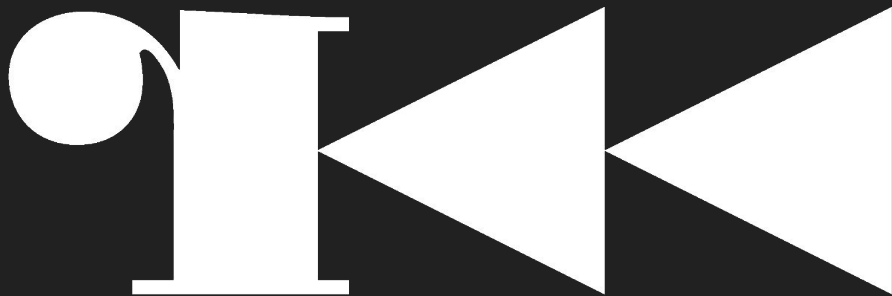


# Digging into radare2 for fun and profit

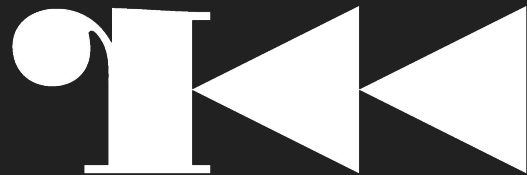
AvTokyo2017 // by  
pancake

*INTRO & Translation: unixfreaxjp*

Radare2とは?



# Radareとは?



- 12年間開発されているオープンソースプロジェクト(無料)
- リバースエンジニアリングのフレームワークとツールセット
- 最初段階では私が開発したソフトウェア
- コミュニティとコントリビューターのコーダーが結構増えて
- 個人でメンテナンスしたプロジェクトからプロジェクトを分けて、それぞれの開発プロジェクト責任者がたちあがりました
  
- 現状開発のペースは6週間1回でマイナーバージョンアップをリリース
- メジャーバージョンはr2conの後にリリース
- r2con 自体はスペイン、バルセロナから始まり、2017年の自体で大体230人の参加皆さんが着ました
- r2conのコンファレンスにあるプレゼンテーションビデオを必ず youtubeにアップロードします

# Who Am I?

- フリーソフトの開発者
  - Barcelona, Cataloniaの生まれ
  - いくつかフリーソフトウェアを開発しています
  - defcon CTF 3年間連続の参加しました
  - ペイントが大好き
  - 父親として頑張ります
- Links
  - 1) githubそして 2) bitbucket でradare コードをアップロード (user: trufae)
  - ツイッター: t <https://twitter.com/trufae>
- 仕事は NowSecure社 (Mobile Security Analyst, フォレンジックとR+D)
  - codecs 開発 (assembly) mips, arm と x86向け
  - Embeddedデバイスのファームウェア開発 (リアルタイム・デバッグ調査機能を含む )  
フォレンジックについてほとんど Windows案件が多い
  - プログラミングとCTF関係の講師



