Defense in Depth for a Small Office/Home Office

GIAC GCIH Gold

Author: Gregory Melton, Gregory.Melton@student.sans.edu
Advisor: Clay Risenhoover

Accepted: November 15th, 2019

Abstract

Much attention is given to enterprise security with expensive solutions and teams of both IT and security personnel, but the home office may only ever be proactively defended by a single amateur or hobbyist. Large scale corporate solutions may deal with Advanced Persistent Threats (APTs) and corporate espionage, but there are far fewer solutions to home office threats. This paper focuses on best practices for a home network running minimal servers to protect from casual browsing and careless home users. This research intends to demonstrate meaningful defense of endpoints in a local network by drastically reducing potential communication to C2 nodes and data exfiltration with proper filtering and minimal extra hardware.

Introduction

The Small/Home Office is an often-overlooked demographic when compared to enterprise security in today's information security market. One 2017 report found at least 2.8% of people in the U.S. workforce work from home at least half the time (Shepherd 2019). A more staggering statistic claims that as much as 70% of working professionals work remotely at least one day a week (Browne, 2018). While there are a substantial number of people working from home, there is significantly less money to be made by industry products targeting home users that lack compliance mandates and information security budgets. Furthermore, very little is marketed to such people because it is more cost-effective as an organization to sell large scale solutions with subscription fees.

The largest threat of organizational breach occurs at the endpoint level (Murray 2019). Endpoints, due to user interaction, are widely seen as the primary weak point in any network (Lord 2018). A multi-billion dollar corporation and a simple home network set-up have this weak point in common. To address such a deficiency, this paper will look to prove that a Small/Home office can achieve significantly heightened security for very little time and money invested. Hardening a small network by running a local DNS server combined with some simple-to-implement filtering can defeat a wide variety of commonly encountered endpoint threats.

Furthermore, the steps taken to ensure a hardened security posture will seek to minimize damage from interactive threats should an endpoint be compromised. To achieve a baseline, compromise of a test network will be assumed to verify the effectiveness of the proposed network and host implementations. In short, the goal after compromise is to minimize or stop data exfiltration and interactive connections while logging evidence for follow on analysis and situational awareness.

The solution should be simple, affordable, and maintainable with readily available resources and open source products. By setting up a local DNS server and logging connections, a user can significantly improve network and endpoint security without paying for expensive subscriptions or dealing with third-party security solutions. The proposed layering effect is key to an effective security posture (Horton 2012).

1. SOHO discussion

A Small Office/Home Office (or single office/home office; SOHO) typically refers to the category of business that employs anywhere from 1 to 10 workers. In New Zealand, for example, the Ministry of Business, Innovation, and Employment (MBIE) defines a small office as 6 to 19 employees and a micro office as 1 to 5.

For this research, a SOHO is defined as an organization with less than ten workers or a home network (2018 US Legal Inc). Both a small office and home network would include a Gateway Router, Primary workstation (desktop or laptop), and mobile devices. The test environment for this research will represent a 'typical' home office network. The defensive measures proposed should work for up to 10 devices with a primary work station computer as the focus.

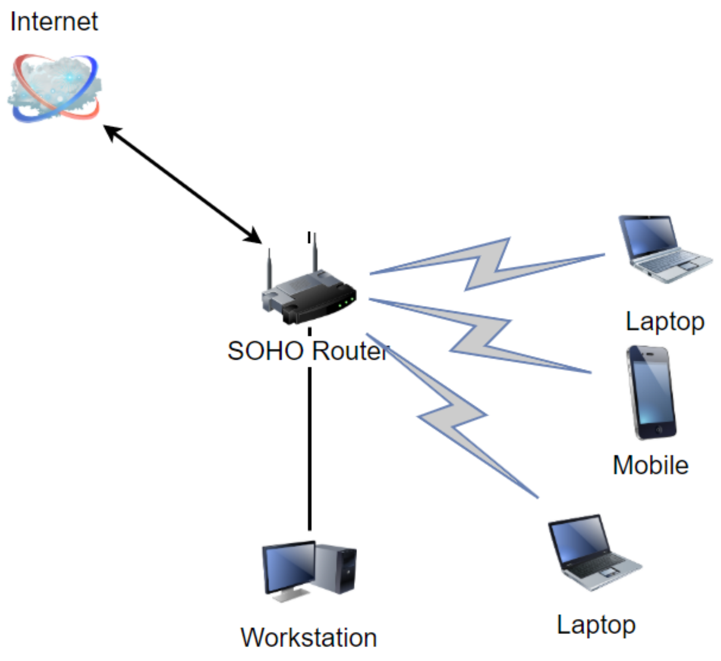


Figure 1. SOHO Example

1.1 Threats to SOHOs

A Small/Home office shares a major point of risk with that of an enterprise. The end-users in both environments are often seen as low hanging fruit and low barriers to entry for both cybersecurity professionals and cybercriminals alike. End-user systems are notoriously difficult to defend due to human interaction, which can often negate technical controls put in place by security measures. Organizations try to mitigate this risk through compliance requirements, mandatory training, and periodic phishing email simulations to gauge effectiveness. Both technical controls and user training are common defenses against such attacks, but since a small/home office lacks this training, it remains a weak point.

On a positive note, in a typical home environment, there is a smaller attack surface. Most home networks are not running a bunch of servers, which cuts down the number of devices and services to exploit from the outside. The downside may be a less-aware user population that may give little thought to system security or clicking malicious links.

To achieve a more secure operating posture in light of human error, a small/home office can implement and maintain appropriate filtering through blacklists at the domain level. Blacklisting will counteract known malicious domains that are tracked by the information security community and blocklist maintainers.

2 DNS

To harden network security, a network owner must have at least a vague understanding of how networks function and how the internet works. One of the most important protocols used to run the modern internet is the Domain Name Service (DNS). When a user wants to load a webpage, a translation must occur between what a user types into their web browser and the machine-friendly address necessary to locate a webpage

(How DNS works n.d.). The details of a DNS request are beyond the scope of this paper, but a basic example is shown below.

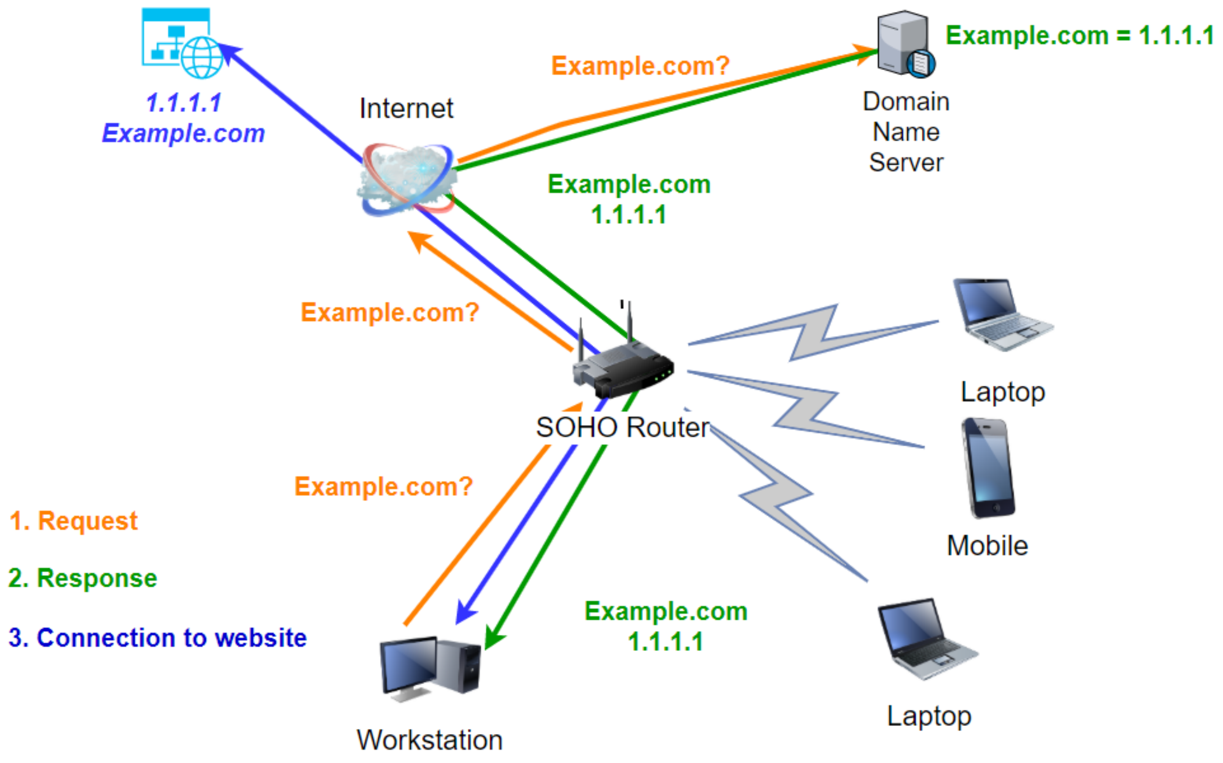


Figure 2. Basic Domain Name Resolution

Running a user-controlled local DNS server can mitigate many threats to a network's endpoints. A local DNS server is a logical choke point to intercept or block malicious content as it is responsible for all the URL to IP mappings in a network. By filtering out known malicious domains, a small office owner can defeat many potential threats.

