeit.

I want to express my sincere gratitude to my colleague and friend, Clément Lahoche, whose human and technical skills have been a great source of inspiration and motivation throughout this project; especially considering that we have been working on closely related subjects. It has been a great pleasure to share our ideas and insights, and to collaborate on the development of several programs necessary for the experimentations.

Another acknowledgments go to my esteemed supervisors Prof. Dr. Granitzer and Prof. Dr. Portier for their support and feedback during the Masterarbeit.

I would also like to express my sincere gratitude to all the persons that have helped me, even punctually, during the Masterarbeit with their valuable help, insights, discussions and contributions as well as all the persons involved in the PhDTrack program that made this Masterarbeit possible, including but not limited to:

- Lionel Brunie, Director of CS Department at INSA Lyon, that makes this PhDTrack program possible from the French side.
- Harald Kosch, Head of the Chair of Distributed Information Systems at the University of Passau, that makes this PhDTrack program possible from the German side.
- Natalia Lucari, PhDTrack coordinator at INSA Lyon, for her support and help during the PhD-Track program.
- Ophelie Coueffe, PhDTrack coordinator at the University of Passau, for her support and help during the PhDTrack program.
- Elöd Egyed-Zsigmond, PhDTrack coordinator at the University of Passau, for the subject selection and administrative support.
- All the other PhDTrack students for the great atmosphere, mutual help and the interesting discussions during almost two years.

Finally, my last acknowledgments go to my family and friends for their support and encouragements.

Contents

1 Introduction

The digital age has brought with it an unprecedented increase in the volume and complexity of data that is being generated, stored, and processed. This data is often sensitive in nature, and its security is of paramount importance, making cybersecurity a critical focus area. This evolving landscape is fraught with challenges that continue to amplify the importance of digital forensics in IT systems. One area that stands out for its widespread use and importance is the Secure Shell protocol (SSH) and its most popular implementation, OpenSSH. SSH is a cryptographic network protocol widely used for secure remote access to systems. It is also used for secure file transfer, and as a secure tunnel for other applications. SSH is a key component of IT systems whose encryption capabilities are critical to the security of IT systems. However, it also presents a unique set of challenges, most notably the concealment of malicious activities.

A common case is when a unauthorized actor gains access to SSH keys so as to get access to a system. This can happen through a malicious human actor, but more commonly through automated processes such as malwares and botnets. This situations present a formidable and growing threat to cybersecurity, affecting a broad range of stakeholders from governments and financial institutions to individual users. In just 2019, the number of Command and Control (C&C) servers for botnets increased by 71.5%, leading to an estimated $19 billion in advertising theft [4]. Many malwares and botnets "have in common that they have used as attack vector the Secure Shell (SSH) remote access service" [4].

At the heart of the issue lies the fact that SSH veils its communications through encryption, making it difficult to detect malicious activities. To be able to detect those potential malicious actors, it is possible to replace SSH by a honeypot that enable to monitor pseudo-SSH activities. There is a range of readily available honeypots, such as Kippo or Cowrie, which are designed to emulate a vulnerable SSH system and attract attackers [13]. The problem lies that thoses honeypots are not able to mimic perfectly a real system, which makes them easy to detect by experienced attackers. As stated by „Analysis of SSH Honeypot Effectiveness": "The ability to collect meaningful malware from attackers depends on how the attackers receive the honeypot. Most attackers fingerprint targets before they launch their attack, so it would be very beneficial for security researchers to understand how to hide honeypots from fingerprinting and trick the attackers into depositing malware. [...] What is certain is that if a cautious attacker believes they are in a honeypot, they will leave without depositing malware onto the system, which reduces the effectiveness of the honeypot" [14].

