Beave's Blog Documentation

Champ Clark III

Jan 02, 2019

Contents

I've been involved in computer security since before it was an industry. I started "hacking" in mid 1980's and I haven't stop. I've written books, white papers and given talks at conferences like Defcon, HOPE, CCC and Suricon. I've been programming for 30 years (Perl, C, PHP, etc).

I am the CTO at Quadrant Information Security (https://quadrantsec.com) which is an awesome place to work. I'm the primary developer of the Sagan Log Analysis Engine and Meer. Thoughts, ideas and opinions expressed in this blog are my own.

Linux junkie since the early 90s. Unix enthusiast before that. OpenVMS administrator. Occasional "software defined" radio hacker.

I like do things. Many of them involve computers. IRL - I travel as much as possible around the world with my family. I play Bass. I play Guitar. I play Drums.

CHAPTER 1

Readthedocs.org as a blog [2018/12/10]

1.1 Why I've never had a blog

I've always wanted to have blog.

The biggest problem is that I hate most "blogging" platforms. WYSIWYG editors irritate me. Blogging on the "web" isn't always that convenient.

So, no blog.

As a side note, I am a developer and one of my biggest weakness is documentation of the software I write. For reasons similar to why I hate most blogging platforms, I find most documentation platforms kludgy and annoying. I hate Wiki's, largely because they are easy to forget about and don't get updated. Being a "reader" of a Wiki is okay. Being the "writer" of a Wiki is a pain.

Not documenting software isn't an option. Without documentation, people can't learn about your work. In my opinion, developers who don't document or comment in code are doomed to failure. While documentation is a "burden", it is a key part of making any software complete and successful.

It is still a pain in the ass.

1.2 Concepts behind Sphinx/Readthedocs.org

I've noticed a trend of projects using the Sphinx Python Documentation Generator. I decided to give it a try.

The general concept is to keep your software documentation close to the source code. That is, make documentation part of the development cycle. With Sphinx, you use a very simple and easy to learn mark up language to create your documentation. You can then "build" (quite literally) your documentation. When you "compile" your documentation, the mark up language is converted to HTML, PDF, etc. Using different "themes" allows you to quickly create visually appealing and useful documentation. Since the documentation is inside your source code tree, you are more likely to update it.

Another bonus of Sphinx is that is makes documentation fun, or at the very least, bearable.

How does readthedocs.org fit into this?

'Readthedocs.org <https://readthedocs.org'_ provides hosting for your documentation. For example, the Suricata IDS uses Readthedocs.org. I've started moving all my documentation for Sagan and Meer to Readthedocs.org (Note: my documentation is far from complete and is still a work in progress!). However, Readthedocs.org is more than just a hosting place for documentation. It a great and easy (ie - automatic) way for publishing documentation as well.

Here's what I mean.

I keep most of my source code for projects on Github. For example, Sagan (https://github.com/beave/sagan) and Meer (https://github.com/beave/meer). The documentation for both these projects are in the https://github.com/beave/sagan/tree/master/doc/source and https://github.com/beave/meer/tree/master/doc/source directories.

When I updated my documentation for either project and 'push' it to Github.com, Readthedocs.org automatically "see's" the update and automatically "builds" ("make html") and hosts my documentation. This makes creating, publishing and hosting all very simple. Readthedocs.org has tied several great concepts together.

The Readthedocs team have a great "getting started" video at https://docs.readthedocs.io/en/latest/intro/getting-started-with-sphinx.html.

1.3 Building a Sphinx/Readthedocs Blog

Here's how I built out my "blog" platform.

First, I registered a domain. While "beaves-blog.redthedocs.org" is okay, I wanted something a bit more personal. With that in mind, I registered "beave.io". The idea is to redirect traffic from the "beave.io" to my "beaves-blog.readthedocs.org".

1.3.1 Apache

Setting up Apache to handle "beave.io" redirects to "beaves-blog.readthedocs.org." was pretty simple to accomplish. After getting Apache setup, i created a `.htaccess` file in my web root for "beave.io".

```
# This will redirect all "beave.io" requests to my "beaves-blog.readthedocs.io".
Redirect 301 / https://beaves-blog.readthedocs.io/
```

I also installed a LetsEncrypt certificate. Because, why not. More information about how to use LetsEncrypt and `certbot`, see: https://certbot.eff.org/lets-encrypt/debianstretch-apache.html

1.3.2 Github

I then made a new Github repo at https://github.com/beave/beave. This site will only contain blog postings like the one your are reading now. I then cloned the repo and made a "doc" directory. I then installed on my local system the tools I would need to use Sphinx.

```
sudo apt-get install python-dev build-essential python-setuptools
sudo easy_install sphinx
sudo pip install sphinx_rtd_theme
sphinx-quickstart
```

The "sphinx-quickstart" will setup your "documentation" environment.

1.3.3 Readthedocs

Once I have my templates created for my "blog", I then "pushed" it to Github.com.

I then pivoted to Readthedocs.org and logged in. I added a new repo in Readthedocs.org that points to https://github.com/beave/beave. I told Readthedocs.org to "import" and build documentation that repo.

From here on, when I "push" a new blog entry up to https://github.com/beave/beave, Readthedocs.org will clone my repo, build the "documentation" (blog) and publish it.

1.3.4 Putting it all together

If I want to make a new blog entry, this is what I do:

```
git clone git@github.com:beave/sagan # You could also use https://github.com/beave/
↪beave
cd beave/doc/source
vi index.rst        # Add my new blog entry to the index.rst
vi newblowentry.rst # Edit my new blog entry.
git commit -a
git push
```

If I want to see what the blog will look like before a commit/push, in the "doc" directory you can type this:

```
make html
```

That's it! I can now edit and publish blog entries from any where using the tools I enjoy!

