Time to retire Linux (and C) in IoT

Lin Zhong
http://www.recg.org
By analyzing the problems of Linux, create a software systems research agenda for secure, efficient IoT devices.
Our mission

better computers

http://www.recg.org
World first massive MIMO system prototype
Small

Mobile & embedded software systems & hardware
Linux is taking over the world
Linux IoT devices in our home
Large Hadron Collider
@CERN
DeLaval automatic cow milking machine
Cars

Linux is under your hood
BMW, Chevrolet, Honda, Mercedes, and Tesla all have one thing in common besides making cars: Their cars all run Linux.

By Steven J. Vaughan-Nichols for Linux and Open Source | April 11, 2018 -- 23:40 GMT (16:40 PDT) | Topic: Hardware
Microsoft keeps cozying up to former rival Linux, and will now use it to secure web-connected gadgets

- Microsoft's latest use of Linux comes weeks after a leader of the Windows division lost a seat on the company's senior leadership team.
- Microsoft has previously looked to Linux to boost its public cloud and operate equipment in its data centers.

Jordan Novet | @jordannovet
Published 6:03 PM ET Mon, 16 April 2018 | Updated 7:54 PM ET Mon, 16 April 2018

Microsoft, which once called Linux a "cancer," is now embracing the open-source operating system and using it to secure internet-connected devices.
Why IoT devices embrace Linux

• Moore’s Law made silicon cheaper
  • 8-bit => 16-bit => 32-bit => \textbf{32-bit with MMU}
• Linux is free and ready available
  • well seasoned network stack
The security crisis
175,000 IoT cameras can be remotely hacked thanks to flaw, says security researcher

Researchers have found that it's trivial to remote camera.

By Danny Palmer | July 31, 2017 -- 16:01 GMT (09:01 PDT) | Topic

Hackers found 47 new vulnerabilities in 23 IoT devices at DEF CON

The results from this year's IoT hacking contest are in and it's not a pretty picture

By Lucian Constantin | Romania Correspondent, DIG News Service | SEP 13, 2016 8:28 AM PT

IoT devices being increasingly used for DDoS attacks

Malware is infesting a growing number of IoT devices, but their owners may be completely unaware of it.

By: Symantec Security Response | Created 22 Sep 2016 | SYMANTEC EMPLOYEE | 0 Comments

Update your Belkin WeMo devices before they become botnet zombies

Researchers have disclosed a critical vulnerability in Belkin's WeMo Switch and possibly other devices

By: COMPUTERWORLD | Created 12 Oct 2016 | WeMo
Hackers Used New Weapons to Disrupt Major Websites Across U.S.

A map of the areas experiencing problems, as of Friday afternoon, according to downdetector.com.

By Nicole Perlroth
Oct. 21, 2016
Four botnets generated 10 DDoS attacks exceeding 300 Gbps between July 2014 and December 2016. Seven of these occurred in 2016.

Source: akamai
A rapid increase in scans of port 23 and 2323 began on May 13, 2016 as the Mirai botnet attempted to log into unsecure IoT devices.

**Security Spotlight:** Internet of Things and the Rise of 300 Gbps DDoS Attacks

Source: akamai
The crisis is rooted in Linux/C
C is not safe

- No memory (type) safety
- No language support for concurrency

SoK: Eternal War in Memory

László Szekeres†, Mathias Payer‡, Tao Wei*†, Dawn Song‡

Threads Cannot Be Implemented As a Library

Hans-J. Boehm
Most kernel vulnerabilities rooted in C

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Mem. corruption</th>
<th>Policy violation</th>
<th>DoS</th>
<th>Info. disclosure</th>
<th>Misc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing pointer check</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Missing permission check</td>
<td>0</td>
<td>15</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Buffer overflow</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Integer overflow</td>
<td>12</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Uninitialized data</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>Null dereference</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Divide by zero</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Infinite loop</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Data race / deadlock</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Memory mismanagement</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>32</strong></td>
<td><strong>16</strong></td>
<td><strong>60</strong></td>
<td><strong>37</strong></td>
<td><strong>2</strong></td>
</tr>
</tbody>
</table>

