Towards Evaluating the Effectiveness of Botnet Detection Techniques

Ashley Woodiss-Field[1], Michael N. Johnstone[1], and Paul Haskell-Dowland[1]

Edith Cowan Univeristy
{a.woodiss-field,m.johnstone,p.haskelldowland}@edu.edu.au

Abstract. Botnets are a group of compromised devices taken over and commanded by a malicious actor known as a botmaster. In recent years botnets have targeted Internet of Things (IoT) devices, significantly increasing their ability to cause disruption due to the scale of the IoT. One such IoT-based botnet was Mirai, which compromised over 140,000 devices in 2016 and was able to conduct attacks at speeds over 1 Tbps. The dynamic structure and protocols used in the IoT may potentially render conventional botnet detection techniques described in the literature incapable of exposing compromised devices. This paper discusses part of a larger project where traditional botnet detection techniques are evaluated to demonstrate their capabilities on IoT-based botnets. This paper describes an experiment involving the reconstruction of a traditional botnet detection technique, BotMiner. The experimental parameters were varied in an attempt to exploit potential weaknesses in BotMiner and to start to understand its potential performance against IoT-based botnets. The results indicated that BotMiner was able to detect IoT-based botnets surprisingly well in various small-scale scenarios, but produced false positives in more realistic, scaled-up scenarios involving IoT devices that generated traffic similar to botnet commands.

Keywords: Botnet · Internet of Things · Mirai · BotMiner.

1 Introduction

A botnet is a collection of compromised devices used by a malicious actor to conduct various activities. A device taken over by a botnet is often referred to as a bot or a zombie. The controller of a botnet is known as a botmaster. The structure from which bots are coordinated, to conduct attacks or takeover more devices, is known as a command and control structure, often abbreviated as CnC, C&C or C2. Botnets pose a threat primarily through their ability to grant a single malicious actor the processing power of the bots under their control. Botnet threats commonly include large scale Distributed Denial of Service (DDoS) attacks, spamming, phishing, malware distribution, click fraud, and crypto-jacking [6, 14, 11, 2].

Traditionally botnets would target workstations, personal computers, and server to propagate. Historically botnets would first utilise the IRC protocol

to send commands to infected devices. Later botnets would utilise various peer to peer protocols and the HTTP protocol, the former allowing for complex decentralised botnets and the latter facilitating a pull-based CnC approach [6, 7]. Most recently, the advent of botnets that utilise the Internet of Things (IoT) to propagate and perpetrate attacks presents new challenges in contrast with their traditional counterparts, both in terms of the threats posed and approaches towards detection [3].

Various botnet detection techniques have been developed, most commonly based on honeypots and passive signature-based detection techniques but these techniques are incapable of detecting zero-day (or unknown/unpublished) threats. More advanced techniques that focus on particular botnet attributes such as deterministic command responses, netflow patterns among compromised networks, and infection lifecycles have been more effective for specific types of botnet. Many detection techniques were designed for traditional botnets as opposed to more recent IoT-based threats [19, 9, 8, 10].

This paper describes a reconstruction of a botnet detection technique known as BotMiner, with the goal of understanding its capabilities towards detecting IoT-based bots. The reconstructed BotMiner is tested on several simulated networks which include both traditional and IoT infection targets, various kinds of botnets using known CnC protocols, and variable infection rates. The goals of the experiment are to both test hypotheses surrounding the strengths and weaknesses of the BotMiner detection technique, while also validating the reconstruction thereof by applying it to the traditional targets and environments for which it had originally been built.

