SCTE·ISBE CABLE-TEC EXPO 2019 NEW ORLEANS, LA SEPT. 30-OCT. 3

2019 Fall Technical Forum SCTE·ISBE • NCTA • CABLELABS

Customer Safety Initiative (CSI)

A Technical Paper prepared for SCTE•ISBE by

Matt Carothers
Security Architecture
Cox Communications
6305A Peachtree Dunwoody, Atlanta, GA 30328

Damien Whaley,
Cox Communications

Table of Contents

List of Figures

List of Tables

The Problem

In today's world, more and more companies create internet-connected products. It is estimated that the global internet population includes upwards of 17 billion connected devices (Leuth, 2018). The so-called "internet of things" (IoT) means that even common household items such as light bulbs and refrigerators feature internet connectivity. Companies with no prior networking experience now rush to market with little thought for security. Their inexperience or outsourcing to the lowest bidder creates a fertile ground for cybercrime. Criminals write worms to infect devices such as home routers, cameras, and even teapots. They compromise millions of vulnerable devices, join them together in a network called a "botnet," and use them to launch cyber-attacks. Such attacks are growing rapidly. During the first half of 2018 alone, IoT malware grew three-fold (Spadafora, 2018).

As internet service providers, we play an important role in fighting these botnets. First, we perform penetration tests against devices we deploy to our customers in order to avoid becoming part of the problem. Second, we work with third parties who report malicious activity in order to identify the Command and Control (C2) servers for botnet infections on our network.

Secure Product Development Life Cycle (PDLC)

Security is embedded into the product development lifecycle to proactively identify vulnerabilities and ensure compliance with security requirements.

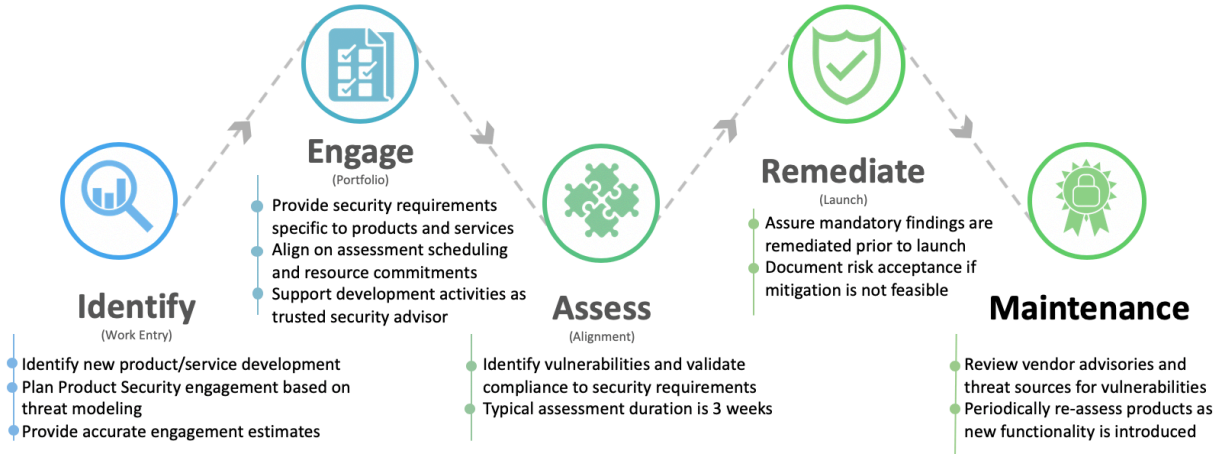

Figure 1 – Product Development Lifecycle

Identifying Malicious Control Channels

1. Stating the Problem

We often receive lists of Internet Protocol (IP) addresses participating in some malicious activity, such as sending spam, port scanning, or launching Distributed Denial-of-Service (DDOS) attacks. Third parties request that we examine network traffic in order to identify the server controlling the bots. We would like to see something like this, a single, clear host in common with all the bots:

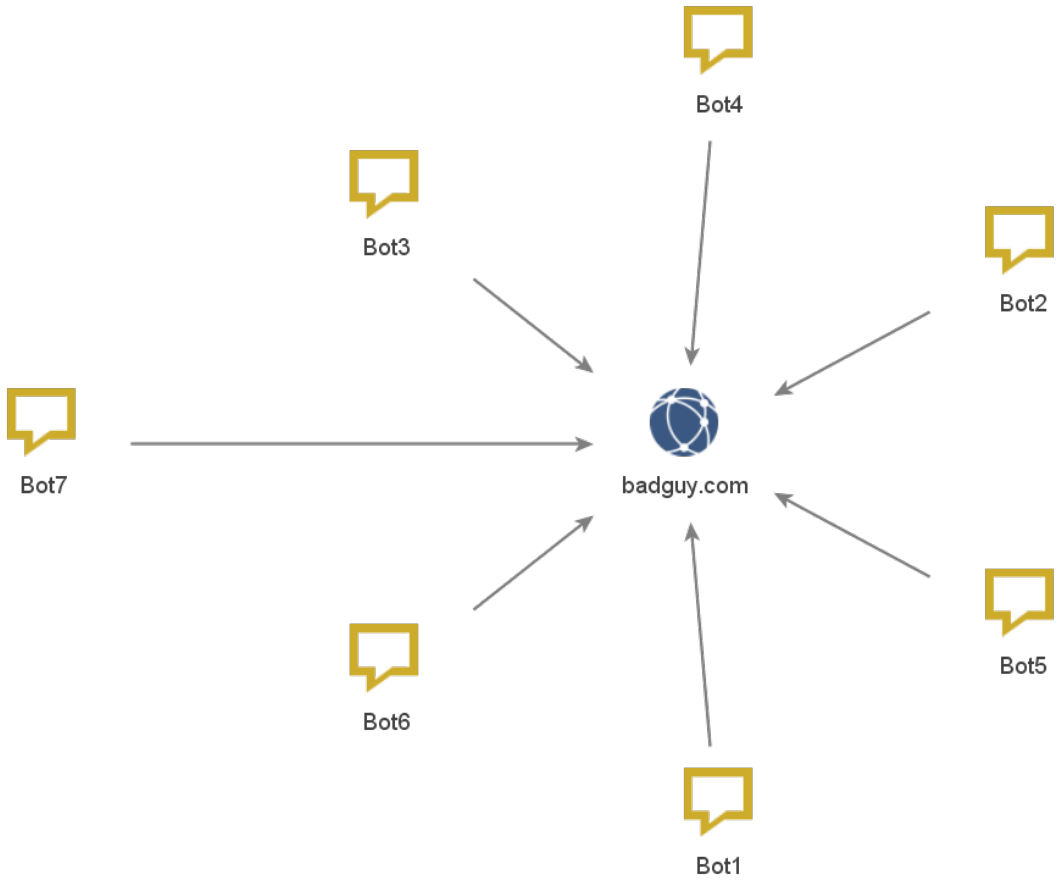

Figure 2 – Example of a Single Clear Host

However, real network graphs look more like this:

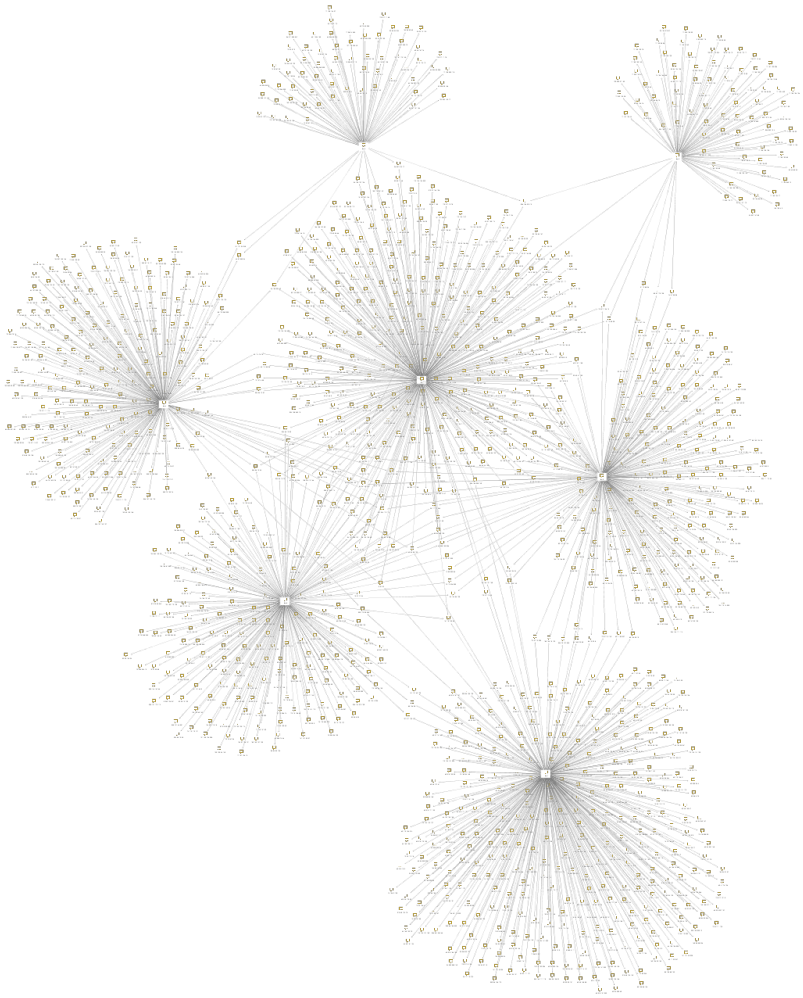

Figure 3 – Real Network Graph

While manual investigation is possible, we need an automated process.

