Why has computer security failed to scale, and what we can do about it?

Paul Kocher
(paul@paulkocher.com)

Korea Crypto Forum, Invited Lecture
Nov 14, 2019

My background

Researcher & entrepreneur

Technical projects include:

- Protocols (incl. SSL v3.0 / TLS 1.0 "🔒")
- Timing attacks
- Differential power analysis & countermeasures
- Renewability & forensics systems
  - Blu-ray BD+, pay TV, Vidity/SCSA…
- Many hardware/ASIC projects
  - Pay TV, anti-counterfeiting, keysearch…
- CryptoManager ASIC & manufacturing solutions
- Spectre (indep. discovery w/ Jann Horn)

Founded+ran Cryptography Research (1995-2017)

- No outside investors – each stage funded the next
  Consulting → Licensing → Products → Solutions
- Acquired by Rambus ($342.6M)

Co-founded ValiCert

- Venture-backed data security company. IPO 2001, acquired 2003

Member of U.S. National Academy of Engineering & National Academies' Forum on Cyber Resilience; Fellow of the IACR

Advisor + investor for security start-ups

We are working hard But not succeeding

Metrics of security effort…

- Companies founded
- Papers published
- Algorithms designed
- Software developed
- Products purchased
- Code reviewed
- Users trained
- Patches applied
- Money spent

HIV status of over 14,000 people leaked online, Singapore authorities say

South Africa hacked: about 30 million ID numbers leaked

Cybercrime Damages $6 Trillion By 2021

More than $1.5b in cryptocurrency stolen since early 2017

FBI warns Russians hacked hundreds of thousands of routers

At Least Three Billion Computer Chips Have the Spectre Security Hole

Companies are rushing out software fixes for Chipmageddon.

More than 2.5 billion records stolen or compromised globally in 2017

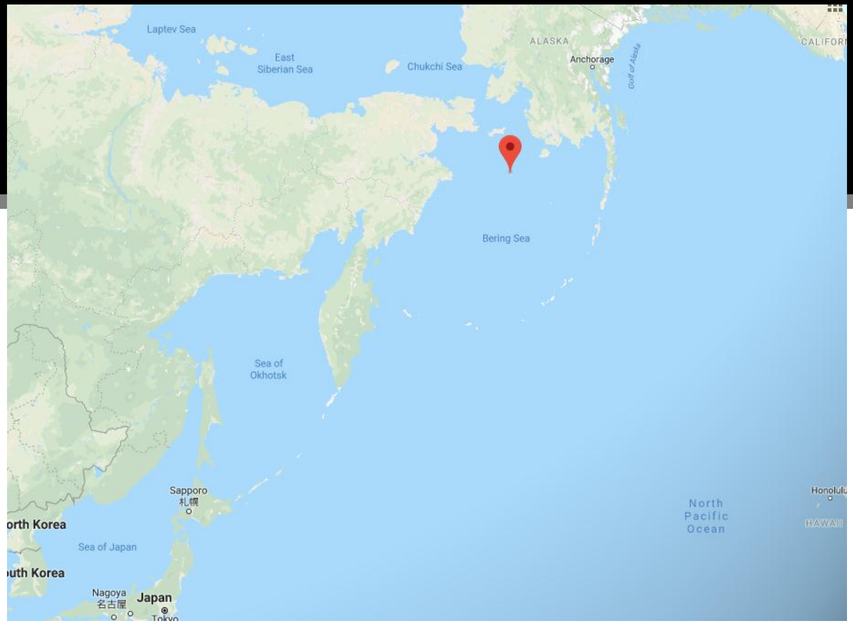
FBI: 300,000 reported internet crimes cost victims at least $1.4 billion in 2017

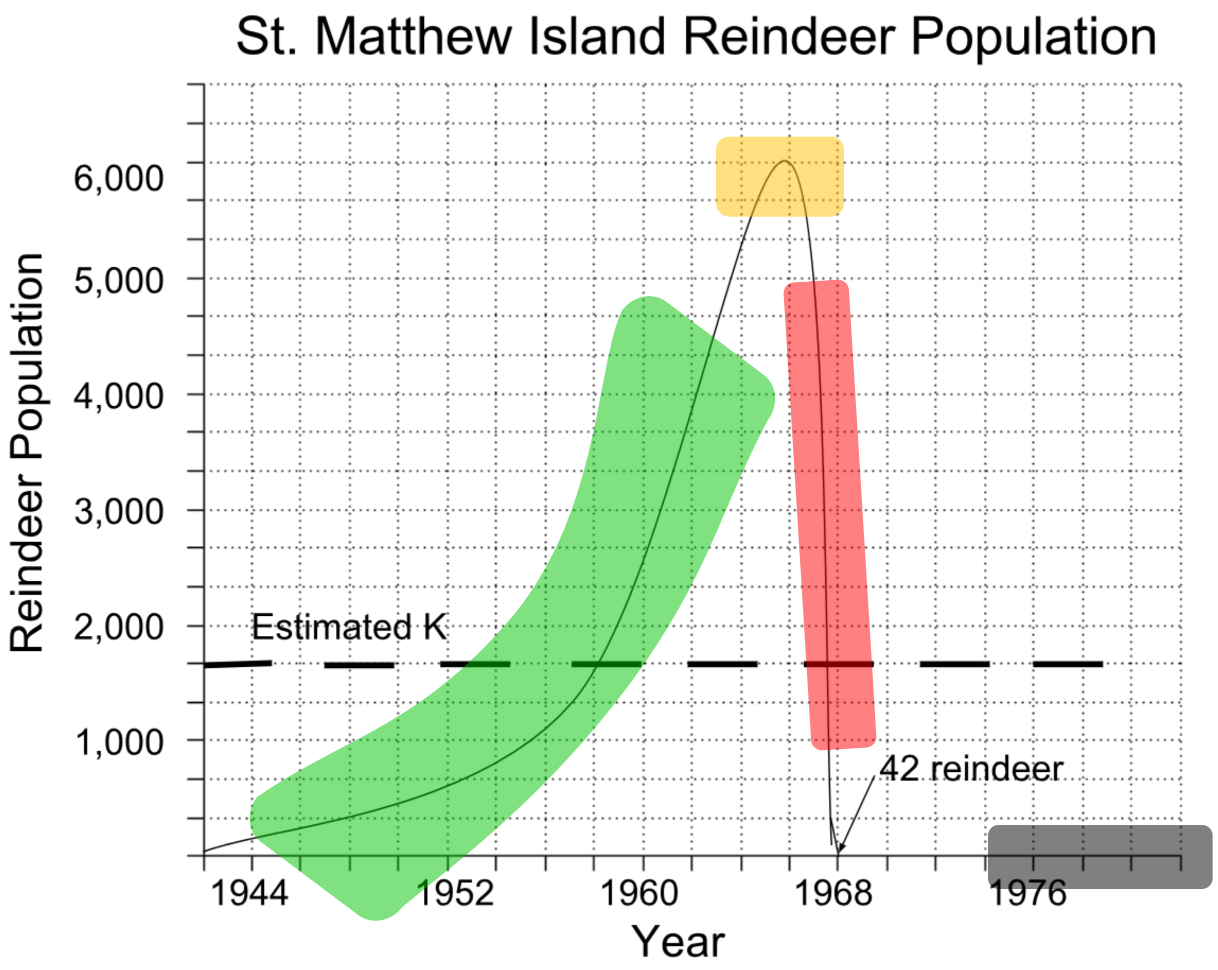
Canadian banks warn: Hackers might have stolen data from nearly 90,000 customers

Reindeer

Strolling Reindeer photo CC-SA 3.0 Unported by Nattfodd https://commons.wikimedia.org/wiki/File:20070818-0001-strolling_reindeer.jpg

