Linux Memory Forensics: Dissecting the User Space Process Heap

Frank Block (a,c) , Andreas Dewald (b,c)

a: ERNW GmbH, Heidelberg, Germany

b: ERNW Research GmbH, Heidelberg, Germany

c: Friedrich-Alexander University Erlangen-Nuremberg (FAU), Germany

Agenda

Introduction

Motivation

Heap Fundamentals

Evaluation

Demo

Conclusion

ERNW
providing security.

Introduction

- Most of the previous work in the memory forensics area focused on information residing in the kernel space.
- The user space is however also a rich source of artifacts/data, valuable for the investigator, especially the heap.
- Might contain for example passwords or (en/de)crypted data from
  - legitimate software (Mail Client, Password Manager, …)
  - malware (passwords used for encrypted communication, decrypted binary data, …)

ERNW
providing security.

Introduction

- For Windows, there is a plugin to analyze the heap in a more detailed way, thanks to the research by Michael Cohen.
  - http://www.rekall-forensic.com/docs/References/Papers/p1138-cohen.pdf

- But there was no public available plugin for any heap implementation for Linux (at least as far as we know).

ERNW
providing security.

Introduction

- Existing plugins that investigate the Linux process memory space for artifacts/data, normally search the whole memory dump or the heap/stack as a big blob for a given pattern.

- Depending on the data of interest, there are however not always specific patterns marking data of interest, or these patterns are yet not known to the investigator.

Agenda

Introduction
Motivation
Heap Fundamentals
Evaluation
Demo
Conclusion

ERNW
providing security.

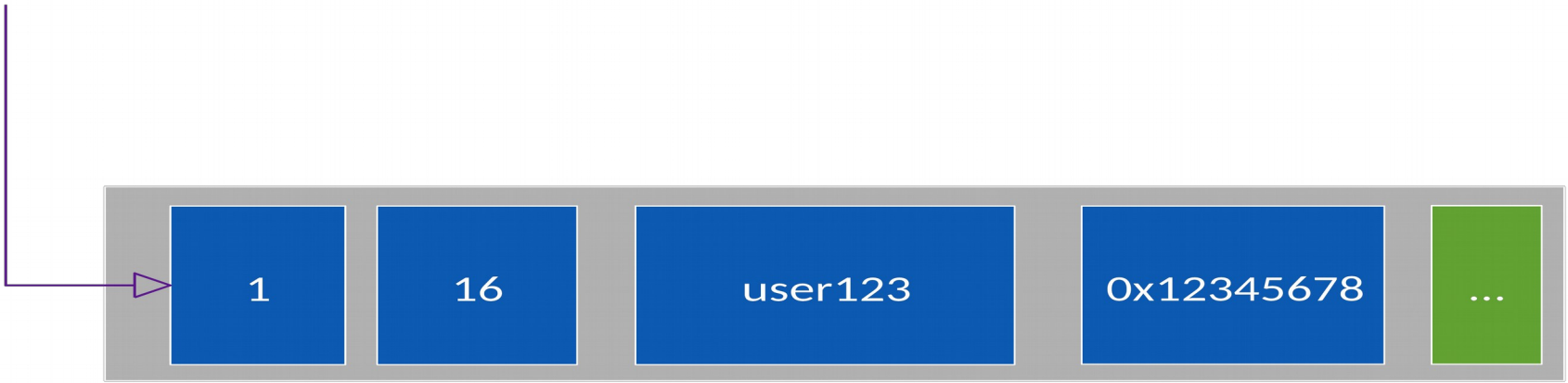
Challenge: Pattern-less data

- Data of interest, which is not identifiable by a special pattern and also not easily distinguishable from data surrounding it.
  - An IP address stored as an integer
  - A password, not consisting of printable characters
  - …