The are other approaches that allows to decrypt SSH connections without relying on a honeypot, like the *man-in-the-middle* or *binary manipulation* with their own set of challenges [1]. Instead of relying on softwares that mimics or modify a real system, it is possible to use a real unmodified system directly. The idea is to be able to decrypt SSH connection channels, which is possible if the SSH keys are known. Since SSH encryption keys are typically stored in the main memory of a system, it is possible for the administrators to extract them through the exploitation of memory dumps of a targeted system. In this context, the ability to detect SSH keys in memory dumps, and specifically OpenSSH keys, is critical to the development of effective SSH honeypot-like systems. The research introduced by the SmartVMI project with SSHKex, SmartKex, the present thesis and the future related work could be used to develop such a new type of system-monitoring tools. This new kind of tools would be

very difficult to detect by attackers, increasing their effectiveness, and wouldn't require the alteration of the system. The present report is focused on the SSH key detection in memory dumps, which is a key component allowing to decode SSH communications such that it become possible to intercept malicious communications and to detect malicious activities.

2 Research Questions

At the very beginning of this thesis, the objective was to answer the following research questions:

- RQ1: What is the state of the art in the field of security key detection in heap dump memory?
- RQ2: What are the challenges of security key detection in heap dump memory?
- RQ3: How can the existing methods for detecting SSH keys in OpenSSH heap dumps be improved?

The SmartVMI project has already made significant progress in the detection of SSH keys in OpenSSH heap dumps. An open dataset of memory dumps has been created, and a simple yet effective method for detecting SSH keys has been developed. The dataset has been used to train and test simple machine learning algorithms, and the results have been promising. The research has been published in the form of two papers, SSHkex [1] and SmartKex [2], which is the basis of this thesis.

However, there is still room for improvement, particularly in the area of machine learning algorithms. This thesis seeks to build upon the existing research by refining feature extraction techniques and exploring innovative methods for effective key detection prediction. The objective is to accurately predict the presence and location of SSH keys within memory dumps. Rooted in this context, this Masterarbeit aims to address several key research questions:

- RQ4: What features are most indicative of SSH keys in memory dumps?
- RQ5: How can these features be extracted from memory dumps and used to train machine learning algorithms?
- RQ6: How can machine learning algorithms be optimized for the prediction of SSH keys in memory dumps?

By tackling these research questions, this thesis seeks not only to advance the academic understanding of SSH key prediction and digital forensics but also to provide practical insights that could lead to the development of more secure and effective systems.

3 Structure of the Thesis

4 Background

The evolving landscape of cybersecurity necessitates robust techniques for safeguarding digital communications. OpenSSH, a pivotal element in this landscape, is a popular implementation of the Secure Shell (SSH) protocol, which enables secure communication between two networked devices. The protocol is widely used in the industry, particularly in the context of remote access to servers. Using digital forensic techniques, it is possible to extract the SSH keys from memory dumps, which can then be used to decode encrypted communications thus allowing the monitoring of controlled systems. At the crux of this Masterarbeit is the development of machine learning algorithms to predict SSH keys within these heap dumps, focusing on using graph-like-structures and vectorization for custom embeddings. With an interdisciplinary approach that fuses traditional feature engineering with graph-based methods as well as memory modelization for inductive reasoning and learning inspired by recent developments in Knownledge Graph (KG)s, this research not only leverages existing machine learning paradigms but also explores new avenues, such as Graph Convolutional Networks (GCN).

The objective of this background section is multifaceted. Since the project has seen two distinct phases, one more classical Machine Learning (ML) approach, and a second one centered aroung graph-based advanced learning methods, the background section is divided into several subsections that introduce the reader to the different concepts and techniques used in the project. First, it aims to offer an overview of SSH protocols, particularly focusing on OpenSSH key implementations subsection 4.1. Second, it delineates the dataset and prior work on key extraction techniques including SmartKey subsection 4.2. Third, it delves into the technical aspects of graphs modelization subsection 4.3, followed by feature engineering and embeddings both traditional and graph-based subsection 4.4. Finally, it addresses the machine learning models employed in the research, emphasizing their suitability for maximizing recall in key prediction subsection 4.5. By fusing these distinct but interrelated areas, this section lays the foundation for the research methodologies and hypotheses tested in this study.

