SANS Institute
InfoSec Reading Room

To Block or not to Block? Impact and Analysis of Actively Blocking Shodan Scans

This paper details an experiment constructed to evaluate the effectiveness of blocking Shodan search engine scans in reducing overall attack traffic volumes. Shodan is considered to be part of an attacker s toolset, and there is a persistent perception that blocking Shodan Scans will reduce an organization s attack surface. An attempt was made to determine what effect, if any, such a block would result in by comparing attacker traffic before and after implementing a block on Shodan scans, and by determining the complex...

To Block or not to Block? Impact and Analysis of Actively Blocking Shodan Scans

GSTRT Gold Certification

Author: Andre Shori, andreshori@gmail.com

Advisor: Dr. Johannes Ullrich

Accepted: 25 August 2018

Abstract

This paper details an experiment constructed to evaluate the effectiveness of blocking Shodan search engine scans in reducing overall attack traffic volumes. Shodan is considered to be part of an attacker's toolset, and there is a persistent perception that blocking Shodan Scans will reduce an organization's attack surface. An attempt was made to determine what effect, if any, such a block would result in by comparing attacker traffic before and after implementing a block on Shodan scans, and by determining the complexity of performing such a block. The analysis here may provide defenders and managers with useful data when deciding on whether or not to devote resources to blocking Shodan or other similar internet-connected device search engines.

1. Introduction

Launched in 2009 by John Matherly, Shodan is a search engine that focuses on internet-connected devices. Shodan is akin to other internet search engines such as Google or Bing. Shodan functions by scanning the internet for internet-connected devices (comparable to how Google "crawls" the web indexing web pages). Other similar search engines that focus on internet-connected devices are Censys (Censys, 2018) and Zoomeye (Zoomeye, 2018). In 2013, CNN famously described Shodan as "[t]he scariest search engine on the Internet [sic]" (Goldman, 2013). ZDNet magazine also labeled Shodan a "dangerous tool" and a "[V]oyeurs dream" that allows strangers to snoop on sleeping kids and intimate personal moments (Osborne, 2016).

Granted, Shodan does appear to be a favorite tool with hackers and penetration testers. The search engine makes it simpler for attackers to look for internet-connected devices and then check if they contain exploitable vulnerabilities. Guides do exist on how to use Shodan as part of an attack. DeepDotWeb, a site for the dark web, touts Shodan as one of the best hacker-friendly search engines to use (Ciphas, 2016). Online educational site, Cybrary, labels Shodan as "[t]he Hacker's Search Engine [sic]" and offers a course on how to use Shodan as part of an attacker's toolbox (Fernandes, 2016). Shodan's API further enables easy integration into tools solely purposed for an attack. Autosploit, a Python-based tool that combines Shodan and Metasploit into an automated weaponized platform for the masses is the most recent example of how attackers integrate Shodan into their tactics (Townsend, 2018).

As such, cybersecurity operations may be choosing to devote resources to blocking Shodan in the hope that it will reduce attack surfaces. Evidence supporting such efforts emerge continually year after year in various guides such as those found on GitHub (Gamblin, 2018) and Reddit ([Question] Blocking Shodan.io?, 2018). These guides and others like them detail various efforts on how to block Shodan scans and prevent one's devices from appearing on the Shodan database. The possibility and concern over Shodan playing a role in data breaches has also been used by vendors such as Checkpoint to promote their products and services (Checkpoint, 2017).

The following experiment attempts to evaluate how useful blocking an internet-connected device search engine like Shodan is in reducing or preventing attacks on an organization's network by comparing attack traffic before and after the implementation of a popular blocking method against Shodan scans. There is currently little to no available

empirical evidence or evaluation of methods used to block scans from Shodan scanners or crawlers. Effectively preventing Shodan scans may involve more than just a simple blocklist on a firewall (currently the most common method espoused by guides on this subject). Results of this experiment may provide researchers and cybersecurity practitioners useful data to help in deciding if it is worth it for companies or organizations to devote resources toward actively blocking Shodan or other similar scanners. Based on a review of available literature, this experiment comprised the first-ever attempt to measure the impact of blocking Shodan scans.

Evidence gathered and analyzed during this experiment may provide a better understanding of efforts needed to block Shodan scans more effectively and highlights some of the unexpected challenges associated with blocking Shodan, as encountered over the course of this research.

1.1. Shodan Search Engine

Like all modern search engines, Shodan "crawls" the internet. Shodan focuses on internet-connected devices and related metadata of those devices. When encountering a device, Shodan attempts to query the device banners (the message presented whenever someone or something connects to the device) to identify what that device is. Shodan also gathers information on the devices such as the IP address, port number, hostname, ISP host and GeoIP information (Shodan, 2018).

Shodan then correlates returned information results into a publicly searchable database. Shodan's database is indexed and searchable by various filters including the information mentioned above or the city, country, a range of dates or even the device's operating system. Shodan also has plugins to run searches from the convenience of a browser and an Application Programming Interface (API) that allows communication between Shodan and customized applications created by a developer.

2. Research method

A previous study on internet-connected device search engine impact on attack volume was conducted in 2017. Dubbed "SIPHON" by its authors, the study was conducted by utilizing many geographically distributed IP addresses to connect several Internet of Things (IoT) devices to the internet and then watched traffic grow as these devices were picked up by internet-connected device scanners (Guarnizo et al., 2017). No attempt was made during

SIPHON to implement a block of incoming search engine scans such as Shodan and record subsequent traffic impact.

For this study, the experiment instead utilized an existing Honeypot host to determine what impact, if any, blocking Shodan scanners would have on the daily volume of incoming attack traffic. Using a pre-existing Honeypot for the experiment had several anticipated benefits. First, as Honeypots are designed to trap and analyze attacker traffic, almost all incoming traffic to the host could then be considered hostile and unauthorized. Second, predefined firewall logs provided by the host also made it simpler to isolate and analyze incoming traffic. Third, some unique characteristics of this host made it simpler to select an appropriate sample for analysis, avoiding the need for more time-consuming analysis of large log files. Finally, employing a pre-existing infrastructure also significantly reduced setup time and configuration.

The experiment consisted of three phases. Phase One of the study recorded the initial configuration of the host and then baselined existing attacker traffic to establish patterns of behavior and overall volume. This phase also set the analysis framework and methodology for changes in traffic patterns in response to Shodan blocks. A sample of available IP addresses was selected and tested for a minimum +-10 Confidence Interval at a 99% Confidence Level to ensure they were an appropriate focus for the experiment. Phase One was run from 30 June to 12 July 2018. Phase Two utilized the Linux iptables firewall utility (iptables) to block incoming IPs from the Shodan.io domain (the domain where most Shodan scans appeared to originate). After the implementation of new firewall rules (Shodan Block), the host's log files were analyzed to ensure that the firewall was working as configured. Phase Two was run from 13 to 20 July 2018. Phase Three examined attack traffic volume after Shodan Block. Phase Three attempted to measure the effectiveness of Shodan Block by checking if the sample IPs and related ports utilized in the experiment no longer showed up on Shodan. Incoming attack traffic was compared to earlier baselines to determine changes in volume and type. Phase Three ran from 21 July, and the data gathering portion of the experiment was concluded on 31 July 2018.

All logs and results from the experiment have been archived and are available for request for any subsequent study. Requests for access to these log files can be made to Dr. Johannes Ullrich at jullrich[at]sans.edu. Please note that because the host machine utilized in the

experiment is a live host; for security reasons, some experiment details have been redacted, obscured or intentionally omitted.

2.1. Detailed Experiment Setup

This experiment utilized a pre-existing host, designated here as "HoneyHost," on which several medium and low interaction Honeypots are running. HoneyHost, running on Linux 4.4.0-130-generic #156-Ubuntu (x86_64), includes amongst its Honeypot appliance offerings, an Apache Honeypot that emulates several vulnerable Web Applications (Apache) and a Cowrie SSH/Telnet Honeypot (Cowrie). Both Honeypot appliances offer multiple ports for attackers to connect to using IPv4 (IPv6 was already disabled on HoneyHost before the experiment). Apache and Cowrie are both configured as low interaction Honeypots, which effectively simulate available services for attackers to attempt to connect to, but do not allow any actual further interaction (the servers are not actually "running").

