Measuring and Optimizing Network Security Operations: An Open Model

Findings from the Network Security Operations Quant Research Project

Version 1.6
Released: October 22, 2010

Securosis, L.L.C. 20930 North Tatum Blvd Suite #110-116 Phoenix, AZ 85050 **T** 602-412-3051 info@securosis.com www.securosis.com

Author's Note

The content in this report was developed independently of any sponsors. It is based on material originally posted on the Securosis blog <http://securosis.com/blog>, but has been enhanced, reviewed, and professionally edited.

Special thanks to Chris Pepper for editing and content support.

Licensed by SecureWorks

SecureWorks is a market leading provider of world-class information security services with over 2,900 clients worldwide spanning North America, Latin America, Europe, the Middle East and the Pacific Rim. Organizations of all sizes, including more than 15 percent of the Fortune 500, rely on SecureWorks to protect their assets, improve compliance and reduce costs. The combination of award-winning security technology, strong client service, and experienced security professionals makes SecureWorks the premier provider of information security services for any organization.

Contributors

The following individuals contributed significantly to this report through comments on the Securosis blog and follow-on review and conversations:

Jason Thacker
Tim McCreight
Ray Strubinger
Chris Walsh
Richard Bejtlich
Rob Scudiere
Bill Farrell
Spencer Ingram
Billy Smith
Tamer
Dan Finkle
David Collette
Mike Heiser
Jonathan Nitschke
Michael Vincent
ralexander
Gary Deckerd
Andy Wagoner
Charles Mattson

Copyright

This report is licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0.

http://creativecommons.org/licenses/by-nc-nd/3.0/us/

Executive Summary

Developing an Open Network Security Operations Metrics Model

This report includes the findings of the Network Security Operations (NSO) Quant research project. NSO Quant was initiated to kick-start development of a refined and unbiased metrics model for monitoring firewalls, IDS/IPS devices, and servers. Additionally, we have included managing firewalls and IDS/IPS devices in this research. Our goal is to provide organizations with a tool to better understand the costs of monitoring and managing their network security devices, and to guide improvements through an operational efficiency model capable of capturing accurate and precise performance metrics. NSO Quant was developed through independent research, community involvement, and an open industry survey.

The NSO Quant Survey

As part of the NSO Quant research, we conducted an open, industry-wide survey. 75 organizations responded, representing a broad range of organizations from small business (under 10 employees) to mega-enterprise (over 400,000). WIthout a statistically significant sample, we'll be careful not to draw firm conclusions,. Rather using the data to validate assumptions. The survey results and raw data are available for your own analysis at the Project Quant site.

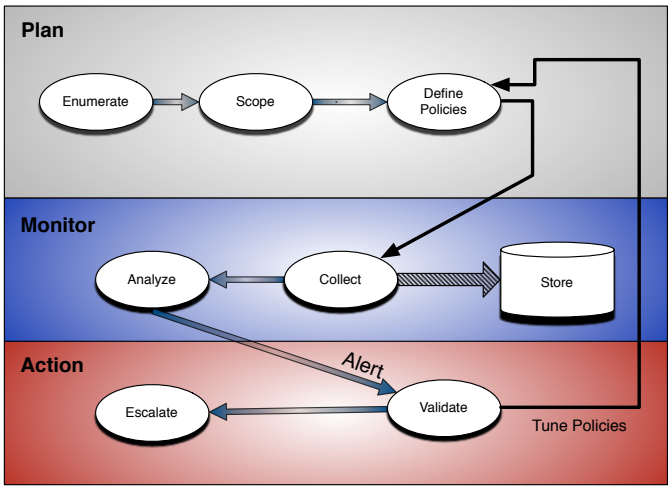
Key Findings

1. There is no public, industry-standard process framework for monitoring and managing network security devices. As a result, NSO Quant provides a superset framework to encompass activities required across three separate functions: Monitoring (firewall, IDS/IPS, & servers), Managing (firewalls & IDS/IPS), and Device Health — within any organization regardless of size or vertical. The Monitor process includes 8 phases with 32 discrete steps. The Manage process includes 11 phases with 46 steps. The Device Health process is self-contained with 4 separate steps.
2. A majority of organizations do monitor firewalls, IDS/IPS, and servers under their control. This is good news because we strongly believe that monitoring needs to be the centerpiece of every security program. About half of organizations view their NSO monitoring processes as mature, which mean about half run their security operations on an *ad hoc* basis. From a management perspective, it's not surprising that many organizations have well-defined workflows to manage the change control process on these devices. We attribute this to the maturity of firewall and IDS/IPS solutions.
3. Our results also indicate that security teams are heavily involved in setting up the monitoring infrastructure and analyzing the data, as well as managing the firewall and IDS/IPS devices. General IT ops plays a role, but primarily in the rote processes of collecting and storing the monitoring data.
4. Outsourcing either monitoring or management of network security devices is done by less than a quarter of the survey respondents, and most don't have any plans to look at an outsourced solution within their planning horizons. We did not ask follow-up questions in the survey to dig deeper, so can't draw conclusions as to why this is the case

without adding bias. Though we will point out this is inconsistent with other surveys we've seen and may be attributed to both the reality that the survey is not statistically reliable (with 75 responses) and the sophistication of the Securosis community (who tend to want to do everything themselves).

Monitor Process

The Monitor process is broken up into 8 separate phases across three categories:



1. **Plan:** Define the depth and breadth of monitoring activities. These are not one-time events, but a process to revisit every quarter and after any incident that triggers a policy review.

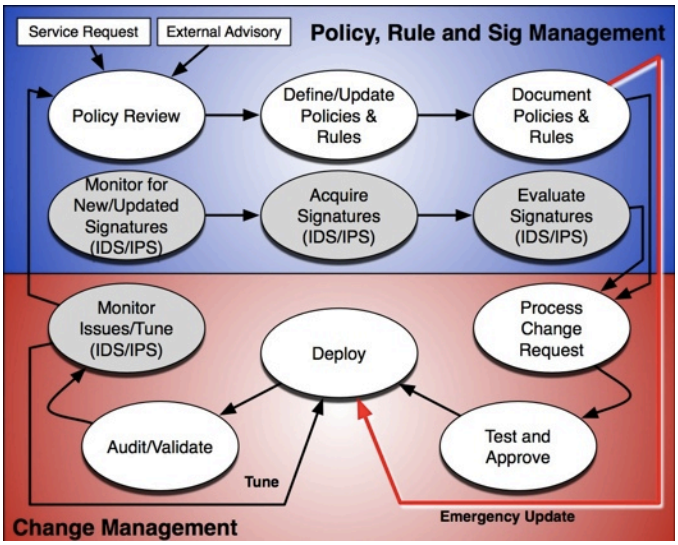
2. **Monitor:** Put the monitoring policies to use, gathering the data and analyzing it to identify areas for validation and investigation. All collected data is stored for compliance, trending, and reporting as well.

3. **Action:** If an alert fires in the analyze step, this phase kicks in to understand the issue and determine whether further action/escalation is necessary.

For more detail on the Monitor process, refer to the high level description and the detailed process explanations.

Manage Process

The Manage process consists of 11 process steps, divided into two main buckets:



1. **Policy, Rule, & Signature Management:** This involves managing the content that underlies the activity of the network security devices, including both attack signatures and the policies/rules that control attack response.

2. **Change Management:** Here additions, changes, updates, and deletions to the rules and signatures are implemented.

For more detail on the Manage process, refer to either the high level description or the detailed process explanations.

To be clear, there is overlap between the Monitor and Manage Processes we have defined. Manage tends to be a superset of monitoring and involves maintaining the blocking rules for firewalls and IPS devices. Along with our philosophy of Monitoring Everything, we advise organizations to streamline their Monitor processes first both because that activity is not dependent on other groups (like network ops) and because in most cases it has the greatest impact on security.

How to use NSO Quant

The value of any research is in how you use it to improve your operations and day to day activities. In this paper you will find a very detailed set of process steps, each of which may or may not be relevant to network security operations within your organization. Use what makes sense and forget the rest. In terms of the metrics, we had an excellent comment on one of blog posts which puts this initiative into the proper context.

> *Who is the intended audience for these metrics? [Metrics] are part of the job, but I'm not sure what the value is. To me the metrics that are critical around process [focus on whether] the number of changes align with the number of authorized requests. Do the configurations adhere to current policy requirements, etc...*
>
> *Just thinking about [my last] presentation to the CIO, I spent 3 hours getting consensus and 2 hours on prioritizing. [How do these metrics] get me much traction?*

One of the pillars of our philosophy on metrics is that there are really three sets of metrics that network security teams need to worry about. This comment is about the first type: the stuff you need to substantiate what you are doing for audit purposes. Those are key issues and things that you must be able to prove.

The second bucket is numbers that are important to senior management. These tend to focus around incidents and spending. Basically how many incidents happen, how that is trending, and how long it takes to deal with each one. On the spending side, senior folks want to know about percentage of expenditure relative to total IT spending, relative to total revenues, and how that compares to peers.

Then there is the third bucket, which are **the operational metrics that we use to improve and streamline our processes**. It's the old saw about how you can't manage what you don't measure — well, the metrics defined within NSO Quant represent pretty much everything we can measure. That doesn't mean you *should* measure everything, but the idea of this project is to decompose the processes as much as possible to provide a basis for useful measurement. Again, not all companies do all the process steps. Actually most companies don't do much from a process standpoint — besides fight fires all day.

Gathering this kind of data requires a significant amount of effort and will not be for everyone. But if you are trying to understand operationally how much time you spend on things, and then use that data to trend and improve your operations, you can get payback. Or if you want to use the metrics to determine whether it even makes sense for you to be performing these functions (as opposed to outsourcing), then you need to gather the data.

Clearly the CIO and other C-level folks aren't going to be overly interested in the amount of time it takes you to monitor sources for IDS/IPS signature updates. They care about outcomes, and most of the time you spend with these executives needs to be focused on getting buy-in and updating status on commitments you've already made. Which is the way it should be.

But if you don't measure and tune your internal processes, odds are you'll be less efficient — eating up budget and being forced to rely on FUD (fear, uncertainty, and doubt) to justify future spending. Which is most definitely how it *shouldn't* be. These metrics provide the fundamental tools for you to optimize your processes, even if you only use a fraction of them.

Table of Contents

Introduction 9

Background of the Project 9

Project Assumptions 10

High Level Monitor Process 11

High Level Manage Process 13

NSO Quant Metrics 16

Monitor Process 17

Monitoring Processes in the Real World 18

Enumerate 20

Scope 23

Define Monitoring Policies 25

Collect 29

Store 33

Analyze 35

Validate 38

Escalate 41

Manage Process 43

Manage Process Maturity 45

Policy Review 47

Define/Update Policies & Rules 49

Document Policies & Rules 53

Monitor for New/Updated Signatures (IDS/IPS) 55
Acquire Signatures (IDS/IPS) 56
Evaluate Signatures (IDS/IPS) 57
Process Change Request 59
Test & Approve 61
Deploy 63
Audit/Validate 65
Monitor Issues/Tune (IDS/IPS) 67
Organizational Responsibilities for Manage 69
Device Health Process 71
Conclusion 74
About the Analysts 76
About Securosis 77

Introduction

Background of the Project

The lack of credible and relevant network security metrics has been a thorn in the side of security practitioners for years. We don't know how to define success. We don't know how to communicate value. And ultimately, we don't even know what we should be tracking operationally to show improvement — or failure — in our network security activities.

Most security professionals seem happier just complaining about this, or flaming each other on mailing lists, than focusing on finding a solution. Some folks have tried to drive toward a set of metrics that makes sense, but most of the attempts are way too academic. Not to mention that most of our daily activities aren't even included in those models.

Not to pick on them too much, but I think these issues are highlighted in the way the Center for Internet Security has scoped out network security metrics. Basically, they didn't. They have metrics on Incident Management, Vulnerability Management, Patch Management, Configuration Change Management, Application Security, and Financial Metrics. So the guy monitoring all these devices and managing the network security doesn't count? Again, I know CIS is working toward a lot of other stuff, but the reality is the majority of security spending is targeted at the network and endpoint domains, and there are no good metrics for those. Moreover, what we spend on tools is literally dwarfed by the cost of our time. That, by far, is the biggest cost bucket for network security, and where we need to focus our efforts.

The formal objective and scope of this project are:

> *The objective of Network Security Operations Quant is to develop a cost model for monitoring and managing network security devices that accurately reflects the associated financial and resource costs.*

Our design goals for the project were to:

- Build the model in a manner that supports usage as an operational efficiency model to help organizations optimize their network security monitoring and management processes, and compare costs of different options.
- Produce an open model, using the Totally Transparent Research process.
- Advance the state of IT metrics, particularly operational security metrics.

As you read through this report, it's wise to keep the philosophy of Quant in mind: the high level process framework is intended to cover **all** the tasks involved. That doesn't mean you need to *do everything*, but does mean this is a fairly exhaustive list. Individual organizations then pick and choose those steps which are appropriate for them. As such, this model is really an exhaustive framework that can kickstart your efforts to optimize network security operational processes.

Project Assumptions

To achieve the NSO Quant project goals, we made certain assumptions:

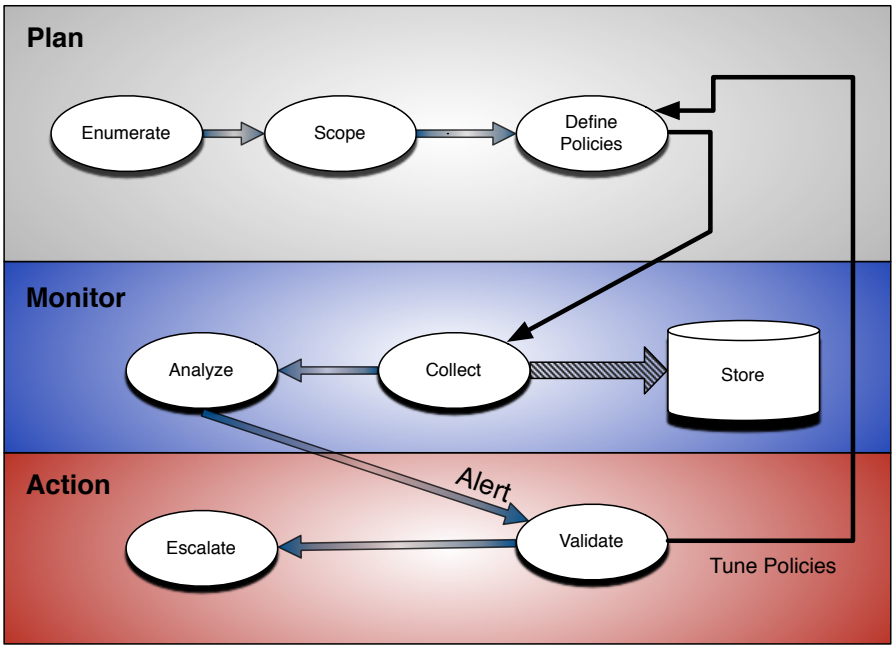
- *This should be a quantified metrics model, focused on costs:* All the metrics or variables in the model should be measurable with accuracy and precision. "Qualified" metrics, such as risk and threat ratings, are not included. This model is designed only to measure the costs of monitoring and managing network security devices, and to identify operational efficiencies or deficiencies in specific process areas. It relies on measurable, quantifiable inputs, rather than assessments or other unquantifiable values based on human judgement.
- *The model should apply to all relevant activities in scope:* The scope includes monitoring firewalls, IDS/IPS devices, and servers. We have also built models for managing firewalls and IDS/IPS devices. Obviously there are many other types of network security devices that could be included and most of the operational processes would be consistent. But we had to decide on a scope, and this seemed the best compromise.
- *The model should apply to organizations of any size or vertical:* This is not designed only for large organizations in particular vertical markets. Although smaller organizations work with fewer resources and different processes, the model still provides a functional framework.
- *The model thus represents a superset of monitoring and network security device management activities:* To achieve the dual goals of covering every activity in scope, and applying to organizations of differing sizes and verticals, the model was designed as a superset of any one organization's activities. *We do not expect users to utilize the entire model, and you are encouraged to adapt it for your own particular needs.* We understand collecting all of the data could actually cost more than monitoring and managing the devices. Over time we hope that more and more of these metrics will be available through automation and included in support tools or from your service provider(s).
- *The model should break out costs by process to support optimization:* One reason for the extensive detail on each process is to support identifications of specific operational efficiencies and problems. Our goal is to help organizations identify and correct problem areas, so this project defines all aspects of each process in gory detail to enable data collection, analyses on process efficiency, and trending.
- *The model cannot measure the costs of not monitoring or managing the devices:* Clearly, the easiest way to reduce your network security operational costs to zero is to do nothing. While there are many ways to 'measure' the business impact of not protecting your networks, they are not part of this model. In this project we are concerned only with measuring the costs when you *do* protect your networks. In large part this is due to our strict focus on quantified metrics: addressing the impact of *not* monitoring or managing network security devices would require us to include predictive and more subjective elements.

With that preamble to provide context, let's go over how we've broken up a very large set of operational processes. For this project we focus on three areas:

1. Monitor
2. Manage
3. Device Health (discussed here)

Let's consider each process at a high level before diving into the details.

High Level Monitor Process



Plan

In this phase, we define the depth and breadth of our monitoring activities. These are not one-time tasks, but processes to revisit every quarter, and after any incident that triggers a policy review.

1. **Enumerate:** Find all the security, network, and server devices which are relevant to determining the security of the environment.
2. **Scope:** Decide which devices are within the scope of monitoring activity. This involves identifying the asset owner; profiling the device to understand data, compliance, and policy requirements; and assessing the feasibility of collecting data from it.
3. **Develop Policies:** Determine the depth and breadth of the monitoring process, which really consists of both *Organizational* policies (which devices will be monitored and why), and *Device and Alerting* policies (which data will be collected from the devices, and the frequency of collection). This process is designed to be extensible beyond our scope of firewall, IDS/IPS, and server monitoring to include any other kind of network, security, computing, application, or data capture/forensics device.

Policies

For devices types in scope, device and alerting policies will be developed to identify potential incidents requiring investigation and validation. Defining these policies involves a Q/A process to test the effectiveness of alerts. A tuning process must also be built into the alerting policy definitions — over time the alert policies will need to evolve, as what you are defending changes, and the tactics of adversaries evolve.

Finally, monitoring is part of a larger security operations process, so policies are required for workflow and incident response. They define how monitoring information is leveraged by other operational teams and how potential incidents are identified, validated, and investigated.

Monitor

In this phase the monitoring policies are put to use, gathering the data and analyzing it to identify areas for validation and investigation. All collected data is stored for compliance, trending, and reporting as well.

1. **Collect:** Collect alerts and log records based on the policies defined under Plan. This can be performed within a single-element manager or abstracted into a broader Security Information and Event Management (SIEM) system for multiple devices and device types.
2. **Store:** For both compliance and forensics purposes, the collected data must be stored for future access.
3. **Analyze:** The collected data is then analyzed to identify potential incidents based on alerting policies defined in Phase 1. This may involve numerous techniques, including simple rule matching (availability, usage, attack traffic policy violations, time-based rules, etc.) and/or multi-factor correlation based on multiple device types.

