Artillery

By Russ McRee – ISSA Senior Member, Puget Sound (Seattle) Chapter

Prerequisites

Linux or Windows system with a Python interpreter

I was reminded of Dave Kennedy's Artillery while attending and presenting at DerbyCon 4.0 in September, given that Dave is a DerbyCon founder. Artillery has recently benefitted from a major update and formal development support as part of Dave's Binary Defense Systems (BDS) company. The BDS blog[1] announced version 1.3 on November 11, which further prompted the discussion here. Artillery first surfaced for me as part of the ADHD project I covered during my C3CM discussion in October 2013's *toolsmith*.[2] While Dave's investments in the security community have grown, Artillery was initially maintained by Dave's TrustedSec organization then transitioned to BDS, given that group's "defend, protect, secure" approach to managed security solutions.

Artillery is an open source project created to provide early warning indicators for various attacks. Artillery was included in ADHD, the Active Defense Harbinger Distribution, because it spawns multiple ports on a system, a honeypot-like activity that creates "exposures" for attackers to go after. Dave described the fact that it also made blue teaming a bit easier for folks. Additional Artillery features include active file-system change monitoring, detection of brute force attacks, and generation of other indicators of compromise. Artillery protects both Linux and Windows systems against attacks and can integrate with threat intelligence feeds, allowing correlation and notification when an attacker IP address has previously been identified. Artillery supports multiple configurations, different versions of Linux, and can be deployed across multiple systems with centralized event collection.

With a ton of support, and additions coming in from all over the world to make Artillery better, Dave mentioned that plans for Artillery include much better support for Windows and expansion to allow a server/client model, moving away from purely standalone implementations. BSD's plan is to continue significant development for Artillery while ensuring it maintains its open source origins, allowing continued contribution back to the community Dave so readily embraces.

Laying in Artillery

Artillery installation is well documented on the Binary Defense Systems site[3] and is remarkably straightforward. Note that I only installed and tested Artillery on a Linux system.

On the Linux system you wish to install Artillery, simply execute `git clone https://github.com/trustedsec/artillery`, change directory to the artillery directory just created, run `sudo ./setup.py`, then edit to the config file to suit your preferences. I'll walk you through my configuration as a reference.

1. I created a directory called *holisticinfosec*, and added "/holisticinfosec/" to MONITOR_FOLDERS.
2. I enabled HONEYPOT_BAN=ON; you'll want to consider your implementation before you do this or you may inadvertently block legitimate traffic. You could also use WHITELIST_IP to prevent this issue and allow specific hosts, but as good as they are, whitelists can quickly become arduous to maintain. Bans and IPS-like blocks suffer from ye olde false positive issues when left unchecked.
3. I used Yahoo for my SMTP settings (USERNAME, PASSWORD, ADDRESS) and set ALERT_USER_EMAIL to my holistincinfosec.org address for alert receipt. Caution here as well as you can quickly SPAM yourself or your security operations center-monitored mail, and even cause an inbox DoS if your Artillery server(s) is busy. You can control frequency with EMAIL_TIMER and EMAIL_FREQUENCY; I suggest default settings initially (email alerts off) until you fine tune and optimize your Artillery implementation.
4. Brute force-attempts monitoring is cool and worthwhile. I enabled both SSH and FTP via SSH_BRUTE_MONITOR and FTP_BRUTE_MONITOR. Experiment with "attempts before you ban" as, again, you may not ban initially.
5. The THREAT_INTELLIGENCE_FEED and THREAT_FEED allow you to consume the BDS Artillery Threat Intelligence Feed (ATIF). It's represented as banlist.txt. You can pull from the BDS site or establish your own ATIF server for all your Artillery nodes to pull from. As with any blacklist, again, ensure that you want to block them all

1 https://www.binarydefense.com/bds/artillery-version-1-3-released-new-features-and-bug-fixes/.

2 http://holisticinfosec.blogspot.com/2013/10/c3cm-part-3-adhd-active-defense.html.

3 https://www.binarydefense.com/files/Artillery_Installation_Manual.pdf.