HoneyHost's Honeypots are primarily used or implemented for attacker traffic logging and analysis. As HoneyHost only runs Honeypots, it was assumed that (apart from a minimal amount of remote management traffic) all incoming traffic to HoneyHost constitutes attacker traffic and is of relevance to the experiment. Incoming connections to HoneyHost's firewall are continually logged to a log file, called DSHIELDLOG, and uploaded daily to the Internet Storm Centre as part of the Dshield project. Dshield is a community-based firewall log correlation and aggregation project (SANS Institute, 2018). During the experiment, the uploads of the firewall logs to Dshield continued uninterrupted.

DSHIELDLOG was the primary log file for analysis of traffic volume during the experiment. DSHIELDLOG is a standard iptables log which lists various header fields such as the Date, Time, Source and Destination IPs, MAC address, packet length, TCP Flags, Protocol type and Source and Destination Port, but does not list the actual packet contents. A CRON job (an automated scheduled task) was running every 24 hours on HoneyHost to create a copy of DSHIELDLOG for that day and then compress the file. Each daily DSHIELDLOG was then renamed using the naming convention dshield.log-2018<MMDD>.gz. For experimental purposes, each log entry in DSHIELDLOG was counted as a single record of an incoming attack.

Statistically, DSHIELDLOG was considered the total population during the experiment for incoming all records (this log recorded traffic AFTER the rate limit was

applied). Some attempts were made to capture and analyze all incoming traffic before the rate limit, however the sheer volume of unfiltered traffic (as much as 10 GB per day) made this unfeasible due to storage limitations on the server and the hours it took to run even a simple query on one day's records. For example, on 27 July there were 1,421,933 records on Port 23 as compared to 31,347 on DSHIELDLOG.

After preliminary examination of DSHIELDLOG, default ports 22 (SSH), 23 (Telnet) and 80 (web) were selected for analysis. All three ports did provide sufficient traffic for baselining and proved adequate for recording changes in traffic volume.

During the experiment, a subset /24 block of sample IPs (255 consecutive IP addresses) was selected from HoneyHost and was designated SAMPLEBLOCK. SAMPLEBLOCK, at a 99% confidence level, yielded confidence intervals of +-7.79 for Port 22 traffic, +-1.89 for Port 23 and +-3.19 for Port 80 over the experiments phases and was confirmed as an appropriate sample for analysis.

2.2. ISP Special Arrangement

The current arrangement is for the ISP to allocate available (spare) IPv4 addresses dynamically and to forward traffic destined for those IPs to HoneyHost. HoneyHost connects to its ISP via a single external interface, and all traffic passes through this interface and onwards to HoneyHost's Honeypots. This arrangement with the ISP predated the experiment and remained unaltered for the duration of the experiment.

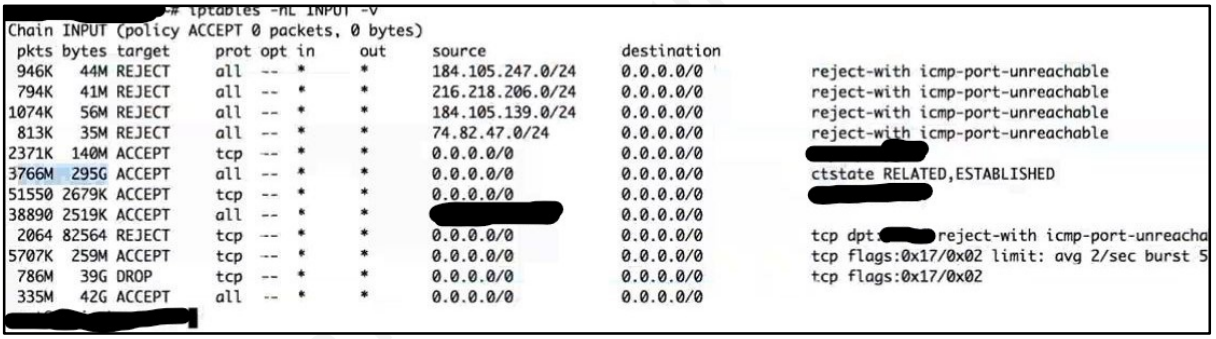
These dynamic IP addresses from the ISP were periodically and randomly reallocated to other customers, or they were de-allocated and then automatically reassigned to HoneyHost during the experiment. A mechanism for continuous monitoring of when the ISP allocated or deallocated IPs to HoneyHost was unavailable during the experiment duration.

It was observed, however, that all SAMPLEBLOCK addresses were present on Shodan during the entire duration of the experiment. Although some SAMPLEBLOCK IPs may have been deallocated temporarily during the experiment, so long as all SAMPLEBLOCK IPs were still showing up in Shodan, attackers could still utilize this information to craft scans and attacks. To compensate for any changes in IP allocations to SAMPLEBLOCK, reports from Shodan were generated several times during the experiment. Based on incoming traffic analysis during the experiment, impact from one or two SAMPLEBLOCK IPs that were

temporarily allocated elsewhere was observed to be minimal and did not significantly impact results.

2.3. Pre-experiment Packet Flow

All incoming traffic forwarded from the ISP to HoneyHost was screened utilizing Netfilter, Ubuntu's built-in Linux kernel firewall (Ayuso, 2014). Netfilter was configured using iptables which allows listing and editing of the firewall ruleset. Before commencement of the experiment, the following rules were in operation on HoneyHost as detailed in Figure 1.



```
[redacted]# iptables -nL INPUT -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
 946K   44M REJECT     all  --  *      *       184.105.247.0/24     0.0.0.0/0           reject-with icmp-port-unreachable
 794K   41M REJECT     all  --  *      *       216.218.206.0/24     0.0.0.0/0           reject-with icmp-port-unreachable
1074K   56M REJECT     all  --  *      *       184.105.139.0/24     0.0.0.0/0           reject-with icmp-port-unreachable
 813K   35M REJECT     all  --  *      *       74.82.47.0/24        0.0.0.0/0           reject-with icmp-port-unreachable
2371K  140M ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           [redacted]
3766M  295G ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
51550 2679K ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           [redacted]
38890 2519K ACCEPT     all  --  *      *       [redacted]           0.0.0.0/0
 2064 82564 REJECT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:[redacted] reject-with icmp-port-unreacha
5707K  259M ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp flags:0x17/0x02 limit: avg 2/sec burst 5
 786M   39G DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp flags:0x17/0x02
 335M   42G ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
[redacted]
```

Figure 1: HoneyHost pre-existing Firewall ruleset. Note: some information has been redacted for security purposes.

Incoming packets were screened according to the pre-existing ruleset and if permitted, were then logged in DSHIELDLOG. Of note were the rules to ACCEPT: "tcp flags:0x17/0x02 limit: avg 2/sec burst 5" and DROP "tcp flags:0x17/0x02" which effectively limited the amount of incoming traffic forwarded to the Honeypot applications to about one packet per second and dropped the rest. Before implementation of this rate limit, the volume of incoming traffic would cause Cowrie to become overwhelmed and crash. The rate limit was left in place during the experiment to ensure that all Honeypot applications remained stable. Other pre-existing rules were to reject ICMP (pings). ICMP traffic averaged less than 0.5% of daily traffic volume and for simplicity, were also included in overall incoming traffic volume results of the experiment. DSHIELDLOG was the primary log file used for traffic analysis during this experiment and was uploaded daily to the Dshield project.

Packets that were permitted through the firewall were then sent on to the NAT ruleset to be forwarded to HoneyHost's internal IP and ports, as detailed in Figure 2.

```
[redacted]:~# iptables -t nat -n -L PREROUTING
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
LOG        all  --  [redacted]           0.0.0.0/0           limit: avg 1/sec burst 5 LOG flags 0 level 4 prefix " PREROUTING "
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:12222 to:[redacted]4422
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:22 to:[redacted]4422
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:2323 to:[redacted]4423
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:23 to:[redacted]4423
DNAT       all  --  0.0.0.0/0            0.0.0.0/0           to:[redacted]
[redacted]:~#
```

Figure 2: NAT PREROUTING rules. Some details have been redacted for security purposes.

Packets for Ports 22 and 12222, destined for Cowrie, were forwarded to an internal port of 4422. Packets destined for Cowrie on Ports 23 and 2323 were similarly forwarded to internal Port 4423. All other remaining traffic was forwarded with the destination port unaltered. NAT also generated a log file; however, the destination addresses were already translated in that log to HoneyHost's internal IP. Due to this condition, the NAT log file was not used for traffic analysis, but rather DSHIELDLOG was used instead. A simplified diagram of HoneyHost's packet flows is visible in Figure 3.