As a naïve first approach, we simply sample some Domain Name System (DNS) requests from known infected IP addresses and find out what they have in common. We hope to find a clear picture like this:

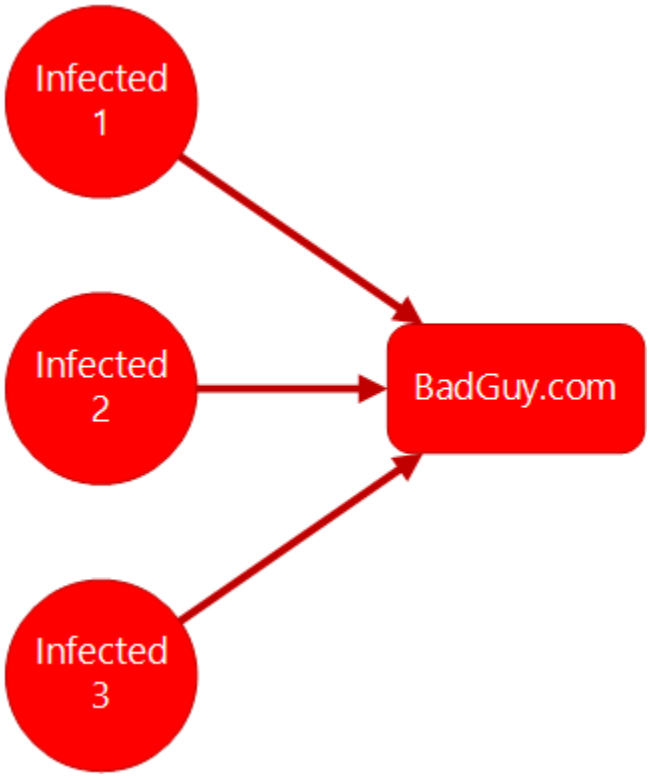

Figure 4 – Illistration of simple Domain Name System

Unfortunately, such an approach fails because most subscribers have devices contacting popular destinations such as Google, Facebook, Microsoft, Twitter, and various Content Delivery Networks (CDNs) in addition to anything malicious they might share. Simply flagging everything a group of infected devices have in common will therefore generate many false positives.