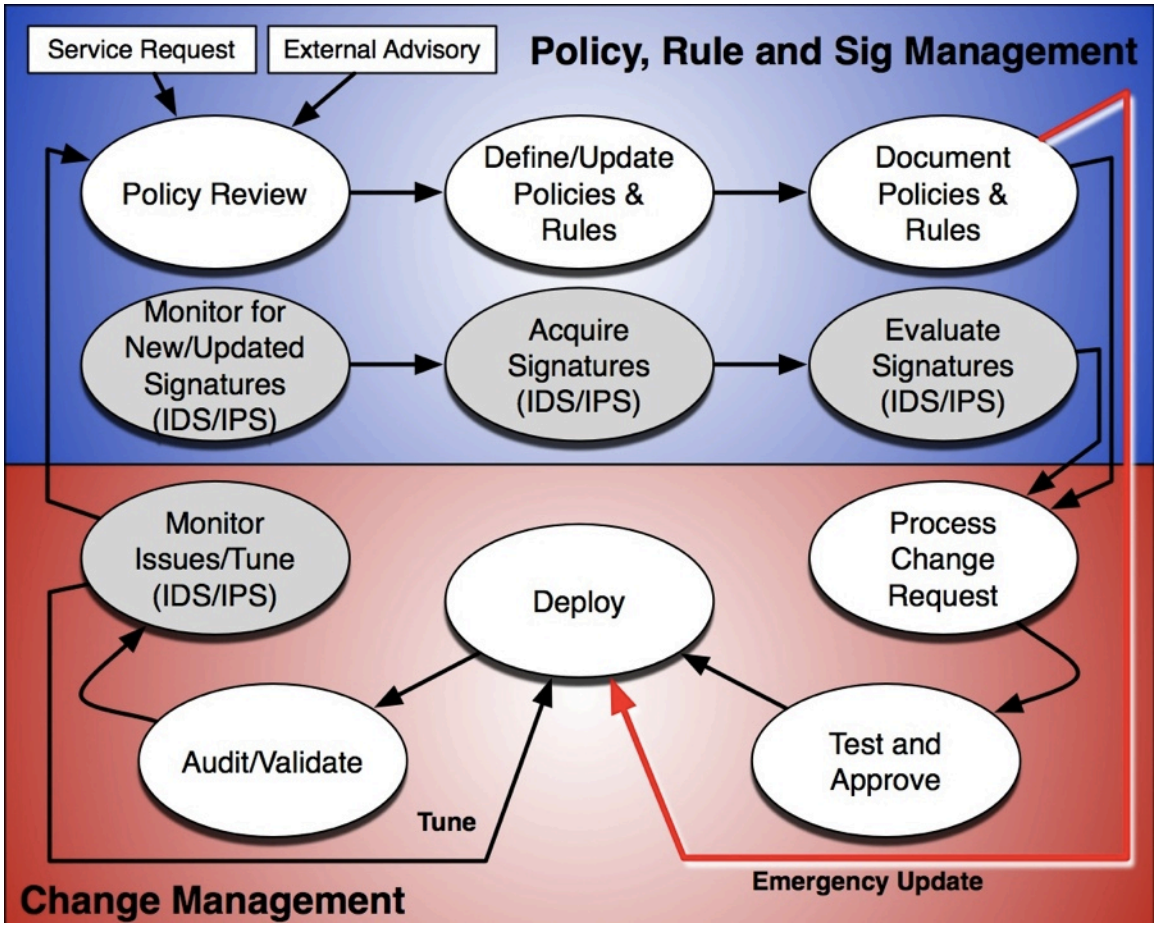
Action

When an alert fires in the analyze step, this phase kicks in to understand the issue and determine whether further action/escalation is necessary.

1. **Validate/Investigate:** If and when an alert is generated, it must be investigated to validate the attack. Is it a false positive? Is it an issue that requires further action? If the latter, move to the Action phase. Determine whether policies need to be tuned based on the accuracy of the alert.
2. **Action/Escalate:** Take action to remediate the issue. May involve hand-off/escalation to operations team.

After a few alert validations, it is time to determine whether any policies must be changed and/or tuned. This feedback loop must be recurring rather than a point-in-time activity, as the dynamic nature of networks and attacks requires ongoing diligence to ensure monitoring and alerting policies remain relevant and sufficient.

High Level Manage Process



Policy, Rule, and Signature Management

In this phase we manage the content that underlies the network security devices. This includes attack signatures and the policies & rules that control response to an attack.

1. **Policy Review:** Given the number of potential monitoring and blocking policies available on network devices, it is important to keep rules on the devices current. Keep in mind the severe performance hit (and false positive issues) of deploying too many policies on a device. It is a best practice to review network security device policies and prune rules that are obsolete, duplicative, overly exposed, prone to false positives, or otherwise unneeded. Policy review triggers include signature updates, service requests (new application support, etc.), external advisories (to block a certain attack vector or work around a missing patch, etc.), and policy updates resulting from the operational management of the device (change management process described below).

2. **Define/Update Policies & Rules:** This involves defining the depth and breadth of the network security device policies, including the actions (block, alert, log, etc.) taken by the device if an attack is detected — whether via rule violation, signature trigger, or other method. Note that as the capabilities of network security devices continue to expand, a variety of additional detection mechanisms will come into play. They will include increasing visibility into application traffic and identity stores. Time-limited policies may also be deployed to activate or deactivate short-term policies. Logging, alerting, and reporting policies are defined in this step. Here it is important to consider the hierarchy of policies that will be implemented on the devices. You'll have organizational policies at the highest level, applying to all devices, which may be supplemented or supplanted by business unit or geographic policies. Those highest level policies feed into the policies and/or rules implemented at a location, which then filter down to the rules and signatures implemented on a specific device. The hierarchy of policy inheritance can dramatically increase or decrease complexity of rules and behaviors. Initial policy deployment should include a Q/A process to ensure none of the rules impacts the ability of critical applications to communicate either internally or externally through the device.
3. **Document Policies and Rules:** As the planning stage is an ongoing process, documentation is important for operational and compliance reasons. This step lists and details the policies and rules in use on the device according to the associated operational standards, guidelines, and requirements.

The next three steps are specific to IDS/IPS devices, as they deal with managing signatures.

4. **Monitor for New/Updated Signatures (IDS/IPS):** Identify signature sources for the devices, and then monitor for new signatures on an ongoing basis. New attacks emerge constantly, so it's important to follow an ongoing process to keep IDS/IPS devices current.
5. **Acquire Signature (IDS/IPS):** Locate the signature, acquire it, and validate the integrity of the signature file(s). Most signatures are downloaded these days, so this is to ensure the download completed properly.
6. **Evaluate Signatures (IDS/IPS):** Perform the initial evaluation of each signature to determine whether it applies within your organization, what type of attack it detects, and whether it is relevant in your environment. This is the initial prioritization phase to determine the nature of each new/updated signature, its relevance and general priority for your organization, and any possible workarounds.

Change Management

In this phase the rules & signature additions, changes, updates, and deletions are handled.

1. **Process Change Request:** Based on either a signature or a policy change within the Content Management process, a change to the network security device(s) is requested. Authorization requires both ensuring the requestor is allowed to request the change, and the change's relative priority to select an appropriate change window. The change's priority is based on the nature of the signature/policy update and risk of the relevant attack. Then build out a deployment schedule based on priority, scheduled maintenance windows, and other factors. This usually involves the participation of multiple stakeholders — ranging from application, network, and system owners to business unit representatives if downtime or changes to application use models is anticipated.

2. **Test and Approve:** This step requires you to develop test criteria, perform any required testing, analyze the results, and approve the signature/rule change for release once it meets your requirements. Testing should include signature installation, operation, and performance impact on the device as a result of the change. Changes may be implemented in "log-only" mode to observe their impact before committing to blocking mode in production. With an understanding of the impact of the change(s), the request is either approved or denied. Obviously approvals may be required from a number of stakeholders. The approval workflow must be understood and agreed on to avoid significant operational issues.
3. **Deploy:** Prepare the target devices for deployment, deliver the change, and return them to normal operation. Verify that changes were properly deployed, including successful installation and operation. This might include use of vulnerability assessment tools or application test scripts to ensure no disruption of production systems.
4. **Audit/Validate:** Part of the full process of making the change is not only having the operations team confirm it during the Deploy step, but also having another entity (internal or external, but **not** part of the ops team) audit it as well to provide separation of duties. This involves validating the change to ensure the policies were properly updated, and matching it to a specific request. This closes the loop and makes sure there is a documentation trail for every change to the box.
5. **Monitor Issues/Tune (IDS/IPS):** The final step of the change management process (specifically for IDS/IPS devices) involves a *burn-in* period, where each rule change is scrutinized to detect unintended consequences such as unacceptable performance impact, false positives, security exposures, or undesirable application impact. To a degree this step applies to firewalls as well, but tuning is generally required for IDS/IPS. The testing process in the Test and Approve step is to minimize these issues, but there are variances between test environments and production networks, so we recommend a probationary period for each new or updated rule just in case. This is especially important when making numerous changes at the same time, as it requires diligent effort to isolate which rule(s) created any issues.

Emergency Update

In some cases, including data breach lockdowns and imminent or active zero-day attacks, a change to the network security device's signature/rule set must be made immediately. An 'express' process should be established and documented as an alternative to the full normal change process, ensuring proper authorized approval for emergency changes, as well as a rollback capability in case of unintended consequences.

Without further ado, we now jump into the detailed subprocesses within each step.

Process Overlap

As you look through the high level process descriptions there is overlap clearly. Both processes involve setting up policies and deploying the associated rules on the devices. Monitoring involves *taking no action* — only watching and alerting for things requiring investigation — and this can happen on all applicable network, security and computing devices. Managing adds the creation and maintenance of blocking rules for firewalls and IPS devices as well. The reality is that these processes are intertwined at multiple steps, and there isn't really a clean way around it. So as you are modeling your environment remember to factor in this overlap when building the cost models.

NSO Quant Metrics

For each Network Security Operations process we have determined a set of metrics to quantify the cost of performing the activity. We designed the metrics to be as intuitive as possible while still capturing the necessary level of detail. The model collects an inclusive set of potential network security operations metrics, and as with each specific process we strongly encourage you to use what makes sense for your own environment.

Because the model includes so many metrics, we have color coded the metrics to help you prioritize:

Key	The most important metrics in a given category. Using only key metrics will provide a rough but reasonably accurate overview of costs. These are the most useful metrics for determining costs and operational efficiency, and can be reasonably collected by most organizations.
Valuable	Metrics that are valuable but not critical for determining costs and efficiency. They provide greater accuracy than key metrics alone, but require more effort to collect.
Standard	Detailed metrics to help with deep quantification of a process, but these are either less important or more difficult to quantify. They may be more difficult to collect, or might involve complex interdependencies with other metrics.

Using key metrics alone will provide a reasonable picture of network security operations costs and a basis for improving operational efficiency and program effectiveness. Including valuable metrics, or valuable and standard metrics, will provide greater detail.

How to Use the Metrics

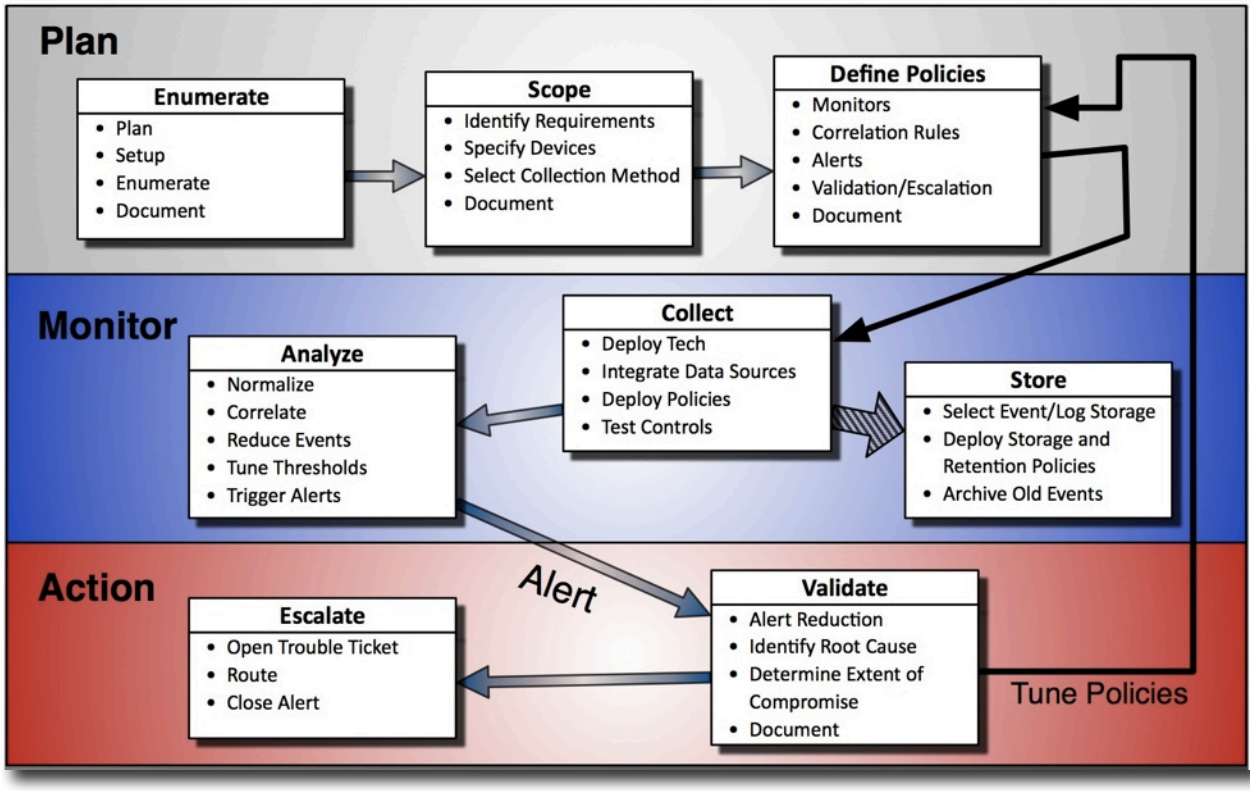
We recommend most organizations start at the process level. That involves matching each process in use within your organization against the processes described in this research, before delving into individual metrics. This serves two purposes:

- First, it helps document your existing process or lack thereof. Since all the metrics in the model correlate with steps in the NSO Quant processes, you'll need this to begin quantifying your costs.
- Second, you may find that this identifies clear deficiencies in your current process, even before evaluating any metrics. This provides an opportunity for a quick win early in the process to build momentum.

We include the applicable metrics for each specific process and subprocess, which can be built up to quantify your entire Network Security Operations program. Thus you make detailed measurements for all the individual processes and then combine them subtracting out overlapping efforts. Most of the metrics in this model are in terms of staff hours or ongoing full-time equivalents; others are hard costs (*e.g.,* licensing fees, test equipment, etc.).

It's important to keep the purpose of these metrics (and the entire Quant research program) in context. **The precision of the measurement is less important than the consistency and completeness of your efforts.** If you do have the ability to fully quantify costs for each step in the process you'll get a more accurate result, but this isn't realistic for most organizations. That said, with the right tools and automation you may be able to come extremely close for certain processes. Metrics will vary for any given process, so *think in terms of the average cost (or time) for any given step*. As long as the method of your estimation is consistent, you'll have metrics you can work with.

Monitor Process



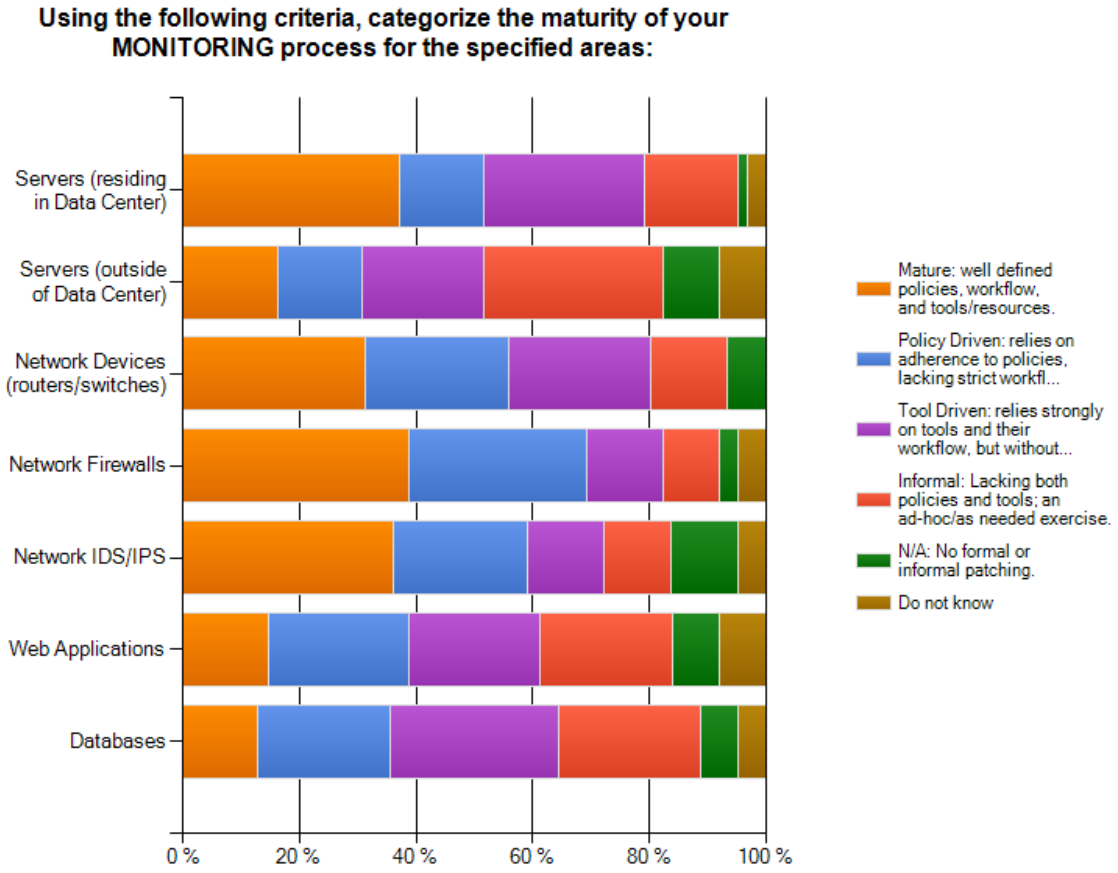
Let's dig into the Monitor process and break each high level step from in the Introduction down into its subprocesses and associated operational metrics. Here are links to each of the applicable process steps:

1. Enumerate
2. Scope
3. Define Policies
4. Collect
5. Store
6. Analyze
7. Validate
8. Escalate

Monitoring Processes in the Real World

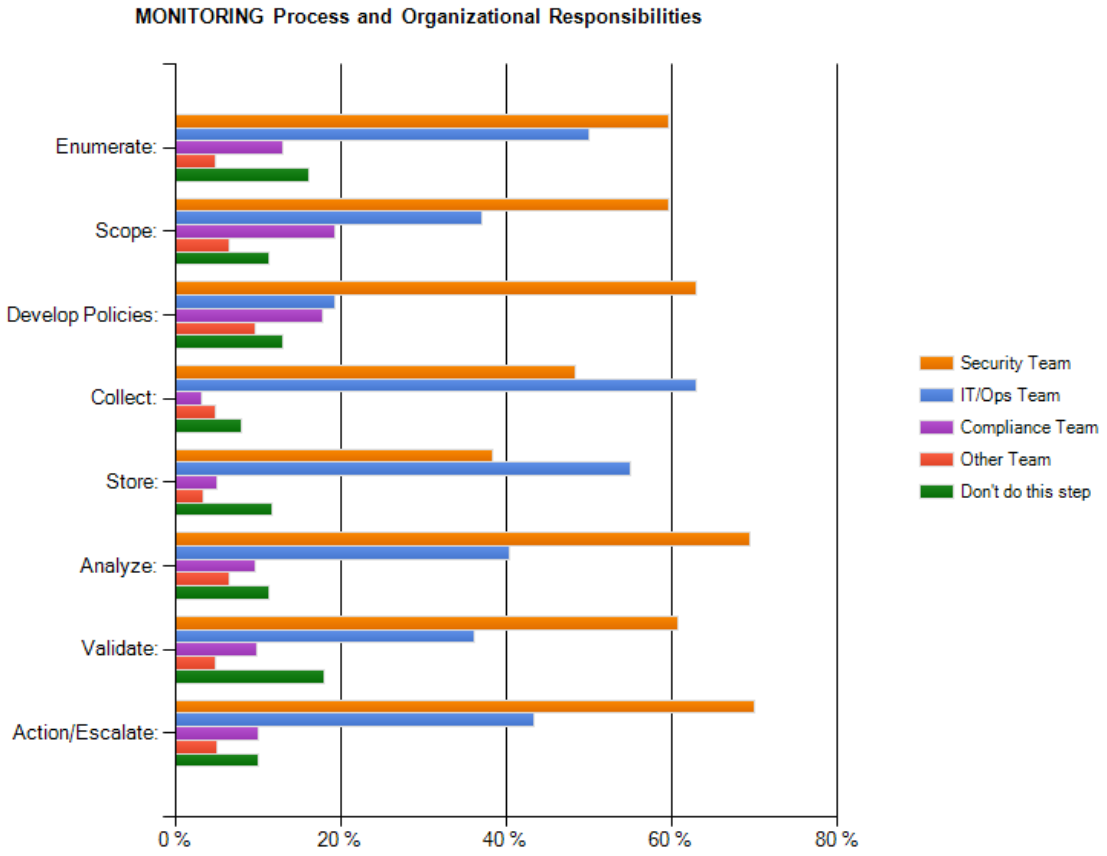
Along with our primary research, we surveyed 75 end users (in organizations ranging from 5 to 400,000+ employees) about what processes they have in place. We did this primarily to improve the research (and add some charts to this report), but also to get a firm grasp on what organizations do in practice, as opposed to pure process modeling theory.