4.1 SSH and OpenSSH Implementation

4.1.1 Basics of the Secure Shell Protocol (SSH)

The Secure Shell Protocol, commonly known as SSH, is designed to facilitate secure remote login and other secure network services over insecure networks. SSH has been design since its inception with security in mind, as a successor of the Telnet protocol, which is not secure, and other "unsecured remote shell protocols such as rlogin, rsh and rexec" [1]. As stated by the authors of the *SSH Annual Report 2018*, "The founder of SSH, Tatu Ylönen, designed the first version of the SSH protocol after a password-sniffing attack at his university network. Tatu released his implementation as freeware in July 1995, and the tool quickly gained in popularity. Towards the end of 1995, the SSH user base had grown to 20,000 users in fifty countries. By 2000, there were an estimated 2,000,000 users of the protocol. Today, more than 95% of the servers used to power the Internet have SSH installed in them. The SSH protocol is truly one of the cornerstones of a safe Internet." [5].

SSH is defined in *The Secure Shell (SSH) Protocol Architecture* [3]. It is divided into three major components:

- **Transport Layer Protocol:** This provides server authentication, confidentiality, and integrity. It can also optionally provide compression. Typically, the transport layer runs over a TCP/IP connection but can also be used on top of any other reliable data stream.
- **User Authentication Protocol:** Running over the transport layer, this protocol authenticates the client-side user to the server. Multiple methods of authentication such as password and public key are supported.
- **Connection Protocol:** This multiplexes the encrypted tunnel established by the preceding layers into several logical channels. Channels can be used for various purposes, such as setting up secure interactive shell sessions or tunneling arbitrary TCP/IP ports.

"The client sends a service request once a secure transport layer connection has been established. A second service request is sent after user authentication is complete. This allows new protocols to be defined and coexist with the protocols listed above" [3].

For the scope of this Masterarbeit, understanding SSH's key exchange and encryption mechanism is important. As Summarized in SSHKex [1], the SSH key exchange procedure results in a derived master key K and a hash value h. These are critical for client-server communication encryption and session identification.

During the key exchange, multiple session keys are computed for different purposes:

- **Initialization Vectors:** Key A and Key B are used for initialization vectors from the client to the server and vice versa.
- **Encryption Keys:** Key C and Key D serve as encryption keys for client-to-server and server-to-client communications, respectively.
- **Integrity Keys:** Key E and Key F are used to maintain the integrity of the data transmitted between the client and server.

These keys are computed using hash functions that take the master key K and a hash value h, a unique letter (A, B, C, D, E, or F), and the session ID as inputs. Decrypting encrypted SSH communication thus necessitates either to retreive these session keys and variables so as to recompute the keys, or to retreive those keys directly, which is the focus of this Masterarbeit.

4.1.2 OpenSSH Implementation

OpenSSH is an open-source implementation of the SSH protocol suite, and it is the most widely used SSH implementation. It is the default SSH implementation on most Linux distributions, and it is also available for Windows. OpenSSH is used for a wide range of purposes, including remote command-line login and remote command execution. It is also used for port forwarding, tunneling, and transferring files via SCP and SFTP. It is also used by many automated processes, such as backup systems, configuration management tools, and automated software deployment tools.

4.1.3 The state of SSH security

Since its origins, SSH has been developed with cybersecurity in mind, and is generally considered a secure method for remote login and other secure network services over an insecure network. However, as with any technology, it can be exploited if not configured or managed correctly. The protocol is used by system administrators to manage remote systems, and it is also used by automated processes to transfer data and perform other tasks. This makes SSH a valuable target for attackers. In fact, SSH has been a popular target for cyber-attacks. Due to being so prevalent, it is often used by threat actors either as a vector for initial access, as a means to move laterally across a network or as a covered exit for exfiltration of sensitive data [12]. The encrypted nature of its communications makes it an attractive option for attackers, as it can be difficult to detect malicious activity.

