Public Sector
From Stories to Getting Started

Presented by
Gabe Alford
Member of Technical Staff, Office of the Chief Technologist, Red Hat U.S. Public Sector

Shawn Wells
Chief Security Strategist, Red Hat U.S. Public Sector
shawn@redhat.com || +1-443-534-0130
- Member of “Kids First”, discovering genetic causes for childhood cancer and structural birth defects [0]

- 70 petabytes of genetic sequence and analysis data

- Data, drawn from samples donated by patients, shared with researchers to discover underlying causes of diseases and potential treatments

Protecting Patient Data

- Health Insurance Portability and Accountability Act (HIPAA)
  - United States Public Law 104-191 [0]
  - Requires adoption of security standards for “Electronically Protected Health Information” or EPHI
  - The EPHI created, received, maintained, or transmitted must be protected against reasonably anticipated threats, hazards, and impermissible uses [1]

- Federal Information Processing Standard 140 (FIPS 140)
  - United States Public Law 104-106 [2]
  - Defines cryptography standards and review process
  - Unvalidated crypto considered plaintext, usage not permissible [3]
Modules are Validated, not Products

Commonly Certified Modules taken through FIPS validation process by Red Hat

- OpenSSL
- Network Security Services (NSS)
- Linux kernel Crypto API
- GnuTLS
- OpenSSH (Client and Server)
- Libreswan (IPSec)

Search for certified FIPS products and modules at the following url:

Handling FIPS in Code or Builds

Example: Build Targets

```
cmake -DENABLE_OPENSSL=ON
```

Example: Configuration File

```
$ sudo grep -i fips /etc/myproject/config.conf
cipherSuite = <list_of_FIPS_ciphers>
```
Handling FIPS in Code or Builds

Example: python code

```python
options = read_conf_file_options(“/etc/myproject/project.conf”)
my_ciphers = options[“cipherSuite”]
# Now... do something with these ciphers
```
Handling FIPS in Code or Builds

Example: Use Crypto policies in Fedora and Red Hat Enterprise Linux

```
$ sudo head -4 /etc/crypto-policies/back-ends/openssh.config

Ciphers aes256-gcm@openssh.com,aes256-ctr,aes256-cbc,aes128-gcm@openssh.com,aes128-ctr,aes128-cbc
MACs hmac-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha2-256,hmac-sha1,hmac-sha2-512
GSSAPIKeyExchange no
```

Example: sshd Configuration File

```
$ sudo head -2 /etc/ssh/ssh_config.d/05-redhat.conf

# Follow system-wide Crypto Policy, if defined:
Include /etc/crypto-policies/back-ends/openssh.config
```

crypto-policy project: https://gitlab.com/redhat-crypto/fedora-crypto-policies
- Over two million passengers per day, averaging 8,000 to 9,000 simultaneous airplanes in air, rely on the Traffic Flow Management systems to ensure safe, on-time flights [0]

- After porting 1.5 million lines of code from legacy program, FAA replaced 1000 systems including 700 workstations

- Cost per system reduced from $25,000 to $3,000

- 30% more operational efficiency, 50% less cost

NIST SP 800-53
The source of your Government security controls

- 4th revision
- ~1,500 controls
  - Not all controls are technical
- Many broken down with enhancements
  - More like ~1,700
- Agency-specific overlays
- Getting us closer to 7,000 data points to consider
| A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1 | I1 | J1 | K1 | L1 | M1 | N1 | O1 | P1 | Q1 | R1 | S1 | T1 | U1 | V1 | W1 | X1 | Y1 | Z1 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC | NC |

**Legend:**
- **NC:** Not Applicable
- **X:** Not Applicable
- **C:** Compliance
- **S:** Security
- **M:** Management
- **O:** Operations
- **R:** Risk

**Notes:**
- [Notes for each cell can be added here, describing the specific requirements and implications for each category.]

**Security Controls Traceability Matrix - Down-selected 20160606**

**Purpose:**
- To track and document the traceability of security controls across various categories, ensuring compliance and risk management.

**Sections:**
- Compliance
- Security
- Management
- Operations
- Risk

**Categories:**
- NC: Not Applicable
- X: Not Applicable
- C: Compliance
- S: Security
- M: Management
- O: Operations
- R: Risk

**Additional Information:**
- Tracing the implementation and enforcement of security controls across different phases and environments is crucial for maintaining an effective security posture.
- Regular updates and reviews are necessary to adapt to new threats and improve existing controls.
Taking the ATO process from 6 months to 30 days

By Aidan Feldman  •  July 19, 2018

security  lessons learned  how we work

Security compliance is a major factor in launching a software system in the federal government, in terms of technology choices, and even more importantly, time and effort. The Authority To Operate compliance process for systems within our division of the General Services Administration (GSA) was taking more than six months for every system, with a long and growing backlog. With the new process, we have cleared the backlog and reduced the turnaround time to under a month. We think that deserves a celebration and makes for a good opportunity to share the lessons we’ve learned.
Security Configuration Guides

- Nerd Knobs!
- Documentation to tell how to secure or lock down your application or code
- **Bonus:** Security Content Automation Protocol (SCAP) content and configuration management scripts

[https://github.com/ComplianceAsCode/content](https://github.com/ComplianceAsCode/content)
Step 1 - Document Security Features/Configns

Example 4.1. Configuring password strength-checking in `/etc/security/pwquality.conf`

