Assessing the Suitability of Traditional Botnet Detection against Contemporary Threats

Ashley Woodiss-Field
School of Science
Edith Cowan University
Joondalup, Australia
a.woodiss-field@ecu.edu.au

Michael N. Johnstone
School of Science
Edith Cowan University
Joondalup, Australia
m.johnstone@ecu.edu.au

***Abstract*—Botnets are groups of compromised devices used by malicious actors to perpetrate various forms of cyber-attacks. The Internet of Things involves the use and operation of (often small, low power) devices such as household appliances, industrial sensors and actuators, and media devices. Contemporary botnets have been known to target IoT devices for use in their attacks. Traditional botnet detection techniques may not be adequate in detecting contemporary botnet threats. BotMiner is one such technique. This paper discusses the attempted recreation of BotMiner and the limitations found in the context of IoT-based Botnet detection.**

***Keywords*-Botnet; Internet of Things; Mirai; BotMiner**

I. Introduction

A botnet is group of compromised devices, known as bots, that are used as a platform for perpetrating attacks by a malicious party. Bots traditionally include desktop computers and servers but in recent years have come to include Internet of Things (IoT) devices [1][2].

IoT-based botnets pose a greater threat than their traditional counterparts due to their greater numbers for compromise, ongoing availability and lack of security. This has manifested in several high-profile attacks, particularly, the Mirai attack. Mirai was able to take over 140,000 devices and perpetrate attacks at speeds over 1Tbps [3][4].

The coalescence of the 5th domain (cyber) with the 1st-4th domains (land, sea, air and space) means that state actors can effectively use cyber to achieve their aims with or without attribution. This is problematic with respect to systems that use cyber-enabled command and control. Woodcock, cited by Carlin and Graff [12] stated that, at 4 cents per machine (device used for a cyber attack), an entire cyber warfare campaign could be funded for the replacement cost of a tank tread. As noted recently [13], both the US public and private sectors are currently on a cyber attack high alert, particularly with respect to critical infrastructure. In an Australian context, the 2016 Defence White Paper [14] clearly states that cyber attacks are a direct threat to the ADF's warfighting ability given its reliance on information networks. Further, cyber is identified as a capability stream in the White Paper and a priority is protection of Defence and other critical Australian government systems from malicious cyber intrusion and disruption—including enhancing the resilience of Defence networks. The emergence of the Mirai botnet, suggests that botnets represent a particularly insidious threat to capability. Mirai was an interesting case as it exhibited properties that are expected in defensive systems. First, it used redundant systems in the botnet. Second, it was resilient in that there was no centralised command server which could be removed to disable the botnet. Finally, it removed competitor malware. Therefore, research into existing and new botnet detection techniques is essential.

Further compounding the threat of IoT-based botnets, current traditional botnet detection techniques appear unprepared for IoT-based botnet threats. Techniques designed to operate over traditional botnet protocols, such as HTTP and IRC, are limited against any botnet, IoT-based or otherwise, that utilises peer-to-peer or specialised communication protocols. The availability of insecure IoT devices has enabled IoT-based botnets, such as Mirai, to takeover individual bots instead of entire networks. This change in typical botnet structure has the potential to undermine some detection technique, including BotMiner [5].

BotMiner is a detection technique that operates by clustering network flows (netflows) and IDS alerts, and then cross-clustering the results to determine common communications then presumed to be botnet traffic [6]. The BotMiner technique was recreated for testing against IoT-based botnets. The overall goal is to determine if any traditional botnet detection techniques, BotMiner among them, are capable of IoT-based botnet detection. This paper discusses the recreation development and pre-experimental analysis of BotMiner's capabilities. Other detection technique recreations have been undertaken towards the overall goal as well, such as BotProbe [7].

II. Related Work

The recreation of BotMiner is one part of an overall goal for determining the suitability of traditional botnet detection. Other techniques that have been or are considered

978-1-7281-8019-9/20/$31.00 ©2020 IEEE
DOI 10.1109/ETSecIoT50046.2020.00008

for recreation include BotProbe, BotHunter, and a DNS-based detection technique [5].