Here are some cases where SSH can involved in cyber-attacks, although it's important to note that SSH itself is not inherently insecure:

- **SSH Brute-Force Attacks:** One of the most common types of attacks involving SSH is a brute-force attack, where an attacker tries to gain access by repeatedly attempting to login with different username-password combinations. These attacks are not sophisticated but can be effective if strong authentication measures are not in place. For instance, the botnet *Chabulo* was used to launch a large-scale brute-force attack "through compromised SSH servers and IoT devices" in 2018 [5].
- **SSH Key Theft:** In some advanced attacks, threat actors have stolen SSH keys to move laterally across a network after initial entry. This allows them to authenticate as a legitimate user and can make detection much more challenging. It can " occurs when users have their SSH password or unencrypted keys stolen through a variety of methods (sniffed via a key-logging console program, shoulder-surfed via bad security awareness, poor key management practices, etc.)." [6].
- **Man-in-the-Middle Attacks:** Although SSH is designed to be secure, it can be susceptible to man-in-the-middle attacks if proper verification of SSH keys is not done during the initial connection setup.
- **Misconfiguration:** As with any technology, misconfiguration can lead to security issues. For example, leaving default passwords, using weak encryption algorithms, or enabling root login can all make an SSH-enabled system vulnerable.

4.1.4 SSH vulnerabilities and use in cyber-attacks

In cybersecurity, it is generally considered that any system that is connected to the Internet will be attacked at some point. Similarly, it is a common saying that no system is 100% secure. This is true for SSH as well. Although it is a secure protocol, it can be exploited if not configured or managed correctly.

Some vulnerabilities have also been discovered in the protocol itself, although these are rare.

- **SSH-1 Vulnerabilities:** A series of vulnerabilities in the first implementation of SSH were

discovered from 1998 to 2001, with its subsequent fixes leading to unauthorized content insertion and arbitrary code execution. SSH-1 had many design flows and is now considered obsolete. [8], [7].

- **CBC Plaintext Recovery:** A theoretical vulnerability discovered in 2008 affecting all versions of SSH, allowing the recovery of up to 32 bits of plaintext from CBC-encrypted ciphertext [9].
- **Suspected Decryption by NSA:** Leaked information in 2014 suggested that the NSA might be able to decrypt some SSH traffic, although the protocol itself was not confirmed to be compromised [10].

SSH has been used in many high-profile cyber-attacks and malwares, including the following:

- **Operation Windigo:** This was a large-scale campaign that infected over 25,000 UNIX servers. SSH was one of the vectors used for maintaining control over compromised servers. A report by ESET mentions that the OpenSSH backdoor Linux/Ebury was first discovered in 2011 as a component of the aforementioned operation. "This operation has been ongoing since at least 2011 and has affected high profile servers and companies, including cPanel - the company behind the famous web hosting control panel - and Linux Foundation's kernel.org - the main repository of source code for the Linux kernel" [11].
- **Linux/Hydra:** Initially unleashed in 2008, this malware is a fast login cracker that targets a range of popular protocols including SSH. Hence, SSH is one of its primary vectors to gain initial access to Internet of Things (IoT) devices. Once a device is infected by Linux/Hydra, it joins an IRC channel and initiates a SYN Flood attack [13].
- **Psyb0t:** Discovered in early 2009, Psyb0t is an IRC-controlled malware specifically designed to target devices with MIPS architecture, such as routers and modems. Notably, it was responsible for orchestrating a DDoS attack against the DroneBL service, infecting up to 100,000 devices for this purpose. The malware is equipped to conduct UDP and ICMP flood attacks and employs a brute-force attack mechanism against Telnet and SSH ports. Remarkably, it uses a pre-configured list of 6,000 usernames and 13,000 passwords to perform these attacks [13].
- **Chuck Noris:** Similar to Psyb0t in its objectives and methods, Chuck Noris targets routers and DSL modems, focusing on SoHo (small office/home office) devices. However, unlike Psyb0t, which uses ICMP flood attacks, Chuck Noris deploys ACK flood attacks. The malware carries out brute-force attacks on Telnet and SSH open ports, drawing parallels to the tactics employed by Psyb0t but with the specific variation in flooding techniques [13].

