Global Information Assurance Certification Paper

An Organic Approach to Implementing the Critical Security Controls

GIAC (GCCC) Gold Certification

Author: Jim Hendrick, jrhendri@roadrunner.com
Advisor: Stephen Northcutt
Accepted: 12/20/2015

Template Version September 2014

Abstract

This paper describes a method (almost a philosophy) for using the Critical Security Controls (CSCs) to drive long term improvement by carefully choosing specific metrics linked with operational processes. In contrast to formal process models, this method begins with identifying existing areas where (often small) changes can be used as starting points. Several examples are given using specific controls, concepts for driving change are presented, and the use of metrics as an underlying mechanism is discussed. The resulting "organic" approach promotes continuous improvement by taking advantage of natural behavioral tendencies of people and organizations.

1. Introduction

The Critical Security Controls (CSCs) describe a set of specific actions designed to improve an organization's ability to resist or recover from information security incidents ("CIS critical security controls," 2015). Developed and maintained through analyzing current breaches and reported security events, the CSCs recommend activities that can be (largely) automated that are designed and prioritized to produce defenses against observed attack behavior.

Recent surveys have shown that organizations are increasingly choosing the CSCs as part of their security strategy. Yet barriers continue to slow implementations. A 2013 survey indicated that 73% of respondents had or were planning to adopt the controls, yet identified that "Operational silos within the IT security organization and other business departments are still the greatest impediment to implementing repeatable processes based on the Controls" (Pescatore, 2013). And a 2014 survey reported that 90% were or were planning to implement but also found "Disconnect between IT/Operational silos" as a barrier to adoption (Tarala, 2014).

The goal seems simple; reduce risk by implementing or improving a set of security controls. And the CSCs themselves provide very prescriptive sub-controls with accompanying measures and metrics. So why are organizations struggling with the implementation?

The approach presented here looks for ways to address some of these barriers by identifying forces within the "operational silos" and hopefully change them from "impediments and barriers" into forces for change.

2. Creating an Initial Assessment

The method presented here uses the measurements and metrics within the CSCs themselves. Designed to track progress over time, these metrics are categorized as either "Automation":

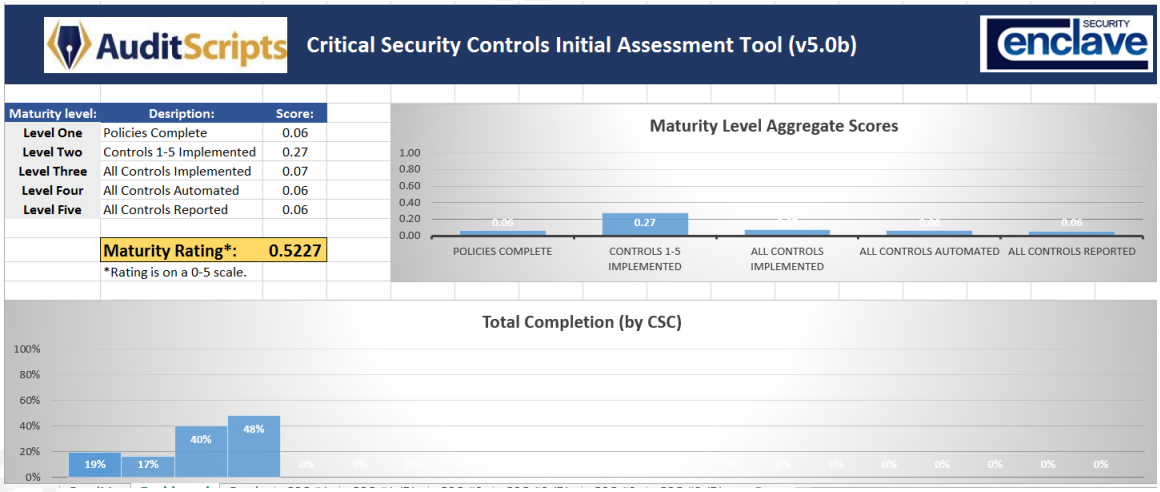
Jim Hendrick, jrhendri@roadrunner.com

"How many unauthorized devices are presently on the organization's network (by business unit)?"

Or "Effectiveness:

"How long does it take to detect new devices added to the organization's network?"