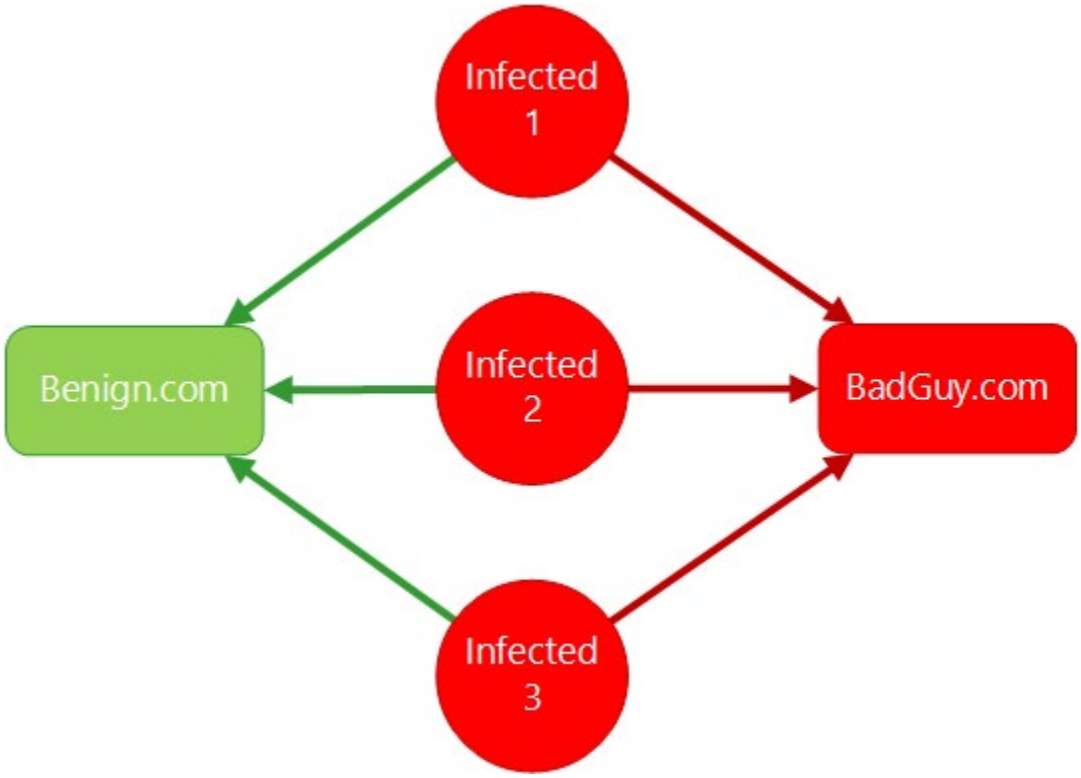

Figure 5 – Illistration of Bengin Traffic

Our second approach adds a control group. Perhaps we can map the connections from a group of noninfected devices and eliminate those results from the set?

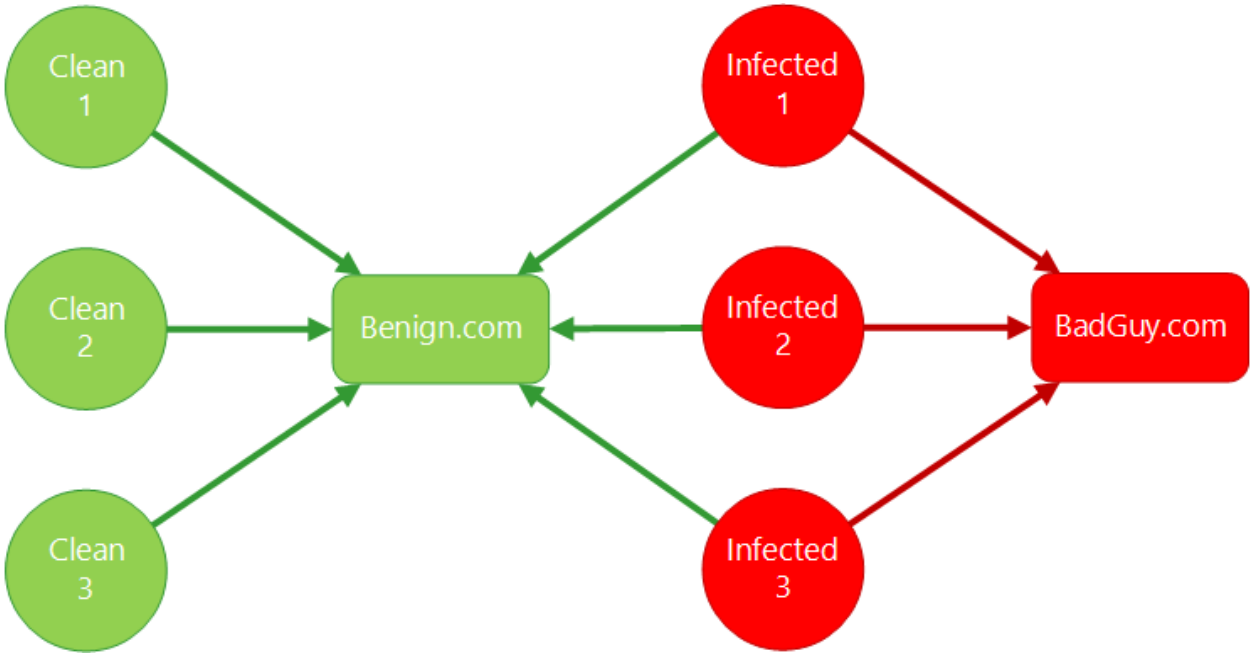

Figure 6 – Adding a Control Group to Identify Benign Traffic

This too fails because we can never say for certain that a given device is not infected. At best we can say we do not yet know it is infected. The presence of an infected device in the control group spoils this approach.