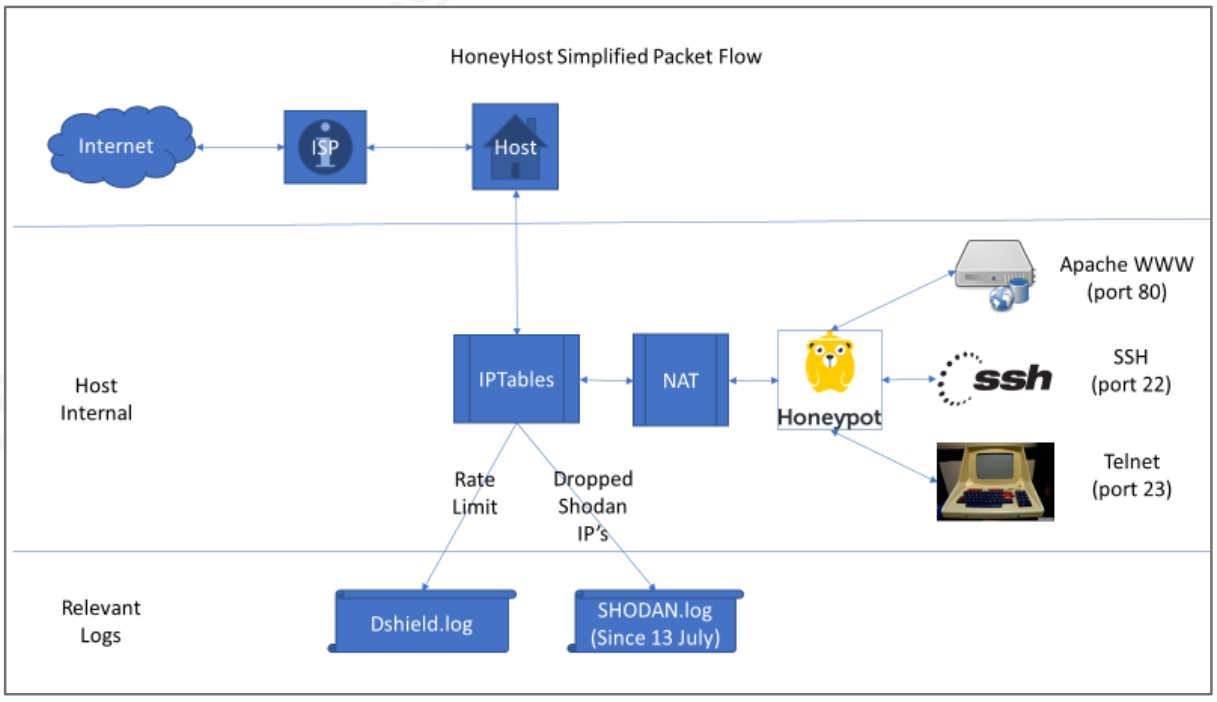


Figure 3: Simplified Packet Flow on HoneyHost. Incoming packets from the ISP first go through the built-in firewall (iptables) then are NAT'ed to the appropriate Honeypot appliance/port.

Outgoing packets followed the same path, in reverse. No additional filter or firewall rules were applied to outgoing packets.

Andre Shori, andreshori@gmail.com

2.4. Pre-Baseline Shodan Check and Iptables Improvement

On 25 June, before the start of the Baseline, an initial check on Shodan revealed that required information to commence the experiment was missing on the Shodan database. Shodan listed the presence of only 226 out of 255 possible IPs from SAMPLEBLOCK. In addition, none of the SAMPLEBLOCK entries listed Ports 22, 23 or Port 80 as open on Shodan at that time. It was critical for the Baseline that Shodan listed all target IPs and related ports as open on HoneyHost. If attackers could not view the target IPs and ports on Shodan, then incoming attack traffic volume could never be attributed to attackers utilizing Shodan as part of their attack toolsets. Ports that did show up at that time for SAMPLEBLOCK were for a WebLogic Honeypot running on Ports 9000 and 9001, however attack traffic volume on Ports 9000 and 9001 was deemed insufficient for the experiment.

After the initial check of Shodan was performed and the missing SAMPLEBLOCK IPs and ports were noted, it was then postulated that the rate limit on the firewall may have been impacting successful scans by Shodan of available ports on HoneyHost. The decision was then made to use iptables to add more rules before the rate limit. These rules were to accept all traffic on ports 22, 23 and 80 then apply the rate limits. Iptables was then used to reconfigure the firewall and the updated ruleset is seen in Figure 4. These additions proved to be effective in allowing Shodan to scan the Honeypot appliances on HoneyHost correctly and by the start of the Baseline, 144 Port 80 entries, 32 Port 23 entries, and 18 Port 22 entries had shown up for SAMPLEBLOCK.

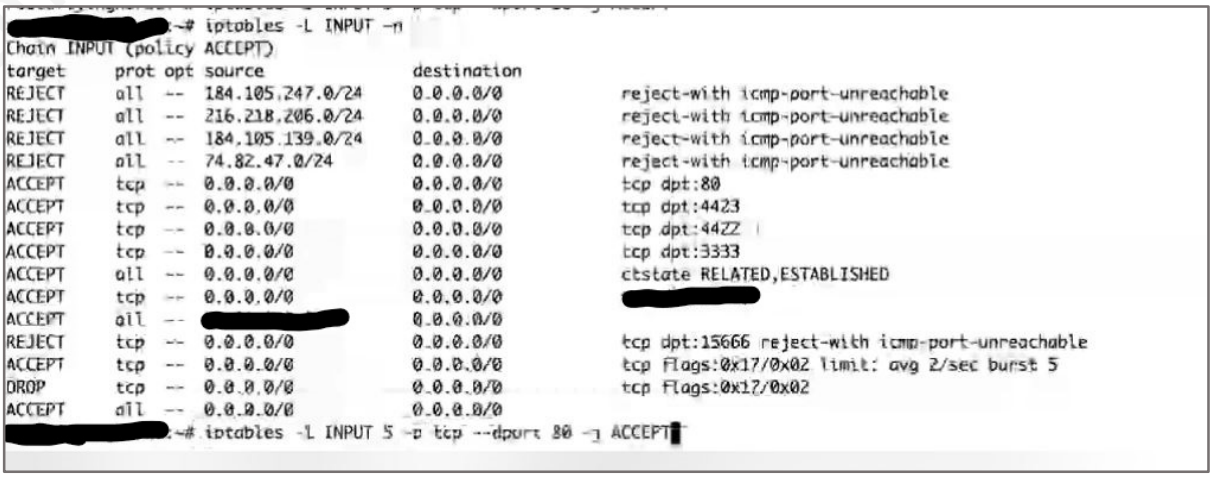


Figure 4: Edited firewall rules to allow ports 22,23,80 before the rate limit. Some details have been redacted for security reasons.

Before commencing the Baseline, manual scans of SAMPLEBLOCK were conducted daily to ensure that these changes to iptables were valid and to help speed up the population of Shodan's database with relevant information for the experiment. The Shodan command

line interface was then installed using instructions found on an online guide ("Shodan Command-Line Interface," 2018), accessed from another machine. The command "# shodan scan submit --force --verbose (SAMPLEBLOCK)/24" was used from that machine to manually force these scans. Shodan reports were then run on SAMPLEBLOCK to determine if new entries on Shodan were present. Shodan does limit manual scans every 24 hours if an IP address has been scanned already, which meant that SAMPLEBLOCK entries on Shodan populated more slowly than initially expected.

By 29 June, just before the Baseline commenced, all 255 SAMPLEBLOCK IPs were finally present on Shodan. SAMPLEBLOCK IPs were also beginning to report an increased presence of port 80, 22 and 23, as displayed in Figure 5 This number continued to grow during the experiment duration, although Shodan did not ever report these ports as "Open" for all 255 SAMPLEBLOCK IPs. Despite not all SAMPLEBLOCK IPs on Shodan each reporting the presence of all three ports for the experiment, the Baseline on incoming traffic volumes commenced on 30 June according to the planned experiment timeline.