For example, in a household with children, one can assume a system will be subject to rampant browsing. Children or casual users are not likely to have a security mindset or understand the dangers of the open internet. Running a self-updating blacklist via some of the proposed GitHub projects will help prevent many sites from being loaded.

At one point in the not too distant past, it would be a potentially arduous task to set up and maintain a local DNS server. Advances in modern hardware, software, and user-friendliness have made owning a local server inexpensive and easy. Before purchasing any equipment, a user can first try a security-conscious DNS solution to see an immediate benefit.

2.1 QUAD9

A DNS solution that does not require hardware, set up, or maintenance is the "QUAD9" DNS service. QUAD9 is a freely available nonprofit organization that provides a public DNS service and will block known malicious domains, as discussed previously. QUAD9 is a good option that is easy to set up but lacks the control and monitoring this paper hopes to achieve through a local DNS service. If, for example, a user notices "advertisements.com" showing up in their social media feed, they would not have the ability to add that domain to the QUAD9 blacklist. In this case, the browser will continue to load the website as advertised. A safer and more complete protection would be to run a local DNS server, mirror good blocklists, and fine-tune a blacklist to specific needs.

Another potential setback for relying on the QUAD9 service is the inability to control a white list. If a desired website is on the QUAD9 blacklist, it would be inaccessible. Conversely, if a particular advertiser is supported by a user, and the QUAD9 service reduces their exposure, a user would not have the ability to white-list said advertiser. To get the most out of DNS protections, to include fine-tuning of white and blacklists, a user can run their own DNS sinkhole in the form of a project called "Pi-Hole".

2.2 Pi-hole

The Rasberry Pi is the name for a series of single-board computers made by the Raspberry Pi Foundation. The Raspberry Pi Foundation is a UK charity that aims to educate people in computing and create easier access to computing education (What is a Raspberry Pi, n.d.). The Raspberry Pi is a very inexpensive computer that runs Linux and is easy to use thanks to abundant and free projects available on the internet. One such project is a DNS server called the Pi-hole. A Pi-hole DNS server is an easy to use server that provides a web interface allowing users to easily filter internet content via blocklists, blacklists, and "regular expressions" (REGEX). The Pi-hole software is free, actively maintained and is the chosen DNS software for the remainder of this research.

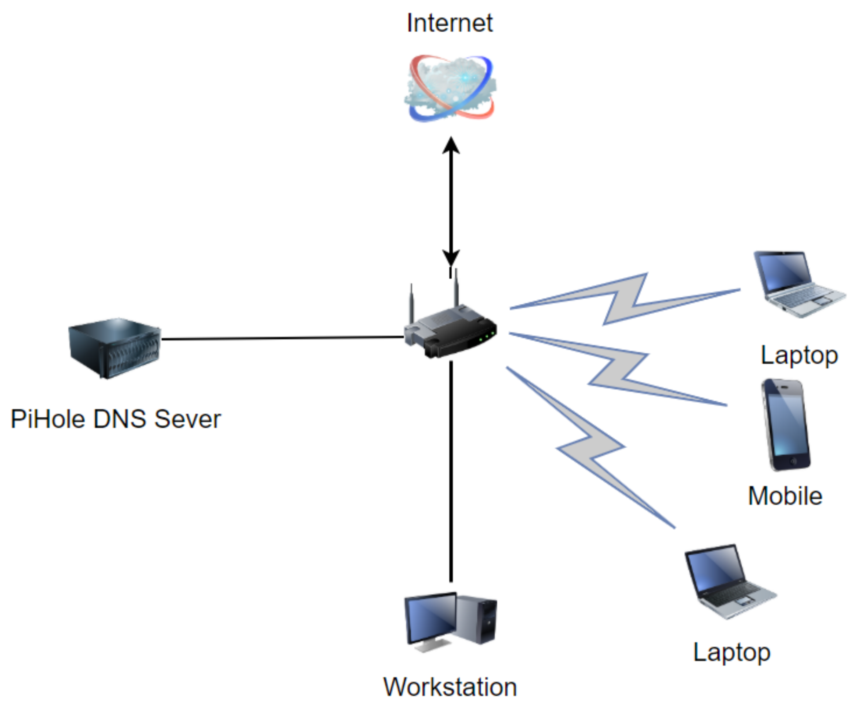
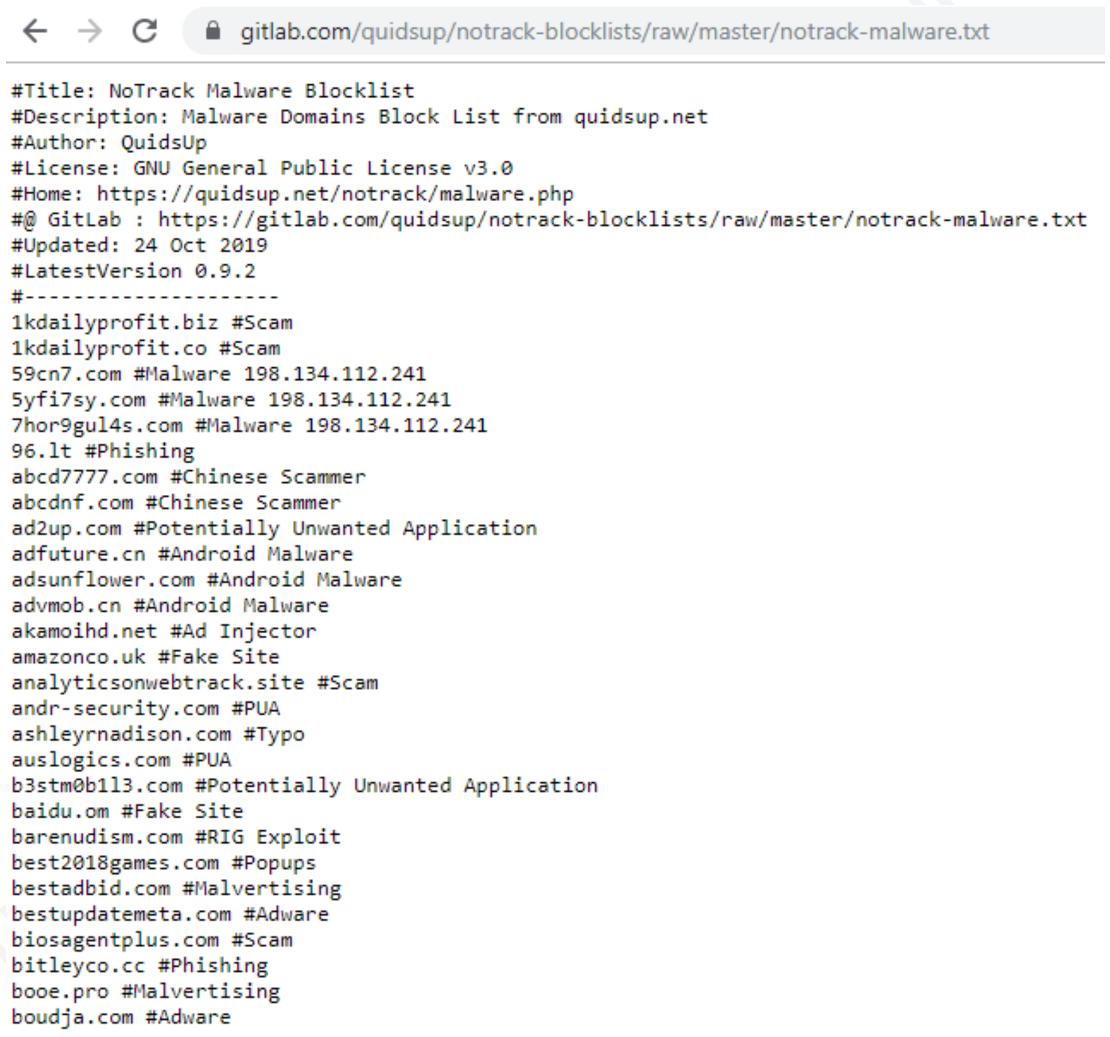


Figure 3: Pi-Hole DNS incorporated into a small network

The Pi-Hole's rise to popularity was based on its effective blacklisting/blocklisting techniques, with a primary function of blocking overwhelming advertisements. The Pi-Hole will, by default, update chosen blocklists (a blacklist maintained by others) every week or can be configured to update at specific, chosen intervals. A simple google search for "Pi-Hole blocklists" is enough to find a variety of blocklists and will show how abundantly available they are. While the original intent may have been to thwart unwanted advertisements, the closely related benefit to a DNS sinkhole is blocking established malware domains in the same fashion (Bruneau 2010). There are abundant blocklists available, and some are written with defense against malicious intent in mind. The best blocklists are actively maintained and are a key ingredient to the first layer of network defense. One of the best repositories of blocklists can be found at The Firebog website Firebog.net which is maintained by Wally3k.