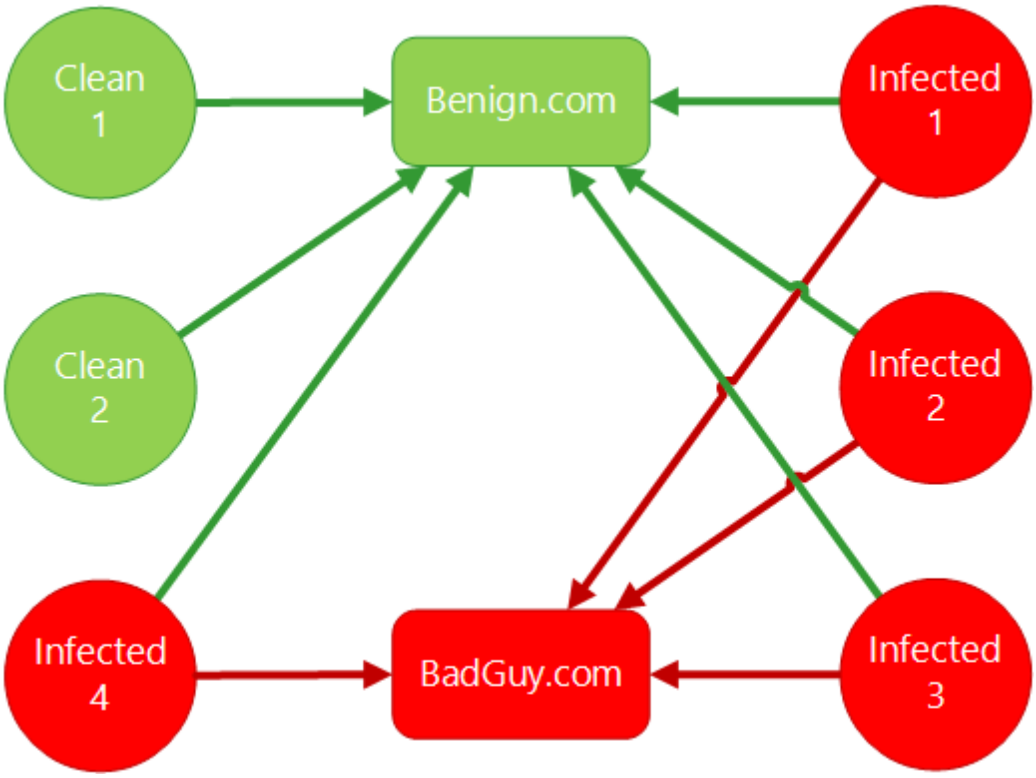

Figure 7 – Problems Caused by Infected Hosts in the Control Group

Sampling compounds the problem. We sample our netflow at 1:2000, meaning that we only send flow data for 1 out of every 2000 connections. Logging all DNS data all the time is not practical, so we sample DNS for a short period of time and hope that the malicious activity takes place during that time window. Thus, we cannot assume that every device will be found to connect to a given target during our sampling period.