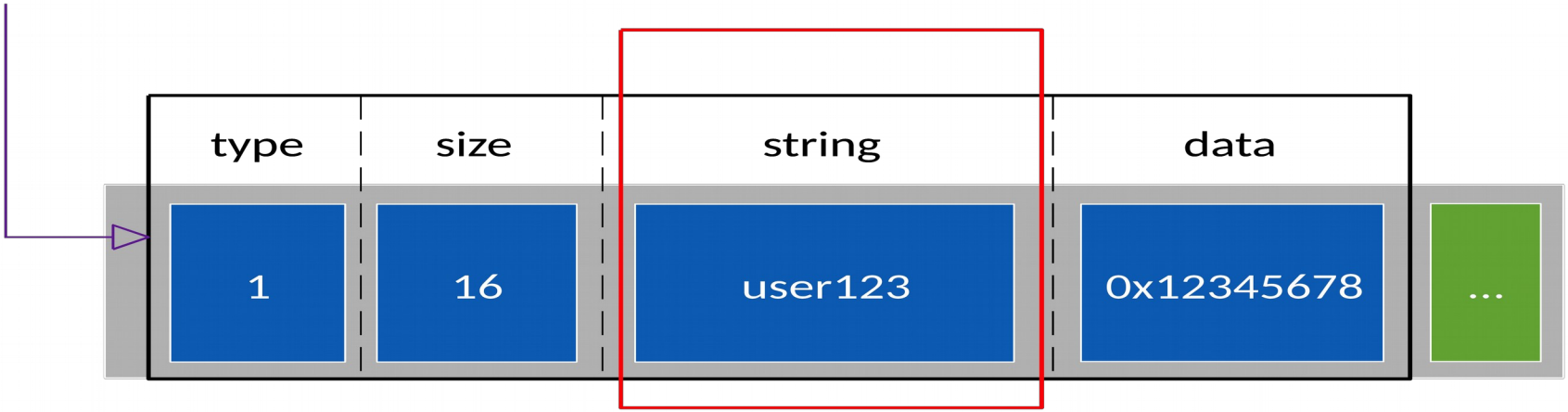
`9dcfa4b5f5a9de67`**`2db4728b05de2197`**`05deca9575e4f9471a51e5a8627`

ERNW
providing security.

Challenge: A pointer to a Struct



Challenge: A pointer to a Struct



Goals

- Understand and document **Glibc's** Heap Implementation for forensic purposes.
- Implementation of Rekall plugins that assist an investigator in the analysis process.

ERNW
providing security.

