

# Re: [fw-wiz] Security policy language

---

*Source:* <http://www.derkeiler.com/Mailing-Lists/Firewall-Wizards/2007-01/msg00033.html>

---

- *From:* "Stephen P. Berry" <[spb@xxxxxxxxxxxxxxxx](mailto:spb@xxxxxxxxxxxxxxxx)>
  - *Date:* Wed, 24 Jan 2007 12:43:45 -0800
- 

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA1

We would like to implement/adopt a high-level specification language for the definition of a security policy, something that should let to specify the policy at organizational level. Such a policy should then be translated into specific fw rules. I'm puzzled because it's not a new problem, but I can't find good references. Several standards, especially in the XML-Web Services area, have been proposed by W3C, OASIS etc., to define security policies, but to me they seem quite useless in our case since I can't see how and why Web Services should be integrated in this context.

Just about any standard expressed in XML isn't going to help you because about ten times out of ten you'll discover that it is just a descriptive taxonomy, not an actual grammar. And if I understand what you're trying to do, you need a grammar.

Put in slightly different terms, most existing quote standards unquote do very little apart from defining a list of terms that can be applied to specific kinds of things---particular characteristics of a packet or a log line or whatever (i.e., a packet filter or regex), a vulnerability announced on bugtraq or wherever (i.e., a CVE number), and so forth. Being able to apply consistent labels to things like this is a Good Thing (so I'm not try to disparage it, although it's sure gonna sound like it in a second here), but it's still just the equivalent of Og the caveman pointing at objects and grunting `tree', `rock', or whatever. If you're Og, you can't use this system to explain to someone how to build a fire, how far you should be from your cave before you start a fire, or what to do to avoid accidentally burn down the forest when you're doing it.

In other words, you can't actually enunciate a procedure using just a descriptive taxonomy; you can't define a policy using one; and you can't express contingencies, confidences, counterindicators, and so forth with one.

I've actually spent a fair amount of time thinking about/coding around this general class of problem---not specifically concerned with producing as output specific firewall rules, but rather the more general case of attempting to formally enunciate one or more risk assessments, one or more threat models, and then evaluate incoming raw data (syslog output, captured packets, and so forth) in terms of them.

My observation is that most people approach the problem by trying to develop something that looks sorta like perl---an elaborate procedural language in which the details of the analyst-level decisions (the risk assessment and so forth) are enunciated as hard-coded logic as part of the design of the system itself. I.e., in designing your solution, what you're effectively doing is coding up all the special cases you can think of, and you end up with something like a library call to handle each of them.

The problem I see with this approach is, well, the standard litany of problems you could enumerate with any security product of any stripe: it's tryin to hit a moving target; it is only really suited for solving a narrow class of problems (whatever the developer---or the developer's customers---were worried about at the moment); it leads you to trying to state (and then solve) your problems in terms of what you -can- do instead of what you -want- to do; and so forth. I'm not really trying to make a case here; I assume most people interested enough to think about the problem (or read this far) know the sort of things I'm alluding to here.

So what's the alternative? Well, I'll start with a disclaimer: I can't give you working code to do what I'm talking about right now, today, and that should make you immediately suspicious. The fact that I don't have working code to demonstrate it works leaves open the possibility that what I'm about to describe is yet another security pipe dream involving an elaborate system that will never see the light of day and therefore constitutes a net contribution to the world of security of exactly zero. Dwell on this point. It's probably the most important thing I'll say in this message.

That said, the approach I've been working on is using the formal specification to allow enunciation of the actual grammar, rather than being the grammar itself. If that didn't mean anything to you, I'll go back to my earlier example: most existing efforts in effect give you an interpreter (the moral equivalent of perl, for example) and expect you to use (or write) scripts that will read and manipulate input data (i.e., packets) to do whatever you want to do (generate an alert as the canonical example). My idea is to treat all the raw data (packets, log lines, and so forth) as a raw data stream which is merely tokenised by things like filters---look at the process of filtering a packet and saying it matches filter `foo' as lexical analysis. The output of this process is a token stream. We then use our formal specification to enunciate the rules by which we do semantic analysis of the token stream. So think of this as being more like what lex/flex

Re: [fw-wiz] Security policy language

(the packet filter or your log regexen) and yacc/bison (the formal specification we're interested in developing) do than what perl or another script interpreter does.

Does this make sense? I'll hasten to point out that this isn't exactly earth-shattering stuff. All we've really done here is slightly re-define the problem. The point of doing this is that in the `traditional' model, the interesting bits of what we're trying to accomplish (the rules/logic involving the risk analysis or whatever) are effectively buried in the process---they're just canned routines sitting on a leaf node of our analysis tree. This means it will be difficult to manipulate/change/integrate/whatever them with respect to the rest of the rules.

By forcing your rules to be, explicitly, statements of semantic relationships you're `centering' the expressiveness of your language in the stuff you're actually interested in. If you're trying to talk about something like a risk analysis, you're not going to be able to just rattle off a list of things that you should or should not be present (although such a list may certainly be -part- of such an analysis); you're going to want to express contingencies and conditionals and so forth. With a `traditional' system, you end up burying this as canned procedure; with the sort of think I'm suggesting the rule that you'd create -is- the expression of the contingency or whatever.

Again: this isn't intended to be revelatory or anything like that. All I'm describing is really just a shell-game in which we shuffle around the problem description to allow us to gain expressiveness in the areas we need it. I happen to think this is a Big Win, but only in the sense that any notational/data model/whatever change is---it can make it easier for us to do certain things, and it can lead us to think about problems in different terms, but it (in and of itself) doesn't acutally -do- anything.

So going back to your original query, I suppose my advice is that you should think about think about developing a grammar specification---something in which the analyst can enunciate a grammar which describes his network/policy/whatever, rather than trying to develop the grammar itself.

I'm not sure how interested the list is in this sort of stuff, but I'd be happy to go on at tiresome length about the subject off-list if you're interested.

--spb

-----BEGIN PGP SIGNATURE-----

Version: GnuPG v1.4.6 (OpenBSD)

iD8DBQFFt8TVP32VcPQQS7wRAnYTAJ9+zt4LS/hidoEHT/Q2mmZwM9zxGQCfTqwZ

Re: [fw-wiz] Security policy language

Re: [fw-wiz] Security policy language

bfWl6HU407aY3ximFijsRbY=  
=XOXF  
-----END PGP SIGNATURE-----

---

firewall-wizards mailing list  
firewall-wizards@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
<https://listserv.icsalabs.com/mailman/listinfo/firewall-wizards>