Building Useful Security Infrastructure For Free

Brad Lhotsky <brad.lhotsky@gmail.com>
National Institutes on Health, National Institute on Aging, Intramural Research Program

Tags: security, Perl, database, open source, syslog-ng, postgresql, FISMA

Introduction

Working as a Security Engineer for a research program in the Federal government is a lot of fun, but incredibly challenging. Research, rightfully, receives the lion's share of funding, leaving very little for support services like IT and no funding for security specific activities. However, the burden of designing, implementing, analyzing, and reporting compliance to weighty government IT Security mandates like FISMA falls squarely on the IT section.

Our IT staff is less than 10 people. We provide Help Desk, Linux server administration, networking (switches, IDS, firewalls, NMS), SQL Databases, SMB file shares, programming support, training, and implement in-house applications for scientific research mostly in Perl and PHP for our institute of 700-900 users. We are also responsible for reporting compliance with Federal, Institutional, and Divisional mandates to our oversight.

In order to achieve all of this with a small staff, we've designed and implemented a lot of automation based on Open Source Software. We've learned how to leverage these tools to meet the needs of our institute and the requirements of those above us.

As a pragmatic group with very little free time, we focus on building security tools that provide daily operational value. We simply do not have the resources to implement controls for the sake of the controls themselves.

The More You Know

Most IT Security controls focus on first understanding your systems. A system in this sense is defined as the computers, people, and networks that work together to perform a task. In order to begin classifying, we need to know what we have and where. The first step was to rollout a comprehensive centralized logging infrastructure for our UNIX and Windows servers.

We chose to use syslog-ng as a basis for our centralized logging platform for one incredibly useful feature:

```
      destination d_subscriptions {
            program("/usr/local/eris/bin/syslog-ng-service.pl");
      };
```

This feature starts the program specified keeping a handle of that program's STDIN open to dispatch messages to based on the "`log {}`" definitions specified in the configuration file. This removes the startup overhead from the called program allowing the use of programs written in dynamic scripting languages which incur enormous startup penalties. It also ensures that the program end point is available while syslog-ng is running, meaning there's no additional program supervision necessary.

In order to facilitate rapid development of syslog based event correlators, we developed this program to convert the incoming syslog stream into a TCP based service that a script can connect to and "subscribe" to feeds of interest. For instance, the inventory application subscribes to dhcpd, MSWinEventLog, sshd, arpwatch, and smbd. This keeps the database size smaller and focussed on the events that prove most useful to operations.

