CHEF™
CHEF.IO

THE CHEF AUTOMATE GUIDE TO FFIEC COMPLIANCE

Published December, 2018

EXECUTIVE SUMMARY

If you are a financial institution, you are subject to a variety of governance and compliance regimes. Modern IT infrastructure management techniques can help your organization significantly reduce the time spent on compliance activities while also improving your organization's compliance posture. Chef Automate helps teams burdened with verifying compliance and proving policy enforcement free up valuable time to innovate, while also satisfying compliance audits with less work. This guide helps operations and compliance personnel apply Chef Automate and InSpec to gain visibility and control over the compliance status of their IT systems quickly and automatically.

Among the most critical financial compliance activities is oversight by the Federal Financial Institutions Examination Council (FFIEC), a five-member U.S. government agency composed of banking regulators empowered to establish consistent guidelines, uniform principles, and practices for financial institutions. The member agencies include the Board of Governors of the Federal Reserve System (FRB), the Federal Deposit Insurance Corporation (FDIC), the National Credit Union Administration (NCUA), the Office of the Comptroller of the Currency (OCC) and the Office of Thrift Supervision (OTS).

Rapidly changing technology landscapes pose a particular challenge to financial institutions, especially those with an online presence. In response, the FFIEC provides and frequently updates its information technology examination manual to account for the increasing pace of changes and advancements in technology adopted by financial institutions and technology service providers. That manual, the FFIEC IT Examination Handbook, is a compilation of eleven booklets that provide financial institutions with expectations for compliance.

FFIEC compliance is considered to be a layered approach to security and is not limited to any one specific technology. Rather, it incorporates a number of different tactics and strategies working together. FFIEC provides high-level process requirements instead of low-level prescriptive practices for implementation in order to establish compliance. The scope of a FFIEC audit includes examination of your organizational policies for compliance and then focuses on demonstrably showing that your technology abides by your policies.

Verifying compliance and proving policies are enforced is time consuming and error prone, particularly in light of the complexity and frequent change common to the financial IT landscape. The amount of work for an IT team to demonstrate compliance across your infrastructure can be staggering when using traditional manual approaches. When considering FFIEC standards are not the only regime to which your organization is subject, the work required to manage any reasonably sized application typically requires hundreds or thousands of hours in labor to complete.

It is critical for any organization subject to FFIEC audits to quickly pass their audit with as little manual work necessary so that teams can instead focus on achieving business objectives. By automating these processes, engineering teams can spend less time hunting down information to satisfy audit requests and more time doing product development.

Chef Automate can help you implement continuous security assessments that allow you to satisfy audit requirements at any time and make audits painless.

Adopting a continuous compliance approach allows you to quickly answer audit questions and provide an audit trail that demonstrably proves your systems are--and have been--in compliance with your organizational policies at any time. Enter an audit cycle knowing your exact compliance posture, rather than being surprised by auditors finding blind spots buried within your IT infrastructure. You can identify compliance issues or policy breaches rapidly and react quickly to triage and remediate problems even before auditors show up, allowing you to demonstrate how compliance has evolved and improved over time.

Continuous assessments are Chef's recommended approach. This guide is written for members of both technical and compliance teams working in financial institutions subject to FFIEC audits. Traditionally, these teams must manually gather and provide overwhelming amounts of data to auditors using ad-hoc methods. FFIEC auditors probe not just your technology stack, but also the vulnerability of your teams to unwittingly violate your organizational policy. By integrating a demonstrable method for proving compliance into your team's daily workflow, you can also ensure that the process followed to provide auditors with required data also follows your organizational policies.

This guide illustrates how to start using InSpec and Chef Automate to validate system configurations during an audit in order to map existing manual functions to automated controls. A recommended next step would be to use Chef Automate to integrate automated compliance assessments into your continuous delivery workflow. Find out more about how Chef Automate enables continuous assessments by visiting https://www.chef.io/automate/.

INTRODUCTION

A typical approach to passing a FFIEC audit is to issue ad-hoc remote commands to gather information, compose verification scripts to run by hand, or to manually verify a number of system settings in tandem with auditors using approaches like screenshots. That approach is fundamentally unsustainable since it requires custom work that is specific only to the context of your FFIEC audit and cannot be leveraged in other parts of business-critical workflows.

InSpec is an auditing tool that turns your compliance, security, and other policy requirements into automated tests. These tests are designed to continuously gather data about your systems using a standardized language and data format that clearly maps to the compliance controls your auditors care about. InSpec has several packaged compliance rule sets that cover industry-standard use cases. However, if rules for your specific use case do not exist, InSpec allows you to easily compose them. InSpec is designed to leverage the same automated testing used in other parts of your organization to make discussions with your compliance auditors easier and faster to resolve.

Chef Automate is a continuous automation platform that provides dashboards and reporting to monitor and respond to compliance status gathered through InSpec tests. It integrates the Chef infrastructure automation engine, providing a single place for operations, security, and compliance teams to collaborate in order to detect and correct compliance issues across the IT estate. Chef Automate ships with approximately 100 pre-built and continuously maintained InSpec profiles, most of which are based on Center for Internet Security (CIS) benchmarks.