BotMiner is a botnet detection technique that operates by clustering alerts and netflows, and the performing cross-clustering analysis on the results. Utilising alert categories, X-Means clustering on flow features, and an established threshold, selected hosts on a given network are evaluated with a results that breaches the threshold indicating that it is a bot. BotMiner is designed to find common flow and alert patterns which are typical for devices that have become part of a single botnet. However, potential weaknesses in BotMiners detection approach may include over reliance on alert signatures and reliance on a botnet taking over entire target networks. Alert signatures that are triggered by benign or non-bot related activities may produce false positives. Bots that operate alone on a given host network may better evade detection. The conditions that allow IoT-based botnets to thrive may facilitate these circumstances, rendering BotMiner as an either inadequate or weakened solution to current botnet detection needs [8].

This paper seeks to address the potential weaknesses of BotMiner on IoT-based botnets, particularly in the context of non-standard CnC protocols, the potential for false positives, and lone bots among host networks. A particular case, that of the infamous Mirai botnet, forms the basis by which one the IoT-based botnets are simulated. Mirai is known to have used a non-standard bitstream-based CnC protocol which, unlike IRC, would likely evade detection by an Intrusion Detection System (IDS) on its own. Mirai's propagation approach,

本文试图解决BotMiner在基于物联网的僵尸网络上的潜在弱点，特别是在非标准的CnC协议、潜在的假阳性和主机网络中的孤独僵尸的背景下。一个特殊的案例，即臭名昭著的Mirai僵尸网络，构成了一个基于物联网的僵尸网络的模拟基础。众所周知，Mirai使用了一种非标准的基于比特流的CnC协议，与IRC不同，它很可能会逃避入侵检测系统（IDS）本身的检测。Mirai的传播方式与其他物联网僵尸网络（如Aidra、Linix/IRCTelnet和Bashlite）类似，并不针对整个网络。大量可用的不安全的物联网设备适应了这种方法，在这种情况下，僵尸管理员不需要产生比必要的更大的感染足迹。

similar to other IoT botnets such as Aidra, Linix/IRCTelnet, and Bashlite, did not target entire networks. The large number of available insecure IoT devices accommodates such an approach where a botmaster does not need to produce a greater infection footprint than what is necessary.

2 Botnets

The experiment undertaken for this paper involves testing the recreated botnet detection technique on traditionally structured botnets. Three broad categories of traditional botnets exist, two of which were simulated for the experiment. The two botnets simulated include an HTTP-based Pull Botnet and an IRC-based Push Botnet. The third type of traditional botnet is the Peer-to-Peer (P2P) botnet. The latter is typically more complex than its centralised counterparts and requires expert knowledge to set up [6, 17]. HTTP-based botnets use a centralised CnC structure where each bot routinely queries an HTTP server to receive commands [17, 15]. IRC-based botnets also use a centralised CnC structure where all bots listen on an IRC server for commands from the botmaster [17, 15] . Many IoT-based botnets also continue to use the IRC protocol, with the original 2016 Mirai being a prominent exception [3].

IoT-based botnets are a more recent threat compared to traditional botnet structures. IoT-based botnets have an advantage over other botnets due to the attributes of the devices they target. IoT devices are exceptionally numerous by design, with approximately 9 billion devices distributed worldwide in 2016 [3] and was predicted to have reached over 20 billion devices by the end of 2020 [16], which allows for IoT-based botnets to not only propagate faster but also to be selective in which devices they compromise. IoT devices are often insecure, either due to poor deployment practices such as not changing default credentials or software and firmware exploits [3, 12]. IoT devices are typically always switched on (although, not always active), whereas workstations and home computers are often shut down when not in use. Being always on means that IoT bots are always available to their botmaster [17]. A particularly notable IoT-based botnet was Mirai, which was able to infect over 140,000 devices and conduct attacks at speeds over 1 Tbps [5]. Other IoT-based botnets include Linux/IRCTelnet, Aidra, Bashlite, The Moon, and Linux/Hydra. Among other IoT-based botnets Bashlite; Mirai's predecessor, Aidra, and Linux/IRCTelnet would follow similar propagation approaches, performing dictionary attacks in order to compromise poorly secured devices over Telnet [4].