Recommended Blocklists

- https://gitlab.com/quidsup/notrack-blocklists/raw/master/notrack-malware.txt
- https://www.stopforumspam.com/downloads/toxic_domains_whole.txt
- https://phishing.army/download/phishing_army_blocklist_extended.txt

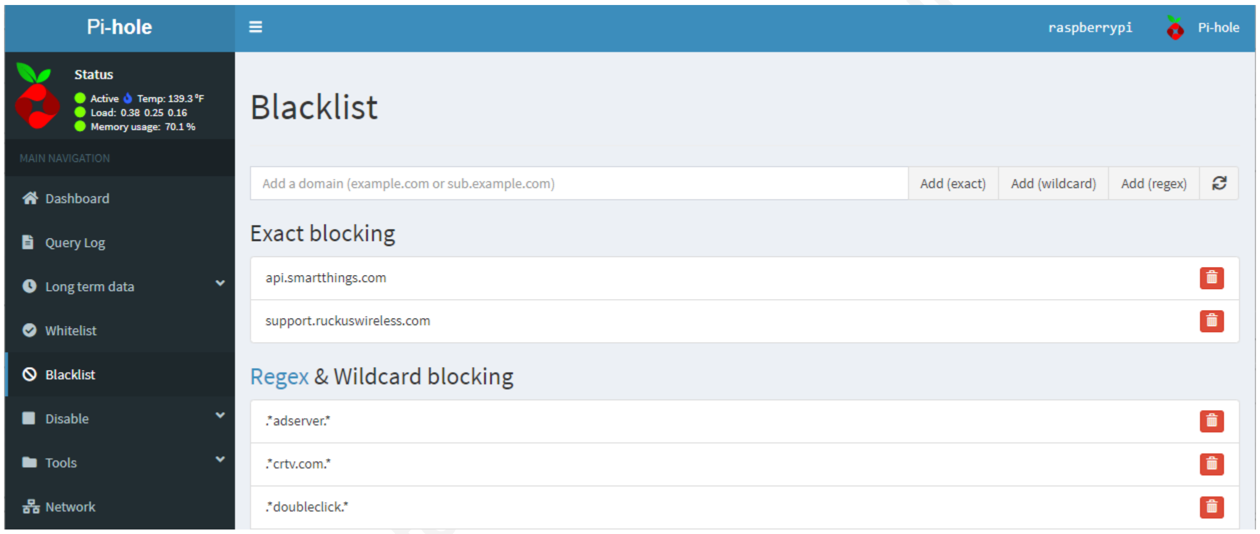


gitlab.com/quidsup/notrack-blocklists/raw/master/notrack-malware.txt

```
#Title: NoTrack Malware Blocklist
#Description: Malware Domains Block List from quidsup.net
#Author: QuidsUp
#License: GNU General Public License v3.0
#Home: https://quidsup.net/notrack/malware.php
#@ GitLab : https://gitlab.com/quidsup/notrack-blocklists/raw/master/notrack-malware.txt
#Updated: 24 Oct 2019
#LatestVersion 0.9.2
#---------------------
1kdailyprofit.biz #Scam
1kdailyprofit.co #Scam
59cn7.com #Malware 198.134.112.241
5yfi7sy.com #Malware 198.134.112.241
7hor9gul4s.com #Malware 198.134.112.241
96.lt #Phishing
abcd7777.com #Chinese Scammer
abcdnf.com #Chinese Scammer
ad2up.com #Potentially Unwanted Application
adfuture.cn #Android Malware
adsunflower.com #Android Malware
advmob.cn #Android Malware
akamoihd.net #Ad Injector
amazonco.uk #Fake Site
analyticsonwebtrack.site #Scam
andr-security.com #PUA
ashleyrnadison.com #Typo
auslogics.com #PUA
b3stm0b1l3.com #Potentially Unwanted Application
baidu.om #Fake Site
barenudism.com #RIG Exploit
best2018games.com #Popups
bestadbid.com #Malvertising
bestupdatemeta.com #Adware
biosagentplus.com #Scam
bitleyco.cc #Phishing
booe.pro #Malvertising
boudja.com #Adware
```

Figure 4. Example of blocklist contents (quidsup)

Regular Expressions are a sequence of characters that define a search pattern. They can also be used to give a user more flexibility when all permutations of a site should be blocked. For example, if a user wanted to block all varieties of the “adserver” domain names, they could add .*adserver.* to the regex enabled blacklist. A “.*” sequence means “any number of any character combination,” and in this situation, it will block anything with the string “adserver” in any part of the domain name.



Blacklist

Add a domain (example.com or sub.example.com)

Exact blocking

api.smartthings.com

support.ruckuswireless.com

Regex & Wildcard blocking

.*adserver.*

.*crtv.com.*

.*doubleclick.*

Figure 5. Pi-hole Regular Expressions

2.3 Blocking Direct IP calls as a theory

Some malware, however, may be coded with a direct connection to the C2 node's IP address. A hard-coded IP address would effectively bypass the DNS server since no name resolution is requested and, therefore, would not be present in the DNS logs. The short answer to this challenge is to disallow direct IP calls to external networks. There are very few reasons an endpoint for a non-technical person would make connections to external IP addresses without first logging in the DNS server. The difficulty lies in how to check connections against the DNS cache effectively. Even if a network device could compare connections to DNS results, it would require modification not to break network functionality of legitimate outbound connections. It is important to know which legitimate programs are making direct IPs connections behind the scenes. Windows updates, video games and other applications might make direct IP connections without the use of DNS. As such, a good IP white list would need to be implemented and understood. A white list in this circumstance is a list of IPs that are permitted connections, even if they are not present in a DNS entry.

One challenge of this approach is getting the necessary data into the same location and vetting network connections against this data. A higher-level scripting language could manage the tasks necessary to vet connections, but network latency would be a problem. Current servers and systems are not, by default, set up to keep track of connection states as described previously. Therefore, this research will work towards prevention by tracking network connections in conjunction with DNS query logs at the system level. Enabling such visibility is the first step towards a layered security posture.

3 Sysmon to the rescue

It may sound trivial at first to create a list of DNS requests to compare against a separate list of network connections. Assuming a user runs their own DNS server, they have a log of requests at the server. Unfortunately, it is difficult to then compare DNS logs to network connections by endpoints at the network level. To maintain scope for this research, it will not be possible to proactively block direct IP connections. Instead, Sysmon will be invoked in conjunction with powershell/python scripts to achieve periodic alerts and first-tier triage.

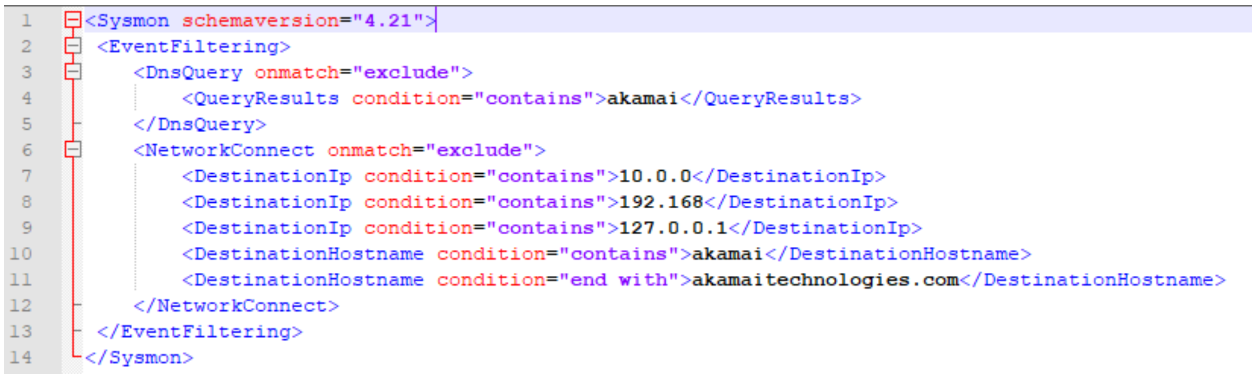
To discover when a system communicates directly to an external resource without making a DNS request, an administrator must enable robust logging. To achieve this goal, it is necessary to install and configure the System Monitor (Sysmon) services on Windows systems.

> Sysmon is a Windows system service and device driver that, once installed on a system, remains resident across system reboots to monitor and log system activity to the Windows event log. It provides detailed information about process creations, network connections, and changes to file creation time. By collecting the events it generates using Windows Event Collection or SIEM agents and subsequently analyzing them, someone can identify malicious or anomalous activity and understand how intruders and malware operate on the network. (Russinovich 2019)

Sysmon, as a windows logging suite, is incredibly powerful and relatively easy to set up. Once installed and configured, the resulting logs can be formatted, parsed and used to alert on 'odd' behavior such as direct IP connections from the endpoint. As of June 2019, SYSMON is capable of logging DNS requests and responses, which is exactly what is needed to check against the network connections, which are also logged by SYSMON. Powershell and python can be set to periodically parse the SYSMON logs, convert them to a CSV file format, and output a list anytime the system connects to an IP address that was never among the DNS logs.