# radare2の便利機能について

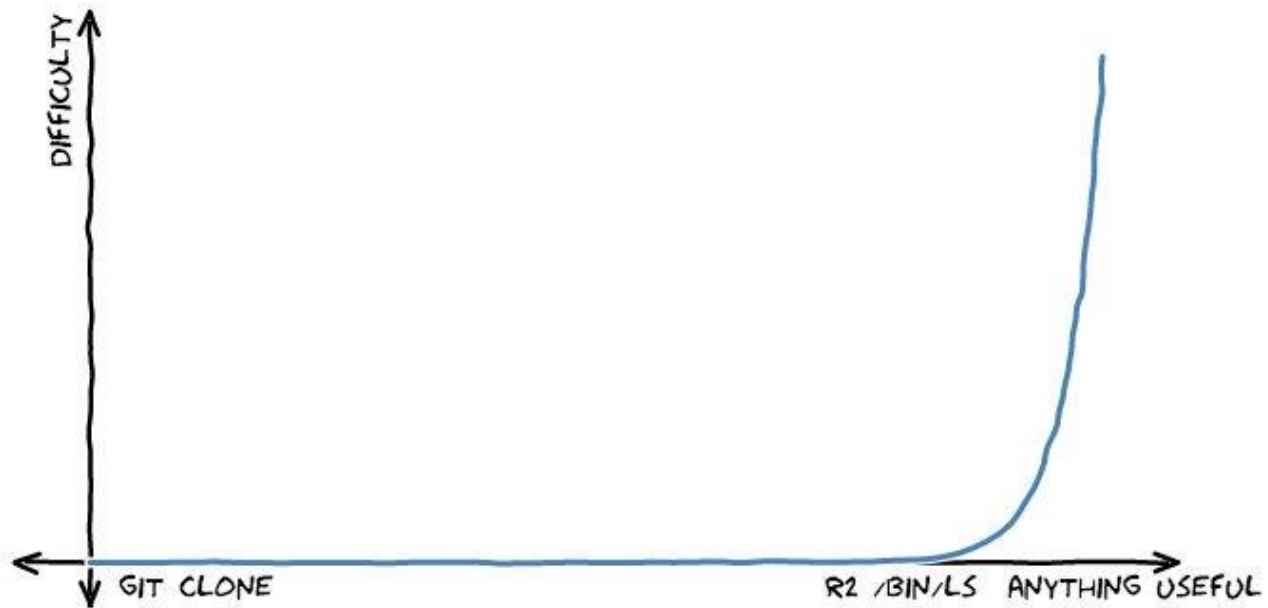
- プログラムのデバッグ調査ツール
- エンコードされたcharsetの検知と検索 (日本語, 中国語, 韓国語, cyrilic, など)
- スtringスのリファレンス調査、特徴な技術方法が持っている
- メモリダンプのカービングツール("magic"を調べるフォレンジック調査)
- 色んなファイルシステムのマウントと認識機能、partition tableの読み込み
- その他のデバッガとの連携機能(gdb, r2, frida, windbg, など)
- エミュレーター機能を使い、解読の調査ツールとして非常に便利
- 外部デコンパイラまたはグラフ作成ツールを使用する
- バイナリのdiff 機能
- ゲームやシミュレーションツーツ、例えば2048 や r2wars!

# ほぼどんな環境にも使えます！

- 様々なOSの対応:
  - Windows, Linux, Mac, QNX, Solaris, NetBSD, FreeBSD, BeOS, Android, iOS, ....
- 色んなcpuアーキテクチャーにも対応しています、例えば:
  - x86, arm, mips, sparc, ppc, z80, 6502, 8051, avr, wasm, snes, java, dalvik, hppa, ...
- ネイティブデバッガの対応
- web-assembly と asm.jsのコンパイルする事もできます。
- ローカルとリモート(socket)の使い方
- Foll

(demo rasm2 -L rabin2 -L r2 -L)

## R2 LEARNING CURVE



## 学習曲線

- 最初段階はステップバイステップでの学ぶ事ですが、長期的には楽しい
  - 10個の実行コマンドが分かれば使えるようになる
  - 直交性がコマンドを可能ですし、拡張し改善することができます
  - 初めてのの方は2週間ぐらいの学ぶ事が可能
- 
- 実践から学ぶ
  - 関心と献身が必要
  - 他の似たようなツールと比べたらやり方がいくつか違う
  - オープンソースなので、rwxをどうぞ！



# It's Documented®

- ソースコードはCでドキュメンテーション課されている
- オンラインでマニュアルブックがあり(古いバージョン)
- ブログの記事(フォロー: @radareorg)
- 使い方としてはYouTube とVimeoビデオが多い
  - r2con 2016, 2017のビデオ
- コマンドラインのヘルプマニュアル("?コマンド)
- UNIX系のmanページ
- IRC と Telegram チャンネルのQ & Aコミュニティ(800人)