3 BotMiner

BotMiner was developed over a decade ago by Gu, Perdisci, and Lee [8]. An analysis and reconstruction of BotMiner with respect to its potential for detecting IoT botnets was discussed in [18]. BotMiner operates on the premise that bots on the same network will operate in patterns distinguishable from ordinary traffic but similar to other bots. Under this assumption, bots can be grouped

by their netflow activity and the alerts that they trigger. BotMiner operates in several stages that involve collecting and clustering netflow and alert data. The clusters found are then put through a cross-clustering process, which is used to establish a bot score for each examined host. The components responsible for this functionality were labelled as the A-Plane Monitor, C-Plane Monitor, A-Plane Clustering, C-Plane Clustering, and Cross-Plane Correlation respectively by both the original work and this reconstruction [8].

The technique, using the A-Plane components, collects alerts and clusters them based on certain categorisations. The original technique used its own categorisation based on bot activity. However, the categorisation was modified in this current work due to its arbitrary nature, as noted in [18]. Bot activities can be dynamic, so any approaches based on a fixed taxonomy will fail over time. The examples provided include scan activity, spam activity, and binary downloads, which may be inadequate or ill-defined for certain attacks. DDoS attacks and click fraud for example can be loosely defined as spam activity, but that does not properly describe them. Botnet crypto-jacking cannot be categorised into any of the provided example categories. Both the original technique and this version use a Snort IDS ruleset to find alerts. The original technique also used a statistical scan anomaly detection engine for finding and categorising scan activities; however, this was deemed out of scope for the experimental setup. Instead, standard Snort categorisations were used to cluster alerts, which proved to be effective. The primary purpose of categorisation was to apply differential weightings to specific categories when determining the final bot score. Gu et al. [8] however did not utilise these. In order to remain true to the original published work, our reconstruction also does not utilise these weights. It is suggested, however, that if these weights can be applied in such a way that requires minimal curation, they may prove a valuable addition to the technique [8, 18].

Alongside alert collection and clustering is netflow collection and clustering, the C-Plane components. The original technique used fcapture to collect netflow, whereas a custom collection program was developed for the reconstruction operating over the same features. Both the original and reconstruction collect TCP and UDP flows. Flow features include time, duration, source IP, source port, destination IP, destination port, and the bidirectional volume of bytes and packets. Once collected, flows were grouped together into C-Flows if they had the same properties, namely: source IP, source port, destination IP, destination port, and protocol. These C-flows were split over 13 intervals where sample distribution features for flows per hour (fph), packets per flow (ppf), average bytes per packets (bpp), and average bytes per second (bps) were collected. For each C-Flow, the collected sample distribution features over the 13 intervals produced a 52-feature dataset. The means and variances of the fph, ppf, bpp, and bps generated an 8-feature dataset, one for each C-Flow. This 8-feature dataset was put through X-means clustering, where X provided the best value of K. The 52-feature dataset was then put through K-means clustering, using the best K value, to produce the netflow clusters [8, 18].

Once the alert clusters and netflow clusters had been calculated, each host found within the clusters was put through a cross-clustering process, utilising the Cross-Plane Correlation component. Equation 1 is used to calculate a final bot likelihood score. The score is compared to a threshold to determine if it is a bot, which is 1 by default. Activity type weights are available but are not used by the original technique as presented in the original paper or the reconstruction created for this experiment.

$$s(h) = \sum_{\substack{i,j \\ j>i \\ t(A_i)\neq t(A_j)}} w(A_i)w(A_j)\frac{|A_i \cap A_j|}{|A_i \cup A_j|} + \sum_{i,k} w(A_i)\frac{|A_i \cap C_k|}{|A_i \cup C_k|} \quad (1)$$

Equation 1: Where A is the sequence of hosts within alert clusters; C is the sequence of hosts within flow clusters; w are activity type weights.