Such a process effectively simulates and automates a CIRT methodology searching out suspicious behavior. While this won't actively stop direct IP calls, it can automate and alert when that sort of event takes place. This Indicator of Compromise (IOC) will need filtering in the Sysmon XML configuration to cut down on the noise of internal network connections and system routines.

The first step is to download and configure Sysmon. Sysmon can be configured upon initial installation or re-configured with an XML document. By default, Sysmon does not log DNS traffic or network connections on installation. The rationale for not logging these activities by default is due to the amount of traffic it would generate. Therefore, an administrator must explicitly enable these functions with a configuration file. To create an appropriate configuration file, it would be necessary to understand the logging intent. The primary purpose of this Proof of Concept (POC) is to determine if the system is making direct IP connections to an external network. To cut down on network noise, the following configuration file will exclude logging for private IP ranges, local loopback, and the string "akamai" if present in the DNS query or the destination host. "Akamai" was excluded only for this POC and would be a glaring security risk in a live network.



```
<Sysmon schemaversion="4.21">
 <EventFiltering>
    <DnsQuery onmatch="exclude">
        <QueryResults condition="contains">akamai</QueryResults>
    </DnsQuery>
    <NetworkConnect onmatch="exclude">
        <DestinationIp condition="contains">10.0.0</DestinationIp>
        <DestinationIp condition="contains">192.168</DestinationIp>
        <DestinationIp condition="contains">127.0.0.1</DestinationIp>
        <DestinationHostname condition="contains">akamai</DestinationHostname>
        <DestinationHostname condition="end with">akamaitechnologies.com</DestinationHostname>
    </NetworkConnect>
 </EventFiltering>
</Sysmon>
```

Appendix A

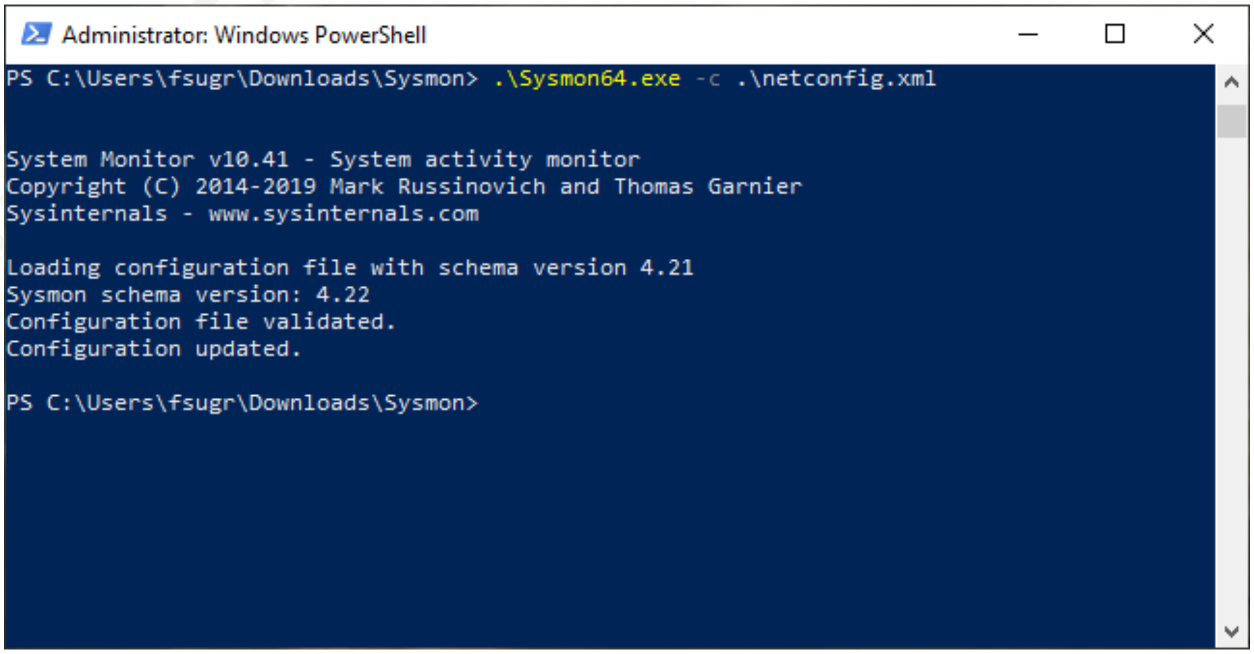
Appendix A (above) is the complete configuration file for a Sysmon set-up. Declaring "*exclude*" sections will log everything not explicitly defined in that code block (Stevens 2019). For example, the code above excludes logging DNS queries with the word "akamai" in the results. Since "akamai" is the one exclusion, every other DNS result on the system will be logged. To install or update a running Sysmon configuration, an administrator should run the following commands.

Initial Install Syntax Example:

```
Sysmon64.exe -i C:\Users\path\to\SysmonConfigurationFile.xml -accepteula
```

Update configuration file if already running Symon Example:

```
Sysmon64.exe -c C:\Users\path\to\SysmonConfigurationFile.xml
```



```
PS C:\Users\fsugr\Downloads\Sysmon> .\Sysmon64.exe -c .\netconfig.xml


System Monitor v10.41 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.21
Sysmon schema version: 4.22
Configuration file validated.
Configuration updated.

PS C:\Users\fsugr\Downloads\Sysmon>
```

Figure 6. Sysmon Configuration Update

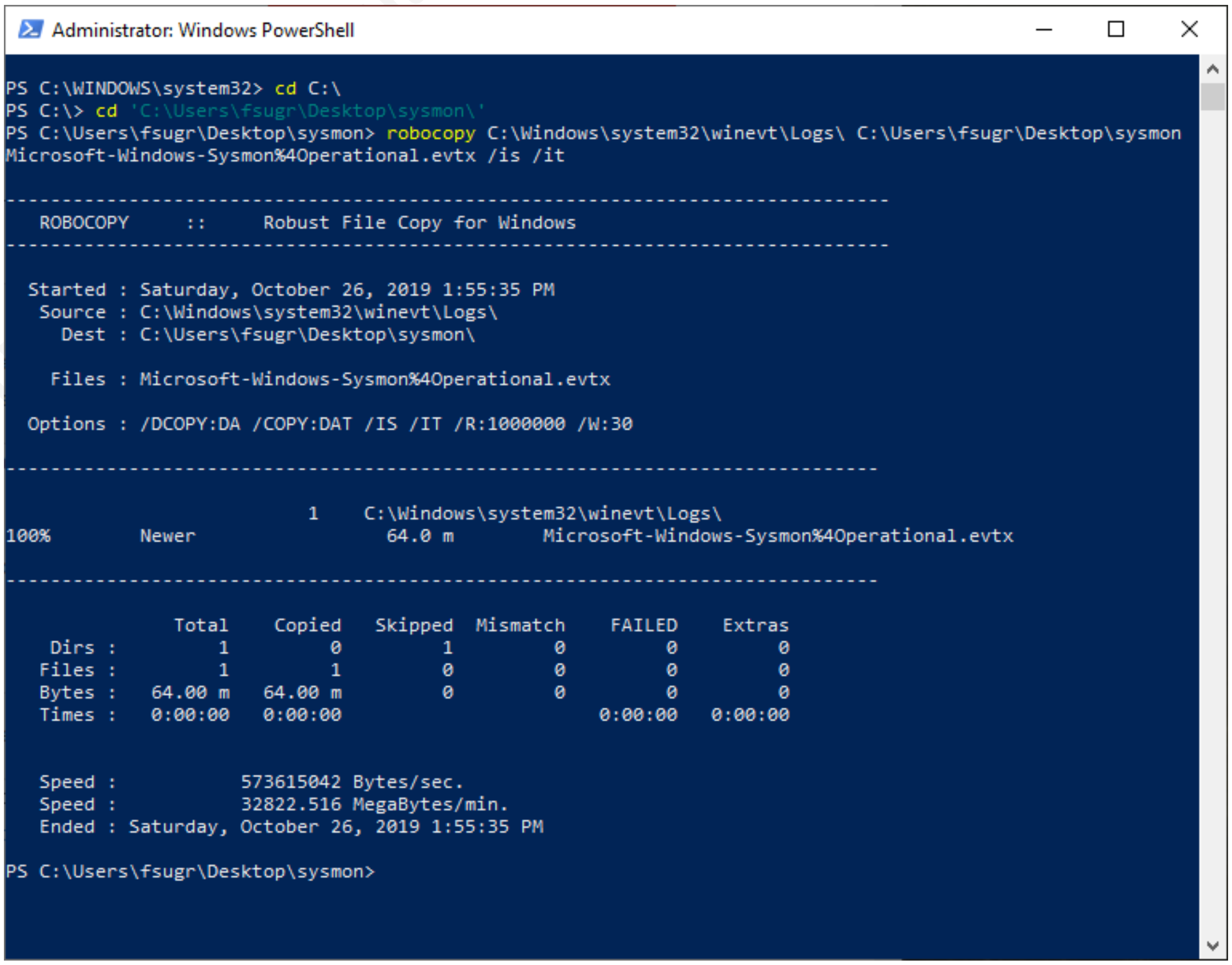
An excellent CrowdStrike blog by Matt Churchill (2015) explains how once installed, SYSMON records everything to a standard Windows event log. On a Windows 7 system and above, this file is located here:

C:\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%4Operational.evtx

The problem with this *evtx* file is that it dynamically logs and is not easily manipulated or parsed outside of the Microsoft Event Viewer. To circumvent formatting issues, it is helpful to copy the log file and convert it to something more workable such as a CSV (Churchill 2015). Windows Powershell can be used to both copy and convert a file as needed.