Feast & famine

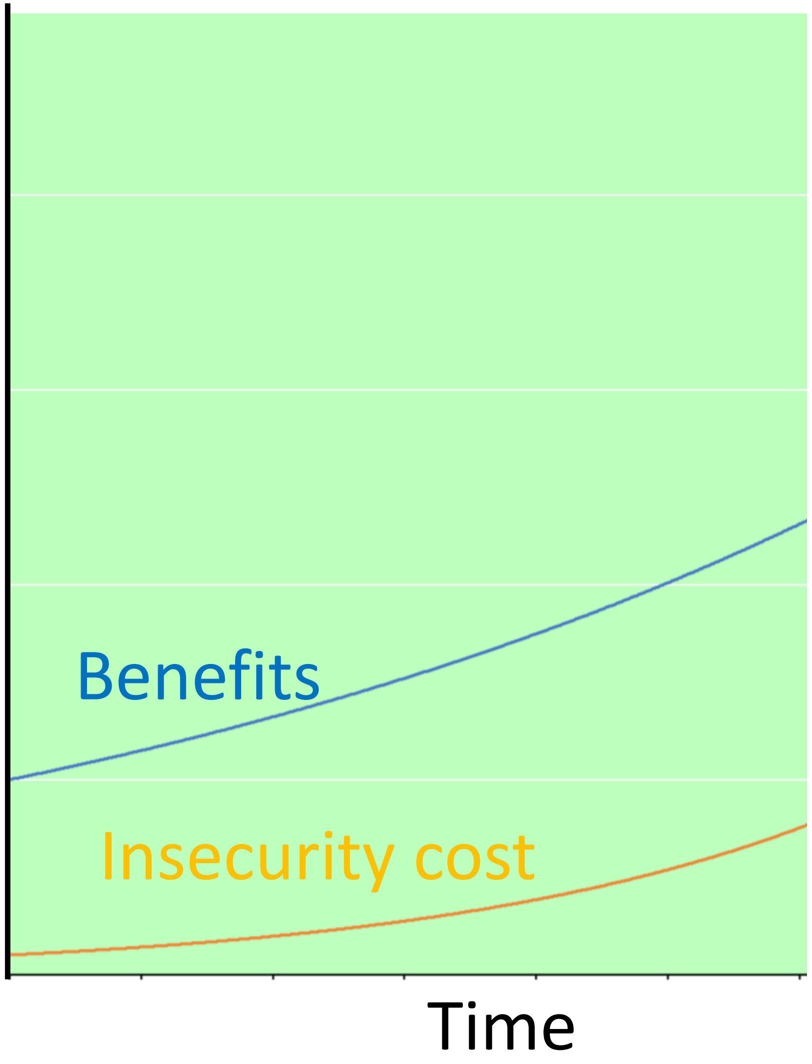




Distinct phases with different challenges / opportunities

Figure by Soudibert97, licensed under CC-SA 4.0. From https://en.wikipedia.org/wiki/File:St._Matthew_Island_Reindeer_Population.svg. 29 reindeer introduced in 1944, soared to 6000, crashed to 42 then died out.

Past

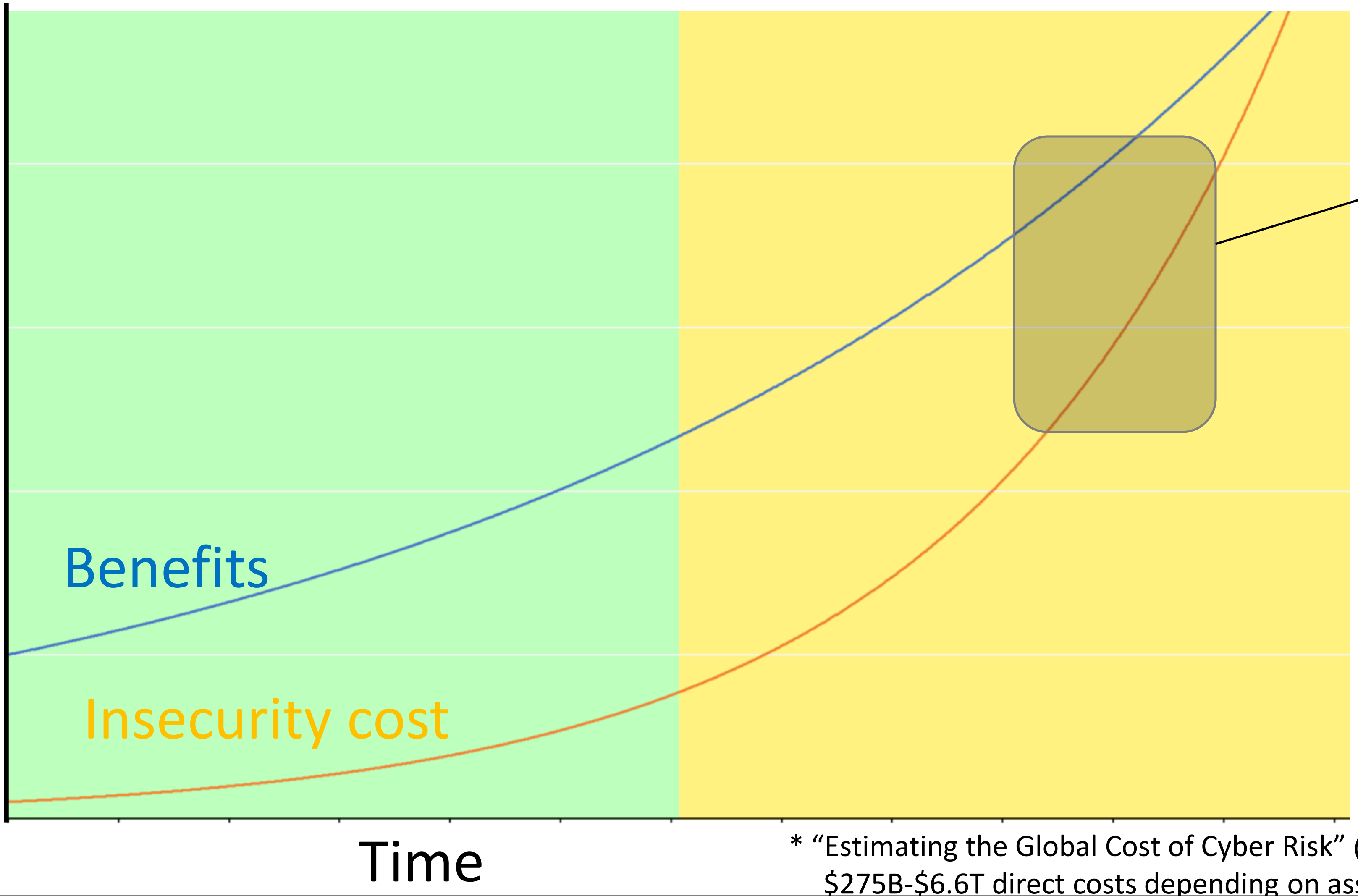


For many years, technology's benefits scaled exponentially

← Benefits are larger

← Costs of insecurity much smaller than benfits
economic importance of benefits outweighs risks

Present



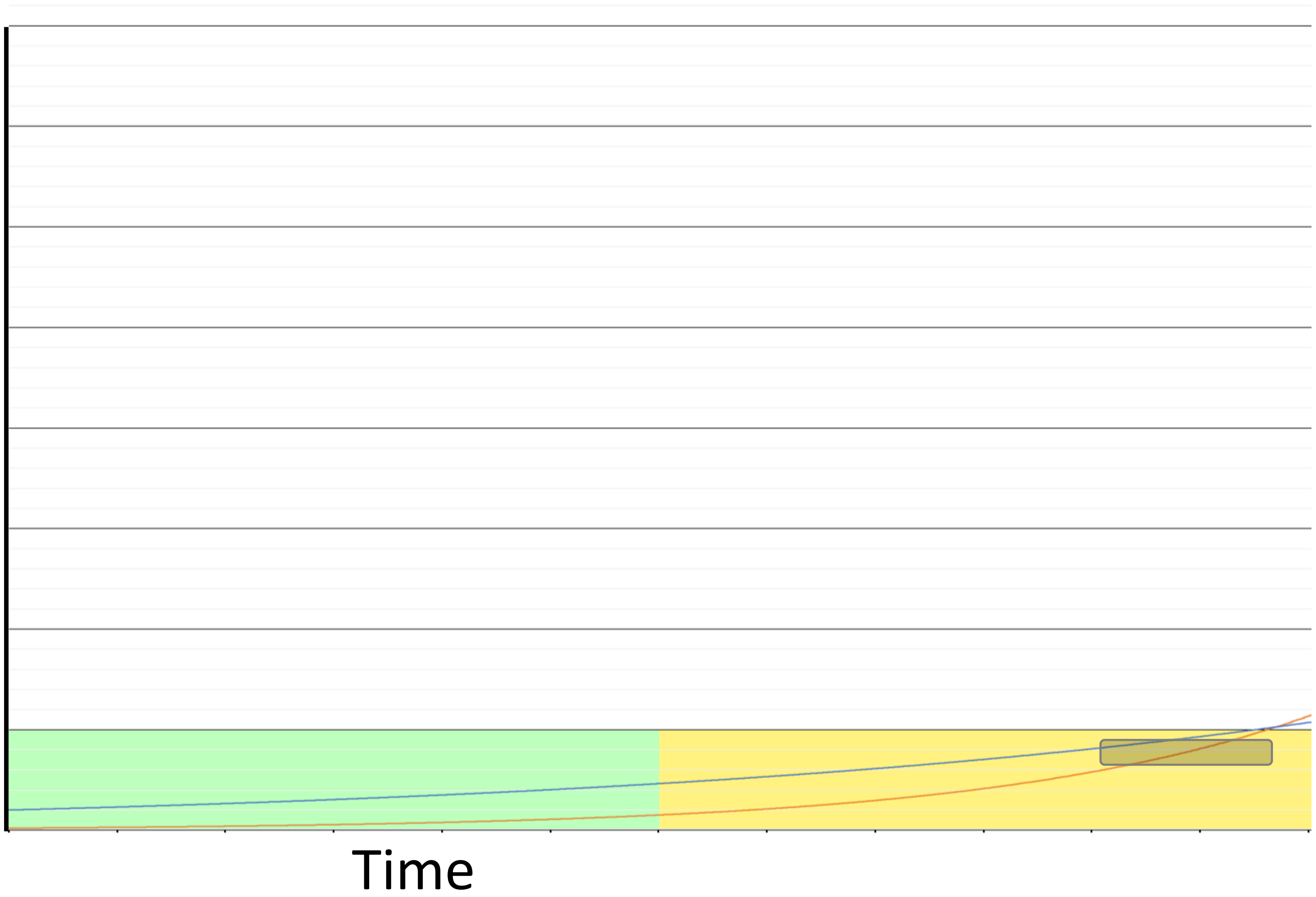
Insecurity $\$10^{11}$-$\$10^{13}$/year*

Insecurity cost YoY growth on par with growth in tech benefits

Worse for mature applications (e.g., benefit growth is slower)

* "Estimating the Global Cost of Cyber Risk" (https://www.rand.org/pubs/research_reports/RR2299.html): $275B-$6.6T direct costs depending on assumptions, and $799B-$22.5T incl. indirect costs.

The larger exponent dominates



The larger exponent dominates

Technology scales faster than traditional economy

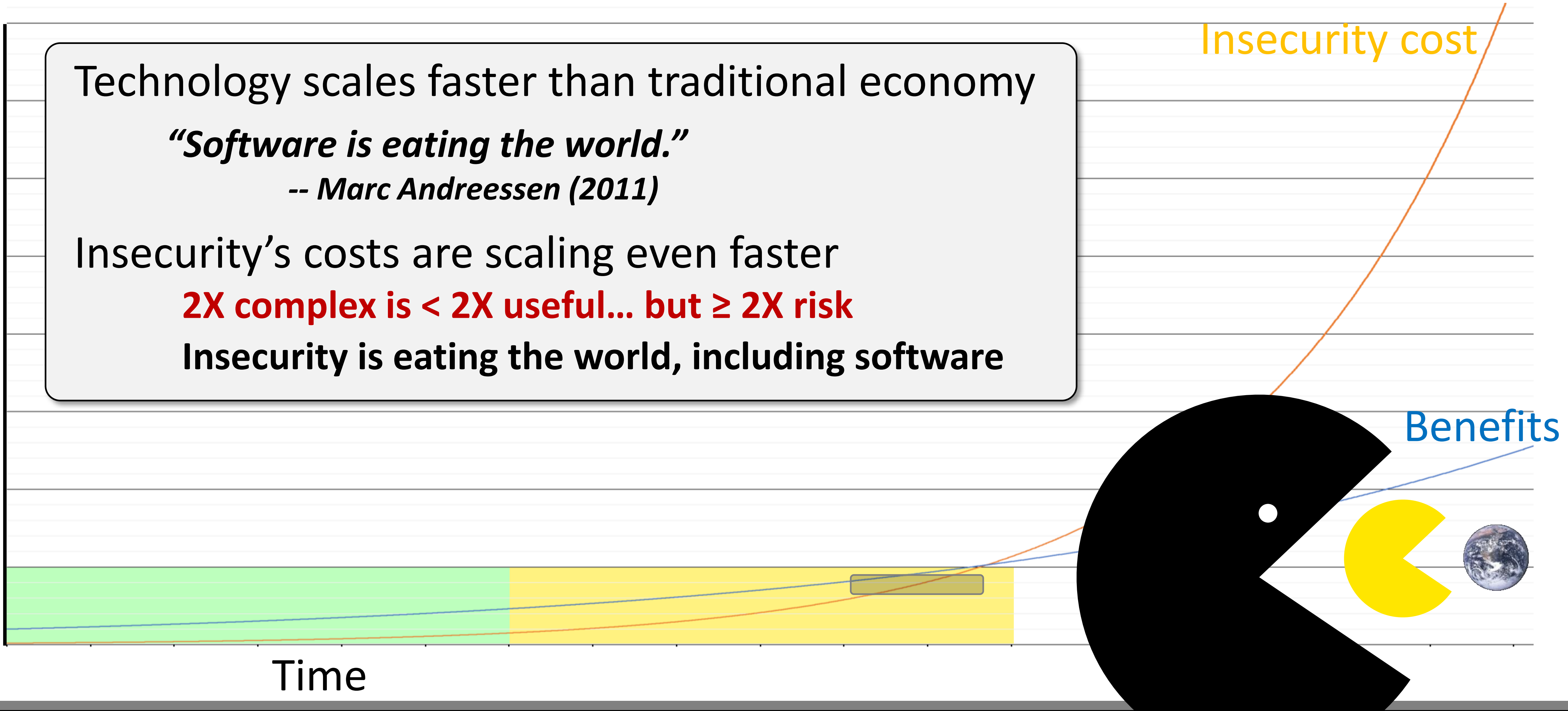
"Software is eating the world."

-- Marc Andreessen (2011)

Insecurity's costs are scaling even faster

2X complex is < 2X useful… but ≥ 2X risk

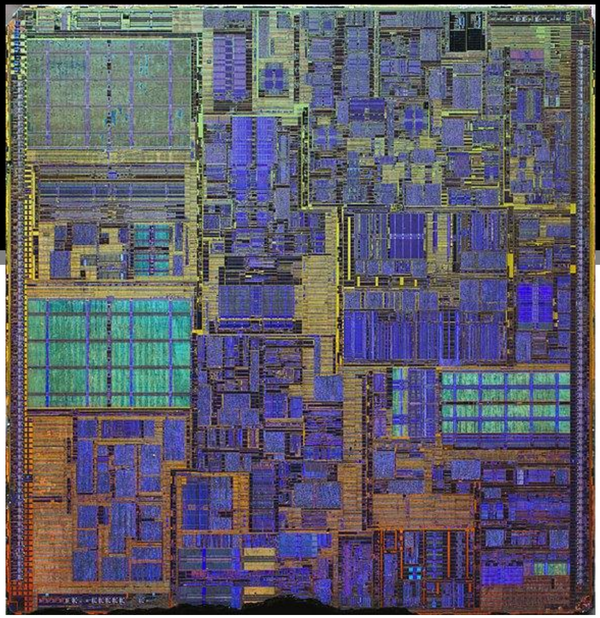
Insecurity is eating the world, including software



Spectre as a symptom:

Scaling & computer architectures

CPU performance



No more easy sources of performance scaling

- Memory latency is not improving much
- Clock rates maxed out: Pentium 4 reached 3.8 GHz in 2004

Single-thread speed gains require getting **more done per clock cycle**

- Reducing memory delays → Caches
- Working during delays → Speculative execution

Public domain image of Pentium 4 die by Ritzchens Fritz

Memory caches

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory

CPU
Sends address,
Receives data

Addr: 2A1C0700
Data: 9E C3 DA EE B7 D3..

Addr: 132E1340
Data: AC 99 17 8F 44 09..

Addr: 132E1340
Data: AC 99 17 8F 44 09..

MEMORY CACHE

h(addr) to map to cache set

Fast

Slow

Fast

Set	Addr	Cached Data ~64B
0	F0016280	B5 F5 80 21 E3 2C..
	31C6F4C0	9A DA 59 11 48 F2..
	339DD740	C7 D7 A0 86 67 18..
	614F8480	17 4C 59 B8 58 A7..
1	71685100	27 BD 5D 2E 84 29..
	132A4880	30 B2 8F 27 05 9C..
	2A1C0700	9E C3 DA EE B7 D9..
	C017E9C0	D1 76 16 54 51 5B..
2	311956C0	0A 55 47 82 86 4E..
	002D47C0	C4 15 4D 78 B5 C4..
	91507E80	60 D0 2C DD 78 14..
	55194040	DF 66 E9 D0 11 43..
3	9B27F8C0	84 A0 7F C7 4E BC..
	8E771100	3B 0B 20 0C DB 58..
	A001FB40	29 D9 F5 6A 72 50..
	132E1340	AC 99 17 8F 44 09..
4	6618E980	35 11 4A E0 2E F1..
	BA0CDB40	B0 FC 5A 20 D0 7F..
	89E92C00	1C 50 A4 F8 EB 6F..
	090F9C40	BB 71 ED 16 07 1F..

Address:
132E1340

Data:
AC 99 17 8F 44 09..

MAIN MEMORY

Big, slow
e.g. 16GB SDRAM

Reads change system state:
- Next read to newly-cached location is faster
- Next read to evicted location is slower

Spectre Variant 1

```
if (x < array1_size)
    y = array2[array1[x]*512];
```

Unsigned integer `x` comes from untrusted source (e.g. attacker)

Execution <u>without</u> speculation is safe

- CPU will not evaluate `array2[array1[x]*512]` unless `x < array1_size`

What about speculative execution?

Spectre Variant 1

```
if (x < array1_size)
   y = array2[array1[x]*512];
```

Attack setup:

- Train branch predictor to expect if() is true (e.g. call with `x` < `array1_size`)
- Evict `array1_size` and `array2[]`

Memory & Cache Status

`array1_size = 00000008`

`Array1[0..7]:01 01 01 01 01 01 01 01`

...lots of memory up to `array1[N]`...

then something secret: `09 F1 98 CC 90...`

```
array2[ 0*512]
array2[ 1*512]
array2[ 2*512]
array2[ 3*512]
array2[ 4*512]
array2[ 5*512]
array2[ 6*512]
array2[ 7*512]
array2[ 8*512]
array2[ 9*512]
array2[10*512]
array2[11*512]
   ...
```

Contents don't matter
only care about cache ***status***

Uncached | Cached

Spectre Variant 1

```
if (x < array1_size)
   y = array2[array1[x]*512];
```

Attacker calls victim code with x=N (where N>>8)

- Speculative exec while waiting for array1_size
  - Predict that if() is true
  - Read address (array1 base + x) w/ out-of-bounds x
  - Read returns secret byte = **09** [fast – in cache]
  - Request memory at (array2 base + **09***512)

Memory & Cache Status

array1_size = 00000008

Array1[0..7]:01 01 01 01 01 01 01 01

...lots of memory up to array1[N]...

then something secret: **09** F1 98 CC 90...

array2[0*512]
array2[1*512]
array2[2*512]
array2[3*512]
array2[4*512]
array2[5*512]
array2[6*512]
array2[7*512]
array2[8*512]
array2[9*512]
array2[10*512]
array2[11*512]
...

Contents don't matter
only care about cache ***status***

Uncached Cached

Spectre Variant 1

```
if (x < array1_size)
    y = array2[array1[x]*512];
```

Attacker calls victim code with x=N (where N>>8)

- Speculative exec while waiting for array1_size
  - Predict that if() is true
  - Read address (array1 base + x) w/ out-of-bounds x
  - Read returns secret byte = 09 [fast – in cache]
  - Request memory at (array2 base + 09*512)
  - Brings array2[09*512] into the cache
  - Realize if() is false: discard speculative work
- Finish operation & return to caller

Attacker times reads from array2[i*512]

- Read for i=09 is fast (cached), revealing secret byte

Memory & Cache Status

array1_size = 00000008

Array1[0..7]:01 01 01 01 01 01 01 01

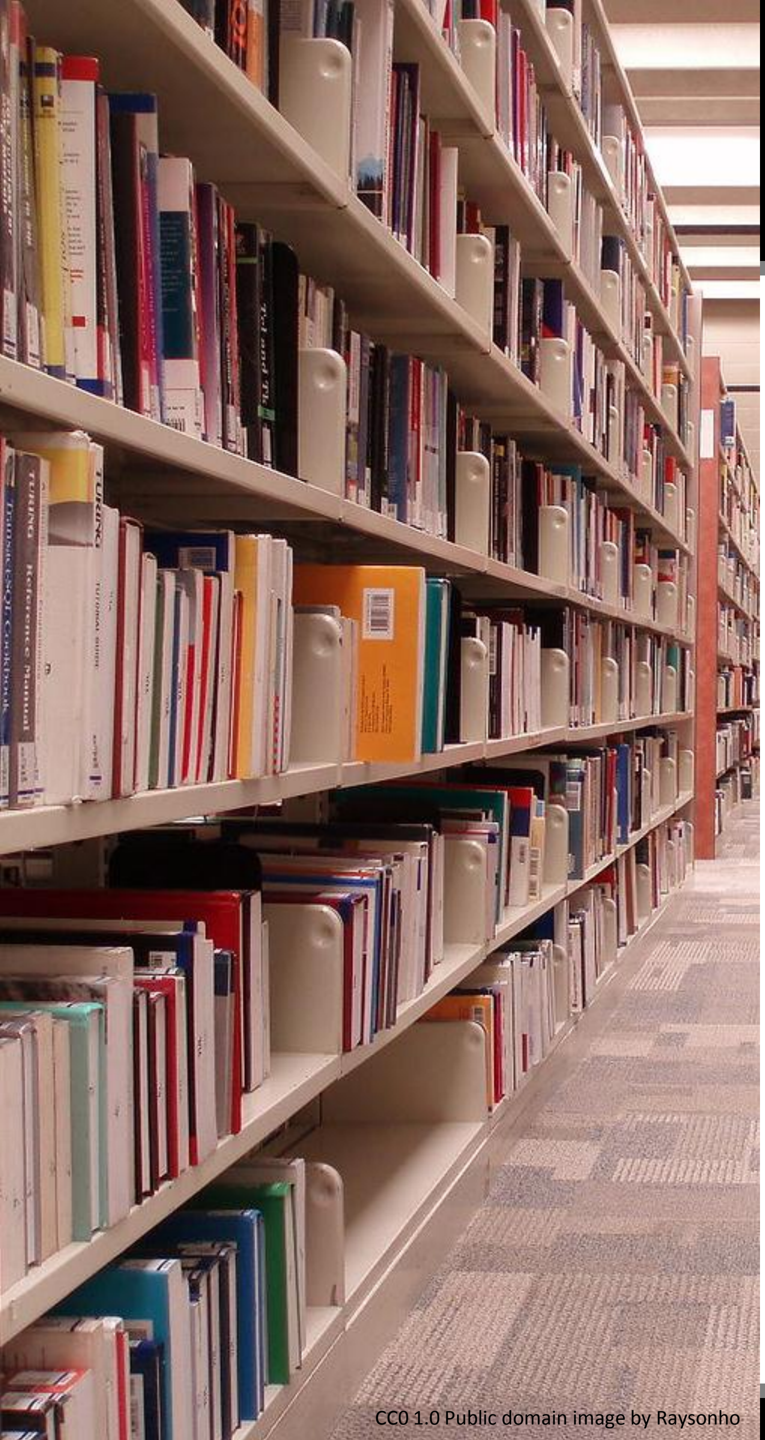
...lots of memory up to array1[N]...

then something secret: 09 F1 98 CC 90...

array2[0*512]
array2[1*512]
array2[2*512]
array2[3*512]
array2[4*512]
array2[5*512]
array2[6*512]
array2[7*512]
array2[8*512]
array2[9*512]
array2[10*512]
array2[11*512]
...

Contents don't matter
only care about cache ***status***

Uncached Cached



Analogy

"Bob the CPU" is a fast, naïve researcher with top-secret data

- Bob starts work immediately when a request arrives
- Checks authorization before committing (sending) results
- Mallory wants to steal a secret that Bob can access

Mallory asks questions that fail authentication:

"What color is a book whose title begins with the **1st** letter of the secret"

- Bob checks out a book, writes down the color… and deletes his work

"What color is a book whose title begins with the **2nd** letter of the secret"

- Bob checks out a book, writes down the color… and deletes his work

…

What can Mallory learn from list of books checked out by Bob?

- Hint: Mallory doesn't care about colors

"It's a NOT bug. Everything is working correctly."

- Branch predictor works as expected
- Speculative execution unwinds architectural state correctly
- Caches are working as designed
- **Vulnerable CPUs comply with architectural specs**

Spectre is a symptom

1 Architecture guarantees are not sufficient for security

No easy fix for Spectre [and Spectre isn't the only concern]

- Many variations (speculation scenarios, covert channels, security boundaries…)
- Mitigations conflict with desire to allow optimization flexibility

Spectre is a symptom

1 Architecture guarantees are not sufficient for security

2 Security involves trade-offs: Can't be fastest and safest

Performance goals conflict with security

Advance execution 'leading edge' as fast as possible

- Defer safety checks -> Meltdown

Optimizations make average case faster than worst case

- Caches, dead code removal, speculation, turbo clock rates, …
- -> Side channel attacks

[Many other examples, e.g. compilers, JIT…]

Spectre is a symptom

1 Architecture guarantees are not sufficient for security

2 Security involves trade-offs: Can't be fastest and safest

3 Architectures are based on incorrect assumptions

Assumption:

- Implementation = mathematical model
- Perfect hardware
- Perfect software

Reality

Side channels, covert channels, etc.

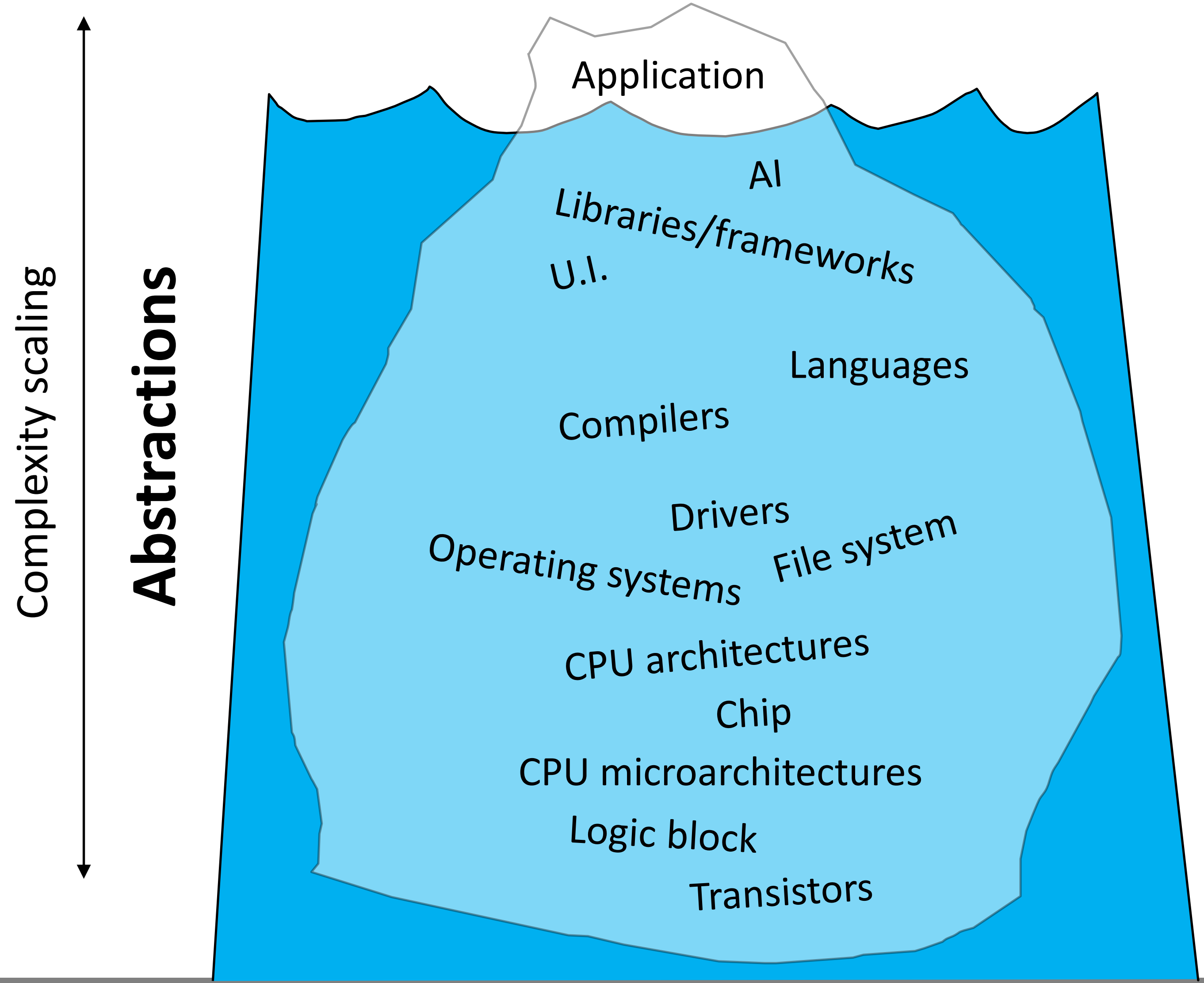
Bug density > 0

Bugs will occur, and should be survivable

… and Spectre mitigations make things much harder for software

Scaling functionality vs. scaling security

Scaling & abstraction: Technological problems



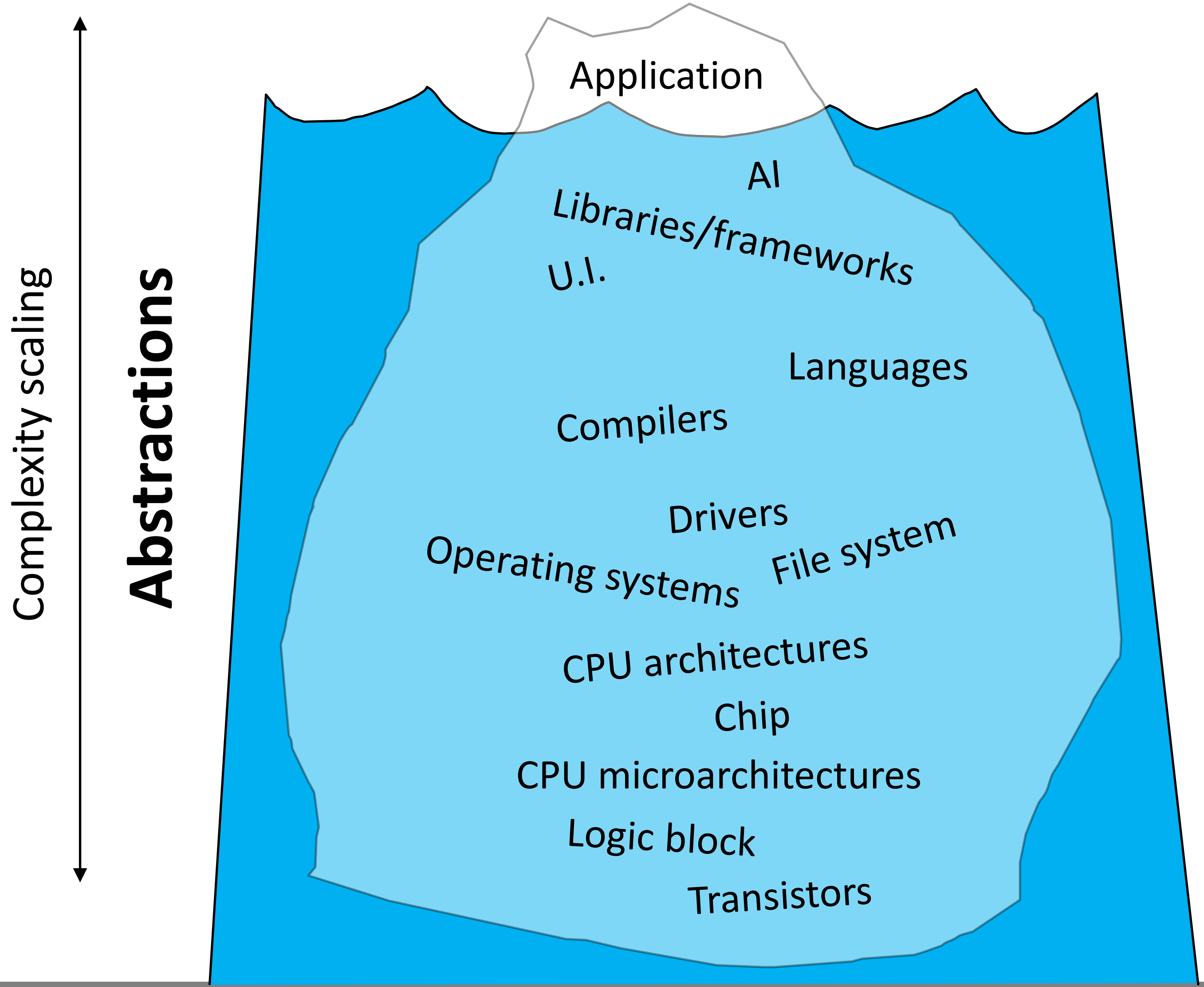
abstraction helps scaling

- Enables humans to build complex systems

abstraction is dangerous for security

- hidden complexity leads to wrong assumptions
- components aren't built with system awareness

Scaling & abstraction: Economic problems



abstraction creates an economic disincentive for security

Little economic benefit in security spending for one component if system security depends on other likely-to-be-exploitable component(s) that aren't being improved (e.g., outside the developer's control).

N-way deadlock:

- Developers cannot justify big investments in securing their components, unless all other developers invest first

Coping: Abstraction collapsing

One team (or company…) handles everything

- Designs entire security-critical portion – avoids the deadlock problem
- Typical deliverable: Logic block or entire chip
- Takes responsibility for outcome

May choose security <u>instead of</u> scalability

- Must minimize complexity (= limited use cases)
- Little or no logic reuse
- Requires team with broad skills

Can we have both complexity and security?

Temporal separation

Spatial separation

Formal methods, safer languages…

Redundancy?

Temporal separation

Separate security domains in time, with reset (zeroization) between tasks

Ancient idea:

- Reboot PC between tasks
- Store floppy disks in a safe when not in use
- Lost when internal hard drives arrived

Cloud computing business built on HW reuse:
Can (micro-)architectures reintroduce reliable temporal separation?

Temporal separation

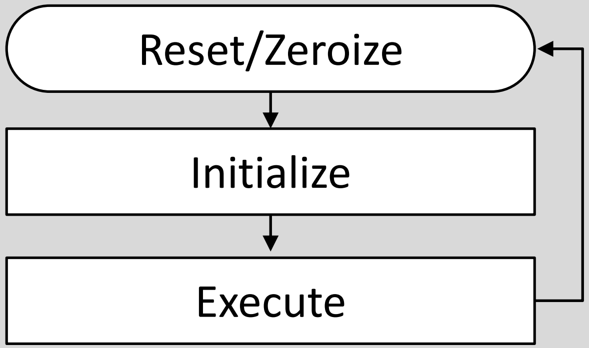
Separate security domains in time, with reset (zeroization) between tasks

Well suited to hardware:

- Reset is simple, already well understood
- Approach: zeroize almost all logic between operations

Scales well: Reset logic complexity doesn't scale with overall circuit complexity

Effective across many layers of abstraction



Need to be careful about…

State that survives reset (registers/RAM/E^2/flash/OTP)

Glitch attacks, SW damage to hardware

Side channels

Manufacturing tests

Peripherals

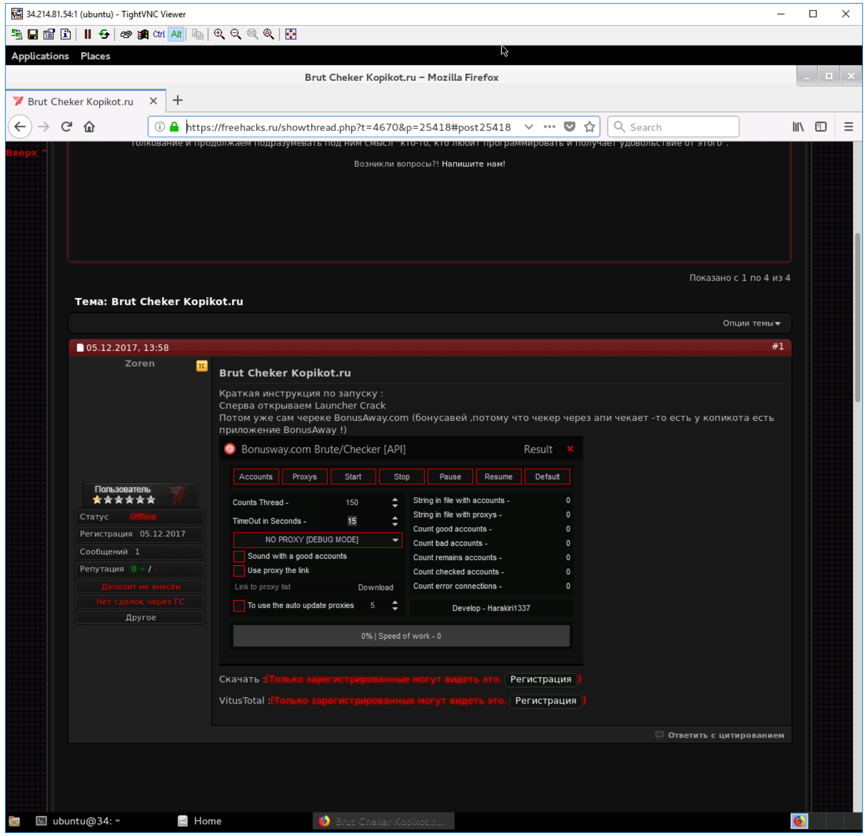
…

Spatial separation

Separate security domains in space, with limited interfaces between

Ancient idea:

- Different hardware for different tasks
  - Separate computers (classified vs. unclassified)
  - Burner phone
  - Separate authentication hardware (YubiKey, smart cards…)



Example: Browsing unsafe sites

Spatial separation

Separate security domains in space, with limited interfaces between

Intra-device separation

- Why do we use the same CPUs to approve wire transfers and play video games?
- Don’t build all cores the same: Faster + safer on chip
  - ‘Safer’ should also run safer software – doesn’t need to be backward compatible
    - Allows use of CHERI-style capabilities, security-centric memory management…
  - Like ARM’s big.LITTLE (but for security instead of power)
  - Like SGX (but with a separate CPU instead of sharing hardware)
- Can use main CPU for coordination if availability guarantees not required
  - Resource allocation: RAM…
  - Fast context switches: Interrupts, networking, file system
  - Scheduling: Assign execution leases for secure core

Spatial separation

Separate security domains in space, with limited interfaces between

Important research topic: Memory models

- What kinds of memory “should” the supported? How are they managed + exposed to software?

SRAM	fast, internal	small fixed size limits software flexibility
Cache	minimum SW impact	problematic side channels
Shared DRAM apertures	untrusted	needed to talk with other components
Encrypted RAM	simple, fast	leaks addresses (esp if shared) + vulnerable to replay
Encrypted RAM + anti-replay	addresses replay attacks	state for freshness guarantee tricky to manage well
Oblivious RAM	nice side channel properties	slow/complex

…

Static analysis, formal methods, safer languages

Enables automate insights into properties of complex logic

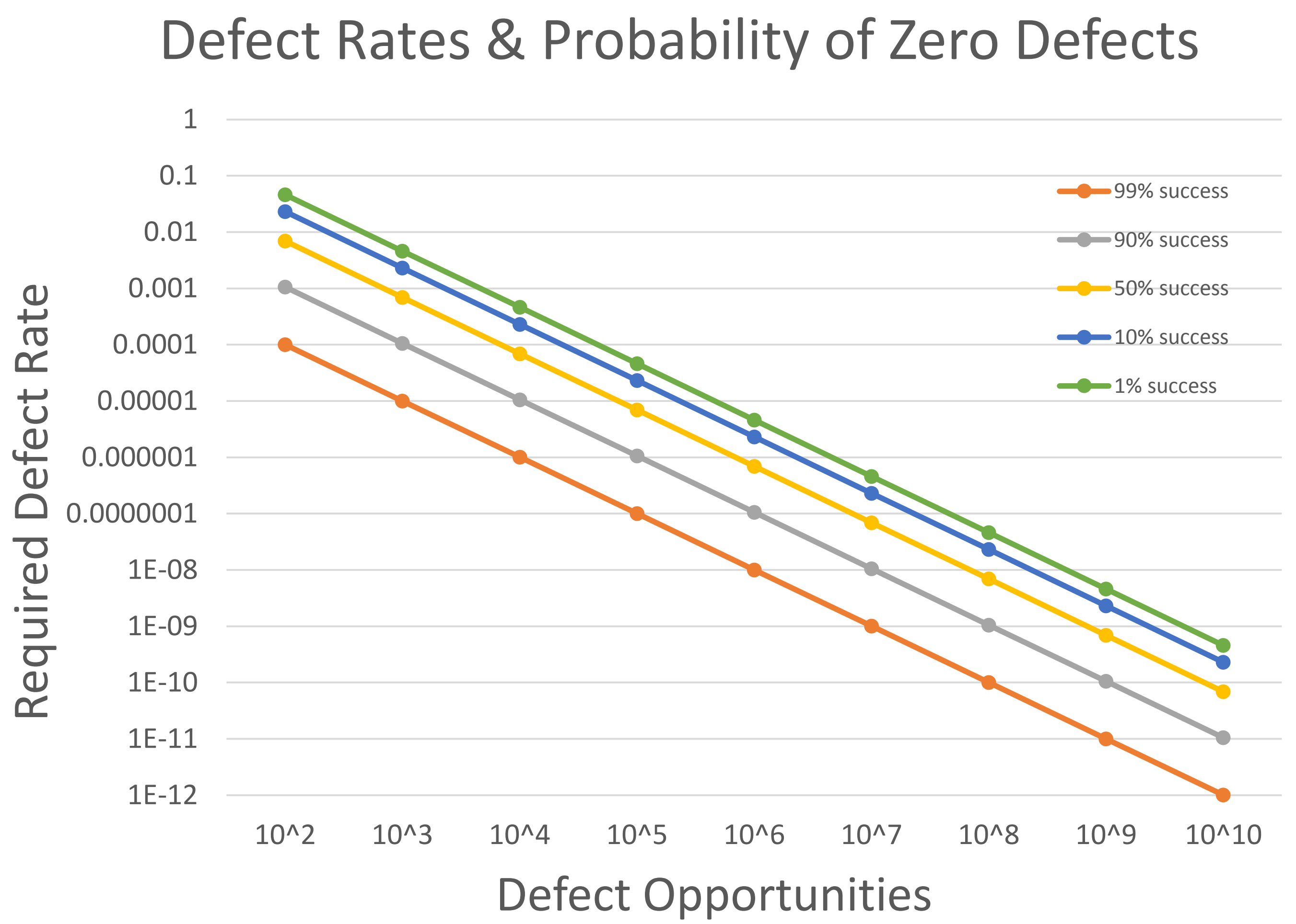
Slow progress – but still limited adoption

- seL4, better tools, safer languages like Rust…
- Interesting interplay with new hardware architectures (RISC-V, CHERI….)

Scaling concern: Constant factor improvement, exponentially growing problem

- In-scope and out-of-scope complexity both scale exponentially

Defect rates vs. success probabilities



Example:

10^{-4} defects/element

10^{7} critical elements

P(0 defects) $\approx 10^{-434}$

Trying harder won't work

Need (10^{6}+) reduction in defects

… to address *today's* situation

Redundancy

Redundancy is widely used in other industries

- Aviation (engines, components, pilots, pilot meals…)
- Structural engineering
- Signers on financial accounts
- Power supplies, fans, disk drives (RAID)…

Original Example:
10^{-4} defects/element
10^{7} critical elements
P(0 defects) $\approx 10^{-434}$

What if each element became a triplet that survives 1 failure?

P(uncorrected defect) per triplet
$\approx (3\times10^{-8})$

Revised Example:
3×10^{-8} uncorrected defects/triplet
10^{7} critical triplets
P(0 uncorrected defects) $\approx$ 0.74

- Redundant weak elements = additive strength (effort to break each)
- Redundant stronger elements = potential for exponential strength (probability of flaws in all)

Redundancy

Many research questions:

- Algorithms: Multi-party computation (MPC) in useful settings
- Hardware: How to apply at smaller granularity beyond redundant computers?
- Human: Are engineers willing to accept inefficiency to improve security?

Scaling will continue

	Traditional	IoT
Number of vendors	Few	1000's+
Vendor security expertise	World-class	Low/none
User attention per device to security	High	None
On-device security analysis tools	Advanced	None
Network-based security capabilities	Advanced	Limited/none
Can cause harm in physical world	No	Yes
Funding for security maintenance	Upgrade/sub	None
Typical operational life	<10 years	20-100 years

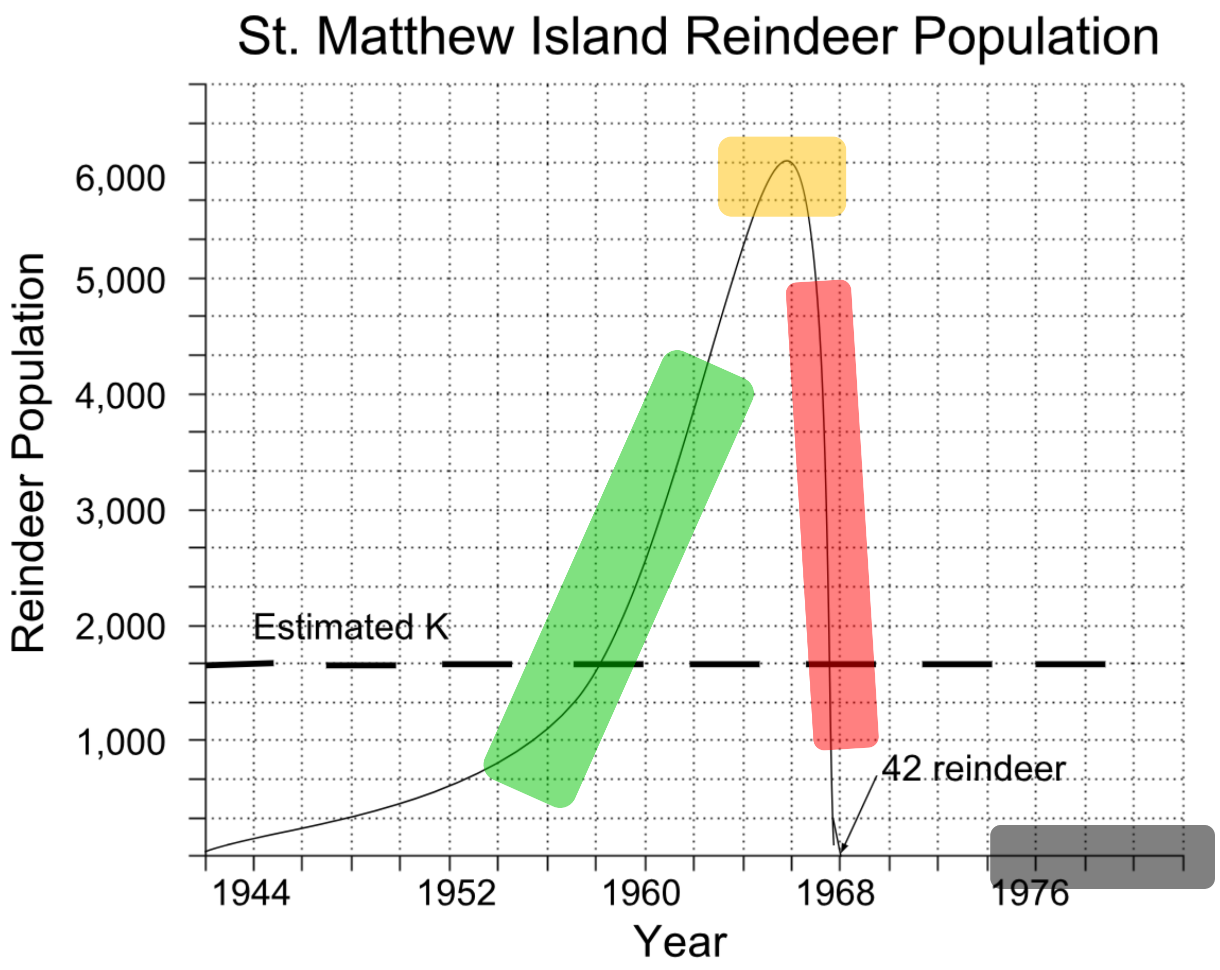
Machine learning -> complexity beyond what humans can create

Human challenges

During the exponential phase, we developed today's…

- Instincts
- Leaders
- Business models
- Dominant companies
- Regulations
- Practices
- Folklore
- Legacy designs
- Standards

These may not be suitable for the risks and opportunities ahead



Conclusion

Hardware is the foundation: Computations generally no more secure than the HW

- Need to revisit tradeoffs between security + performance
- Need reputation + strength across silicon + software + devices
- Opportunities favor integrated companies (Samsung, Apple…) + small innovators

Combination of technical + leadership innovation

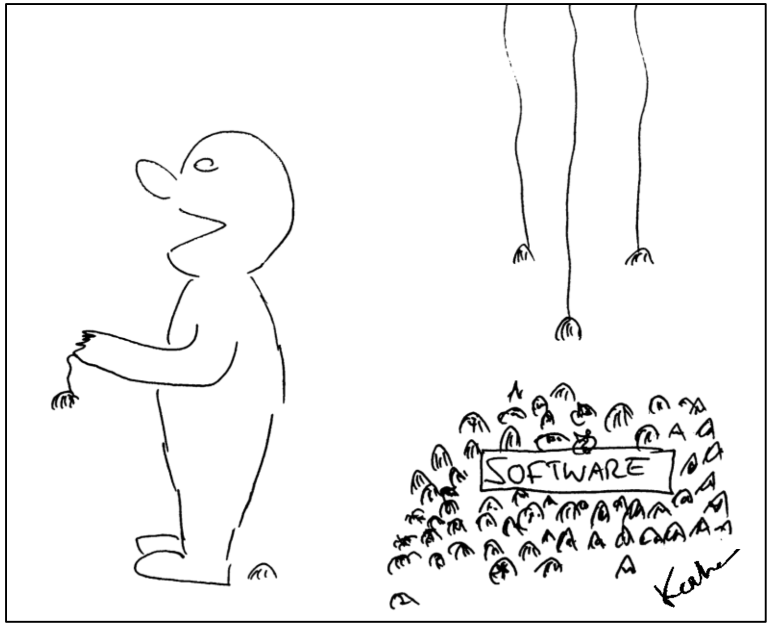
- Can't rely on approaches that worked in the past

Urgent problem

- Long deployment runway (research -> product -> deployment -> retire legacy)
- Insecurity costs are growing exponentially

Thank you!

My email: paul@paulkocher.com



I found the security bug!