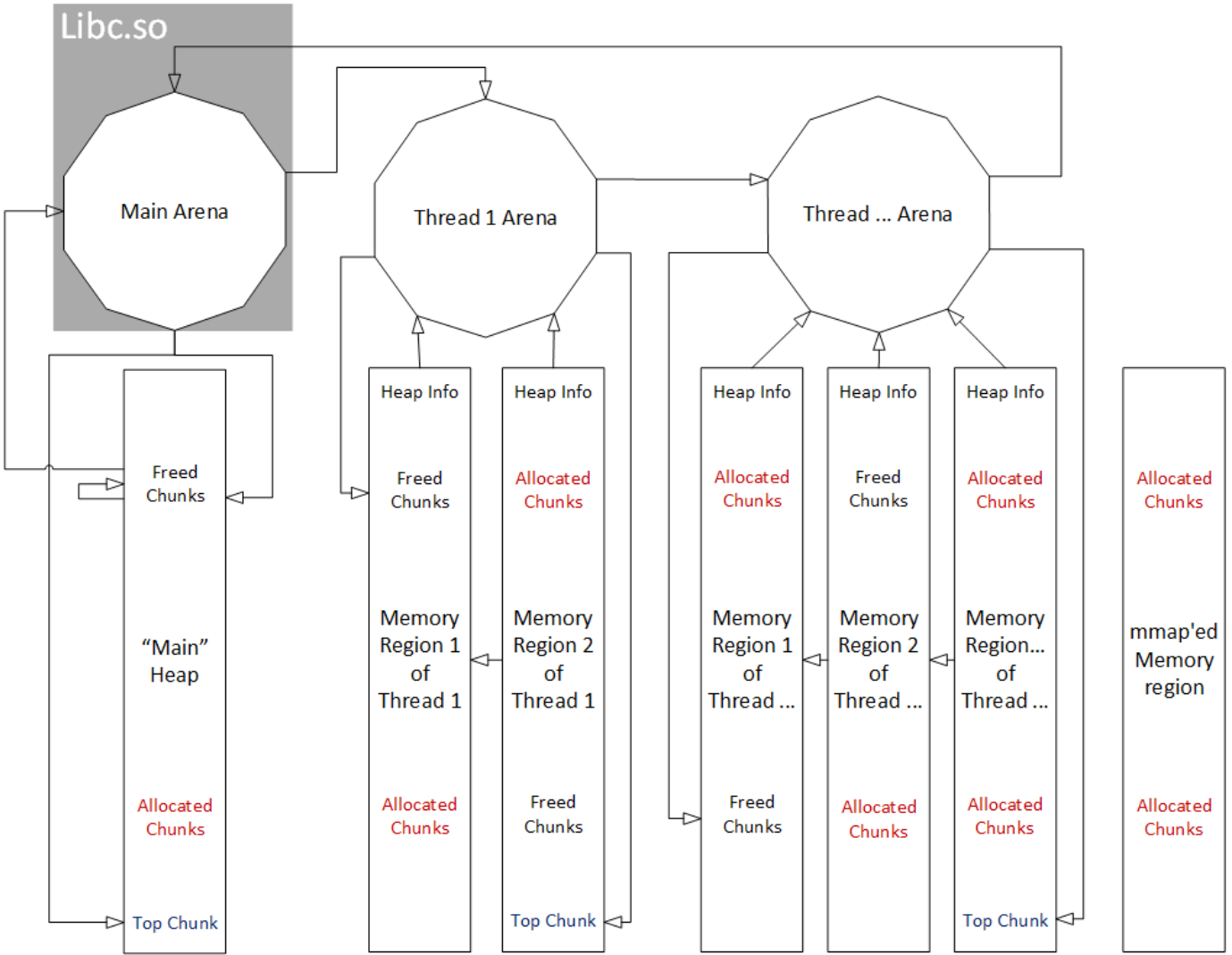
Agenda

Introduction
Motivation
Heap Fundamentals
Evaluation
Demo
Conclusion

ERNW
providing security.

