Building an Internet Security Feeds Service

JOHN KRISTOFF

John is a Network Architect at DePaul University's Information Services division and an adjunct faculty member at DePaul's College of Computing and Digital Media. He is also enrolled as a PhD student in computer science at the University of Illinois Chicago. John's primary career interests, experience, and expertise are in Internet infrastructure, Internet measurement, and internetwork security. John is or has been associated with a number of organizations and projects in associated fields of research and technology, some of which include DNS-OARC, IETF, FIRST, Internet2, NANOG, and REN-ISAC. jtk@depaul.edu

I produce a set of threat intelligence security feeds compiled from unsolicited communications to a distributed network of Internet systems. The umbrella platform for the project has a home at DataPlane.org where pipe-delimited text-based data feeds are freely available for non-commercial use. Read on for a behind-the-scenes look at how a mix of open source software, leased Internet hosts, and a dash of system administration deliver security feed data to some well-known and widely relied upon security projects and organizations.

Not long ago I proposed an antivirus programming-related idea for a class research project as part of my graduate course work. My professor felt "virus checkers are [not] an effective mechanism, because they are backward looking (at past history)." Presumably other types of threat intelligence systems that construct lists from observed, malicious activity associated with IP addresses, URLs, and domain names would be summarily dismissed along a consistent line of thinking.

My operational friends might mock a sneer and mouth "ivory tower, sheesh" under their breath at the very suggestion of their ineffectiveness. While there is an appeal to the idea that these sorts of approaches to security protection are discouragingly insufficient and futile, the use of threat data learned from past events is relied upon by many as a part of their security strategy. Whatever you believe about historical data for mitigation, threat intelligence in the form of black lists is widely used and can fetch premium prices when the data is unique, comprehensive, and reliable.

System Overview

The core components of the DataPlane.org security feeds are made up of three distinct subsystems as depicted in Figure 1. A set of sensor nodes collect unsolicited communications and relay logs of activity back to a central collection and processing system. The central collector stores events in raw log files and extracts fields of interest for insertion into a master database. Periodically, the database is scanned for recent suspicious activity seen by sensor nodes, which is extracted and pushed to a website for public consumption.

Producing security feed data would be nothing without a source from which to derive insight. How does one go about compiling source data? There are essentially three ways. One way is to get it from someone else. This is surprisingly very common in the security community. People and organizations share, sell, barter, and trade raw data all the time. If you ever compare threat intelligence between providers, do not be surprised to see overlap. Sometimes vendors produce the same intel independently, but when you see redundancy they are just as likely if not more so to have obtained raw data from a common original source.

The second way to obtain threat intelligence data is to actively seek it out. This may come from active monitoring, probing, data capture, crawling, and so forth. Obtaining data this way is often how one threat intelligence provider differentiates itself from another, since

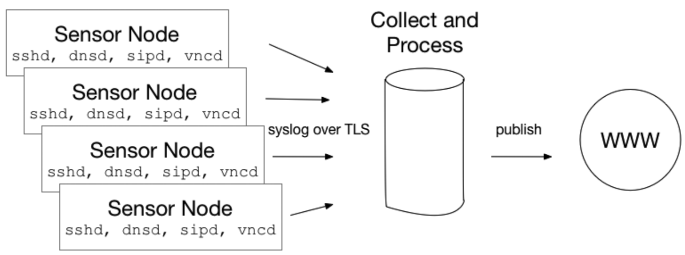

Figure 1: DataPlane.org security feeds system overview

these methods are often distinctly proprietary and unknown to others. This can also be the most costly and least robust approach. As targets of data collection activities change, move, react, or go away, data gathering processes must adapt else the end product may prove untrustworthy or absent of any insight at all.

The third way, a passive approach, is the easiest and cheapest, but it is not without limitations. Passive data collection is when you let the data come to you, from a honeypot or darknet monitor, for instance. The security feeds from the DataPlane.org project use a type of passive approach. DataPlane.org sensors mimic real applications, but they never allow access beyond simple unauthenticated application requests nor allow access to the system beyond an authentication phase.

I've had a fair amount of experience designing and compiling security feeds for nonprofit and commercial use. A few years ago I decided to run my own independent, free service for the community. Why do I do it? I can afford it, but most importantly because it pays dividends in subtle ways. For example, since I am also a PhD student, I can leverage the DataPlane.org platform for research ideas and data measurement experiments. Running DataPlane.org also gives me a platform with which to remain in the good graces of the security community. If nothing else, the security community is largely built upon reputation and trust. I've recently had offers of support and kudos from an array of benefactors. There is some non-zero amount of street cred that helps ingratiate myself with others I might not otherwise have had a chance to please.

Sensors

One drawback to a sensor network as used by DataPlane.org stems from what it does not or cannot see: targeted attacks, for example. It will fail to see threats that simply never cross its paths. My aim with the DataPlane.org project is to obtain a reasonably broad, sampled view of undirected Internet threats at diverse geographic locations (both from a physical location and an Internet routing perspective). Passive monitoring is of almost no value in IPv6 because of the sheer size of the address space. I focus on IPv4 networks with all the limitations this implies.

At recent count, the DataPlane.org project has approximately 100 sensor systems dispersed around the globe on six continents and at least one IP address in roughly 1/3 of all routable IPv4 /8 prefixes. While this isn't the world's biggest, most diverse, distributed network of systems, it might be one of the larger ones of this type run by a single individual.

This may lead to an obvious question. How much does this infrastructure cost? Before answering, let's just briefly consider how the network is not constructed.

I've been involved in similar projects in the past where people or organizations donate a sensor or threat intelligence data for the good of the project. While this can be a source of tremendous data, the reliability of the underlying source infrastructure is frequently a problem. Processes mysteriously stop, systems go down, or the friend at the organization who provided access to the raw data has left the organization and now no one left knows you or is motivated to fix a problem.

An approach used by many reasonably well-funded research groups such as CAIDA and RIPE is to send hosting volunteers a disposable system that can be plugged in, turned on, and then remotely managed with minimal additional supervision from host networks. These include the CAIDA Ark project (http://www.caida.org/projects/ark/) and the RIPE Atlas project (https://atlas.ripe.net/). These systems, too, can only gather data to which they are exposed, but at least in this scenario the only worry is the availability of power and connectivity. However, acquiring, provisioning, and delivering more than a handful of sensors to those who agree to host them may be cost-prohibitive for anyone operating on a tight budget.

For the DataPlane.org sensor network, I've opted to lease Internet nodes, usually from low-end virtual machine hosting providers. Two popular places to find low-cost hosting providers are https://www.webhostingtalk.com and https://www.lowendtalk.com. Prices vary but typically range from approximately $15 (US) to $60 per year for a minimally sized VM with one public IPv4 address.

I've built the network perhaps a little larger than it really needs to be with a little over 100 sensors, and my total cost is approximately $3000 per year. Luckily, the cost of running the DataPlane.org project is a luxury I can afford to fund myself. I plan to continue to do so as long as I'm gainfully employed and as long as it provides a value to myself and the community. More modest sensor networks could be set up for significantly less money.

One of the biggest challenges for the DataPlane.org project isn't so technical. Hosting providers come, go, get bought out, and change their infrastructure. Managing hosting provider dynamics accounts for most of the time I spend on the project. If you'd like to build your own network of leased systems, I can offer you a handful of tips, summarized below, having dealt with dozens of providers:

- **Historicity**: Consider the history of the provider. Beware of fly-by-night operations.
- **Reputation**: Many low-cost providers have mixed reviews, but the handful that consistently receive low marks probably deserve them for a reason.
- **Payment option**: PayPal is generally the safest for the customer. Do you really want to entrust your credit card information to providers with such slim margins? On a related note, I recommend avoiding any provider who wants a scan of government-issued identification. They don't need it, and you don't want them to have it.
- **Support**: You might not expect platinum service, but you should expect to receive a response to an email within one or two business days. An easy way to evaluate the liveliness of a provider is to send them a low-priority inquiry and see how they respond, if they do.
- **Professionalism**: This attribute applies to both the provider and customer. Customers should want a provider who is courteous in public and when interacting with customers. Likewise, the customer should be mindful of low-cost provider limitations, adjust expectations accordingly, and interact appropriately.

Setting up a DataPlane.org sensor consists of three basic steps: installing the OS, deploying the sensor applications, and configuring logging. I standardize on a minimal Debian stable distro. It is lightweight for low-powered VMs, easy to maintain, and almost always an option with every provider. My sensors require very little disk, memory, or network bandwidth. I can get away with just 256 MB of RAM, and was running an older system with just 64 MB not long ago. The DataPlane.org sensor configuration places only modest demands on system resources.

A sensor build includes multiple common network application listeners with which to produce threat intelligence data. These include DNS, SIP, SSH, and VNC, for example. For some applications, such as DNS and SSH, I use slightly customized versions of well-known implementations (e.g., BIND and OpenSSH, respectively). The SIP and VNC listeners are custom daemons specifically written for the DataPlane.org project rather than full protocol implementations. The custom daemons support enough of the base protocol to interpret unsolicited requests and log application-specific detail. These daemons can be found in the DataPlane.org GitHub repository (https://github.com/dataplane).

The final core capability of the sensor is to log all the desired monitored applications with syslog. Sensor applications of interest must log sufficient detail to be useful for threat intelligence purposes. For sensor applications like DNS, SIP, SSH, and VNC, this should include not only the source IP address responsible for generating the event, but also an NTP-synchronized timestamp set to UTC and a source port when transport protocols like TCP or UDP are involved. A source port helps networks doing network address translation correlate a specific event to an internal IP address. The syslog daemon should forward events of interest to a central collector. How DataPlane.org does this is detailed in the next section.

Central Collector and Processor

Within many networks, syslog is used to send locally generated logs from a host, daemon, or application to a remote collector for safekeeping and later analysis. The DataPlane.org sensor network is little more than a distributed set of syslog clients and a syslog server. However, because sensors are distributed globally on various types of hosting networks, I wanted to ensure some amount of log message reliability and privacy. Therefore all logs sent from sensors to the central collector are over a TLS connection. The sensor is configured with the central collector certificate, and likewise the central collector has a copy of the sensor certificate, providing some assurance each end is known to the other.

I prefer using syslog-ng as the syslog daemon at both the collector and sensor even though most modern Linux systems have migrated to rsyslog by default. The open source version of syslog-ng is reliably robust and includes some features I've grown accustomed to.

The central collector logs everything from each sensor system to a daily log file based on the unique IP address of the sensor system. The DataPlane.org project receives anywhere from a few KB to a few MB per day per sensor depending on how many public IPv4 addresses are active on the sensor.

I leverage two syslog-ng features to interpret received syslog messages and extract desired insight from them for insertion into a database. First, I make use of the pattern database. This is essentially an elaborate regular expression capability applied to syslog messages. Generally, syslog messages of interest have some structure or pattern to them, even if they are essentially text. When you know this structure, you can use the pattern database feature of syslog-ng to capture fields in a log message and then refer to them later in the processing chain as you might with back references in many scripting languages. Working with the pattern database feature requires close attention to detail and will take some getting used to, but once mastered it can prove quite powerful. The following is a very simple example to match on an sshd log message capturing the incoming source IP address:

```
<pattern>Connection closed by @IPvANY:SSH.SADDR@</pattern>
```

This pattern will match not only the connection formatting shown, but will capture the IP address (IPv4 or IPv6) of the host hitting the sensor. syslog-ng will store the IP address value in a variable named `SSH.ADDR`, which can be referenced later in the syslog-ng configuration. I make extensive use of the pattern database feature to capture various attributes of log messages, including source IP addresses, source ports, and application-specific detail. As log messages arrive and matches are made, the second syslog-ng feature I leverage is the ability to insert a processed version of a pattern-matched message into a database table. Once the pattern matches are defined, it is simply a matter of associating a matching pattern with a syslog-ng database destination. The following code block is an abbreviated syslog-ng configuration to demonstrate this concept with a PostgreSQL database:

```
parser p_patterndb {
  db_parser(file("/etc/syslog-ng/patterndb.d/example.xml") );
};
destination db_ssh{
  sql(type (pgsql) host("127.0.0.1") port("5432")
   database("example") table("ssh") columns ("logaddr",
          "stamp", "saddr",) values( "${SOURCEIP}", "${ISODATE}",
          "${SSH.SADDR}")
         );
};
filter f_ssh{
         match(
   "0123456789abcdef" value(".classifier. rule id")
          type("string")
         );
};
log{
  parser(p_patterndb); filter(f_ssh);
           destination(db_ssh);
};
```

Publication

The final core component of the security feeds system is to publish the final output to the community. This is a two-step process. The first step is to compile a feed from a data set in the database. The second is to push the feed to the DataPlane.org website for public dissemination. I've found an hourly update of the data feeds is generally sufficient for most users. I extract the most recent week's worth of events per feed category and generate a simple pipe-delimited text file that contains one event entry per line as defined in the commented section of the feed file. Intelligence threat providers or other interested parties can periodically pull these text-based security feeds from my website and process them further. I am currently in the process of making the security feed data available in real-time to users of the Security Information Exchange (SIE) platform run by Farsight Security (https://www.farsightsecurity.com/solutions/security-information-exchange/).

Conclusion

A number of open source projects, commercial providers, and incident response organizations make use of the security feeds DataPlane.org produces. I've been told that these security feeds are among the best and most reliably robust public set of feeds available. This seems somewhat surprising, because today I'm only producing feeds for a handful of basic network services. There are plenty more I could and want to do. The bad news is that I have not spent much time producing more varied security feeds for the past year since I started my PhD work. The good news is that I haven't had to actually do much to keep this security feeds system running as it largely runs itself. Additional detail about the implementation, including some source code for how many parts of the system are set up, can be found at the DataPlane.org GitHub project page. I invite you to take a look, contribute, or adapt what I have done to your own projects.

Perhaps one day, decades from now, the early 21st century may become known as the Internet's gangster era, a heyday where botnets, phishing emails, and DDoS attacks were commonplace. Awaiting that day implies an optimism that suggests we are now living in what will eventually be judged to be "simpler times." Whether or not this comes to bear, it seems plausible that, unlike 1920s America, the Internet do-gooders may be better remembered in the coming story than those G-men of yesteryear. Thanks to the proliferation of excellent, freely available software, sharing of insight between people and organizations, and the motivation to prevent the spread of malicious activity, few misdeeds or criminals run rampant for long.

The story, our story, is currently in progress. This article describes one modest approach to support a cast of characters helping to limit the spread of abuse on the Internet through the distillation and dissemination of security feeds. One day, we may all consider it "backward" and not worth the effort. Until that day comes, we hack.