Through analysis and reconstruction of BotMiner, it was determined that the technique has various strengths but also some potential flaws. It is, unlike BotProbe, protocol independent. However, based on the assumptions of the original work and some of the flaws found with alert weighting, it was determined that in botnets using only one device per network, such as Mirai, the technique is likely to fail. However, if the weights can be properly implemented, it would strengthen the ability for BotMiner to find alerts using generic signatures while preventing false positives [18]. The determinations made during the analysis and reconstruction of BotMiner form the basis of the hypotheses proposed in the next section.

通过对BotMiner的分析和重建，我们确定该技术具有各种优势，但也有一些潜在的缺陷。与BotProbe不同，它是独立于协议的。然而，基于原始工作的假设和警报加权所发现的一些缺陷，我们确定在每个网络只使用一个设备的僵尸网络中，如Mirai，该技术很可能会失败。然而，如果权重能够正确实施，它将加强BotMiner使用通用签名查找警报的能力，同时防止误报[18]。在分析和重建BotMiner过程中做出的判断构成了下一节中提出的假设的基础。

4 Experiment

The purpose of the experiment undertaken was to determine the strengths of BotMiner when applied to hosts infected to become part of an IoT-based botnet. Another goal was to ensure that the BotMiner reconstruction could faithfully carry out its original functionality, by successfully detecting infected hosts on traditional networks particularly in instances where the number of bots is greater than one. To achieve those ends aberrantboth IoT-based and traditional networks were simulated. Each of the two networks had multiple kinds of botnets applied to them. To further test BotMiner's capabilities, some instances of the simulated networks would be assigned a device that would frequently emit aberrant behaviour that would purposefully trigger a false alert on a standard Snort IDS ruleset. This was done in order to test BotMiner's resistance to false positives, both on traditional and IoT-based networks.

Two traditional botnets were simulated for the experiment, one used HTTP as its communication protocol and the other used the IRC protocol. The HTTP

进行实验的目的是确定BotMiner在应用于被感染成为基于物联网的僵尸网络的主机时的优势。另一个目的是确保BotMiner的重建能够忠实地执行其原始功能，成功地检测出传统网络上受感染的主机，特别是在机器人数量大于1的情况下。为了达到这些目的，我们模拟了基于物联网的网络和传统网络。这两个网络中的每一个都有多种僵尸网络被应用到它们身上。为了进一步测试BotMiner的能力，模拟网络的一些实例将被分配到一个经常发出异常行为的设备上，该设备将有目的地触发标准Snort IDS规则集的错误警报。这样做是为了测试BotMiner在传统网络和基于物联网的网络上对误报的抵抗力。

botnet had its bots periodically access a command server from which commands would be issued. The IRC botnet would periodically send commands to its bots via an IRC server. Each time the bots would retrieve/receive their (attack) commands they would initiate an attack on a target host.

Both simulated botnets consisted of a fixed number of hosts, of which a certain proportion were infected as bots. All simulated hosts, including the infected hosts, produced conventional HTTP traffic to present a realistic environment for the experiment. The CnC of both botnets routinely commanded the infected hosts to send an ICMP attack to a target server external to the simulated network (a common Denial of Service attack achieved with botnets).

Traffic over the traditional networks would be recorded to include several variation. Variables included the total number of hosts, the number of infected hosts and the inclusion of an aberrant host. The variable number of hosts would allow for examination of any significant difference based on the total number of hosts, of which none was expected. However the total number of hosts was still relevant in relation to examining IoT-based networks due to their greater scale. The total number of infected hosts, however, was expected to make a significant difference due to BotMiner would operate. The aberrant host would test the technique's capabilities in the face of IDS false positives, triggering the IDS with internal ICMP communications, which should trigger the same category of alert as the bots themselves without generating the same flow patterns as the bots. The chosen experimental scenarios accommodate networks with 10 and 100 total devices, 1 and 3 infected devices, and 1 or no aberrant device. Each scenario, for both traditional botnet protocols, produce 16 variations-of which 5 generations were created.