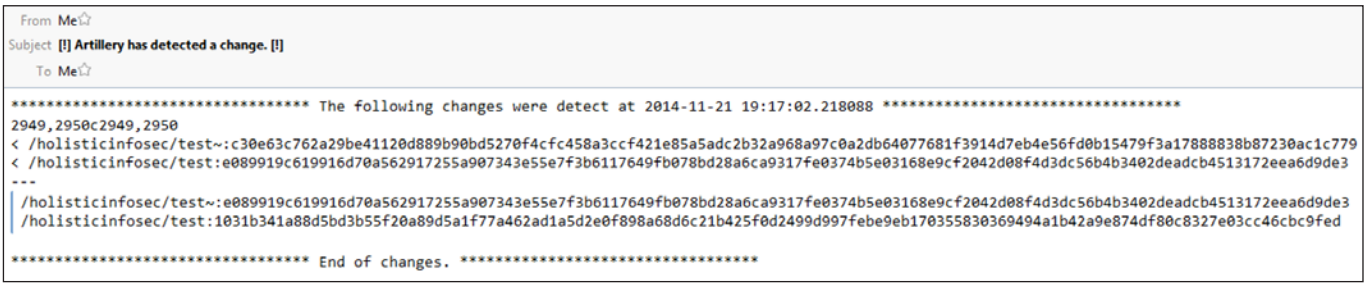

Figure 1 – Artillery alert for change to test file

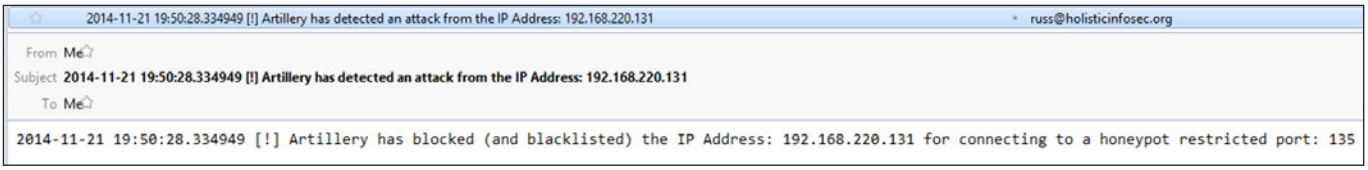

Figure 2 – Blocked and banned! Artillery stops attack traffic cold

as there are more than 86,000 entries in this list. You can also add other lists if you wish as well.

6. One of the most important settings is for your syslog preferences; you can log locally or write to a remote collector. This speaks to my defender's sensibilities and we'll discuss this further later as such.

The South Base Camp blog[4] (@johnjakem) also has a nice writeup on Artillery nuances; it's a quick read and worth your time.

Indirect Artillery fire

I conducted basic tests of Artillery functionality with email alerts and banning enabled.

For the folder-monitoring feature I wrote a file called *test* to the /holisticinfosec directory, including the sentence "Monkey with me," and restarted Artillery. When I monkeyed with test by adding snarky commentary, I was alerted via email and a log entry in the local syslog file. Figure 1 is the email alert indicating the change to the test file.

To blast the honeypot functionality, a nice Nmap scan sufficed with `nmap -p 1-65535 -T4 -A -v -PE -PS22,25,80 -PA21,23,80,3389 192.168.220.130`.

The result was an alert flood stating that [!] Artillery has detected an attack from the IP Address: 192.168.220.131 with example content as seen in figure 2.

I used Bruter to try and pound the FTP and SSH services, but because the Artillery configuration was set to only four attempts before banning, my dictionary attacks were almost immediately kicked to the curb. Darn you, Artillery! For this experiment I enabled SYSLOG_TYPE=FILE in my Artillery configuration, which writes event to /var/artillery/logs/alerts.log instead of syslog.

Remember, if you find yourself unable to connect to your Artillery server on a specific port or aren't writing test events, check your configuration file as you may have banned yourself. I did so more than once. Instantly solve this problem as follows: `sudo ./remove_ban.py 192.168.220.1`, where the IP address is that which you want to free from the bonds of iptables purgatory.

