Hacking TWP

with

radare

Francesco Tamagni

mrmacete / @bezjaje

About Thimbleweed Park

- retro-style adventure game
- released in early 2017
- by Ron Gilbert, Gary Winnick, David Fox and other folks from Monkey Island team
- go get it at https://thimbleweedpark.com/
- i bought it twice

TWP data files

- ThimbleweedPark.ggpack*
- in game's data folder
- obfuscated
- contain all resources and some logic

TWP data files

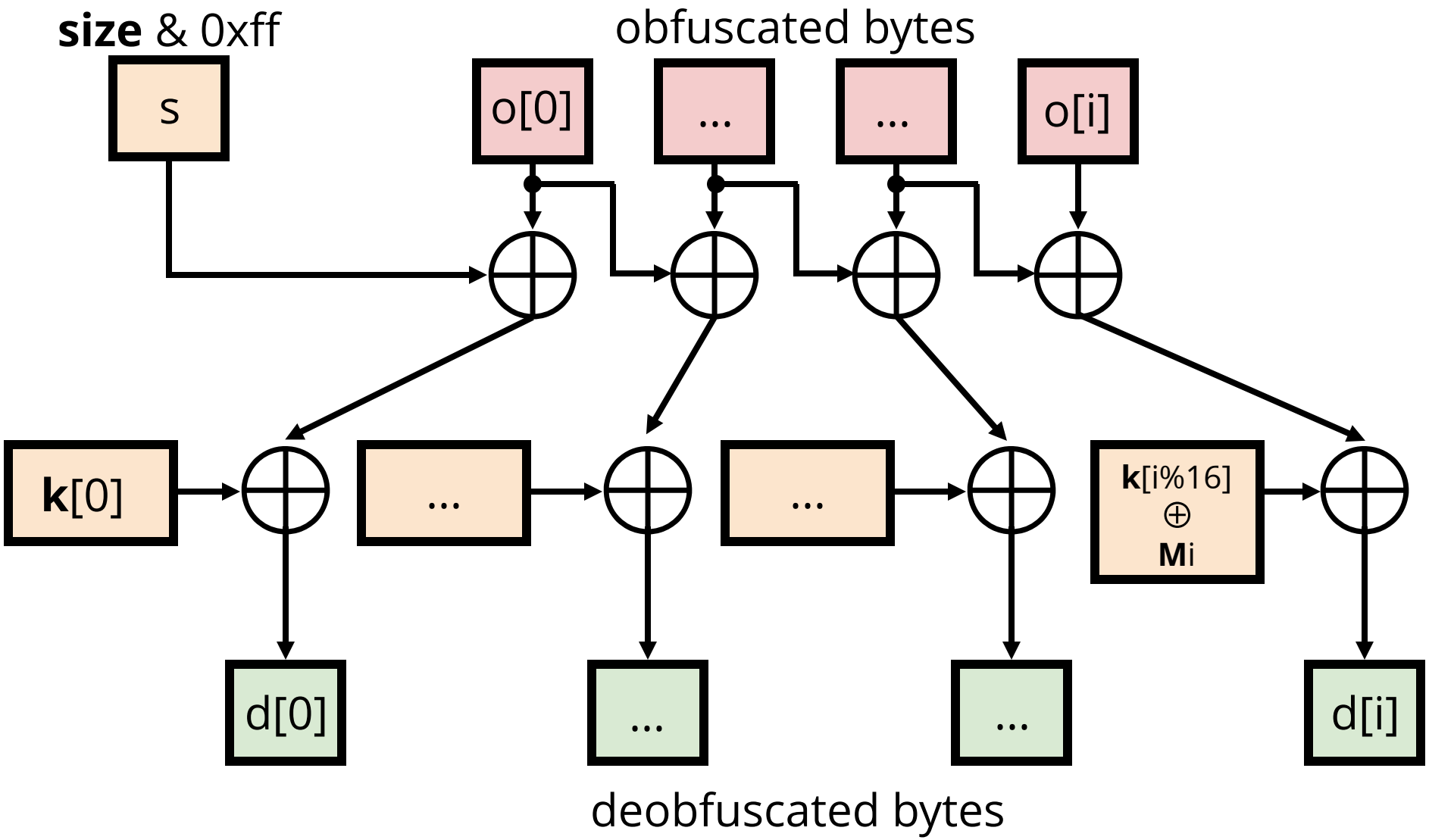
DEMO: how TWP loads its data files

- trace fopen() using r2frida
- analyze sorroundings on the executable with r2 to find deobfuscation logic