This guide covers the Risk Mitigation components of the FFIEC Information Security booklet. The Information Security booklet provides guidance to examiners assessing the adequacy of a financial institution's information systems and their security program as part of an overall risk management strategy. Once your organization has adequately identified and measured potential risks, it should also have implemented an appropriate plan to mitigate those risks. That plan should include having clear visibility into the current state and quality of your environment. This guide illustrates how to use existing InSpec controls, or create new controls, to provide that visibility.

ABOUT FFIEC AND CIS CONTROLS

FFIEC provides a set of guidelines designed to provide adequate organizational risk mitigation strategies when used collectively. The FFIEC Information Security booklet contains several risk mitigation strategies specific to your information systems. Some strategies apply to non-technical concerns such as physical access and processes governing human interaction. However, several strategies apply directly to your digital systems. The exact details of those strategies may be different based on your specific implemented technologies. However, this guide will use standard examples to illustrate how an effective approach to satisfying FFIEC audit requirements could be established in your organization.

The Center for Internet Security (CIS) publishes a set of controls that collectively form a defense-in-depth set of best practices that mitigate the most common attacks against systems and networks. The CIS Benchmarks are developed by a community of IT experts who apply their first-hand experience as cyber defenders to create a set of globally accepted security best practices. As a result, the CIS Benchmarks have matured into a trusted baseline of controls that not only identify, track, and remediate cyber security attacks but that also map those controls to regulatory and compliance frameworks.

Several CIS controls map directly to mitigation techniques outlined by the FFIEC Information Security booklet. Using the CIS controls available via Chef Automate can help jumpstart your team's efforts to pass an FFIEC audit. This guide uses mappings of CIS controls to the FFIEC Examiner's Handbook[1] to inspect a Red Hat Enterprise Linux 7 system for compliance. A subscription to CIS benchmarks for other operating systems, expressed in InSpec, is included with a Chef Automate license.

ABOUT INSPEC

InSpec is an open-source language for describing security & compliance rules that can be shared between software engineers, operations, and security engineers. InSpec is designed to be used at all stages of the software delivery process, from a developer's workstation all the way to production, with minimal performance impact, allowing companies to validate compliance continuously, rather than only during regular audit cycles. And, in contrast to other compliance languages, InSpec is designed to be easy to use, even by users with no background in programming.

1 https://www.cisecurity.org/wp-content/uploads/2017/03/Poster_Winter2016_CSCs.pdf

UNDERSTANDING FFIEC REQUIREMENTS FOR INFORMATION SECURITY

FFIEC Information Security requirements vary in scope. Chef Automate can be most impactful when it comes to Change Management within the IT Environment (§ II.C.10). In FFIEC terms, an IT environment consists of operating systems, middleware, applications, file systems, and communications protocols. Requirements are geared toward verifying effective processes to manage application and system changes, including hardware, software, and network devices in the IT environment. Software deployment processes should encompass securely developing, implementing, and testing changes to both internally developed and third-party software.

Configuration Management is a core strategic focus for FFIEC Information Security Requirements (§ II.C.10(a)) in order to securely maintain an organization's technology stack by developing expected baselines for tracking, controlling, and managing system settings. When information systems change, management should update baselines; confirm security settings; and track, verify, and report configuration items. Configurations should be monitored for unauthorized changes, and misconfigurations should be identified. FFIEC encourages use of automated solutions to help track, manage, and identify necessary corrections.

Chef Automate provides a comprehensive approach to managing change within the IT Environment. It provides configuration management (via Chef) to track and manage necessary system configuration corrections. It also provides InSpec to identify necessary corrections using a separate auditable process that ensures proper chain of custody for information security compliance. The following sections illustrate how to use a combination of pre-packaged and custom InSpec controls to create a comprehensive strategy to more quickly satisfy the requirements of your FFIEC audit.

HOST SECURITY

FFIEC provides several considerations for host-based security for system in the IT environment (§ II.C.10(b)). The variety of devices found in the IT environment include network infrastructure, servers, desktops, laptops, and others. As such, organizations rely on commercial off-the-shelf software to provide operating systems and applications for these diverse platforms. However, these commercial solutions often provide greater functionality than what is needed for the organization-specific purposes for which they are employed. Therefore, it's critical to harden these systems beyond their default installations to ensure better information security.

FFIEC does not provide specific recommendations for system hardening and instead refers to general practices and following prescriptive vendor best practices. We can use several off-the-shelf CIS controls to ensure we follow industry best practices for hardening our example system. Below are a few sample controls we could use to better harden a default system installation against virtual and physical access.

