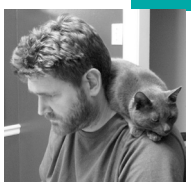
DAVE JOSEPHSEN

iVoyeur: breaking up is hard to do



Dave 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 co-authored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project.

dave-usenix@skeptech.org

ON MAY 6, 2009, A MESSAGE WITH THE subject line "Nagios is dead, long live Icinga" was posted to the nagios-devel list [1]. It briefly described a fork of Nagios core called Icinga and outlined the reasons why the developers thought a fork necessary. That alone was pretty earth-shattering news to the Nagios community at large, but as the thread developed, it became even more controversial; secret meetings, corporate conspiracies, deceit, betrayal, public hangings . . . well, all of that stuff except the public hangings. Anyway, I thought this month it would be fun to put on my journalist hat (otherwise known as my "prejudiced troll" hat) and relate the story thus far.

Nagios is a fantastic project, and a fork could affect a lot of folks, including a large percentage of the readership of this magazine. My first reaction to the idea of a fork was negative; it didn't seem as though good things could possibly come from it. When I saw who was behind the fork, several German developers from the Nagios Advisory Board as well as Netways.de (the guys who run nagiosexchange.com and organize arguably the largest annual Nagios conference), it became obvious that there were some legitimate gripes behind it. Although there are a fair number of personalities and motivations involved, I think those gripes are probably the best place to start. Let's take a look at Icinga's stated reasons for forking, the reaction of Ethan Galstad (the creator of Nagios) to them, and also some analysis of my own (because what would you do without my razor-sharp insight?).

Icinga's reasons, taken from their Web site and the thread mentioned in [1], are:

- Nagios development has stalled, because there is only one person with CVS commit access (namely, Ethan), and he's been inactive (meaning he's been ignoring bug fixes and new feature patches from the community, and further that he's not responding to email or list traffic).
- Nagios Enterprises, the commercial face of Ethan, has lately begun "harassing" developers about trademark infringement (meaning Ethan's lawyers are going around demanding devs and community members turn over domain names they've registered that have the name "Nagios" in them).
- NDOUtils (a database connectivity toolkit) is "still buggy."

- The Web interface is old and/or ugly (arguably a question of esthetics, but the general feeling for a number of years has been that the UI should be rewritten in PHP; it is currently a CGI written in C).

Although these seem like four unrelated issues at first glance, they actually break down into two groups of very closely related issues. The first group being the first two bullet points—that development has stalled because Ethan is a bottleneck, and that community members are being hassled by Nagios Enterprises lawyers. As evidenced by [2] and [3], the fork members felt that Ethan went AWOL after the 3.0 release. Simple bug fix patches were ignored (although a few critical bug fixes were, in fact, merged into core), Ethan was not providing a detailed road map forward, and new features they desperately wanted were nowhere near being merged into core.

Understanding that Ethan is a busy guy, several members of the Advisory Board got together during and after the 2008 Netways conference and created a patch queue system to streamline the process [4]. Ethan seemed to ignore the system, as well as attempts on the part of the community to open up the development process. When the trademark infringement notices started coming in, it was particularly frustrating to many members of the community. Their concerns went unheard while the gift of their labor and their years of enthusiastic evangelizing were being met with cease-and-desist letters.

For his part, Ethan responded that he had been forced away from development work by some legal trouble [5]. Without providing any real detail, he did mention Netways.de as the source of the problem. Although Ethan doesn't tell us this, what apparently happened was Netways.de (a German corporation) registered the name "Nagios" as a trademark in Germany [6], and Ethan was forced to sue them to get his name back. His attentions thusly diverted, Nagios core began to rot. Given this context, we gain some insight into the lack of attention to project work on his part, as well as the trademark authoritarianism. Given the legal entanglement, when Ethan was blindsided by a fork, his off-the-cuff reaction was that it was an attempt by Netways.de to undermine his credibility with the community.

From Ethan's perspective, Nagios had had a falling-out with Netways.de, so Netways.de was attempting to fork Nagios and pirate the community away to a project they controlled. If they couldn't steal Nagios in the courtroom, then they would attempt a PR coup and rename the project Icinga. Netways.de has explicitly denied this accusation [7], but the fact that the Icinga project lead is also the CTO of Netways.de lends some strength to this perspective. Netways.de's case is also not helped by the fact that the fork was planned and orchestrated in secret and without any attempt to notify or warn Ethan.

Tinfoil hats off and drama aside, I think I share the opinion of most of the community and tech press out there when I say that Nagios did not have a scalable development model, and that's a bad thing. If a project the size of Nagios can be halted by diverting one person's attention, that's not just a gaping procedural problem but also a security bug. Netways.de didn't just cause Ethan legal headaches, they DoS'd his development model (whether they meant to or not is another interesting question). Clearly Nagios needs a few more lead devs with commit access.

If Icinga has done anything, it's opened Ethan's eyes to this fact, and he claims to be taking steps to rectify that situation [8]. Other devs in the community have confirmed that he's approaching them about lead dev roles and commit access [6]. He's also vowing to correct his admitted communication deficiencies [8], and he's written numerous blog posts to clarify his position on the various topics surrounding the fork.

I've never personally had any contact with Ethan (having written the Prentice Hall book on Nagios, I can personally attest to the fact that he's a hard guy to get ahold of—even my publisher couldn't do it), but one does get a sense of a person from reading their code, and the sense I get from the work I've done in the Nagios Core source is that Ethan is a bright and meticulous craftsman. I don't think it's going to be easy for him to find people he feels have the right combination of skill and common sense to give them commit access. Hopefully, by the time you're reading this he'll have found at least two or three.