Start by conducting an initial assessment. This can make use of formal methodologies which provide framework for repeating this assessment over time (Riffat, 2015). However a great deal of benefit can be gained quickly simply by a subjective review by your security team. A useful resource for helping create and document this baseline is the AuditScripts "Critical Security Controls Manual Assessment Tool" (Tarala, 2014). By working through this tool with representatives from your internal security teams (for example Security Operations Center / SOC, Risk Management, Compliance / Audit) you can come up with an initial overall profile:



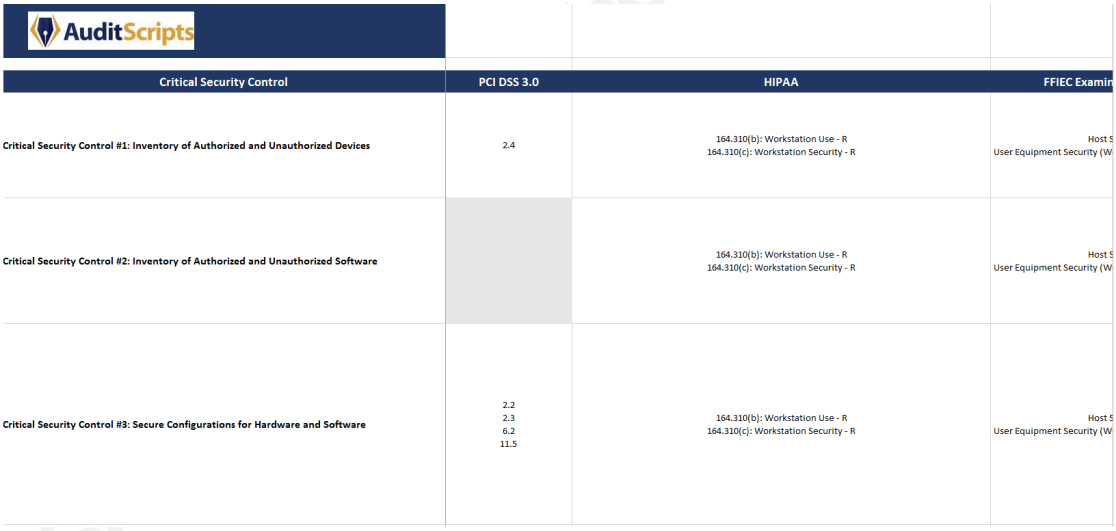
There are two primary objectives of this initial effort. The first is to get an idea where the organization is with respect to the controls, but more importantly to identify teams that operate current controls and begin understanding their processes. In order to prevent operational silos from inhibiting your implementation effort, it is critical to understand what forces influence these teams currently. Common examples of these processes include

- Ticketing systems - Operational teams are measured by number of tickets closed.

Jim Hendrick, jrhendri@roadrunner.com

- Incident operations processes – Amount of time spent per incident.
- Regulatory or compliance efforts – Artifacts that need to be provided for reporting.

These directly impact staffing and budget, and by choosing from these areas to generate the CSC metrics, departments will naturally look for areas to improve. Be sure to include the teams and activities from regulatory or compliance requirements in your effort. The SANS surveys in 2013 and 2014 showed compliance as a major driver for implementing the CSCs (38% and 42% respectively). Automating and sharing this data with the CSCs will directly support these efforts. The designers of the CSCs have created a body of work designed to be "mapped" against these frameworks. One tool is AuditScripts "Critical Security Control Master Mappings" (Tarala, 2014). This screenshot shows three of twenty one frameworks mapped against critical controls.



AuditScripts

Critical Security Control	PCI DSS 3.0	HIPAA	FFIEC Examin
Critical Security Control #1: Inventory of Authorized and Unauthorized Devices	2.4	164.310(b): Workstation Use - R 164.310(c): Workstation Security - R	Host S User Equipment Security (W
Critical Security Control #2: Inventory of Authorized and Unauthorized Software		164.310(b): Workstation Use - R 164.310(c): Workstation Security - R	Host S User Equipment Security (W
Critical Security Control #3: Secure Configurations for Hardware and Software	2.2 2.3 6.2 11.5	164.310(b): Workstation Use - R 164.310(c): Workstation Security - R	Host S User Equipment Security (W

For example, CSC #1 would provide evidence for PCI DSS v3 section 2.4: "Maintain an inventory of system components that are in scope for PCI DSS"
In this way, the CSCs should be seen as the "what and how" aspects of implementing controls that produce common data to be presented in whatever format is required.

3. Collecting Metrics & Identifying Existing Processes

The initial assessment will identify and document activities and data that can be associated with the CSCs. This need not be an exhaustive effort, but should provide a

reasonable idea of the teams, controls, data and processes that exist. Since "operational silos" have been identified as a potential obstacle, identifying them and understanding their perspective is where you should focus your effort.

The next step begins the technical work of identifying measurements to collect for each specific control. But before you can actually use data to guide improvements, you need to have a way to collect and analyze that data.

3.1. Establishing a Repository for Measurements

A repository for this data may be an existing database or perhaps a new system to collect, merge or transform data to fit reporting requirements. One model that works particularly well is to use what is called "referential data". This is the data that is needed by a SIEM to provide context specific to your organization, and includes things like system inventory and classification, users and privileged groups, and other data needed by your use-cases (Hendrick, 2014). Regardless of how it is implemented, it is important that you use the same data as the operational use-cases (SIEM) for your reporting. This is absolutely critical; Make sure your security systems use consistent data. It is OK to manually analyze data as part of the initial assessment. But using manually manipulated data will ultimately lead to confusion and doubt since reports from different operational sources for the same time period will not match and can lead to completely logical (but wrong) conclusions, (much like the hapless peasants " If... she... weighs the same as a duck.. she's made of wood. .. And therefore? … A witch! "(Monty Python, n.d.)) Even if there are gaps in the data initially, simply present and qualify the sources of that data. Don't get hung up on making it perfect. If you work directly with operational teams to identify and present the data you become allies in the overall security control effort.