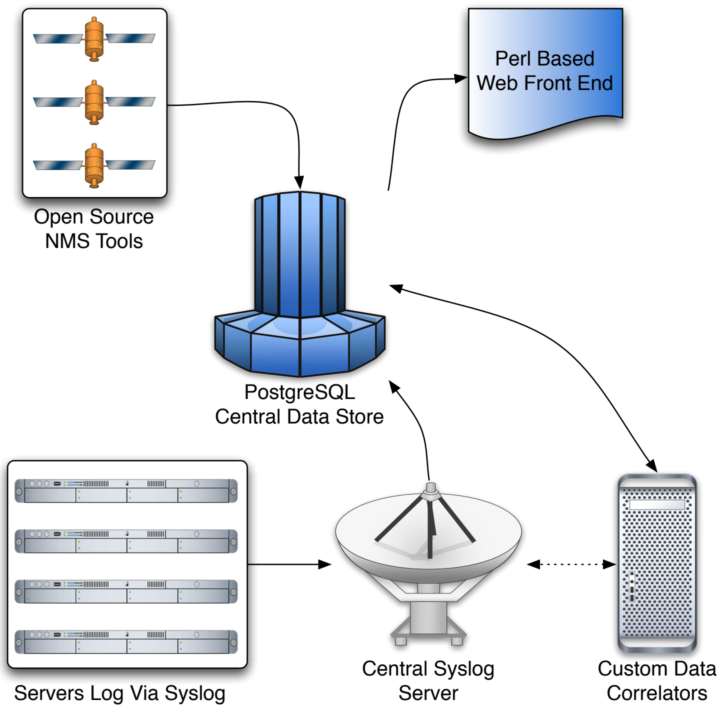
A Safe Place to Keep Our Data

In order to facilitate strange and novel concepts in correlation, it was clear that a Relational Database would be awesome as a storage engine to allow indexing and searching of the data we received. A PostgreSQL database server was setup and configured to allow data storage and retrieval. The reasons for this are numerous, but at the time of initial development it was the only Open Source database with views, stored procedures, triggers, and a slew of particularly relevant native data types including network types for IP addresses, networks, and MAC addresses. PostgreSQL has continued to make dramatic improvements to performance and usability since that time, and continues to be a leading Open Source RDBMS with unparalleled features and reliability.

PostgreSQL's PL/PgSQL language extension which was designed to be as close to Oracle's PL/SQL provides the option to do data correlation and validation in the database through the use of stored procedures and triggers. This facilitates rapid development of scripts placing data in the database as the "business logic" can be implemented at the data storage level. There is a performance penalty for doing this, but it allows for correlation to occur automatically as new datasources are added.

The setup we've designed is represented via the diagram on the following page.

Conceptual Overview of the System



Connecting the Dots

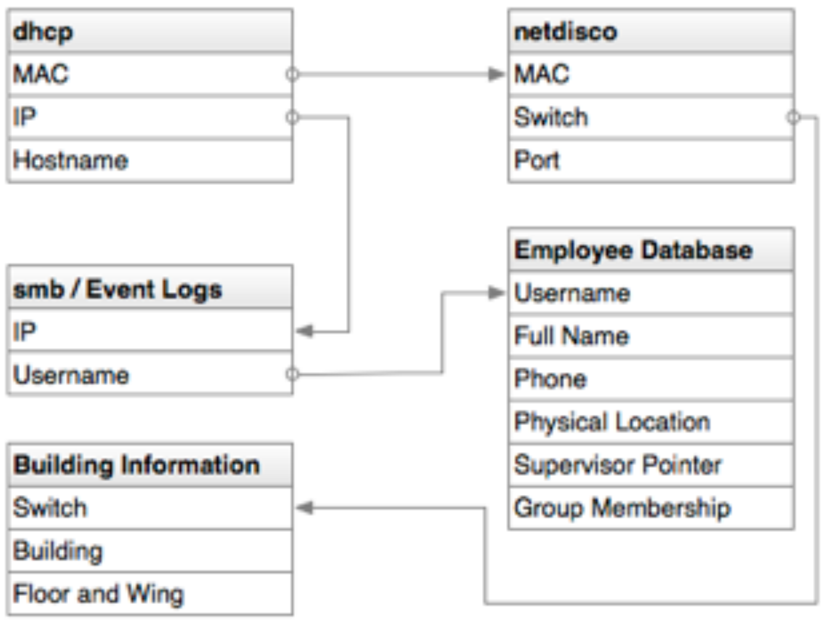
DHCP logs serve as a primary jumping off point for data correlation. We can simply store MAC, IP, and hostname attributes in a table and use them for lookups. We chose Netdisco as an Open Source layer 2 network management system that could be deployed to PostgreSQL. With a few triggers added to the Netdisco system, we can correlate MAC addresses to switch and port which allow our staff to quickly establish building, floor, and wing for any IP address on our network.

Using Samba and MS Windows Event Logs, we were able to discover the ActiveDirectory account name logged in to any client system. This allows simple IP to username correlation for things like IDS, but more importantly username to IP matching so our Help Desk don't have to walk every user through the "Start -> Run -> cmd -> ipconfig" routine, saving a few minutes with each call to the Help Desk.

Years prior to this system, our staff was asked to supplement the existing Enterprise Directory Service which was developed for all of NIH, with a number of specific enhancements specific to our Intramural Research Program. One of the features required every user to be assigned a "supervisor" attribute that links to another person object. Each person object contains full name, email, AD account, phone number, building, room, lab, and a pointer to the supervisor person object. This data was imported and synced to the PostgreSQL database as it would prove useful.