1. ENSURE BOOTLOADER PASSWORD IS SET

If an attacker has physical access to your systems, it is possible for them to make unauthorized modifications by rebooting your systems and entering custom boot parameters on the command line that can grant them privileged access when the system restarts. Requiring a boot password prevents unauthorized users from entering boot parameters or changing the boot partition, thereby hardening your systems against intruders with physical access.

```
control "cisecurity.benchmarks_rule_1.4.2_Ensure_bootloader_password_is_set" do
    title "Ensure bootloader password is set"
    desc "Setting the boot loader password will require that anyone rebooting
the system must enter a password before being able to set command line boot
parameters. Rationale: Requiring a boot password upon execution of the boot
loader will prevent an unauthorized user from entering boot parameters or
changing the boot partition. This prevents users from weakening security (e.g.
turning off SELinux at boot time)."
    impact 1.0
    tag "cis-rhel7-2.1.1": "1.4.2"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe file("/boot/grub2/grub.cfg") do
```

```
        its("content") { should match(/^\s*set\s+superusers\s*=\s*"[^"]*"\s*(\
s+#.*)?$/) }
        its("content") { should match(/^\s*password_pbkdf2\s+\S+\s+\S+\s*(\
s+#.*)?$/) }
    end
end
```

This control inspects the contents of the configuration file for the GRUB 2 boot loader (/boot/grub2/grub.cfg) to ensure the file contains an entry requiring superuser access to modify boot entries and that a hash of the boot password exists.

2. ENSURE SELINUX IS NOT DISABLED IN BOOTLOADER CONFIGURATION

In the previous control, we were able to check that unauthorized users are not able to make changes to our bootloader configuration. However, we also want to ensure that authorized users are not unwittingly introducing configurations that make our systems less secure. We can use a control to check that SELinux (Linux kernel security module that provides access control security policies) is enabled at boot time, and verify that it has not been overwritten by GRUB 2 boot parameters.

```
control "cisecurity.benchmarks_rule_1.6.1.1_Ensure_SELinux_not_disabled_in_
bootloader" do
    title "Ensure SELinux is not disabled in bootloader configuration"
    desc "Configure SELINUX to be enabled at boot time and verify that it has not
been overwritten by the grub boot parameters. Rationale: SELinux must be enabled
at boot time in your grub configuration to ensure that the controls it provides
are not overridden."
    impact 1.0
    tag "cis-rhel7-2.1.1": "1.6.1.1"
    tag "level": "2"
    tag "type": ["Server", "Workstation"]
    describe.one do
        describe package('libselinux') do
            it { should_not be_installed }
        end
        describe file("/boot/grub2/grub.cfg") do
            its("content") { should_not match(/^\s*linux\S+(\s+\S+)+\
s+selinux=0\s?(\s+\S+)*$/) }
            its("content") { should_not match(/^\s*linux\S+(\s+\S+)+\
s+enforcing=0\s?(\s+\S+)*$/) }
        end
    end
end
```

This control uses "describe.one" to check that either of two configurations are true.

The libselinux library provides an API for SELinux applications to get and set process and file security contexts, and to obtain security policy decisions. If the libselinux package is not installed, then it ensures that even if insecure parameters are inadvertently set, they cannot be successfully passed to the system at boot time.

If the libselinux library is installed, then this control inspects the contents of /boot/grub2/grub.cfg to ensure that SELinux has not been set to disabled or to a mode where it is not enforcing system policies.

3. ENSURE STICKY BIT IS SET ON ALL WORLD-WRITABLE DIRECTORIES

A sticky bit is a permission setting for filesystem directories that allows only the owner of the file within that directory or the root user to delete or rename the file. No other user has the needed privileges to delete the file created by some other user. By default, Linux systems include some world-writable directories (such as /tmp) that are necessary for accomplishing specific tasks. If our system includes world-writable directories, we should also ensure that non-privileged users cannot delete or rename files that are owned by another user.

```
control "cisecurity.benchmarks_rule_1.1.21_Sticky_bit_for_world-writable_dirs"
do
    title "Ensure sticky bit is set on all world-writable directories"
    desc "Setting the sticky bit on world writable directories prevents users
from deleting or renaming files in that directory that are not owned by them.
Rationale: This feature prevents the ability to delete or rename files in world
writable directories (such as /tmp ) that are owned by another user."
    impact 1.0
    tag "cis-rhel7-2.1.1": "1.1.21"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe command('find / -path /proc -prune -o -type d \( -perm -0002 -a ! 
-perm -1000 \) -print 2>/dev/null') do
        its('stdout') { should eq ""}
    end
end
```

This control uses the InSpec built-in function for running system commands to call `find` to search for appropriate directories using octal file permission values. If our system is configured correctly, this find command should return an empty set of matching results.

