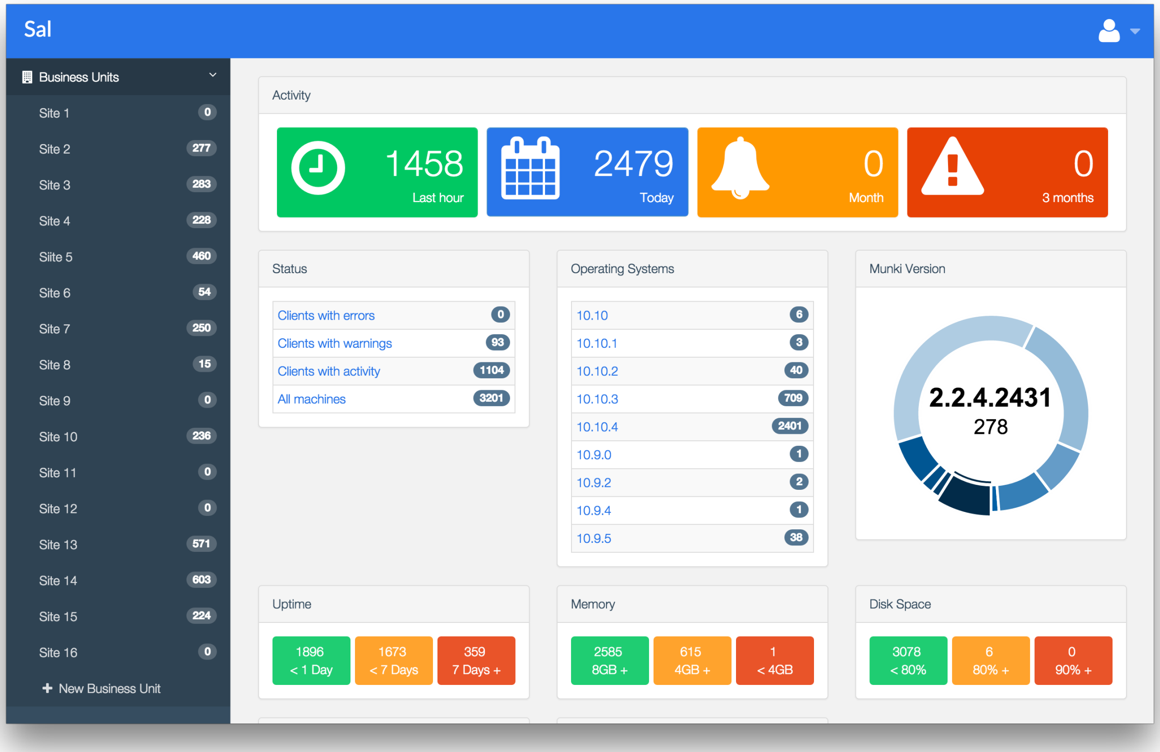
Sal Dashboard Tool for Munki Software Management, Whitepaper



Business Units	
Site 1	0
Site 2	277
Site 3	283
Site 4	228
Siite 5	460
Site 6	54
Site 7	250
Site 8	15
Site 9	0
Site 10	236
Site 11	0
Site 12	0
Site 13	571
Site 14	603
Site 15	224
Site 16	0

Activity

1458	2479	0	0
Last hour	Today	Month	3 months

Status	
Clients with errors	0
Clients with warnings	93
Clients with activity	1104
All machines	3201

Operating Systems	
10.10	6
10.10.1	3
10.10.2	40
10.10.3	709
10.10.4	2401
10.9.0	1
10.9.2	2
10.9.4	1
10.9.5	38

Munki Version

2.2.4.2431
278

Uptime		
1896	1673	359
< 1 Day	< 7 Days	7 Days +

Memory		
2585	615	1
8GB +	4GB +	< 4GB

Disk Space		
3078	6	0
< 80%	80% +	90% +

Prepared by Allister Banks, November, 2015

Background

This whitepaper illustrates the role Sal performs in relation to the Munki software management system. Created and maintained by Graham Gilbert for the thousands of Macs under his care, it is the foremost reporting dashboard for Munki written in the robust Django web framework. Sal allows a view specific to the health of machines as patches are applied, along with the compliance state of a fleet according to highly customizable criteria.

Executive Summary

Without monitoring and getting a consistent stream of reports, the effectiveness of a workstation management tool is hard to prove. Sal works within the framework Munki provides to send data before and after a scheduled check and/or software changes, staying as close to real-time about the status of its management as possible. It is easily extensible with python development, and the display can be customized for the metrics that are important to organizations managing Macs. With controls to grant view access based on business unit, graphing to display the various metrics an administrator can specify, and integration with data collection products like Puppetlab's facter and security auditing tools like Facebook's osquery, Sal provides a web console to the state of the Munki software management system, so compliance can be measured and reported on.