---

**Linux kernel vulnerabilities:**
State-of-the-art defenses and open problems

Chen et al APSys 2011
Linux depends on human developers for correctness

“Give enough eye balls, all bugs are shallow”

— Linus’ Law
Linux depends on human developers for correctness

Keep incompetent programmers out by choosing C

C++ is a horrible language. It's made more horrible by the fact that a lot of substandard programmers use it, to the point where it's much much easier to generate total and utter crap with it. Quite frankly, even if the choice of C were to do *nothing* but keep the C++ programmers out, that in itself would be a huge reason to use C.

In other words: the choice of C is the only sane choice. I know Miles Bader jokingly said "to piss you off", but it's actually true. I've come to the conclusion that any programmer that would prefer the project to be in C++ over C is likely a programmer that I really *would* prefer to piss off, so that he doesn't come and screw up any project I'm involved with.

Linus Towallds, 2007
No guarantee

Introduced by Robin Seggelmann in 2011, code reviewed by Stephen Henson, into OpenSSL source code, 12/31/2011
Bug reported 04/01/2014
It’s worse for IoT devices

• Not many eye balls
• Not many competent developers
• Low profit margin
• Short-time to market
IoT system kernel is driver-rich

System-on-a-chip for IoT devices teeming with non-CPU devices

Drivers are the most buggy part of kernel
There are a lot of IoT devices. There are a lot of IoT vendors.
Sturgeon’s Law:
90% of *everything* is crap
Sturgeon’s Law:
90% of **IoT vendors** are crap

- Device shipped with debug access enabled
- Hard-coded passwords
- Unused features left in
- Difficult to manage
- Impossible to update

Christopher Biggs, “The Internet of Scary Things - tips to deploy and manage IoT safely”, 2017
Sturgeon’s Law:
90% of IoT devices are crap
“When identical devices are manufactured and sold in huge quantities, the possibility for mass takeover of those devices is real.”

All top IoT malware identified by Symantec affect Linux systems

- Linux.Darlloz (aka Zollard)
- Linux.Aidra / Linux.Lightaidra
- Linux.Xorddos (aka XOR/DDoS)
- Linux.Gafgyt (aka GayFgt, Bashlite)
- Linux.Ballpit (aka LizardStresser)
- Linux.Moose

- Linux.Dofloo (aka AES/DDoS, Mr. Black)
- Linux.Pinscan / Linux.Pinscan.B (aka PNScan)
- Linux.RouteRem (aka Remainten, KTN-Remastered, KTN-RM)
- Linux.Wifatch (aka Ifwatch)
- Linux.LuaBot

Source: Symantec 2016
https://www.symantec.com/connect/blogs/iot-devices-being-increasingly-used-ddos-attacks
Emergent Tech ➤ Internet of Things

Linux is part of the IoT security problem, dev tells Linux conference

Does that 'thing' really need to run Linux, given alternatives have smaller attack surfaces?

By Richard Chirgwin 19 Jan 2017 at 07:01

https://www.theregister.co.uk/2017/01/19/iot_will_get_worse_before_it_gets_better_dev_tells_linux_conference/
IoT security is about securing the bottom 10%
Linux/C invites hardware-based isolation

- Privilege levels
  - User, kernel, hypervisor, monitor….
- MMU
- Intel SGX, ARM TrustZone
Microsoft Azure Sphere requires more complicated hardware

Figure 1. Architecture of the MT7687 Wi-Fi-enabled Microcontroller.