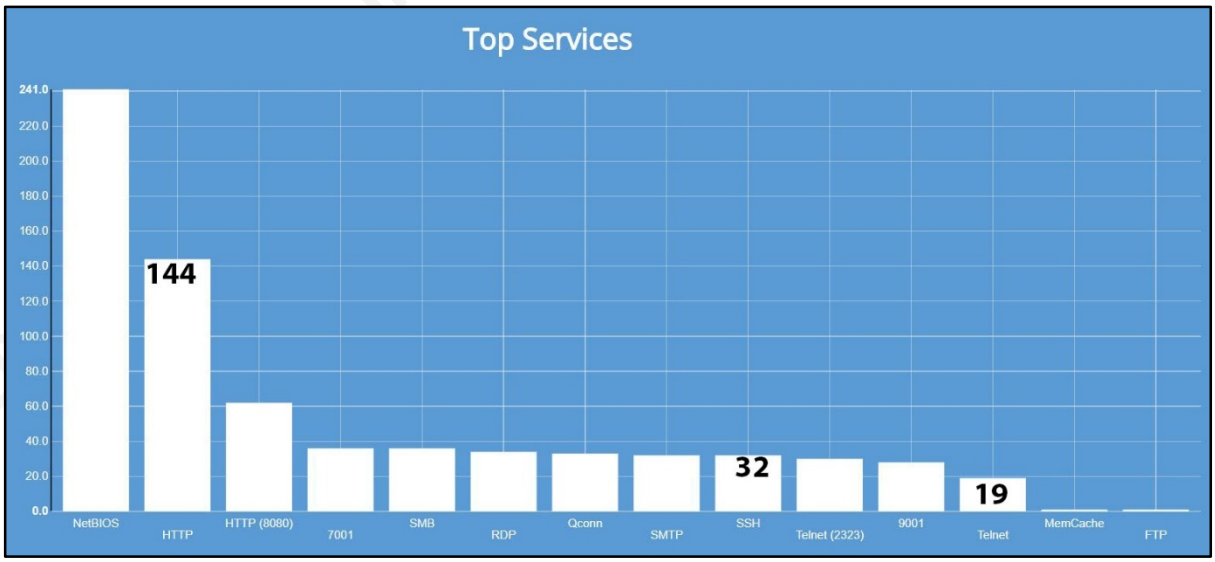


Figure 5: Shodan statistics for SAMPLEBLOCK on 29 June, just before the Baseline. 180 IPs reported the presence of port 80, 32 IPs reported the presence of port 22 and 19 IPs reported port 23.

3. Phase One: Baseline Results

The Baseline was run from 30 June to 12 July 2018. During this period, DSHIELDLOG was examined daily for traffic destined for SAMPLEBLOCK Ports 80, 22 and 23. Daily counts of the number of individual Source IPs sending traffic to SAMPLEBLOCK was also recorded to provide an additional statistic for later comparison with the Dshield Project.

SAMPLEBLOCK traffic volume on Port 22 averaged 178 log entries per day with an increase of seven percent over the Baseline period. SAMPLEBLOCK Port 80 averaged 679 logs per day, a decrease of -2.8%. Despite only having 43 entries on Shodan for SAMPLEBLOCK, Telnet Port 23 reported the most traffic with an average of 2,314 log entries per day and an increase of 6.2% over this period. Figure 6 displays a chart of daily traffic (log entries) per port for the Baseline period.

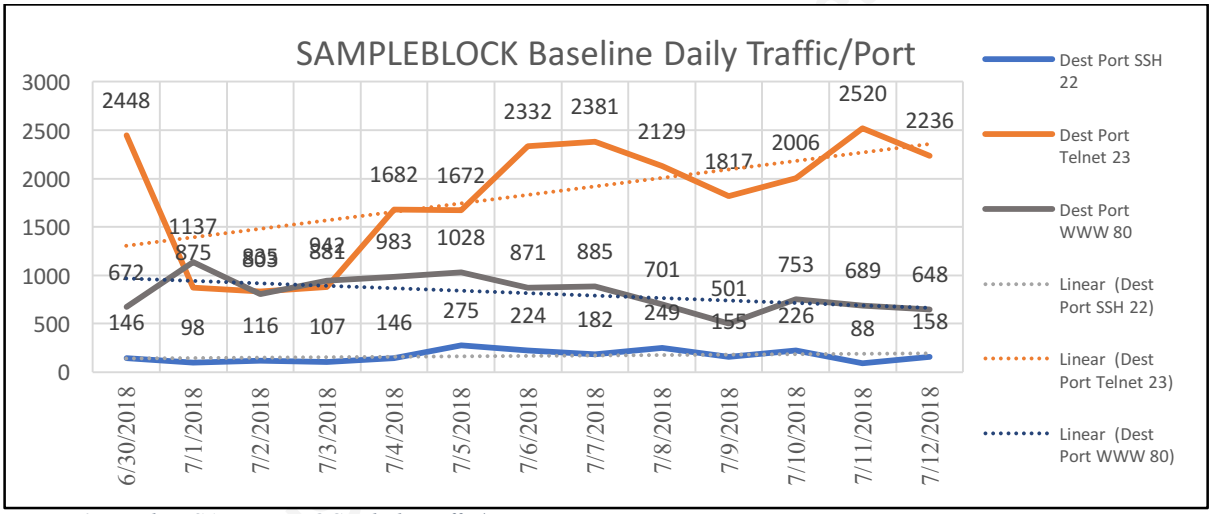


Figure 6: Baseline SAMPLEBLOCK daily traffic/port.

By the end of the Baseline period, Shodan was reporting that SAMPLEBLOCK had 219 IPs with Port 80 open, 68 IPs reported the presence of Port 22 and 43 IPs reported Port 23, as shown in Figure 7.

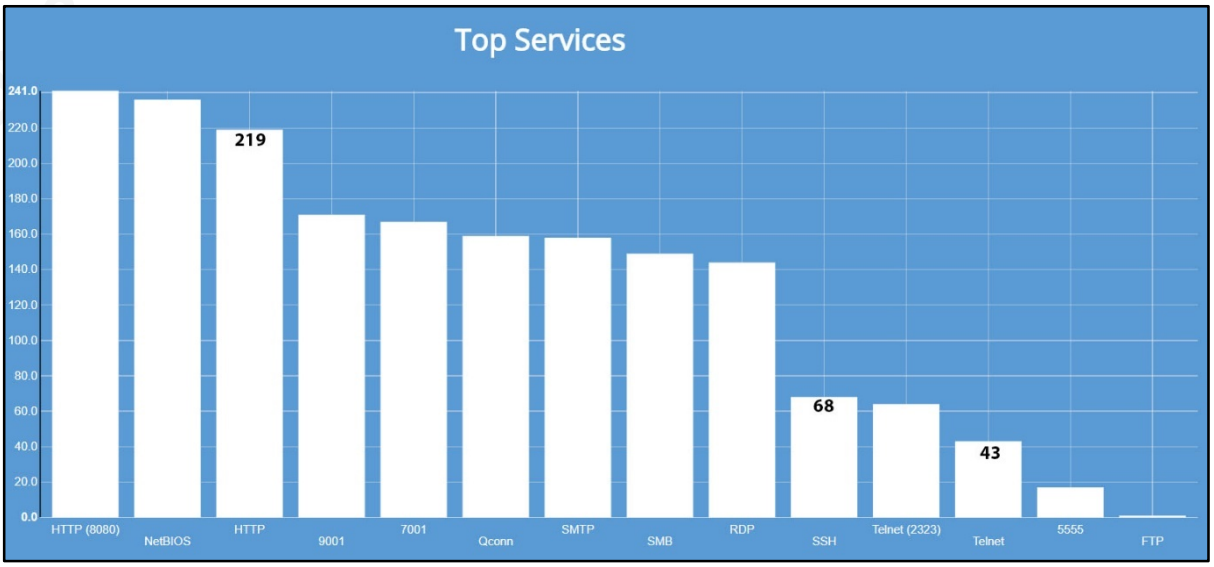


Figure 7: Shodan report run on 12 July, at the end of the Baseline. More IPs reported open ports, although the list was never complete during the experiment.