Basic Inventory Information

So with a carefully designed PG database and a few hundred lines of Perl code, we've managed to establish the following relationships:



What this means is:

- any event on the network containing a MAC address, IP address, or username can be correlated to any of the others
- and can also be correlated back to the metadata on the username and location

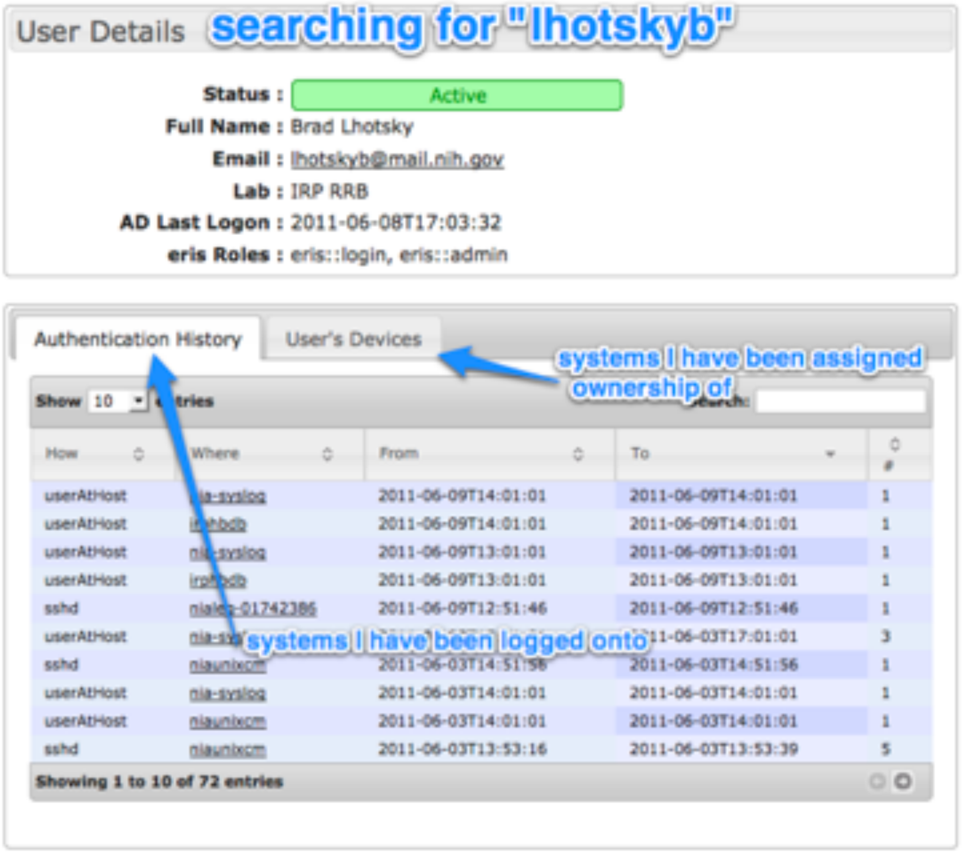
This allows classification of events in terms of business structure or geographic location all from data already available.

The tables which implement the storage keep track of dates and times inventory events such as DHCP, login, and ARP discovery occur. Since we're siphoning this data from the network servers and switches, we're not relying on high level or complicated protocol like IBM Tivoli End Point Manager to discover the devices. When a device is plugged into the network, we are able to immediately see it's been connected and record the date, time, and location of the event.

Now, when we need to report on the number of Apple computers, we can loop through the MAC Addresses seen in the past 6 months, look up the manufacturer data from the OUI Database, and report more accurately the number of active Apple Computers on our network.