With MediaTek’s assistance we modified and extended the MT7687. We made three changes to the MT7687 to convert it into Sopris (see Figure 2): we added a Pluton security subsystem, we upgraded the primary CPU to a CPU including a memory management unit (MMU), and we increased the amount of on-die SRAM. The Pluton security subsystem forms the hardware root of trust for Sopris. Unlike the much more primitive memory protection unit (MPU) found in most microcontrollers, the MMU on the Sopris processor supports multiple levels of isolation and multiple independent address spaces from which an OS can create process-isolation compartments. The addition of on-die SRAM allows easy experimentation with many OS configurations while maintaining the security of on-die memory.

Figure 2. Architecture of the Experimental Sopris Highly Secure WiFi-enabled Microcontroller.

The Seven Properties of Highly Secure Devices
Galen Hunt, George Letey, and Edmund B. Nightingale
Microsoft Research NExT Operating Systems Technologies Group

Figure 3. Prototype USB-powered developer board based on Sopris.
Shall we trust hardware for isolation?

**Meltdown**
- Unprivileged process accessing privileged data
- Failure of privilege levels

**Spectre**
- Process accessing address out of bound
- Failure of MMU isolation
Monsters in our home
aka the toaster apocalypse
Monsters in our home
aka the toaster apocalypse
IoT OS Wishlist

- C => Safe language
- No reliance on hardware isolation
Linux has an efficiency problem
Runtime enforcement of correctness is expensive

Expensive system calls

- Mode switch overhead: Stack access, exception handling
- Locality reduction: cache, TLB pollution

Soares & Stumm, OSDI 2010
IoT services are I/O and network-oriented

• A piece of sensor data has to cross the user-kernel boundary twice!!!
IoT OS Wishlist

- C => Safe language
- No reliance on hardware isolation
- Runtime enforcement => Static enforcement
Linux has a maintainability problem
Too big too complex for innovation

Lines of code (Million)

Linux Kernel

<table>
<thead>
<tr>
<th>Year</th>
<th>Lines of Code (Million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1991</td>
<td>0.1</td>
</tr>
<tr>
<td>1994</td>
<td>1</td>
</tr>
<tr>
<td>1997</td>
<td>10</td>
</tr>
<tr>
<td>2000</td>
<td>100</td>
</tr>
<tr>
<td>2003</td>
<td>1000</td>
</tr>
<tr>
<td>2006</td>
<td>10000</td>
</tr>
<tr>
<td>2009</td>
<td>100000</td>
</tr>
<tr>
<td>2012</td>
<td>1000000</td>
</tr>
<tr>
<td>2015</td>
<td>10000000</td>
</tr>
<tr>
<td>2018</td>
<td>100000000</td>
</tr>
</tbody>
</table>
Highly modularized but entangled

Modules contain submodules
Modules hold states for each other

Fate sharing
Live update almost impossible
State spill is ubiquitous and deep

- When one software entity’s state undergoes a lasting change as a result of handling an interaction with another entity.

- Migration, fault isolation, fault tolerance, live update, hot-swapping, maintainability.....

State spill explained

Body man

Vice President
Solution in Data center

Leverage redundancy
Add layers of indirection
Redundancy is a luxury out of Data Center
Either bring the service down
Reboot a powerplug?
Or never updated

Mirai botnet, a DDoS nightmare turning Internet of Things into Botnet of things
Graying Linux developers look for new blood

The top Linux developers are getting older and The Linux Foundation is addressing the issue.

By Steven J. Vaughan-Nichols for Linux and Open Source | August 29, 2013 -- 19:03 GMT (12:03 PDT) | Topic: Linux
Getting older and disappearing

https://blog.bitergia.com/2013/02/01/demographics-of-linux-kernel-developers-how-old-are-they/
Voyager 2: 40 years after launch, 20 Billion km away
The Loyal Engineers Steering NASA’s Voyager Probes Across the Universe

As the Voyager mission is winding down, so, too, are the careers of the aging explorers who expanded our sense of home in the galaxy.

