This column focuses on secure computing, providing tools and tips for those in the information security trenches. Each issue, we'll evaluate new technologies (primarily in the open source space) and discuss ways to integrate them into your organization.

We want to hear from you. Got a great utility or "magic" script that's saved you hours of tedious keyboard pounding? Something new we haven't heard about? Let us know at ciocorner@sandstorm.net.

Doc Talk

Effective documentation will help your organization keep pace with technological complexity

Unfortunately, one of the most important aspects of a successful IT infrastructure is also the dullest. It's often the worst part of the job: When you must stop creating and start explaining.

The benefits of effective documentation cannot be denied. When performing vital tasks such as support, troubleshooting, event response or managing employee transitions, an organization would be lost without detailed instructions. It is increasingly difficult for any one person to fully understand and recall the operations and interactions of all the various systems in an organization. We must all "step up" our documentation to keep pace with the ever-increasing technological complexity.

There are several key components to effective documentation. The documentation must be thorough and up-to-date. It must be easy to use, consistent and easy to find. Finally, the docs must be followed. In an effort to improve the way we handle internal IT documentation at Sandstorm, we recently examined these components in order to develop a system that would meet our needs.

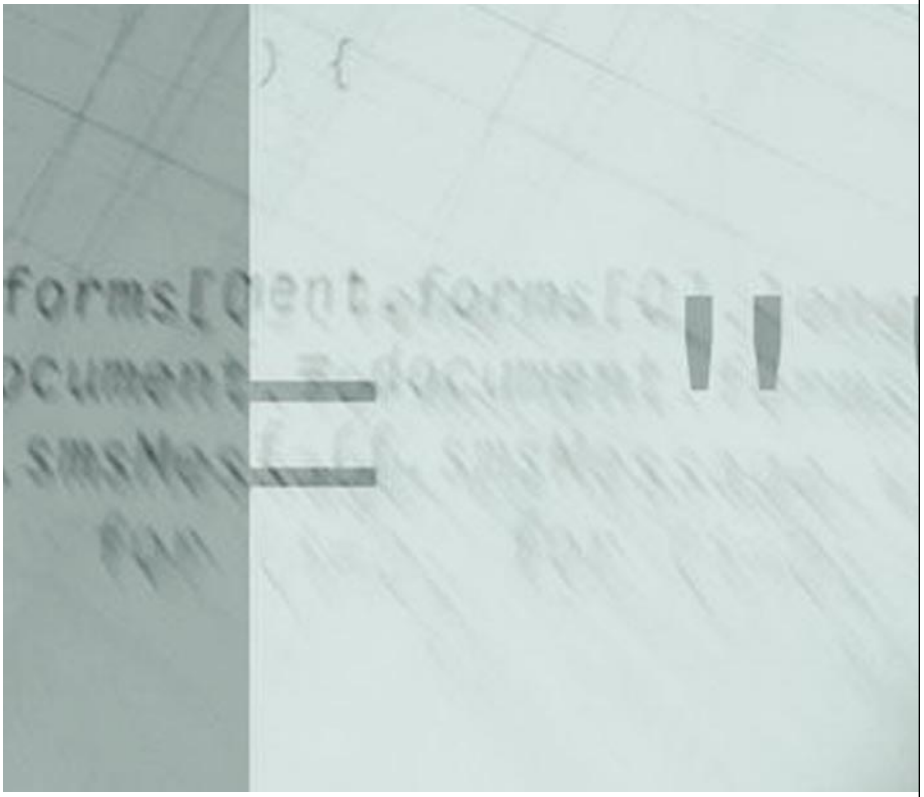
Our solution incorporates many of the ideas used by the Gentoo documentation project. Documents are stored as XML files on a web server running Apache and AxKit, which dynamically generates an HTML file from the XML data. We created our own DTD (used to define the data in our documentation), XSLT (used to translate our XML into HTML) and CSS (used to format the HTML), based loosely on DocBook and GuideXML. Additionally, CVS is used for revision control and multi-user editing capabilities.

HOW DOES IT WORK?

Let's say, for example, that Dave writes an XML document explaining a system backup procedure. Dave uses the DTD he checked out of the CVS tree to validate that the document is "well-formed." Upon validation, the new document is checked into the appropriate place in the CVS tree. The keyword expansion capabilities of CVS automatically update the author, date, version and revision history of the document.

On the organization's internal web server (intranet), a cron job automatically updates its copy of the documentation directory from CVS, notices the new document addition, and adjusts its index to include the new document. Dave's work is now available on the intranet.

When a client accesses the document, AxKit applies the XSLT processing instructions and delivers an HTML document to the client. The HTML



The benefits of effective documentation cannot be denied. When performing vital tasks such as support, trouble-shooting, event response or managing employee transitions, an organization would be lost without detailed instructions.

document includes an appropriate style sheet (CSS) for the type of media used for display (screen, print, etc).

The great thing about this system is that authoring documents is easy. You don't need a special editor to create the doc — just a plain text editor. We're talking XML, so by design the tags represent data in a human readable way. For example, if you want to create a warning box, the author need only write:

```
<warn>Danger Will Robinson</warn>
```

This system also allows us to control the formatting of various types of text. For example, we can easily format "F2" in a different font and color. Since this formatting is consistently applied using our style sheet, users of our documentation system more easily recognize when they have to perform an action.

The XSLT/CSS components take care of the formatting. Designing this system is just a matter of defining the appropriate tags that you will need to describe your various categories of information, then independently deciding how to format that information.

There is no one-size-fits-all solution to developing document tags. Industry specific processes will require specialized tags. You'll have to consider your environment when designing your DTDs. Here are a few examples that we included in our system.

To start off, paragraph text is a consistent size and font. Headings and sub-headings are bigger, bolder and underlined. Paths to files on disk, as well as commands entered by a user, are displayed using a mono-spaced font in a different color. Sections of the document displaying both command input and output are formatted in a color table. Special notes and warnings are displayed in appropriately colored boxes.

Required user input is defined with the <c> tag. So a document describing how to enter the BIOS on a system might be formatted like so:

```
<p>
Press <c>F2</c> to enter BIOS.
</p>
```

It is important to note that most problems surrounding documentation involves the human element. That is, at some point someone doesn't do what he or she's supposed to, and the whole system falls apart. While technology is extremely useful in combating some of these issues, it isn't a complete solution. All the IT acronyms in the world won't make employees write good docs.

WHAT ABOUT SECURITY?

Documentation isn't a security solution, per se, but rather a component to baseline management. Just because you have a document describing the configuration of your mail server, doesn't mean that it's safe from a remote root exploit. However, proper documentation tends to force you to "cover your bases." Greater security can be achieved by preventing important details from slipping through the cracks.

Take the example of a person leaving the company. Removing user accounts can often be a daunting task. An easy to read (and therefore easier to execute) document detailing the proper steps to disabling user accounts would minimize the chances of the administrator forgetting a crucial step. Without such a document, one might disable a user's password, but forget to delete the public key in ~/.ssh. Oops.

As with most aspects of security, it all comes down to the human element. Documentation is completely irrelevant if it isn't written, followed and updated (and you certainly won't be any more secure). However you can help encourage participation by making the process more attractive. Better output with less work averages out to a nice incentive. CDM

References:

http://www.gentoo.org
http://axkit.org/
http://www.oasis-open.org/docbook/
http://www.csszengarden.com

Walker Whitehouse *is CIO and* **Mike Yamamoto** *is a Network Systems Engineer at Sandstorm Enterprises, which develops aggressive software products for network monitoring, network forensics analysis, and security auditing including telephone scanning, penetration testing, and vulnerability assessment.*