A separate database for these measurements and the referential data has several advantages. Figure 1 shows the basic components needed for log collection and retention, reporting and alerting, analysis and investigations and collection and presentation of control measurements and metrics. Note the different retention periods associated with each component. While the actual numbers may differ for each organization, each function has a different requirement of how long the data is useful. In this model, the

relational database is used for metrics as well, since much of the data for these metrics either resides there directly or can be produced automatically by the SIEM.

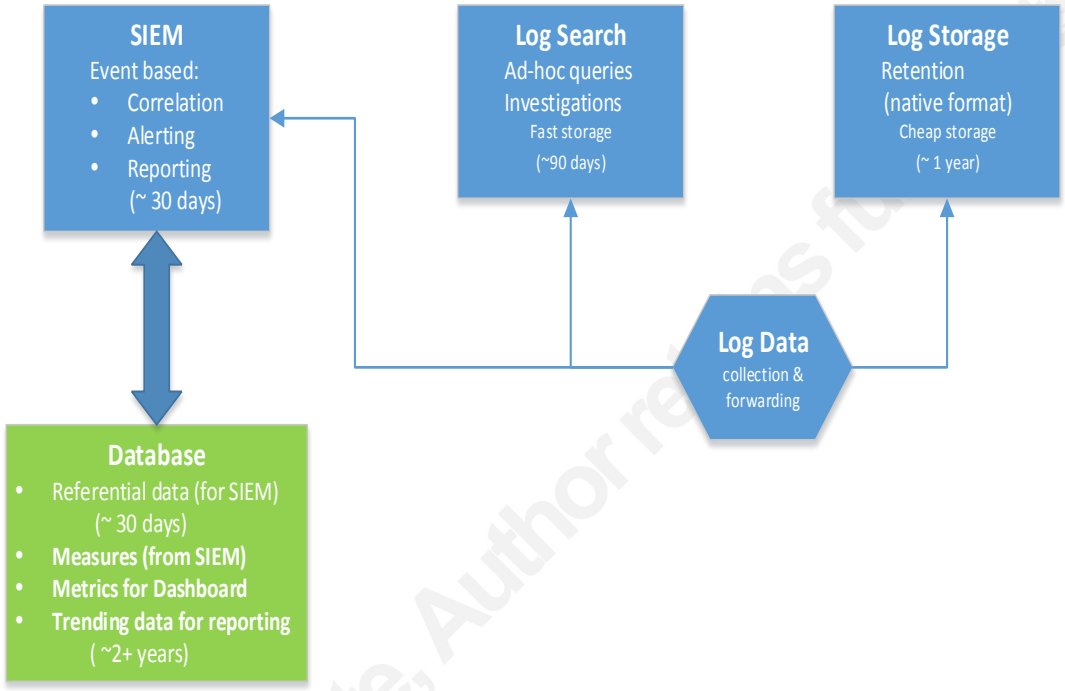


Figure 1: Security Data Functions

3.2. Example using Control 1: Inventory of Devices

In every organization there is a hardware inventory system created in order to track capital expenses and depreciation over time. And it should include a process that adds systems to inventory when they are purchased or deployed address when something is fully depreciated or otherwise "off the books". This data source provides departmental "ownership" along with serial numbers or other device identification data.

In addition, there is typically a process for tracking systems on the network. Often part of the work flow for teams responsible for the data center, networking, desktop or some combination. This data source includes IP addresses and potentially network or data center location, and also includes a process for removing systems.

The initial assessment should have identified these sources, teams and processes. You should know how items get added to and removed from inventory, as well as department transfers, re-use of (fully depreciated) systems, disposal, etc. When meeting with these teams, you should understand "How do you reconcile this data today? Is it audited? Are tickets cut to reconcile the data? Is this done regularly, and how often?

This control needs to associate each IP address with a department. The referential database coupled with the SIEM can then create alerts or reports comparing this (merged) data with active and passive scans to answer:

"How many unauthorized devices are presently on the organization's network (by business unit)?"

This first metric presents an opportunity to address common subtleties of these controls. For instance, how do you define "unauthorized"? On the surface, "unauthorized" may simply be defined as "not present in the inventory system". But what devices are in the inventory system? Can system names and IP addresses be tied to specific hardware? What about dynamically assigned addresses? Devices with multiple NICs? Do you track virtual systems? (How) are they represented in inventory?

The complexity of this seemingly simple control metric is precisely the point. The goal is to measure and report on the control as it exists. You choose how to collect and report on the data. Initially, you may limit scope based on the data available, using the reporting to focus the discussions on where improvement is needed. And by involving the various teams these discussions and recommendations will go more smoothly.