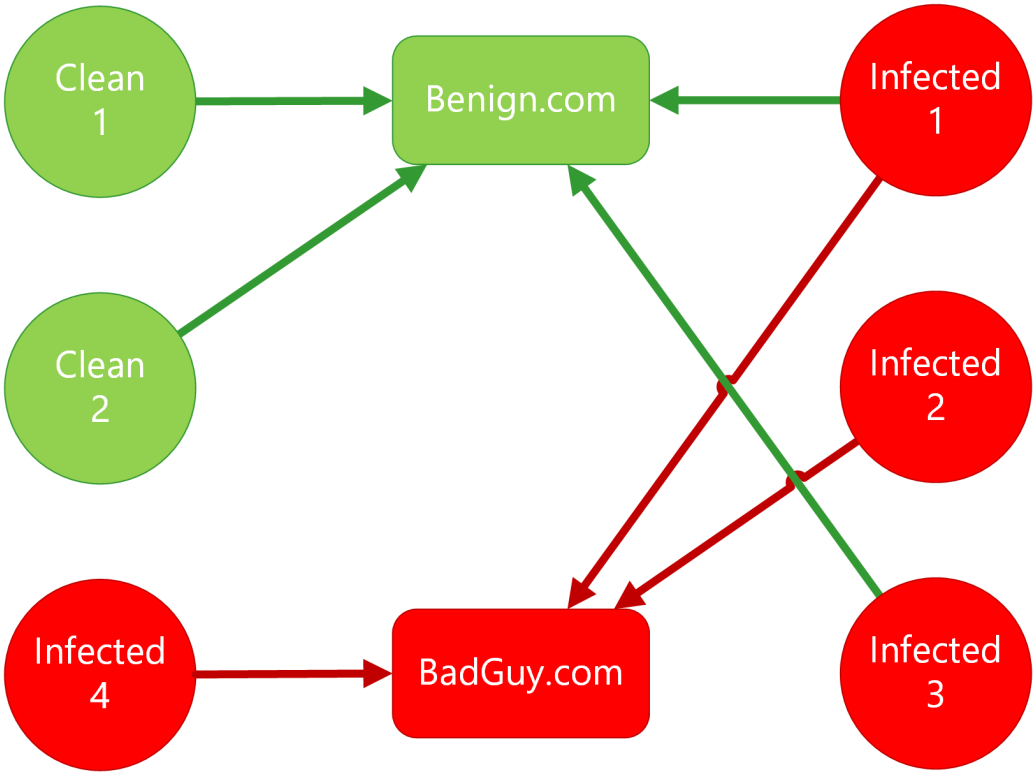

Figure 8 – Problems Caused by Sampling

2. Identifying the Malicious Traffic

In order to find the malicious control channel, we need a score rather than a binary yes or no. Our score must increase for a given target as more known infected devices communicate with it. Our score must decrease as more presumed uninfected devices communicate with it. One might be tempted to apply machine learning to the problem and attempt to cluster the devices, but we developed a much easier solution: fractions. Our score is simply the percentage of known infected hosts contacting a target divided by the number of presumed uninfected hosts:

$$\frac{\text{Percentage of infected hosts contacting a target}}{\text{Percentage of clean hosts contacting a target}}$$

3. Scoring Example

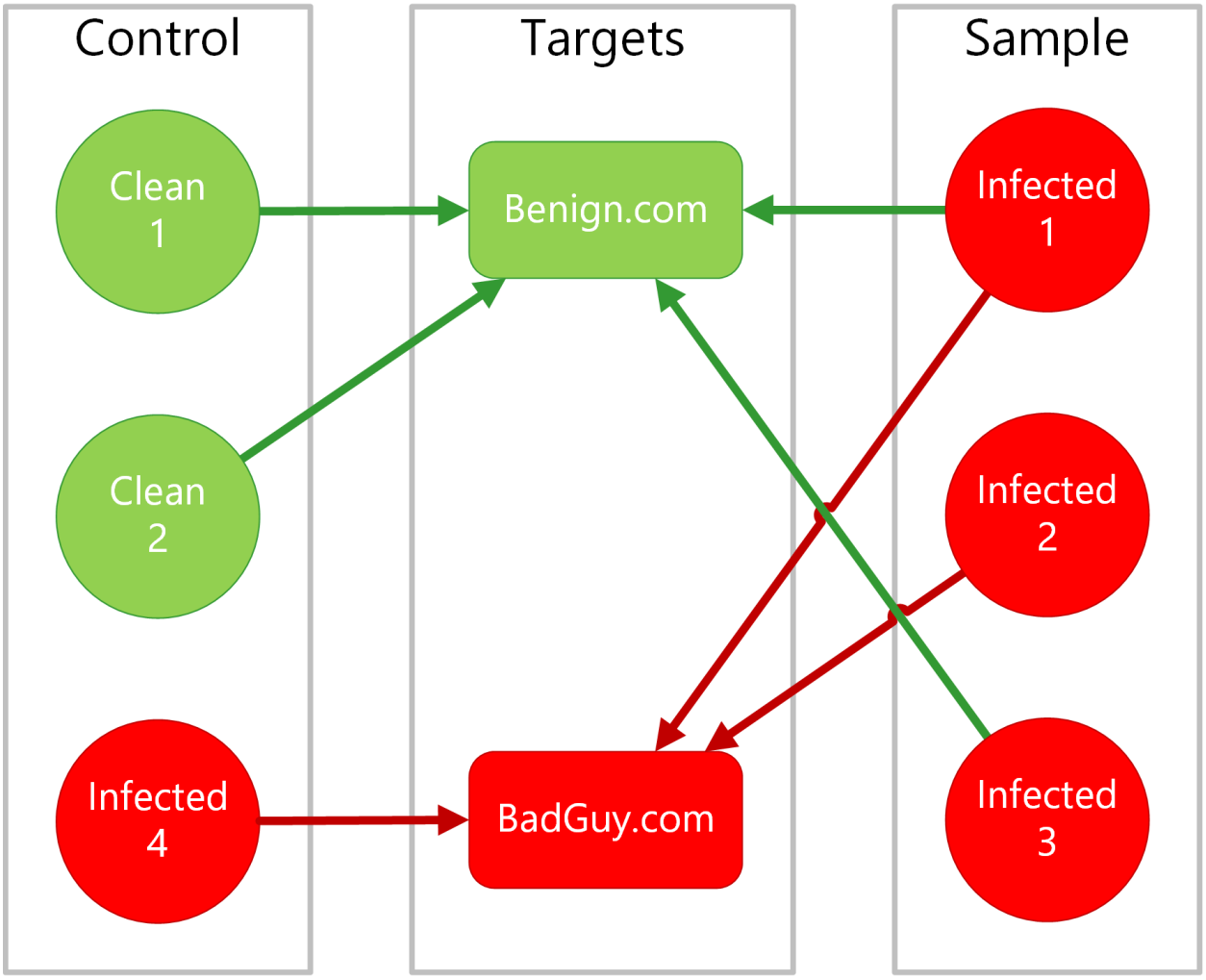

Figure 9 – Scoring Example

- 2 / 3 of infected hosts connecting to benign.com and 2 / 3 of clean hosts as well.
  Score = (2/3) / (2/3) = 1
- 2 / 3 of infected hosts connecting to badguy.com and 1 / 3 of "clean" hosts.
  Score = (2/3) / (1/3) = 2.