Features and Operation Overview

The name Sal came from its integration with the Puppet configuration management tool, and in addition it provides several integration points for Mac client management and inventory software. It does its data shipping through a series of pre-flight and post-flight scripts, so it knows what state the device was in before Munki checks its configuration against the server, and after it has performed its tasks, helping collect results or warning information. By leveraging Django, many best practices like brute force protections and cross-site scripting attacks can be mitigated. At its simplest, you can customize the layout of metrics on the dashboard page, referred to as plugins, and configure which users can view which business units. The real power comes from when you push new scripts to the various data collection facilities Sal can then gather results from and report on them.

Facter

There are a bevy of details about a computer that can be discovered out-of-the-box with Facter, a tool from Puppetlabs that gathers 'facts' about the device it's running on. Written in C++, it pulls things that would be costly to use a shell script for, but can be extended with custom criteria in a few ways. Once Facter accesses the data, Sal can present it in the tab in a computer detail interface, and many of the widgets on Sal's dashboard are fed by it. Things like RAM and computer uptime are displayed to help provide techs with a resource when responding to customer concerns about their device, and everything from encrypted memory to mountpoints to ssh keys can become a list of remediation tasks to perform if compliance needs to ensure a particular state.

The community of Mac administrators using Puppet have developed custom facts to check on the state of FileVault2, Apple's full disk encryption, and other criteria which can be added, but a lot of these things are covered by osquery, mentioned below.

ManagedInstallReport and ConditionalItems

At its most basic, Munki needs to keep a local accounting of how it performed, and gather a selection of criteria to act on when it evaluates what to do with the software or configurations offered to it. Traditionally folks who were using the 'ConditionalItems' aspect of Munki to perform more dynamic evaluations client-side would consult the ManagedInstallReport that gathered the info that resulted from static evaluations built into Munki (e.g. if the computer is a laptop or desktop) or the conditional item scripts being run. These provide another avenue to tell the compliance state of a computer which can then be surfaced by Sal, and has its own place on the last tab in the computer detail webpage.

osquery

One of the newer features of Sal is its capability to provide an endpoint that the osquery daemon can log to. Integration with this tool is still being fleshed out, but for binary, on/off or 'counter'-style metrics, you can now see the result of query packs in the same dashboard as the other reporting conduits. This makes both telling the state across tools and deploying more advanced configurations of osquery much easier to get started with and maintain.

Similar Products and Compliance

There are various tools with similar features, including one that is popular with many admins called MunkiReportPHP, but some prefer the Django web framework and the ease of extension that a tool written in python allows. JAMF's Casper Suite has a dashboard that does not integrate with Munki, and provides crude ways to write custom scripts that can cause interruptions and/or overhead on the workstations under management. The built-in inventory tool that JAMF distributes as a binary is lossy in the actions it performs, and can easily be fooled into reporting an inaccurate state.

Through its unique way of collecting the best-of-breed tools up in one interface, Sal can help auditors see a rich set of metrics at a glance. It reinforces the same strengths of Munki's assistance with enforcing PCI, NIST, and HIPAA standards.

PCI DSS

Munki's compliance with the PCI-DSS requirement 6.2b to apply critical patches within one to three months of release[1] goes hand-in-hand with Sal's reporting to ensure it actually happened.

NIST 800-53

Of the NIST guidelines, CM-8[2] in particular calls out the flexibility of how Munki's model allows customizations if a small group of hosts need a particular configuration, and granting admins filtered and read-only access ensures accountability for who can see the collected data. Similar to Munki and made reference to in CM-10, Sal is Apache licensed there are no issues about patents or copyright-related issues.

[1] https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-1.pdf pg. 53

[2] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf pg. F-73

HIPAA

As specified in the HIPAA Administrative Simplification 164.312(a)[3], Sal sends collected and parsed log data. Commonly the simple web server that Munki needs can be configured to only grant access over https with client certificates per-device through the use of a Certificate Authority. This allows revocation of a client that goes missing, denying further access of resources.

Conclusion

Sal follows the Unix philosophy of making small tools that do one thing well. By focusing on the reporting components, and using battle-tested technologies like Django, it is scalable and can incorporate many sources of data in its display. The customizability is top notch, and through being custom-built to integrate with the best software management system, Munki, it delivers a lot of value just by giving an accurate, snapshot-in-time status of your management.

[3] http://www.hhs.gov/ocr/privacy/hipaa/administrative/combined/hipaa-simplification-201303.pdf pg. 67