3.2.1. Example – Normalizing IP data from multiple sources

CSC #1 states that "both active tools that scan through network address ranges and passive tools that identify hosts based on analyzing their traffic should be used" Since the focus is to start with what you have, consider this simple example.

There are many tools for actively scanning your network. But simple nmap has a command-line option to ping network ranges and return a list of systems that respond. However any active scanner depends on being able to send a stimulus to an address and

receive a response. Internal firewalls and protected "enclaves" need to be taken into account when determining where to position your active scanners.

Similarly, there are tools designed specifically for passively listening for devices on your network. But you can use existing data very nicely as well using simple techniques. Anything that generates log data including IP addresses can be a valid source, as can netflow collected from your routers. And also like active scanners, the accuracy and completeness of the data depends on the perspective of these "scanners"

Whatever the active and passive data, it need to be compared to an inventory. Due to the widely different data formats, this can be a challenge itself. One solution to "normalize" this into IP addresses using the SiLK suite of netflow tools (Carnegie Mellon University, n.d.). In this example, combining IP addresses from nmap and syslog data, then comparing that with the inventory to identify "unknown" systems (yet being able to tie them back to the source of the data for investigation and follow-up).

See the Appendix for more details from this example using command line tools from the SiLK tool suite.

3.2.2. Adding a measure of effectiveness

So far the discussion has focused on data that can be obtained automatically. This is intentional in the design of the CSCs, and is essential to implement controls at any large or dynamic scale. However it does not tell the whole story Examining two "Effectiveness metrics" will make additional points:

"How long does it take to detect new devices added to the organization's network?"

This can be answered in at least two possible ways. A basic response would be to report how often (active) network scans are performed and how often passive detection covers the same network space. A more detailed answer would include how long it takes for a person to receive an alert (or report) based on the active and passive data. Both of these answers are more process than technology, and these (Effectiveness) metrics often need to be manually generated and therefore get reported on a longer cycle.

A second effectiveness metric asks: "How long does it take to isolate/remove unauthorized devices from the organization's network?"

This type of metric directly addresses process. And how you select and report on the data should include operational work flow around what actually happens once an unauthorized device is found.

On the surface it may be seen as simply as "How long does it take to disable a switch port?" But more examination is worthwhile. The effectiveness of this control addresses how well the inventory is maintained in addition to how quickly devices can be removed. An imperfect inventory can miss devices that have legitimate reason to be on the network, and process should not immediately disable an IP address simply because it is not in your inventory system. (You might get to do that about once.) A verification step is needed that may include phone calls or identifying the physical location of the device to get other information (e.g. an asset tag or a device model and serial number). Then identifying who to contact to determine the device's purpose and steps to feed this data back to the inventory management process. And only if none of this determines a legitimate reason how to treat the device. Disable the network port? Disconnect the power? And how to respond if it is then identified as a valid business server? This should also be guided by whether or not that address appears in other security alerts based on its behavior on the network.

By choosing to include the end-to-end response process time (e.g. time to close an incident ticket) can use this existing "natural process" to create valuable process improvement.

But care must be taken to choose the response. For example, if a ticket is created for each unauthorized device, the SOC may be flooded with tickets. And while to the data should call attention to problems in the inventory process, overwhelming the SOC with false positives does more harm than good. To avoid these "operational silos" from being "barriers to adoption" the first process should work directly with the SOC to define a process that improves the control while managing resources appropriately. A regular report of all "unaccounted for" devices and a fixed amount of analyst time to review that report will do this and your effectiveness metric will reflect this improvement.

Jim Hendrick, jrhendri@roadrunner.com

3.3. Control 2: Authorized and Unauthorized Software

Like control #1, this control sounds simple at the outset. And processes exist around software management. However from the security risk perspective, definitions of what software is authorized or unauthorized may not be completely clear.

This control identifies application whitelisting as a "quick win" (note that this does not mean it is easily or quickly implemented, rather that *once in place* the benefits are quick to appear.) However even without this technology it is possible to obtain metrics from existing processes. And the effort expended gathering this data applies directly to whitelisting solutions as well.

From a security perspective, the consideration is in identifying software that increases risk to your organization. The cyber kill-chain (Hutchins, Cloppert, & Amin, n.d.) defines the steps of the intrusion process to be: reconnaissance, weaponization, delivery, exploitation, installation, command & control and action on objectives. And unauthorized software can play a part in each of these areas.

Defining what is authorized is not easy. While many business users have a clearly defined set of applications needed to do their jobs, this is not the entire story. Unauthorized tools can be used to identify vulnerabilities, yet security groups need scanning and testing tools for legitimate functions. Compilers can be used to build malicious software and debuggers can identify weaknesses that could be exploited, but development teams needs to build and test software. And remote access tools, encryption libraries, and other utilities can be used to establish or maintain malicious access, but IT staff need these utilities for remote access & management. Fortunately, building an all-inclusive list not necessary when implementing this control. The goal is to begin, and to choose metrics that will influence the desired change.