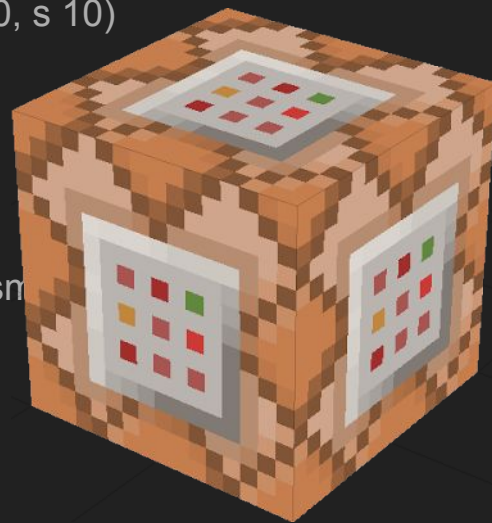
<https://twitter.com/radareorg>

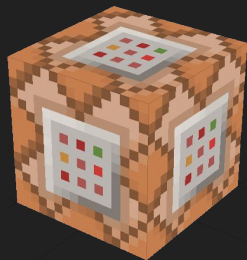
<https://t.me/radare>



# 基本の機能とコマンド

- 検索
  - Relative / Absoluteのアドレスと部分的なアドレス (s+0x10, s..10, s 10)
  - 履歴/History (!, s!)
  - ブロックサイズ / blocksize (b, @!)
- ダンプ/プリント
  - Hexdump (px, pxr, pxa, prc, pxA, ...)
  - アセンブリの種類に夜、ディスアセンブリ (pd, pD) @a:mips, e asm
  - デコード構造 (pf)
  - Checksums, entropy, statistics (p=, ph)
- バイナリの書き込み
  - アセンブリの書き込み (wa)
  - ストリンスや文字列 (w)
  - Hexpairs (wx)
  - 全体的なファイルの中身 (wf)





# コマンド修飾子

## プレフィックス

- [1-9]
  - Repeat command n times
- ` (backtick)
  - Interpret the output of the command as r2
- “
  - Ignore special characters
- ` (grave accent)
  - Insert the output of a command
- !
  - Shell escape
- \$
  - Alias command
- \
  - Alias for =!

## サフィックス

- @
  - temporal seek
- |
  - system pipe
- ~
  - internal grep
- >
  - file redirect
- #
  - Comment
- ?
  - Show help
- j
  - Output in JSON

Demo

# ファイルシステムのマウントと検索機能

- 最初段階でradare2はフォレンジックツール飲みでした。
  - ファイルの中身のoffset 検索,その逆もOK
  - バイナリのパターンの検索、そして結果をダンプする機能
  - HFS, FAT, NTFS, EXT2に対応
  - まだ対応していません: Squash, jffs2
- 
- コマンド 'm' を使うとパーティションテーブルとファイルシステムの認識・読み込みができます。
    - GRUB開式のプラグイン. (GPL warning)
    - 他のコマンドもあり、例えば: io and r2 filesystems (wip)



Demo

# バイナリヘッタのパーシング

- 色々な種類バイナリヘッタの対応
- rabin2 -l
- 破損されたバイナリの認識・調査機能
- ディスクやメモリからの読み込み機能
- IO レーヤーのAPIやりとりの調査
- Virtual Address えむレーションスペース
- r2 -nn
- rabin2 -H
- memory ヘッタのパーシング (oba, !rabin2 -)
- リソースなどのダンプ機能

```
[0x00000000 0% 8736 /bin/ls] pcc 0 mach0_header
0x00000000
0x00000020
0x00000040
0x00000060
0x00000080
0x000000a0
0x000000c0
0x000000e0
0x00000100
0x00000120
0x00000140
0x00000160
0x00000180
0x000001a0
0x000001c0
0x000001e0
0x00000200
0x00000220
0x00000240
0x00000260
0x00000280
0x000002a0
0x000002c0
0x000002e0
0x00000300
0x00000320
0x00000340
0x00000360
0x00000380
0x000003a0
0x000003c0
0x000003e0
0x00000400
0x00000420
0x00000440
0x00000460
0x00000480
0x000004a0
0x000004c0
0x000004e0
0x00000500
0x00000520
0x00000540
0x00000560
0x00000580
0x000005a0
0x000005c0
0x000005e0
0x00000600
0x00000620
0x00000640
0x00000660
0x00000680
0x000006a0
0x000006c0
0x000006e0
0x00000700
0x00000720
```

Demo



# 調査とディスアセンブリについて

## 初心者向けのジェネリック調査機能

- -A, aa, aaa, aaaa, aaaaa, aaaaaaaaah!

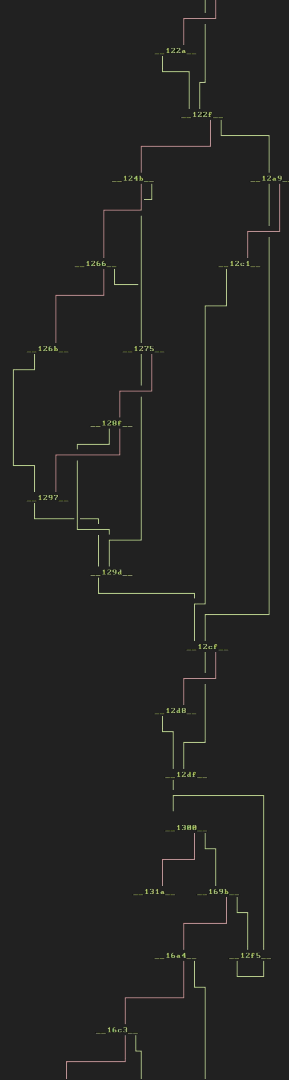
実はもっと面白いコマンドが沢山あります:

- e??anal.