Direct Artillery fire

Artillery represents a golden opportunity to harken back to my C3CM guidance, particularly Part 2,[5] wherein I discussed use of the ELK stack, or Elasticsearch, Logstash, and Kibana. You can quickly set up the ELK stack up using numerous guides found via search engine and customize it for Artillery analysis.

Rather than repeat what I've already documented, I took a slightly different tack and utilized my trusty and beloved Security Onion VM. Security Onion includes ELSA (Enterprise Log Search and Analysis) which is a "centralized syslog framework built on Syslog-NG, MySQL, and Sphinx full-text search."[6] I could and should do a *toolsmith* on ELSA alone, but it's so well documented by project developers and Security Onion's Doug Burks, you'd do well just to read their content. To make use of ELSA, I needed only point Artillery syslog to my Security Onion server by changing the /var/artillery/config file as follows:

1. Changed SYSLOG_TYPE=LOCAL to SYSLOG_TYPE=REMOTE
2. Set the IP address for my Security Onion server with SYSLOG_REMOTE_HOST="192.168.220.131"
3. Restarted Artillery from `/var/artillery with sudo ./restart_server.py`

"That's it?" you ask. Indeed. I logged into ELSA on my SO server after hammering the Artillery node with varied malfeasance, queried with host=192.168.220.130; you can see the results in Figure 3.

4 http://www.southbasecamp.com/blog/setting-up-a-honeypot-artillery/.

5 http://holisticinfosec.blogspot.com/2013/09/c3cm-part-2-broids-with-logstash-and.html.

6 https://code.google.com/p/enterprise-log-search-and-archive/.