1.3.5 Final Thoughts

There are some limitations to this approach. Readthedocs.org doesn't have any statistics about who has viewed your pages.

There isn't a "comment" area. Since I've gone down the road of using Readthedocs.org as a blogging platform, I might as well use Github.com "issues". If you would like to leave a comment about this or any other blog entry, submit an "issue" at https://github.com/beave/beave/issues

CHAPTER 2

Your "Threat Intel" is over rated.

. . . and here is why. . .

2.1 What is "Threat Intel"

Before we begin, let's define what I'm referring to as `Threat Intel`. In this arcticle, I am strictly referring to the use of `Indicators of Compromise` (IoC) lists. This includes the use of `bad` or `known` threat actor TCP/IP addresses, file hashes, file names, e-mail addresses, domain names, etc. The attacker in question in this article would be semi-advanced and possibly targetting your network.

2.2 Okay, It isn't useless

When I say, "Threat Intel is over rated", I do not mean that it is useless. I am repeatedly asked about Threat Intelligence and how we use it. I get asked in such a way that people expect it to bring "magic sauce" to their securty program. The general concept is that the more "intelligence" (data) one has, the better. This isn't the case.

First off, Threat Intel's usefulness is better applied backward facing and not forward facing. That is, Threat Intel is good to finding historical events that have happened in your network. In many cases, when you try to apply it to future events, you come up empty handed because the attacker has already changed there techniques.

2.3 Attackers know about "Threat Intel"

Do you know why? *Because attackers use "Threat Intelligence" too!*. This shouldn't be astonishing. What is astonishing to me is that many security minded folks don't seem to relize this. How do we know this? Because attackers have told us. Case and point, "Hacking Team Hack" (https://pastebin.com/0SNSvyjJ).

```
"I didn't want to make the police's work any easier by relating my hack of
Hacking Team with other hacks I've done or with names I use in my day-to-day
work as a blackhat hacker. So, I used new servers and domain names, registered
with new emails, and payed for with new bitcoin addresses."
```

From an attackers point of view, "Threat Intel" can be just as useful to them as it is to your team. You might look at Threat Intel as a "what to look for" list. The attacker might look at it as a "what they might be looking for and to avoid doing" list.

An attacker might use pervious undisclosed or new launch sources, modificaiton of attack binaries to modify file hashes) to avoid your threat intel. It is pretty simple and easy to stay ahead of "Threat Intel".

2.4 So when is Threat Intel useful?

As I stated before, the usefulness is in post success or attempted attack discovery. Using this data can show if you have been successfully or unsuccessfully compromised in the past from any individual threat.

This is a good thing and mining for such data within your organization is a smart thing to do. However, it doesn't much help you forward because it is likely the attacker has already changed there tactics, methods and proceedures.

Should you apply that data forward facing? Sure, you might get lucky. You might also introduce many false positives. Anyone who has worked in Threat Intel has seen shared hosting providers listed, public NTP servers and DNS server listed which can cause headaches.

Which brings me to another point.

2.5 Consider your sources

The general thougth is "put every source into a big pot and stir it up!". That is, the more sources you have for Threat Intel, the better. We use to have a term about this when I was younger about beer:

::

```
"It's about quantity not quality."
```

That gererally sums up what I believe many people feel. As I've gotten older, my views on Threat Intel (and Beer) have flipped.

::

```
It's about quality not quantity."
```

More != Better. In fact, it can cause you more headaches. Here is a simple example; how old is the data? If the TCP/IP address listed is several years old, then it's likely not being used in active attacks. If it is being used, *when was the last time it was used*. Your Threat Intel should be able to supply this information to your tools. This way you can make queries like, "Show me Threat Intel hits with a last seen data of 6 months or earlier". This is functinality we have built into `Bluedot` and Sagan.

We use to use a major Threat Intelligence provider. A lookup of an IP address might tell us that it was associated with a known botnet. This raise the following questions:

1. What sort of "botnet"?
2. When was it discovered?
3. When was it last seen?

4. Are there any notes or references around this type of bot?

We asked the provider about supplying this type of data. We where told that wouldn't be possible.

Verify and understand the feeds you are pulling from. More is not better. In face, there has been discussions about attackers "poisoning the data well" of Threat Intel simply to throw off security teams.

Verify your data. Even if it comes from a reputable souce, mistakes can be made. For example, we use `FBI Flash reports` as part of our data collection. We have seen incomplete MD5 sums in flash reports. We have seen feeds mistakingly put IP addresses like 8.8.8.8 as IOCs.

Don't just consume, validate before consumption.

2.6 WHen to play it forward

So if Threat Intel is really good on historical data but not as good forward facing, what do you do? As stated, applying publically disclosed Threat Intel will have limited use but you might get lucky. If you have the ability, adding internally generated data to your Threat Intel sources can be a good thing. If you've seen some cleaver attacks, collecting the IOCs and adding them into your back end (MISP, etc) is a great thing to do.

Another good source is "Honeypots" located within your network. Placement from the inside (to catch insider threats/lateral movement attempts) and external. Typcally, external Honeypots lead to a lot of data that isn't terribly useful. However, if done right and placed within your network, you might catch recon attampts from a more advanced attacker. This is cheap

2.7 Not your primary path

Threat Intel is only a tool or another part of the puzzle. Threat Intel shouldn't be dependant on by itself as a primary indicator. When coupled together with other indicators, it can add confidence to the event.