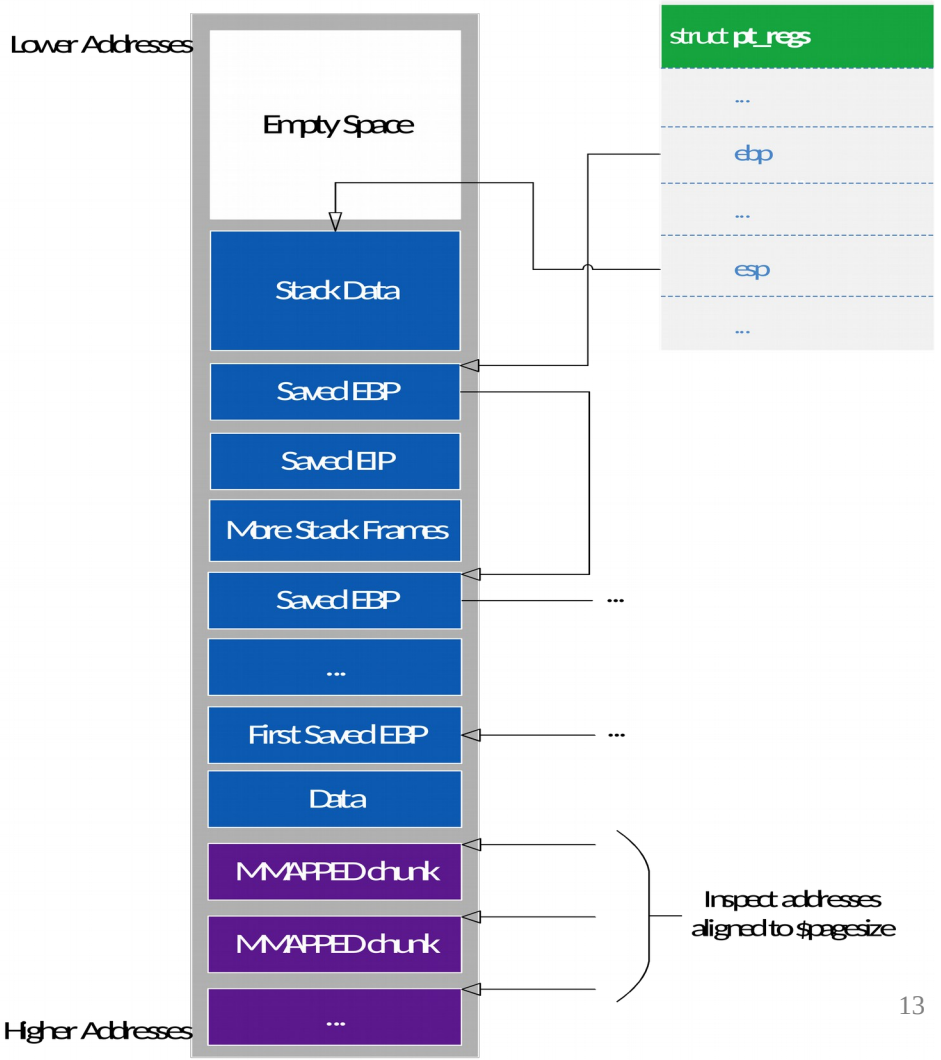
Heap Fundamentals

Heap Overview



ERNW
providing security.

EBP unrolling



ERNW
providing security.

MMAPPED Chunks

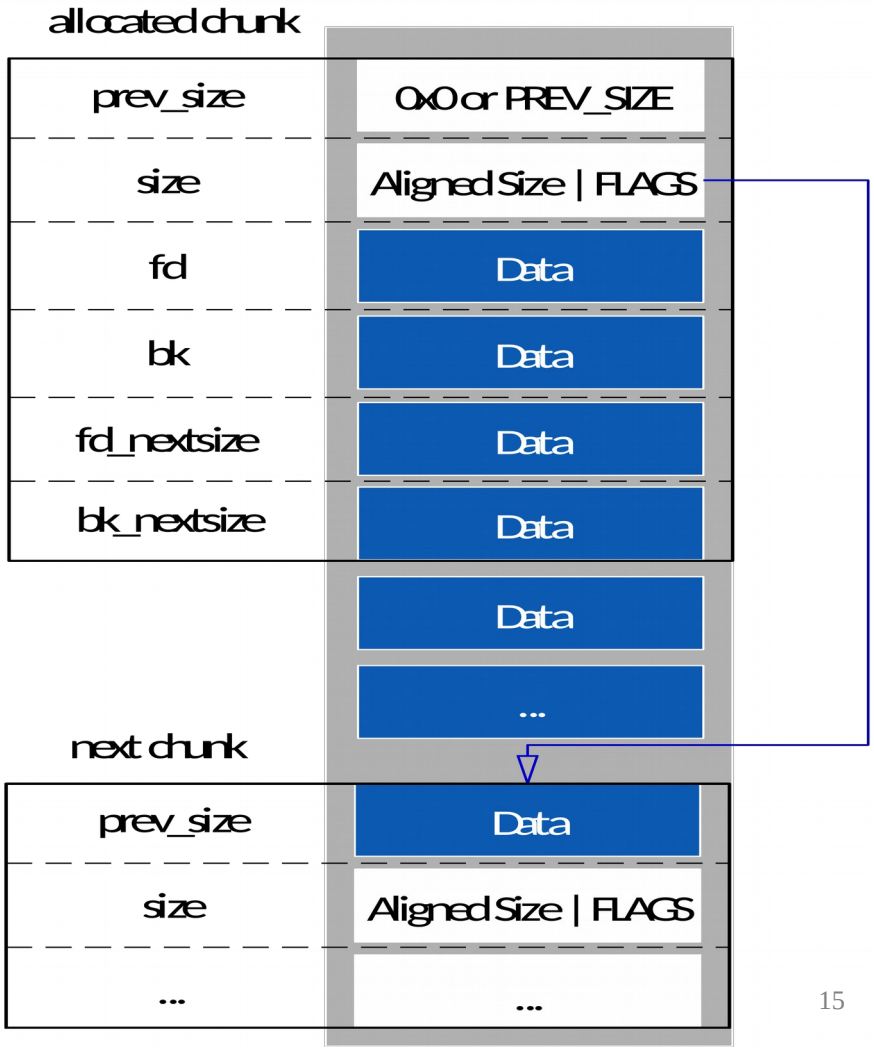
- The identified MMAPPED chunks can be verified, when the offset to the *malloc_par* struct is given (must be gathered from the Glibc library of the target system).