Make a Copy of the Log File:

```
robocopy C:\Windows\system32\winevt\Logs\ C:\Users\path\to\store\copied\file\Sysmon Microsoft-Windows-Sysmon%4Operational.evtx /is /it
```



```
PS C:\WINDOWS\system32> cd C:\
PS C:\> cd 'C:\Users\fsugr\Desktop\sysmon\'
PS C:\Users\fsugr\Desktop\sysmon> robocopy C:\Windows\system32\winevt\Logs\ C:\Users\fsugr\Desktop\sysmon
Microsoft-Windows-Sysmon%4Operational.evtx /is /it

-------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
-------------------------------------------------------------------------------

  Started : Saturday, October 26, 2019 1:55:35 PM
   Source : C:\Windows\system32\winevt\Logs\
     Dest : C:\Users\fsugr\Desktop\sysmon\

    Files : Microsoft-Windows-Sysmon%4Operational.evtx

  Options : /DCOPY:DA /COPY:DAT /IS /IT /R:1000000 /W:30

------------------------------------------------------------------------------

                           1    C:\Windows\system32\winevt\Logs\
100%        Newer                 64.0 m        Microsoft-Windows-Sysmon%4Operational.evtx

------------------------------------------------------------------------------

               Total    Copied   Skipped  Mismatch    FAILED    Extras
    Dirs :         1         0         1         0         0         0
   Files :         1         1         0         0         0         0
   Bytes :   64.00 m   64.00 m         0         0         0         0
   Times :   0:00:00   0:00:00                       0:00:00   0:00:00


   Speed :           573615042 Bytes/sec.
   Speed :           32822.516 MegaBytes/min.
   Ended : Saturday, October 26, 2019 1:55:35 PM

PS C:\Users\fsugr\Desktop\sysmon>
```

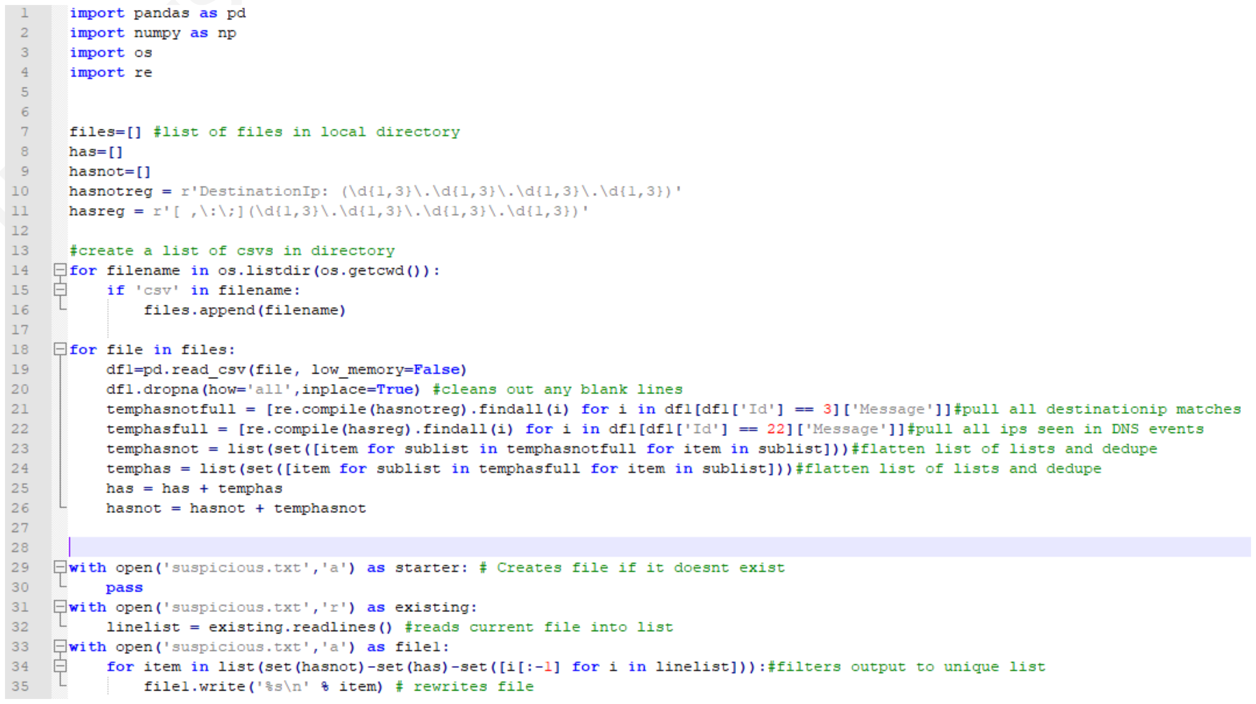
Figure 7. Robocopy Syntax and Output

Convert File to a CSV:

```
Get-WinEvent -path C:\Users\fsugr\Desktop\Sysmon\Microsoft-Windows-Sysmon%4Operational.evtx | Export-Csv C:\Users\fsugr\Desktop\Sysmon\events.csv -UseCulture
```

This example creates a static file called 'events.csv.' Deleting the first row in this newly created file will place header information into the first row, which will enable an upcoming python script to work properly.

Once the .csv is prepared, a python parsing script written by Joseph Hohmann and Gregory Melton can be run to find direct IP connections. The script makes a list of IPv4 network connections and checks those against another list of the system DNS requests. If a connection is made to an IP address that does not exist in the DNS log, it appends that IP to a file called "suspicious.txt."

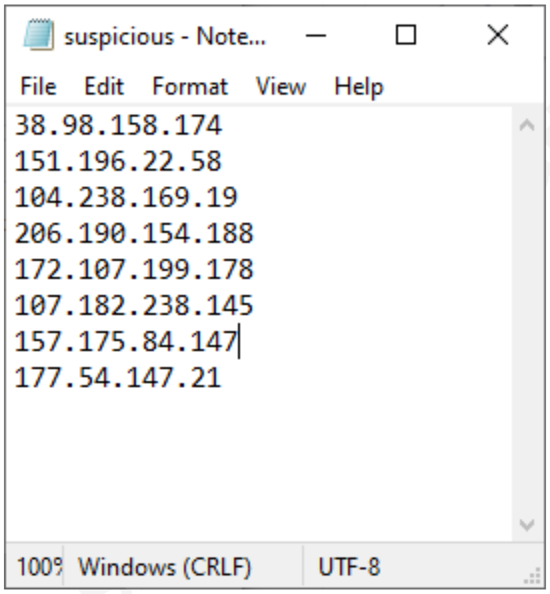


```python
import pandas as pd
import numpy as np
import os
import re


files=[] #list of files in local directory
has=[]
hasnot=[]
hasnotreg = r'DestinationIp: (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})'
hasreg = r'[ ,\:\;](\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})'

#create a list of csvs in directory
for filename in os.listdir(os.getcwd()):
    if 'csv' in filename:
        files.append(filename)

for file in files:
    df1=pd.read_csv(file, low_memory=False)
    df1.dropna(how='all',inplace=True) #cleans out any blank lines
    temphasnotfull = [re.compile(hasnotreg).findall(i) for i in df1[df1['Id'] == 3]['Message']]#pull all destinationip matches
    temphasfull = [re.compile(hasreg).findall(i) for i in df1[df1['Id'] == 22]['Message']]#pull all ips seen in DNS events
    temphasnot = list(set([item for sublist in temphasnotfull for item in sublist]))#flatten list of lists and dedupe
    temphas = list(set([item for sublist in temphasfull for item in sublist]))#flatten list of lists and dedupe
    has = has + temphas
    hasnot = hasnot + temphasnot


with open('suspicious.txt','a') as starter: # Creates file if it doesnt exist
    pass
with open('suspicious.txt','r') as existing:
    linelist = existing.readlines() #reads current file into list
with open('suspicious.txt','a') as file1:
    for item in list(set(hasnot)-set(has)-set([i[:-1] for i in linelist])):#filters output to unique list
        file1.write('%s\n' % item) # rewrites file
```

Appendix B

Once the script completes, it will output a file named "suspicious.txt". Suspicious.txt will contain all IP addresses that made network connections without first performing domain name resolution.