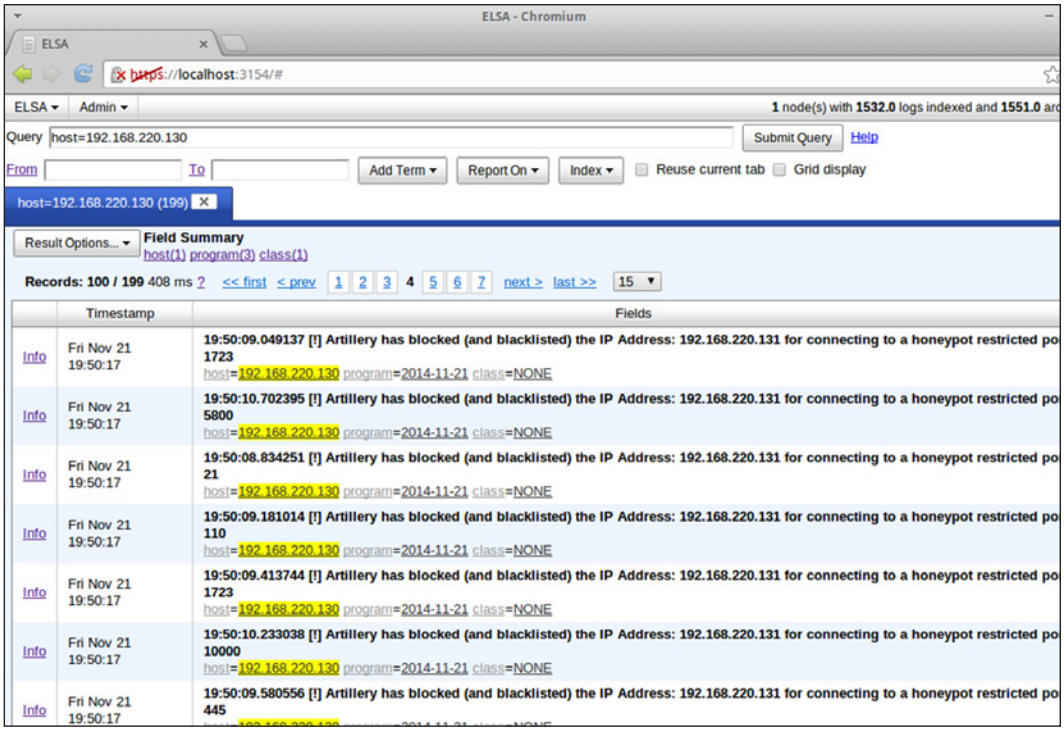

Figure 3 –Artillery events written to a remote Security Onion ELSA instance

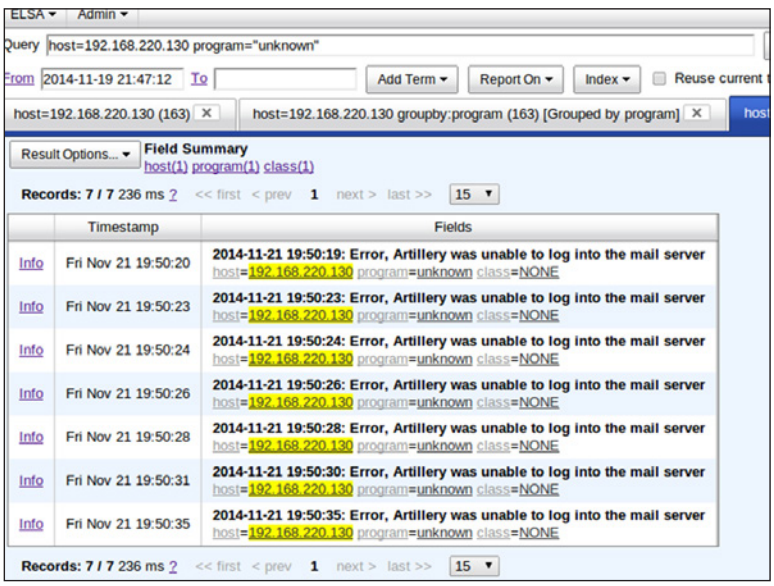

Figure 4 – Artillery alerts me that I am a spammer

ELSA provides you with a number of query options and filters so even if you have multiple Artillery servers, you can zoom in on specific instances, dates, or attack types. A query such as host=192.168.220.130 groupby:program led me to program="unknown," which in turn alerted me that I ended up being banned from Yahoo for spamming my account with alerts as seen in figure 4.

It's always good to check your logs from a variety of perspectives☺.

I intend to write some additional scripts for Artillery analysis and parsing and devise additional means for incorporating threat intelligence to and from my Artillery instance. Let me know via the blog comment or Twitter how you've done the same.

In conclusion

Artillery, on many levels, is the epitome of simplicity, which is part of why I love it. If you possess even the slightest modicum of Python understanding, the Artillery source files should make complete sense to you. Properly tuned, I can't really think of a reason not to run Artillery on Linux servers for sure, and maybe Windows boxes where you have Python installed. Just remember to practice safe banning; you don't want to drop production traffic. I'm really glad Dave's Binary Defense Systems interest has taken over care and feeding for Artillery and can't wait to see what's next for this fine little defender's delight.

It's that time of year again! Be ready to vote for your favorite tool of 2014. I'll soon post the survey to my website or blog and tweet it out by mid-December. We'll conclude voting by January 15, 2015, and announce a winner soon thereafter. Please vote and tell your friends and coworkers to do the same.

Ping me via email or Twitter if you have questions (russ at holisticinfosec dot org or @holisticinfosec).

Cheers…until next month.

Acknowledgements

—Dave Kennedy (@HackingDave), TrustedSec, Binary Defense Systems, and DerbyCon

About the Author

Russ McRee manages the Threat Intelligence & Engineering team for Microsoft's Online Services Security & Compliance organization. In addition to toolsmith, *he's written for numerous other publications, speaks regularly at events such as DEFCON, Black Hat, and RSA, and is a SANS Internet Storm Center handler. As an advocate for a holistic approach to the practice of information assurance Russ maintains holisticinfosec.org. He serves in the Washington State Guard as the Cybersecurity Advisor to the Washington Military Department. Reach him at russ at holisticinfosec dot org or @holisticinfosec.*