Two IoT-based botnets were also simulated. One utilised a TCP-based bitstream communication protocol and heartbeat similar to the Mirai botnet. The other used the more common IRC communication protocol, still used among many IoT-based botnets. The use of the Mirai-based approach represents the use of non-standard communication protocols which may not trigger the same rates of detection from an IDS. As with the traditional network simulation, the IoT network simulated would include a variable number of devices, infection rates, and aberrant devices to trigger IDS false positives. Whereas the traditional network sizes were 10 and 100 devices, the IoT networks would also include 500 device scenarios to model the scale of IoT networks. The chosen experimental scenarios accommodate networks with 10, 100, and 500 total devices, 1 and 3 infected devices, and 1 or no aberrant device. Each scenario, for both IoT botnet protocols, produced 24 variations-of which 5 generations were created.

The approach of simulating the botnets allows to establish a generic baseline derived from the common attributes of IRC and HTTP botnets, instead of a specific known botnet that fall under either category. Experimentation on established botnet datasets can be conducted alongside the simulated botnets for further validation and will be undertaken in future work. The choice to base one of the simulated IoT-based botnet on Mirai was made due to the challenges it would present to BotMiner, among other techniques. Many common IoT-based

botnets still utilise an IRC-based CnC, whereas Mirai uses a binary stream, an uncommon CnC protocol. Mirai also had the tendency to only infect one device per network due to its randomly directed method of propagation, whereas BotMiner is designed to detect bot activity by groups. Based on the original Mirai bot code, the heartbeat and attack commands would be sent out using a binary stream protocol. The bot would periodically send a heartbeat message to the CnC every 60 seconds, to which the CnC would respond with its own heartbeat. The format of the attack commands would include the attack duration, vector (attack type), targets (and number of them), and attack options (and the number of them) [13].

For the experiment, as with the HTTP and IRC botnets, commands would be periodically sent to initiate an ICMP attack against a single target server. The simulated hosts for the IoT-based botnet would operate differently to the host for the traditional botnets as they would simulate IoT traffic. Whereas the traditional networks would generate randomly distributed HTTP traffic, the IoT network would produce AMQP traffic that would operate over fixed periods. AMQP is an IoT protocol that sends messages through queue exchanges which are then consumed by subscribed workers [1]. This would act as realistic traffic for the simulated IoT network.

These different scenario combinations produced 40 enviroment variations, from which five hours were recorded each. The BotMiner detection technique was applied five times, one for each hour, producing a 200 row dataset. The results included the number of True Positives (Bots found), True Negatives (Non-Bot Hosts found), False Positives (Hosts incorrectly identified as Bots), and False Negatives (Bots not found). Also taken into consideration were the bot scores produced by the technique. If the bot score for a host exists and is greater than 1, that bot score will be indicative of a bot detection. In some instances the scores could also be used to determine the results reliability, where values closer to 1 may indicate a result subject to change under subtle condition changes or the adjustment of the threshold.

4.1 Hypotheses

Based on an examination of the original work and determinations made during the reconstruction process, the following hypotheses were constructed:

- H_1 The BotMiner detection technique shall successfully detect all bots on the simulated networks where the number of bots is greater than one.
- H_2 The BotMiner detection technique shall fail to detect all bots on the simulated networks where there is only one bot.
- H_3 The BotMiner detection technique shall generate false positives where an IDS has been triggered by non-bot aberrant activity on all simulated networks.
- H_4 The total number of hosts on all simulated networks shall have no observable effect on the number true positives or false positives generated by the BotMiner detection technique.

Each hypothesis was designed to operate over all datasets that would be produced from the experiment. H_1 tests the validity of the technique reconstruction. Under experimental conditions the reconstructed BotMiner should be able to detect the bots when more than one is present.

Based on the idea that IoT-based botnets are not constrained to infecting entire networks, as seen with those which randomly select propagation targets, H_2 asserts that singularly infected devices will not be detected. As with H_1 being applied to simulated IoT-based networks, H_2 is also applied to traditional networks. By applying H_1 and H_2 to their opposing targets the cause of detection failure, that BotMiner can only detect multiple bots, can be further validated if hypotheses are proven true.