A count of Source IPs sending traffic to SAMPLEBLOCK was also maintained during the Baseline to compare with total numbers of Source IPs reported to the Internet Storm

Andre Shori, andreshori@gmail.com

Centre's Dshield Project. This count was done to see if the traffic volume recorded on SAMPLEBLOCK could be pegged to another measurement to qualify that any change in traffic happened after the implementation of the Shodan block. Source IPs were selected as the measure to compare as the Dshield project does not report traffic volume per port. If a strong correlation was seen between both sets of data, changes in traffic could be more easily attributed to the block and not merely to differences in overall attack traffic being seen on the broader internet. Unfortunately, as seen when comparing Figure 8 and Figure 9, a strong correlation between the two data sets could not be determined.

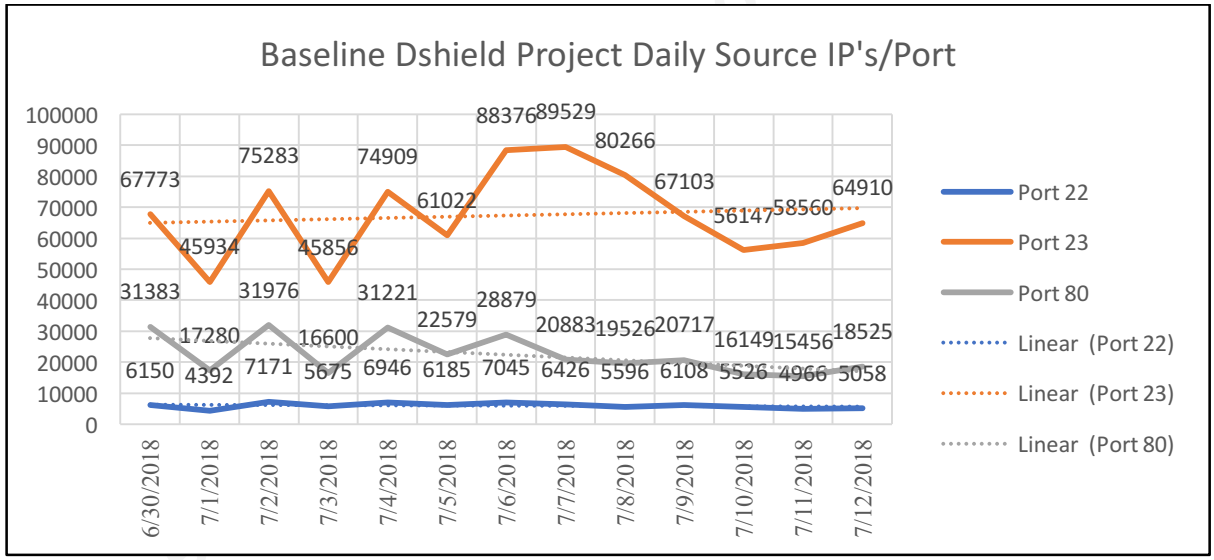


Figure 8: Dshield Project daily Source IPs per Port during the Baseline period

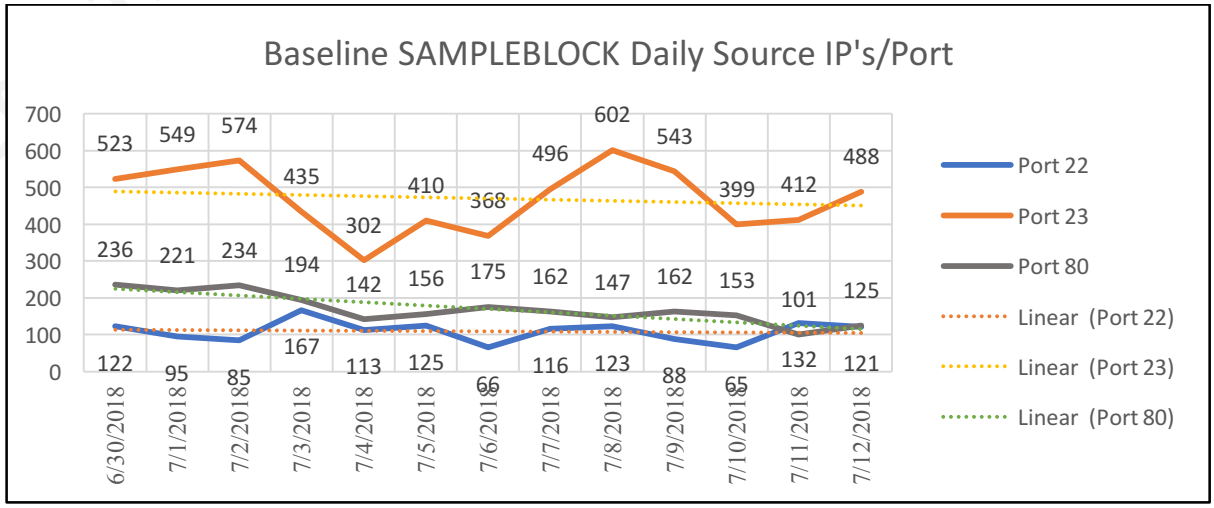


Figure 9: SAMPLEBLOCK daily Source IPs per Port during the Baseline Period

Correlation between the Dshield Project and SAMPLEBLOCK Source IP data were 0.15 for Port 22, 0.11 for Port 23 and 0.46 for Port 80, which is too weak to be used as a reliable gauge (a strong correlation would be between 0.5 and 1.0). However, Port 80 did show some promise but by the end of the experiment, it had dropped to 0.21. Given that HoneyHost

traffic volume was feeding into the Dshield Projects results, this was unsurprising as HoneyHost traffic was a small subset of the overall traffic as seen on the internet.

Incoming traffic to HoneyHost was also checked for the presence of IPs originating from the Shodan.io domain. Traffic originating from the Shodan.io domain was likely Shodan scans of HoneyHost and were targeted for our subsequent block. Traffic from the Shodan.io domain during the Baseline represented an average of only 0.92% of incoming traffic observed on HoneyHost, indicating that Shodan scans were likely only a small portion of daily traffic.

4. Phase Two: Shodan Block

Once the Baseline data was recorded, iptables was used to alter the firewall rules on HoneyHost to block Shodan scans (Shodan Block). Utilizing a pre-existing Shodan blocklist found on the Internet Storm Centre (SANS Institute, 2018), rules were added manually as a new chain to iptables and were then tested. A new log file named SHODANBLOCK was also created to log all blocked Shodan.io Source IPs to ensure the firewall was functional and to help with subsequent traffic analysis. The Shodan blocklist from the Internet Storm Centre was then monitored daily to ensure that any changes to the blocklist would be implemented on HoneyHost. No changes to the Internet Storm Centre blocklist were observed for the remainder of the experiment.

The firewall adjustment and monitoring period ran from 13 to 20 July. Traffic to HoneyHost was first checked using NSLOOKUP for the presence of any Source IPs originating from the Shodan.io domain that were not being blocked by the Internet Storm Centre Shodan blocklist. Several IPs were detected and then compared to an additional Shodan blocklist found on the website RomCheckFail (Hiltz, 2017). As many identified Shodan.io IPs were on the RomCheckFail list, iptables was manually updated again to include these additional Source IPs. It was also noted that neither the Internet Storm Centre's Shodan blocklist or the RomCheckFail blocklist (or any of the other examined free Shodan-focused blocklists) contained a complete list of all detected Shodan.io Source IPs found during the experiment. Checks of incoming traffic for any remaining Source IPs originating from the Shodan.io domain were repeated several times, and the few remaining Shodan.io Source IPs were then also added via iptables. Follow up checks of all HoneyHost Source IPs in DSHIELDLOG during Phase Three demonstrated that all Shodan.io IPs were now effectively blocked. Appendix A contains a complete list of the blocked Shodan.io IPs.

Andre Shori, andreshori@gmail.com

5. Phase Three: Shodan Block Results and Analysis

The experiment concluded on 31 July 2018. As DSHIELDLOG was found to log all incoming packets before the Shodan Block filter, during the impact period, entries containing Shodan.io Source IPs were manually subtracted from DSHIELDLOG for more accurate measurement of changes in traffic volume. Analysis of incoming traffic after implementation of Shodan Block revealed a -25.4% decrease in average Port 22 traffic, an increase of 19.8% in average Port 23 traffic and a nominal 4% increase in average Port 80 traffic volume when compared to the Baseline. Figure 10 shows SAMPLEBLOCK daily traffic volume per port after Shodan Block was implemented.