In terms of the maturity of each function, it is clear that monitoring is a fairly mature process — at least to folks who do it every day.



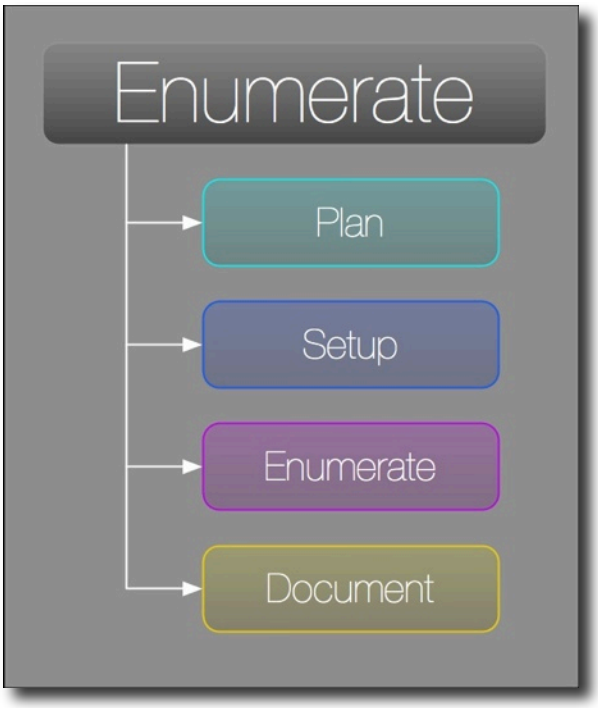
What does this tell us? That for devices/infrastructure within the control of IT, monitoring is pervasive and reasonably mature. As you'd expect, the areas not as pervasive are servers residing outside the data center (and therefore not fully under the control of IT ops), web applications, and databases, which tend to be controlled by other parties.

In terms of specific steps in the process map actually being done and by whom, the data confirms what you'd expect. The security team is heavily engaged in setting up the monitoring policies and analyzing the alerts, while ops folks tend to focus on the operational processes of collecting and storing the data.



Enumerate

There are four major subprocesses in this step:



1. **Plan:** We believe in planning the work and then working the plan, so you'll see a lot of planning subprocesses throughout our process maps. In this step you figure out how to enumerate the devices on the network, including what tools and techniques to use. You also need to identify the business units to search, mapping their specific network domains (assuming you have business units) and developing a schedule for the activities.
2. **Setup:** Next you need to set up your enumeration by acquiring and installing tools (if you go the automated route) or assembling your kit of scripts and open source methods. You also need to inform each group that you will be in their networks, looking at their stuff. In highly distributed environments, it may be problematic to do ping sweeps and the like without giving everybody a 'heads up' first. You also need to get credentials (where required) and configure the tools you'll be using.
3. **Enumerate:** Ah, yes, you actually need to do the work after planning and setting up. You may be running active scans or analyzing passive collection efforts. You also need to validate what your tools tell you, since we all know how precise that technology stuff can be. Once you have the data, you'll spend some time filtering and compiling the results to get a feel for what's really out there.
4. **Document:** Finally you need to prepare an artifact of your efforts, if only to use in the next step when you define the scope of your monitoring efforts. Whether you generate PDFs or kill some trees is not relevant — it's about simply making sure you've got a record of what exists at this point in time, as well as a reference point to compare against periodically.

Device Type Variances

Is there any difference between enumerating a firewall, an IDS/IPS, and a server — the devices we are focused on with this research project? Not for our purposes. You will use the same tools and techniques to identify what's out there, because fundamentally they are all IP devices. That said, there are significant differences in the types of data you get from a firewall, an IDS/IPS, and a server, or any other device. But for figuring out what's there, the general process is the same.

There will be some variation in what you do to validate what you've found. You may want to log into a server to verify it *is* actually a server. You may want to blast packets at a firewall or IDS/IPS. Depending on the size of your environment, you may need to statistically verify a subset of the found devices — it's not like you can log into 10,000 servers. Actually you could, but hopefully you can find a better way to spend your time. You are looking for the greatest degree of precision you can manage, but that must be balanced with common sense to figure out how much validation you can afford.

Also keep in mind that you will likely have to spend some time trying to figure out who owns specific devices. Especially the ones you didn't expect to find. This involves a lot of legwork, and don't underestimate the hassle you'll get when telling someone their pet project needs to be monitored by, or at least on the radar of, the central security group.

Large vs. Small Company Considerations

One of the downsides of trying to build generic process maps is trying to factor in every potential scenario and reflect them all in the process. But in the real world many of the steps in any process are built to support scaling for large enterprise environments. So for the monitoring subprocesses where there is a difference between monitoring 10 or 10,000 devices we'll comment on how things change.

Regarding enumeration, the differences crop up both when planning and during the actual enumeration process — specifically when verifying what you found. Planning for a large enterprise needs to be fairly detailed to cover a large IP address space (likely several different spaces) and there may be severe ramifications to disruptions caused by the scanning. Not that smaller companies don't care about disruption, but with fewer moving parts there is a smaller chance of unforeseen consequences.

Clearly the verification aspect of enumeration varies depending on how deep you go. There is a lot of information you can gather here; so it's a matter of balancing time to gather, against time to verify, and the need for the data.

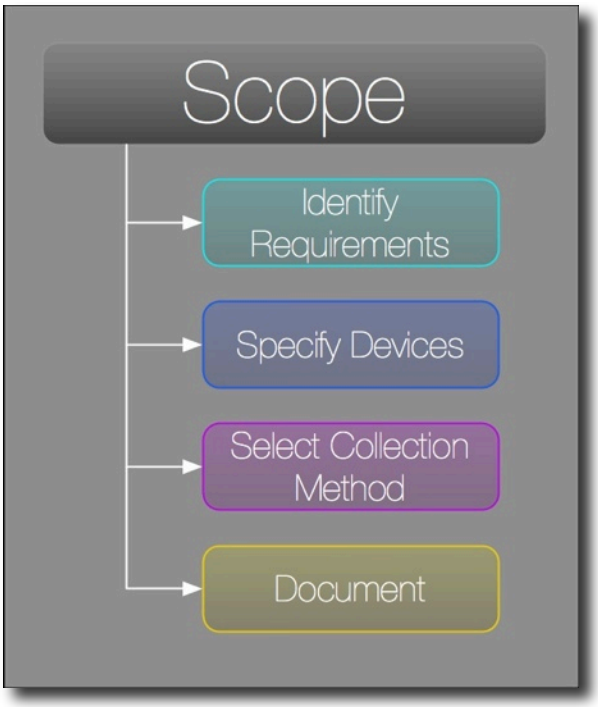
Enumerate Metrics

Process Step	Variable	Notes
Plan	Time to determine scope, device types, and technique (manual, automated, combined)	
	Time to identify tools (automated)	Only needs to happen once.
	Time to identify business units	
	Time to map network domains	
	Time to develop schedule	
Setup	Cost and time to acquire and install tools (assuming automation)	Tools are optional, but scaling is a problem for manual procedures.
	Time to contact business units	
	Time to configure tools (automated)	
	Time to obtain permissions and credentials	You need authorization before you start scanning networks.
Enumerate	Time to schedule/run active scan (automated)	Point in time enumeration. Repeat as necessary.
	Time to run passive/traffic scan (automated)	Identify new devices as they appear.
	Validate devices	
	Time to contact business units and determine ownership	Must identify rogue devices.
	Time to filter and compile results	
	Repeat as necessary	Enumeration must happen on an ongoing basis.
Document	Time to generate report	
	Time to capture baseline	

Scope

Once you have finished enumeration, it's time to figure out what you will actually monitor on an ongoing basis. This can be driven by compliance (all devices handling protected data must be monitored) or a critical application. Of course this tends not to be a decision you can make by yourself, so a big part of the scoping process ensures you get buy-in on what to monitor.

Here are the four steps in the Scope process:



1. **Identify Requirements:** Monitoring is not free, though your management may think so. So we need a process to figure out what and why you monitor. That means building a case for monitoring devices, perhaps leveraging things like compliance mandates and/or best practices. You also need to meet with the business users, risk/compliance team, legal counsel, and other influencers to understand what needs to be monitored from their perspectives and why.
2. **Specify Devices:** Based on those requirements, weigh each possible device type against the requirements and then figure out which devices of each type should be monitored. You may look to geography, business units, or other means to segment your installed base into device groups. In a perfect world you'd monitor everything, but the world isn't perfect. So it's important to keep economic reality in mind when deciding how deeply to monitor.
3. **Select Collection Method**: For each device you'll need to figure out how you will collect the data, and which data you want. This may involve research if you haven't fully figured it out yet.
4. **Document:** Finally you document the devices in scope, and then undertake the fun job of achieving consensus. Yes, you already asked folks what should be monitored when identifying requirements, but we remain fans of *both* asking ahead of time and then reminding them what you heard, *and also* confirming they still agree when it comes time to start actually *doing*. Consensus building takes time — which is why most folks skip it — but minimizes the chance that you'll be surprised down the road. Remember, security folks hate surprises.

Large vs. Small Company Considerations

These are generally the same as in the Enumerate process. The bigger the company, the more moving pieces, the harder requirements gathering is, and the more difficulty in getting a consensus. Got it? Okay, that was a bit tongue-in-cheek, but as a security professional trying to get things done, you'll need to figure out the level of research and consensus to undertake with each of these steps. Some folks would rather *ask for forgiveness* than permission, but as you can imagine there are risks to that.

The good news is there is plenty of leverage in figuring out how to collect data from the various device types. Doing the research on collecting data from Windows or Linux servers is the same whether you have 15 or 1,500. That goes for firewalls and IDS/IPS devices as well. You'll spend the extra time gaining consensus, right?

Scope Metrics

Process Step	Variable	Notes
Identify	Time to build case for monitoring devices	
	Time to monitor regulations mandating monitoring	One of the easiest ways to justify monitoring is a compliance mandate.
	Review best practices	
	Time to check with business units, risk team, and other influencers	Factor business requirements into the analysis.
Specify Devices	Time to determine which device types need to be monitored	Start with enumerated device list as starting point. Then look at geographic regions, business units, and other devices to define final list.
Select Collection Method	Time to research collection methods and record formats	For in-scope devices
	Time to specify collection method	For each device type
Document	Time to document in-scope devices	This is a key step due to its role in compliance. Although still important in organizations without compliance mandates, it can be reduced to Valuable in such cases.
	Time to gain consensus on in-scope list	Consensus now avoids disagreement later.

Define Monitoring Policies

So many attacks, so little time. We feel your pain and know you are under the gun to figure out what's happening, understand what's under attack, and fix it.

As you plan your monitoring process, you'll be making a ton of decisions once you figure out what you have and what's in scope. These decisions are the policies that will govern your monitoring, set the thresholds, determine how data is analyzed, and trigger the alerts.

We described the Define Policies step as follows:

> *Define the depth and breadth of the monitoring process, what data will be collected from the devices, and frequency of collection.*

The scope of this project is specifically firewalls, IDS/IPS, and servers; so our examples will be tailored to those device types; but there is nothing in this process that precludes monitoring other network, security, computing, application and/or data capture devices.

Just to clarify terminology here, we refer to a **policy** as a high-level business-oriented description. For example, you may need to monitor in-scope devices to comply with PCI -- that would be a policy. These are high-level and distinct from the actual implementation rules which would be installed on the monitoring devices to implement the policies.

Rules implement the policy within the device. So you'd need to define the types of correlation analysis, alerting, and reporting you'll implement in the monitoring system.

Monitors

Our first set of policies will be around monitoring, specifically which activities on which devices will be monitored. A first set of high level *Organizational Policies* determines what to monitor at a strategic level. Then we'll dig deeper into *Device Policies* to determine the frequency of data collection and data retention. We also need to give some thought to the risk to the organization based on each policy. For example, your firewall will be detecting all sorts of attacks, so those events have one priority. But if you find a new super-user account on the database server, that's a different level of risk.

The more you think through what you are monitoring, what conclusions you can draw from the data, and ultimately what the downside is when an alert triggers from that device, the more smoothly the rest of the monitoring process will go.

Correlation Rules

The issue with correlation is that you need to know what you are looking for in order to set rules to find it. That also requires you to understand the interactions between the different data types you are analyzing (firewall logs vs. identity stores vs. database transaction records). How do you know that? Basically you need to do a threat modeling exercise based on the kinds of attack vectors you want to find. Sure, we'd love to tell you that your whiz-bang monitoring thing will find all the attacks out of the box and you can get back to playing Angry Birds. But it doesn't work that way.

A big part of this step is spending time on the white board mapping out the different kinds of attacks, suspect behavior, and exploits you've seen and expect to see. Then map out how you'd detect that attack, including specifics about the data types and sequences of events that must happen for the attack to be successful. Out of this model comes the correlation policy. Put yourself in the shoes of the hacker and think like them. Easy, huh? Not so much, but at least it's fun. This is more art than science.

Depending on the device, there may be some resources — perhaps an open source tool with a set of default rules, think Snort or OSSEC or OSSIM — that can get you started. Again, don't just expect to download some stuff and get moving. You need to actively think about what you are trying to detect and build your policies accordingly. In order to maintain sanity, understand that defining (and refining and tuning and refining and tuning some more) correlation policies is an ongoing process.

Finally, it's important to recognize that you cannot possibly build a threat model for every kind of attack that might be launched at your organization. So you are going to miss some stuff, and that's part of the game. The objective of monitoring is to react faster, not to respond perfectly to every attack. Threat modeling focuses you on the most significant risks to your organization.

Alerting Rules

In this step you define the scenarios that trigger alerts and configure those rules on the device. You need to define what makes an alert notify-only, as well as a variety of alert severities depending on the attack and the risk it represents. The best way we know to do this is to go back to your threat models. For each attack vector, there are likely a set of situations that are low priority and a set that are higher. What are those priorities and why? Go back and question all of your assumptions, because firing a low-priority alert on a high-priority attack makes everyone look bad because you usually don't get to deal with low priority alerts. Then determine which thresholds on the alert determine the difference between priority and low priority. These decisions drive your alerting rules and substantiate your monitoring approach, so you want to make sure to clearly document how you are doing things and why.

Next you need to define your notification strategy, by alert severity. Will you send a text message or an email or does a big red light flash in the SOC calling all hands on deck? We spent a lot of time during the scoping process getting consensus, but it's not over yet. You also need to make sure you have everyone on the same page regarding how you will notify them when an alert fires. You don't want an admin to miss something important because they expected a text and don't live in their email. Also make sure the ops team (or whoever has to respond to the alert) understands what their responsibilities are and what they should do, if anything.

Validation/Escalation Policies

As we work through policy definition, we have built threat models and the associated correlation and alerting rules to implement on the devices. Next we need to think about how to prove whether the attack is legitimate or whether it's just

a false positive. Here you map out what you think an analyst can/should do to prove an attack. Perhaps it's looking at the log files or logging into the device and/or confirming with other devices. The specifics vary depending on the threat. At the end of validation, you should be able to definitively answer the question: Is this attack real? So your policies should define what 'real' means in this context, and set expectations for the level of substantiation you expect to validate the attack.

You also need to think about escalation, depending on the nature of the alert. Does this kind of alert go to network ops or security? Is it a legal counsel thing? What is the expected response time? How much data about the alert do you need to send along? Does the escalation happen via a trouble ticketing system? Who is responsible for closing the loop? All these issues need to be worked through and **agreed to** by the powers that be. Yes, you need consensus (or at least approval) for the escalation policies because they involve sending information to other groups and expecting a particular response.

Also make sure to communicate your expectations about escalation and response time to the operations teams throughout the Policy Definition process. Most ops teams will nod their heads in agreement until it comes time for them to take responsibility for the process. The more you communicate up front, the less wiggle room they'll have later on to 'renegotiate' the deal.

Document

Finally, you have worked through all the issues. The last step is to document all the policies and communicate responsibilities and expectations to the operations teams (and anyone else in the escalation tree). These policies are living documents, and change frequently as new attacks are found in the wild and new devices & applications appear on your network.

Other Considerations

- **Baselines:** You also need to think about whether you will be building a baseline from the data you collect. This involves monitoring specific devices for a period of time, assuming what you find is normal, and then looking for behavior that is *not normal*. This is a valid approach to streamline the process of getting your monitoring system on line, but understand it bakes in assumptions about what is good and bad, and those assumptions may be wrong. We prefer *both* performing the threat modeling exercise and establishing a baseline. We'd say together they provide the best of both worlds, but that would be corny.
- **Large vs. Small Company Considerations:** The biggest differences we tend to see in how big companies monitor compared to small companies are the different amounts of data and numbers of policies used to drive correlation and alerting. Obviously big companies have different economics to invest in tools and services to get them operational and keep them operational. Small companies need to compromise, which means you won't be able to monitor everything, so don't try. Again, that's where threat models come into play. By focusing on the highest risk models, you can go through the process and do a good enough job with a handful of models. Large companies, on the other hand, need to be aware of analysis/paralysis because there is literally an infinite number of threat models, correlation policies, and alerts that can be defined. At some point the planning has to stop and the *doing* has to start.

Define Monitoring Policies Metrics

Process Step	Variable	Notes
Define Monitors	Time to identify which activities will be monitored, on which devices	
	Time to define data collection frequency and retention rules	
	Time to define threat levels to dictate different responses	
Build Correlation Rules	Time to define suspect behavior and build threat models	You need to know what you are looking for.
	Time to define correlation policies to identify attacks	
	Time to download available rule sets to kickstart effort	Vendors generally offer out-of-the-box correlation rules to get you started.
	Time to customize rule sets to cover threat models	
Define Alerts	Time to define specific alert types and notifications for different threat levels	Based on the defined responses, you may want different notification options.
	Time to identify criticality of each threat and select thresholds for specific responses	
Define Validation/ Escalation Policies	Time to define validation requirements for each alert	What type of confirmation is required before firing an alert?
	Time to establish escalation procedures for each validated alert	
	Time to gain consensus for policies	These policies will drive action, so it's important to have buy-in from interested parties.
Document	Time to document policies	
	Time to communicate responsibilities to operations teams	It takes time to manage expectations.

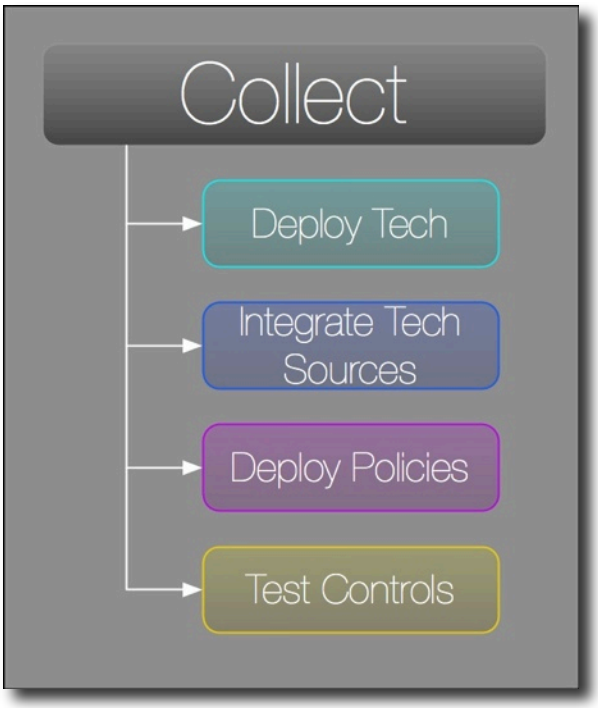
Collect

Now it's time to get busy and start actually monitoring these devices. First you build the infrastructure and systems to actually collect the data. Then you determine where to store it. Operationally these two subprocesses tend to be integrated, as most organizations select tools with built-in storage, but part of our philosophy is not to assume that anyone will buy (or deploy) a tool for any of these processes, so we'll spell everything out in sufficient detail to model a manual activity.