The second set of bullet points, that NDOUtils is buggy and that the UI is old, ugly and/or poorly designed, are legitimate in that they are undeniably factually correct. NDOUtils is in fact buggy, and although I don't personally have a problem with CGIs written in C, I assume most people under 40 would concede that, for better or worse, that's just not how things are done anymore.

Ethan, for his part, has not commented on these points, but other developers in the community have [9]. The thing is, there are quite a few deficiencies in Nagios Core that might cause one to consider forking, and although the UI might be up there for some people, NDOUtils probably wouldn't be, so I think it's a little disorienting for most Nagios devs to see these two issues so high on Icinga's list. From the Icinga devs perspective, however, these two issues are so closely related they're actually interdependent. The relationship between NDOUtils and the UI is subtle for the rest of us, because it revolves around a technical assumption being made by Icinga's UI design team. It's an assumption that I don't think is necessary, but I do see that it's being made and why, so I'll attempt an explanation.

The Icinga UI devs have a problem typical for anyone writing Nagios tools that deal with state data: namely, there is no trivial way to ask Nagios for the current state of a given service or host (now *there's* something one would expect to see in a list of "reasons to fork Nagios core"). So the Icinga devs want to export state data to an interface other than the built-in Nagios UI (because in this case they're re-implementing that UI). As it turns out, this is a problem these particular devs have run into before, because some of them wrote NagViz (a visualization add-on for Nagios) during which they solved this problem by exporting state data to MySQL using NDOUtils. So the Icinga UI devs are using a bit of constructive laziness and making the underlying assumption that NDOUtils is going to be a core component of the new PHP-based UI they're creating.

There's a lot I'm tempted to say about this line of thinking, but the most topical point is that the design they've chosen to pursue could be implemented in Nagios without any changes in core whatsoever. Indeed, the current version of Icinga appears to be Nagios with a patched NDOUtils add-on and a stand-alone PHP application reading state data from a database. The Icinga guys don't need a fork to make this UI thing happen, and trying to fight for mind-share for a new UI would probably be a great deal easier than fighting for mind-share for an entire fork. So why fork to glean functionality that you already have? Why not just fork the NDOUtils add-on?

Another point of interest is that NDOUtils is not what I think most Nagios admins would consider the "right answer" for this job. It's fine for people who want to write add-ons that need access to the daemon's state data, but it's not what I'd call the optimal interface to build a replacement UI around. The Icinga guys are beyond writing add-ons here. If I were them, I'd be looking to write my own NEB module, one that was lightweight and specialized to my purpose, and I'd probably avoid using an external database tier at all. Why sync state data to an external DB, and then sync the UI to the DB?

Instead, why not just sync the UI directly to the daemon using a specialized message-passing NEB module, or even something like op5's Merlin [10]? Having written an NEB module or two [11], I can assure you this sort of thing is completely within the current operational capabilities of Nagios Core, no forking required.

But since they are forking core, they could take it one step further and fix the actual problem inside the daemon. The more I think about it, the more puzzling it is to me that the Icinga devs have decided on the direction they have. Given that they're forking Nagios core, and given that they are a smart bunch of guys with plenty of experience hacking Nagios, this dependency chain they're creating between a database add-on and the core UI just doesn't make sense. I would hope that if one had the opportunity to rewrite Nagios one would consider functional improvements first and eye candy later. State data is hard to export, so write an API that more directly exports it, or, if databases are what you want, then internally replace the state data file with a database. Writing the sexy UI first and then patching an already kludgy add-on to provide it data smacks of a dangerous and extravagant naiveté. It screams "kewlist GUI wins!" and it's that kind of thinking that would make me think twice about handing out CVS commit access too.

As you might have guessed by now, I'll be sticking with Nagios, but I'll also be keeping an eye on Icinga. Hopefully, something interesting will develop from this fork, but my real hope is that Ethan fixes the bottlenecks and communication problems, and Nagios and Icinga can get past their differences and merge back into a single project again. The legal complications of their parent companies makes that unlikely in the foreseeable future, but I think it's a net loss for the community to have to split its effort between these two projects for any length of time. It also makes tenuous the positions of the myriad companies out there who are providing commercial products and services based around Nagios. These companies, such as Groundwork [12] and op5 [13], are currently where much of the ground-breaking work on Nagios is being done. The community loses if those companies have to switch gears and worry about supporting a fork. Time will tell.

Take it easy.

REFERENCES

[1] http://thread.gmane.org/gmane.network.nagios.devel/6246.

[2] http://article.gmane.org/gmane.network.nagios.devel/6267.

[3] http://article.gmane.org/gmane.network.nagios.devel/6270.

[4] http://www.mail-archive.com/nagios-users@lists.sourceforge.net/msg21839.html.

[5] http://article.gmane.org/gmane.network.nagios.devel/6253.

[6] http://blogs.op5.org/blog4.php/2009/05/07/the-future-of-nagios.

[7] http://markmail.org/message/yba2p2p3w7aq7p4v.

[8] http://community.nagios.org/2009/05/11/the-future-of-nagios-where-do-we-go-from-here/.

[9] http://article.gmane.org/gmane.network.nagios.devel/6273.

[10] https://wiki.op5.org/merlin:start.

[11] http://www.usenix.org/publications/login/2008-12/pdfs/josephsen.pdf.

[12] http://www.op5.com.

[13] http://www.groundworkopensource.com.