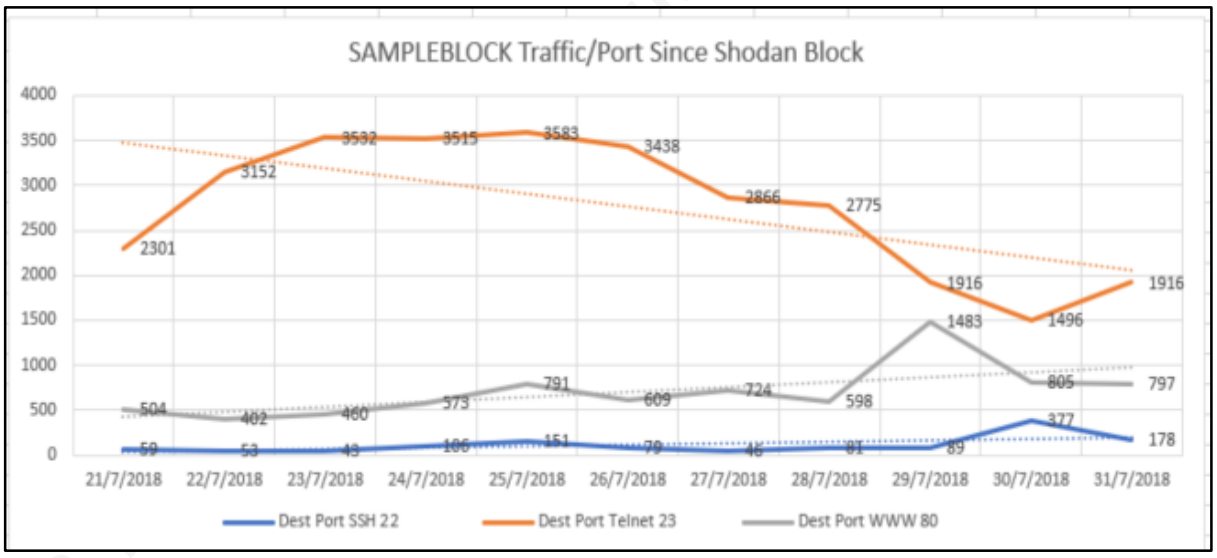


Figure 10: SAMPLEBLOCK Daily Traffic Volume/Port. Average Traffic for this period was 115 packets on Port 22, 2772 packets on Port 23 and 704 packets on Port 80.

However, without a strong correlation between Dshield and HoneyHost traffic, it was difficult to conclude if observed changes were just a result of overall changes in attack traffic volumes on the general internet. Over the entire duration of the experiment, the correlation between SAMPLEBLOCK traffic and the Dshield Project was only 0.04 for Port 22, -0.09 for Port 23 and 0.21 for Port 80. The lack of a reliable gauge to peg attack traffic/port against made it difficult to definitively derive any firm conclusions as to the effectiveness of the Shodan Block on reducing traffic volumes.

There was also evidence that the Shodan.io static IP filtering method employed during the experiment may have been insufficient in blocking 100% of all Shodan scans. While the Shodan Block method did slow down Shodan updates (25 SAMPLEBLOCK updates after the block was in place as compared to 231 updates in Shodan during the Baseline), Shodan Block still allowed some updates to occur. Checks on DSHIELDLOG for an hour before and after

each timestamp of these updates yielded no new Shodan.io IPs. In a further attempt to locate how Shodan added these entries, several manual scans were initiated and their times recorded. A similar analysis of DSHIELDLOG after each manual scan also yielded no additional Shodan.io IPs.

The continued presence of new SAMPLEBLOCK entries on Shodan after Shodan Block was implemented also suggested that Shodan may also be initiating scans from domains other than Shodan.io. Some of these IPs and related domains appear to be temporary and short-lived. Analysis of DSHIELDLOG revealed that some Source IPs from around the time of manual scans and updated Shodan entries were unresolvable shortly afterward.

As a result, Shodan may be using these techniques to bypass Source IP-based filtering mechanisms. The presence of unresolvable Source IPs suggests that Shodan may be utilizing mechanisms such as fast flux to mask their Source IPs, in combination with IPs originating from non-Shodan.io domains. By using a randomly changing pool of IPs, Shodan may also thwart many free and commercial blocklists as most of these blocklists are only updated daily.

It was also noted that even if a 100% effective Shodan scanner block could be crafted, entries on Shodan's database were found to persist for a long time. During the experiment, it was expected that entries without new updates would eventually "fall" off the Shodan database after one or two days. Analysis of a Shodan SAMPLEBLOCK report dated 6 August revealed the presence of an entry last updated on 6 July and the full range of all SAMPLEBLOCK IPs. Due to the limited timeframe available to run the experiment, exactly how long it would take for an entry to be removed from Shodan remained undetermined. The persistence of these entries on Shodan may also mean that results from attempts to block scans may take weeks or even months to appear. A more successful Shodan Block may require long-term implementation and monitoring of any blocking efforts.

Due to the limited run-time of the experiment and the unforeseen complications that occurred, results were ultimately deemed to be inconclusive as to whether blocking Shodan scans resulted in a decrease in attacker traffic volume.

 Andre Shori, andreshori@gmail.com

6. Lessons Learned

Overall, the experiment did present interesting and useful results. Even if traffic volume analysis after Shodan Block proved to be inconclusive, details uncovered about how Shodan conducts its scans and populates its database during the experiment may prove helpful when considering if it is worth the required effort and resources to block Shodan (and other internet-connected device search engines). Organizational defenders and managers need to factor in that implementing an effective block of Shodan and other internet-connected device search engines is a potentially complex task. Blocking Shodan proved to be more challenging than a simple IP blocklist and indicated that more resources might be required to prevent scanners from gaining any information about an organization's internet-connected devices. The slow updates on Shodan's database also suggested that these efforts might take several months to bear fruit.

6.1. Time Requirements

The limited availability for the experiment's run-time also impacted results. Running the Baseline over a more extended period, perhaps 60 days, may have more introduced more reliability. Detected SAMPLEBLOCK open ports continued to grow in number, and IPs there kept showing new open ports right up till the end of the experiment. Because Shodan limits manual scans of IPs to once every 24 hours, a more extended Baseline period was required to have eventually allowed for all 255 SAMPLEBLOCK IPs to register Port 22, 23 and 80 as "open." Registering all SAMPLEBLOCK IPs and ports as open on Shodan may have allowed attackers to target them across the entire sample, increasing the probability of attackers using Shodan as part of their reconnaissance and subsequent attack. Future experiments to determine the effectiveness of blocking Shodan (or other search engine) scanners would be prudent in factoring in a longer experimental timeframe to incorporate slow updates of these search engines databases.

Like the Baseline, a more extended Phase Three Impact period (or a more extended Phase Two firewall adjustment period that incorporated time needed for the Shodan database to update itself) could have yielded better results. More time may have been required to allow for previous SAMPLEBLOCK entries on Shodan to eventually fall off their database (although no entries were ever observed to fall off). Having SAMPLEBLOCK IPs continue to report open target ports, also meant that attackers using Shodan continued to see these ports as open throughout Phase Three (the Impact period). Had these ports begun to show as

 Andre Shori, andreshori@gmail.com

unavailable, it is plausible that attack traffic that uses Shodan for information may have decreased. Such a drop (or lack thereof) would have provided a better indication of the effectiveness of blocking Shodan in reducing HoneyHost's attack surface.

6.2. Different Logs

Further insight into attack traffic may have also allowed for better analysis of the impact of Shodan Block. Configuring the Honeypot appliances on HoneyHost as medium interaction Honeypots may have provided a more detailed analysis of attack types. For example, analysis of actual packet content destined for SAMPLEBLOCK may have allowed for differentiation between automated scans of the internet by attackers (based on their frequency, timestamps, and interaction with the Honeypot) and actual attacks by a human on the Honeypot appliance. Utilizing full packet capture was considered midway through the experiment, however this would have necessitated restarting it, which was not an option, given the time limit.

6.3. A Fresh Host

In hindsight, more robust crafting of the experiment environment may have provided better control over encountered variables. Acquiring access to a live Honeypot host was excellent for this experiment, and reduced setup time and complexity. This time reduction allowed the experiment to be run and concluded within the dictated timeframe but did introduce several factors that were difficult or impossible to compensate for. For example, if HoneyHost had started out as a brand-new host on the internet, total traffic volume could have been better analyzed for type and trend as they increased from zero, instead of using the pre-existing traffic that already existed. Pre-existing traffic did introduce additional variables that made impact conclusions more complex. For instance, Baseline traffic may have been inflated by ongoing attacks started before Phase One. Attack traffic pattern changes may also have been influenced by the existence of other HoneyHost Honeypots not included in the experiment scope.