3.3.1. Start by defining Authorized and Unauthorized

Whether an organization has (or is planning to have) technology that actively prevents installation or execution of unauthorized software (i.e. whitelisting technology), the first requirement is to be able to create and maintain this definition. The initial

assessment will identify tools and teams already working in this area. And looking at the first automation metric will guide selection of the data:

"How many unauthorized software applications are presently located on business systems within the organization (by business unit)?"

Clearly a list of systems by business unit is required, (obtained from CSC #1.) And a way to identify and count unauthorized applications per system. Typically at least two IT processes exist that can start to provide this data:

Organizations track purchased and licensed software for financial reasons. Existing processes include managing allocated licenses and tracking purchased software to predict future needs. This data will comes from a specialized scanning tool.

In addition to financial management, support teams need to keep software updated and manage multiple versions across the environment. Often this is maintained as a separate set of data and using their own scanning application.

For this metric, another scanning tool isn't necessary if the data needed is already collected by the other teams. What is required is to define and identify "unauthorized software applications" from the data that is present.

As part of the assessment, determine what details are provided by which scans. From the financial perspective, tools, libraries, applications, etc. identified as "free software" often fall outside their scope. Similarly, if software is available for users to download directly, it may be ignored by the support team as they are primarily measured by the patching and maintenance processes. However even if the processes do not address these categories today, the scans should provide sufficient data for the measurement.

Two primary methods for building lists of authorized applications are identified in a NIST "Guide to Application Whitelisting" (Sedgewick, Souppaya, & Scarfone, 2015). One is to begin with information provided by a vendor including file characteristics for specific versions. The second relies on building this baseline data by scanning a known good system (one for each standard system image allowed on your network). In addition, services exist that provide "reputation" information about software (and its sources).

Jim Hendrick, jrhendri@roadrunner.com

But regardless of the source of the data, both require the organization to define what is acceptable. And both of these require regular maintenance, whether via an update service from the vendor or by an internal process to regularly add new versions (including patches) to the software inventory.

Whether using vendor provided information or not, a simple way to begin is to define three categories. Software should fit into:

- Allowed - software generally permitted for use.
- Not allowed - software generally prohibited (without specific approval.)
- Unclassified - may be the majority discovered initially.

To create an initial whitelist, run the scanners against freshly built standard images, using this as the standard base of authorized software. This should include user desktops, standard server builds and both hardware and virtual deployments. Similarly, add all software available for internal use (e.g. from an internal "application portal") to the list. This should be sufficient to start. Control the scope so that the process has a greater chance of success over time.

The blacklist is actually less critical, since almost by definition if software is not on an approved list its use should be investigated. However many vendors provide categories that are immediately questionable including many security tools (network or vulnerability scanners, application penetration testing tools, code analysis tools, etc.) If this data is available, it can "jump start" a blacklist.

Most importantly, define the work flow around this control. The referential database can maintain the approval status for each software title. And the SIEM can compare the regular scan data to this and create reports of software per system. Depending on the number of "exceptions" for systems that are allowed to have normally unauthorized software (like specialized security tools), this data can be maintained in the same referential database, or even manually in the beginning. As long as it will support the need to validate and respond to software that is discovered and not on the whitelist. Security should do the initial assessment of identified "blacklisted" software, since improper handling even at the identification phase can tip off a malicious person, lose

evidence and make the situation worse. But for many incidents, the actual removal task may be done by another group. As with any new use-case, begin with a regular report that is assigned a fixed amount of time and avoid a flood of automated alerts. And the process must include a maintenance step to move discovered software from the "grey" area into one of the others. This should be a joint effort, with security risk analysts making the final decision on specific software categories.

Clearly this simplified process does not address different roles or systems. But the data collected will allow this as the control matures. Initially, getting this granularity right is less important than working with the teams that would be affected. And whether it is done as a preventive control (i.e. whitelisting software that prohibits the installation or execution of unauthorized applications) or a detective one (simply reporting on unknown or unauthorized software), the process and the control will improve as it matures. The "grey" list will become smaller, groups of users and systems will have separate approved lists, and different levels of risk can start to be addressed

3.3.2. Effectively managing applications

Building in the right metrics is necessary to improve the process of identifying and remove unauthorized software. There are four effectiveness metrics provided as part of the CSCs:

1. How long does it take to detect new software installed on systems in the organization (time in minutes)?
2. How long does it take the scanners to alert the organization's administrators that an unauthorized software application is on a system (time in minutes)?
3. How long does it take to alert that a new software application has been discovered (time in minutes)?
4. Are the scanners able to identify the location, department, and other critical details about the unauthorized software that is detected (yes or no)?

Reading these quickly, it appears that the first three are all asking the same question. However, when thinking about how the control would be implemented identifies crucial differences. And how these measurements are used can be extremely useful when communicating the effectiveness of this control at a given organization.