H_3 tests BotMiner's resistance to false positives, which was observed during its reconstruction as potentially weak. The IDS alert triggered by the aberrant host would be the same category as that of the bot attack. This scenario would provide a realistic situation that would bypass the alert cluster weighting were it applied as the alerts would belong to the same cluster. The distinguishing feature of the traffic would be the flow patterns and the communication endpoints, which would not involve a victim host or CnC server unlike the bots themselves. If H_3 is shown to be true it could reveal a potential weakness of BotMiner regardless of the bots it is applied to, though the result may be different between traditional and IoT-based networks.

H_4 focusses on the total number of hosts to observe the potential differences between traditional networks and larger IoT networks. Though no observation has been made towards BotMiner's capabilities based on the scale of a given target network, the feature of scale with regards to common IoT deployments warrants observation. H_4, if proven true, will demonstrate that BotMiner is, or is not, capable of detection regardless of network scale.

5 Results

The results recorded through tables 1 to 6 include those for both traditional and IoT-based botnet detections. Table 1 and 2 contain the results for traditional HTTP and IRC botnets respectively, while tables 3 to 6 contain results for IoT-based botnets. Tables 3 and 4 contain results for the Bit-Stream botnet based on Mirai, the first table containing results for single bot infections and the latter for multiple bot infections. Tables 5 and 6 pertain to the IRC IoT-based botnet results, divided in the same as 3 and 4. Each hypothesis was addressed with the results found in the tables.

H_1 asserts that where more than a single bot would exist on a network, BotMiner would detect those bots with no false negatives. Such detection would be rendered regardless of protocol or network type, traditional or IoT-based. The primary purpose of this hypothesis would be to demonstrate that the technique reconstruction would be valid by its capabilities compared to the original work, as well demonstrating BotMiner's ability in its intended environment. H_1 was

proven true, all bots were detected on the networks where more than a single infection has occurred.

In contrast to H_1, H_2 would assert that not all bots would be detect in such a case where only a single device would be infected. This hypothesis was designed to demonstrate BotMiner's potential failure in the face of certain IoT-based botnet initiatives that would not necessarily utilise entire networks. Despite assumptions made based on the original work and analysis, this hypothesis was actually proven false. Not only did BotMiner detect all bots where many existed, it also detected all bots where only one existed. This observation indicates that BotMiners strength in this respect was underestimated, and under the right circumstances in can be used to detect IoT-based botnets.

Despite the outcomes of H_1 and H_2 reflecting positively on BotMiner's performance, H_3 would bring insight to another predicted weakness of the detection technique. H_3 was designed to demonstrate that BotMiner, due to its approach towards alert detection and clustering, could be prone to false positives. This hypothesis appears to have been proven true, the majority of the results demonstrate false positives where aberrant device behaviour would trigger an alert. However, not all instances did trigger false positives by the established BotMiner threshold (1). In the majority instances where the IRC protocol had been employed by the botnet, false positives were either not produced or the scores thereof were significantly closer to the threshold that those for other devices detected. It appears that the IRC communications would produce their own alerts, where non-IRC communications would not. The additional alerts would produce a stronger basis for alert clustering among the infected devices, distinguishing them from the aberrant device.

H_4 pertains to the number of total hosts on the network, asserting that that would have no effect on the BotMiner results. Reviewing the results over all tables appears to indicate that H_4 is true, no significant differences in detection rates appears to be present between networks solely based on different total host numbers.

