DeforaOS: A Journey into OSDev

area41 Security Conference
Zurich, Switzerland
June, Monday 2nd 2014

Pierre Pronchery <khorben@defora.org>

Why DeforaOS

- I don't want to care if it's my desktop, laptop, tablet, phone, media server or whatever I use
- I want my data and my preferences always consistent and available
- Securely, easily, transparently
- Started the project 14 years ago, 10 under its current shape
- Still kicking it! (200,000+ SLOC)

Structure of DeforaOS

1. Distributed framework
   the main goal of the project
2. Desktop environment
   because I need one that features the framework
3. Self-hosting environment
   because it's fun (and I have tons of ideas)

I'm not going to speak about it

- Well, not directly
- It's been a rough journey, with its frustrations and discoveries
- It might be more interesting to speak about some of these encounters actually, let's see

The real agenda today

I. Parsing
   i. Of code duplication and ultimate doom
   ii. How it may be solved at the OS level
II. Networking
   i. IPv6 & HTTP bashing
   ii. Clean-Slate Internet
III. Distribute all the things
   i. Concept
   ii. Some demos

I. Parsing: MD5

- ...the reference implementation from RFC1321 has a bug (integer size on 64-bit machines)
- People copy & paste MD5 code from various sources all the time
  - Heisenbugs
  - Very tough to fix them all, see for Fedora https://fedoraproject.org/wiki/Packaging:No_Bundled_Lib

I. Parsing: GZIP

- According to the specification in RFC1952, an archive can contain multiple files (just append archives)

```
  2.2. File format

       A gzip file consists of a
series of "members" (compressed
data sets). The format of each
member is specified in the
following section. The members
simply appear one after another in
the file, with no additional
information before, between, or
after them.
```

I. Parsing: GZIP

- But the reference implementation (GNU) uncompresses to a single file:

```
$ echo abc > 1; gzip 1
$ echo def > 2; gzip 2
$ cat 1.gz 2.gz > 3.gz
$ gunzip 3.gz
$ cat 3
abc
def
```

- What should we do?

I. Parsing: It's worse than that

- Think about two differing implementations of the same specification
- They will not see (or output) the same data for the same input, because they're not identical
- Case study: anti-viruses
  - They check files while applications open them
  - But they don't have the same code as the application! (so yes, they can be evaded by definition)
- We need a single implementation for everything

I. Parsing: Elements of solution

- Every high-level language must bind the reference library (and only this code!)
- (think about what this means for PyPy)
- File formats must be implemented as plug-ins
  - Common API for each file type (image/*, video/*...)
  - Simpler is better
  - Still allows proprietary software to interface with GPL code
  - Should mention respective license information though

I. Parsing: Going deeper

- Probably time for a new libc
- Object-Oriented, modern idioms:
  - A new string library! (a new everything really)
  - Mutators, hashing, parsing, iterators...
- In DeforaOS:
  - System/src/libc for learning how to write one
  - System/src/libSystem for its future replacement
  - See <System/license.h>

I. Parsing: License information

System libSystem: `/include/System/license.h:`

```
typedef unsigned int LicenseFlags;
# define LF_CAN_JAIL                    0x00000001
# define LF_CAN_MODIFY                  0x00000002
# define LF_CAN_PATENT                  0x00000004
# define LF_CAN_REDISTRIBUTE            0x00000008
# define LF_CAN_REVERSE_ENGINEER        0x00000010
# define LF_KEEP_ACKNOWLEDGEMENT        0x00000100
# define LF_KEEP_COPYRIGHT              0x00000200
# define LF_KEEP_DISCLAIMER             0x00000400
# define LF_KEEP_SOURCE_CODE            0x00000800
# define LF_VIRAL                       0x00001000
# define LF_VIRAL_LIBRARIES             0x00002000

# define LICENSE_PUBLIC_DOMAIN_FLAGS \
        (LF_CAN_JAIL \
         | LF_CAN_MODIFY \
         | LF_CAN_REDISTRIBUTE \
         | LF_CAN_REVERSE_ENGINEER)
```

I. Digression: About the libc

- Supports multiple kernel and architecture combinations
- Gets rid of the historical heritage (goal: latest POSIX specification)
- Unify the API regardless of the underlying kernel
- Forces the code linked to be strictly portable
- ...Allows cross-compilation without a cross-compiler in some cases
- And some more cool stuff