Creating Useful Security

By wrapping this in a searchable web application, the Help Desk staff can now be far more efficient on each call.


searching for "lhotskyb"
systems I have been assigned ownership of
systems I have been logged onto
Authentication History
User's Devices

User Details

Status : Active
Full Name : Brad Lhotsky
Email : lhotskyb@mail.nih.gov
Lab : IRP RRB
AD Last Logon : 2011-06-08T17:03:32
eris Roles : eris::login, eris::admin

Show 10 **entries** **Search:**

How	Where	From	To	#
userAtHost	nia-syslog	2011-06-09T14:01:01	2011-06-09T14:01:01	1
userAtHost	irphbdb	2011-06-09T14:01:01	2011-06-09T14:01:01	1
userAtHost	nia-syslog	2011-06-09T13:01:01	2011-06-09T13:01:01	1
userAtHost	irphbdb	2011-06-09T13:01:01	2011-06-09T13:01:01	1
sshd	nialex-01742386	2011-06-09T12:51:46	2011-06-09T12:51:46	1
userAtHost	nia-syslog	[illegible]	2011-06-03T17:01:01	3
sshd	niaunixcm	2011-06-03T14:51:56	2011-06-03T14:51:56	1
userAtHost	nia-syslog	2011-06-03T14:01:01	2011-06-03T14:01:01	1
userAtHost	niaunixcm	2011-06-03T14:01:01	2011-06-03T14:01:01	1
sshd	niaunixcm	2011-06-03T13:53:16	2011-06-03T13:53:39	5

Showing 1 to 10 of 72 entries

This is excellent, but the real benefits of this system begin to show up as we integrated it with more Open Source security products.

Intrusion Detection and Correlation

We chose Snort as our Open Source IDS. Snort is free, fast, and stable. It does take a considerable amount of setup and configuration, but any signature based IDS solution will require that overhead. After being configured to listen through a network tap to a bonded interface on a CentOS box, we end up getting alerts that look like this:

```
Jun  2 12:10:55 myids snort[2908]: [1:2012647:2] ET POLICY Dropbox.com
Offsite File Backup in Use [Classification: Potential Corporate Privacy
Violation] [Priority: 1] {TCP} 137.x.x.x:1211 -> 199.47.216.144:80
```

This output is familiar to security professionals. We do have an IP address, which we have already established can be attributed back to a username, and a username back to their the organizational unit. This alert is parsed by the system and the relevant data stripped off and classified. An alert like this is classified in our system as “Potential Data Loss Event.”

We can then generate reports based on organizational unit and event classification which can tell us interesting things about a lab or section that we may not really want to know!

Configuration Management or DevOps for Compliance

All you base are belong to ..

Puppet, or any other configuration management engine that suits your needs will be sufficient. Spend some time evaluating the various configuration management engines and choose the one that best suits your organization. I cannot say enough good things about a good CM system that fits your organization.

We deployed Puppet and then put everything into our Version Control System (VCS) with commit hooks to automatically deploy new tagged release to the PuppetMaster. Using Puppet’s Domain Specific Language (DSL), I was able to convince our small staff of old school developers to embrace VCS after I built and demonstrated this:

```
subversion::deploy { ‘project_name’:
      svnurl => ‘svn+ssh://svn-readonly/repos/section/projectname’,
      target => ‘/opt/local/project_name’,
      notify => Service[‘httpd’]
}
```

Which requires only a Subversion project directory with a trunk/ and tags/ subdirectory. A bash script for tagging releases is distributed to /usr/local/bin/svntag which makes creating incremental release tags as easy as typing “svntag.” Puppet will then use $target/RELEASE to maintain the release number that’s been deployed to that target and anytime a new release is tagged, it will be automatically deployed at that location. Additions for allowing hostname-based configuration files was incorporated into a macro based off this:

```
webapp::deploy { ‘name’: project => ‘name’, config => ‘name.yml’ }
```

This expands to doing much the same as the previous example including the httpd restart, but also deploys the application configuration after the checkout completes, overwriting the development configurations that are stored in the subversion repository.

We run RedHat based distributions (CentOS,Fedora, and now Scientific Linux). Puppet was a great start, but Cobbler has solidified our CM platform. Cobbler is a KickStart based systems build platform that utilizes PXE to automate installs of RedHat based distributions. There is some work in process to extend it’s functionality to Debian systems. One reason to choose a build system like Cobbler, instead of an imaging solution like Ghost, is deployment to a hodgepodge of hand-me-down hardware that we maintain in a Research program.