We described the Collect step as follows:

> *Collect alerts and log records based on the policies defined in Phase 1. Can be performed within a single-device-type manager or abstracted into a broader Security Information and Event Management (SIEM) system for multiple devices and types.*

The subprocesses are:



1. **Deploy Tool:** It's hard to collect data without *any* technology, so the first step is about selecting and procuring mechanisms for data collection. This doesn't necessarily mean buying a commercial tool — there are other options. But it does mean taking the time to research and select *some* technology, and to install and configure your choice.
2. **Integrate with Technical Sources:** After the technology is deployed we get to integrate with the data sources identified during the Enumeration and Scoping stage. This involves listing all the data sources, collecting any required permissions for those devices, and configuring both the receiving technology (collectors), and the devices being monitored. Finally you need to test your rig to make sure you are pulling the right data from the right devices.
3. **Deploy Policies:** After we have data in the system it's time to add correlation and alerting rules, and automate the validation/escalation process as much as possible. You'll also need to test the policies to make sure they work as desired.
4. **Test Controls:** The last step of the collection subprocess is effectively a system test of the whole shebang. You have tested the individual components (data sources, correlation, and alerts), so now you need to make sure everything works together. In order to do this right, you'll spend time designing a test scenario and building a testbed — including developing a test data set. Then simulate some attacks and analyze the test data at a high level to be sure the system provides the data you need.

As you can tell, testing is a key part of the Collect step — if any of the subsystems fail the system isn't worth much. One way to bootstrap all this testing is through the proof of concept process during tool selection (assuming you select a

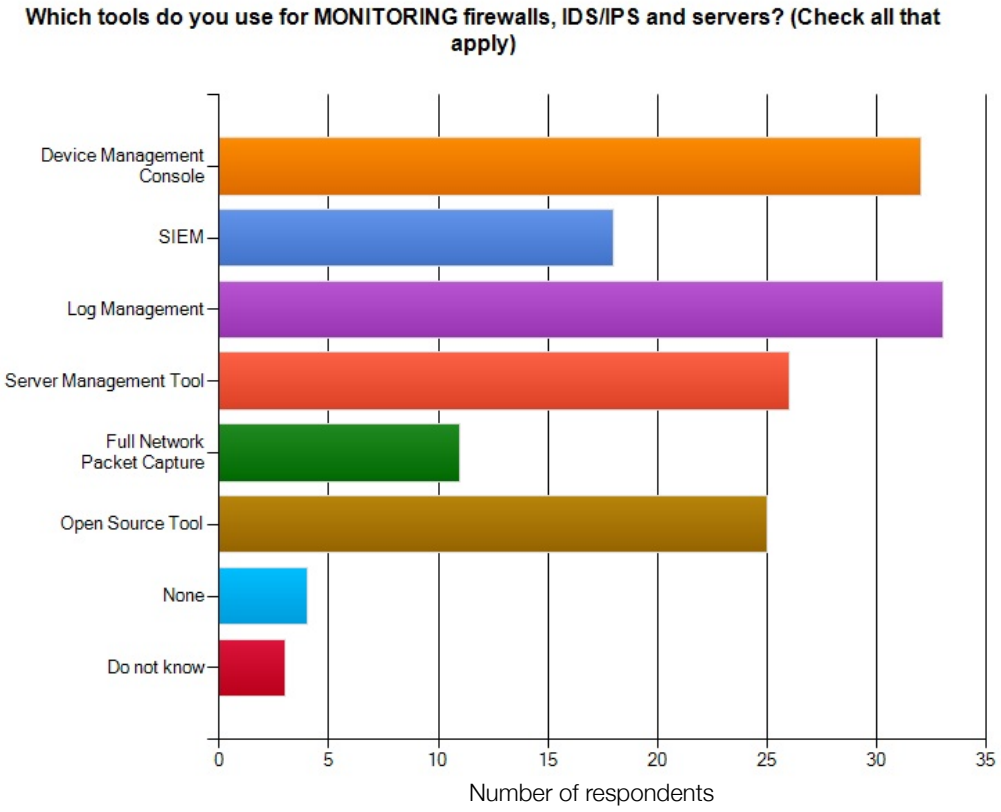
tool). This way the vendor works with your team to set up the system, collect some data, and run through the use case(s) driving the procurement. Obviously as you move into production you'll be adding a lot more data sources, correlation rules, and alerts — and extensively tuning the environment. But the proof of concept can provide a very useful running start.

To Tool or Not to Tool

It's very easy to assume you will be deploying a specific tool to do the monitoring. Most folks look at a log collection device or perhaps even a SIEM platform to monitor devices, and there are several managed service options to consider for those not wanting to build and maintain the monitoring infrastructure. So a big part of this process is figuring out whether you need a tool, and if so whether it makes more sense to deploy a product or outsource it. If you decide to deploy (and manage) yourself, then you must decide whether you need a Cadillac of monitoring tools, or if an open source 'Kia' will suffice.

There is no right or wrong answer — it gets back to how much time you have to deploy and maintain the system. Monitoring devices using the device management console is certainly possible, but this means another console for each device type, and you'll give up the benefits of aggregation and correlation across data types. Depending on your use case, that may work. But in practice we see many organizations moving to centralize monitoring of all the in-scope devices in an attempt to increase efficiency and streamline the compliance process.

In terms of our survey respondents, the following chart shows that most organizations rely on tools to perform their monitoring processes. Organizations could choose multiple options, so it wasn't an either/or proposition.



Device Type Variances

There is quite a bit of difference in data collection between firewalls & IDS/IPS, compared to servers. Most security devices support a `syslog` feed, so you can just point them at the collection point and be on your way. Some devices (such as Check Point firewalls) have their own proprietary collection protocols, but pretty much all the commercial and open source products support the leading security device vendors.

Servers are a very different animal, and most of the time you'll need to go get the log/event files. Depending on the type of server, you may be using either an OS-specific protocol like WMI or an open protocol like `ssh`. There are also open source collection tools for servers like Snare, and more generic logging platforms like Splunk — all of which can and do automate the process of collecting data from servers.

Your policies will dictate your options for all these collection processes, because collecting very detailed information from many different devices and performing deep analysis on the data are cost prohibitive without a tool.

Large vs. Small Company Considerations

The real (and obvious) difference in collection between a large company and a smaller entity is scale. Large companies not only have more people and more funding for these monitoring projects (yes, that is a crass overgeneralization), but they also have a *lot* more devices and device types, and generally want to do *much* more analysis and automation. They tend to be willing to pay for this scale and detail as well, but in a tight economy that's not always the case.

The more modest collection requirements of smaller companies make things much easier. But keeping everything up to date and current can be problematic, given the number of other things for the same personnel to handle.

Buy vs. Build

We see a decent amount of interest in outsourcing monitoring functions — especially for firewalls and IDS/IPS — because of the perception that service providers can bring a great deal of leverage to the table, driving economies of scale and delivering services at lower prices. We don't know if that's the case for your specific environment, nor will you until you do the work required to understand what it **really** costs you to monitor your devices. We suggest you do that level of analysis prior to deciding whether monitoring yourself or outsourcing is the right choice. One of the main deliverables for this research project is a model to help understand the costs of monitoring services to your environment. We aren't religious about whether outsourcing or doing it yourself is the right choice, but we are religious about being able to make informed and objective decisions.

Collect Metrics

Process Step	Variable	Notes
Deploy Tool	Time to select and procure technology	Yes, we are assuming that you need technology to monitor your environment.
	Time to install and configure technology	
Integrate with Data Sources	Time to list data sources	Based on enumeration and scoping steps
	Time to collect necessary permissions	Some devices (especially servers) require permissions to get detailed information
	Time to configure collection from systems	Collection typically via push/`syslog` or pull
	Time to Q/A data collection	
Deploy Policies	Time to deploy policies	Use the policies defined in the Planning phase.
	Time to Q/A policies	
Test Controls	Time to design testing scenario	
	Time to set up test bed for system test	
	Time to run test	You need to simulate an attack to figure out whether rules are implemented correctly.
	Time to analyze test data	Ensure collection and policies work as intended.

Store

Storing means keeping all this good stuff you are collecting. Actually it's a bit more than that, because deciding *where* to store data and the associated architecture are critical to scaling your monitoring system. You also need to make some tough decisions about archiving, because the larger your data set the more compute-intensive the analysis, which slows everything down. We all know speed is our friend when trying to monitor stuff, so it's about balancing sufficient historical data for detailed analysis & correlation against keeping the system manageable and responsive.

The key to dealing with storage is leverage, so you'll use the same storage for data from all your devices — not just firewalls, IDS/IPS, or servers — assuming you do multi-data-type analysis, at least.

Here are the subprocesses for Store:

1. **Select Event/Log Storage:** Now that you have all this data, what are you going to do with it? That's a key question — even if you select a tool to monitor, you'll need to do something with the data at some point. So you'll be researching and selecting storage options, and also spending some time in your data center to figure out what options you have for leveraging your organization's existing storage infrastructure. Once you decide on the storage strategy it's time to implement and deploy the collection targets.
2. **Deploy Storage and Retention Policies:** Back in the Define Policies subprocess you spent time figuring out how long to keep the data and building consensus for those decisions. Now you need to deploy the policies on the storage device(s), which usually involves a configuration and testing step. It's not a good idea to lose data, especially if there are forensics requirements, so testing is key to this effort.
3. **Archive Old Events:** Finally you'll need to move older events out of the main storage system at some point. So you'll be configuring the archival targets — likely leveraging existing archiving capabilities, as well as configuring the event/log storage environment to send old events to the archival system. Yes, once again we have to mention testing, especially to ensure archived data is accessible with sufficient data integrity.

Leverage and Storage

One of the key decisions for those building their own collection systems is whether you can leverage your organization's existing storage environment. There is no easy answer — especially once you start thinking about the sensitivity of the data, the common compliance requirement for separation of duties, and access to private data on a need-to-know basis. Obviously using existing spindles is the most cost effective way to do things, but if you buy and deploy your own SIEM/Log Management platform for monitoring it will include its own storage, so the question is more applicable to archival. Obviously if you outsource monitoring, storage is a non-issue, though some organizations keep a local copy of the data just in case. Those costs are balanced against piece of mind.

Is Forensics Data Your Friend?

The other consideration in building your storage environment is the need for forensically clean data. If you ever want to move toward prosecuting malfeasance, you'll need to meet certain requirements when collecting and storing data. Our pals at NIST have defined some guidelines for log management (PDF), but in a nutshell you'll need to be able to prove chain of custody and integrity of data — basically to demonstrate that the records weren't tampered with.

Store Metrics

Process Step	Variable	Notes
Select Event/Log Storage	Time to research and select event/log storage	
	Time to implement/deploy storage environment	May involve working with the data center ops team if shared storage will be utilized.
Deploy Storage and Retention Policies	Time to configure collection policies for storage and retention on devices	Based on policies defined during planning phase.
	Time to deploy policies	
	Time to test storage and policies	
Archive Old Events	Time to configure data archival policies	
	Time to deploy archival policies	
	Time to archive data for availability and accuracy	You don't want to figure out data was lost or compromised during or after an incident.

Analyze

Everyone in security knows data isn't the problem. We have all sorts of data — tons of it. The last two steps in the Monitor process focused on gathering data and putting it where we can get it. What's in short supply is *information*. Billions of event/log records don't mean much if you can't pinpoint what's really happening at any given time and send actionable alerts.

So analyzing the data is the next subprocess. Every alert that fires requires a good deal of work — putting all that disparate data into a common format, correlating it, reducing it, checking against appropriate thresholds, and then maybe, just *maybe*, you'll be able to spot a real attack before it's too late. Let's decompose each of these steps to understand whether to perform them in your environment, and if so how.

We described the Analyze step as follows:

> *The collected data is then analyzed to identify potential incidents based on alerting policies defined in Phase 1. This may involve numerous techniques, including simple rule matching (availability, usage, attack traffic policy violations, time-based rules, etc.) and/or multi-factor correlation across multiple device types (SIEM).*

Here are the subprocesses for Analyze:

1. **Normalize Events/Data:** The Collect step didn't mention the need to put data into a common format. Most analyses require some level of event data normalization. As vendor tools become more sophisticated, and can manage more analysis on unstructured data, the need for normalization is reduced, but we always expect some level of normalization to be required — if only so we can compare apples to apples, with data types that are largely apples and oranges.
2. **Correlate:** Once we have the data in a common format we look for patterns which may indicate some kind of attack. Of course we need to know what we are looking for to define the rules that we hope will identify attacks. We spoke extensively about setting up these policies/rules in Define Policies. A considerable amount of correlation can be automated, but not all of it, so human analysts aren't going away.

3. **Reduce Events:** Our systems are so interrelated now that any attack touches multiple devices, resulting in many similar events being received by the central event repository. This gets unwieldy fast, so a key function of the analysis is to eliminate duplicate events. Basically you need to increase the signal-to-noise ratio and filter out irrelevant events. Note that we don't mean *delete* — merely move them out of the main analysis engine to keep things streamlined. For forensics we need to retain the full log record.
4. **Tune Thresholds:** If we see 2 failed logins, that might be an attack, or not. If we spot 100 in 10 minutes, something funky is going on. Each rule needs thresholds defining when to alert. Rules tend to look like "If [something] happens X times within Y minutes, fire off an alert." But defining X and Y is hard work. You start with a pretty loose threshold that generates many alerts, and quickly tune and tighten those thresholds to keep the number of alerts manageable. When building compound policies, such as "If [this] happens X times within Y minutes AND [that] happens Z times within W minutes...", it's even more fun. Keep in mind that thresholds are more art than science, and plenty of testing and observation are required over a significant period of time to determine the correct levels.
5. **Trigger Alerts:** Finally, after you've normalized, correlated, reduced, and blown past the threshold, you need to alert someone to something. In this step you send the alert, based on the rules defined earlier. The alert needs to go to the right person or team, with sufficient information to enable validation and support their response. Depending on your internal workflow, the alert might be sent via the monitoring tool, a help desk system, paper, or smoke signals. Unfortunately smoke signals are out of style nowadays, but you get the point.

Other Considerations

- **The Most Powerful Correlation Engine:** As you can imagine, there is a lot of number crunching involved in this kind of analysis. That usually requires a lot of computing power cranking away on zillions of records in huge data stores — at least if you deployed a tool to facilitate monitoring. We've met more than a few security professionals who use the world's most powerful correlation engine much more effectively than any programmed tool. Yes, we're talking about the human brain. These are unique folks, but there are people who can monitor event streams flying past them and 'instinctively' know when something is *not normal*. Is this something you can count on? Not entirely, but we don't think you can solve this problem either purely in software *or* entirely without it. As usual, somewhere in the middle is best for most organizations. We have seen many situations where risk priorities are set via SIEM, and an analyst can then very quickly validate the alert and determine whether the issue requires further investigation & escalation.
- **Prioritizing the Alert:** Alerts that are ignored don't really help anyone. So a key criteria for success is making sure the ops team knows the urgency of the alert, based on asset/data value and importance to the organization. This is clearly somewhat subjective, but with priorities defined earlier in the process everyone should be on board with the urgency of specific alerts.
- **Feedback Loops:** The attack space is very dynamic, which means the correlation rules and thresholds for alerts that you use today will need to be adapted tomorrow. This doesn't happen by itself, so your process must systematically factor in feedback from analysts about which rules are working and which aren't. The correlation rules and alerting thresholds get updated accordingly, and hopefully over time the system increases in effectiveness and value.

- **Device Type Variances:** As discussed under Define Policies, each device type generates different data and thus requires customized rules, event reduction, and thresholds. But the biggest challenge in monitoring the different device types is figuring out the dependencies of rules that incorporate data from more than one type. Many vendors ship their tools with a set of default policies mapping to specific attack vectors and that's a good start, but tuning those policies for your environment takes time. So when modeling these process steps to understand the cost of delivering this service internally, we need to factor that demanding tuning process into the mix.

Monitoring systems are designed to improve security and increase efficiency, but it's important to recognize the significant time investment required to configure a system sufficiently to actually provide value. And that investment is ongoing, because the system must be constantly tuned to ensure relevance and coverage over time.

Analyze Metrics

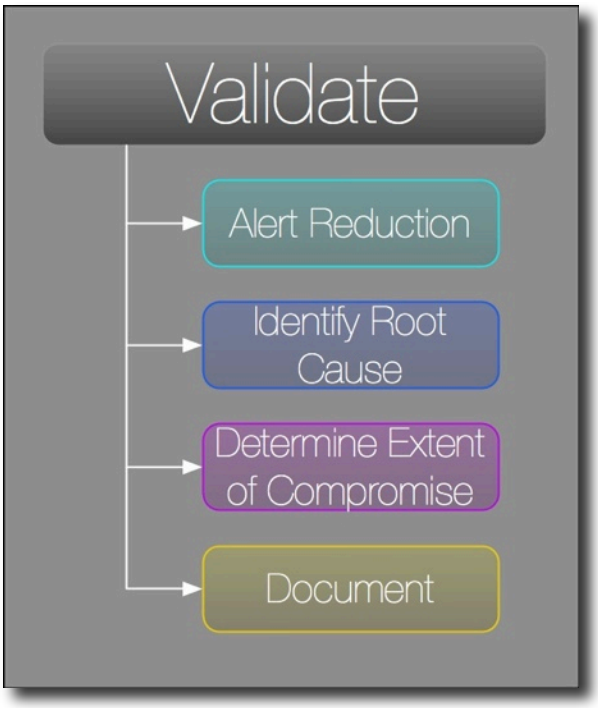
Process Step	Variable	Notes
Normalize Events/Data	Time to put events/data into a common format to facilitate analysis	Not all event logs and other data comes in the same formats so you'll need to massage the data to enable useful comparisons.
Correlate	Time to analyze data from multiple sources and identify patterns based on polices/rules	Seems like a job suited to Rain Man. Or a computer.
Reduce Events	Time to eliminate duplicate events	
	Time to eliminate irrelevant events	
	Time to archive events	Old and irrelevant events can clutter the analysis, so move them to archival storage.
Tune Thresholds	Time to analyze current thresholds and determine applicable changes	Based on accuracy of alerts generated.
	Time to test planned thresholds	It's helpful to be able to replay a real dataset to gauge the impact of changes.
	Time to deploy updated thresholds	
Trigger Alerts	Time to document alert with sufficient information for validation	The more detail the better, so the investigating analyst does not need to repeat work.
	Time to send alert to appropriate analyst	Workflow and alert routing defined during Policy Definition step.
	Time to open alert in tracking system	Assuming a ticketing system has been deployed.

Validate

When you get an *alert*, some set of conditions has been met which might indicate an attack. Great — that means you need to figure out whether there is a real issue or not. That's the *Validate* subprocess. In this step you work to understand what happened to generate the alert, assess the risk to your organization, and consider ways to remediate the issue. Pretty straightforward, right? In concept absolutely, but in reality not so much. Validation requires security professionals to jump into detective mode to piece together the data and build a story about the attack. Think *CSI*, but for real and without a cool lab. Add the pressure of a potentially company-damaging breach and you can see that validation is not for the faint of heart.

Let's jump into the subprocesses and understand how this is done. Keep in mind that different professionals go through these steps in different orders — depending on systems, instrumentation, preferences, and skill set.

Here are the subprocesses for Validate:



1. **Alert reduction:** The first action after an alert fires is to verify the data. Does it make sense? Is it a plausible attack? You need to eliminate the obvious false positives (and then get that feedback to the team generating alerts) — basically applying a sniff test to the alert. A typical attack touches many parts of the technology infrastructure, so you'll have a number of alerts triggered by the same attack. Part of the art of incident investigation is to understand which alerts are related and which are not. Mechanically, once you figure out which alerts may or may not be related, you need to merge them in whatever system you're using to track the alerts — even if it's Excel.
2. **Identify root cause:** If you have a legitimate alert, now you need to dig into the specific device(s) under attack and begin forensic analysis to understand what is happening and the extent of the damage at the device level. This may involve log analysis, configuration checks, malware reverse engineering, memory analysis, and a host of other techniques. The goal of this analysis is to establish the root cause of the attack, so you can start figuring out what is required to fix it — whether it's a configuration change, firewall or IDS/IPS change, workaround, or something else. Feedback on the effectiveness of the alert — and how to make it better — then goes back to whoever manages the monitoring rules & policies.
3. **Determine extent of compromise:** Now that you understand the attack, you need to figure out whether this was an isolated situation or you have badness proliferating through the environment, by analyzing other devices for indications of similar attacks. This can be largely automated with scanners, but not entirely. When you find another potentially compromised device you validate once again — now much quicker, since you know what you're looking for.

4. **Document:** As with the other steps, documentation is critical to make sure other folks know what's happened, so the ops teams know enough to fix the issue, and so your organization can learn from the attack (post-mortem). Here you will close the alert and write up the findings in sufficient detail to inform other folks of what happened, how you suggest they fix it, and what defenses need to change to make sure this doesn't happen again.

Large vs. Small Company Considerations

Finding the time to do real alert validation is a major challenge, especially for smaller companies that don't have resources or expertise for heavy malware or log analysis. The best advice we can give is to have a structured and repeatable process for validating alerts. That means defining the tools and steps involved in investigating an alert in fairly granular detail.

Why go to this level? Because a documented process will help you figure out when you are in deep water (over your head on the forensic analysis), and help you decide whether to continue digging or just re-image the machine/device and move on. Maybe it's clear some kind of Trojan got installed on an endpoint device. Does it matter which one, and what it did to the device? Yes it does, because you should try to understand the attack vector to make sure your defenses adapt based on successful attacks. But we do need to be pragmatic here and be realistic about the amount of time and effort required for this kind of forensic investigation.

So you have a decision to make — if you are just going to re-image the device and start over, you may decide to skip the analysis. Of course that doesn't enable you to really understand how your defenses need to change to protect against this successful attack, but you should at least be able to update rules to more quickly identify the effects of the same attack next time since you've seen it before, even if you don't fully understand the root cause.

Larger companies also need this level of documentation on how alerts are validated because they tend to have tiers of analysts. The lower-level analysts run through a series of steps to try to identify the issue. If they can't come up with a good answer they pass the question along to a group of more highly trained analysts who can dig deeper. Finally, you may have a handful of ninja type forensic specialists (hopefully at least one) who tackle the nastiest stuff. The number of levels doesn't matter — just that you have the responsibilities and handoffs between tiers defined.

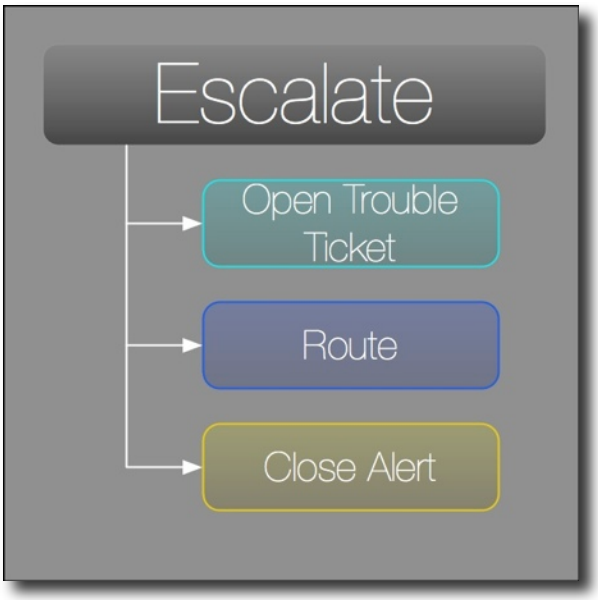
Validate Metrics

Process Step	Variable	Notes
Alert Reduction	Time to determine which alerts reflect the same incident	Don't waste time on overlapping investigations of a single issue.
	Time to merge alerts in the system	
	Time to verify the alert data & eliminate false positives	At the end of this step, you need to know whether the alert is legitimate.
Identify Root Cause	Time to find device under attack	
	Time for forensic analysis to understand attack specifics	What does the attack do? What's the impact on the monitored devices?
	Time to establish root cause and specific attack vector	Once you understand what the attack does, then pinpoint how it happened.
	Time to identify possible remediations, workarounds, and/or escalation plan	Understanding how it happened allows you to put controls in place to ensure it doesn't happen again.
Determine Extent of Compromise	Time to define scan parameters to identify other devices vulnerable to attack	The goal is to quickly determine how many other devices have been similarly compromised.
	Time to run scan	
Document	Time to close alert, if false positive	
	Time to document findings with sufficient detail for remediation	The ops team will need to know exactly how to fix the issue. More detail provides less opportunity for mistakes.
	Time to log into ticketing system	

Escalate

Now that you have identified the attack and what's involved, it needs to be fixed — that is what escalation is about. Every company has a different idea of who needs to do what, so defining the escalation path is key. Don't forget the criticality of communicating these policies and managing expectations around responsiveness. Make sure everyone understands how and when something will fall into their lap, and what to do when it happens.

Here are the subprocesses for Escalate:



1. **Open Trouble Ticket:** You need a mechanism for notifying someone of their task/responsibility. That sounds obvious, but many security processes fail because separate teams don't communicate effectively, and plenty of things fall through the cracks. We aren't saying you need a fancy enterprise-class help desk system, but you do need *some* mechanism to track what's happening, who is responsible, and status — and to eventually close out issues. Be sure to provide enough information in the ticket to ensure the responsible party can do their job. Forcing them to come back to you over and over again to get essential details is horribly inefficient.
2. **Route:** Once the ticket is open it needs to be sent somewhere. The routing rules & responsibilities are defined (and agreed upon) in the Planning phase, so none of this should be a surprise. You find the responsible party, send them the information, and follow up to make sure they got the ticket and understand what needs to be done. Yes, this step can be entirely automated with those fancy (or even not-so-fancy) help desk systems.
3. **Close Alert:** Accountability is critical to the ultimate success of any security program. So if the security team just sends an alert over the transom to the ops team and then washes their hands, we can promise things will fall between the cracks. The security team needs to follow up and ensure each issue is taken to closure. Again, a lot of this process can be automated, but we've found the most effective solution is to make sure someone's behind is on the line for making sure the alert gets closed.

Large vs. Small Company Considerations

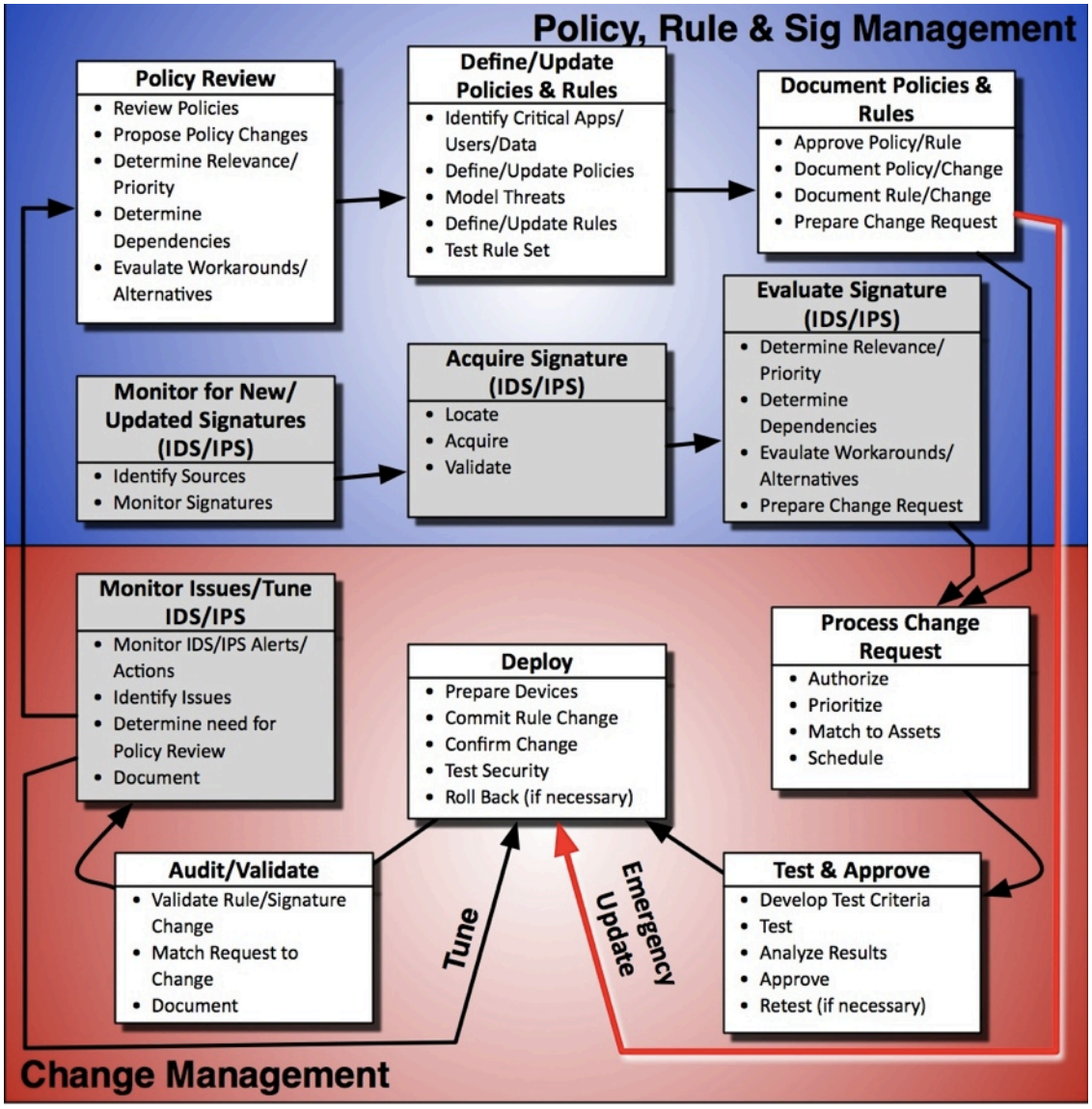
The biggest differences you'll see in this step between large and small companies is the number of moving pieces. Smaller companies tend to have a handful of security/ops folks at best, so there isn't a lot of alert handoff & escalation, because in most cases the person doing the analysis is fixing the issue and then probably tuning the monitoring system as well. Larger companies tend to have lots of folks involved in remediation and the like — so documentation, managing expectations, and follow-up are imperative to successfully closing any issue.

But we don't want to minimize the importance of documentation when closing tickets — even at smaller companies — because in some cases you'll need to work with external parties or even auditors. If you (or another individual) are responsible for fixing something you validated above, we still recommend filling out a ticket and documenting the feedback on rule changes even if you're effectively writing notes to yourself. We realize this is extra work, but it's amazing how quickly the details fade — especially when you are dealing with many different issues every day — and this documentation helps ensure you don't make the same mistake twice.

Escalate Metrics

Process Step	Variable	Notes
Open Trouble Ticket	Time to integrate/access trouble ticket system	Ops team may use a different system than security. Stranger things have happened.
	Time to open trouble ticket	Be sure to include enough information to assist in troubleshooting.
Route Appropriately	Time to find/confirm responsible party	Should be specified in policy definition but things change, so confirmation is a good idea.
	Time to send alert	
	Time to follow up and answer questions	Depending on how segmented the operational responsibilities are from monitoring, you may not be able to close the ticket until all the questions from ops are answered and they accept the ticket.
Close Alert	Time to follow up and ensure resolution	Depending on lines of responsibility, this may not be necessary. Once the ticket is routed, that may be the end of security team involvement.
	Time to close alert	

Manage Process



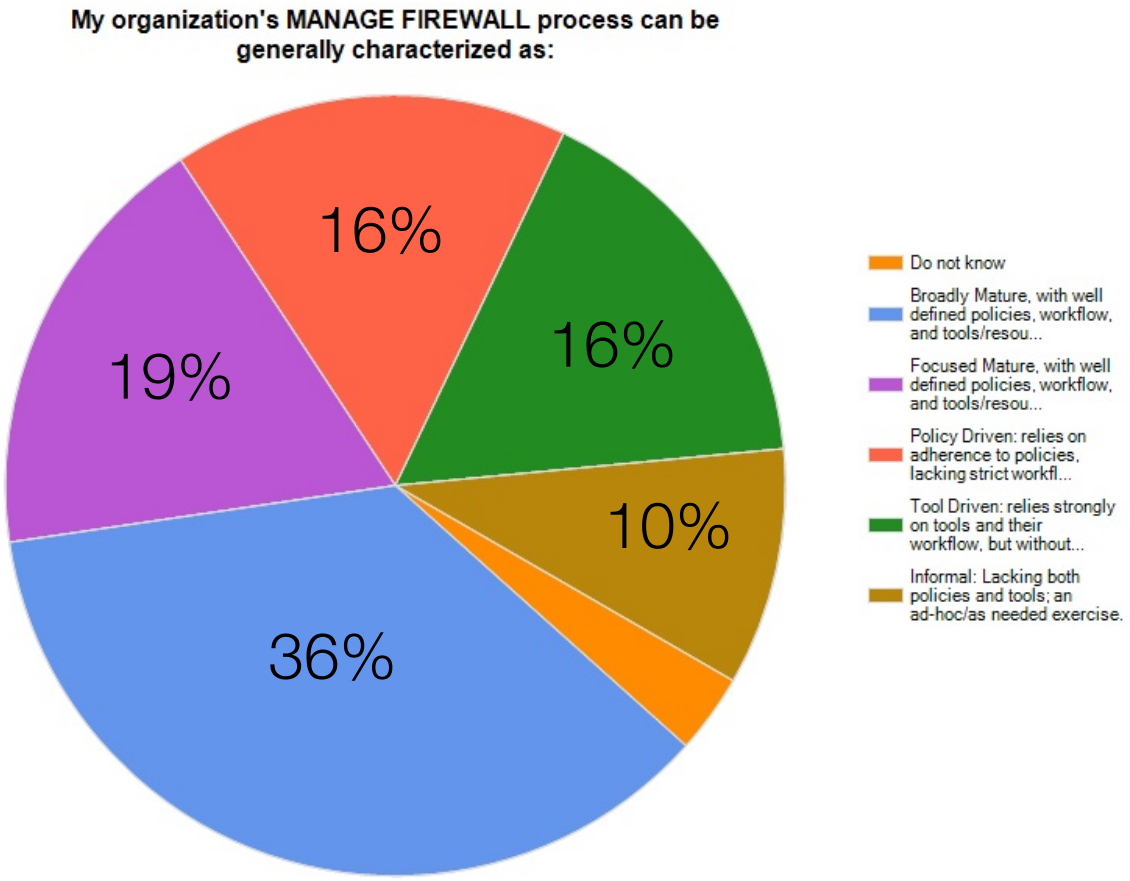
Now let's dig deeper into the Manage process and break each high level step (described in the Introduction) down into its subprocesses and associated operational metrics. Here are links to each of the applicable process steps:

1. Policy Review
2. Define/Update Policies and Rules
3. Document Policies and Rules
4. Monitor for New/Updated Signatures (IDS/IPS)
5. Acquire Signatures (IDS/IPS)
6. Evaluate Signatures (IDS/IPS)
7. Process Change Request
8. Test and Approve
9. Deploy
10. Audit/Validate
11. Monitor for Issues/Tune (IDS/IPS)

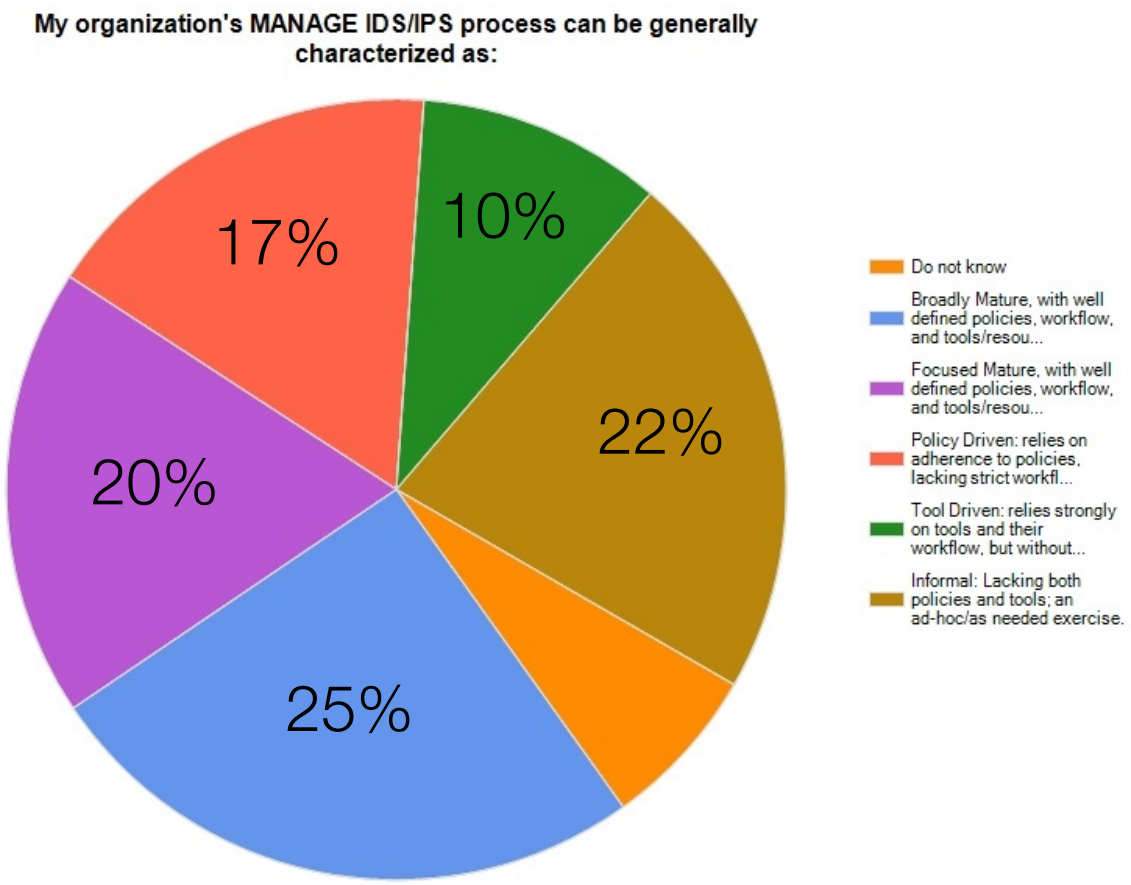
There are significant synergies between managing firewalls and IDS/IPS. Obviously their purposes are different, as are their detection techniques and rules options, but the defining policies and rules are relatively similar across device types, as is change management. You need to explicitly manage signatures and other detection content on an IDS/IPS, so we've broken out some additional subprocess for those specific functions.

Manage Process Maturity

Along with our primary research, we also surveyed 75 end users (in organizations ranging from small business to mega-enterprise) about what processes they have in place. We did this primarily to validate our primary research, but also to get a firm grasp on what many organizations do in practice — as opposed to purely theoretical process modeling.



In terms of the maturity of each function, there are clear disparities in how organizations manage firewalls (a very mature process) compared to managing IDS/IPS devices (reasonably mature). That makes sense given the added complexity of managing and tuning IDS/IPS rules, as well as the ubiquity and maturity of firewalls as a security control.



Policy Review

Although it should happen periodically, far too many folks rarely (or even never) go through their network security device policies and rules to clean up and account for ongoing business changes. Yes, this failure creates security issues. Yes, it also creates management issues, and obsolete and irrelevant rules can place an unnecessary burden on the devices. In fact, obsolete rules tend to have a larger impact on an IDS/IPS than on a firewall, due to the processing overhead of IDS/IPS rules. So at a minimum there should be a periodic review (perhaps twice a year) to evaluate the rules and policies, and make sure everything is up to date.

We see two other main catalysts for policy review:

1. **Service Request:** This is when someone in the organization needs a change to the rules on the device(s), typically driven by a new application or trading partner who needs access to something or other.
2. **External Advisory:** At times, when a new attack vector is identified, one way to defend against it is to set up a detection rule on a network security device. This involves monitoring the leading advisory services and using their information to determine whether a policy review is necessary.

Once you have decided to review policies, we have identified five subprocesses in policy review:

1. **Review Policies:** The first step is to document the latest version of the polices; then you'll research the requested changes. This gets back to the catalysts mentioned above. If it's a periodic review you don't need a lot of prep work, but reviews prompted by user requests require you to understand the request and its importance. If the review is driven by a clear and present danger, you need to understand the nuances of the attack vector to understand how you can detect and/or block the attack.
2. **Propose Policy Changes:** Once you understand why you are making the changes, you'll be able to recommend *policy* changes. These should be documented in as much detail as possible, both to facilitate evaluation and authorization, and also to maintain an audit trail of why specific changes were made.
3. **Determine Relevance/Priority:** Now that you have a set of proposed changes it's time to determine its priority. This is based on the importance of the assets behind the network security device(s) and the catalyst for change. You'll also want criteria for an *emergency update*, which bypasses most of the change management processes in case of an urgent situation.

4. **Determine Dependencies:** Given the complexity and interconnectedness of our technology environment, even a fairly simple change can create ripples that result in unintended consequences. So analyze the dependencies *before* making changes. If you turn certain rules on or off, or tighten their detection thresholds, what business processes and users will be impacted? Some organizations manage by complaint by waiting until users scream about something broken after a change. That is one way to do it, but most at least give users a 'heads up' when they decide to break something.
5. **Evaluate Workarounds/Alternatives:** A rule change may not be the only option for defending against an attack or supporting a new application. As part of due diligence, you should include time to evaluate workarounds and alternatives. In this step, determine any potential workarounds and/or alternatives, and evaluate their dependencies and effectiveness, so you can objectively choose the best option.
6.

Policy Review Metrics

Process Step	Variable	Notes
Review Policies	Time to isolate relevant policies	Based on the catalyst for policy review (attack, signature change, false positives, etc.).
	Time to review policy and list workarounds & alternatives	Focus on what changes you could and should make without judging them — that comes later.
Propose Policy Changes	Time to gain consensus on policy changes	Some type of workflow/authorization process must be defined well ahead of the time to actually review and define policies.
Determine Relevance/ Priority	Time to prioritize policy/rule changes	Based on the risk of attack and the value of the data protected by the device.
Determine Dependencies	Time to determine whether additional changes are required to implement policy update	Do other policies/rules need to change to enable this update? What impact will it have on existing policies/ rules?
Evaluate Workarounds/ Alternatives	Time to evaluate the list of alternatives & workarounds for feasibility	Sometimes a different control will make more sense than a policy/rule change.

Define/Update Policies & Rules

The world is a dynamic place with all sorts of new attacks continually emerging, so defining policies and rules is an iterative process. You need an ongoing process to update the policies as well.

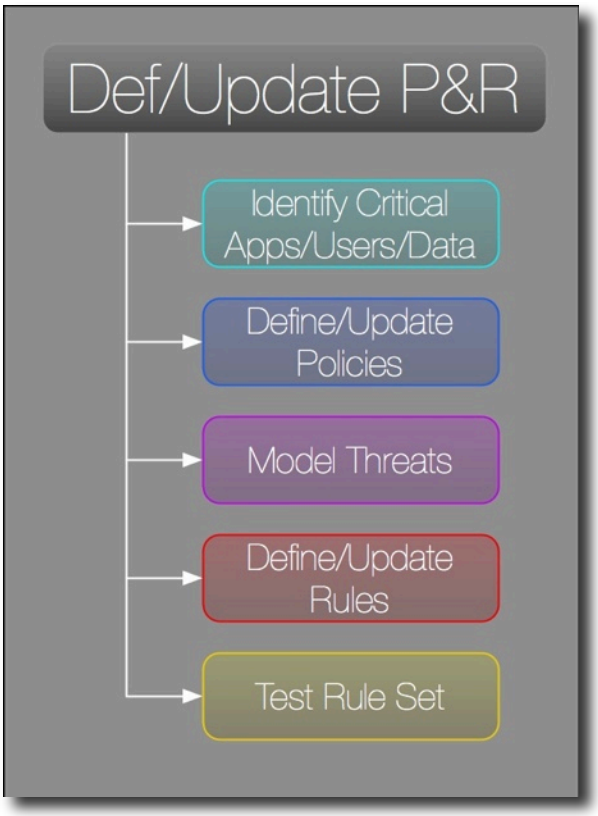
To be clear, the high level policies should be reasonably consistent for all your network security gear. The scope of this research includes Managing firewalls and IDS/IPS, but the same high-level policies would apply to other devices (such as email and web filtering gateways, SSL VPN devices, network DLP gateways, etc.). What will be different, of course, are the rules that implement the policies.

So this step focuses on understanding what we need to protect and building a set of policies to do it. But before we dig in we should clarify what we mean by *policies* and *rules*, because many folks (including Securosis, at times) use the terms interchangeably. For this series we define the **policy** as the high-level business-oriented description of what you need to protect. For example, you may need to protect credit card data to comply with PCI — that would be a policy. These are high-level and distinct from the actual implementation rules which would be installed on the firewall to implement them.

Rules implement the policy within the device. So you'd need to block (or allow) certain ports, protocols, users, networks and/or applications (each a separate rule) during certain time periods to implement each policy. There will be overlap, because the "protect credit card data" policy involves some of the same rules as a "protect private health information" policy. Ultimately you need to bring everything you do back to a business driver, and this is one of the techniques for doing that.

Given the amount of critical data you have to protect, building an initial set of policies can seem daunting. We recommend use cases for building the initial policy set. This means first identify the critical applications, users, and/or data to protect, and the circumstances for allowing or blocking access (location, time, etc.). This initial discovery process will help when you need to prioritize enforcing rules against inconveniencing users, because you always need to strike a balance between these two imperatives. Given those use cases you can define the policies, then model the potential threats to applications, users, and data. Your rules address the attack vectors identified through the threat model. Finally you need to stage/test the rules before full deployment to make sure everything works.

More specifically, we have identified five subprocesses in defining and updating these policies/rules:



1. **Identify Critical Applications/Users/Data:** Here we discover what we need to protect. The good news is that you should already have at least some of this information, most likely through the Define Policies Subprocess. While this may seem rudimentary, it's important *not* to assume you know what is important and what needs to be protected. This means doing technical discovery to see what's out there, as well as asking key business users what applications/users/data are most important to them. Take every opportunity you can to get in front of users to listen to their needs and evangelize the security program. For more detailed information on discovery, check out Database Security Quant on Database Discovery.
2. **Define/Update Policies:** Once the key things to protect are identified, we define the base policies. As described above, the policies are the high-level *business-oriented* statements of what needs to be protected. For policies worry about *what* rather than *how*. It's important to prioritize them as well, because that helps with essential decisions on which policies go into effect and when specific changes happen. This step is roughly the same whether policies are being created or updated.
3. **Model Threats:** Similar to the way we built correlation rules for monitoring, we need to break down each policy into a set of attacks, suspect behavior, and/or exploits which could be used to violate the policy. Put yourself in the shoes of a hacker, and think like them. Clearly there are an infinite number of attacks that can be used to compromise data, so fortunately the point isn't to be exhaustive — it's to identify the most likely threat vectors for each policy.
4. **Define/Update Rules:** Once the threats are modeled it's time go one level down and define how you'd detect the attack using a specific device (firewall or IDS/IPS for this project). This may involve a combination of signatures, traffic analysis, heuristics, etc. Consider when these rules should be in effect (24/7, during business hours, or on a special schedule) and whether they have an expiration date (such as when a joint development project ends or a patch is available). This identifies the base set of rules to implement a policy. Once you've been through each policy you need to get rid of duplicates and see where the leverage is.
5. **Test Rule Set:** The old adage about *measure twice, cut once* definitely applies here. Before implementing any rules, we strongly recommend both testing the specific attack vectors and some regression analysis to avoid breaking existing rules during implementation. You'll need to identify and perform a set of tests for the rules being defined and/or updated. To avoid testing on a production box it's extremely useful to have a network perimeter testbed to implement new and updated rules; this can be leveraged for all network security devices. If any of the rules fail, you need to go back to the Define/Update Rules step and fix. Cycle through Define/Update/Test until the tests pass.

Default Deny and Application Awareness

We know defining all these policies can be daunting. But there are ways to make it a bit easier, and the first is to adopt a default deny perimeter security posture. That means unless you specifically authorize certain traffic to go through the firewall, *the traffic gets blocked*. Each of the rules is about configuring the port, protocol, and/or application to enable an application or user to do their job.

Obviously there is only so much granularity you can apply at the firewall level, which is driving interest in application-aware firewalls which can block certain applications for certain users. You have to allow port 80 in (to the DMZ at minimum) and out of your network, so the more granular you can get within a specific port by application or business use, the better.

Keep in mind that all the commercial (and even some open source) firewalls and IDS/IPS devices ship with sets of default policies and associated rules that can be easily customized to your environment. We recommend you work through the process and then compare your requirements against your available out-of-the-box policies because you want to implement the rules that apply to your environment, not the vendor's generic set.

Default Rule Sets

Given the complexity of making these network security devices work, especially in relatively complicated environments, we see many (far too many) organizations just using the default policies and/or rule sets that come with the devices. You need to start somewhere, but the default rules usually reflect the lowest common denominator. So you can start with that set, but we advocate clearing out the rules that don't apply to your environment and going through this procedure to define your threats, model them, and define appropriate rules. We know it's complicated, but you've spent money on the gear so you might as well get some value from it.

Define/Update Policies & Rules Metrics

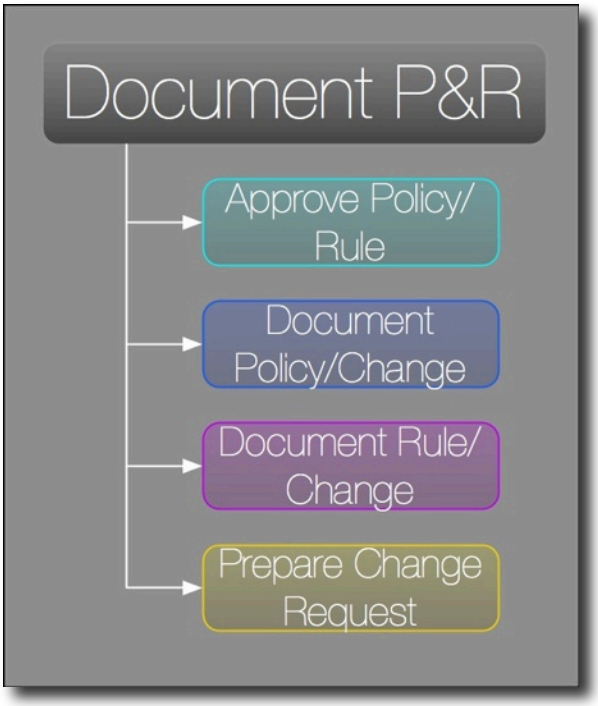
Process Step	Variable	Notes
Identify Critical Applications/ Users/Data	Time to identify critical applications/users/data in use	Involves discussions with key influencers (business and technical), as well as technical discovery to ensure nothing is missed.
Define/Update Policies	Time to find relevant policy examples	There is no need to reinvent the wheel. Lots of examples (books, websites, peers) can provide a head start on defining policies.
	Time to customize the policies for your organization	
	Time to gain consensus on policies	It's important to ensure all interested parties weigh in on the policies, because implementing is very difficult without broad support.
	Time to identify attack vectors & patterns for each policy	

Process Step	Variable	Notes
Model Threats	Time to gather baselines from current environment	Baselines help to identify normal behavior and to make sure there aren't any holes in the policies based on what really happens on your network.
	Time to build applicable threat model	Based on baselines and other potential threat vectors. You can't model every possible threat, so focus on those which present the greatest risks to critical data.
Define/Update Rules	Time to find relevant rule examples	There are numerous examples of publicly available firewall and IDS/IPS rule bases to start with.
	Time to customize specific rules to run on firewall and/or IDS/IPS	Make sure all the threats identified in the model are protected against.
	Time to determine actions to take, based on rules firing	Do you want to block, alert, log, or take some other action when a specific rule is violated?
	Time to gain consensus on rules and actions	It's important to ensure all interested parties weigh in on the rules and especially the actions, because these decisions impact business operations.
Test Rule Set	Time to design testing scenario	
	Time to set up test bed for system test	Many organizations build a simple lab to test rule changes.
	Time to simulate attack	It's probably not a good idea to take down a production network, but you need the tests to approximate reality as closely as possible.
	Time to analyze test data to ensure proper detection and/or blocking of attacks	
	If any of the tests fail, time to update rules to address the issues and to retest	In the event of a minor issue, a quick change and retest may suffice. For a significant issue you likely need to go back to the define/update policy step and restart.

Document Policies & Rules

Keep in mind the level of documentation you need for your environment varies based on culture, regulatory oversight, and (to be candid) 'retentiveness' of the security team. We are fans of *just enough* documentation. To be fair, there are multiple documentation steps in the NSO Quant processes, and that's for an important reason. You need to be able to substantiate your controls (especially to the auditors) and ensure your successor knows how and why you did certain things. *But there is no point spending all your time documenting rather than doing.* Obviously you have to find the right balance, but clearly you want to automate as much of this process as you can.

We have identified 4 subprocesses in the policy/rule documentation step:



1. **Approve Policy/Rule:** The first step is to get approval for the policy and/or rule (refer to Define/Update for definitions of policies and rules), whether it's new or an update. We strongly recommend having this workflow defined *before* you put the operational process into effect, especially if there are operational handoffs required before actually making the change. You don't want to step on a political land mine by going around a pre-determined handoff in the heat of trying to make an emergency change. That kind of action makes operational people very grumpy. Some organizations have a very formal process with committees, while others use a form within their help desk system to provide very simple separation of duties and an audit trail — of the request, substantiation, approver, etc. Again, don't make this harder than it needs to be, but you need some formality.
2. **Document Policy/Change:** Once the change has been approved it's time to write it down. We suggest using a fairly straightforward template which outlines the business need for the policy and its intended outcome. Remember policies consist of high-level, *business-oriented* statements. The documentation should already be about ready from the approval process. Now make sure it gets filed correctly.
3. **Document Rule/Change:** This is equivalent to the Document Policy Change step, except here you are documenting the actual rules so the operations team can make the change.
4. **Prepare Change Request:** Finally we take the information from the documentation and package it up for the operations team. Depending on your relationship with ops, you may need to be *very* granular with the specific instructions. This isn't always the case but we make a habit of not leaving much to interpretation because that leaves an opportunity for things to go haywire. Again we recommend some kind of standard template, and don't forget to include some context for why the change is being made. You don't need a full business case (as when preparing the policy or rule for approval), but if you include *some* justification you have a decent shot at avoiding a request for more information from ops, which would mean delay while you convince them to make the change.

Emergency Updates

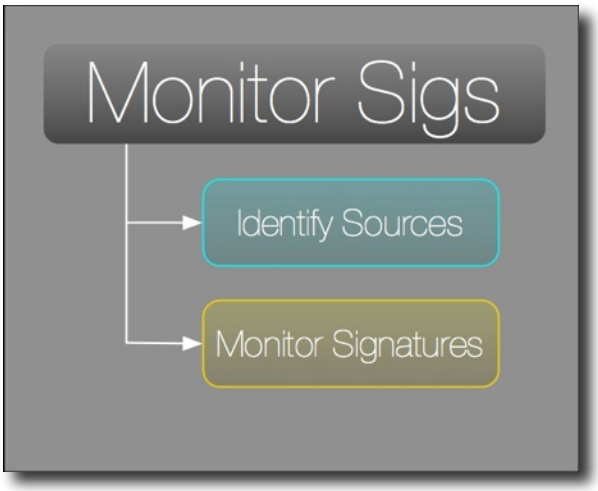
In some cases — including data breach lockdowns, imminent zero-day attacks, and false positives impacting key business processes — a change to the network security device ruleset must be made immediately. *A process to circumvent the broader change process should be established and documented in advance*, ensuring proper approval/authorization for such rushed changes, and that there is a rollback capability in case of unintended consequences.

Document Policies & Rules Metrics

Process Step	Variable	Notes
Approve Policy/ Rule	Time to evaluate policy and/or rule change	Based on policy review documentation, risk of attack vectors, and existing process documents.
	Time to get approval from authorized parties	Involves discussions with key influencers (business and technical). There are multiple levels of approval, not all applicable to your environment. But it's important to make sure everyone is on board with the policies and rules.
Document Policy/Change	Time to document policy change(s)	Amount of time varies based on number of policies/updates to document, time per policy to document, existence & completeness of current documentation, and degree of automation.
Document Rule/ Change	Time to document rule change(s)	Amount of time varies based on number of rules/updates to document, time per rule to document, existence/completeness of current documentation, and degree of automation.
Prepare Change Request	Time to package requested changes into proper format	Based on the granularity of change authorization process.

Monitor for New/Updated Signatures (IDS/IPS)

Unfortunately there is no stork that delivers relevant and timely IDS/IPS signatures to your doorstep as you sleep; so you need to do the grunt work of figuring out which detection techniques need to be added, updated, and switched off on your IDS/IPS devices. The first step is to figure out what is new or updated, and we have identified a couple steps in this subprocess:



1. **Identify Sources:** You need to identify potential sources of advisories. In most cases this will be the device vendor, and their signature feeds are available through the maintenance relationship. Many organizations use open source IDS engines, most or all of which use Snort rules. So these users need to monitor the Snort updates, which are available via the SourceFire VRT premium feed or 30 days later in the public feed. It also makes sense to build periodic checks into the workflow, to ensure your advisory sources remain timely and accurate. Reassessing sources a couple times per year should suffice.
2. **Monitor Signatures:** This is the ongoing process of monitoring your sources for updates. Most can be monitored via email subscriptions or RSS feeds.

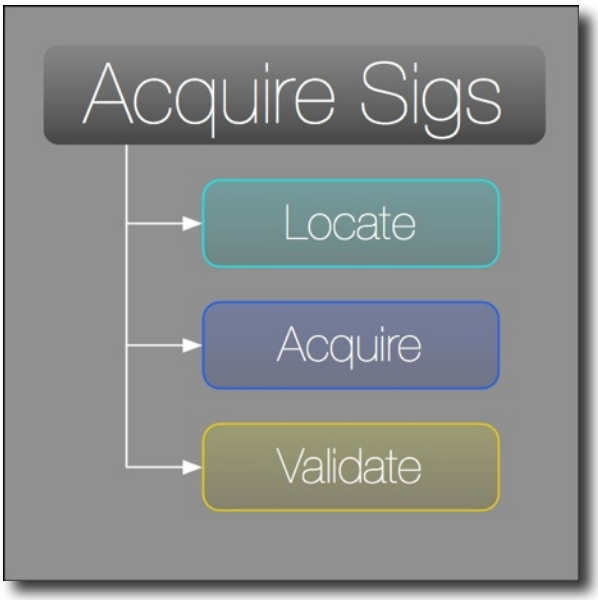
Once you know you want to use a new or updated signature you need to get it and prepare the documentation to get the change made by the operations team.

Monitor for New/Updated Signatures Metrics

Process Step	Variable	Notes
Identify Sources	Time to identify sources of signatures	Could be vendor lists, newsgroups, blogs, etc...
	Time to validate/assess sources	Are the signatures good? Reputable? How often do new signatures come out? What is the overlap with other sources?
Route Appropriately	Time for ongoing monitoring of signature sources	The amount of time will vary based on the number of sources and the time it takes to monitor each.
	Time to assess relevance of signatures to environment	Does the signature pertain to the devices/data you are protecting? There is no point looking at Linux attack signatures in Windows-only shops.

Acquire Signatures (IDS/IPS)

The next step in managing IDS/IPS signatures is actually getting them. We understand that's obvious, but when charting out processes (in painful detail, we know!) we cannot skip any steps.



1. **Locate:** Determine the location of the signature update. This might involve access to your vendor's subscription-only support site or even physical media.
2. **Acquire:** Download or otherwise obtain the new/updated signature.
3. **Validate:** Confirm that the new/updated signature uses proper syntax and won't break any devices. If the signature fails validation, you'll need to figure out whether to try downloading it again, fix it yourself, or wait until it's fixed. If you are a good samaritan, you might even want to let your source know it's broken.