I. Digression: libc on NetBSD

- Compile a freestanding static binary, it will omit the .notes section (used to recognize NetBSD)
- Kernel was running it as Linux emulation:

```
$ gcc -static -ffreestanding \
  -U __NetBSD__ -D __linux__ \
  -isystem /path/to/libc/include \
  -nostdlib -o test \
  src/libc.a src/start.o test.c
$ file test
test: ELF 64-bit LSB executable,
x86-64, version 1 (SYSV),
statically linked, not stripped
$ ./test
```

II. Networking: How it started

- Reminder: I wanted my applications to distribute data access
  - IRC clients: logs, channels, private messages
  - News readers: subscriptions, messages already read
- Original AppEngine idea:
  - minimal functional core logic
  - the UI simply connects to it
- Implement the logic once, interchangeable UI

II. Networking: The eye opener

- Wrote a simple serialization and RPC framework
- Realized one single protocol is enough
- What changes is the remote code that's called
  - Not the messaging infrastructure in use
- It's possible to do everything with one protocol and one port:
  - Authentication, multiplexing and routing of messages
  - Talking to the final application: send_email(), get_page(), download_file()...

II. Networking: Parsing

- We only need one parser!
- We only ***want*** one parser!
- We don't need UDP or TCP ports
  - We can simplify them too!
- The OSI model is ***WRONG***
  - There is only one layer, repeating itself

(when I figured this out on my own I thought either no one would believe me, or I must be wrong, or someone else must have realized)

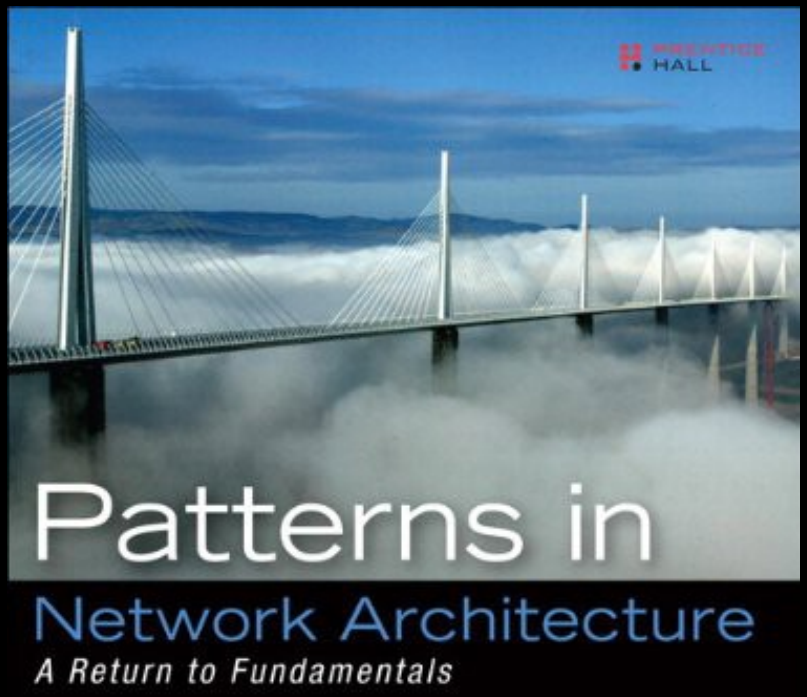
II. Networking: All broken

- We have a different parser for each layer...
- Alright so we need reliable messages:
  - UDP is not reliable
  - TCP does not give us messages

II. Networking: New socket API

- socket(PF_INET, SOCK_SEQPACKET)
- Ah no, it's been there all the time but never implemented for IPv4 (MPP!)
- Notifications and policies upon events such as disconnect/reconnect, address changes, etc...
- See also SCTP by the way

II. Networking: My new bible



- Patterns in Network Architecture, by John Day from Boston University, at Prentice Hall
- Spread the holy messages!
- Here are a few...

II. Networking: It's called Internet

- As in multiple networks interconnected
- BUT WE HAVE ONLY ONE ADDRESS SPACE ...so we are forced to all use the same network
- Private IP ranges (RFC1918) another example
  - One address, thousands of hosts? WTF?!?
- But now, think about the shiny silver bullet forced onto us... IPv6

II. Networking: Mad IPv6

- The “solution”: let each and every network and device be fully reachable always:
  - Your laptop, mobile phone, tablet, media player...?
  - Your super secure Intranet (that's always been designed with this in mind obviously)
  - Your services while you struggle to configure them
  - How do you pull the plug during a netinstall?

IS THAT NOT COMPLETE NONSENSE?!?

