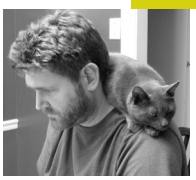
DAVID JOSEPHSEN

iVoyeur: comply



David Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and is Senior Systems Engineer at DBG, Inc., where he maintains a gaggle of geographically dispersed server farms. He won LISA '04's Best Paper award for his coauthored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project.

dave-usenix@skeptech.org

I MISS THE BYGONE DAYS WHEN everyone thought security was a product. You almost never had to deal with vendors beyond trying to convince management that openbsd, pf, and snort were the same no matter how pretty a box the vendor had found to put them on. When security was a product, all it took to placate panic-stricken knee-jerkers was a few tens of thousands of dollars on perimeter appliances, and, when they were placated, we were left alone, free to go back to the actual work of building secure infrastructure and maintaining secure systems.

Then Bruce had to go write that infernal book. You know the one I mean: the one that rhymes with "tree kits and pies." Now everybody thinks security is a process. Now the vendors never go away. Worse, in some sick pantomime of vigilance we pay them to hang around, poking us with their brain-dead scanners and generating nonsensical 50-page lists of prioritized nonvulnerabilities. Suddenly we're surrounded by standards. Now we have CoBIT, COSO, FFIEC, FISMA, GLBA, HIPAA, ISO17799/BS7799, NISPOM, PCI, and SOX to waste time on, incentivizing management to aspire to minimal baselines of system security and to postpone everything in the name of creating hundreds of pages of policies that no one (including the auditors) will ever read and that are hopelessly impossible to actually adhere to.

I'm terribly sorry, but I happen to be armpits-deep in PCI compliance, and I got a bit carried away there. I understand the intent of the policies and documentation (mostly). Lawyers are a wholly different and dangerous kind of script kiddie, and the documentation mitigates them. And of course I see what the community is trying to do in terms of third-party verification and base-line standardization. People can't know what security looks like unless there are some guidelines. That's all great and everything, except that it isn't working. Security is not and has never been a product or a process; it's a state of mind. At the end of the day, you "get it" or you don't, and that mostly boils down to who you have working for you and what you're trying to accomplish.

So while I see the point of the records-keeping aspects of the standards, I can't help feeling sometimes that the systems security aspects are better left to the market. Companies that want to do busi-

ness with each other should ensure that they meet each other's internal criteria for systems security and leave it at that. Those that know what they're doing will do well and those that don't won't. Whatever sickeningly large amount of money is currently going to the myriad horde of mediocre security vendors would be better spent on getting, building, maintaining, and keeping good people.

In the "security as a process" universe, what happens instead is that companies partner with each other using PCIDSS compliance as the criterion and people like you and me end up having to deal with business partners who don't know what it means to verify an ssh key fingerprint. In the "security as a process" universe, companies still buy security; it just costs a lot more, they never stop paying, and it's much more verbose. My biggest gripe with standardization in this context has to be that, in the minds of the executives, compliance equates to security. Meet the minimal requirements of the standard and you are—by definition—secure (and I think this idea is actively encouraged by the aforementioned mediocre security vendors). Here's a hint: If you think you can safely carry out transactions of sensitive data with me because I said "yes" when you asked, "Are you PCI compliant?" you have some security problems. They are severe in scope, trivial to exploit, and of a type that won't show up on the port scans.

There I go getting carried away again. It's been a bad week. Sorry. Anyway, I'm sure compliance is a challenge for pretty much everyone out there, and the company I currently work for is no exception. Being a rather small operation, we're challenged mostly by the documentation aspects of the standards, but we've also run into quite a few requirements that assume a much larger body of employees than we possess. In each of these cases, one of the various monitoring systems we currently employ has satisfied the requirement handily. This of course brings me (in only six paragraphs) to the actual point of this particular article: monitoring tools that can help you comply.

Most of the requirements I'm talking about are audit-related, and much of the advice I'm about to give is probably advice you've heard before about tools you've heard about before. It's all obvious stuff in little pieces, but when it all comes together it loses transparency, so my intent here is to provide a short list of things you should have implemented before going into an audit in order to ensure that as many requirements as possible are met in an automated fashion. The PCIDSS, for example, has a slew of requirements around making sure the policies are being followed by manually auditing various aspects of the environment. Most of these requirements are written in such a way as to suggest humans should be performing audits quarterly, but in my experience so far, they can all be met programmatically and still make the auditors happy—well, not happy, but satisfied.

Account Auditing

The security standards that I'm aware of all have in common some user-account-related auditing requirements. Among these are requirements for detecting old accounts and enforcing password strength policies. How you do this depends a lot on your environment and the size of your organization.

Most companies of any size use some sort of LDAP-like directory system for maintaining user IDs and passwords. The general idea here is to keep a list of valid accounts. If you're large, you probably also have something such as PeopleSoft or SAP. In this case you're looking to write an LDAP diff between your HR system and your directory system. Accounts that don't exist in the former probably shouldn't exist in the latter, with the exception of system

and administrator-related accounts. You'll want to enumerate the exceptions once and monitor changes from then on. Any monitoring system that can execute arbitrary scripts can fulfill this role.

If you're small like us, then you may be using local credentials. In this case a simple list of valid accounts to diff against is probably sufficient. Either way the strategy is pretty much the same. A Nagios plug-in could be used to check accounts on the servers against a list of valid accounts. A file system integrity checker such as Samhain [1] or a configuration management engine could be useful for larger organizations. The auditors are going to want to see how the valid account list is managed and whatever policies you have for incident response in the event a rogue account is encountered or an unexpected change is made.