4. Score Interpetation

- A score of less than 1 indicates the target is under represented in the sample/infected set versus the control/clean set. I.e. the percentage of infected hosts connecting to a site is smaller than the percentage of hosts in the general population.
- A score around 1 indicates the target is found equally in both the sample/infected and control/clean sets.

- A score greater than 1 indicates the target is over represented in the sample/infected set vs. the control/clean set. I.e. the percentage of infected hosts connecting to a site is higher than the percentage of hosts in the general population.

Case Studies

5. Mirai

In 2016 Cox received a list of IP addresses participating in a DDOS attack. Analysis of DNS requests showed this:

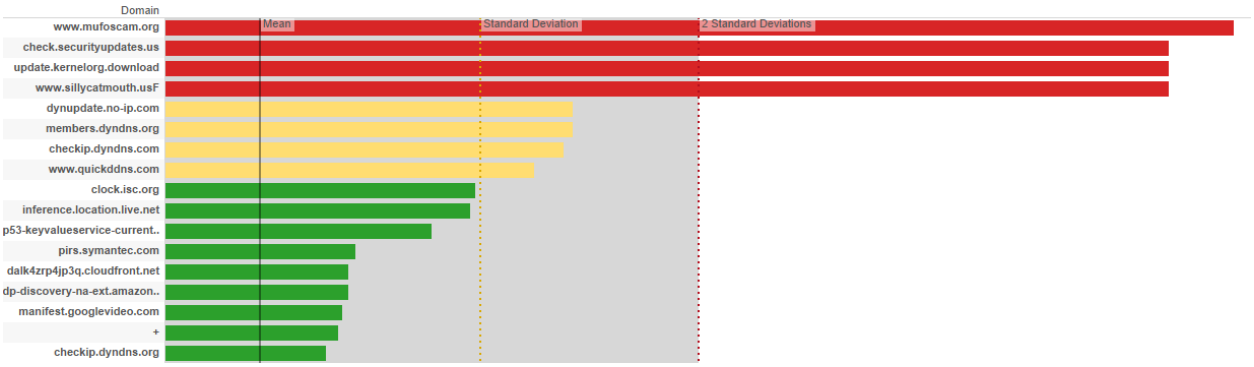

Figure 10 – Analysis of DNS requests

Zooming in, we see the top 4 hosts are Mirai controllers. The next 4, while not malicious themselves, are often associated with malicious activity.