Example Output:

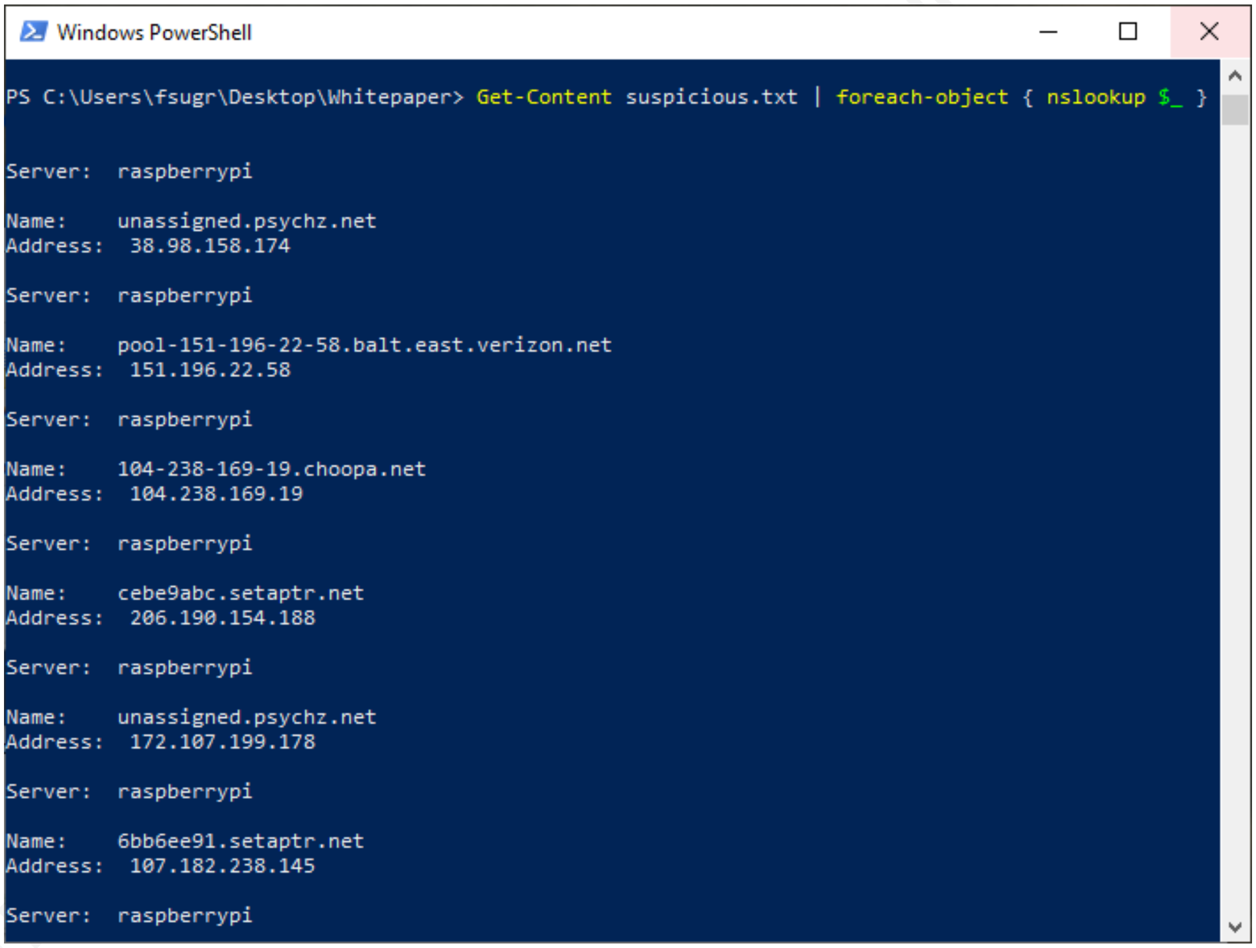


```
38.98.158.174
151.196.22.58
104.238.169.19
206.190.154.188
172.107.199.178
107.182.238.145
157.175.84.147
177.54.147.21
```

Figure 8. Suspicious.txt Output

Once a list has been created, an administrator should check individual IP addresses using NSLOOKUP or WHOIS commands to validate the findings. Any well-known IP blocks that are benign can be added back into the configuration file to exclude from future findings. This task is easily automated with the use of the following PowerShell command:

```
Get-Content suspicious.txt | foreach-object { nslookup $_ }
```



```
PS C:\Users\fsugr\Desktop\Whitepaper> Get-Content suspicious.txt | foreach-object { nslookup $_ }

Server:  raspberrypi

Name:    unassigned.psychz.net
Address:  38.98.158.174

Server:  raspberrypi

Name:    pool-151-196-22-58.balt.east.verizon.net
Address:  151.196.22.58

Server:  raspberrypi

Name:    104-238-169-19.choopa.net
Address:  104.238.169.19

Server:  raspberrypi

Name:    cebe9abc.setaptr.net
Address:  206.190.154.188

Server:  raspberrypi

Name:    unassigned.psychz.net
Address:  172.107.199.178

Server:  raspberrypi

Name:    6bb6ee91.setaptr.net
Address:  107.182.238.145

Server:  raspberrypi
```

Figure 9. Nslookup of suspicious IPs output

If an IP address is deemed to be malicious, it can be blocked at the system level by configuring the Windows firewall via the wf.msc command.

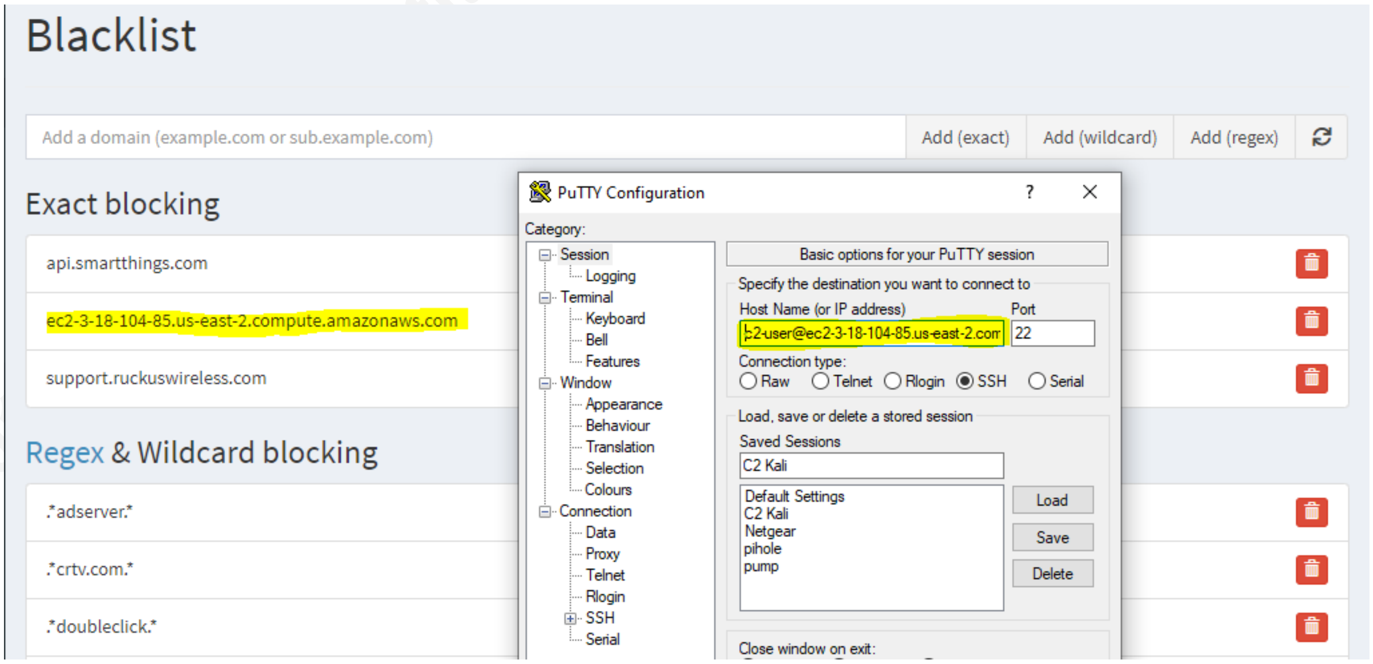
4.0 Simulations

For this research, a primary workstation is assumed to be compromised. The scope of this research overlaps with some methods to prevent initial infections, but the primary purpose is to prevent communications with malicious entities and improve awareness of when they embed in a system. In phase one, the intent is to block the initial resolution to a malicious resource on the internet. If the resource is not yet known to the

DNS server's blocklist, the endpoint will potentially be compromised. Conversely, if a user downloads and executes a malicious email attachment, DNS hardening will not stop the system from installing and running malicious code. At this point, hardening a network means stopping data exfiltration or preventing a Command and Control (C2) node from communicating with an endpoint. To simulate this circumstance, compromise is assumed, and the affected system will attempt a connection.