Jim Hendrick, jrhendri@roadrunner.com

The first one essentially imposes the requirement that the scanning system track history, so that new installations of software will be identified, along with how frequently this happens. So reporting on how often each system is scanned (i.e. how often the entire organization is scanned) should address this. The second adds the aspect of determining "unauthorized", so the measurements should addresses whether or not the response (actions taken) is triggered automatically for every system or manually by a review process. And the third implies that alerts are generated from the scanning, so it can be used as a measure of how quickly changes are made to the approved & unapproved lists. The fourth essentially reinforces the data needed in the system inventory (CSC #1).

By selecting data that represents key aspects of this control organizations can focus visibility to specific processes. The main benefit of the critical controls as a framework is to help organizations not only implement controls, but *instrument* them in ways that guide continuous improvement.

4. Driving Change

The point of this paper is not to present fundamentally new research, but rather to address the problem posed by "operational silos". The critical controls are used as a framework to identify the teams and processes needed to implement controls across organizational boundaries. The main themes presented here include:

- Use of an initial assessment to identify critical teams and processes.
- Work directly with these teams to understand how the organization arrived with the controls it currently has. This is perhaps the most critical part of eliminating barriers.
- Learn how teams are measured. Is it based on trouble tickets? Do the teams need to do things a certain way to meet a business need? What is their current workload? Find ways both teams can get what they want.
- Focus on improving processes rather than implementing more technology.

Jim Hendrick, jrhendri@roadrunner.com

The recognition of what makes up the "culture" of an organization is often helpful. For a large corporation with tens of thousands of employees and many decades of history, recognize that processes have developed within that company, and many employees have years working with those processes. This corporate "inertia" can be a huge impediment to change. But if you can identify the underlying forces that exist, you may gain insight into how they can be changed from opposition into support.

Younger or smaller organizations are less susceptible to this type of cultural inertia, but may resist changes that impose "control". They may believe their success is due to their ability to react quickly, and control is often seen as against to their culture. But in these organizations there is often more willingness to be "data driven". It may be easier in these environments to start by identifying and collecting the data rather than focusing on the controls. In these environments, the metrics themselves may drive the change you need.

5. Metrics, Dashboards & Standards Mapping

In order to effectively use the data from each control, it must present the right message to the right audience. Most consultants will ask, when asked to prepare something for presentation, "Who is the audience?" and "What will it be used for?" The CSC metrics should identify where improved controls are needed and provide prioritization. So the audience will be the people who can make the prioritization and resource decisions. Perhaps the CIO and CISO along with other senior directors and potentially even the board of directors. At this level, the core message needed to guide decisions should be communicated on one page.

To do this over a multi-year period, the measurements come directly from the CSCs. But the specific data chosen for those measurements must communicate how well the organization's control processes work. Senior management does not need detailed data, but should see if processes leave gaps in that data.

For the Automation metrics, collect and update metrics on a daily basis. Senior management will expect to see regular changes or may either lose confidence in the data

or lose interest in the report. Note, not all your measurements change daily. But if the metrics are calculated every day, they will reflect changes in the supporting data. Here too, it is important to be able to explain that if questioned.

For the "Effectiveness" metrics, it is likely that manual effort will be needed. The effort needed to integrate data feeds from some of the sources may simply be more than it is worth. Reporting effectiveness metrics at most monthly and at least quarterly should suffice (unless there is a major change you want to call attention to). Senior management will use this to see results of efforts they have approved, and see where additional support is needed.

One piece of information not part of the CSCs, but worth including is some way to report on security efforts as they pertain to the controls. Senior leadership may not be interested in seeing specific data on system inventories, vulnerability scores, or other details. But they certainly do want to see where they are spending money. If this data is available automatically (say from your project tracking system) you could include in in the daily "automation" report. However it would also be acceptable to use it as part of the monthly (or quarterly) "effectiveness" report.

To support the program over several years, the data should demonstrate the effectiveness over this period. Keep the data at least two years, potentially longer. For small (daily) measurements about the controls, storage should not be an issue.

For the CSCs to simplify standards or compliance efforts, the data should be able to support different views for different purposes. For instance, when designing the database and determining what data elements to collect for (as an example) system inventory, an additional field could be added to represent "standards scope". In this way, the reporting database would collect and maintain the data, and be able to directly supply it as evidence for meeting DSS v3 section 2.4: "Maintain an inventory of system components that are in scope for PCI DSS". Similar mapping could also be done to connect specific Critical Controls metrics to other frameworks for presentation to a different audience.

Consider this screenshot prototype of a single page dashboard. The left side shows the CSCs represented against the NIST standard and whether or not it is improving

from the last measurement. The center represents the top three projects associated with each control as a timeline and a project status (red, yellow, green) and a SOC ticket report. Obviously this will be tailored for each organization, but the concept is here:

It is not necessary to show specific measurements for each control on this page. (It is hard to see here, but) Choosing a threshold for each control to be red, yellow or Green, and using an icon to indicate trending better, worse or staying steady is about right. Supporting information including scope (coverage) and confidence (accuracy) should be available for critical elements. Detailed data visualizations that show trends can be presented on separate pages.