When Cobbler performs an install, it's configured to include Puppet in the build, setting it to run at first boot. Using Puppet and Cobbler to rebuild my IDS sensor when I had to replace the hard drive took 37 minutes from PXE boot to up and running with Snort and syslog-ng for centralized logging.

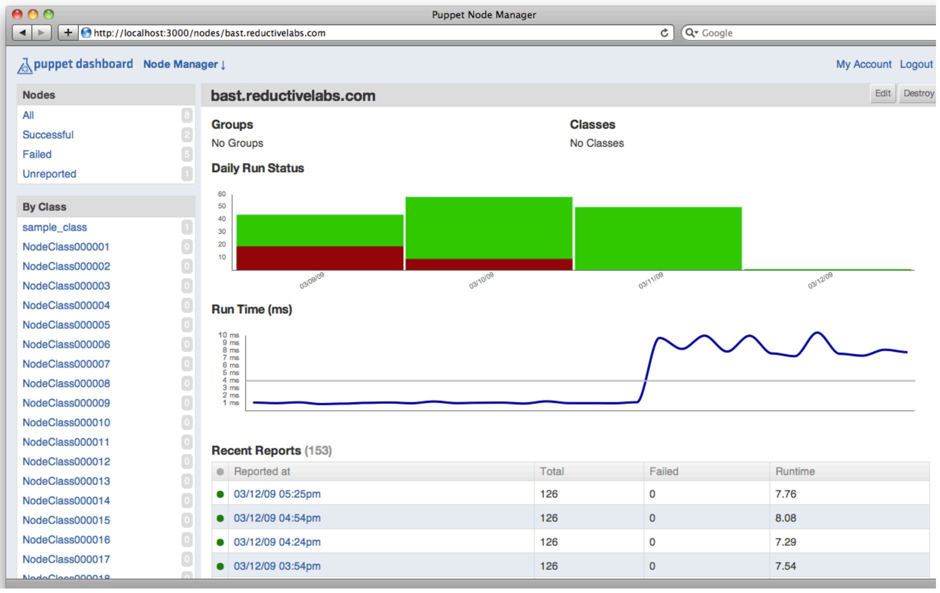
And what exactly does this have to do with Security?

A lot. Using a configuration management suite provides countless security benefits. First, it is the ultimate tool for guaranteeing consistent configuration across your network. It also offers the most benefit when each system is configured as much as possible by the CM. This allows a system administrator to PXE boot a new piece of hardware to replace an existing server and have the box configured identically in under an hour.

What we also get is a free inventory of all our servers. Since, as logical people, we tend to name classes and definitions something meaningful, we can leverage the CM tool to report on system functions and logical groupings. Puppet stores it's catalogs and states in simple YAML files which are parsed quickly and efficiently by your language of choice.

Configuration Managements, System Inventories, Software Inventories, are all provided. It's even possible to view the state of the compliance with the catalogs using the Puppet Dashboard. There are happy green and stressed out red lights for your auditors to admire!

Sample Puppet Dashboard



Nodes	
All	8
Successful	2
Failed	5
Unreported	1

By Class	
sample_class	1
NodeClass000001	0
NodeClass000002	0
NodeClass000003	0
NodeClass000004	0
NodeClass000005	0
NodeClass000006	0
NodeClass000007	0
NodeClass000008	0
NodeClass000009	0
NodeClass000010	0
NodeClass000011	0
NodeClass000012	0
NodeClass000013	0
NodeClass000014	0
NodeClass000015	0
NodeClass000016	0
NodeClass000017	0

bast.reductivelabs.com

Groups
No Groups

Classes
No Classes

Daily Run Status

Run Time (ms)

Recent Reports (153)

●	Reported at	Total	Failed	Runtime
●	03/12/09 05:25pm	126	0	7.76
●	03/12/09 04:54pm	126	0	8.08
●	03/12/09 04:24pm	126	0	7.29
●	03/12/09 03:54pm	126	0	7.54

Extending the Functionality

After the collection and storage of all this data to a relational database, we've found it indispensable to solving problems on a day to day basis. From simple one-off scripts to determine the number of Apple computers on the network, to more extensive systems that were trivial to implement on the back of the data we've collected. Consider our researchers requirement to utilize Skype to collaborate with international colleagues at no cost. The Federal Government prohibits the use of Skype, unless there are adequate compensating controls in place.