It's worth noting that in many of these cases, SSH was not the initial attack vector but was used at some stage in the attack lifecycle. Properly configured and managed SSH is still considered a secure and robust protocol for remote access and data transfer.

4.1.5 The Imperative of SSH Honeypots in Cybersecurity Monitoring

SH (Secure Shell) has become an indispensable protocol for secure communication but can also conceil malicous agents. This reality underscores the urgency for robust monitoring mechanisms capable of

identifying suspicious activities in real-time. Among various countermeasures, SSH honeypots have emerged as a particularly effective tool for monitoring and gathering intelligence on potential threats.

An SSH honeypot is a decoy server or service that mimics legitimate SSH services. The primary aim is to attract cybercriminals and study their tactics, thereby offering an active form of surveillance and data collection. Unlike traditional intrusion detection systems, honeypots do not merely identify an attack; they engage the attacker in a controlled environment, enabling detailed observation and logging of the intruder's actions. This allows for the collection of valuable information, such as the attacker's IP address, the tools used, and the techniques employed. This data can then be used to enhance security measures and develop more robust countermeasures [13].

SSH honeypots serve as an invaluable asset in the cybersecurity arsenal, providing not just a reactive but a proactive measure against evolving cyber threats. They can collect actionable intelligence on new hacking methods, malware, and exploitation scripts. This information can be crucial for proactively securing actual production environments. The data collected can also be used to trace back to the origin of the attack, facilitating legal pursuits against the perpetrators. By diverting attackers to decoy servers, honeypots also protect real assets from being targeted, saving both computational resources and administrative effort needed for post-incident recovery.

Popular SSH honeypots include Kippo, Cowrie, and HoneySSH. Cowrie is a fork of Kippo, with additional features such as logging of attacker's keystrokes and file transfer.

- **Kippo:** Kippo is a medium-interaction honeypot that logs the attacker's shell interaction. It specializes in capturing brute force and Telnet-based attacks [13].
- **Cowrie:** Serving as Kippo's successor, Cowrie emulates various protocols including SSH, SFTP, and SCP. It logs events in JSON format, making it particularly useful for detecting brute force and Telnet-based attacks, as well as spoofing attacks [13].
- **IoTPOT:** This IoT-focused honeypot supports multiple CPU architectures and can detect a variety of attacks including brute force, DoS, and sniffing attacks on Telnet, SSH, and HTTP ports [13].
- **HoneySSH:** HoneySSH is a low-interaction honeypot that emulates an SSH server and logs the attacker's IP address, username, and password [15].

These honeypots are useful tools for gathering intelligence on potential threats. However, they are not without their limitations. For instance, they are not able to mimic a real system, such that attackers might be able to detect them „Analysis of SSH Honeypot Effectiveness“. Hence, the need for more advanced SSH honeypots that can leverage data forensic and machine learning techniques so as to be able to use directly a real server as a honeypot, without the need to emulate a system.

4.2 Prior work on key extraction

4.2.1 SSHKex

4.2.2 OpenSSH memory dumps dataset

4.2.3 SmartKey

4.3 Graph-based memory modelization

4.3.1 Defining memory concepts and modelization

4.3.2 Graphs and Knowledge Graphs

4.4 Data processing for Machine Learning

4.4.1 Feature engineering

4.4.2 Graph-based embeddings

4.4.3 Dataset splitting and sampling

4.5 Machine Learning and Deep Learning

4.5.1 Machine Learning

4.5.2 Machine Learning models

4.5.3 Deep Learning

4.5.4 GCN

5 Methods

Describe the method/software/tool/algorithm you have developed here

Dataset in Methods Chapter: On the other hand, if the specific features and challenges of your dataset are more related to your methodology—for example, how you cleaned, balanced, or sampled the data—then these details would fit well into the Methods chapter. This is especially relevant if the issues tackled are more about "how-to" rather than "why."

6 Results

Describe the experimental setup, the used datasets/parameters and the experimental results achieved

7 Discussion

Discuss the results. What is the outcome of your experimetns?

8 Conclusion

Summarize the thesis and provide a outlook on future work.

A Code

B Math

C Dataset

Acronyms

DEL Directed Edge-labelled Graphs. 3

ER Entity Resolution. 5

GCN Graph Convolutional Networks. 11

GNN Graph Neural Network. 11

KE Knowledge Engineering. 5, 9

KG Knownledge Graph. i, 1

ML Machine Learning. 7

NLP Natural Language Processing. i

QA Quality Assurance. 5

RDF Resource Description Framework. 3, 5, 6

SPARQL SPARQL Protocol and RDF Query Language. 5

SSH Secure Shell Protocol. i

References

[1] Stewart Sentanoe and Hans P. Reiser. „SSHkex: Leveraging virtual machine introspection for extracting SSH keys and decrypting SSH network traffic“. en. In: *Forensic Science International: Digital Investigation* 40 (2022), p. 301337. DOI: 10.1016/j.fsidi.2022.301337. URL: https://linkinghub.elsevier.com/retrieve/pii/S2666281722000063.

[2] Christofer Fellicious et al. „SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump“. In: arXiv:2209.05243 (Sept. 2022). arXiv:2209.05243 [cs]. DOI: 10.48550/arXiv.2209.05243. URL: http://arxiv.org/abs/2209.05243.

[3] T. Ylonen and C. Lonvick. *The Secure Shell (SSH) Protocol Architecture*. RFC 4251. Updated by: 8308, 9141. Network Working Group, Jan. 2006.

[4] José Tomás Martínez Garre, Manuel Gil Pérez, and Antonio Ruiz-Martínez. „A novel Machine Learning-based approach for the detection of SSH botnet infection“. In: *Future Generation Computer Systems* 115 (Feb. 2021), pp. 387–396. DOI: 10.1016/j.future.2020.09.004. URL: https://www.sciencedirect.com/science/article/pii/S0167739X20303265.

[5] SSH Communications Security. *SSH Annual Report 2018*. Annual Report. Accessed: 2023-08-30. SSH Communications Security, 2018. URL: https://info.ssh.com/hubfs/2021%20Investor%20documents/SSH_Annual_Report_2018_final.pdf.

[6] W. Yurcik and Chao Liu. „A first step toward detecting SSH identity theft in HPC cluster environments: discriminating masqueraders based on command behavior“. In: *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.* Vol. 1. May 2005, 111–120 Vol. 1. DOI: 10.1109/CCGRID.2005.1558542.

[7] *Weak CRC allows packet injection into SSH sessions encrypted with block ciphers*. Accessed: 2023-08-30. Nov. 2001. URL: https://www.kb.cert.org.

[8] Core Security Technologies. *SSH Insertion Attack*. https://www.coresecurity.com/core-labs/advisories/ssh-insertion-attack. Archived from the original on 2011-07-08. 2023.

[9] US CERT. *SSH CBC vulnerability. Vulnerability Note VU#958563 - SSH CBC vulnerability*. https://www.kb.cert.org/vuls/id/958563. Archived from the original on 2011-06-22. 2011.

[10] Spiegel Online. *Prying Eyes: Inside the NSA's War on Internet Security*. Spiegel Online. Archived from the original on January 24, 2015. 2014. URL: https://www.spiegel.de/international/germany/inside-the-nsa-s-war-on-internet-security-a-1010361.html.

[11] Olivier Bilodeau et al. *Operation WINDIGO*. en. Mar. 2014, p. 69. URL: https://web-assets.esetstatic.com/wls/2014/03/operation_windigo.pdf.