Test 1: Blacklist Success

First, a known malicious domain is set up for the test. The malicious domain is ec2-3-18-104-85.us-east-2.compute.amazonaws.com and represents a known malicious website in one of the auto-updating blacklists on the Pi-Hole DNS server.



Blacklist

Add a domain (example.com or sub.example.com)

Exact blocking

api.smartthings.com

ec2-3-18-104-85.us-east-2.compute.amazonaws.com

support.ruckuswireless.com

Regex & Wildcard blocking

.*adserver.*

.*crtv.com.*

.*doubleclick.*

PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 2-user@ec2-3-18-104-85.us-east-2.com

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: C2 Kali

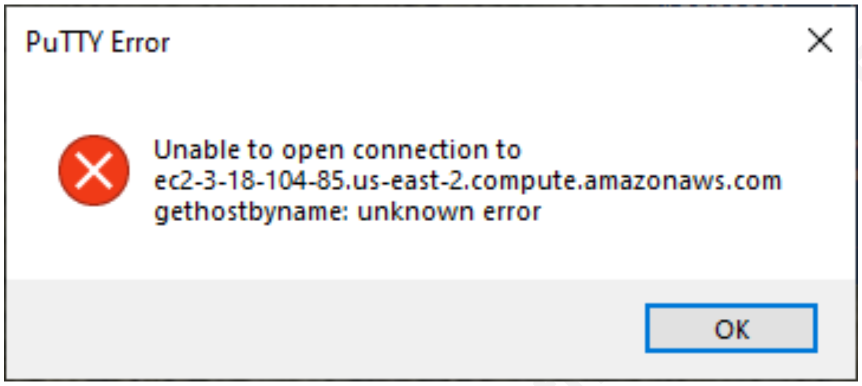
Default Settings
C2 Kali
Netgear
pihole
pump

Close window on exit:

Figure 10: Pi-hole Blacklist

Once configured, the SSH connection fails due to a lack of name resolution. The endpoint does not know how to find the resource because it was blocked at the network DNS level.

Gregory Melton, Gregory.Melton@student.sans.edu



Unable to open connection to
ec2-3-18-104-85.us-east-2.compute.amazonaws.com
gethostbyname: unknown error

Figure 11. Connection Failure

A look through the Pi-Hole log shows the DNS sinkhole in action.

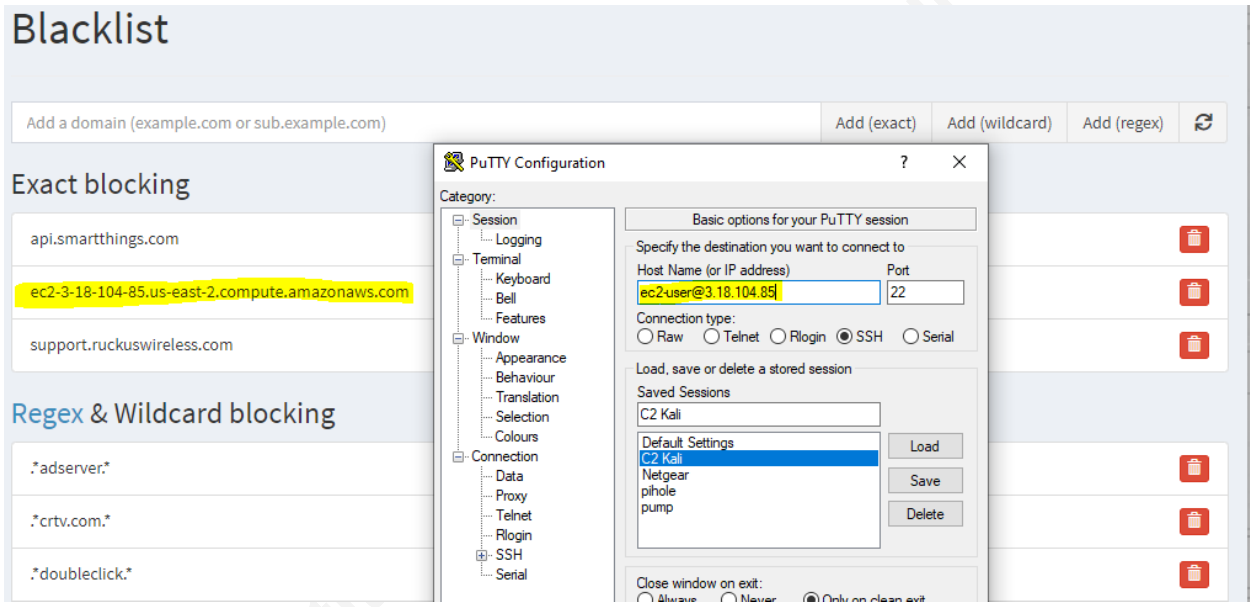
```
Oct 27 19:19:49 dnsmasq[13251]: query[A] ec2-3-18-104-85.us-east-2.compute.amazonaws.com from
Oct 27 19:19:49 dnsmasq[13251]: /etc/pihole/black.list ec2-3-18-104-85.us-east-2.compute.amazonaws.com
```

Summary:

A known malicious domain (ec2-3-18-104-85.us-east-2.compute.amazonaws.com) is expected to fail and fails with the 0.0.0.0 redirect.

Test 2: DNS Bypass

The next connection will be set to avoid name resolution by pointing directly at the C2 node's IP address. This method simulates a piece of malware with a hardcoded IP address and is not expected to be seen or blocked at the Pi-hole DNS server.



Exact blocking

api.smartthings.com
ec2-3-18-104-85.us-east-2.compute.amazonaws.com
support.ruckuswireless.com

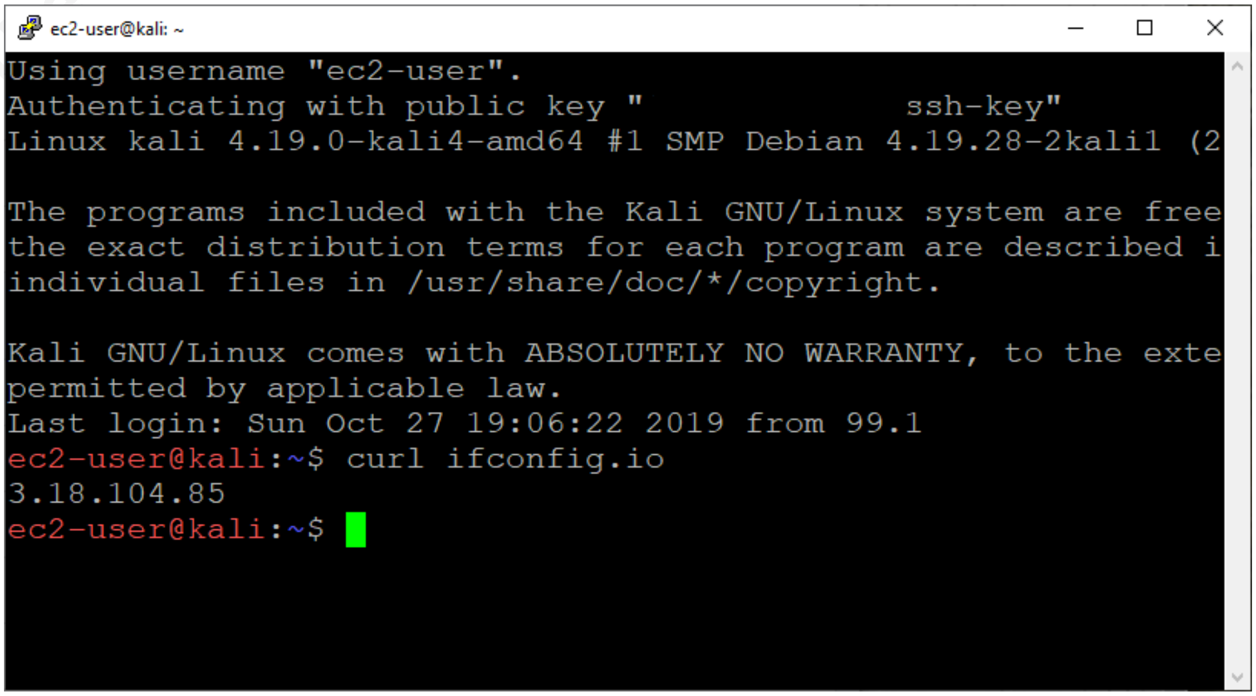
Regex & Wildcard blocking

.*adserver.*
.*crtv.com.*
.*doubleclick.*

Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address): ec2-user@3.18.104.85
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions: C2 Kali
Default Settings
C2 Kali
Netgear
pihole
pump
Close window on exit: Always Never [✓ Only on clean exit]