PATCH MANAGEMENT

Frequently, security vulnerabilities are discovered in operating systems and other software after deployment. Hackers often will attempt to exploit these known vulnerabilities to try to gain access to an organization's systems. Third parties issue patches to address vulnerabilities found on organizational systems and applications. The FFIEC Information Security booklet requires that organizations setup and maintain appropriate Patch Management policies (§ II.C.10(d))

1. ENSURE PACKAGE MANAGER REPOSITORIES ARE CONFIGURED

If a system's package repositories are misconfigured important patches may not be identified or a rogue repository could introduce compromised software. Systems should have their package manager repositories configured in a way that ensures they receive the latest patches and updates.

```
control "cisecurity.benchmarks_rule_1.2.1_Ensure_pkg_manager_repos_are_
configured" do
    title "Ensure package manager repositories are configured"
    desc "Systems need to have package manager repositories configured to ensure
they receive the latest patches and updates. Rationale: If a system's package
repositories are misconfigured important patches may not be identified or a
rogue repository could introduce compromised software."
    impact 0.0
    tag "cis-rhel7-2.1.1": "1.2.1"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    REDHAT_REPOS.each do |repository|
        describe yum.repo(repository) do
            it { should exist }
            it { should be_enabled }
        end
    end
    cmd = command('yum repolist enabled').stdout.split("\n")
    get_other_repos = cmd.slice(2..cmd.length-2) || []
    other_repos = get_other_repos.map { |repositories| repositories.gsub(/\s.+/,
'') }
```

```
    other_repos -= REDHAT_REPOS
    unless other_repos.empty?
        other_repos.each do |repository|
            describe yum.repo(repository) do
                it { should_not exist }
                it { should_not be_enabled }
            end
        end
    end
end
```

This control uses Ruby to iterate through all package repositories defined as Red Hat system repos in the CIS benchmarks. Each of those repositories must be defined on the system and enabled. The control also ensures that repositories set up via yum to install software from additional resources are also defined and enabled on the system.

2. ENSURE UPDATES, PATCHES, AND ADDITIONAL SECURITY SOFTWARE IS INSTALLED

Periodically, patches are released for included software either due to security flaws or to include additional functionality. Newer patches may contain security enhancements that would not be available through the latest full update. As a result, it is recommended that the latest software patches be used to take advantage of the latest functionality. As with any software installation, organizations need to determine if a given update meets their requirements and verify the compatibility and supportability of any additional software against the update revision that is selected.

```
control "mycompany.custom_rule3.0.0_Ensure_updates_and_patches_installed" do
    title "Ensure that all updates, patches, and latest packages are installed"
    desc "Periodically patches are released for included software either due to security flaws or to include additional functionality. Rationale: Newer patches may contain security enhancements that would not be available through the latest full update. As a result, it is recommended that the latest software patches be used to take advantage of the latest functionality."
    impact 1.0
    tag "cis-rhel7-2.1.1": 1.8
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    linux.update.updates.each { |update|
        describe package(update['name']) do
            its('version') { should eq update['version'] }
        end
    }
    only_if { linux_update.updates.length > 0 }
end
```

This control uses a custom InSpec resource to defer management of updates to the local package manager based on your particular Linux distribution. For a more thorough approach to ensuring proper Linux patching, refer to the dev-sec Linux Patch benchmark[2].

3. ENSURE UPDATES, PATCHES, AND ADDITIONAL SECURITY SOFTWARE IS INSTALLED

Red Hat Enterprise Linux systems must be registered with the Red Hat Network (RHN) or Red Hat Subscription Manager (RHSM) to receive patch updates. This is usually configured during initial installation. However, we want to ensure that it is currently configured on our systems or that we flag it as a system out of compliance if it is not. It is important to register with the Red Hat Network to make sure that patches are updated on a regular basis. This helps to reduce the exposure time as new vulnerabilities are discovered.

```
control "cisecurity.benchmarks_rule_1.2.4_Ensure_either_RHN_or_RHSM_configured"
do
    title "Ensure Red Hat Network or Subscription Manager connection is
configured"
    desc "Systems need to be registered with the Red Hat Network (RHN) or Red Hat
Subscription Manager (RHSM) to receive patch updates. This is usually configured
during initial installation. Rationale: It is important to register with the Red
Hat Network to make sure that patches are updated on a regular basis. This helps
to reduce the exposure time as new vulnerabilities are discovered."
    impact 0.0
    tag "cis-rhel7-2.1.1": "1.2.4"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
```

2 https://github.com/dev-sec/linux-patch-baseline

```
    options = { assignment_regex: /^\s*([^:]*?)\s*:\s*(.*?)\s*$/ }
    describe parse_config(command('subscription\-manager identity').
stdout,options) do
        its('system identity') { should_not include 'This system is not yet
registered.' }
    end
end
```

This control uses InSpec built-in resources to issue a system command and parse the output to specifically examine the state of its reported “system identity”. A system that is not currently configured to receive updates from RHM or RHSM will state that it is not yet registered.

OPERATING SYSTEM ACCESS

Unauthorized access to the operating system and system utilities could result in significant financial and operational losses. System and security administrators should restrict and monitor privileged access to operating systems and system utilities. Many operating systems have integrated or third-party software that provide effective management to control access to the operating system and applications. The FFIEC Information Security booklet provides several considerations for Operating System Access Access (§ II.C.15(a)).

1. ENSURE ACCESS TO THE SU COMMAND IS RESTRICTED

Restricting the use of the "su" command, and using "sudo" in its place, provides system administrators better control of the escalation of user privileges to execute privileged commands. The sudo utility also provides a better logging and audit mechanism, as it can log each command executed via sudo. Normally, the su command can be executed by any user. By uncommenting the pam_wheel.so statement in the /etc/pam.d/su file, the su command will only allow users in the wheel group to execute "su".

```
control "cisecurity.benchmarks_rule_5.6_Ensure_access_to_the_su_command_is_
restricted" do
    title "Ensure access to the su command is restricted"
    desc "The su command allows a user to run a command or shell as another user.
The program has been superseded by sudo , which allows for more granular control
over privileged access. Normally, the su command can be executed by any user.
By uncommenting the pam_wheel.so statement in /etc/pam.d/su, the su command will
only allow users in the wheel group to execute su."
    impact 1.0
    tag "cis-rhel7-2.1.1": 5.6
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe file("/etc/pam.d/su") do
        its("content") { should match(/^\s*auth\s+required\s+pam_wheel.so\s+(\S+\
s+)*use_uid\s*(\S+\s+)*$/) }
    end
end
```

This control inspects the contents of /etc/pam.d/su to ensure the correct content is found.

2. ENSURE SSH ROOT LOGIN IS DISABLED

Disallowing root logins over SSH requires system admins to authenticate using their own individual account, then escalating to root only via sudo (if you've disabled access to "su" as in the control above). This restriction limits opportunity for non-repudiation (*Ensuring that a transferred message has been sent and received by the parties claiming to have sent and received the message.*) and provides a clear audit trail in the event of a security incident. The PermitRootLogin parameter specifies if the root user can login using ssh.

```
control "cisecurity.benchmarks_rule_5.2.8_Ensure_SSH_root_login_is_disabled" do
    title "Ensure SSH root login is disabled"
    desc "The PermitRootLogin parameter specifies if the root user can login
using ssh(1). The default is no. Rationale: Disallowing root logins over SSH
requires system admins to authenticate using their own individual account, then
escalating to root via sudo"
    impact 1.0
    tag "cis-rhel7-2.1.1": "5.2.8"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe sshd_config do
        its('PermitRootLogin') { should eq 'no' }
    end
end
```

This control uses InSpec's built-in sshd_config resource to inspect the configuration of the OpenSSH daemon.

3. ENSURE SSH ACCESS IS LIMITED

Restricting which users can remotely access the system via SSH will help ensure that only authorized users access the system. There are several options available to limit which users and group can access the system via SSH. It is recommended that at least one of the following options be leveraged: AllowUsers, AllowGroups, DenyUsers, or DenyGroups. These are each space separated lists of usernames to allow, group names to allow, usernames to deny, or group names to deny. Numeric values for users and groups are not recognized by these constructs.

```
control "cisecurity.benchmarks_rule_5.2.15_Ensure_SSH_access_is_limited" do
    title "Ensure SSH access is limited"
    desc "There are several options available to limit which users and group can
access the system via SSH. It is recommended that at least one of the following
options be leveraged: AllowUsers, AllowGroups, DenyUsers, or DenyGroups.
Rationale: Restricting which users can remotely access the system via SSH will
help ensure that only authorized users access the system."
    impact 1.0
    tag "cis-rhel7-2.1.1": "5.2.15"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe.one do
        describe sshd_config do
            its('AllowUsers') { should match /[\S|\s]+/ }
        end
        describe sshd_config do
            its('AllowGroups') { should match /[\S|\s]+/ }
        end
        describe sshd_config do
            its('DenyUsers') { should match /[\S|\s]+/ }
        end
        describe sshd_config do
            its('DenyGroups') { should match /[\S|\s]+/ }
        end
    end
end
```

The use of “describe.one” is effectively an OR test. Only one of these matches must be found in order for the control to be met.

REMOTE ACCESS MANAGEMENT

FFIEC Information Security considerations for Remote Access Management (§ II.C.15(c)) require that organizations develop policies to ensure that remote access by employees, whether using institution or personally owned devices, is provided in a safe and sound manner. Such policies and procedures should define how the institution provides remote access and the controls necessary to offer remote access securely. Management should employ several measures including use of robust authentication methods for access and encryption to secure communications, as well as auditing and logging of all remote access communications.

ENSURE SSH DOES NOT PERMIT LOGIN TO ACCOUNTS WITH EMPTY PASSWORDS

Disallowing remote shell access to accounts that have an empty password reduces the probability of unauthorized access to the system. The PermitEmptyPasswords parameter specifies if the SSH server allows login to accounts with empty password strings.

```
control "cisecurity.benchmarks_rule_5.2.9_Ensure_SSH_PermitEmptyPasswords_is_
disabled" do
    title "Ensure SSH PermitEmptyPasswords is disabled"
    desc "The PermitEmptyPasswords parameter specifies if the SSH server allows
login to accounts with empty password strings. Rationale: Disallowing remote
shell access to accounts that have an empty password reduces the probability of
unauthorized access to the system"
    impact 1.0
    tag "cis-rhel7-2.1.1": "5.2.9"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe sshd_config do
        its('PermitEmptyPasswords') { should eq 'no' }
    end
end
```

This control uses InSpec's built-in sshd_config resource to inspect the configuration of the OpenSSH daemon.

2. ENSURE SSH LOGLEVEL IS SET TO INFO

SSH provides several logging levels with varying amounts of verbosity. DEBUG is specifically **not** recommended other than strictly for debugging SSH communications since it provides so much verbose information that it becomes difficult to identify important security information. Setting the log level to INFO records login activity of SSH users. In situations like Incident Response, it is important to determine when a particular user was active on a system. The logout record can eliminate those users who disconnected, which helps narrow the search field.

```
control "cisecurity.benchmarks_rule_5.2.3_Ensure_SSH_LogLevel_is_set_to_INFO" do
    title "Ensure SSH LogLevel is set to INFO"
    desc "The INFO parameter specifies that login and logout activity will be
logged. Rationale: SSH provides several logging levels with varying amounts of
verbosity. INFO level is the basic level that only records login and logout
activity of SSH users."
    impact 1.0
    tag "cis-rhel7-2.1.1": "5.2.3"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe sshd_config do
        its('LogLevel') { should eq 'INFO' }
    end
end
```

This control uses InSpec's built-in sshd_config resource to inspect the configuration of the OpenSSH daemon.

3. ENSURE SSH MAXAUTHTRIES IS SET TO 4 OR LESS

Setting the MaxAuthTries parameter to a low number will minimize the risk of successful brute force attacks to the SSH server. While the recommended setting is 4, your setting may be different. The MaxAuthTries parameter specifies the maximum number of authentication attempts permitted per connection. When the login failure count reaches half the number, error messages will be written to the syslog file detailing the login failure.

```
control "cisecurity.benchmarks_rule_5.2.5_Ensure_SSH_MaxAuthTries_set_to_4_or_
less" do
    title "Ensure SSH MaxAuthTries is set to 4 or less"
    desc "The MaxAuthTries parameter specifies the maximum number of
authentication attempts permitted per connection. When the login failure count
reaches half the number, error messages will be written to the syslog file.
Setting the MaxAuthTries parameter to a low number will minimize the risk of
successful brute force attacks to the SSH server."
    impact 1.0
    tag "cis-rhel7-2.1.1": "5.2.5"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe sshd_config do
        its('MaxAuthTries') { should cmp <= 4 }
    end
end
```

This control uses InSpec's built-in sshd_config resource to inspect the configuration of the OpenSSH daemon to look for a setting that is equal to or less than 4.

APPLICATION ACCESS

FFIEC Information Security requirements for Application Access Management (§ II.C.15(b)) ensure that mission-critical applications incorporate appropriate access controls that restrict which functions are available to users and other applications. Some security software programs integrate access control between the operating system and application layers. Such controls are useful when applications do not have their own access controls or when the organization can provide controls to supercede an application's native access controls. Requirements include logging access and events to monitor and respond to anomalies and alerts. The examples below use system software to track and monitor suspicious activity that may indicate attempts to gain unauthorized access to system applications.

1. ENSURE RSYSLOG OR SYSLOG-NG IS INSTALLED

The security enhancements of rsyslog and syslog-ng such as connection-oriented (i.e. TCP) transmission of logs, the option to log to database formats, and the encryption of log data en route to a central logging server, justify installing and configuring the package. These alternative logging systems are recommended replacements to the original syslogd daemon.

```
control "cisecurity.benchmarks_rule_4.2.3_Ensure_rsyslog_or_syslog-ng_is_
installed" do
    title "Ensure rsyslog or syslog-ng is installed"
    desc "The rsyslog and syslog-ng software are recommended replacements to the
original syslogd daemon which provide improvements over syslogd. The security
enhancements of rsyslog and syslog-ng such justify installing and configuring
the package."
    impact 1.0
    tag "cis-rhel7-2.1.1": "4.2.3"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe.one do
        describe package("rsyslog") do
            it { should be_installed }
        end
        describe package("syslog-ng") do
            it { should be_installed }
        end
    end
end
```

The use of "describe.one" ensures that either rsyslog OR syslog-ng must be installed in order for the requirements of this control to be met.

2. ENSURE PERMISSIONS ON ALL LOG FILES ARE CONFIGURED

It is important to ensure that log files have the correct permissions to ensure that sensitive data is archived and protected. Log files stored in /var/log contain logged information from many services on the system, or on centralized logging hosts, data from services on other systems.

```
control "cisecurity.benchmarks_rule_4.2.4_Ensure_perms_on_all_logfiles_
configured" do
    title "Ensure permissions on all logfiles are configured"
    desc "Log files stored in /var/log/ contain logged information from many
services on the system, or on log hosts others as well. Rationale: It is
important to ensure that log files have the correct permissions."
    impact 1.0
    tag "cis-rhel7-2.1.1": "4.2.4"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    command('find /var/log -type f').stdout.split("\n").each do |log_file|
        describe file(log_file) do
            it { should_not be_writable.by('group') }
            it { should_not be_executable.by('group') }
            it { should_not be_readable.by('other') }
            it { should_not be_writable.by('other') }
            it { should_not be_executable.by('other') }
        end
    end
end
```

This control uses InSpec's file resource to ensure that all conditions are true in order for the requirements of this control to be met.

3. ENSURE SUSPICIOUS PACKETS ARE LOGGED

When enabled, this feature logs packets with un-routable source addresses to the kernel log. Enabling this feature and logging these packets allows an administrator to investigate the possibility that an attacker is sending spoofed packets to their system.

```
control "cisecurity.benchmarks_rule_3.2.4_Ensure_suspicious_packets_are_logged"
do
    title "Ensure suspicious packets are logged"
    desc "When enabled, this feature logs packets with un-routable source
addresses to the kernel log. Rationale: Enabling this feature and logging these
packets allows an administrator to investigate the possibility that an attacker
is sending spoofed packets to their system."
    impact 1.0
    tag "cis-rhel7-2.1.1": "3.2.4"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe kernel_parameter('net.ipv4.conf.all.log_martians') do
        its('value') { should eq 1 }
    end
    describe kernel_parameter('net.ipv4.conf.default.log_martians') do
        its('value') { should eq 1 }
    end
end
```

This control uses InSpec's built-in kernel_parameter resource to test kernel parameters on Linux platforms. These parameters are located under /proc/cmdline.

APPLICATION SECURITY

The FFIEC Information Security booklet covers a number of considerations for ensuring Application Security (§ II.C.17). Implementation of application level security is particularly customized to the needs of your particular application architecture. However, there are three general concerns that can be addressed to encompass the various facets involved with ensuring application level security in the IT environment. Application level security can be enhanced using externally available system controls to reduce exposure to unauthorized access, controls can be implemented to inspect the configuration of custom applications, and controls may also be implemented during the software development process to ensure those custom applications are not introducing new vulnerabilities as new features are deployed.

1. ENSURE FIREWALL RULES FOR ALL OPEN PORTS

Applications that are internet-facing can be further protected through additional system-level controls like firewalls to limit inappropriate access or connections to the application or other areas of the network. You can ensure that any ports that have been opened on non-loopback addresses need firewall rules to govern traffic. Without a firewall rule configured for open ports, the default firewall policy should drop all packets to these ports.

```
control "cisecurity.benchmarks_rule_3.6.5_Ensure_firewall_rules_for_open_ports"
do
    title "Ensure firewall rules exist for all open ports"
    desc "Any ports that have been opened on non-loopback addresses need firewall
rules to govern traffic. Rationale: Without a firewall rule configured for open
ports default firewall policy will drop all packets to these ports."
    impact 1.0
    tag "cis-rhel7-2.1.1": "3.6.5"
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    port.where { protocol =~ /.*/ && port >= 0 && address =~
/^(?!127\.0\.0\.1|::1|::).*$/ }.entries.each do |entry|
        rule_inbound = "-A INPUT -p #{entry[:protocol]} -m #{entry[:protocol]}
--dport #{entry[:port]} -m state --state NEW,ESTABLISHED -j>
        rule_outbound = "-A OUTPUT -p #{entry[:protocol]} -m #{entry[:protocol]}
--sport #{entry[:port]} -m state --state ESTABLISHED -j A>
        describe iptables do
            it { should have_rule(rule_inbound) }
            it { should have_rule(rule_outbound) }
            end
        end
    end
```

This control uses Ruby code to create a list of all open system ports listening to non-loopback addresses. The control also gathers a list of all currently implemented IPTables rules. The control ensures each open port listening on non-loopback addresses has an associated IPTables rule with both an inbound and outbound entry.

2. CUSTOM CONTROLS TO INSPECT YOUR APPLICATION

InSpec includes dozens of built-in resources to inspect the configuration of a variety of application architecture constructs ranging from configuration files, to open ports, file contents, Docker containers, or cloud provider service objects. InSpec also has built-ins that allow for creation of custom resources either via Ruby or by using system-level commands. While some examples are included in earlier sections of this guide, there are many additional ways to use InSpec in customized ways specific to the needs of your application. For further examples, check the InSpec documentation for more details.

3. PERFORM CONTINUOUS ASSESSMENTS OF THE APPLICATION DEVELOPMENT PROCESS

A secure software development life cycle is one that ensures applications have the necessary security controls at all times. FFIEC requires that organizations should ensure that all applications are securely developed by verifying controls have been developed and implemented appropriately throughout the software development lifecycle. A key component of that verification is to perform ongoing risk assessments that consider the adequacy of application-level controls in light of changing threat, network, and host environments.

Adopting a continuous compliance approach with Chef Automate allows you to both ensure a more secure software development process and to painlessly satisfy audit requirements at any time. Chef Automate allows you to detect fleet-wide compliance shortcomings and prioritize areas for remediation. Setting up a continuous test and remediation cycle can help you make compliance part of the development process. You can eliminate wasteful pre-production security scans that slow down development and create rework for engineers by integrating those scans regularly into your development processes. You can communicate clear policy and procedural intent by replacing rules written in English (e.g. in imprecise formats like PDF or Excel) with executable code that enables collaboration with development teams.