Although frequently mentioned in literature, honeypots and exclusively signature-based detection techniques were excluded from the scope of researching Botnet detection techniques. Both techniques are unable to detect activity of emerging Botnets as they either rely on previous cases or cannot protect targets of attacks [5].

BotProbe is a technique that probes suspected bots through several methods, including sending suspected commands to suspected bots to gauge their response [8]. While BotProbe may be conceptually capable of detecting IoT-based botnets, as originally developed it is only capable of detecting IRC-based botnets and would require foreknowledge of the command and control (CnC) protocols otherwise [7].

BotHunter is a technique that utilises IDS signatures to build an evidence trail for botnet detection. BotHunter utilises a model based on propagation and CnC structure. BotHunter is effectively protocol independent. However, it can only work if CnC communications trigger IDS alerts. The strength of BotHunter is that it should be able to operate with generic signatures, facilitating the detection of emergent threats-as distinguished from signature-only approaches, which are unable to detect new threats [9].

Another technique is a combination of gradual whitelisting and blacklisting, communication patterns to specific domain names, and the flagging of dynamic DNS use. Some features of this technique are likely weaker on IoT-based botnets, particularly the flagging of dynamic DNS and group communications, common in certain IoT deployments [10].

III. BotMiner

BotMiner functions through the clustering of netflows and signature alerts. BotMiner operates on the premise that botnets typically take over many devices from the same network and that the bots often operate at the same time [6].

A. BotMiner Capabilities

Similar to BotHunter (and in contrast to BotProbe), BotMiner is able to operate independently of botnet communication protocols, only limited by the IDS signatures in use [8][9].

BotMiner is claimed to have a low false positive rate as the results of the netflow and alert clustering are cross-clustered. Any false positives caused by one of the clustering processes can be nullified by the cross-clustering of the results [6].

B. BotMiner Process

Alert collection originally utilised the Snort IDS for general alert detection. The statistical scan anomaly detection engine (SCADE) is used for detecting scan activities. The alert collection module can be further customised for bot activity detection.

$$s(h) = \sum_{\substack{i,j \\ j>i \\ t(A_i)\neq t(A_j)}} w(A_i)w(A_j)\frac{|A_i \cap A_j|}{|A_i \cup A_j|} + \sum_{i,k} w(A_i)\frac{|A_i \cap C_k|}{|A_i \cup C_k|},$$

Where: A is the sequence of hosts within alert clusters;
C is the sequence of hosts within flow clusters; and
w are activity type weights.

Figure 1. Computation of bot likelihood score (s) for a host (h) [6].

Netflow collection was facilitated using fcapture. Only TCP and UDP flows were collected. Flow features included time, duration, source IP, source port, destination IP, destination port, and the bidirectional amount bytes and packets [6].

Alerts are clustered by client and activity type. Example activity types include scan activity, spam activity, and binary downloads. How these activities are grouped together is not clear. Some features are handled differently for specific activity types.

Flow clustering uses X-means clustering to determine groups of similar flows. Flows are first grouped together if they have the same flow features. These flows are then split over 13 intervals and more features are extracted. Features include the sample distributions of flows per hour (fph), packets per flow (ppf), average bytes per packets (bpp), and average bytes per second (bps). The variances and means of these features over the 13 intervals are found to produce an 8-feature dataset and the dataset is put through X-means clustering. Then, all features over all intervals (52 features) are clustered for each cluster from the initial X-means clustering [6].

Cross-clustering involves finding intersections of the results from the alert and flow clustering. A threshold score is established for each host found to have been conducting suspicious activity from the alerts clustering.

For each suspicious host, a botnet score is produced to be compared by the established threshold score. Botnet scores are computed for hosts within the alert and flow clusters using the formula shown in figure 1, with a high score suggesting that the host is part of a botnet.

The threshold in the original BotMiner publication was set to 1. Interestingly, even though the formula in figure 1 accounts for activity-based weights, said weights would be set to 1 rendering them ineffective. It appears that the inclusion of these weights was intended as supplementary [6].