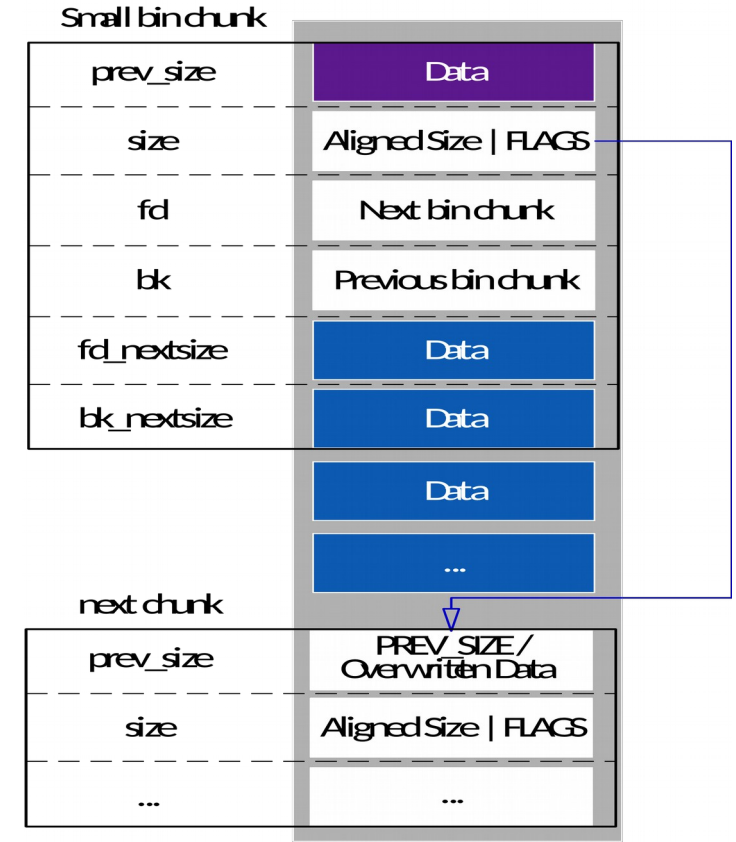
- This struct contains the number and size of all MMAPPED chunks.

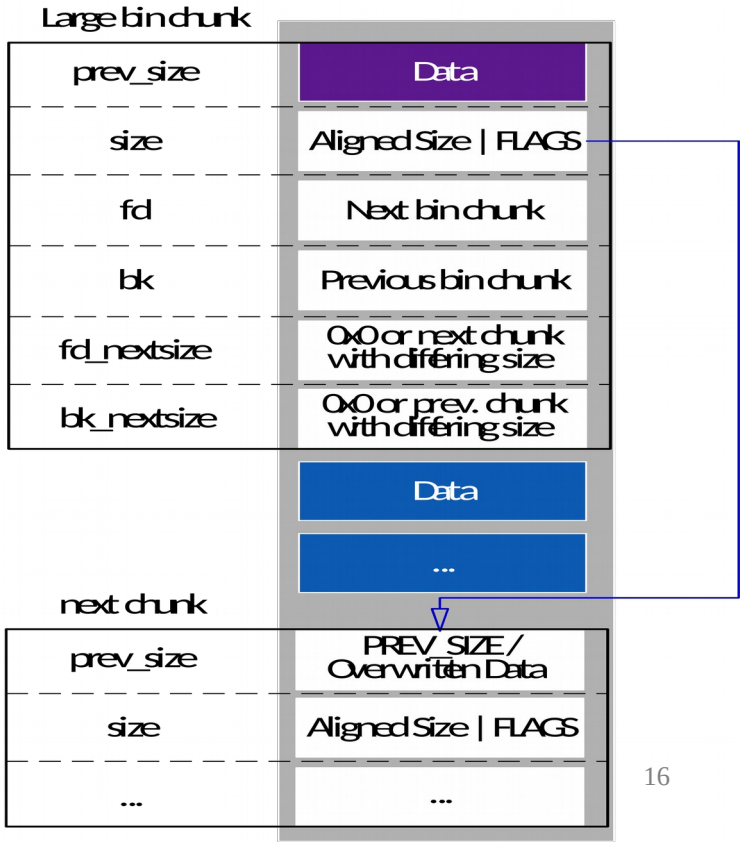
ERNW
providing security.