Figure 12. Bypassing DNS with a Direct IP Connection Setup

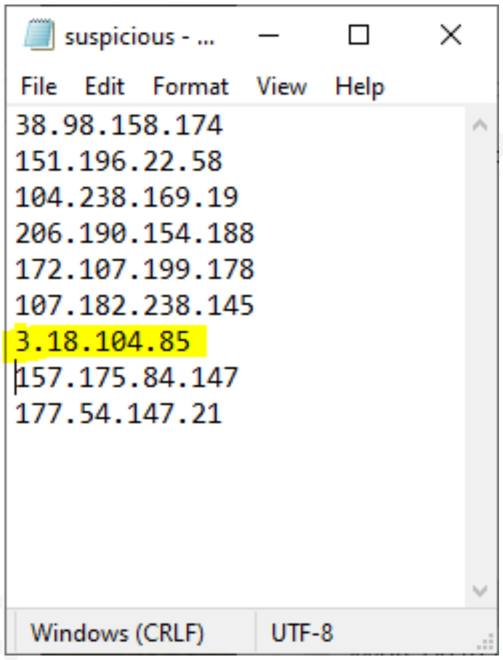
Once a connection is attempted (Figure 12) with an IP address instead of a domain name, the Pi-Hole server is completely bypassed, and the connection succeeds, as evident in Figure 13.



```
Using username "ec2-user".
Authenticating with public key "              ssh-key"
Linux kali 4.19.0-kali4-amd64 #1 SMP Debian 4.19.28-2kali1 (2

The programs included with the Kali GNU/Linux system are free
the exact distribution terms for each program are described i
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the exte
permitted by applicable law.
Last login: Sun Oct 27 19:06:22 2019 from 99.1
ec2-user@kali:~$ curl ifconfig.io
3.18.104.85
ec2-user@kali:~$
```

Figure 13. Proof of DNS/Blacklist Bypass with Successful C2 Connection

At this point, a layered approach in the defensive strategy would depend on robust logging paired with Incident Response (IR) methodologies to catch the suspicious connection. In this way, layered Defense in Depth strategies can be brought into a Small/Home Office.



```
38.98.158.174
151.196.22.58
104.238.169.19
206.190.154.188
172.107.199.178
107.182.238.145
3.18.104.85
157.175.84.147
177.54.147.21
```

Figure 14. Direct IP connection logged and added to 'suspicious.txt.'

Once discovered, a malicious IP address can be configured in the Windows Firewall to block a particular IP address (Adams 2018). A firewall update is a manual process that requires administrative attention and judgment calls. The CIRT methodology can be labor intensive but is especially feasible when dealing with a single system. This process is designed for a Small/Home office that primarily conducts business on a single defensible endpoint.

5. Conclusion

Increasing visibility and control over a Small/Home Office is an important step in small scale Defense in Depth. Enterprise-level concepts can be implemented even in a

Small/Home Office for very little expense. Blocking known malicious websites and setting up workstation visibility through robust logging is possible and potentially free. By implementing various network components and taking advantage of freely available projects, a small/home office can achieve substantially heightened security. Currently, a home user cannot proactively block non-DNS registered network connections. It is possible, however, to monitor suspicious behavior after the initial setup. Any person who works from home, deals with sensitive data, or wishes to exercise increased control over their home network would do well to implement such defensive measures and visibility.

Security is a constant battle against ever-changing adversaries. Even home users should be aware of the potential for compromise. A layered security posture is the best defense and is easier to set up and monitor than some might expect. DNS sinkholes are an excellent security addition but should not be solely relied upon to defend a network. There are many ways around a DNS blacklist, which brings the next choke point for discovering malicious activity to network communications. Enabling robust logging via SYSMON (for Windows users) is a vital step into network defense. Only after enabling visibility into network communications can suspicious activity be addressed and malicious activity thwarted.

References

Browne, R. (2018, May 30). 70% of people globally work remotely at least once a week, study says Retrieved 3 September 2019, from CNBC.com. Available at: https://www.cnbc.com/2018/05/30/70-percent-of-people-globally-work-remotely-at-least-once-a-week-iwg-study.html

Bruneau, G. (2010, August 7). DNS Sinkhole. Retrieved April 4, 2019, from https://www.sans.org/reading-room/whitepapers/dns/dns-sinkhole-33523

Churchill, M. (2015 February 24). Parsing Sysmon Events for IR Indicators. Retrieved from https://www.crowdstrike.com/blog/Sysmon-2/

Department Of Defense, U. (2014, May). Slick Sheet Best Practices For Keeping Your Home Network Secure. Retrieved from https://dodcio.defense.gov/Portals/0/Documents/Cyber/Slicksheet_BestPracticesForKeepingYourHomeNetworkSecure_Web_update.pdf

Horton, P. (2012, March 6). Layering Your Home Network Security. Retrieved from https://lockergnome.com/2012/03/06/layering-your-home-network-security/

How DNS works. (n.d.). Retrieved August 1, 2019, from https://www.cloudflare.com/learning/dns/what-is-dns/.

Lord, N. (2018, December 21). What is Endpoint Security Data Protection 101. Retrieved 12 August 2019, from https://digitalguardian.com/blog/what-endpoint-security-data-protection-101

Murray, K. (2019, April 18). Top Threats to Endpoints and How To Stay Protected. Retrieved from https://www.brighttalk.com/webcast/288/348505/top-threats-to-endpoints-and-how-to-stay-protected

QUAD9. (2018, January 22)"QUAD9 DNS: Internet Security and Privacy in a Few Easy Steps." *Quad 9*, https://www.QUAD9.net/.

Russinovich, M. (2019, September 16) Sysmon – Windows Sysinternals | Microsoft Docs. Retrieved September 26, 2019, from https://docs.microsoft.com/en-us/sysinternals/downloads/Sysmon#configuration-files

Shepherd, Maddie. (2019, July 23)"11 Surprising Working From Home Statistics." *Fundera*, Fundera, 23 July 2019, https://www.fundera.com/resources/working-from-home-statistics [Accessed 3 Sep. 2019].

Shivdas, S. (2015, June 25). Protecting Home Devices from Malicious or Blacklisted Websites. Retrieved April 20, 2019, from https://www.sans.org/reading-room/whitepapers/hsoffice/protecting-home-devices-malicious-blacklisted-websites-36152

Stevens, D. (2019, June 16). Sysmon Version 10: DNS Logging. Retrieved September 19, 2019, from https://isc.sans.edu/diary/Sysmon+Version+10%3A+DNS+Logging/25036

US Legal, Inc. (2018) Small Office/Home Office [SOHO] Law and Legal Definition. Retrieved July 14, 2019 from https://definitions.uslegal.com/s/small-office-home-office-soho

What is a Raspberry Pi? (n.d.). Retrieved September 9, 2019, from https://opensource.com/resources/raspberry-pi.

Appendix A
Sysmon XML configuration file

```
<Sysmon schemaversion="5">
 <EventFiltering>
    <DnsQuery onmatch="exclude">
       <QueryResults condition="contains">akamai</QueryResults>
    </DnsQuery>
    <NetworkConnect onmatch="exclude">
       <DestinationIp condition="contains">10.0.0</DestinationIp>
       <DestinationIp condition="contains">192.168</DestinationIp>
       <DestinationIp condition="contains">127.0.0.1</DestinationIp>
       <DestinationHostname
condition="contains">akamai</DestinationHostname>
       <DestinationHostname condition="end
with">akamaitechnologies.com</DestinationHostname>
    </NetworkConnect>
 </EventFiltering>
</Sysmon>
```

Appendix B
Python Parser

```
#Written by Joseph Hohmann and Gregory Melton
import pandas as pd
import numpy as np
import os
import re

files=[] #list of files in local directory
has=[]
hasnot=[]
hasnotreg = r'DestinationIp: (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})'
hasreg = r'[ ,\:\;](\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})'

#create a list of csvs in directory
for filename in os.listdir(os.getcwd()):
   if 'csv' in filename:
      files.append(filename)

for file in files:
   df1=pd.read_csv(file, low_memory=False)
   df1.dropna(how='all',inplace=True) #cleans out any blank lines
   temphasnotfull = [re.compile(hasnotreg).findall(i) for i in df1[df1['Id'] ==
3]['Message']]#pull all destinationip matches
   temphasfull = [re.compile(hasreg).findall(i) for i in df1[df1['Id'] ==
22]['Message']]#pull all ips seen in DNS events
   temphasnot = list(set([item for sublist in temphasnotfull for item in
sublist]))#flatten list of lists and dedupe
   temphas = list(set([item for sublist in temphasfull for item in sublist]))#flatten
list of lists and dedupe
   has = has + temphas
   hasnot = hasnot + temphasnot

with open('suspicious.txt','a') as starter: # Creates file if it doesnt exist
   pass
with open('suspicious.txt','r') as existing:
   linelist = existing.readlines() #reads current file into list
with open('suspicious.txt','a') as file1:
   for item in list(set(hasnot)-set(has)-set([i[:-1] for i in linelist])):#filters output to
unique list
```

```
file1.write('%s\n' % item) # rewrites file
```

Gregory Melton, Gregory.Melton@student.sans.edu