UNIVERSITÄT PASSAU

Masterarbeits

Structure embeddings for OpenSSH heap dump analysis

A report by

Lahoche, Clément Claude Martial

Prüfer

Prof. Dr. Michael Granitzer

Christofer Fellicious

Prof. Dr. Pierre-Edouard Portier

August 24, 2023

Abstract

Acknowledgements

Contents

List of Figures

List of Tables

1 Introduction

Digital forensics is a linchpin in cybersecurity, enabling the extraction of vital evidence from devices like PCs. This evidence is key for detecting malware and tracing intruder activities. Analyzing a device's main memory is a go-to technique in this field. The fusion of machine learning promises to amplify and streamline these analyses.

With the rising need for encrypted communication, Secure Shell (SSH) protocols are now commonplace. However, these security-focused channels can inadvertently shield malicious actions, posing challenges to standard investigative approaches. Cutting-edge research offers solutions. The work in *SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump* [3] highlights how machine learning can boost the extraction of session keys from OpenSSH memory images. In a complementary vein, „SSHkex: Leveraging virtual machine introspection for extracting SSH keys and decrypting SSH network traffic“ [7] showcases the power of Virtual Machine Introspection (VMI) for direct SSH key extraction.

Inspired by *SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump*, this thesis zeroes in on a central challenge: data embedding. While previous studies set the stage for key extraction, the data embedding technique, especially windowing, can be optimized. The design of data embeddings is pivotal for machine learning efficacy, especially in nuanced tasks like memory analysis. This research introduces fresh embedding strategies, aiming to refine extraction and unearth deeper memory snapshot patterns. Merging graph embeddings with advanced machine learning, the goal is to craft a sophisticated toolkit for OpenSSH heap dump studies, bridging digital forensics and machine learning.

2 Research Questions

Write down and explain your research questions (2-5)

3 Structure of the Thesis

Explain the structure of the thesis.

4 Background

In the complex world of cybersecurity and digital forensics, innovative approaches are crucial for revealing hidden or encrypted information. OpenSSH stands out as a key instrument for ensuring secure communication. The memory snapshots, or heap dumps, of OpenSSH are treasure troves of data. Through graph generation from these dumps, we can uncover the detailed connections between data structures, identified by their malloc headers, and their associated pointers.

This research delves deep into the smart embedding of these connections, aiming to use machine learning classifiers to identify structures that contain OpenSSH keys. The journey is not just about representing data through graphs but also about understanding the raw sequences of bytes in the heap dump. Classical techniques like Shannon entropy, Byte Frequency Distribution (BFD), and bigram frequencies provide foundational knowledge. However, the rapidly evolving domain of deep learning opens up a plethora of avenues. Models such as Recurrent Neural Networks (RNN) [6] (Long Short-Term Memory (LTSM)[5] and Gated Recurrent Units (GRU)[2]) and sequence-to-sequence learning [8] offer unique perspectives on raw byte embedding. The transformative power of attention mechanisms, as highlighted by the transformer architecture[9]. Furthermore, the efficacy of convolutional approaches (CNN), both standalone and in conjunction with recurrent networks, for sequence modeling is well-documented [1]. Notably, the application of neural networks in file fragment classification, especially with lossless representations, has shown promising results [4].

The aim of this background section is to provide a comprehensive overview of graph creation from heap dumps, techniques for raw byte embedding, and their role in identifying OpenSSH key structures. By merging age-old techniques with modern approaches, we strive to highlight the most effective methods for analyzing OpenSSH heap dump.

4.1 Graph Generation from Heap Dumps

4.1.1 Secure Shell (SSH)

The Secure Shell (SSH) is designed to enable encrypted communication across potentially unsecured networks, ensuring the confidentiality of data during transmission. Each SSH session utilizes a specific set of session keys, encompassing six distinct keys:

- **Key A:** Client-to-server initialization vector (IV)
- **Key B:** Server-to-client initialization vector (IV)
- **Key C:** Client-to-server encryption key (EK)
- **Key D:** Server-to-client encryption key (EK)
- **Key E:** Client-to-server integrity key
- **Key F:** Server-to-client integrity key

To decrypt the encrypted traffic within an SSH session, knowledge of the IV and EK pair (either Key A with Key C or Key B with Key D) is essential, assuming the presence of passive network monitoring tools. OpenSSH, a prevalent implementation of SSH, is the primary subject of this research, covering versions from V6_0P1 to V8_8P1. OpenSSH incorporates various encryption methodologies, including Advanced Encryption Standard (AES) Cipher Block Chaining (CBC), AES Counter (AES-CTR), and ChaCha20, with IV and EK key lengths varying between 12 and 64 bytes. This information is derived from the paper titled *SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump* [3].

4.1.2 heap dumps of OpenSSH

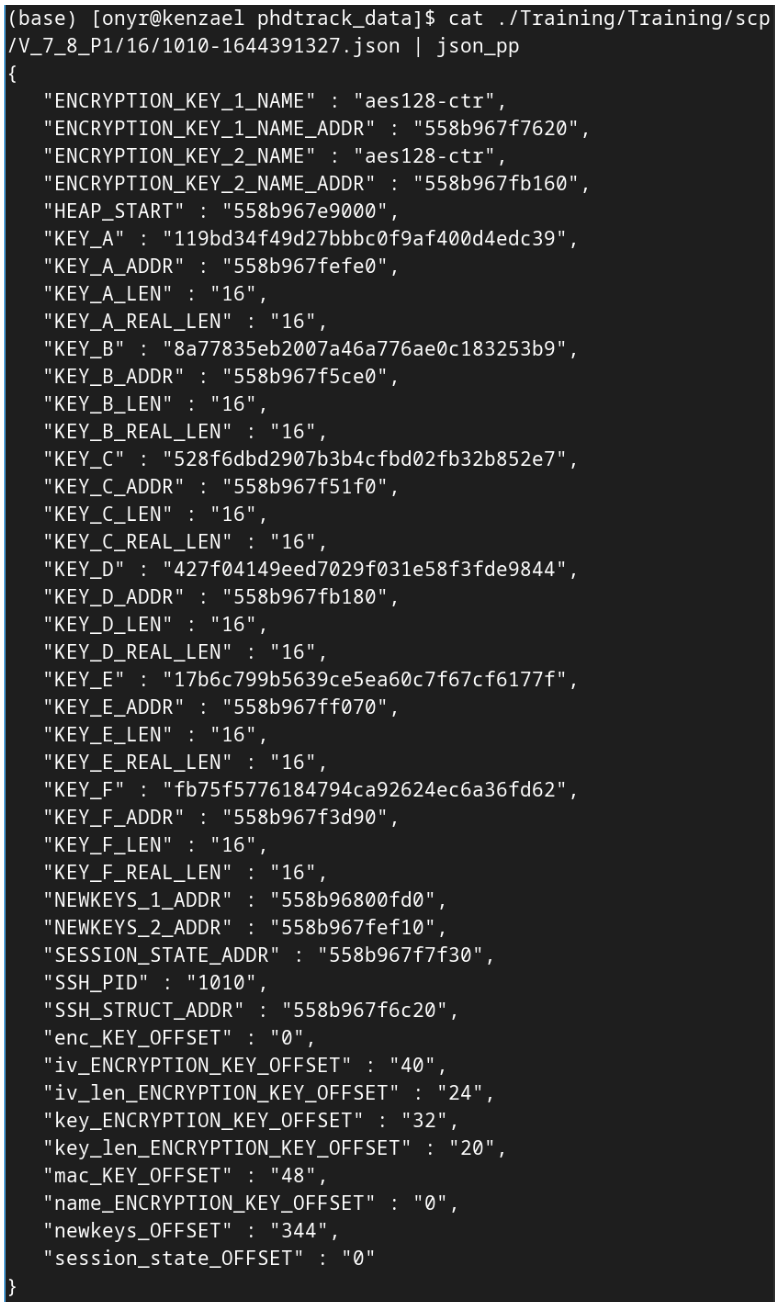
Heap memory, distinct from local stack memory, is a dynamic memory allocation mechanism. While local stack memory is responsible for storing and deallocating local variables during function calls, heap memory requires explicit memory allocation and deallocation. This is achieved using operators such as `new` in Java and C++, or `malloc/calloc` in C.

OpenSSH, which is primarily written in C, employs `calloc` for memory block allocation. These blocks are designated to store session-related data, including the cryptographic keys. By leveraging this knowledge, one can deduce that if the heap of an active OpenSSH process is dumped at an opportune moment (for instance, during an ongoing SSH session), the resulting heap dump will encompass the SSH session keys. This information is also derived from the paper titled *SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump* [3].

4.1.3 Dataset

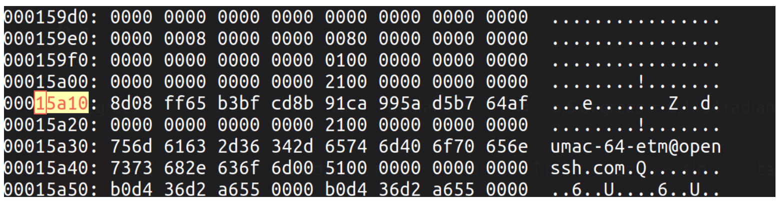
We use SSHKex[7] as the primary method to extract the SSH keys from the main memory. In addition, we add two features to SSHKex: automatically dump OpenSSH's heap and add support for SSH client monitoring.

For this paper, we are using four SSH scenarios: the client connects to the server and exits immediately, port-forward, secure copy, and SSH shared connection. Two file formats, JSON and RAW, are utilized to store the generated logs. The JSON log file encompasses meta-information, including the encryption name, the virtual memory address of a key, and the key's value in hexadecimal representation (as depicted in Figure 1). Conversely, the binary file captures the heap dump of the OpenSSH process (illustrated in Figure 2 using the `xxd` command).



```
(base) [onyr@kenzael phdtrack_data]$ cat ./Training/Training/scp
/V_7_8_P1/16/1010-1644391327.json | json_pp
{
   "ENCRYPTION_KEY_1_NAME" : "aes128-ctr",
   "ENCRYPTION_KEY_1_NAME_ADDR" : "558b967f7620",
   "ENCRYPTION_KEY_2_NAME" : "aes128-ctr",
   "ENCRYPTION_KEY_2_NAME_ADDR" : "558b967fb160",
   "HEAP_START" : "558b967e9000",
   "KEY_A" : "119bd34f49d27bbbc0f9af400d4edc39",
   "KEY_A_ADDR" : "558b967fefe0",
   "KEY_A_LEN" : "16",
   "KEY_A_REAL_LEN" : "16",
   "KEY_B" : "8a77835eb2007a46a776ae0c183253b9",
   "KEY_B_ADDR" : "558b967f5ce0",
   "KEY_B_LEN" : "16",
   "KEY_B_REAL_LEN" : "16",
   "KEY_C" : "528f6dbd2907b3b4cfbd02fb32b852e7",
   "KEY_C_ADDR" : "558b967f51f0",
   "KEY_C_LEN" : "16",
   "KEY_C_REAL_LEN" : "16",
   "KEY_D" : "427f04149eed7029f031e58f3fde9844",
   "KEY_D_ADDR" : "558b967fb180",
   "KEY_D_LEN" : "16",
   "KEY_D_REAL_LEN" : "16",
   "KEY_E" : "17b6c799b5639ce5ea60c7f67cf6177f",
   "KEY_E_ADDR" : "558b967ff070",
   "KEY_E_LEN" : "16",
   "KEY_E_REAL_LEN" : "16",
   "KEY_F" : "fb75f5776184794ca92624ec6a36fd62",
   "KEY_F_ADDR" : "558b967f3d90",
   "KEY_F_LEN" : "16",
   "KEY_F_REAL_LEN" : "16",
   "NEWKEYS_1_ADDR" : "558b96800fd0",
   "NEWKEYS_2_ADDR" : "558b967fef10",
   "SESSION_STATE_ADDR" : "558b967f7f30",
   "SSH_PID" : "1010",
   "SSH_STRUCT_ADDR" : "558b967f6c20",
   "enc_KEY_OFFSET" : "0",
   "iv_ENCRYPTION_KEY_OFFSET" : "40",
   "iv_len_ENCRYPTION_KEY_OFFSET" : "24",
   "key_ENCRYPTION_KEY_OFFSET" : "32",
   "key_len_ENCRYPTION_KEY_OFFSET" : "20",
   "mac_KEY_OFFSET" : "48",
   "name_ENCRYPTION_KEY_OFFSET" : "0",
   "newkeys_OFFSET" : "344",
   "session_state_OFFSET" : "0"
}
```

Figure 1: Json exemple



```
000159d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000159e0: 0000 0008 0000 0000 0080 0000 0000 0000  ................
000159f0: 0000 0000 0000 0000 0100 0000 0000 0000  ................
00015a00: 0000 0000 0000 0000 2100 0000 0000 0000  ........!.......
00015a10: 8d08 ff65 b3bf cd8b 91ca 995a d5b7 64af  ...e.......Z..d.
00015a20: 0000 0000 0000 0000 2100 0000 0000 0000  ........!.......
00015a30: 756d 6163 2d36 342d 6574 6d40 6f70 656e  umac-64-etm@open
00015a40: 7373 682e 636f 6d00 5100 0000 0000 0000  ssh.com.Q.......
00015a50: b0d4 36d2 a655 0000 b0d4 36d2 a655 0000  ..6..U....6..U..
```

Figure 2: Xxd exemple

The dataset is structured into two primary directories: `training` and `validation`. Each of these directories is further segmented into subdirectories reflecting the specific scenario, such as OpenSSH, port-forwarding, or secure copy (SCP).

Subdirectories under OpenSSH or SCP are categorized based on the software version responsible for the memory dump. These directories are further organized by the software version that generated the memory dump. The heaps are then classified based on their key lengths, with each key length possessing its dedicated directory beneath the version directory. These version-specific directories are further divided based on the different key lengths present in a heap.

Accompanying every raw memory dump is a JSON file, distinguished by the same alphanumeric sequence, barring the "-heap" suffix. This JSON file encapsulates various encryption keys and additional metadata, such as the process ID and the offset of the heap. Consequently, the dataset's utility is not confined to extracting session keys but also extends to identifying crucial data structures harboring sensitive information. The dataset, along with the associated code and tools, is open-sourced. The dataset is accessible via a Zenodo repository[1]. The code can be found in a public GitHub repository[2]. This data is the same as the data used in the paper titled *SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump* [3].

4.1.4 Definitions : Structures, Pointers, and the role of malloc headers

Through the use of the regular expressions (REGEX) `"[0-9a-f]{12}0{4}"`, we identified potential pointers within the dump. This heuristic approach acts as a sieve, filtering the extensive data to spotlight possible pointer candidates. Nonetheless, it's crucial to understand that while many pointers might be correctly pinpointed, some detected sequences may not be authentic pointers.

One notable characteristic of the heap dump is the *malloc header* found at the start of allocated structures. This header, often the initial non-null bytes in a series, signifies the size of the following structure. By sequentially reading the heap dump and identifying these headers, it becomes feasible to determine the dimensions and limits of every allocated structure, thereby methodically dividing the heap dump into distinct structures.

[1] `https://zenodo.org/record/6537904`
[2] Link to the GitHub repository

4.2 Traditional Statistical Embedding

4.2.1 Shannon entropy and its application in byte sequence analysis

4.2.2 Byte Frequency Distribution (BFD)

4.3 Deep Learning Models for Raw Byte Embedding

4.3.1 Introduction to the role of deep learning in byte sequence analysis

4.3.2 RNNs : Understanding sequence data

4.3.3 CNNs : Pattern detection in raw bytes

4.3.4 Autoencoders

4.3.5 Transformers

4.4 Graph Embedding Methods

4.4.1 Introduction to graph embedding

4.4.2 Popular embedding techniques

Node2Vec, GraphSAGE, and others

4.4.3 Applications and significance in OpenSSH heap dump analysis

4.5 Conclusion and Transition to the Next Section

5 Methods

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

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

Acronymes

BFD	Byte Frequency Distribution. 2
CNN	Convolutional Neural Networks. 2
GRU	Gated Recurrent Units. 2
LTSM	Long Short-Term Memory. 2
REGEX	regular expressions. 5
RNN	Recurrent Neural Networks. 2
SCP	secure copy. 4
SSH	Secure Shell. 1
VMI	Virtual Machine Introspection. 1

References

[1] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. Apr. 19, 2018. arXiv: 1803.01271[cs]. URL: http://arxiv.org/abs/1803.01271 (visited on 08/23/2023).

[2] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. Dec. 11, 2014. arXiv: 1412.3555[cs]. URL: http://arxiv.org/abs/1412.3555 (visited on 08/23/2023).

[3] Christofer Fellicious et al. *SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump*. Sept. 13, 2022. arXiv: 2209.05243[cs]. URL: http://arxiv.org/abs/2209.05243 (visited on 08/17/2023).

[4] Luke Hiester. „File Fragment Classification Using Neural Networks with Lossless Representations“. In: *East Tennessee State University* (May 2018). (Visited on 08/21/2023).

[5] Sepp Hochreiter and Jürgen Schmidhuber. „Long short-term memory“. In: *Neural computation* 9.8 (1997). Publisher: MIT Press, pp. 1735–1780. (Visited on 08/23/2023).

[6] Siwei Lai et al. „Recurrent Convolutional Neural Networks for Text Classification“. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (Feb. 19, 2015). ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v29i1.9513. URL: https://ojs.aaai.org/index.php/AAAI/article/view/9513 (visited on 08/23/2023).

[7] Stewart Sentanoe and Hans P. Reiser. „SSHkex: Leveraging virtual machine introspection for extracting SSH keys and decrypting SSH network traffic“. In: *Forensic Science International: Digital Investigation* 40 (Apr. 2022), p. 301337. ISSN: 26662817. DOI: 10.1016/j.fsidi.2022.301337. URL: https://linkinghub.elsevier.com/retrieve/pii/S2666281722000063 (visited on 08/17/2023).

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. Dec. 14, 2014. arXiv: 1409.3215[cs]. URL: http://arxiv.org/abs/1409.3215 (visited on 08/23/2023).

[9] Ashish Vaswani et al. „Attention Is All You Need“. In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 5998–6008. (Visited on 08/23/2023).

Additional bibliography

[10] Vivek Gite. *How To Reuse SSH Connection To Speed Up Remote Login Process Using Multiplexing*. nixCraft. Aug. 20, 2008. URL: https://www.cyberciti.biz/faq/linux-unix-reuse-openssh-connection/ (visited on 10/21/2022).

[11] Weijie Huang and Jun Wang. *Character-level Convolutional Network for Text Classification Applied to Chinese Corpus*. Nov. 15, 2016. arXiv: 1611.04358[cs]. URL: http://arxiv.org/abs/1611.04358 (visited on 08/17/2023).

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, August 24, 2023

Lahoche, Clément Claude Martial