By KIM TINGLEY  AUG. 3, 2017
IoT OS Wishlist

- C => Safe language
- No reliance on hardware isolation
- Runtime enforcement => Early enforcement
- Modularization => Spill free modularization
“Let’s retire Linux/C”
“Why? Linux/C must be good since it has taken over the world”
QWERTY phenomenon
Ken Thompson and Dennis Ritchie
DEC PDP-11, 16 bit, 1970-ish
Designed at a time when computer was simpler and more expensive by orders of magnitude.
Humans were relatively cheap; Let developers manage memory and concurrency
Computing is relative cheap; Let **machine** manage memory and concurrency.
“Unix and C are the ultimate computer virus”

Richard Gabriel in *The Rise of "Worse is Better"*
Getting older and disappearing

Maybe this is good news!

https://blog.bitergia.com/2013/02/01/demographics-of-linux-kernel-developers-how-old-are-they/
Science makes progress funeral by funeral

“A new scientific truth does not triumph by convincing its opponents and making them see the light, but rather because its opponents eventually die, and a new generation grows up that is familiar with it.” ——— Max Planck
Then what?
IoT OS Wishlist

• C => Safe language
• No reliance on hardware isolation
• Runtime enforcement => Early enforcement
• Modularization => Spill free modularization
Theseus: a runtime composable OS
Ship of Theseus

The ship wherein Theseus and the youth of Athens returned from Crete had thirty oars, and was preserved by the Athenians down even to the time of Demetrius Phalereus, for they took away the old planks as they decayed, putting in new and stronger timber in their places, in so much that this ship became a standing example among the philosophers, for the logical question of things that grow;

one side holding that the ship remained the same, and the other contending that it was not the same.

— Plutarch (Theseus)
1. Rise up to the language opportunity

- Safe languages have been tried in the past
  - Modula-2, Oberon, Erlang, C#, Java, Go, ....

None took off due to underlying runtime or garbage collection requirement
Rust
born 2010 at Mozilla Research to develop a new web engine

• NO RUNTIME, NO GARBAGE COLLECTION!!!
• Memory (and type) safe
• Concurrency safe
• Performance close to C

• Strong type systems for static enforcement of correctness
2. No reliance on hardware isolation

- Hardware isolation is **expensive** and **difficult to verify**, incurs **runtime overhead**
End-to-end argument

“Functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level.”

Saltzer, Reed & Clark 1984
2. No reliance on hardware isolation

- Hardware should focus on performance and efficiency
- Software (the end) enforces isolation

Tock (SOSP’17) shows software isolation is achievable on low-cost micro controller WITHOUT MMU
3. Earlier enforcement of correctness

Rust opens the door

Strong, novel type systems

Linux/C

Design time
Implmntn. time
Compile time
Install time
Load time
Run time
Post mortem

Time of enforcement
3. Earlier enforcement of correctness
Rust opens the door

• Trust the compiler/type system
• Run all software in the same privilege mode
• Disappearing OS/kernel
If Linux/C is airport security check
Theseus/Rust is TSA Pre

4. Spill free modularization

(a) Monolithic Kernel

(b) Microkernel OS
Building Modules in Isolation

• Each module is a separate Rust crate
  • Compiled into individual binaries, isolated into private “namespaces”

Standard OS

No true distinction between modules, or blurry lines

Theseus

Easy to extricate a single crate due to clear boundaries
4. Spill free modularization

Implementing the OS like a distributed system of tiny modules

Theseus is “a bag of modules”
Open questions

• Performance optimization
  • Spill freedom can incur 3X slowdown

• Do more at Compile time?
  • Can Type System do more? (State spill as session types)

• Design for formal verification
IoT OS Wishlist

- C => Safe language
- No reliance on hardware isolation
- Runtime enforcement => Early enforcement
- Modularization => Spill free modularization
(Competent) C programmers are slowly getting rarer.
(Competent) C programmers are slowly getting rarer