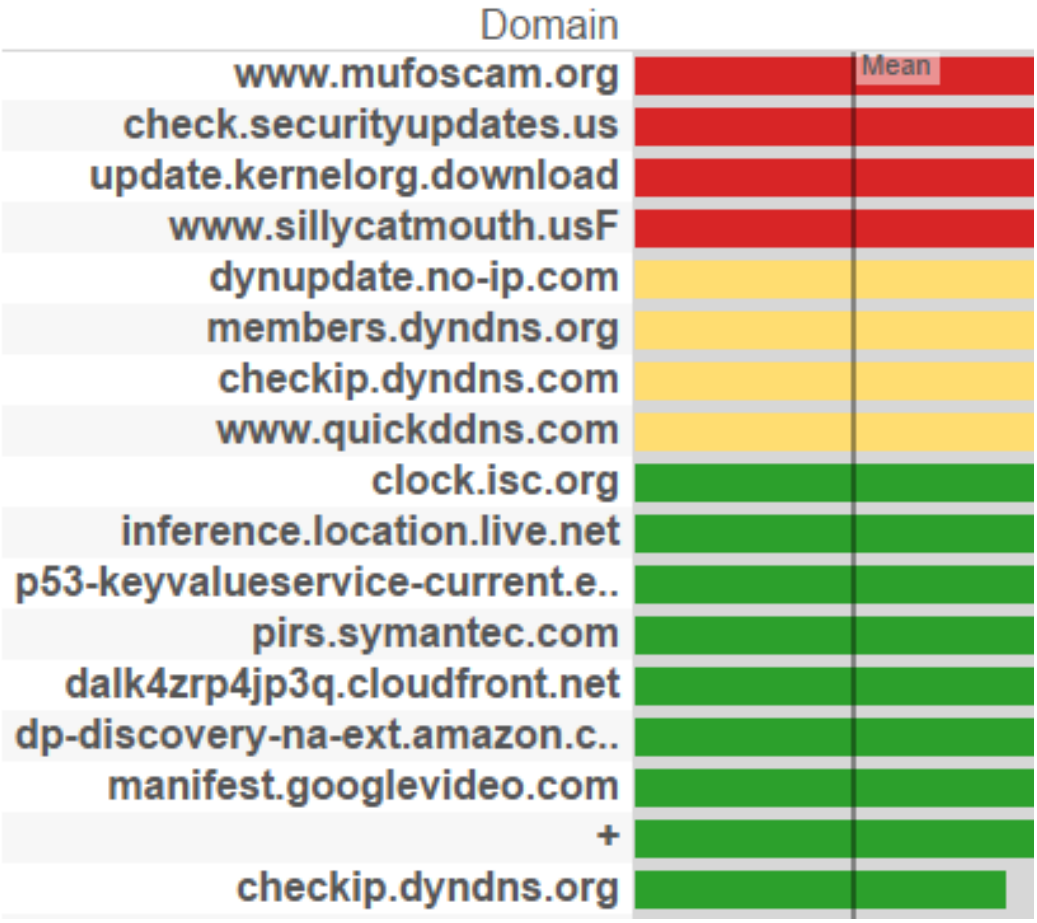

Figure 11 – Zooming in on the Analysis of DNS requests

6. Case Study – Web Scrapers

In 2019 a trusted partner reported to us that Cox IPs were scraping their web site in order to conduct fraud. Analyzing netflow for the addresses gave us these results:

Table 1 – Netflow Results

TargetIP	Port	Score	Standard Deviations	ASN	Org
23.59.30.52	443	3043.648	23.396	9498	BHARTI Airtel Ltd.
54.223.56.208	443	3043.648	23.396	55960	Beijing Guanghuan Xinwang Digital
52.80.75.244	443	2790.011	21.446	55960	Beijing Guanghuan Xinwang Digital
104.31.95.106	443	2790.011	21.446	13335	Cloudflare Inc.

The first IP address in the list was most likely the control channel. More interestingly, analysis of Transport Layer Security (TLS) certifications on the second and third IP addresses showed them to belong to jide[.]com, home of "Remix OS Player - The Most Advanced Android Game Emulator for PC." We thus concluded that the malware is android-based. Also of interest is the fourth IP address, which is Pastebin. Pastebin may have also been a control channel, or it may have been used to exfiltrate data.

Code Example

```
from missinglink import MissingLink

# Instantiate the linker with optional
# labels for the sample and control groups.
linker = MissingLink("infected", "clean")

# Designate some entities as part
# of our test group. All other entities
# are assumed to be part of the control
# group.
linker.label("10.0.0.1")
linker.label("10.0.0.2")
linker.label("10.0.0.3")


# Add some malicious relationships.  6.6.6.6 is our fictitious
malicious site.

linker.link("10.0.0.1", "6.6.6.6")
```

```
linker.link("10.0.0.2", "6.6.6.6")
# Add some benign relationships.
linker.link("10.0.0.1", "8.8.8.8")
linker.link("10.0.0.2", "8.8.8.8")
linker.link("10.0.0.3", "8.8.8.8")
linker.link("10.0.0.4", "8.8.8.8")
linker.link("10.0.0.5", "8.8.8.8")
linker.link("10.0.0.6", "8.8.8.8")
linker.link("10.0.0.6", "9.9.9.9")
# Analyze the results
linker.analyze()

# Analyze the results
linker.analyze()
# Output the results
for result in linker.results:
    print(json.dumps(result))
```

7. Output


```
{
 "target": "6.6.6.6",
 "score": 2.0,
 "deviations_from_mean": 1.224744871391589,
 "infected_count": 2,
 "infected_percent": 0.6666666666666666,
 "clean_count": 0,
 "clean_percent": 0.0
```


}

Target	Score	Deviations from mean	Infected count	Infected percent	Clean count	Clean percent
6.6.6.6	2	1.2	2	67%	0	0%
8.8.8.8	1	0.0	3	100%	3	100%
9.9.9.9	0	-1.2	0	0%	1	33%

References

Spadafora, A. (2018, September 19) *IoT malware grew significantly during the first half of 2018.* Retrieved from

https://www.techradar.com/news/iot-malware-grew-significantly-during-the-first-half-of-2018

Leuth, K. (2018, August 8). *State of the IoT 2018.* Retrieved from https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