Other pre-existing factors on HoneyHost constituted variables over which the experiment had little control. The necessary rate limit on HoneyHost's firewall to ensure that Cowrie did not crash; limited visibility into actual incoming traffic and further slowed down Shodan's ability to register all open target ports. More robust Honeypot appliances may have precluded the need for a rate limit and thus allowed for quicker Shodan database population and more log collection from HoneyHost for later analysis. Disk space limitations on HoneyHost also required standard logs instead of full packet capture. Full packet capture over the experiment

Andre Shori, andreshori@gmail.com

duration may have allowed for better analysis and understanding of incoming attack traffic, instead of being limited to studying overall traffic volume. These limitations and pre-existing conditions contributed to added complexity when analyzing results.

6.4. Pegging of Traffic Volume

More effective pegging of traffic volume to an online standard or data set of overall traffic volume may have provided a more conclusive Baseline and Impact analysis. Given the short timeframe in which this experiment was run, all attempts to adjust the traffic volume results in line with the Dshield Project records proved unsuccessful. Pegging the traffic volumes on HoneyHost to a different internet dataset on attack traffic volume may have been more effective since HoneyHost was sending traffic data to Dshield (Dshield results included the traffic data from HoneyHost). During the experiment, attempts to peg HoneyHost's traffic volumes to alternate internet attack traffic data such as those found on Statista or Akamai's Real Time Web Monitor were unsuccessful. Examination of these alternate sources yielded data that was unsuitable for correlation, only summary results or access to detailed results was limited to paid subscribers. Future experimenters may also wish to consider not sending their data to Dshield Project while running a similar experiment to keep things cleaner (or once concluded, to arrange to possibly transmit the data later). Researchers may then be better able to attribute changes to attack volumes on a Shodan Block.

Another possible method without having to peg traffic changes to another data set for more reliable measurement of the impact of a Shodan Block may have been to better control information that populated Shodan's database. By controlling the type of information that Shodan was able to see, better inferences on which attackers were using Shodan could have been made. For example, a suggested technique to consider for future experiments may be to use Honeypot appliances on non-standard ports (to ensure that Shodan did not make any assumptions on the service based on default port numbers) and allow Shodan to add these services to its database. Once these altered services appear on Shodan (and only Shodan), incoming traffic to these new ports could be inferred as the proportion of attack traffic that may not have existed if Shodan scans were successfully blocked. Any new traffic these ports would have had to originate from information purposefully captured on Shodan's database. This method would also likely require an effective Intrusion Detection System (IDS) to block all other non-Shodan search engine scans so that attackers could only get their information from Shodan. This method could have eliminated the requirement to peg traffic to overall

activity on the internet but due to run-time limits, the complex need to block all other search engine scans and lack of access to an IDS, it was not attempted.

6.5. Suggested Methods for More Effective Shodan Scanner Blocking

Given that Shodan scans appeared to be stealthier than initially anticipated, it is feasible that even with a substantial investment in technology and staff hours devoted to blocking these scans, it may not be possible always to block 100% of all Shodan scans. Neither of the free internet Shodan blocklists utilized during this experiment was found to completely list all Shodan.io IPs. A more effective block of Shodan scans may need to include a subscription to several online blocklists for adequate coverage. Also, it is also feasible that an Intrusion Detection System (IDS) would be beneficial to detect scanner behavior not caught by these blocklists and to dynamically update gateway firewalls to disrupt further scans. Additional investment in manhours and similar appliances to detect and block Shodan may be possible if blocking Shodan is a requirement of an organization's active defense.

One method that was observed to be somewhat effective at reducing the amount of information that Shodan was able to detect over the course of the experiment was the rate limit that was in place before the experiment started. However, rate limits also limit legitimate traffic and would be an inappropriate countermeasure on a non-Honeypot system.

7. Conclusion

Although this experiment's analysis of traffic volume proved to be inconclusive, the experiment did yield other, new interesting results as documented in this paper. Increased attack traffic volumes after initiating a block on Shodan scans; the continued updates on Shodan of SAMPLEBLOCK ports and IPs after the block; and the unexpectedly long persistence of Shodan database entries indicated that actively blocking Shodan and all other IoT or internet-connected device search engines could be more challenging to accomplish than initially estimated. Considerably more investment in manhours, appliances and time are likely required to block these types of scans.

Shodan is also not the only search engine in existence for internet-connected devices. Other IoT search engines like Censys and Zoomeye also offer similar functions. More of these types of search engines may emerge in the future and attempting to block them all would seem a large and complicated endeavor.

Blocking Shodan will do nothing if fundamental problems and security flaws exist within an organization's infrastructure. Shodan is primarily a tool and does not actually "hack" devices that it encounters (Shodan, 2018). The potential of a tool for a good or evil act is not a useful guidepost in deciding to ban it. Instead, it is far more useful to focus on and prevent nefarious users access. Shodan blocking may constitute "Security by Obscurity," a flawed approach to security in which one hopes that masking one's identity enables adequate protection without having to rectify underlying fundamental flaws in a defense. CSO Online Magazine makes a valid point that blocking Shodan is not a "magical fix" preventing exploitation and unauthorized access (Ragan, 2016). It is likely a far better investment to secure all devices within an organization and then use Shodan as part of a defender's toolkit to reconnoiter the organizational perimeter to see what the rest of the internet sees.

André Shori, andreshori@gmail.com

References

Akamai. (2018, August 11). Real-Time Internet Monitor | Akamai UK. Retrieved August 11, 2018, from https://www.akamai.com/uk/en/solutions/intelligent-platform/visualizing-akamai/real-time-web-monitor.jsp

Ayuso, P. N. (2014). *netfilter/iptables project homepage - The netfilter.org "iptables" project.* Retrieved from https://www.netfilter.org/projects/iptables/index.html

Checkpoint. (2017, March 29). *CHECK POINT THREAT ALERT: SHODAN*. Retrieved from https://blog.checkpoint.com/2016/01/04/check-point-threat-alert-shodan/

Ciphas. (2016, September 11). *5 Hacker-Friendly Search Engines You Must Use - Deep Dot Web [Web log post]*. Retrieved from https://www.deepdotweb.com/2016/09/11/5-hacker-friendly-search-engines-must-use/

Censys. (2018). *Censys- About Us*. Retrieved from https://censys.io/about

Gamblin, J. (2018, February 1). Blocks Shodan IPs From Scanning Your Servers. Retrieved August 23, 2018, from https://gist.github.com/jgamblin/2928d45730543fc7ef10cf56e5a980b0

Hiltz, M. (2017, December 7). *Blocking Shodan | Keeping shodan.io in the dark from scanning – RomCheckFail.* Retrieved from http://romcheckfail.com/blocking-shodan-keeping-shodan-io-in-the-dark-from-scanning/

Fernandes, L. (2016, March 29). *Shodan: The Hacker's Search Engine - Cybrary.* Retrieved June 5, 2018, from https://www.cybrary.it/0p3n/intro-shodan-search-engine-hackers/

SANS Institute. (n.d.). *About Us - Internet Security | DShield.* Retrieved from https://www.dshield.org/about.html

Shodan Command-Line Interface. (2018). Retrieved from https://cli.shodan.io/

Shodan. (2018, May 28). *What is Shodan? - Shodan Help Center*. Retrieved from https://help.shodan.io/the-basics/what-is-shodan

Statista. (2018, April). Web application attack traffic by country 2018 | Statistic. Retrieved August 11, 2018, from https://www.statista.com/statistics/276425/internet-attack-traffic-by-originating-country/

Goldman, D. (2013, April 8). *Shodan: The scariest Search Engine on the Internet*. Retrieved from http://money.cnn.com/2013/04/08/technology/security/shodan/index.html

Guarnizo, J. D., Tambe, A., Bhunia, S. S., Ochoa, M., Tippenhauer, N. O., Shabtai, A., & Elovici, Y. (2017). *SIPHON. Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security* - CPSS '17. doi:10.1145/3055186.3055192

Ragan, S. (2016, January 8). *Blocking Shodan isn't some sort of magical fix that will protect your data.* Retrieved from https://www.csoonline.com/article/3020108/techology-business/blocking-shodan-isnt-some-sort-of-magical-fix-that-will-protect-your-data.html