For Snort users, the oinkmaster script can automate many of these monitoring and acquisition processes. Of course commercial products have capabilities built into their various management consoles.

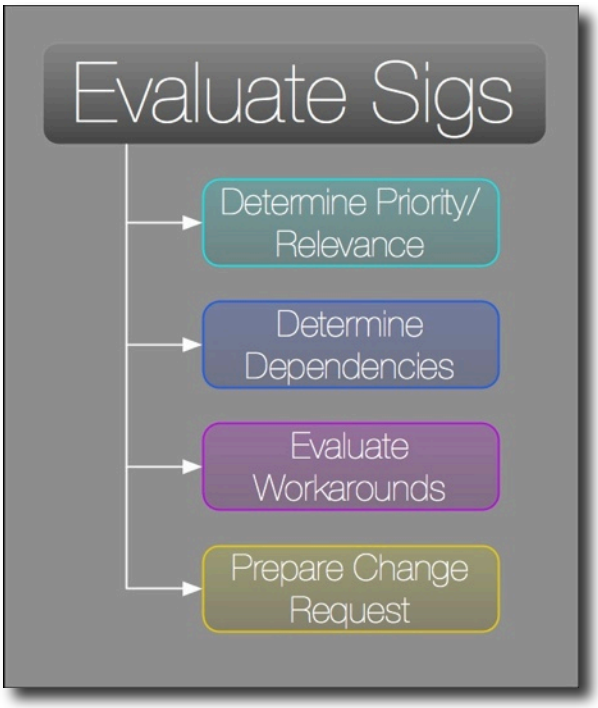
Once you have signature updates you'll figure out whether you need to use them.

Acquire Signatures Metrics

Process Step	Variable	Notes
Locate	Time to find signature	Optimally a script or feed tells you a new signature is available (and where to get it).
Acquire	Time to download signature	This shouldn't take much time (or can be automated), unless there are onerous authentication requirements.
Validate	Time to validate signature was downloaded/acquired properly	Check both the integrity of the download and the signature/rule itself.

Evaluate Signatures (IDS/IPS)

Just because you have access to a new or updated signature doesn't mean you should use it. The next step is to evaluate the signature/detection technique and figure out whether and how it fits into your policy/rule framework. The evaluation process is very similar to reviewing device policies/rules, so you'll recognize similarities to Policy Review.



1. **Determine Relevance/Priority:** Now that you have a set of signatures you'll need to determine priority and relevance for each. This varies based on the type of attack the signature addresses, as well as the value of the assets protected by the device. You'll also want criteria for an *emergency update*, which bypasses most change management processes in case of an emergency.
2. **Determine Dependencies:** It's always a good idea to analyze dependencies *before* making changes. If you add or update certain signatures, what business processes/users will be impacted?
3. **Evaluate Workarounds:** It turns out IDS/IPS signatures mostly serve as workarounds for vulnerabilities or limitations in other devices and software — such as firewalls and application/database servers — to handle new attacks (especially in the short term, as adding a signature may be much quicker than a complete fix at the source), but you still need to verify the IDS/IPS signature change is the best option.
4. **Prepare Change Request:** Finally, take the information in the documentation and package it for the operations team. We recommend some kind of standard template, and don't forget to include context (justification) for the change.

We aren't religious about whether you acquire or evaluate the signatures first. But given the ease (and automation) of monitoring and acquiring updates, it may not be worth the effort of running separate monitoring and acquisition processes — it might be simpler and faster to grab everything automatically, then evaluate it, and discard the signatures you don't need.

Evaluate Signatures Metrics

Process Step	Variable	Notes
Determine relevance/ priority	Time to prioritize importance of signature	Based on attack type and risk assessment. May require an emergency update, which jumps right to deploy step (after proper authorization).
Determine Dependencies	Time to understand impact of new signature on existing signatures/rules	Will this new update break an existing rule, or make it obsolete? Understand how the new rule will affect the existing rules.
Evaluate Workarounds	Time to evaluate other ways of blocking/detecting the attack	Verify that this signature/rule is the best way to detect/block the attack. If alternatives exist, consider them.
Prepare Change Request	Time to package requested changes into proper format	Detail in the change request depends on the granularity of the change authorization process.

Process Change Request

Now that we've been through managing the policies/rules and signatures that drive our network security devices, and developed change request systems for both rule changes and signature updates, it's time to make whatever changes have been approved. That means you must transition from a policy/architecture perspective to operational mode. The key here is to make sure every change has exactly the desired impact (which means a lot of testing), and that you have a rollback option in case of an unintended consequence such as blocking a critical application.

A significant part of the Policy Management section is to document the change request. Let's assume it comes over the transom and ends up in your lap. We understand that in smaller companies the person managing policies and rules may very well also be making the changes, but processing the change still requires its own process — if only for auditing and separation of duties.

The subprocesses are as follows:

1. **Authorize:** Wearing your operational hat, you need to first authorize the change. That means adhering to the predetermined authorization workflow to verify the change is necessary and approved. Usually this involves both a senior level security team member and an ops team member *formally* signing off. Of course this should be documented in some system to provide an audit trail.
2. **Prioritize:** Determine the overall importance of the change. This will often involve multiple teams — especially if it impacts applications, partners, or key business functions. Priority is usually a combination of factors, including the potential risk to your environment, availability of mitigating options (workarounds/alternatives), business needs or constraints, and importance of the assets affected by the change.
3. **Match to Assets:** After determining the overall priority of the rule change, match it to specific assets to determine deployment priorities. The change may be applicable to a certain geography or locations that host specific applications. Basically, you need to know which devices require the change, which directly affects the deployment schedule. Again, poor asset documentation makes this analysis more expensive.
4. **Schedule:** Now that the priority is established and matched to specific assets, build out the deployment schedule. As with the other steps, documentation quality is extremely important here, which is why we continue to focus on it during every step of the process. The schedule also needs to account for any maintenance windows and may involve multiple stakeholders, as it is coordinated with business units, external business partners, and application/platform owners.

Process Change Request Metrics

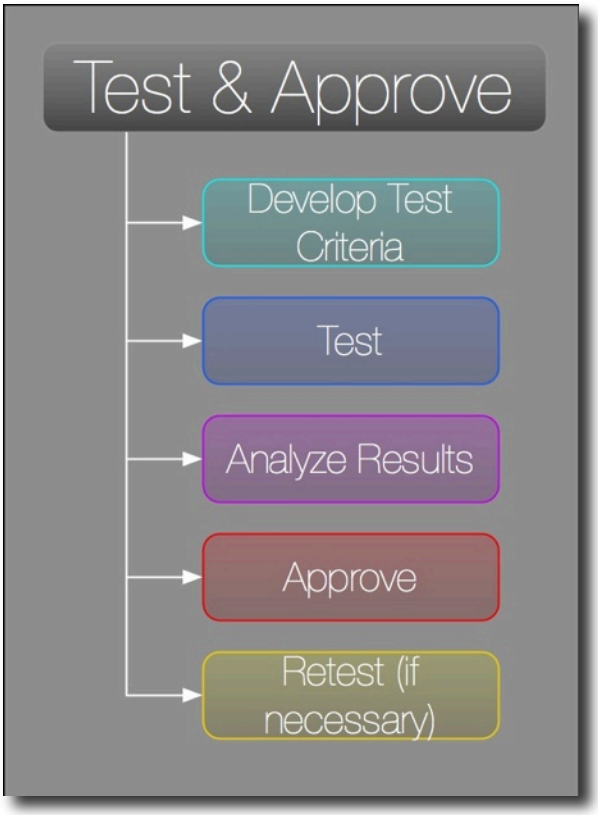
Process Step	Variable	Notes
Authorize	Time to verify proper authorization for request	Varies based on number of approvers, maturity of change workflow, device details, and automation level of change process.
Prioritize	Time to determine priority of change based on risk of attack and value of data protected by device	Varies based on number of requested changes, as well as completeness & accuracy of change request.
Match to Assets	Time to match the change to specific devices	Not all changes are applicable to all devices, so a list of devices to change must be developed and verified.
Schedule	Time to develop deployment schedule around existing maintenance windows	Varies based on number of requested changes, number of devices being changed, availability of maintenance windows, and complexity of changes.

Test & Approve

Still on the operational side of change management, we need to ensure whatever change to the network security devices is being processed won't break anything. That means testing the change and then moving to final approval before deployment.

We should be clear that *operational testing* here is different than a test in the policy management phase. The policy, rules, and signature management test (in the Define/Update Rules & Policies step) really focuses on functionality and making sure the suggested change solves the problems identified during Policy Review. This operational test is to make sure nothing breaks. Yes, the functionality needs to be confirmed later in the process (during Audit/Validation), but *this* test is about making sure there are no undesired consequences of the requested change.

We have identified four discrete steps for the Test and Approve subprocess:



1. **Develop Test Criteria:** Determine the specific testing criteria for the IDS/IPS changes and assets. These should include installation, operation, and performance. The depth of testing varies depending on the assets protected by the device, the risk driving the change, and the nature of the change.
2. **Test:** Perform the actual tests.
3. **Analyze Results:** Review the test data. You will also want to document it, both for the audit trail and in case of problems later.
4. **Approve:** Formally approve the rule change for deployment. This may involve multiple individuals from different teams (who hopefully have been in the loop all along), so factor any time requirements to get to these folks into your schedule.
5. **Retest**: This phase also includes one or more sub-cycles if a test fails and triggers additional testing or reveals other issues. This may involve adjusting the test criteria, environment, or other factors to achieve a successful outcome.

We assume that the ops team has a proper test environment(s) and tools, although we are well aware of the hazards of such assumptions. Remember that proper documentation of assets is necessary for quickly finding assets and troubleshooting issues.

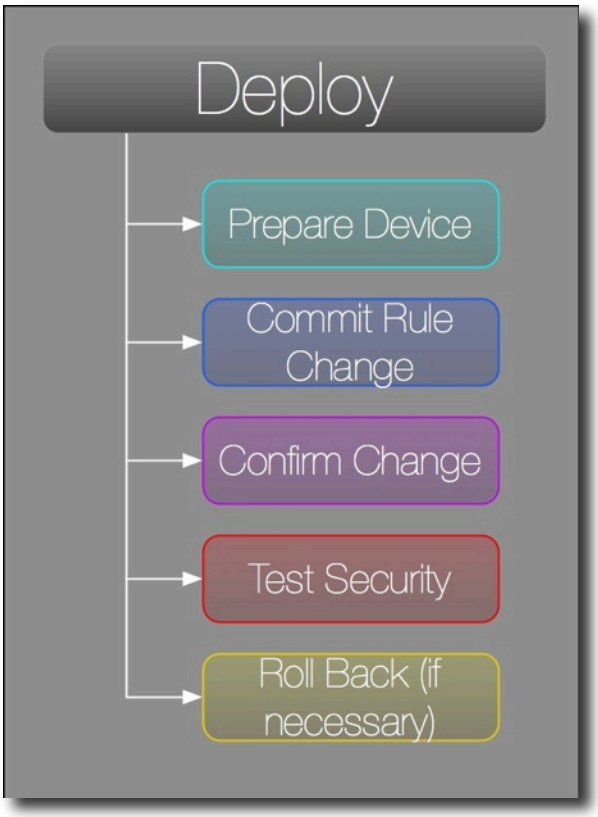
Test & Approve Metrics

Process Step	Variable	Notes
Develop Test Criteria	Time to develop specific firewall or IDS/IPS testing criteria	Should be able to leverage the testing scenarios from the planning phase.
Test	Time to test the change for completeness and intended effects	
Analyze Results	Time to analyze results and document tests	Documentation is critical for troubleshooting (if the tests fail) and for compliance.
Approve	Time to gain approval to deploy change(s)	Varies based on the number of approvals needed to authorize deployment of the requested change (s).
Retest	Time required to test scenarios again until changes pass or are discarded	Varies based on the amount of work needed to fix change(s) or amend criteria for success.

Deploy

Hooray! We are finally finished with planning and ready to actually *do something*. So now we can dig into deploying the new/updated rules and/or signatures.

We have identified 4 separate subprocesses involved in deploying a change:



1. **Prepare Device:** Prepare the target devices(s) for the change(s). This includes activities such as backing up the last known good configuration and rule/signature set, rerouting traffic, rebooting, logging in with proper credentials, and so on.
2. **Commit Rule Change:** Within the device management interface (or third party tool), make the rule/signature change(s). Make sure to clean up any temporary files or other remnants from the change and return the system to operational status.
3. **Confirm Change:** Consult the rule/signature base once again to confirm the change took effect.
4. **Test Security:** You may be getting tired of all this testing, but ultimately making any changes on critical network security devices is dangerous business. We advocate constant testing to avoid unintended consequences which could create significant security exposure, so you'll be testing the new changes. You have test scripts from the Test and Approve step to ensure the rule change delivered the expected functionality. We also recommend a general vulnerability scan of the device to ensure it is functioning and firing alerts properly.

What happens if the change fails the security tests? The best option is to roll back the change *immediately*, figure out what went wrong, and then repeat the deployment with a fix. That's why backing up the last known good configuration during preparation is critical: so you can go back to a known-good configuration in seconds if necessary.

Deploy Metrics

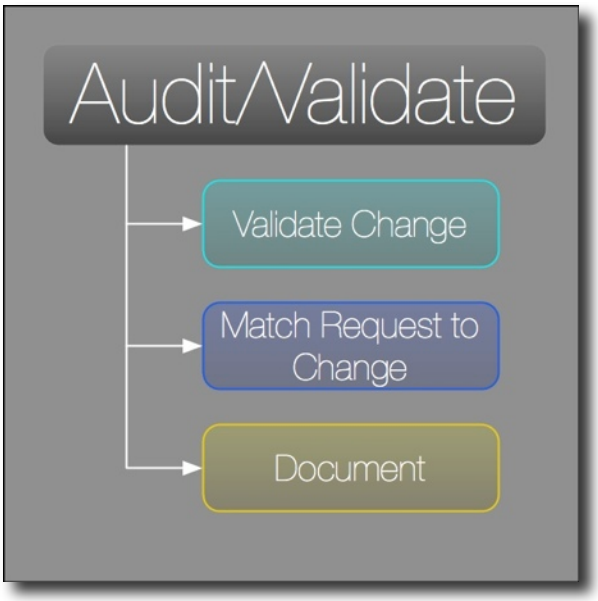
Process Step	Variable	Notes
Prepare Device	Time to prepare the target device	Varies based on number of changes, time to prepare each device, automation for making changes, and requirement (if any) to back up device before change.
Commit Rule Change	Time to make change	Based on automation, number of changes, and number of different devices affected.
Confirm Change	Time to confirm change in device rule base	
Test Security	Time to test effectiveness of change	Leverage tests from other steps.
Roll Back Change	Time to roll back to last known good configuration (if test fails)	This is why rule backups before changes are so important.
	Time to determine why change failed	
	Time to adjust/update rule change(s) and restart Deploy phase	Depending on the amount of troubleshooting, the change may be kicked back to the planning phase.

Audit/Validate

As a result of our Deploy step we have the rule change(s) implemented on the network security devices, but it's not over yet. To keep everything aboveboard (and add steps to the process) we need to include a final audit.

Basically this is about having either an external or internal resource *not* on the operations team validate the changes and make sure everything has been done according to policy. Yes, this type of stuff takes time, but not as much as an auditor spending days on end working through every change you made on all your devices because the documentation isn't there.

This process is pretty straightforward and can be broken down into 3 subprocesses:



1. **Validate Rule/Signature Change:** There is no real difference between this Validate step and the Confirm step in Deploy except the personnel performing each. This audit process provides separation of duties, which means someone other than an operations person must verify the change(s).
2. **Match Request to Change:** In order to close the loop, the assessor needs to match the request (documented in Process Change Request) with the actual change to ensure everything about the change was clean. This involves checking both the functionality and the approvals/authorizations through the entire process resulting in the change.
3. **Document:** The final step is to document all findings. This documentation should be stored separately from the policy and change management documentation to eliminate any chance of impropriety.

Overkill?

For smaller companies this step may be a non-starter. Small shops generally have the same individuals define policies and implement the rules associated with them. We do advocate documentation at all stages, even in this case, because it's critical for passing any kind of audit/assessment. Obviously for larger companies with a lot more moving pieces this kind of granular process and oversight of the changes can identify potential issues early — before they cause significant damage. The focus on documenting as much as possible is also instrumental for making the auditor go away quickly.

The data from our survey validates this. 18% of survey respondents do not perform an audit after a firewall change, and almost 26% don't audit their IDS/IPS changes.

Audit/Validate Metrics

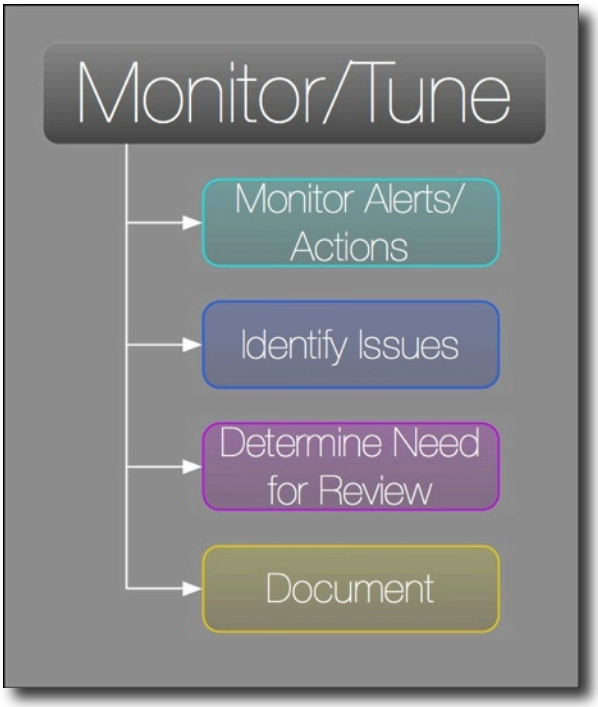
Process Step	Variable	Notes
Validate Change	Time to define specific firewall or IDS/IPS testing scenario	May be the same tests run during the internal test, but by different folks to maintain separation of duties.
	Time to confirm intended effect of the change(s)	
Match Request to Change	Time to confirm change was requested and properly authorized	Varies based on the automation of change management workflow and availability of documentation.
Document	Time to document successful change(s)	

Monitor Issues/Tune (IDS/IPS)

Now we wrap up the Manage process by talking about the need for tuning the new rules and/or signatures we set up. This step applies specifically to IDS/IPS (as opposed to firewalls) because of differences in how each tool detects attacks. The firewall looks for specific conditions, such as traffic over a certain port, or protocols characteristics, or applications performing certain functions inside or outside a specified time window. In contrast IDS/IPS looks for patterns, and pattern recognition requires a lot more trial and error. So it really is an art to write IDS/IPS rules that work as intended. The process is rather bumpy so a good deal of tuning is required once the changes are made. That's what this next step is all about.

As described, once we make a rule or action change/update on an IDS/IPS it's not necessarily instantly obvious whether it's working or not. You have to watch the alert logs for a while to make sure you aren't getting too many or too few hits on the new rule(s), and that conditions are correct when the alerts fire. That's why we've added a specific step for this *probationary period* of sorts for a new rule.