New vulnerabilities are discovered daily. Rather than waiting for a vendor to issue opaque detection rules in a quarterly fix pack, InSpec's open language allows you to write and publish vulnerability detection code that same day and immediately use it to keep your company safe. Making this type of regular test and remediation cycle an integrated part of your work has the added benefit of meeting the requirements in this section of the FFIEC.

DATA SECURITY

Financial institutions gather collections of sensitive consumer and company data. Databases generally have built-in controls and mechanisms configured to provide varying levels of protection. Organizations should strive to protect stored information from theft or unauthorized access. Controls commensurate with the sensitivity of stored data should be implemented and enabled. Data can be accessed by database users that are people (employees, customers, contractors) or other applications. FFIEC Information Security requirements for Database Security (§ II.C.18) assure that users have varying levels of access and authorization, especially to sensitive data. In our examples below, we'll look at a few ways to build custom controls using InSpec resources to ensure proper segmentation of access rights amount different users.

1. ENSURE A NON-AUTHORIZED USER CANNOT REACH CARDHOLDER DATA

In this example, we could write a control to ensure that unauthorized database users cannot access credit card numbers.

```
control "mycompany.custom_rule1.0.0_Ensure_non_privileged_users_cant_access_cc_numbers" do
    title "Ensure that non-privileged database users do not see CC numbers"
    desc "A legitimate and authorized database user account should not be able to access credit card numbers if they are not specifically authorized to do so. Rationale: Only privileged accounts should be able to access cardholder data."
    impact 1.0
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    %w[sys system alice].each do |unprivileged|
        describe oracledb_session(user: unprivileged, password:'password', service:'ora01.mycompany.com').query('SELECT creditcard FROM accounts').rows do
            its('count') { should eq 0 }
        end
    end
end
```

This example control would connect to a known production Oracle database (ora01.mycompany.com) and attempt to authenticate as the users "sys", "system", and "alice". In this example we presume that these users should not have access to stored credit card information. Therefore, when they attempt to select these rows there should be no data returned.

2. ENSURE CVC DATA IS NOT INADVERTENTLY STORED IN SYSTEM LOGS

A card validation code (CVC) is used to protect "card-not-present" transactions (e.g. internet or telephone) where the consumer and the card are not present. This data is used by applications during initial authorization but it should not be stored after its initial use. This example control examines a sample application log file to ensure that CVC entries do not appear in payment authorization transaction logs.

```
control "mycompany.custom_rule1.1.0_Ensure_no_CVC_entries_exist_in_auth_logs" do
    title "Ensure that no entries matching CVC codes are present in application
logs"
    desc "The application server may write verbose logs in verbose debugging
mode. Care should be taken to ensure that no entries in application server logs
contain CVC codes in them. Rationale: CVC codes should not be inadvertently
stored in the system after their initial use."
    impact 1.0
    tag "level": "1"
    tag "type": ["Server", "Workstation"]
    describe file('/path/to/my_app/auth_payment.log') do
        its('content') { should_not match(/^*CVC2.[0-9][0-9][0-9][0-9]{0,}) }
        its('content') { should_not match(/^*CVV2.[0-9][0-9][0-9][0-9]{0,}) }
        its('content') { should_not match(/^*CID.[0-9][0-9][0-9][0-9]{0,}) }
        its('content') { should_not match(/^*CAC2.[0-9][0-9][0-9][0-9]{0,}) }
    end
end
```

This example control uses the built-in "file" resource to examine the contents of the "auth_payment.log" file to look for any entries that might contain information about the various types of CVC codes used followed by a 3 or 4 digit code, which would likely be the code itself. We should not find any of these codes.

CONCLUSION

This guide is not a definitive set of controls necessary to fulfill all of the requirements in a FFIEC audit. This guide is meant to illustrate what these controls do, how they apply to FFIEC Information Security requirements, and provide a practical guide to understanding which CIS benchmarks map to which FFIEC requirements.

In practice, members of both technical and compliance teams working with systems in the IT environment would not have to compose these tests from scratch. A majority of the tests in this whitepaper are a part of the CIS benchmarks for Red Hat Enterprise Linux 7, available for use with a Chef Automate license. Additional profiles referenced in this whitepaper are available in the Chef Supermarket. An accompanying demo InSpec profile for this whitepaper illustrates how teams can quickly reference and include controls from existing profiles.

Our guide demonstrates some of the core functionality available when using InSpec and Chef Automate to implement compliance reporting controls. By adopting a continuous compliance approach, you can quickly answer audit questions at any time--not just quarterly or yearly. By customizing these controls to meet the needs of your particular environment, you can enter an audit cycle knowing your exact compliance posture and avoid being surprised by auditors finding unexpected weak points in your environment.

To see how Chef Automate can help you achieve continuous compliance and reduce the time your teams spend on fulfilling audit requests, visit https://www.chef.io/automate/.

LEGAL DISCLAIMER

This document is for informational purposes only. It is not legal advice and should not be relied upon by the reader to achieve compliance specific to any particular rule or regulation. Please consult legal counsel for guidance on legal regulatory compliance. CHEF MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION PROVIDED IN THIS DOCUMENT.