Jim Hendrick, jrhendri@roadrunner.com

6. Summary and Conclusions

The CSCs provide a rich set of resources around a well-supported set of security controls. They are maintained in a priority order specifically to address recently reported incidents, and what can reasonably be expected in the near future. Using this as the basis for a long-term security strategy has a large number of advantages, but implementation can be daunting due to the organizational tendency to resist change. Recent surveys have identified "operational silos" as an obstacle, so the focus of this method is to identify and address them early. The strategy presented has shown through examples and allegory how to identify internal forces and choose metrics to drive this change.

The selection of specific measurements from data that already exists gives early visibility across the organization. And the use of a separate tracking database can help in several important areas here. First, it will help identify priorities and direct resources to appropriate security areas. Second, historical data displays the effectiveness over time, guiding further changes including security use-cases, procedural and technological improvements where appropriate. Third, this data can be mapped to regulatory and compliance standards, reducing the effort required by those standards.

By focusing on what controls exist, and how operational teams are measured, metrics can guide continuous improvement in security and help your organization use the Critical Security Controls to become more able to resist and effectively respond to risk.

Jim Hendrick, jrhendri@roadrunner.com

References

CIS Critical Security Controls. (2015). Retrieved December 11, 2015, from https://www.cisecurity.org/critical-controls.cfm

Pescatore, J. (2013). SANS 2013 critical security controls survey. Retrieved from https://www.sans.org/media/critical-security-controls/CSC_Survey_2013.pdfcontrols/CSC_Survey_2013.pdf

Tarala, J. (2014). Critical security controls: From adoption to implementation. Retrieved from https://www.sans.org/reading-room/whitepapers/analyst/critical-security-controls-adoption-implementation-35437

Riffat, M. (2015, April 7). A framework for assessing 20 critical controls using ISO 15504 and COBIT 5 process assessment model (PAM). Retrieved from https://www.sans.org/reading-room/whitepapers/auditing/framework-assessing-20-critical-controls-iso-15504-cobit-5-process-assessment-model-36067

Tarala, J. (2014). Critical security control master standards mapping (v.5b). Retrieved December 11, 2015, from http://www.auditscripts.com/wp-content/uploads/mgm/downloads/80905300.xlsx

Gershon, M. (n.d.). Choosing which process improvement methodology to implement. Retrieved from http://na-businesspress.homestead.com/JABE/Jabe105/GershonWeb.pdf

Tarala, J. (2014). Critical security control manual assessment tool. Retrieved December 11, 2015, from http://www.auditscripts.com/wp-content/uploads/mgm/downloads/82185300.xlsx

Hendrick, J. (2014). Security visibility in the enterprise. Retrieved from https://www.sans.org/reading-room/whitepapers/projectmanagement/security-visibility-enterprise-35442

Monty Python. (n.d.). Monty python and the holy grail: Burn the witch. Retrieved December 11, 2015, from http://www.montypython.net/scripts/HG-witchscene.php

Jim Hendrick, jrhendri@roadrunner.com

Carnegie Mellon University. (n.d.). CERT NetSA security suite: Monitoring for large-scale networks. Retrieved December 11, 2015, from https://tools.netsa.cert.org/silk/

Hutchins, E., Cloppert, M., & Amin, R. (n.d.). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Retrieved December 11, 2015, from http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf

Sedgewick, A., Souppaya, M., & Scarfone, K. (2015, October). NIST special publication 800-167: Guide to application whitelisting. Retrieved December 15, 2015, from http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-167.pdf

Appendix: Normalizing IP addresses with command line tools and the SiLK suite.

The SiLK suite is a set of command line tools designed to manage and analyze netflow data. It includes utilities to build and operate on lists of IP addresses as "sets" that are represented in a very efficient binary file format. For this example, we will use:

- rwsetbuild – creating "set" files from its input
- rwsetcat – displaying the contents of set files as text
- rwsettool – performing "union", "intersection" and "difference" operations

Along with some basic scripting, we will create "set" files for each data source as well as the authorized inventory data. Then the differences in these sets provides lists of addresses along with the data sources that identified them.

For example an active nmap ping scan writing output in "grepable" format omitting name resolution:

```
$ nmap -sn -oG -n nmap-ping.out 10.121.98.0/24

$ head nmap-ping.out
# Nmap 6.49BETA5 scan initiated Tue Dec 08 19:31:26 2015 as:
C:\Program Files (x86)\Nmap\nmap.exe -sn -n -oG nmap-ping.out
10.121.98.0/24
Host: 10.121.98.1 ()    Status: Up
Host: 10.121.98.19 ()   Status: Up
Host: 10.121.98.20 ()   Status: Up
Host: 10.121.98.21 ()   Status: Up
Host: 10.121.98.22 ()   Status: Up
Host: 10.121.98.23 ()   Status: Up
Host: 10.121.98.24 ()   Status: Up
Host: 10.121.98.25 ()   Status: Up
Host: 10.121.98.26 ()   Status: Up

$ tail nmap-ping.out
Host: 10.121.98.230 ()  Status: Up
Host: 10.121.98.235 ()  Status: Up
Host: 10.121.98.239 ()  Status: Up
Host: 10.121.98.243 ()  Status: Up
Host: 10.121.98.247 ()  Status: Up
Host: 10.121.98.251 ()  Status: Up
Host: 10.121.98.252 ()  Status: Up
Host: 10.121.98.254 ()  Status: Up
```