Since we are tracking activities that take time and burn resources, we have to factor in this tuning/monitoring step to get a useful model of what it costs to manage your IDS/IPS devices. We have identified four discrete subprocesses in this step:



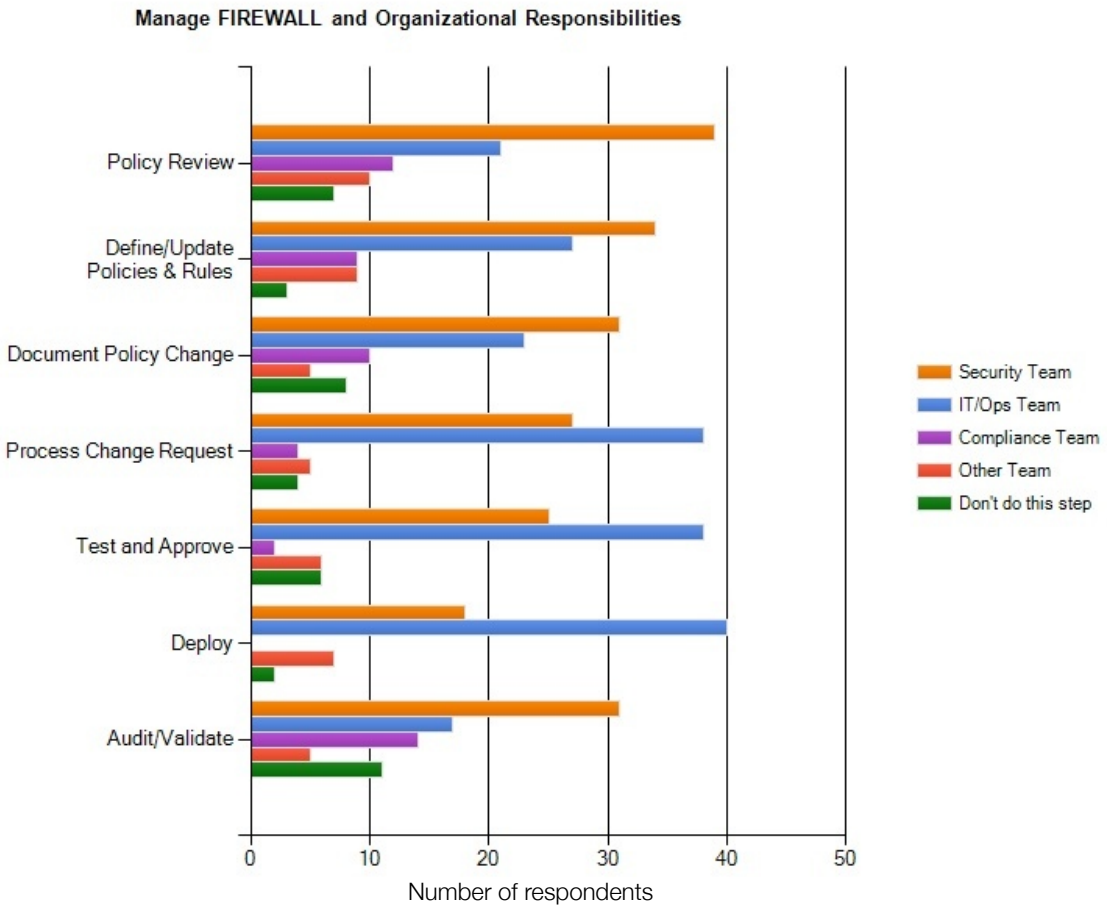
1. **Monitor IDS/IPS Alerts/Actions:** The event log is your friend, unless the rule change you just made causes a flood of events. So the first step after making a change is to figure out how often an alert fires. This is especially important because most organizations phase a rule change in via a "log only" action initially, which is the right thing to do. Until the rule is vetted, it doesn't make sense to put in an action to block traffic or blow away connections. How long you monitor the rule(s) varies, but within a day or two most ineffective rules can be identified and problems diagnosed.
2. **Identify Issues:** Once you have the data to figure out if the rule change isn't working, you can suggest changes to address the issue.
3. **Determine Need for Policy Review:** If it's a small change (threshold needs tuning, signature a bit off), it may not require a full policy review and another pass through the entire change management process. So it is important to be able to iterate quickly over minor changes to reduce the amount of time to tune and get the rules operational. This requires defining criteria for what requires a full policy review and what doesn't.
4. **Document:** This subprocess involves documenting the findings and packaging up either a policy review request or a set of minor changes for the operations team to tune the device.

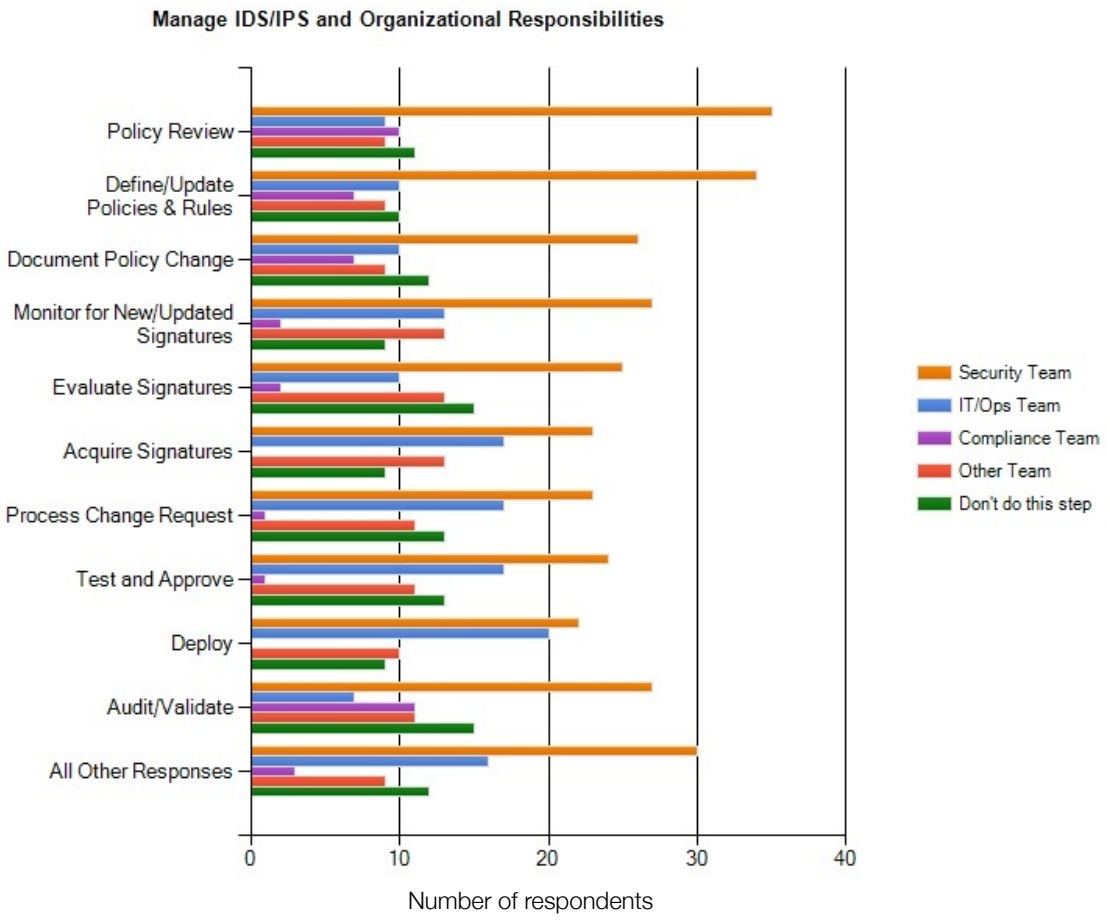
Monitor Issues/Tune Metrics

Process Step	Variable	Notes
Monitor IDS/IPS Alerts/Actions	Time to monitor IDS/IPS alerts (monitor step)	This function is typically performed within the Monitor IDS/IPS process. Refer to that part of the research project for more detail.
	Time to evaluate the effectiveness of actions	Are the right attacks being blocked? Are applications breaking? How many complaints related to the rule(s) are being fielded?
Identify Issues	Time to determine nature of issue and categorize (true vs. false positive; true. vs. false negative)	Effort varies based on number of alerts, number of missed attacks, impact of actions taken/not taken.
Determine Need for Policy Review	Time to evaluate whether issues require formal policy review	This is a subjective assessment of whether the issues warrant a full analysis and review of the policies and underlying rules.
Document	Time to document policy/rule problem	Regardless of whether a full policy review is undertaken, it's important to share feedback on the effectiveness of rules with the policy team.

Organizational Responsibilities for Manage

One other aspect of the survey data bears mentioning: the organizational responsibilities for each step in the process. In general, the data has already told us that the Manage Firewall process is far more mature than IDS/IPS management. As such, general IT ops tends to handle firewall changes, while the security team drives the Manage IDS/IPS process for the most part. You'll also see that there are a larger set of activities skipped for IDS/IPS, another indication of relative process immaturity.




Manage IDS/IPS and Organizational Responsibilities
Policy Review
Define/Update Policies & Rules
Document Policy Change
Monitor for New/Updated Signatures
Evaluate Signatures
Acquire Signatures
Process Change Request
Test and Approve
Deploy
Audit/Validate
All Other Responses
0
10
20
30
40
Number of respondents
Security Team
IT/Ops Team
Compliance Team
Other Team
Don't do this step

Device Health Process

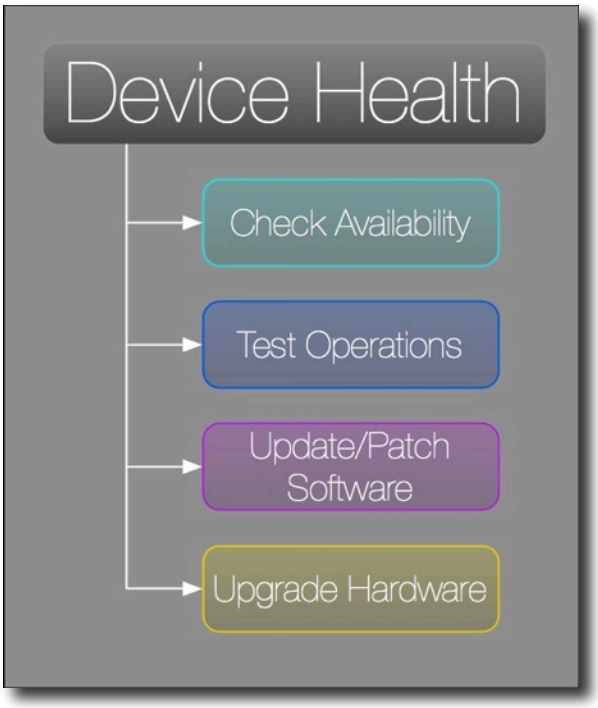
Now that we've decomposed both the Monitor and Manage processes in gory detail, we need to wrap things up by talking about maintaining device health. That means keeping the device up and running and doing its job. This involves stuff like up/down checks, patching, and upgrading hardware as necessary. All these functions take time and possibly tools, and if you are trying to really understand what it costs to monitor and manage your network security environment, they must be modeled and factored into your analysis.

In this step, you make sure the equipment is operational and working in peak form. Okay, perhaps not peak form, but definitely collecting data and/or blocking attacks. The consequences of downtime for any network security device are severe. To be clear, the Device Health process applies to the devices either collecting the data **or** under management. This is not focusing on device health for things like servers or other parts of the computing stack. For the Monitor process you need to ensure all collectors, aggregators, and analyzers are operating — as well as patched and upgraded to ensure reliability at scale. Likewise, for the Manage process you need to make sure the firewalls and/or IDS/IPS devices under management work as intended.

This process is pretty straightforward, so let's step through it:

1. **Availability Checking:** As with any other critical IT device, you need to check availability. Is it up? You likely have an IT management system to do this up/down analysis, but there are other low-cost and no-cost ways to check device availability (it's amazing what you can do with a scripting language...).
2. **Test Operations:** The next step is to make sure the data being collected is as expected and/or the devices are blocking attacks per the defined rules. You should have spot checks scheduled. This involves a sample set of collectors to ensure collection works as required for monitors. You'll also run specific attacks (using a scanner or pen test tool) to check the efficacy of the network security device. You defined test cases during each step of the process, which you can leverage on an ongoing basis to ensure the accuracy and integrity of collected data.

3. **Update/Patch Software:** These devices (whether monitors or network security devices) run some kind of software that runs on some kind of operating system. That operating system needs to be updated every so often to address security issues, software defects, and other issues. Obviously if the collector is a purpose-built device (appliance), you may not need to specifically patch the underlying OS. At times you'll also need to update the collection application, which is included in this step. We explored this involved process in Patch Management Quant, so we won't go into detail again here.



4. **Upgrade Hardware:** Many monitoring systems nowadays use hardware-based collectors and purpose-built appliances for data aggregation, storage, and analysis. Thanks to Moore's Law and the exponentially increasing amount of data we have to deal with, every couple years your monitoring devices hit a scalability wall. When this happens you need to research and procure new hardware, and then install it with minimum downtime for the monitoring system.

Outsourcing

We go into such gory detail on these processes because *someone* has to do the work. Most organizations don't factor health monitoring & maintenance into their analysis, which skews the cost model. One of the perceived advantages of using an outsourced monitoring or network security device management service is the service provider maintains the collectors and/or the devices. Thus you need to weigh the costs to do it yourself in order to get a fair comparison with outsourcing. Through the Network Security Operations Quant process we're attempting to provide the tools to understand the cost of monitoring and managing your environment. Regardless of whether you outsource or not, tracking these operational metrics over time will help you identify processes not working efficiently and allow you to focus on streamlining these operations and reducing cost. Having this kind of information will allow you to make a more informed decision on whether outsourcing is right for your organization.

Device Health Metrics

Process Step	Variable	Notes
Check Availability	Time to set up management console with alerts for up/down tracking	This can be handled through a central IT management system (for all devices) or individual element management systems for specific device classes.
	Time to monitor dashboards, investigate alerts, and analyze reports	
Test Security	Time to run vulnerability scans & pen test specific devices	We recommend you try to break your own stuff as often as practical. The bad guys try every day.
	Time to evaluate results and determine potential fixes	
	Time to prepare device change request(s)	Device changes require documentation for the ops team to make them.
Update/Patch Software	Time to research patches and software updates	See Patch Management Project Quant for granular details of the patching process.
	Time to download, install, and verify patches and software updates	
Upgrade Hardware	Time to research and procure hardware	
	Cost of new hardware	Yes, in fact, there are hard costs involved in managing network security (just not many of them). Shop well — the market is very competitive.
	Time to install/upgrade device	This depends on the number of devices to upgrade, the complexity of configuration, the presence of a management platform, and ability to provision devices.

Conclusion

Process Maturity, Metrics Immaturity

Now that we have concluded the research for NSO Quant, we believe it represents a comprehensive model for how organizations monitor their networks and manage their network security devices. But having a model is only the beginning of the journey. Most of the organizations responding to the NSO Quant survey reported fairly mature processes in place, especially for change management. But along with that perceived process maturity comes a general lack of precision as to how much time really is spent on the associated monitoring and management activities.

Service providers are not helping the matter either. Every single MSSP offering monitoring and device management services talks about improved efficiencies and economies of scale. But until now, there was no framework to evaluate these marketing claims with real-world evidence. Many organizations that end up engaging a service provider do so, not because of a specific fact-based cost model showing ROI, but instead as a leap of faith or due to a skills gap that pushes a customer toward a managed service.

Moreover, given the global pressure to increase efficiency in all aspects of operations, it's surprising that no one has turned the spotlight on IT and specifically security operations to figure out how to streamline these activities. Equipped with the NSO Quant model and a commitment to collect data over time, organizations can figure out where they are spending their time and make the processes more efficient, which will provide more time for *strategic* activities. A side benefit is having the data to provide fair comparisons between outsourcing and doing it yourself.

Where to Start?

We present many operational steps and associated metrics in this project. We recommend organizations start small. Attack one aspect of the process model at a time, achieve some early success, and then gradually expand the scope. It most likely makes the most sense to look at one of the more mature process models — perhaps managing firewalls or IDS/IPS. Or maybe monitoring servers, so long as the number of participants is reasonable and the amount of data gathered is manageable.

The steps to introduce this approach to your organization are pretty straightforward and very replicable.

1. Pick a place to start.
2. Map the process.
3. Choose the metrics.
4. Collect the data.
5. Analyze the data.
6. Adapt the process.

Then go back to Step 1, with another subset of your network security operational processes. We don't mean to oversimplify things, but it's not hard. Your organization just needs the commitment to systematically collect data and adapt the processes based on what the data tells you.

Finally, the authors of this report would like to encourage additional open, independent, community research and analysis projects in IT and security metrics. Utilizing a transparent research process enables new kinds of collaboration capable of producing unbiased results. We are investigating other opportunities to promote open research and analysis, particularly in the areas of metrics, frameworks, and benchmarks. If you have any suggestions as to additional research opportunities, feel free to drop us a line at info@securosis.com.

About the Analysts

Mike Rothman, Analyst/President

Mike's bold perspectives and irreverent style are invaluable as companies determine effective strategies to grapple with the dynamic security threatscape. Mike specializes in the sexy aspects of security, such as protecting networks and endpoints, security management, and compliance. Mike is one of the most sought-after speakers and commentators in the security business, and brings a deep background in information security. After 20 years in and around security, he's one of the guys who "knows where the bodies are buried" in the space.

Starting his career as a programmer and a networking consultant, Mike joined META Group in 1993 and spearheaded META's initial foray into information security research. Mike left META in 1998 to found SHYM Technology, a pioneer in the PKI software market, and then held executive roles at CipherTrust and TruSecure. After getting fed up with vendor life, Mike started Security Incite in 2006 to provide a voice of reason in an over-hyped yet underwhelming security industry. After taking a short detour as Senior VP, Strategy and CMO at elQnetworks, Mike joined Securosis with a rejuvenated cynicism about the state of security and what it takes to survive as a security professional.

Mike published *The Pragmatic CSO* <http://www.pragmaticcso.com/> in 2007 to introduce technically oriented security professionals to the nuances of what is required to be a senior security professional. He also possesses a very expensive engineering degree in Operations Research and Industrial Engineering from Cornell University. His folks are overjoyed that he uses literally zero percent of his education on a daily basis. He can be reached at mrothman (at) securosis (dot) com.

Rich Mogull, Analyst/CEO

Rich has twenty years of experience in information security, physical security, and risk management. He specializes in data security, application security, emerging security technologies, and security management. Prior to founding Securosis, Rich was a Research Vice President at Gartner on the security team. Prior to his seven years at Gartner, Rich worked as an independent consultant, web application developer, software development manager at the University of Colorado, and systems and network administrator. Rich is the Security Editor of *TidBITS*, a monthly columnist for *Dark Reading*, and a frequent contributor to publications ranging from *Information Security Magazine* to *Macworld*. He is a frequent industry speaker at events including the RSA Security Conference and DefCon, and has spoken on every continent except Antarctica (where he's happy to speak for free — assuming travel is covered).

Prior to his technology career, Rich also worked as a security director for major events such as football games and concerts. He was a bouncer at the age of 19, weighing about 135 lbs (wet). Rich has worked or volunteered as a paramedic, firefighter, and ski patroller at a major resort (on a snowboard); and spent over a decade with Rocky Mountain Rescue. He currently serves as a responder on a federal disaster medicine and terrorism response team, where he mostly drives a truck and lifts heavy objects. He has a black belt, but does not play golf. Rich can be reached at rmogull (at) securosis (dot) com.

About Securosis

Securosis, L.L.C. is an independent research and analysis firm dedicated to thought leadership, objectivity, and transparency. Our analysts have all held executive level positions and are dedicated to providing high-value, pragmatic advisory services.

Our services include:

- *Primary research publishing*: We currently release the vast majority of our research for free through our blog, and archive it in our Research Library. Most of these research documents can be sponsored for distribution on an annual basis. All published materials and presentations meet our strict objectivity requirements, and follow our Totally Transparent Research policy.
- *Research products and strategic advisory services for end users*: Securosis will be introducing a line of research products and inquiry-based subscription services designed to assist end user organizations in accelerating project and program success. Additional advisory projects are also available, including product selection assistance, technology and architecture strategy, education, security management evaluations, and risk assessment.
- *Retainer services for vendors*: Although we will accept briefings from anyone, some vendors opt for a tighter, ongoing relationship. We offer a number of flexible retainer packages. Example services available as part of a retainer package include market and product analysis and strategy, technology guidance, product evaluations, and merger and acquisition assessments. Even with paid clients, we maintain our strict objectivity and confidentiality requirements. More information on our retainer services (PDF) is available.
- *External speaking and editorial*: Securosis analysts frequently speak at industry events, give online presentations, and write and/or speak for a variety of publications and media.
- *Other expert services*: Securosis analysts are available for other services as well, including Strategic Advisory Days, Strategy Consulting engagements, and Investor Services. These services tend to be customized to meet a client's specific requirements.

Our clients range from stealth startups to some of the best known technology vendors and end users. They include large financial institutions, institutional investors, mid-sized enterprises, and major security vendors.

Additionally, Securosis partners with security testing labs to provide unique product evaluations that combine in-depth technical analysis with high-level product, architecture, and market analysis.