それとも、もっと詳細なデータを狙う為に別途コマンド:

- aac, aar, aae, aav, aab, ...

ESILエミュレーターのコマンドもあります



# 調査とディスアセンブリについて

## Listing functions

- afl

## Listing basic blocks

- afb

## Graphing them

- agf

## Rename function

- afn

## Analyzing a single opcode

- ao

## Analyze single function

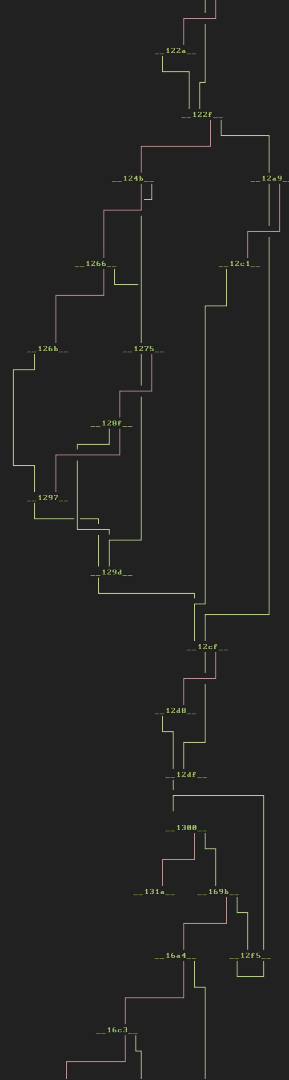
- af (e anal.hasnext)

## Alternative analysis loop

- a2f

## Autoname function

- afna



# 調査についてのオプション

もっと調べるコードがある場合:

- `anal.hasnext`

調査した上でもっと文字列を探すコマンド:

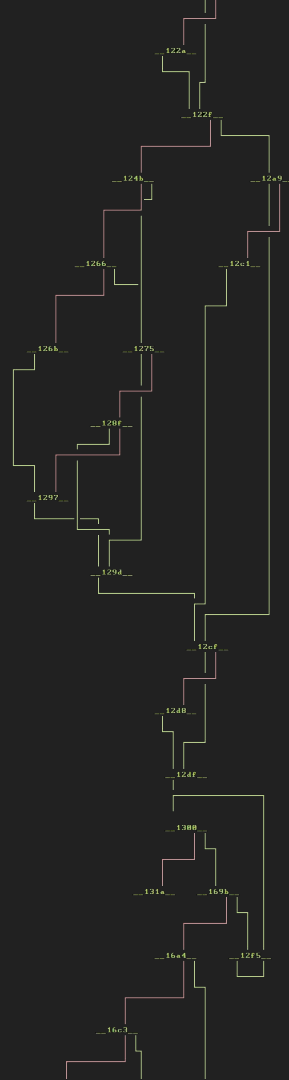
- `anal.strings`

実行コマンドなしの場所に下記のコマンドが便利

- `anal.noncode`

xrefやJump-Tables調査

- `anal.jmptbl`



Demo

# マルチプルフォーアートのデータをプリント

‘b’ コマンドを使い、ブロックサイズ(blocksize)の設定

ズームニューワー機能, entropy調査, データブロックのカラーリング, 実行命令コードなど

- "p" コマンドとは？

Format string-like strings by parsing the memory at given address

特定なメモリアドレスにある文字列データのパーシングについて

- pf xxi foo bar cow @ addr

Demo

# デバッグとエミュレーター機能

スタティックバイナリの調査モードで エミュレーターエンジン(ESIL VM)を使い、radare2 はデバッグ行動のシミュレーションが出来る。実行命令コードはESILにパーシング、そしてシミュレーションのレジスターのバリューの計算が出来る、ようはバイナリを実行せず、解読が出来る機能機能。なお、デバッグをする時に -d のパラメーターを使えばダイナミック調査モードを動かす事が出来る

- r2 -d
- ジャンプ命令(addrのアドレス迄は) ⇨ dcu addr
- ステッピングは ⇨ ds や dso

# デバッグの上でエミュレータを動かす方法

ネイティブとリモートデバッグサーバに接続が可能

- `dbg:// winedbg:// gdb:// windbg:// qnx:// ..`

その他 low level デバッグについて:

- バックステップ (Thanks Ren Kimura!)
- メモリスナップショット
- Software/Hardware ブレークポイント
- Assisted debugging (emulation + debug)
- Tracing
- Filedescriptor manipulation



# Rarun2 プロフィール

実行環境のコンフィグについて下記の方法で設定ができます:

- r2 -r or dbg.profileの情報でテキストファイルフォーマットの洗濯が可能
- カンマのof directives via dor or -R commandline flags

radare2環境の上でgid, uid, chroot, chdir, environment, arguments, filedescriptorsの変更する事ができます。

- コマンドのパラメーターについてストリングス、ファイル、もしくはstdoutにも対応する事が可能

\$ man rarun2

# デバッガーでのデータをプリントする時に

スタックの中身を見せる事ができます

- dbt - backtrace
- pxr@r:SP

ローラル変数のバリューを見せる事が可能(varやレジスター)

- afvd

カラーバーの変更も可能

- p=

Demo

# インターフェース

- メインはコマンドプロンプトCLI
- 最近ではビジュアルモードが人気
- ウェブサーバのGUI (r2 -c=H)

ビジュアルモードの時に、コマンドインのターフェースはキースツロックとなります

- ビューの変更: pP”|=...
- テッピング: ‘s’, bPトグル, など
- コマンド履歴/history
- ビジュアルアセンブラー
- インタラクティブ・グラフ

```
3. radare2
[0x100001200 11% 125 /bin/ls]> pd $r @ main
;-- entry0:
;-- func.100001200:
;-- rip:
(fcn) main 1190
bp: 6 (vars 6, args 0)
sp: 0 (vars 0, args 0)
rg: 0 (vars 0, args 0)
0x100001200 55          push rbp
0x100001201 4889e5      mov rbp, rsp
0x100001204 4157       push r15
0x100001206 4156       push r14
0x100001208 4155       push r13
0x10000120a 4154       push r12
0x10000120c 53         push rbx
0x10000120d 4881ec180600. sub rsp, 0x618
0x100001214 4989f7     mov r15, rsi
0x100001217 4189fe     mov r14d, edi
0x10000121a 480d85c0fdff. lea rax, [local_240h]
0x100001221 488945d0   mov qword [local_30h], rax
0x100001225 4505f6     test r14d, r14d
0x100001228 7f05      jg 0x10000122f
0x10000122a e8d1310000 call sym.func.100004400
0x10000122f 488d35ba3800. lea rsi, 0x100004af0
0x100001236 31ff     xor edi, edi
```

# グラフィックUI / GUIについて

必要な時に r2pm でインストールが可能

- Gradare2 (simple gtk2/3 + vte ui)
- Ragui (unreleased)
- Bokken (unmaintained)
- Blessr2 (nodejs-blessed based UI)
- WebUIs (material, enyo, tiled, ...)
- Radare2gui (.net for windows)
- Cutter (previously known as laito)
- ...

The screenshot displays the Cutter application interface, which is a GUI for the Radare2 framework. The main window is titled "Cutter" and shows the assembly code for the function `sym___gcc_register_frame`. The assembly code is displayed in a list view, with the current instruction highlighted. The right side of the window shows a control flow graph (CFG) with nodes representing instructions and edges representing control flow. The nodes are labeled with addresses, such as `0x00401330`, `0x00401335`, `0x00401331`, `0x00401339`, `0x00401343`, `0x00401346`, and `0x0040134F`. The bottom of the window shows a dashboard with various tabs, including "Dashboard", "sym\_\_\_gcc\_register\_frame", "Entry Points", "Functions", "Strings", "Imports", "Symbols", and "Notepad". The "Sections" tab is active, showing a list of sections and a pie chart representing the distribution of sections. The sections list includes:

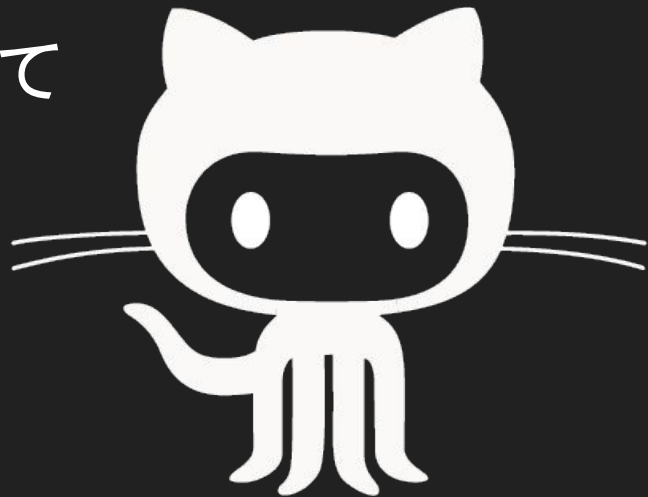
Address	Size	Type
0x00401300	3.5K	text
0x00402000	512	data
0x00403000	3K	data
0x00404000	0	bss
0x00405000	512	data
0x00406000	512	text
0x00407000	512	text
0x00408000	512	text
0x00409000	3K	text
0x0040A000	512	text
0x0040B000	512	text
0x0040C000	3K	text
0x0040D000	512	text
0x0040E000	512	text
0x0040F000	512	text
0x00410000	512	text
0x00411000	512	text
0x00412000	512	text

The pie chart shows the distribution of sections, with the largest section being `text` (3.5K) and the smallest being `bss` (0).

Demo

# カスタマイズ(read:"r2 hack")について

- ライブラリー
  - symlinksのライブラリー連携の読み込み
  - `cd libr/* ; vim ; make; run`
- プラグイン
  - `./configure-plugins`
  - `r2pm`
- バインド
  - C バインドのAPI の使う事が可能 (Python, Perl, Ruby, Scheme, Haskell)
  - Thanks to Valabind
- スクリプト
  - スクリプトは RLangでPython, C, やValaでコーディングが可能
  - バンドの自動ローディング設定も可能



# r2pipe

## r2の自動課されている方法

- シングル api 機能: 1個コマンドを実行し、結果を見る事
- その他のスクリプトを使いマルチプル実行コマンドが可能

## マルチプルpipeコミュニケーションチャンネル

- Pipe
- Socket
- HTTP
- Native
- Spawn





Demo

# Third Party アドオンについて

色んなアドオン機能の開発プロジェクトがあります、例えば:

- scripts, plugins, patchesの対応...
- r2pmまたはパッケージマネージャ経由のインストール
  - Install everything by default in your home (unless -g is used)
- Cでのデコンパイル機能, SMT Solvers, tools, ...
- r2pm -sのコマンドで一覧を見えます。

(DEMO)

# r2frida

- Fridaはradare2のフックエンジンであり, javascript インジェクションのコマンドを使い、ローカルやリモートのプロセスにフックした後色々なコマンドやりとりができます。
  - It comes with a REPL, a tracer, process list, etc..
- Radare2 を使えばFridaのフロントエンドとして使う事ができます。
  - Uses the power of the IO plugins
  - Access functionality via io->system
  - Using the \ or =! Command
- もしくはrarunにあるr2preload を使うと自分自身のプロセスインジェクションも可能です。コマンド: `self://`

# WineDBG

- Wine is not a Windows Emulator
- Comes with winepdb, a very rustic commandline low level debugger
- The io.winedbg plugin allows to interface with it
- Similar to the bochs:// one
- Allows to debug window programs with r2 on Linux and Mac platforms.
- In early stage of development
  - Lot of potential here

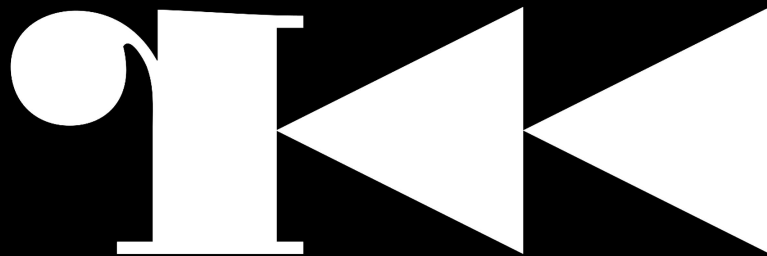
# その他 3rd Partyデバッガーバックエンド

- GDB / LLDB
  - Debug kernels via the gdbserver embedded in qemu, vmware, vbox, ..
  - Apple's debugserver, GNU's gdbserver
  - AVR emulator and jtag
- WINDBG
  - Connect to a windbg server
- WINEDBG
  - Debug Windows programs on wine (Linux, Mac, ..)
- QNX
  - The debugserver used in automobile
- Bochs
  - X86 CPU debugger

Demo

しつもん がありますか？

(Questions?)



Thanks For Watching!