IV. Experimental Approach

The recreation of the BotMiner detection technique included netflow and alert collection then netflow clustering and alert clustering. The technique was designed to operate

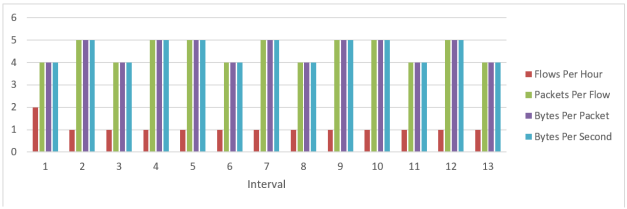

Figure 2. Feature distribution for sample CNC C-Flow. Each feature is the sample distribution of specified values over 13 intervals.

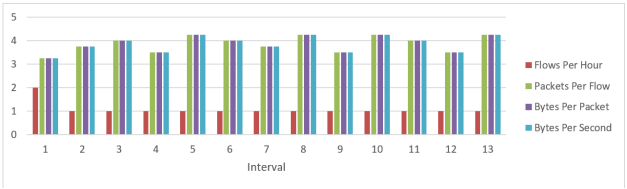

Figure 3. X-mean cluster centre features to which the C-Flow in figure 2 belongs.

over pre-recorded pcap files. This was considered essential for experimental repeatability, particularly for contrasting results with other techniques.

Preliminary testing was conducted during the recreation process. IRC-based botnet activity was simulated using virtual machine hosts. The simulated botnet communicates with the infected hosts to ping the targets at consistent intervals.

Alert collection was conducted using Snort community rules. In order to cluster the alerts, a way of determining how the alerts would be considered similar had to be devised. To this end, clustering of alerts was done primarily on the affected host and the category of the alert. The direction from where the alert originated and whether the affected client host was the source, or the destination was also considered.

This approach departs somewhat from the original approach primarily as the type specific feature-based clustering would be too precise. Some bot activities, such as crypto mining [11], may not be categorised correctly, and there is no certain way to predict future changes in botnet use.

For flow collection, a custom tshark-based solution was developed for initial collection. Standard flow features were collected, and large ongoing flows were split on either a running time limit or an inactivity time limit. Flows would then be grouped into C-Flows and the sample distribution of the features mentioned in section 3.2 would be calculated. The C-Flows would then be clustered using X-means clustering. X-means clustering was conducted using available Python-based libraries, such as scikit learn.

Figure 2 shows an example of the distributed features that make up a C-Flow. The C-Flow shown in figure 2 constitutes consistently paced CnC communications between a bot and an IRC server across a total of 47 individual flows. Because the botnet traffic operates consistently and uniformly, as mentioned in section 3.2, these individual flows are grouped into C-Flows. The duration of the flows in each C-Flow is determined and then split in to 13 intervals. The fph, ppf, bpf, and bps are determined for each interval and then the sample distribution for each of those features is recorded across all 13 intervals, producing the 52-feature sample displayed.

The C-Flow in figure 2 would be clustered with other recorded C-Flows. Figure 3 displays one of the cluster centres produced through clustering all the C-Flows. The data in figure 2 and figure 3 show some similarities because the C-Flow in figure 2 belongs to the cluster with the centre shown in figure 3.

Once alert and flow clusters have been created, they are then put through the cross-clustering process. The cross-clustering formula shown in figure 1 was recreated to produce the bot score. As with the original publication, the botnet score threshold was kept at 1 and activity type weightings were not utilised. The decision to not set activity weights was determined by both the original work, as well as the ever-changing nature of what those weights would attempt to represent [6][11].

V. Limitations

Some limitations of BotMiner can be inferred from [6]. It is not designed to handle cases where only single devices would be taken over from many networks, as with Mirai. Instead it appears that BotMiner expects multiple similar bots operating at the same time on the same network. Even if multiple devices were taken over, careful planning may allow botnets to evade detection by altering the timings of the CnC communications and actions.