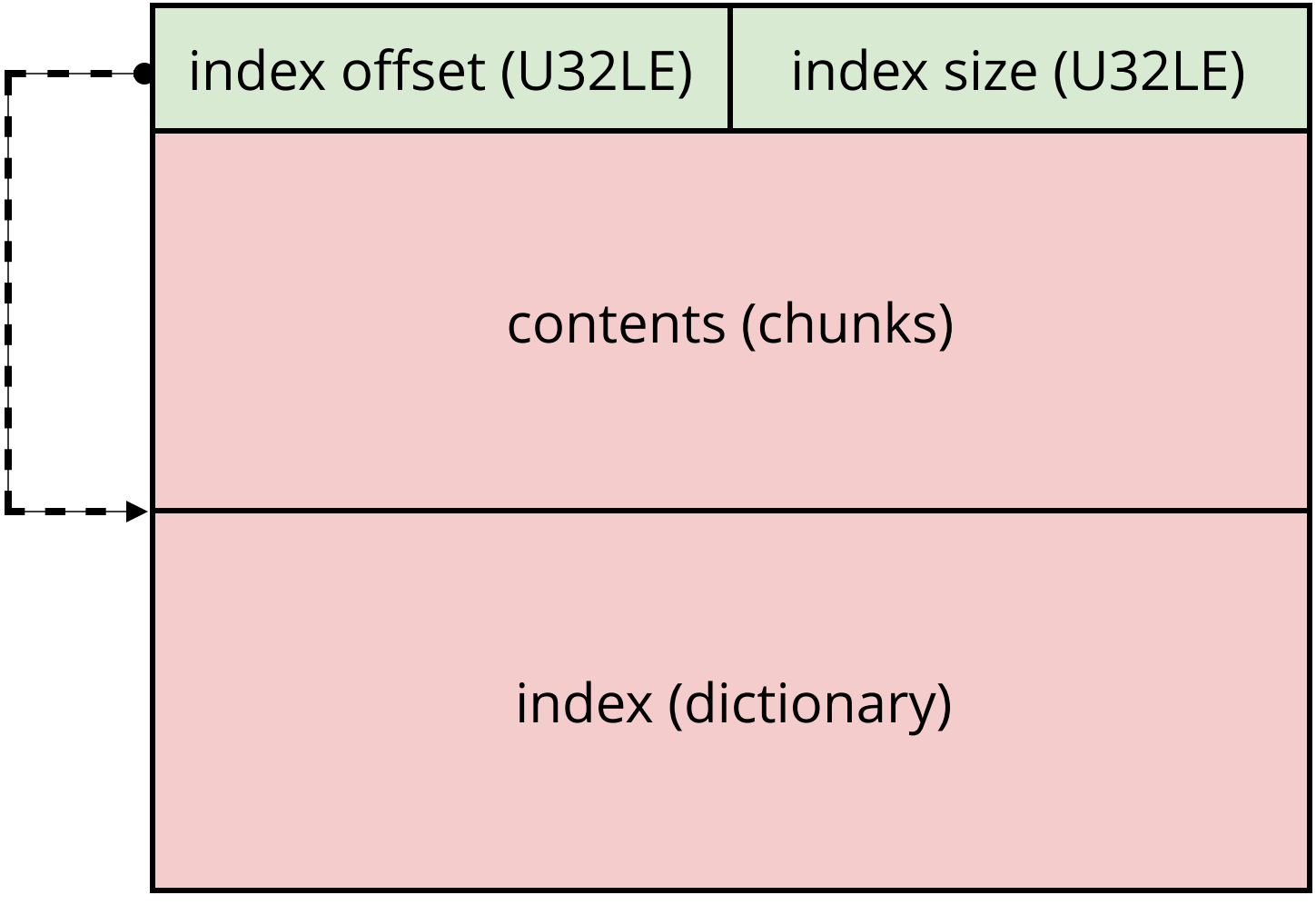
Deobfuscation

- each chunk (de)obfuscated independently
- parameters for each chunk
  - chunk **size**
  - **k**, 16-bytes secret (constant, repeated)
  - **M**, constant 1-byte multiplier
- the constants can change across versions / platforms
- early PoC on deobfuscating these files on github but incomplete: https://github.com/mstr-/twp-ggdump

Deobfuscation (chunk)



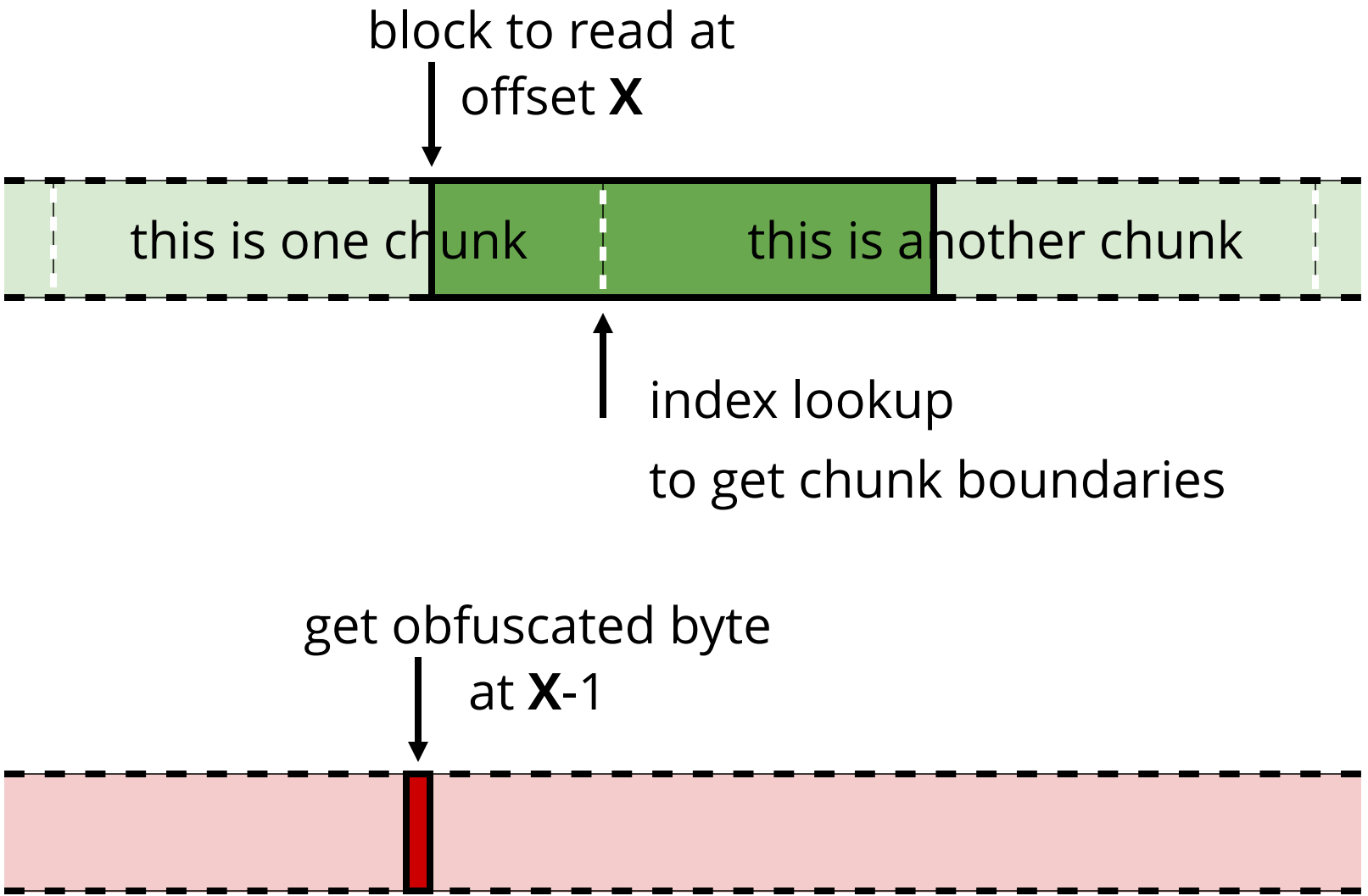
ggpack file format



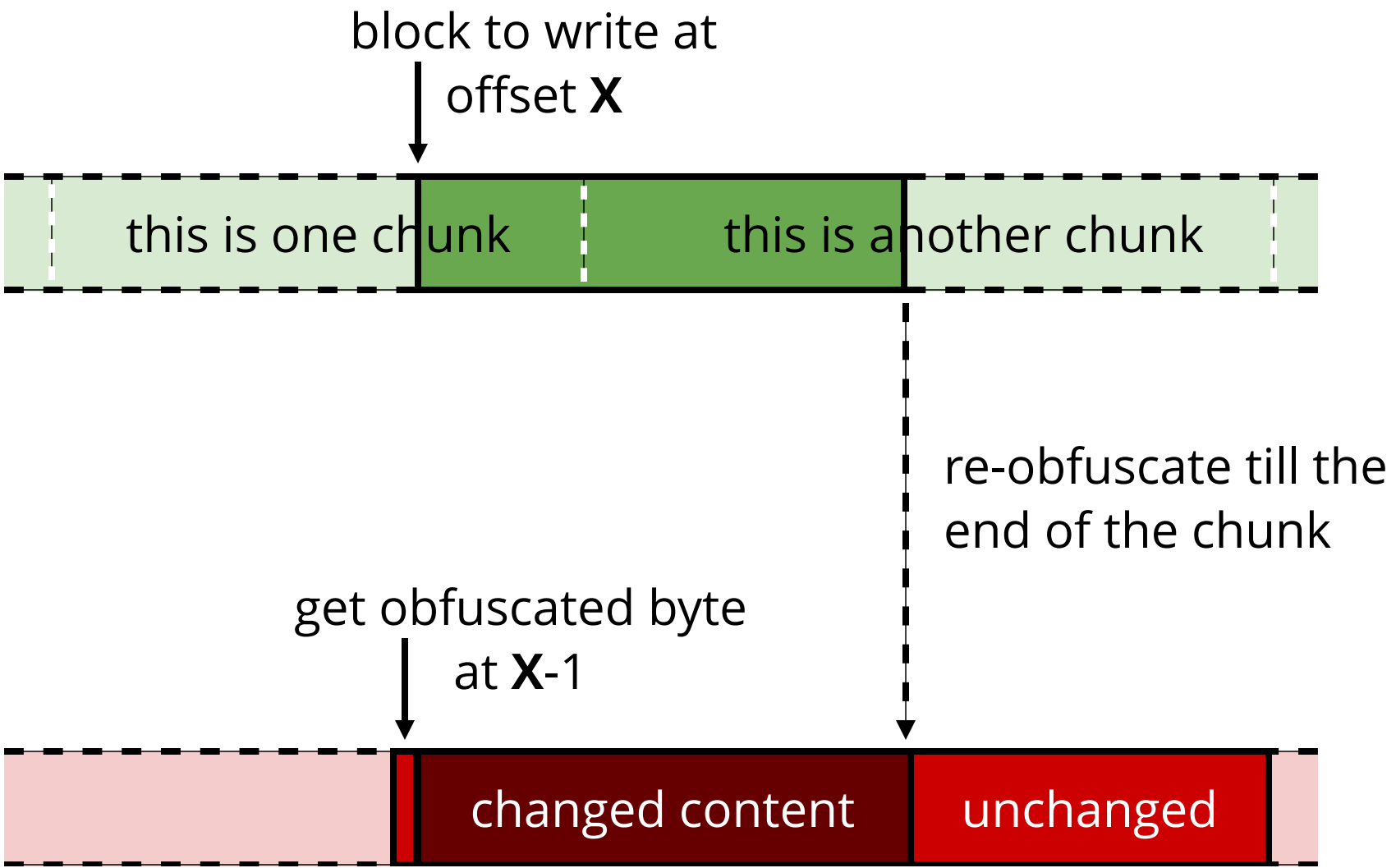
I/O plugin

- deofuscate / obfuscate data files on the fly
- read / write / insert / delete bytes
- explore game data types
- source code: https://github.com/mrmacete/r2-ggpack

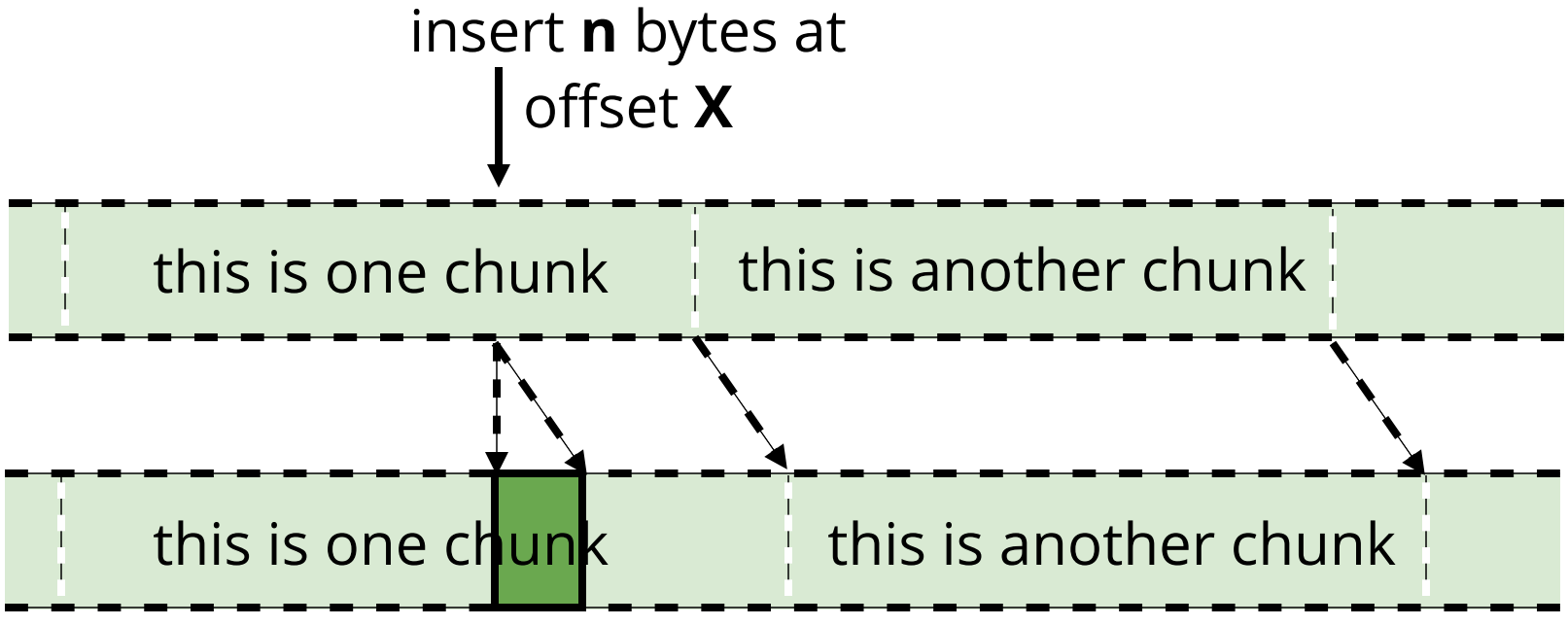
Linear random access (read)



Linear random access (write)

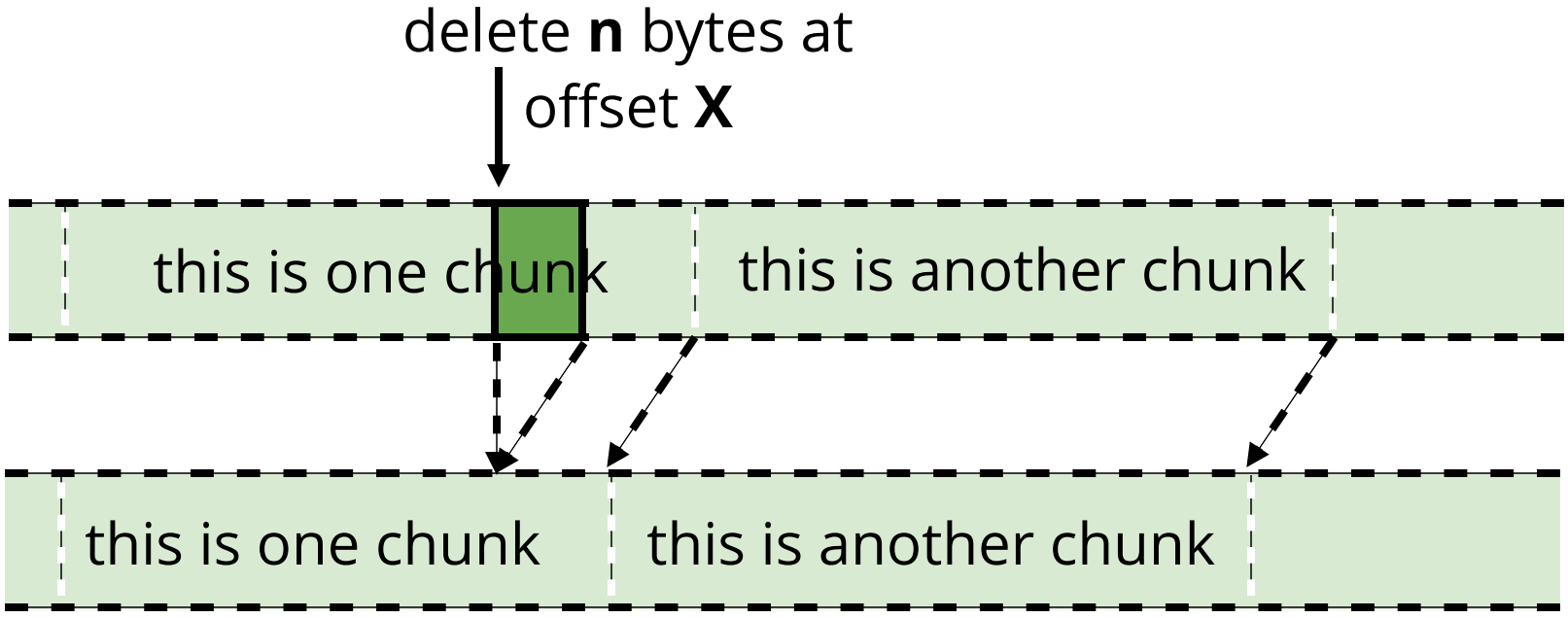


Insert bytes



- the first chunk is resized, following ones moved
- update offsets and sizes and re-obfuscate
- wait for r2 core to perform the actual shift (hack)
- rebuild the index in the file

Delete bytes

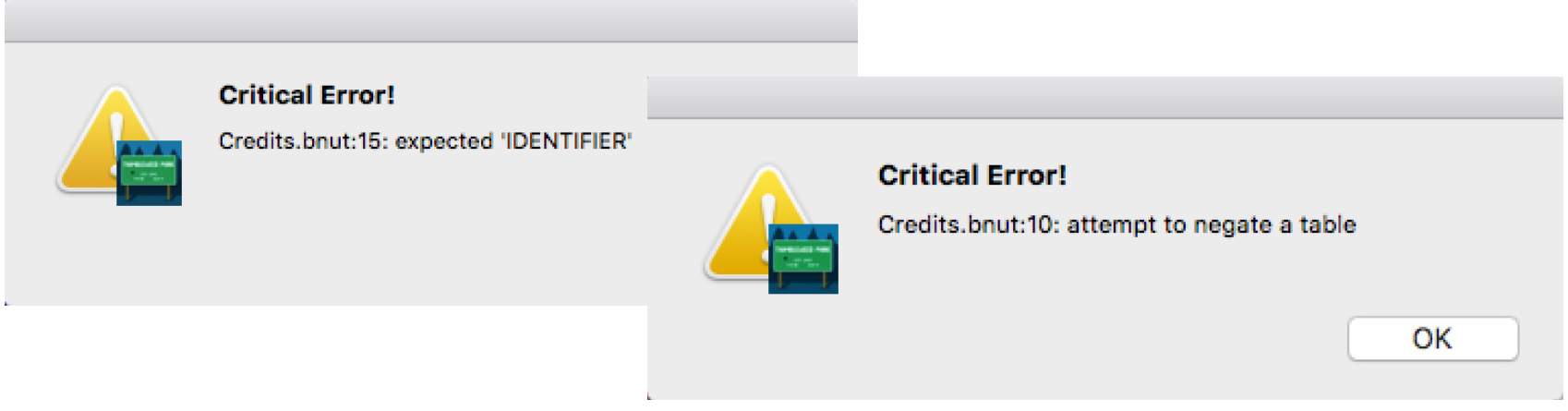


- the first chunk is resized, following ones moved (start from the end)
- update offsets and sizes and re-obfuscate
- r2 core already performed the shift
- rebuild the index in the file right away

Data types

Type	**Description**
wimpy	binary dictionary, use **=!pDj** to get a JSON representation
json	regular JSON text
png	regular png
byack, bnut	game logic
txt, tsv	plain text, mainly translations
wav, ogg	sfx and music
fnt	bitmap font metadata
lip	lipsync metadata (rhubarb format)

Second pass obfuscation



bnut files are obfuscated Squirrel source code

```
__<-"Terrible Toybox (c) 2017. Any modification or redistributio
n of the Thimbleweed Park game, code, or contents, without the p
rior written consent of Terrible Toybox is strictly prohibited."
```

Second pass obfuscation

Squirrel compiler reads one byte at a time from the input source in the Lexer:

```
void SQLexer::Next()
{
    SQInteger t = _readf(_up);
    if(t > MAX_CHAR) Error(_SC("Invalid character"));
    if(t != 0) {
        _currdata = (LexChar)t;
        return;
    }
    _currdata = SQUIRREL_EOB;
    _reached_eof = SQTrue;
}
```

So this is the logical place to plug obfuscation in, but it could also be in the external _readf() function

Finding the lexer

Demo

Lexer source

https://github.com/albertodemichelis/squirrel/blob/master/squirrel/sqlexer.cpp#L87

no symbols! strings are your friends

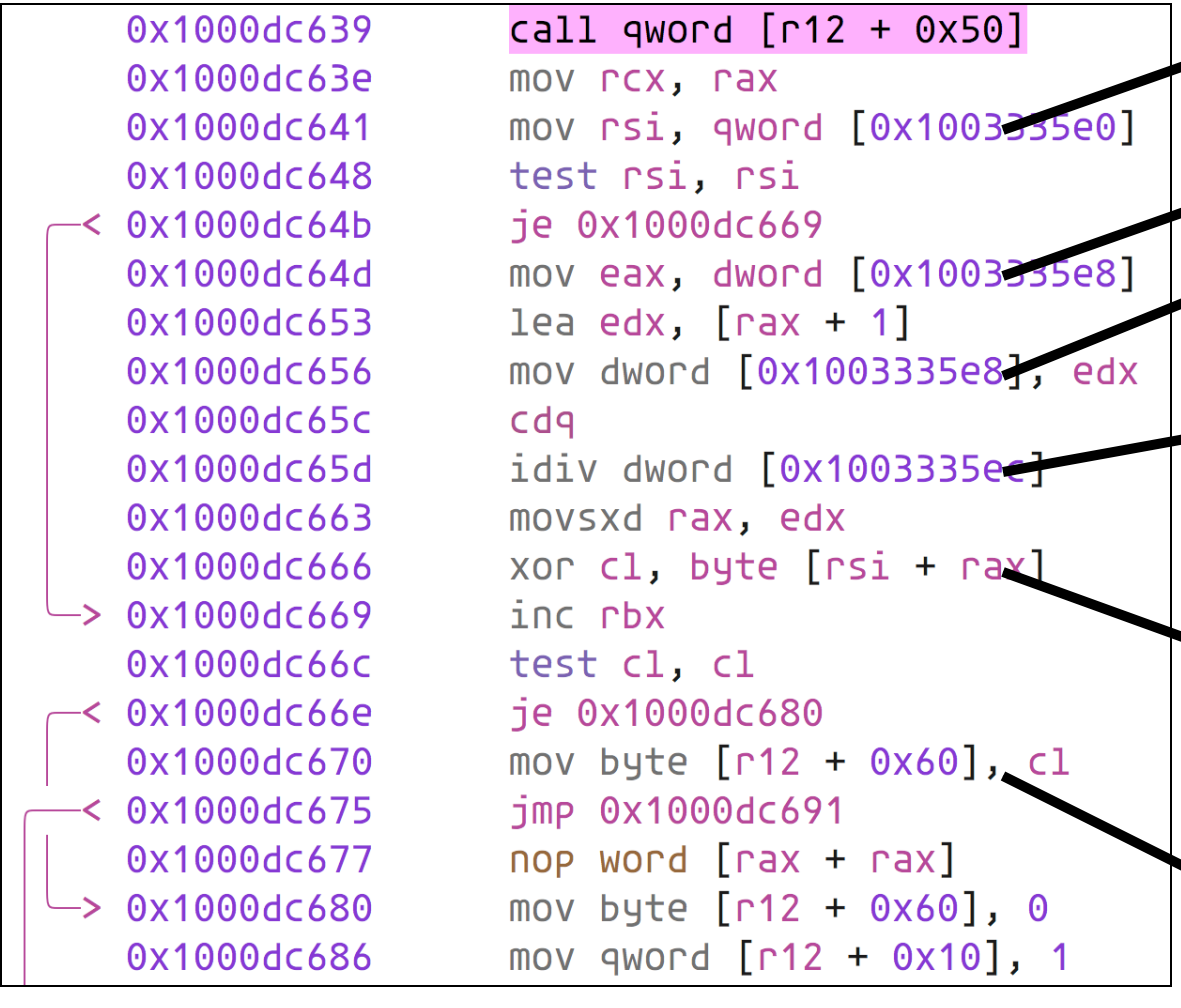
Finding the lexer

```
void SQLexer::Next()
{
    SQInteger t = _readf(_up);
    if(t > MAX_CHAR) Error(_SC("Invalid
    if(t != 0) {
        _currdata = (LexChar)t;
        return;
    }
    _currdata = SQUIRREL_EOB;
    _reached_eof = SQTrue;
}
```

```
0x1000dc639    call qword [r12 + 0x50]
0x1000dc63e    mov rcx, rax
0x1000dc641    mov rsi, qword [0x1003335e0]
0x1000dc648    test rsi, rsi
0x1000dc64b    je 0x1000dc669
0x1000dc64d    mov eax, dword [0x1003335e8]
0x1000dc653    lea edx, [rax + 1]
0x1000dc656    mov dword [0x1003335e8], edx
0x1000dc65c    cdq
0x1000dc65d    idiv dword [0x1003335ec]
0x1000dc663    movsxd rax, edx
0x1000dc666    xor cl, byte [rsi + rax]
0x1000dc669    inc rbx
0x1000dc66c    test cl, cl
0x1000dc66e    je 0x1000dc680
0x1000dc670    mov byte [r12 + 0x60], cl
0x1000dc675    jmp 0x1000dc691
0x1000dc677    nop word [rax + rax]
0x1000dc680    mov byte [r12 + 0x60], 0
0x1000dc686    mov qword [r12 + 0x10], 1
```

- it's inlined
- they modified the Next() function itself instead of passing a custom _readf()

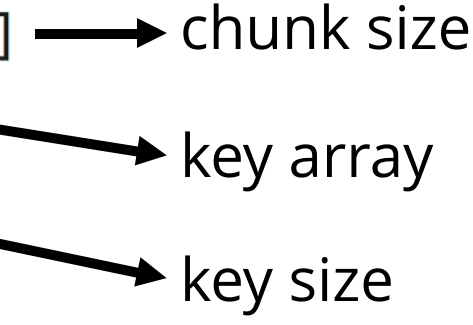
Finding the lexer



- get the base of a constant array
- get the cursor
- increment it and update
- get the remainder of the division by the size
- index the array with that and xor with input
- that's the current onubfuscated byte

Finding the lexer

```
0x10006a334      movzx edx, byte [r14 + 0x20]
0x10006a339      lea rdi, [0x100324040]
0x10006a340      mov esi, 0x1000
0x10006a345      call initLexObfuscation
```




```
"td initLexObfuscation(char * key, int key_size, int chunk_size)"
```

```
(fcn) initLexObfuscation 25
  initLexObfuscation (char *key, int32_t key_size, int32_t chunk_size);
          ; arg char *key @ rdi
          ; arg int32_t key_size @ rsi
          ; arg int32_t chunk_size @ rdx
          ; CALL XREFS from sym.func.10006a2b0 (+0x95, +0x1c4, +0x256)
          0x1000da6e0      push rbp
          0x1000da6e1      mov rbp, rsp
          0x1000da6e4      mov qword [0x1003335e0], rdi    ; [0x1003335e0:8]=0 ; key
          0x1000da6eb      mov dword [0x1003335ec], esi    ; [0x1003335ec:4]=0 ; key_size
          0x1000da6f1      mov dword [0x1003335e8], edx    ; [0x1003335e8:4]=0 ; chunk_size
          0x1000da6f7      pop rbp
          0x1000da6f8      ret
```

Demo

- export all resources
- modify one of them
- replace it back
- enjoy!

Questions?

and thanks!