Using the data we've collected, we developed an automated tracking of Skype users. To receive a waiver from the Departmental Policy we were required to Skype users at our Institute affirm a monthly "Rules of Behavior" (RoB) update. The process of discovery and tracking of Skype usage looks something like this:

1. IDS signatures classified as "Skype" are correlated to Usernames using the database.
2. The usernames are checked against a table of "Accepted RoB's" and compared.
    a. If this is the first event, ie, no rows in the RoB table, the user is emailed the RoB and must click a link, sign in, and agree to the terms
    b. Otherwise, the "last detected skype usage" timestamp is updated
3. If at the time of detection, it has been a month or more since the acceptance of the RoB, the user is again emailed the link, asked to sign in, and agree to the terms of the RoB.
4. Everyday a list of users required to accept the RoB is compounded and emailed to the Administrators with their status included.
    a. The administrators receive the Phone number, Building/Room information, and the Lab Manager details for each user, aiding in persuading the user to accept the RoB.

This system is mostly automated, except for the occasional phone calls to the users requesting that they agree to the RoB terms. Other potential solutions to this problem exist, but often require complex proxy configurations that break if the user takes their laptop offsite, or manual exceptions by statically assigned IP addresses and manual tracking of RoB Acceptance. Our solution saves time, energy, and resources. It is only minimally invasive to the end-users and barely noticeable to the administrators!

Lessons Learned

By choosing to develop this system in house, we have gained invaluable experience and knowledge. The development infrastructure to support the development of our custom inventory and security correlation engine has lead to near-mastery of Modern Perl, PostgreSQL, centralized logging infrastructure, and VCS, both Subversion and Git.

The fact that we're storing everything in a relational database provides us with the ability to "mash up" data from disparate sources. We've been able to successfully respond to data calls from our parent organization with SQL statements that we can reproduce time and time again.

Sure, we didn't get free t-shirts, calendars, and pens from vendors. We didn't go to training sessions at fancy hotels to learn to use each piece of the system. We can't hire someone with a specific vendor certification to replace a team member if they leave. However, the entire team has learned to work together and everyone has increased their abilities in many different areas. Our Help Desk staff know Windows, Linux, Mac OS X, and some rudimentary programming. They also understand the network and how it works. This type of training, which lacks the polish and formality offered by the big name vendors, is invaluable to day to day Operations.

But, the best part of the system is it's being used, every day,by Ops team members to make a difference.

Resources

Open Source Software Mentioned

- **syslog-ng**: Replacement for standard syslog
  - http://www.balabit.com/network-security/syslog-ng
- **PostgreSQL**: Open Source Relational Database
  - http://postgresql.org
- **Netdisco**: Open Source Network Management System
  - http://netdisco.org
- **Perl Components** (http://perl.com)
  - **Catalyst**: MVC Framework
    - http://catalyst.perl.org
  - **POE**: Event driven Perl library
    - http://poe.perl.org
- **Snort**: Open Source Intrusion Detection System
  - http://snort.org
- **Puppet**: Open Source Configuration Management Engine
  - http://www.puppetlabs.com
- **Subversion**: Open Source Centralized Version Control System
  - http://subversion.tigris.org/
- **Cobbler**: Open Source Installation Server
  - https://fedorahosted.org/cobbler/

Projects by Brad Lhotsky (https://github.com/reyjrar)

- **svnutils**: A collection of Subversion utilities for automatic deployment and integration with Puppet
  - https://github.com/reyjrar/svnutils
- **optperl**: Spec files for installing Perl into /opt including integration with Puppet
  - https://github.com/reyjrar/optperl
- **POE::Component::Client::eris**: Perl module for subscription based log tailing
  - https://github.com/reyjrar/POE-Component-Client-eris
- **eris**: Network Console which collects and correlates data
  - https://github.com/reyjrar/eris