Allocated chunk

In memory




Small bin chunk
prev_size
Data
size
Aligned Size | FLAGS
fd
Next bin chunk
bk
Previous bin chunk
fd_nextsize
Data
bk_nextsize
Data
Data
...
next chunk
prev_size
PREV_SIZE / Overwritten Data
size
Aligned Size | FLAGS
...
...


Large bin chunk
prev_size
Data
size
Aligned Size | FLAGS
fd
Next bin chunk
bk
Previous bin chunk
fd_nextsize
0x0 or next chunk with differing size
bk_nextsize
0x0 or prev. chunk with differing size
Data
...
next chunk
prev_size
PREV_SIZE / Overwritten Data
size
Aligned Size | FLAGS
...
...

ERNW
providing security.

Another Challenge: Extracting Strings from the Heap

"**He said:** I will kill him"

- might become

non_printable_chars " I will kill him"

"**Please, do not** kill him**"**

- might become

non_printable_chars "**I'll** kill him"

Agenda

Introduction
Motivation
Heap Fundamentals
Evaluation
Demo
Conclusion

ERNW
providing security.

Evaluation

- Test Environment: Arch Linux x86/x64, Kernel Version 4.4.5-ARCH with Glibc Versions 2.20 – 2.24
- Self written programs, which use Glibc‘s *mallinfo* function.
- Analyzing all processes running in the test environments.
- Reading a lot of source code. Mainly from:
  - malloc/malloc.c
  - malloc/malloc.h
  - malloc/arena.c

ERNW
providing security.

Evaluation

- Background checks are performed during the execution of each plugin.
  - For each chunk: Size/address alignment, allocation status, flags, …
  - Some special tests for MMAPPED chunks, and a comparison to the information from the *malloc_par* struct.
  - Does following all chunks in a memory region lead to the expected end?
  - Comparison of the size information from all arenas with all objects from the heap.

Demo Time?

ERNW
providing security.

Agenda

Introduction
Motivation
Heap Fundamentals
Evaluation
Demo
Conclusion

ERNW
providing security.

Conclusion

- The plugins ease the analysis process and serve more reliable information.
- They enable to gather all parts belonging to the heap (e.g. also hidden MMAPPED chunks).

- Limitations
  - Only Glibc support (officially, versions 2.20 – 2.25).
  - MMAPPED chunks without debug symbols can’t reliably be identified.
  - Swapped pages

- Future Work
  - Add linked list detector to the heaprefs plugin.
  - Keep up with new Glibc versions.

ERNW
providing security.

Links

- The paper:
  - https://authors.elsevier.com/sd/article/S1742287617301895
- The technical report (contains more heap details):
  - https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/8340
- The plugins are merged in the Rekall project:
  - https://github.com/google/rekall
- Blogpost, containing a brief manual:
  - https://insinuator.net/2017/07/release-of-glibc-heap-analysis-plugins-for-rekall/

Thank you for your Attention!

Questions? Feedback? Suggestions? Criticism?