Change Auditing

You will be required to display policies and procedures for making changes to production systems. These need to be backed up with a change-detection methodology of some type. For large installs configuration management engines are certainly the best way to go here. Unfortunately, I haven't taken that particular plunge myself as of yet, so I can't speak intelligently about it beyond observing that if LISA attendees are any indication, the "big three" appear to be (in alphabetical order) bcfg2, cfengine, and puppet.

If you're like me (writing articles telling other people what to do when you should be implementing configuration management), then you can meet the criteria with a filesystem integrity checker such as Samhain [1]. You'll also need something like RANCID [2] for your network gear, even if you do use a configuration management engine. For those of you who aren't familiar with it, RANCID is an ingenious bit of glue among CVS, expect, and SSH. It logs into your network gear, dumps the configuration, and maintains a revision history of the dumps with CVS for you automatically. You'll probably want to send change notifications from these systems to /bin/logger so they get sent to your centralized syslog server, which brings me to . . .

Incident Detection

The PCI DSS wants you to have a policy for collecting and auditing logs from systems and security appliances. It's not enough to have the logs; you need to show how you audit them and what you do when your log audits encounter something unexpected and possibly bad. You can meet these criteria with automated log parsing or event correlation software if you implement it well enough.

The first thing you're going to need to do is to get your security-related logs in a single place and in a single format. The auditors know that the log-watching software is only as good as the logs it has to watch, so they're going to want to see what information you have, how you're getting it there, and what's preventing it from being modified by a malicious entity. Centralized syslog infrastructure works well for this for most organizations. I'd suggest a syslog daemon that supports the newer (RFC 3195) reliable delivery mechanisms such as SDSC Syslog [3] if you're going to go the syslog route.

There are several implementations of the syslog protocol for Windows, including EventReporter [4] and Snare [5], if you're blessed with Windows machines. I can't think of any network appliances that don't have native support for syslog, but if you've managed to find one, there are several SNMP to syslog translators [6] available. Creating a centralized syslog architecture is

beyond the scope of this article, but a great place to start is Tina Bird's excellent log analysis portal [7].

Once you have it all in one spot there are several ways to parse it. Most people I know are fond of either logsurfer [8] or SEC [9] for this purpose. Both of these tools are fodder for future articles, and both do an excellent job of mining log data in real time for interesting events, and both will meet the audit criteria. I tend to mostly use logsurfer because it's written in C and therefore has a smaller footprint, saving SEC for one-off or more complex situations. The auditors will want to see what you're parsing for and what you're doing with the alerts. (Hint: This should be detailed in an incident response policy, and the policy should actually be followed.)

Another tool that's gained much popularity in the past few years is Splunk [10]. Having played with the open source version, I'm convinced this isn't wholly to do with the company's cool t-shirts. Splunk is, in fact, a fantastic tool. It won't be replacing grep, logsurfer, or SEC in my environment, but it certainly augments them, and since Splunk added support for taking arbitrary actions based on regex-style criteria, it certainly bears mentioning in this context. Meeting compliance requirements is actually a stated goal of the software, according to Splunk's Web site, so it seems the company had some experience in this regard.

For bonus points, consider implementing a nonaddressable monitoring system for forensics purposes. Logs cannot be modified by a malicious entity if they are on a box that cannot be accessed via the network (probably). Passive network taps work well here. I've personally had good experiences with NetOptics [11] aggregating network taps. One type of tap we use can listen to four 10/100 networks and aggregate them all to a single gigabit port. The box connected to that port need not actually be on any network at all.

System and Service Discovery

The auditors will want to know how you're managing network access, both via WiFi and via rj45 wall sockets. Even if you use a NAC system, they'll want you to audit the network for unauthorized systems as well as new services running on existing systems. Of course, whatever you use needs to be backed up with a policy document. PCI wants a firewall policy, for example, which details the services that are authorized for use on the network and written justification for services it considers insecure.

There are several Nagios plug-ins that can help here. Many of them are wrappers around nmap and some do their own scanning and system discovery. I've recently started using OSSIM [12] for this sort of stuff and I have to say I'm pretty happy with it so far. OSSIM is like a portal for all things security in your organization. It's one of the better tools I've come across for pooling output from existing tools and presenting it in a way that is actually pretty flexible. The best thing about it is that it seems to have silenced the auditors with its built-in support for scanning tools such as nmap, p0f, Pads, and Nessus.

Having spent the better part of two months in PCI-land, what have I implemented to make my workplace more secure? Well, nothing, actually. All of the systems security and monitoring currently in place was in place pre-audit, and I've spent 100% of my time churning out policies outlawing bittorrent in the requisite Microsoft Word format. If we are more secure as a result of that then I'm glad, but I kind of doubt that we are. Bruce himself once said, "Amateurs hack systems; professionals hack people," and I'm beginning to think I've been hacked by a gaggle of standards bodies. I'm not

sure what the answer to the problem of organizational security is, but this can't be it. What we have in the standards is nothing but a license for vendors to take our money and give us in return someone else's static definition of an utterly subjective concept.

I don't harbor any hope that ranting about it here has convinced anyone of anything, but I do hope you found a few tools in there that will help you with your particular set of auditors. If not, at least you know I feel your pain.

Take it easy.

REFERENCES

[1] http://www.la-samhna.de/samhain.

[2] http://www.shrubbery.net/rancid.

[3] http://sourceforge.net/projects/sdscsyslog.

[4] http://www.eventreporter.com.

[5] http://www.intersectalliance.com.

[6] http://snmptt.sourceforge.net.

[7] http://www.loganalysis.org.

[8] http://www.dfn-cert.de/eng/logsurf.

[9] http://simple-evcorr.sourceforge.net.

[10] http://www.splunk.com.

[11] http://www.netoptics.com.

[12] http://www.ossim.net.