During extensive analysis in recreating BotMiner, it became apparent that the alert clustering mechanism was originally developed with arbitrary feature categories. These arbitrary categories may be inappropriate in a dynamic environment, due to changes in how different botnets conduct their activities. For example, when BotMiner was originally developed, the emergence of botnets conducting crypto-mining activities had not yet occurred. This example, and others which pertain to both botnet targets and methods of propagation constantly changing, renders the limited amount of alert categorisations a liability. This limitation also weakens the supplementary activity weighting present in the cross-clustering component. A method of constantly updating the new ways in which botnets are created and how they are used would be required to mitigate such limitations.

As with BotHunter, a reliance on alert signatures still exists. If the signatures can remain generic, this may prove to be adequate. However, signatures will have to be updated constantly to keep up with emerging threats and vulnerabilities.

VI. Conclusion

BotMiner is a technique with some potential for detecting IoT-based botnets. Most of BotMiner's weaknesses are not necessarily IoT botnet-oriented. A stronger approach towards alert categorisation would allow for greater future-proofing. Nonetheless, BotMiner's greatest pitfall is the premise of many bots per network.

Against a Mirai-type takeover, this technique is unlikely to succeed. One mitigating approach may be to orient the clustering technique towards single devices and comparing frequent similar activities and communication over its own timeline. Such an alteration would operate on the premise that Bots will communicate in predictable patterns over time.

Future work will include further experimentation with BotMiner, and other recreated detection techniques, on simulated IoT-based botnets. This will also involve testing with simulated background activity.

References

[1] Eslahi, M., Salleh, R., & Anuar, N. B. (2012). Bots and botnets: An overview of characteristics, detection and challenges. In Control system, computing and engineering (iccsce), 2012 ieee international conference on (pp. 349-354).

[2] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. Future generation computer systems, 29 (7),1645-1660.

[3] Angrishi, K. (2017). Turning internet of things (iot) into internet of vulnerabilities(iov): Iot botnets. arXiv preprint arXiv:1702.03681 .

[4] Elzen, I., & Heugten, J. (2017). Techniques for detecting compromised iot devices (Master's thesis, University of Amsterdam). Retrieved from http://work.delaat.net/rp/2016-2017/p59/report.pdf

[5] Woodiss-Field, A., & Johnstone, M. (2018). Towards a method for detecting botnet code on IoT devices. The Proceedings of the 2018 Cyber Forensic & Security International Conference (pp. 30 - 34). Nuku'alofa, Kingdom of Tonga: Christ's University in Pacific.

[6] Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). Botminer: Clustering analysis of network traffic for protocol- and structure-independent Botnet detection. In Usenix security symposium (Vol. 5, pp. 139-154).

[7] Woodiss-Field, A., & Johnstone, M. (2019). BotNets in the Internet of Things: The Next Wave. 18th Australian Cyber Warfare Conference. Melbourne, Victoria, Australia.

[8] Gu, G., Yegneswaran, V., Porras, P., Stoll, J., & Lee, W. (2009). Active Botnet probing to identify obscure command and control channels. In Computer security applications conference, 2009. acsac'09. annual (pp. 241-253)

[9] Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., & Lee, W. (2007). Bothunter: Detecting malware infection through ids-driven dialog correlation. In Usenix security (Vol. 7, pp. 1-16).

[10] Lee, J., Kwon, J., Shin, H.-J., & Lee, H. (2010). Tracking multiple c&c botnets by analyzing dns trace. In Secure network protocols (npsec), 2010 6th ieee workshop on (pp. 67-72).

[11] J. Wyke. The ZeroAccess botnet: Mining and fraud for massive financial gain. Sophos Technical Paper, 2012.

[12] Carlin, J.P. and Graff, G.M. (2019). Dawn of the Code War. PublicAffairs: New York.

[13] Murphy, H. (2020). US on high alert for Iran-backed cyber attacks. Financial Times. January 5.

[14] DoD (2016). 2016 Defence White Paper. Department of Defence. Commonwealth of Australia.