II. Networking: IPv6 kthxno

- Call to reality: we have a network of networks
- A good deal of them ARE PRIVATE
- They still need to be able to access other networks as authorized
- To me the key is with access control:
  - On the final resource (or closest endpoint smart enough)
  - On the home gateway and routers

WE NEED ANOTHER ALTERNATIVE TO IPv4

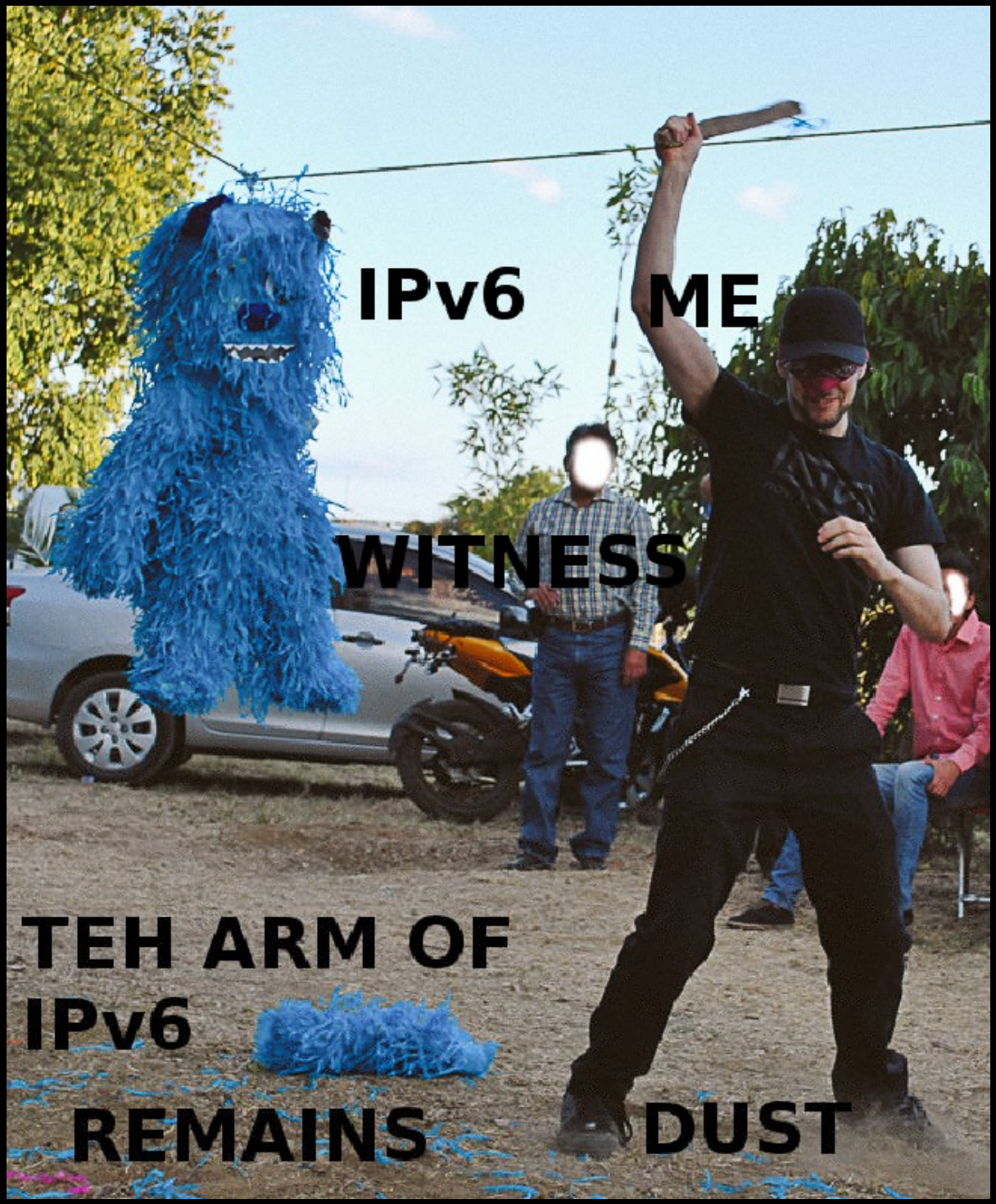
II. Networking: IPv6 love to hate

- The idea is to ban private networks altogether
- Oh yeah and let's change the notation too
  - "." instead of ":" for ports
  - Like it's not confusing enough as it is for everyone?

WE DON'T EVEN NEED PORTS ANYWAY!!1

II. Networking: IPv6 & parsing

- Remember what I said about multiple parsers?
- Enter the dual stack world
- Who will guarantee that some.host.tld will reply the same way in IPv4 and IPv6?
  - Answer: it is impossible and will not happen
- That's a security risk again


IPv6
ME
WITNESS
TEH ARM OF
IPv6
REMAINS
DUST

II. Networking: Name & Address

- We mix names and addresses
- Sockets should be bound to names and follow the address of the remote peer (and their own)
- Naming is difficult and nobody agrees :(
  (but I don't care since I make my own shit)
  (I don't have a silver bullet for this as of yet)

II. Networking: Privacy

- Privacy deserves at least one slide of its own
- Topological addressing: you know your immediate neighbors and they know you
- With this addressing paradigm, it is possible to improve privacy by forcing routers to assign a new virtual address (and port) for each connection (a bit like Tor without the onion)
- Should benefit from an increased address space
- This could be an option for MPP

II. Networking: Bashing Intarwebz

- You thought I would forget?
- HTTP is UDP over TCP
  - And cookies just a bigger port number *__duh__*
- Javascript introduced CSRF in the first place
  - THE RFC OF HTTP WARNED AGAINST IT http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
  - Should have never been used for more than UI
  - Disable it, remember: “simpler is better”

II. Networking: Kill Javascript

« […] user agents to represent other methods, such as POST, PUT and DELETE, in a special way, so that the user is made aware of the fact that a possibly unsafe action is being requested »

Javascript in browsers does not respect RFC 2616 for HTTP/1.1

It *is* the mother of tons of security issues!

III. Distribute all the things

Let's get back to the creative stuff

Quick reminder:

- Everything is IPC
- One protocol is enough
- Can we make it even simpler?

Enter DeforaOS libApp...

Yes we can...

III. Distribute: libApp

DeforaOS libApp is a library to:

- Hook a configurable list of function calls
- Run them over the network instead
  ...over just about any transport protocol
- Write distributed applications without writing networking code

III. Distribute: Possibility 1

Think about (no particular order)
open(2), read(2), write(2), close(2), opendir(2), readdir(2), closedir(2), access(2), chmod(2), link(2), symlink(2), unlink(2), rename(2), chown(2), fchown(2), flock(2), lseek(2), mkdir(2), rmdir(2)...

III. Distribute: VFS

- Remote file sharing! In just 679 lines
- DeforaOS libVFS uses LD_PRELOAD and the file:// protocol (it supports remote hosts!)
- Known issues:
  - compatibility interface for syscalls
  - absolute versus relative paths
- But it works^Wworked
  - the DeforaOS libc can help here

III. Distribute: Possibility 2

Think about (no particular order)
getaddrinfo(2), socket(2), bind(2), listen(2), connect(2), accept(2), recv(2), send(2), recvmsg(2), sendmsg(2)...

III. Distribute: VPN

- Virtual Private Networking!
  312 lines, *no network code*
- DeforaOS libVPN uses LD_PRELOAD but is still far from complete
- Known issues:
  - the compatibility interface for syscalls again
- But it should work!
  - the DeforaOS libc can help here again (libsocket)

III. Distribute: Possibility 3

Think about (you know the drill)
getloadavg(3), getvfsstat(3), getifaddrs(3)...

III. Distribute: Probe

- Remote system monitoring!
- Known issues:
  - These APIs are not portable
    (but they can wrapped easily)
- Again, it worked!

III. Distribute: Possibility 4

Think about (this one is exciting) glBegin(3), glClear(3), glClearColor(3), glClearDepth(3), glColor3f(3), glColor3i(3), glEnd(3), glFlush(3), glLoadIdentity(3), glRotatef(3), glTranslatef(3), glVertex3f(3)...

III. Distribute: GServer

- Distributed 3D graphics!
- From multiple applications to a single screen
- From one machine to multiple screens
- From many machines to a single screen
- Same scene, different perspective
- Endless possibilities
- It did work!
- Draw each client into a texture, you have X btw

III. Distribute: Possibility 5

- Bind all the things!
- Generic command-line tool: AppClient(1)
- Remote OpenGL programming from the shell
- It worked!

Demo: Building an image

```
$ ./build.sh -O PREFIX="/usr" \
  -O IMAGE_FILE="DeforaOS.img" \
  -O IMAGE_KERNEL="/netbsd.gz" \
  clean image
```

Demo: Running the image

```
$ qemu-system-x86_86 \
  DeforaOS-image.img
```

Demo: compilation framework

- Subject for another talk but can't resist
- Where the libc is useful again

Special thanks

- John Day
- Wikileaks, Tor
- Edward Snowden
- Bruce Sterling, Cory Doctorow, c-atre <3
- LANGSEC, Phenoelit, THC, ÜberWall...
- Everyone neighborly

Conclusion

- Love & hate: khorben@defora.org, @khorben
- I hope I convinced you
- Otherwise feel free to try to save me from conquering^Wsaving the world
- Have fun!