Shodan. (2018, May 28). *What is Shodan? - Shodan Help Center*. Retrieved June 4, 2018, from https://help.shodan.io/the-basics/what-is-shodan

SANS Institute. (2018). *threatlist*. Retrieved from https://isc.sans.edu/api/threatlist/shodan/

Townsend, K. (2018, February 1). *AutoSploit: Automated Hacking Tool Set to Wreak Havoc or a Tempest in a Teapot?* | SecurityWeek.Com. Retrieved from https://www.securityweek.com/autosploit-automated-hacking-tool-set-wreak-havoc-or-tempest-teapot

Osborne, C. (2016, January 26*). Shodan: The IoT Search Engine for watching sleeping kids and bedroom antics | ZDNet*. Retrieved from https://www.zdnet.com/article/shodan-the-iot-search-engine-which-shows-us-sleeping-kids-and-how-we-throw-away-our-privacy/

[Question] Blocking Shodan.io?. (2018, April). r/PFSENSE - [Question] Blocking Shodan.io? Retrieved August 23, 2018, from https://www.reddit.com/r/PFSENSE/comments/87s5ew/question_blocking_shodanio/

Zoomeye. (2018, September). *What's Zoomeye?* Retrieved from https://www.zoomeye.org/about

Appendix A – iptables Shodan Blocklist

Chain SHODANDROP (1 references)

target	prot	opt	source	destination
DROP	all	--	66.240.192.138	0.0.0.0/0
DROP	all	--	66.240.205.34	0.0.0.0/0
DROP	all	--	66.240.236.119	0.0.0.0/0
DROP	all	--	71.6.135.131	0.0.0.0/0
DROP	all	--	71.6.158.166	0.0.0.0/0
DROP	all	--	71.6.165.200	0.0.0.0/0
DROP	all	--	71.6.167.142	0.0.0.0/0
DROP	all	--	80.82.77.33	0.0.0.0/0
DROP	all	--	80.82.77.139	0.0.0.0/0
DROP	all	--	82.221.105.6	0.0.0.0/0
DROP	all	--	82.221.105.7	0.0.0.0/0
DROP	all	--	85.25.43.94	0.0.0.0/0
DROP	all	--	85.25.103.50	0.0.0.0/0
DROP	all	--	89.248.167.131	0.0.0.0/0
DROP	all	--	89.248.172.16	0.0.0.0/0
DROP	all	--	93.120.27.62	0.0.0.0/0
DROP	all	--	93.174.95.106	0.0.0.0/0
DROP	all	--	94.102.49.190	0.0.0.0/0
DROP	all	--	94.102.49.193	0.0.0.0/0
DROP	all	--	188.138.9.50	0.0.0.0/0
DROP	all	--	198.20.69.74	0.0.0.0/0
DROP	all	--	198.20.69.98	0.0.0.0/0
DROP	all	--	198.20.70.114	0.0.0.0/0
DROP	all	--	198.20.99.130	0.0.0.0/0
DROP	all	--	216.117.2.180	0.0.0.0/0
DROP	all	--	66.240.219.146	0.0.0.0/0
DROP	all	--	71.6.135.131	0.0.0.0/0

```
DROP       all  --  198.20.69.98         0.0.0.0/0
DROP       all  --  80.82.77.139         0.0.0.0/0
DROP       all  --  80.82.77.33          0.0.0.0/0
DROP       all  --  71.6.165.200         0.0.0.0/0
DROP       all  --  198.20.70.114        0.0.0.0/0
DROP       all  --  93.174.95.106        0.0.0.0/0
DROP       all  --  66.240.192.138       0.0.0.0/0
DROP       all  --  94.102.49.190        0.0.0.0/0
DROP       all  --  89.248.172.16        0.0.0.0/0
DROP       all  --  82.221.105.6         0.0.0.0/0
DROP       all  --  71.6.167.142         0.0.0.0/0
DROP       all  --  71.6.158.166         0.0.0.0/0
DROP       all  --  89.248.167.131       0.0.0.0/0
DROP       all  --  66.240.236.119       0.0.0.0/0
DROP       all  --  66.240.205.34        0.0.0.0/0
DROP       all  --  71.6.146.130         0.0.0.0/0
DROP       all  --  185.181.102.18       0.0.0.0/0
DROP       all  --  198.20.69.0/24       0.0.0.0/0
DROP       all  --  71.6.146.0/24        0.0.0.0/0
DROP       all  --  198.20.87.0/24       0.0.0.0/0
DROP       all  --  209.126.110.38       0.0.0.0/0
DROP       all  --  104.236.198.48       0.0.0.0/0
DROP       all  --  104.131.0.69         0.0.0.0/0
DROP       all  --  162.159.244.38       0.0.0.0/0
DROP       all  --  159.203.176.62       0.0.0.0/0
DROP       all  --  188.138.1.119        0.0.0.0/0
DROP       all  --  71.6.146.130         0.0.0.0/0
DROP       all  --  185.163.109.66       0.0.0.0/0
DROP       all  --  198.20.70.0/24       0.0.0.0/0
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
```


 Andre Shori, andreshori@gmail.com

Last Updated: November 30th, 2018

Upcoming SANS Training

Click here to view a list of all SANS Courses

SANS Nashville 2018	Nashville, TNUS	Dec 03, 2018 - Dec 08, 2018	Live Event
SANS Santa Monica 2018	Santa Monica, CAUS	Dec 03, 2018 - Dec 08, 2018	Live Event
SANS Dublin 2018	Dublin, IE	Dec 03, 2018 - Dec 08, 2018	Live Event
Tactical Detection & Data Analytics Summit & Training 2018	Scottsdale, AZUS	Dec 04, 2018 - Dec 11, 2018	Live Event
SANS Frankfurt 2018	Frankfurt, DE	Dec 10, 2018 - Dec 15, 2018	Live Event
SANS Cyber Defense Initiative 2018	Washington, DCUS	Dec 11, 2018 - Dec 18, 2018	Live Event
SANS Bangalore January 2019	Bangalore, IN	Jan 07, 2019 - Jan 19, 2019	Live Event
SANS Sonoma 2019	Santa Rosa, CAUS	Jan 14, 2019 - Jan 19, 2019	Live Event
SANS Amsterdam January 2019	Amsterdam, NL	Jan 14, 2019 - Jan 19, 2019	Live Event
SANS Threat Hunting London 2019	London, GB	Jan 14, 2019 - Jan 19, 2019	Live Event
Cyber Threat Intelligence Summit & Training 2019	Arlington, VAUS	Jan 21, 2019 - Jan 28, 2019	Live Event
SANS Miami 2019	Miami, FLUS	Jan 21, 2019 - Jan 26, 2019	Live Event
SANS Dubai January 2019	Dubai, AE	Jan 26, 2019 - Jan 31, 2019	Live Event
SANS Las Vegas 2019	Las Vegas, NVUS	Jan 28, 2019 - Feb 02, 2019	Live Event
SANS Security East 2019	New Orleans, LAUS	Feb 02, 2019 - Feb 09, 2019	Live Event
SANS SEC504 Stuttgart 2019 (In English)	Stuttgart, DE	Feb 04, 2019 - Feb 09, 2019	Live Event
SANS Northern VA Spring- Tysons 2019	Vienna, VAUS	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS London February 2019	London, GB	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS Anaheim 2019	Anaheim, CAUS	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS Secure Japan 2019	Tokyo, JP	Feb 18, 2019 - Mar 02, 2019	Live Event
SANS Scottsdale 2019	Scottsdale, AZUS	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS New York Metro Winter 2019	Jersey City, NJUS	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS Dallas 2019	Dallas, TXUS	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS Zurich February 2019	Zurich, CH	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS Riyadh February 2019	Riyadh, SA	Feb 23, 2019 - Feb 28, 2019	Live Event
SANS Reno Tahoe 2019	Reno, NVUS	Feb 25, 2019 - Mar 02, 2019	Live Event
Open-Source Intelligence Summit & Training 2019	Alexandria, VAUS	Feb 25, 2019 - Mar 03, 2019	Live Event
SANS Brussels February 2019	Brussels, BE	Feb 25, 2019 - Mar 02, 2019	Live Event
SANS Khobar 2018	OnlineSA	Dec 01, 2018 - Dec 06, 2018	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced