# Linux and NIST SP 800-171 or: How I Learned To Stop Worrying and Love Compliance Enforcement

Craig Lewis

James Ralston

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA  15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM18-0156

# Agenda

Challenges and Background

- What is the problem?
- How can we solve it?

Technology high-level overview

- What Puppet can do
- What Splunk can do

Examples of NIST SP 800-171 enforcement

- 3.4.2, 3.4.6, 3.4.9
- visualizing conformance with Splunk

Linux and NIST SP 800-171

# Challenges and background

# Challenges and background

About us:

- A team of 4 people that manages:
  - 500+ Linux servers
  - Many mission-critical enterprise services
- There’s always more to do
- We try to do the #devops thing, but we have lots of silos
- We try hard to leverage existing technologies
- We need administrator amplification
  - We don’t have the time or manpower for artisanal hand-crafted hosts or services
  - “Cattle not pets”

# Challenges and background

The goal: comply with NIST SP 800-171

- But how?

Issues:

- NIST SP 800-171 dialog is Windows user endpoint-centric
- Few automated compliance enforces/checkers for Linux
- We don’t have just Windows hosts
- Linux isn’t typically discussed

# Challenges and background

Potential Solutions?

Apply configuration changes manually or at build?

- Doesn’t scale
- How to detect configuration drift?

Is there a better way?

# Challenges and background

How about a configuration management system (CMS)?

- The point of a CMS is to *apply* and *enforce* a desired state

Disadvantages:

- Initial setup is more complex

Advantages?

- Can scale to many hosts
- Should be able to detect (and correct!) drift

# Challenges and background

What CMS to use?

Should you write your own?

- Spoiler alert: *NO*.

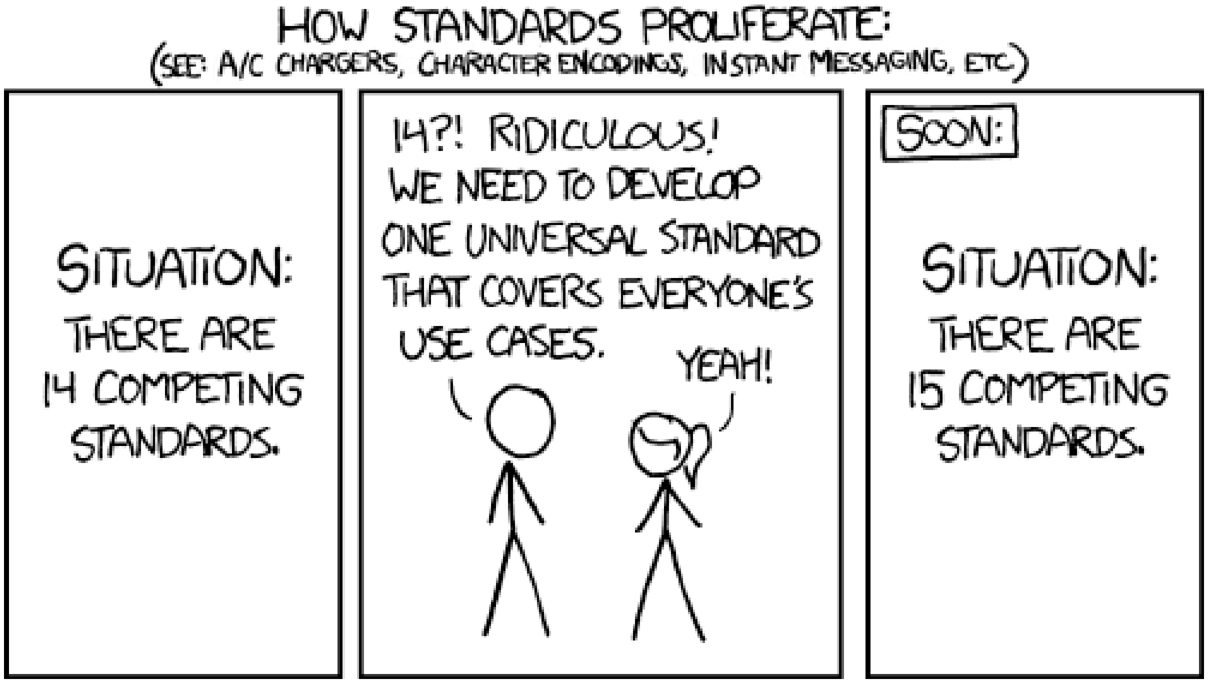

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Image credit: xkcd

# Challenges and background

So, which one?

Some potential options:

- Ansible
- Chef
- Otter
- Puppet
- Salt

“The nice thing about standards is that you have so many to choose from.” – Andrew S. Tanenbaum

# Challenges and background

We chose Puppet. Why?

- We needed to replace our aging, custom CMS
- At the time (early 2011), Puppet seemed like the most mature solution
- We didn’t agonize over the choice… and you shouldn’t, either

Linux and NIST SP 800-171

# Puppet and Splunk overview

# Puppet overview

Puppet uses a client/server model

- A Puppet agent runs on each client
- The agent periodically enforces that the actual state of the system matches its intended state
- The agent obtains the client's intended state from the Puppet server

# Puppet overview

How do you describe the intended state of a client?

Using a declarative language

- a package that should be installed
- the contents of a configuration file
- a service that should be running

Descriptions are called *resources*.

- Resources can be ordered via dependencies
- Puppet will respect those dependencies when making changes to the system

# Puppet overview

A *class* contains one or more resource declarations

A *module* is a collection of classes, data, files, templates, and so forth, laid out in a specific directory structure

The Puppet Forge contains thousands of freely-available Puppet modules

- So don’t reinvent the wheel!

The Puppet server knows which modules should be applied to which clients

# Puppet overview

What is the update process?

- The client agent gathers facts about the system
- The agent gives those facts to the Puppet server
- The server takes the facts, the list of modules the client should use, and other configuration data, and produces a *catalog*
- The catalog describes the desired state of the system
- The server sends the catalog to the agent
- The agent applies whatever actions are necessary to ensure the state of the client matches the catalog
- The agent sends a report of its actions to the server

# Splunk overview

What is Splunk?

- A tool for analyzing and visualizing machine data
- Can consume almost any type of data you can throw at it
  - System logs, audit logs, server logs, network logs
- Helps address the paradox of machine data
  - Machine data itself is rarely valuable…
  - …but the derivative information is

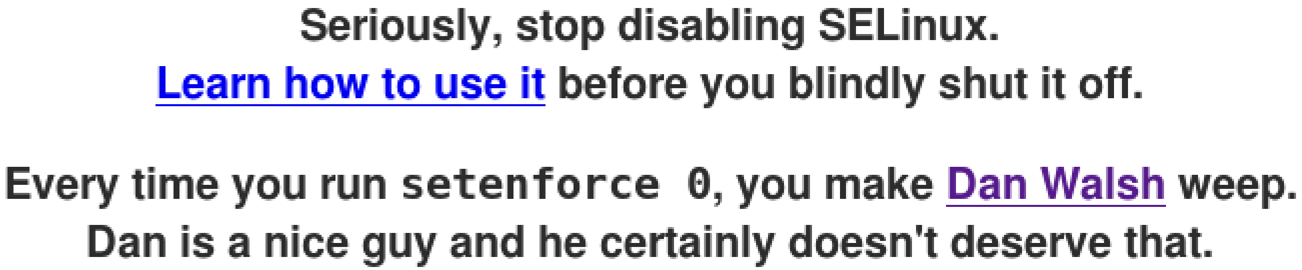
Linux and NIST SP 800-171

# Examples of NIST SP 800-171 enforcement

# NIST SP 800-171 requirement 3.4.2

3.4.2 states: “Establish and enforce security configuration settings”

- An example of this is SELinux.
- Developer attitude toward SELinux: “that NSA security thing that breaks my applications, so I turn it off.”
- IT attitude:



Seriously, stop disabling SELinux.
Learn how to use it before you blindly shut it off.

Every time you run `setenforce 0`, you make Dan Walsh weep.
Dan is a nice guy and he certainly doesn't deserve that.

https://stopdisablingselinux.com/

# NIST SP 800-171 requirement 3.4.2

Problem: we’re IT, so no one listens to us.

- Solution: use Puppet to ensure that SELinux runs in enforcing mode.
- When users realize they can't “solve” their SELinux problems by disabling SELinux, they ~~complain~~ come to us.

# NIST SP 800-171 requirement 3.4.2

```
class selinux (
  Enum['enforcing', 'permissive'] $selinux_mode = 'enforcing',
) {

   file { '/etc/selinux/config':
    ensure  => file,
    owner   => root,
    group   => root,
    mode    => '0644',
    content => "SELINUX=${selinux_mode}\nSELINUXTYPE=targeted\n",
  }

   if $facts['os']['selinux']['current_mode'] != $selinux_mode {
    exec { "setenforce ${selinux_mode}":
      path => ['/bin', '/usr/bin', '/usr/sbin', '/sbin'],
    }
  }

}
```

# NIST SP 800-171 requirement 3.4.6

3.4.6 states: “Employ the principle of least functionality by configuring the information system to provide only essential capabilities.”

- Firewall policy contributes to 3.4.6:
  - Permit authorized connections
  - Deny all other connections
- Problem: define “authorized”
  - How do you know whether a particular firewall allowance is intended/authorized to be there?

# NIST SP 800-171 requirement 3.4.6

Our approach:

- A firewall allowance is authorized if an included Puppet module contributes the rule.
- If an allowance isn't authorized, Puppet will remove it
- How to implement this?
  - Use the firewall module from the Puppet Forge
  - https://forge.puppet.com/puppetlabs/firewall

# NIST SP 800-171 requirement 3.4.6

```
class iptables {
  firewallchain { 'INPUT:filter:IPv4':
    purge  => true,
    policy => 'drop',
  }
}

class openssh::server {
  firewall { '500 IPv4 permit incoming ssh connections':
    provider  => iptables,
    proto     => tcp,
    dport     => 22,
    tcp_flags => 'FIN,SYN,RST,ACK SYN',
    action    => 'accept',
  }
}
```

# NIST SP 800-171 requirement 3.4.9

3.4.9 states: “Control and monitor user-installed software.”

- Just a wee bit Windows centric
- Monitoring individual packages is a challenge of scale

```
$ rpm -qa | wc -l
1478
```

- Our approach: use Splunk to enumerate and report

# NIST SP 800-171 requirement 3.4.9

The nuts and bolts:

- Write a Splunk input script that enumerates installed packages and sends that machine data to Splunk
- Use Puppet to ensure:
  - All hosts have the Splunk forwarder installed
  - All hosts have the Splunk input script installed

# NIST SP 800-171 requirement 3.4.9

Forwarder script:

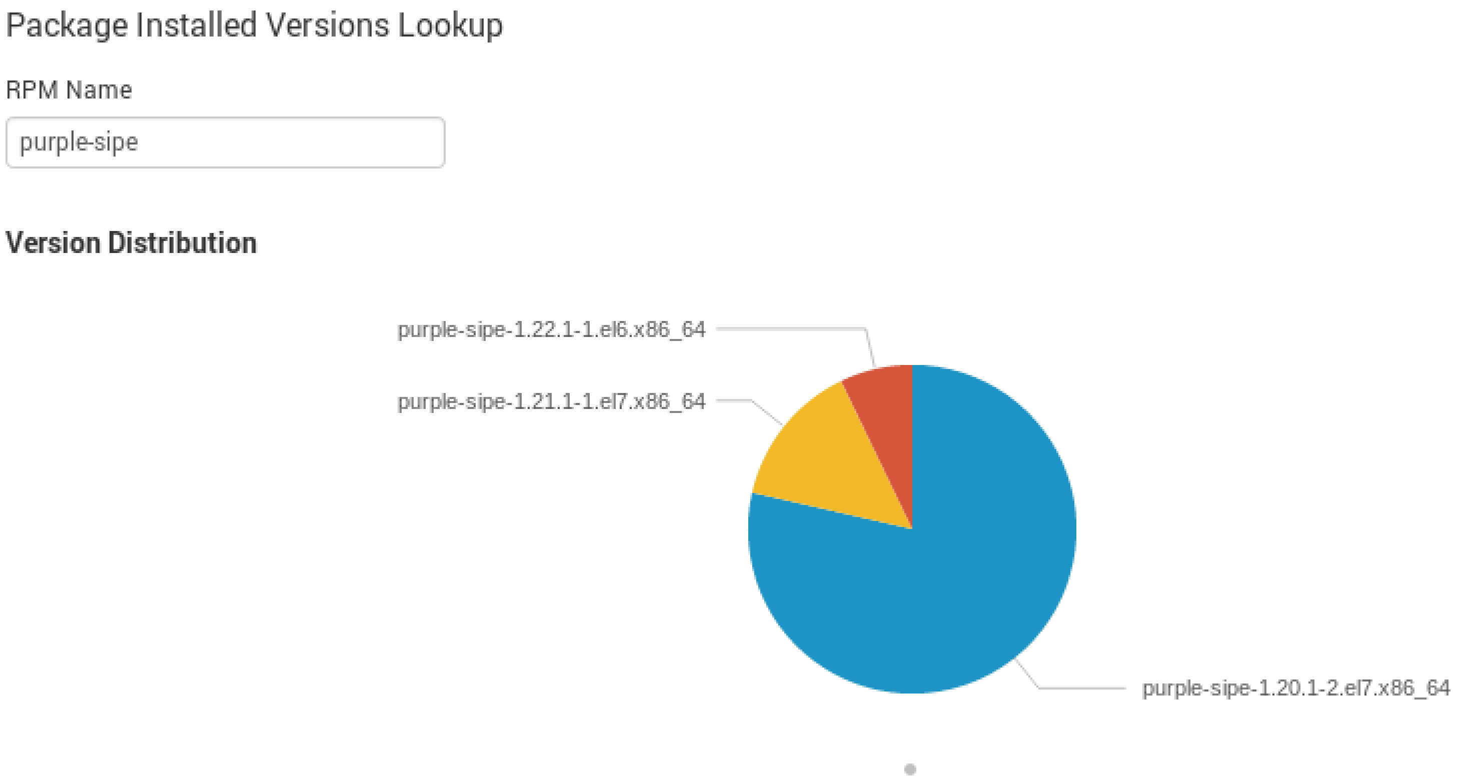
```
#! /bin/sh
# Yes, we could do it in one line. But readability is better.
TIMESTAMP=$(date "+%b %d %T")
FORMAT="rpmname=%{name},rpmvers=%{version}-%{release}.%{arch}"
rpm -qa --qf "${TIMESTAMP} ${FORMAT}\n"
```

Forwarder script configuration:

```
[script://$SPLUNK_HOME/bin/scripts/rpmInventory.sh]
interval = 1800
sourcetype = rpm_inventory
```

# NIST SP 800-171 requirement 3.4.9

Visualization:



Package Installed Versions Lookup

RPM Name

purple-sipe

**Version Distribution**

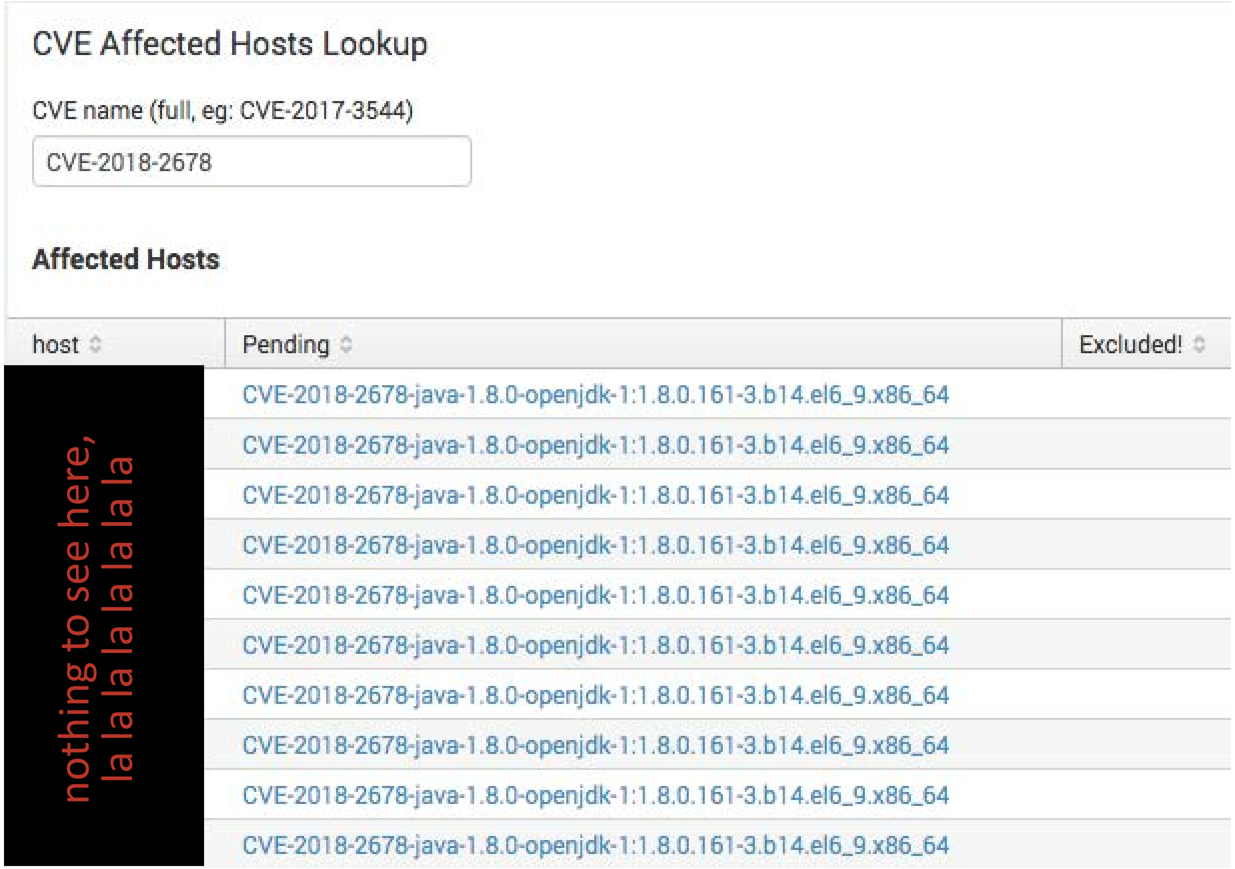
purple-sipe-1.22.1-1.el6.x86_64
purple-sipe-1.21.1-1.el7.x86_64
purple-sipe-1.20.1-2.el7.x86_64

# Other Puppet/Splunk uses

- By leveraging this simple workflow:
  - Use Puppet to install a script to log the data you want
  - Use Splunk to collect that data and present it
- It is possible to speak to other NIST SP 800-171 requirements:
  - 3.11 Risk assessment
  - 3.12 Security assessment strategy
- For example, the output of this command…
  - **`yum updateinfo list cves`**
- When aggregated into Splunk…

# Other Puppet/Splunk uses

…can tell you whether you missed applying security updates:



CVE Affected Hosts Lookup

CVE name (full, eg: CVE-2017-3544)

CVE-2018-2678

**Affected Hosts**

| host | Pending | Excluded! |
|---|---|---|
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |
| [redacted] | CVE-2018-2678-java-1.8.0-openjdk-1:1.8.0.161-3.b14.el6_9.x86_64 | |

nothing to see here,
la la la la la la la

Linux and NIST SP 800-171

# Conclusions

# Conclusions

- Conformance involves dull, repetitive tasks
  - Computers are pretty good at performing dull, repetitive tasks
  - Phrased differently: *your* time is order of magnitudes more valuable than a *machine’s* time.
  - Therefore, leverage tools to achieve conformance more efficiently

# Conclusions

- *What* tool you use isn't important, as long as you use it
  - Incremental improvement is better than no improvement

Spectre Server
@sadserver

optimisim in 2018 is hoping that the interface for launching the missles is better than the one for sending the alerts.

8:10 AM - 16 Jan 2018

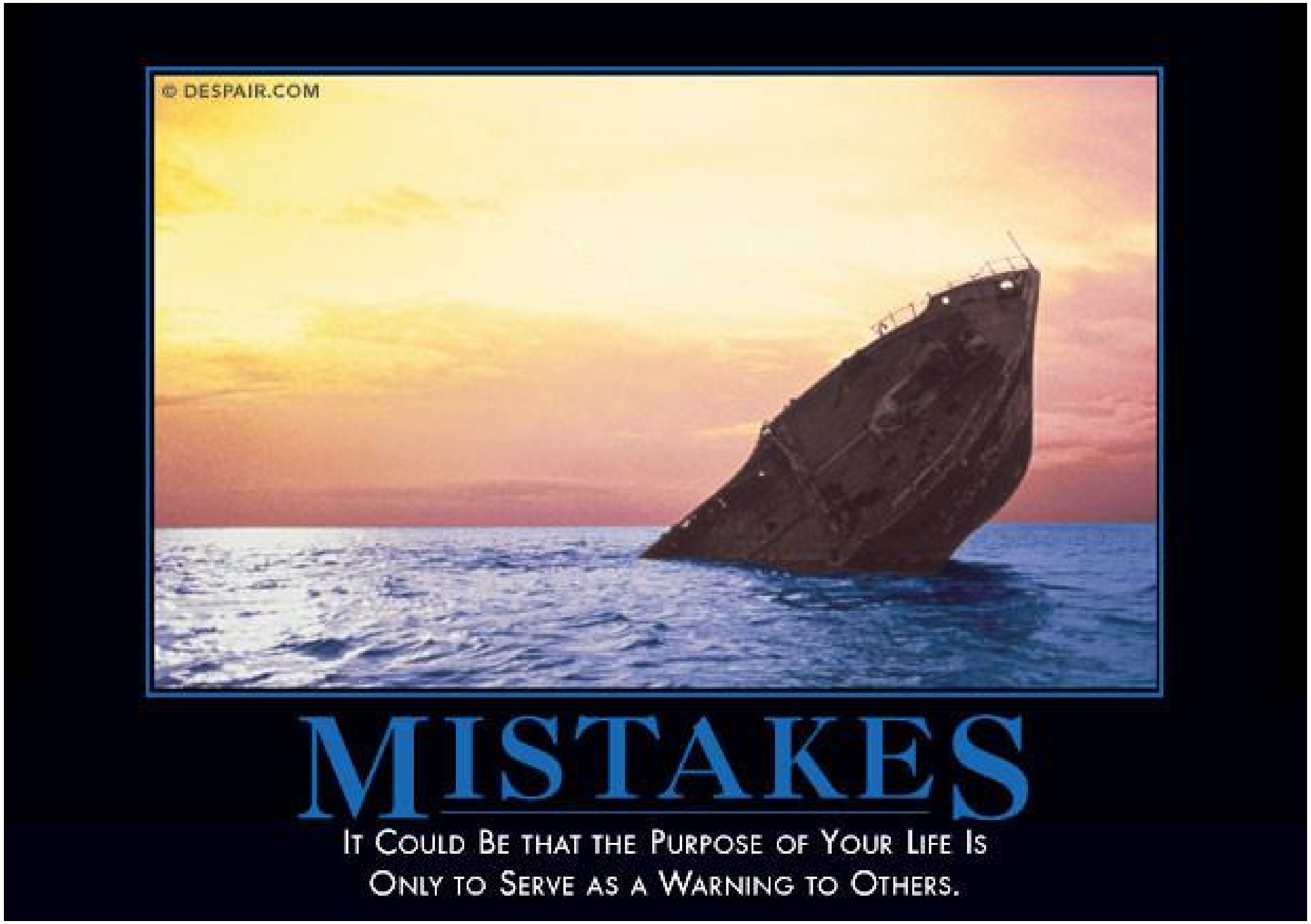
968 Retweets 1,627 Likes

# Conclusions

- You may not receive much recognition for employing best security practices and being good at conformance…

# Conclusions

- …but you’ll receive plenty of recognition if you’re bad at it.


© DESPAIR.COM
MISTAKES
It Could Be that the Purpose of Your Life Is
Only to Serve as a Warning to Others.

# Resources and links

- Puppet and the Puppet Forge:
  - https://www.puppet.com/
  - https://forge.puppet.com/
- Splunk:
  - https://www.splunk.com/
- Open Source Puppet modules for NIST, STIG, et. al. compliance:
  - https://simp-project.com/

Linux and NIST SP 800-171

# Questions?