To enable using `pam_quality`, add the following line to the `password` stack in the `/etc/pam.d/password` file:

```
password  required  pam_pwquality.so  retry=3
```

Options for the checks are specified one per line. For example, to require a password with a minimum length of 8 characters, including all four classes of characters, add the following lines to the `/etc/security/pwquality.conf` file:

```
minlen = 8
minclass = 4
```
Step 2 - Collaborate with ComplianceAsCode to automate

<table>
<thead>
<tr>
<th>OpenShift API Server</th>
<th>22x fail</th>
<th>1x error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the kubelet Certificate Authority for the API Server</td>
<td>high</td>
<td>pass</td>
</tr>
<tr>
<td>Configure the Certificate Key for the API Server</td>
<td>medium</td>
<td>pass</td>
</tr>
<tr>
<td>Enable the EventRateLimit Admission Control Plugin</td>
<td>medium</td>
<td>fail</td>
</tr>
<tr>
<td>Configure the Certificate for the API Server</td>
<td>medium</td>
<td>pass</td>
</tr>
<tr>
<td>Configure the Client Certificate Authority for the API Server</td>
<td>medium</td>
<td>pass</td>
</tr>
<tr>
<td>Configure the Audit Log Path</td>
<td>high</td>
<td>fail</td>
</tr>
<tr>
<td>Use Strong Cryptographic Ciphers on the API Server</td>
<td>medium</td>
<td>fail</td>
</tr>
<tr>
<td>Enable service-account-lookup on the API Server</td>
<td>medium</td>
<td>fail</td>
</tr>
<tr>
<td>Disable use of AlwaysAllow for the API Server Authorization Mode</td>
<td>medium</td>
<td>pass</td>
</tr>
</tbody>
</table>
Open source in space!

- Over 1 teraflop computing power
- Launched by SpaceX to International Space Station in August 2017 [0]

- Hundreds of gigabits/second to hundreds of thousands of viewers
- NGINX, Railo CMS, GlusterFS [1]

Primary Take Aways

- Does your technology function when operating system is configured in FIPS mode? Is this part of your QE plan?

- Are you using operating system provided crypto? (think OpenSSL libraries, not custom crypto)

- To ensure users can easily implement your technology, is there a configuration guide?

- ComplianceAsCode community will help you with these! https://github.com/ComplianceAsCode/content
Accessibility
Accessibility - Section 508

- Expected as part of today's modern development environment
- Broadens your project’s reach to a bigger audience
- Checklists and automated tools provided by the U.S. General Services Administration (GSA) 18F office

https://accessibility.18f.gov/
Checklist

This checklist helps developers identify potential accessibility issues affecting their websites or applications. It's broken down into three sections of decreasing importance: A, B and C. Please check and address these issues in the order in which they appear.

For more detail on accessibility standards, please see [WCAG2.0 AA](https://www.w3.org/WAI/intro/wcag20).

- **A** - Critical issues that will cause serious problems and/or stop most users of assistive technology from using the site
- **B** - Issues that may cause problems or increased frustration for certain users
- **C** - Minor issues that will cause problems or frustration for a small number of users

It is important to note, while B and C are noted as less critical, they are still required to be truly 508 compliant. This checklist should be used as a reference for development and is not a substitute for compliance checks by a section 508 coordinator.

**A - Critical**

1. **Site is keyboard accessible**
   - All interactions can be accessed with a keyboard
2. **Site is free of keyboard traps**
   - The keyboard focus is never trapped in a loop
3. **All form inputs have explicit labels**
4. **All relevant images use an `alt` tag**
5. **All images have `alt` attributes**
6. **Multimedia is tagged**
   - All multimedia has appropriate captioning and audio description
7. **Text has sufficient color contrast**
   - All text has a contrast ratio of 4.5:1 with the background
List of Accessibility Tools

Tools

Color Tools
- WebAIM color contrast checker compares two hex colors and tells you whether they meet WCAG AA and AAA contrast thresholds.
- Snook's color contrast analyzer lets you adjust RGB and HSV values and reports contrast compliance interactively.
- NC State palette accessibility evaluator lets you compare contrast between three or more colors for WCAG AA or AAA compliance.
- Color Safe is a guide for choosing colors that meet WCAG contrast thresholds.
- Color Contrast Analyzer is a desktop application for contrast checking that also simulates different forms of color impairment.

COLOR IMPAIRMENT
- Color Oracle is another desktop application for simulating color impairment on your entire screen.
- Daltonize is a collection of bookmarks that simulate the three most common forms of color impairment (protanopia, deuteranopia, and tritanopia) on any web page.
- colourblind is another simulation tool similar to Daltonize, but with more options (protanopia, protanomaly, deuteranopia, deuteranomaly, tritanopia, tritanomaly, achromatopsia, and achromat anomaly) in a single bookmarklet.
- postcss-colorblind is a CSS build tool that modifies colors in your CSS to simulate four common impairment groups.

Accessibility Review Tools
These tools can be used to test sites for Section 508 and WCAG compliance in browser:
- achecker is an accessibility reporter for HTML only.
- Google's Accessibility Developer Tools is a Chrome plugin for running basic accessibility tests from the comfort of your browser.
- Web Accessibility Toolbar (WAT) is an IE tool that has been developed to aid manual examination of web pages for a variety of aspects of accessibility. It is used by DHS's Trusted Tester program.