Jim Hendrick, jrhendri@roadrunner.com

```
Host: 10.121.98.81 ()   Status: Up
# Nmap done at Tue Dec 08 19:31:27 2015 -- 256 IP addresses (57
hosts up) scanned in 1.21 seconds
```

A quick bit of command-line (many tools available here) strips out just the IP address:

```
$ cat nmap-ping.out | perl -ane 'if (
($ip)=/.*\s+(\d+\.\d+\.\d+\.\d+)\s+.*/ ){print "$ip\n";};'
```

OR (since the nmap command line says “grepable output”)

```
$ egrep -oe '([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)' nmap-ping.out
```

And then build the “set” file.

```
$ rwsetbuild nmap-ping.out nmap-ping.set
```

Passive data from proxy or firewall logs include addresses seen on your network. Here is the first part of a Bluecoat proxy log line as it appears in syslog:

```
Dec  4 00:00:00 192.168.31.174 2015-12-03 23:59:59 1
10.102.250.167 200 TCP_HIT 22679 560 GET http
global.fncstatic.com 80 <rest deleted>
```

Note there are two IP addresses - One is the specific proxy that handled the request, and the second is the “client IP” – the system making the HTTP request. So the parsing needs to extract only the client. Pretty straightforward with a bit of – well lots of things really. Some rough Perl and “rwsetbuild” does the job:

```
$ cat <logfile> |perl -ane 'if
(($proxy,$client)=/^.*\s+(\d+\.\d+\.\d+\.\d+)\s+.*\s+(\d+\.\d+\.\
d+\.\d+)\s+.*/){print "$client\n";};'   > address-list.txt

$ rwsetbuild address-list.txt  observed-addresses.set
```

Depending on the log, not all addresses will belong to your systems. Other sources like firewall or intrusion detection logs will have sources and destinations that may be external to your network. Set tools will help you. Create a set that describes all addresses that could belong inside your network (including RFC 1918) and use this to filter out external addresses. Here a file listing the CIDR notation for RFC 1918 is used:

```
$ cat rfc
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16

$ rwsetbuild rfc  rfc.set
```

Now using our "log" data, extract all the RFC (internal) addresses from that set.

$ rwsettool --intersect observed-addresses.set rfc.set > rfc-observed.set

Now the "rfc-observed.set" file contains only the addresses found in your logs that have addresses in this range. Of course, you would also include your routable internal addresses along with the RFC ranges. Something like this:

```
$ rwsetbuild internal-networks.txt internal-networks.set
```

And add the RFC addresses

```
$ rwsettool –union internal-networks.set rfc.set > internal-
ranges.set
```

Similarly, create a set file containing your device inventory of IP addresses.

```
$ rwsetbuild inventory-list.txt inventory-list.set
```

This will then be used to identify systems (addresses) found by your passive scanners (i.e. in your logs) that do not appear in your inventory.

There is a choice whether to operate on multiple log files as separate "sets" or join them all in a single large set. You have chosen to compare each source set independently by taking the intersection of each source set with the internal address set. The processing is a bit more iterative, but it gives you the ability to easily identify the source of the data.

For example, given three files (one each from firewall, proxy, and intrusion detection logs) you have three sets. Next extract only the internal addresses:

```
$ rwsettool –intersect firewall.set internal-ranges.set >
firewall-internal.set

$ rwsettool –intersect proxy.set internal-ranges.set > proxy-
internal.set

$ rwsettool –intersect intrusion.set internal-ranges.set >
intrusion-internal.set
```

Jim Hendrick, jrhendri@roadrunner.com

This now identifies addresses that are not in your overall device inventory.

```
$ rwsettool –difference firewall-internal.set inventory-list.set
> firewall-notfound.set

$ rwsettool –difference proxy-internal.set inventory-list.set >
proxy-notfound.set

$ rwsettool –difference intrusion-internal.set inventory-list.set
> intrusion-notfound.set

$ rwsettool –difference nmap-ping.set inventory-list.set > nmap-
notfound.set
```

Each "notfound" set contains addresses from your internal network ranges that appeared on your network, but were not found in your device inventory. And again there is a set "tool" designed to list them back as text:

```
$ rwsetcat firewall-notfound.set
$ rwsetcat proxy-notfound.set
$ rwsetcat intrusion-notfound.set
$ rwsetcat nmap-notfound.set
```

By keeping these separate, SOC analysts can divide and conquer, identifying when and what occurred. (Knowing the source, the original log can provide these details).