SCAP Security Guide

https://fedorahosted.org/scap-security-guide/

SHAWN WELLS
shawn@redhat.com
(+1) 443-534-0130

What is the SSG Project?

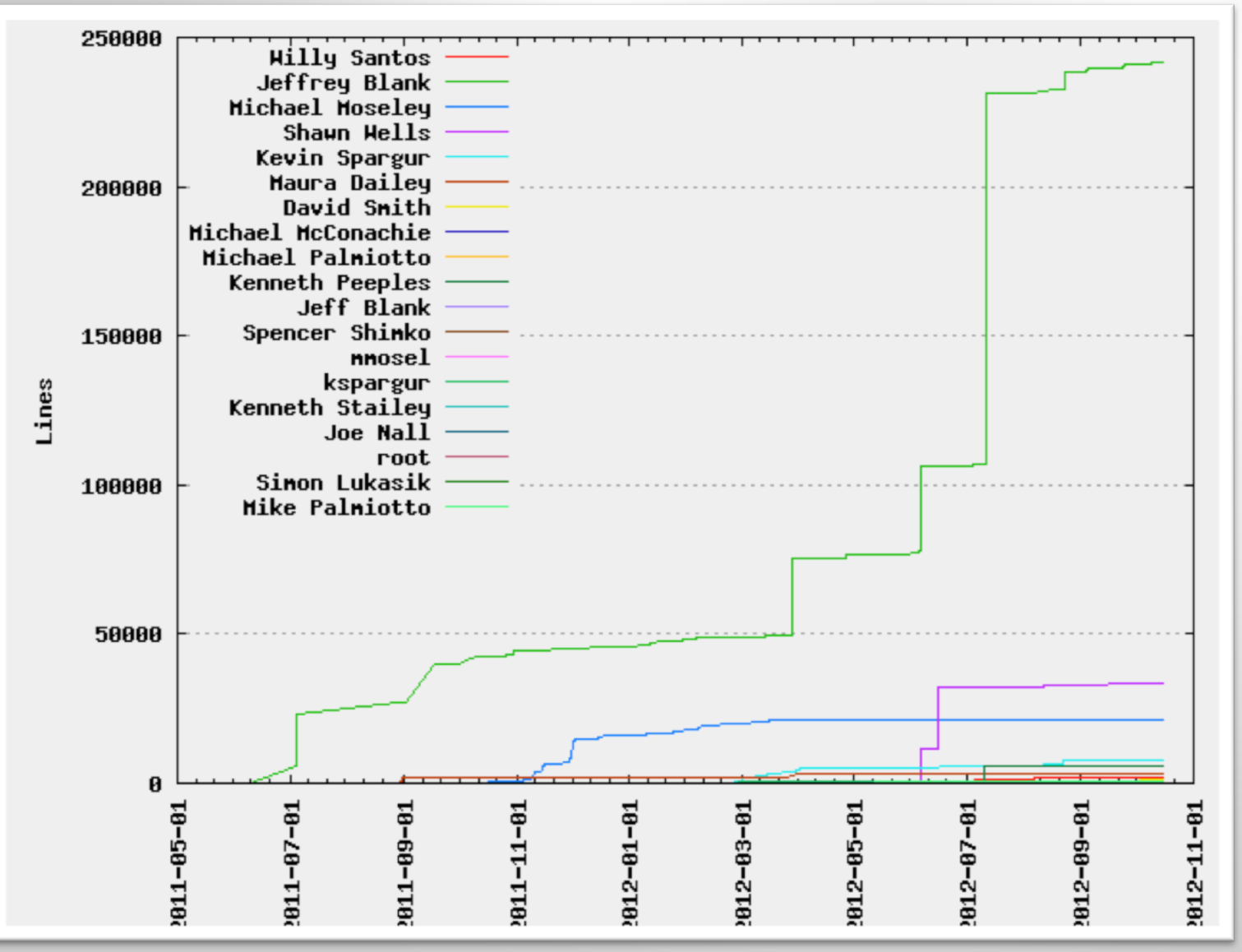
- Delivers practical security guidance, baselines, and associated validation mechanisms using the Security Content Automation Protocol (SCAP)
    - Current content for RHEL 6, JBoss EAP5
- Upstream source for government *implementation* guidance moving forward
    - JBoss Enterprise Application Platform 5 STIG
    - Red Hat Enterprise Linux 6 STIG, SNAC Guide, NIST 800-53 baselines

What is the SSG Project?

- Recommendations map to government policies where applicable

- Because of this mapping, we can create custom profiles:
  - RHEL 6 STIG (collaboration with DISA FSO)
  - RHEL 6 Security Guide (collaboration with NSA)
  - Baseline content for NIST 800-53 (e.g. H/H/H, H/L/L, etc)

Who is the SSG Project?



Domains	Total (%)
eclipse.ncsc.mil	595 (49.30%)
redhat.com	578 (47.89%)
tresys.com	14 (1.16%)
gmail.com	9 (0.75%)
nall.com	3 (0.25%)
kspargur.csb	3 (0.25%)
kde.example.com	3 (0.25%)
rhel6.(none)	1 (0.08%)
fornax.eclipse.ncsc.mil	1 (0.08%)


I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO, IT'S DURING LIVE DEMOS
memegenerator.net

Step 1: Download the Code

1. Download required packages:

```
# yum install git openscap-utils python-lxml
```

2. Grab the latest code (...eventually we will have release RPMs):

```
$ cd /tmp/
$ git clone \
  git://git.fedorahosted.org/scap-security-guide.git
```

3. Run make

```
$ cd /tmp/scap-security-guide/; make all
```

Step 2: Review Human Readable Outputs

1. Review prose guide

   ```
   $ firefox /tmp/scap-security-guide/RHEL6/output/rhel6-guide.html
   ```

- Reads like NSA SNAC Guide
- Includes all controls

2. Review RHEL6 STIG candidate

   ```
   $ firefox /tmp/scap-security-guide/RHEL6/output/table-rhel6-stig-server-shorttitles.html
   ```

- Includes manual check details (OCIL text)
- Using DoD Consensus process with Red Hat, NSA, DISA FSO
- RHEL6 STIG by Christmas?

3. Notice the last column! Mappings back to CCI, NIST 800-53.... foundations of automatically generated SRTM

Step 3: Review the XCCDF ("Prose Code")

1. XCCDF: Fancy way of saying XML-ized prose

```
$ cd /tmp/scap-security-guide/RHEL6/input/
$ vim system/auditing.xml
/enable_auditd_service
```

2. Notice the structure

Rule ID's	Random name for the rule, machine readable
Title	Human name
Description	Tells how to meet rule
OCIL	How to manually verify compliance
Rationale	Describes why the rule is important
IDENT	Unique identifier for CCE's
OVAL ID	Maps back to an OVAL check, the machine language for automated compliance language
REF	We made this up, to create correlations back to policy documents

Example Code: XCCDF Rule

```
<Rule id="enable_auditd_service”>
    <title>Enable auditd Service</title>
    <description>
      The <tt>auditd</tt> service is an essential userspace
      component of the Linux Auditing System, as it is
      responsible for writing audit records to disk.
      <service-enable-macro service="auditd" />
    </description>
    <ocil><service-enable-check-macro service="auditd" /></ocil>
    <rationale>
      Ensuring that the <tt>auditd</tt> service is active ensures that
      audit records generated by the kernel can be written to disk, or
      that appropriate actions will be taken if other obstacles exist.
    </rationale>
    <ident cce="4292-9" />
    <oval id="service_auditd_enabled" />
    <ref nist="CM-6, CM-7"
disa="169,172,174,1353,1462,1487,1115,1454,067,158,831,1123,1190,1312,
1263,130" />
</Rule>
```

Step 3: Review the OVAL ("Scanning Code")

OVAL == Open Vulnerability and Assessment Language

Three primary purposes:

- Represent configuration information of systems for testing;
- Analyzing the system for the presence of specified machine state
- Reporting results

1. Review the code

```
$ cd /tmp/scap-security-guide/RHEL6/input/
$ vim checks/sysctl_net_ipv6_disabled.xml
```

2. Notice the structure

<affected>	What OS's to perform the check on
<ind:textfilecontent54_test>	Outlines conditions for the test, points to "object"
<ind:textfilecontent54_object>	Spells out what to check for

Example Code: OVAL Check

```
<ind:textfilecontent54_object id="obj_20134" version="1">

   <ind:path>/etc</ind:path>

   <ind:filename>sysctl.conf</ind:filename>

   <ind:pattern operation="pattern match">^\s*net\.ipv6\.conf\.all
\.disable_ipv6\s*=\s*1$</ind:pattern>

   <ind:instance datatype="int">1</ind:instance>

 </ind:textfilecontent54_object>
```

Step 4: Review Profiles

- As a body of work, SSG represents a catalog of recommendations
- Not all recommendations are applicable to everyone, so we create or logical groupings of rules called “Profiles,” e.g.:
  - Red Hat Enterprise Linux 6 STIG
  - Red Hat Enterprise Linux 6, NIST 800-53 H/L/L

1. Review the code

```
$ cd /tmp/scap-security-guide/RHEL6/input/profiles/
$ vim STIG-server.xml
```

2. Notice the structure

<select idref>	Pulls in the rule
<refine-value>	For some things we included variable substitution to allow for customizations, e.g. length of passwords (vim: /var_password_max_age)

Example Code: STIG Profile

```
<select idref="daemon_umask" selected="true" />
<refine-value idref="var_umask_for_daemons" selector="027"/>


<select idref="no_netrc_files" selected="true" />

<select idref="ftp_present_banner" selected="true" />

<!-- from inherited Rule, limiting_password_reuse -->
<refine-value idref="password_history_retain_number" selector="24"/>

<refine-value idref="var_password_max_age" selector="60"/>
<!-- from inherited Rule, deny_password_attempts -->
<refine-value idref="var_accounts_passwords_pam_faillock_deny" selector="3"/>
```

Step 5: Run a Scan

- Guide online at https://fedorahosted.org/scap-security-guide/wiki/usageguide

1. Some checks require root access

```
$ sudo su -
```

2. Run a scan for STIG compliance

```
# cd /tmp/scap-security-guide/RHEL6/output
# oscap xccdf eval --profile stig-server ssg-rhel6-xccdf.xml
```

Step 6: *DEVELOPER PREVIEW of <fix> tags*

- The protocol also supports inclusion of remediation scripts
- Eventually, we'll merge in bash and puppet remediation. When we do this depends on how loudly people scream for it.
- Development done under RHEL6/input/fixes
  (not much there as of 16-OCT-2012, patches welcome!)