[12] Pooneh Nikkhah Bahrami* et al. „Cyber Kill Chain-Based Taxonomy of Advanced Persistent Threat Actors: Analogy of Tactics, Techniques, and Procedures“. In: *Journal of Information Processing Systems* 15.4 (Nov. 2019), pp. 865–889. DOI: 10.3745/JIPS.03.0126. URL: http://xml.jips-k.org/full-text/view?doi=10.3745/JIPS.03.0126.

[13] Sanjay Madan and Monika Singh. „Classification of IOT-Malware using Machine Learning“. In: *2021 International Conference on Technological Advancements and Innovations (ICTAI)*. Nov. 2021, pp. 599–605. DOI: 10.1109/ICTAI53825.2021.9673185.

[14] Connor Hetzler, Zachary Chen, and Tahir M. Khan. „Analysis of SSH Honeypot Effectiveness“. en. In: *Advances in Information and Communication*. Ed. by Kohei Arai. Lecture Notes in Networks and Systems. Cham: Springer Nature Switzerland, Mar. 2023, pp. 759–782. ISBN: 9783031280733. DOI: 10.1007/978-3-031-28073-3_51.

[15] ppacher. *honeyssh*. https://github.com/ppacher/honeyssh. GitHub repository. 2017.

Additional bibliography

[16] Dimitrios Georgoulias et al. „Botnet Business Models, Takedown Attempts, and the Darkweb Market: A Survey“. en. In: *ACM Computing Surveys* 55.11 (Nov. 2023), pp. 1–39. DOI: 10.1145/3575808. URL: https://dl.acm.org/doi/10.1145/3575808.

[17] Aidan Hogan et al. „Knowledge Graphs“. In: *ACM Comput. Surv.* 54.4 (July 2021). ISSN: 0360-0300. DOI: 10.1145/3447772. URL: https://doi.org/10.1145/3447772.

[18] Paul Groth et al. „Knowledge Graphs and their Role in the Knowledge Engineering of the 21st Century“. In: *Dagstuhl Reports* 12.9 (2022). Report from Dagstuhl Seminar 22372. Specific usage: pp. 60-72, Subsection "3.2 A Brief History of Knowledge Engineering: A Practitioner's Perspective", pp. 60–120. DOI: 10.4230/DagRep.12.9.60.

[19] Marvin Hofer et al. „Construction of Knowledge Graphs: State and Challenges“. In: *arXiv preprint arXiv:2302.11509* (2023). URL: https://doi.org/10.48550/arXiv.2302.11509.

[20] Lisa Ehrlinger and Wolfram Wöß. „Towards a Definition of Knowledge Graphs“. In: (2016), pp. 1–4.

[21] Frederick Edward Hulme. *Proverb Lore: Many Sayings, Wise Or Otherwise, on Many Subjects, Gleaned from Many Sources*. E. Stock, 1902, p. 188.

[22] Google. „Introducing the Knowledge Graph: Things, not strings“. In: *Google Blog* (May 2012). Accessed: 2023-06-16. URL: https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[23] Michelle Venables. *An Introduction to Graph Theory*. Accessed: 2023-06-12. 2019. URL: https://towardsdatascience.com/an-introduction-to-graph-theory-24b41746fabe.

[24] Gianluca Fiorelli. *Best of 2013: No 13 - Search in the Knowledge Graph era*. Accessed: 2023-06-12. 2013. URL: https://www.stateofdigital.com/search-in-the-knowledge-graph-era/.

[25] Jackson Gilkey. *Graph Theory and Data Science*. Accessed: 2023-05-25. 2019. URL: https://towardsdatascience.com/graph-theory-and-data-science-ec95fe2f31d8.

[26] M.S. Jawad et al. „Adoption of knowledge-graph best development practices for scalable and optimized manufacturing processes“. In: *MethodsX* 10 (2023), p. 102124. ISSN: 2215-0161. DOI: https://doi.org/10.1016/j.mex.2023.102124. URL: https://www.sciencedirect.com/science/article/pii/S2215016123001255.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie, dass ich die Masterarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Passau, September 4, 2023

Rascoussier, Florian Guillaume Pierre