6 Conclusion

The purpose of this work was to assess the capabilities of the traditional botnet detection technique BotMiner when applied on IoT-based botnets, while also validating a reconstruction thereof. The results indicate that the reconstruction of BotMiner operates as per the original work, it appears able to detect bots at a rate of 100% when more than a single bot exists on a network. BotMiner then went on to defy expectations and proceeded to detect bots on networks where only a single bot would be present. The single bot behaviour was meant to reflect the infection patterns of certain IoT-based botnets, including Mirai. BotMiner demonstrated that even when only a single bot exists on a given network, it is able to perform adequately when detecting said bots.

However, despite its strengths it was found that BotMiner still suffers from a notable weakness in that it can be prone to producing false positives. A device

通过表1至表6记录的结果包括传统僵尸网络和基于物联网的僵尸网络检测的结果。表1和表2分别包含了传统HTTP和IRC僵尸网络的结果，而表3至表6包含了基于物联网的僵尸网络的结果。表3和表4包含基于Mirai的Bit-Stream僵尸网络的结果，第一个表包含单个僵尸感染的结果，后者包含多个僵尸感染的结果。表5和表6涉及基于IRC IoT的僵尸网络的结果，划分方式与3和4相同。每个假设都是根据表中的结果来解决的。H1断言，如果一个网络上存在不止一个僵尸，BotMiner将检测到这些僵尸，而且没有假阴性。无论协议或网络类型如何，无论是传统的还是基于物联网的，这种检测都会呈现出来。这一假设的主要目的是证明，与原始工作相比，技术重建的能力是有效的，同时也证明了BotMiner在其预期环境中的能力。H1被证明是正确的，所有的机器人都在发生过一次以上的感染的网络上被检测到。与H1相反，H2认为，在只有一台设备被感染的情况下，并非所有的机器人都会被检测到。这一假设是为了证明BotMiner在面对某些基于物联网的僵尸网络计划时可能会失败，这些僵尸网络不一定会利用整个网络。尽管有基于原始工作和分析的假设，这一假设实际上被证明是错误的。BotMiner不仅检测到了存在许多机器人的地方的所有机器人，而且还检测到了只存在一个机器人的地方的所有机器人。这一观察表明，BotMiners在这方面的实力被低估了，在适当的情况下，它可以被用来检测基于物联网的僵尸网络。尽管H1和H2的结果正面反映了BotMiner的性能，但H3将使人们了解该检测技术的另一个预测的弱点。H3旨在证明BotMiner由于其对警报检测和聚类的方法，可能容易出现假阳性。这一假设似乎已经被证明是正确的，大多数结果显示了假阳性，即异常的设备行为会触发警报。然而，按照BotMiner的既定阈值（1），并非所有情况都会触发误报。在僵尸网络使用IRC协议的大多数情况下，要么没有产生误报，要么其分数明显接近于其他设备检测的阈值。看来，IRC通信会产生自己的警报，而非IRC通信则不会。额外的警报会在受感染的设备中产生一个更强大的警报集群基础，将它们与异常设备区分开来。H4与网络上的主机总数有关，断言这对BotMiner的结果没有影响。审查所有表格的结果似乎表明，H4是真实的，仅根据不同的主机总数，网络之间的检测率似乎没有明显差异。

这项工作的目的是评估传统僵尸网络检测技术BotMiner在应用于基于物联网的僵尸网络时的能力，同时验证其重建。结果表明，BotMiner的重建工作与原来的工作一样，当网络上存在不止一个僵尸时，它似乎能够以100%的速度检测到僵尸。然后，BotMiner违背了预期，继续在只有一个机器人存在的网络上检测机器人。单个僵尸的行为是为了反映某些基于物联网的僵尸网络的感染模式，包括Mirai。BotMiner表明，即使在一个给定的网络上只有一个僵尸存在，它也能够在检测上述僵尸时有足够的表现。然而，尽管BotMiner有其优势，但它仍然有一个明显的弱点，即它可能容易产生假阳性结果。一个设备10 A. Woodiss-Field等人。实验中，在网络模拟中植入了一个会表现出异常行为的设备，它可以触发BotMiner的警报检测组件。在大多数结果中，BotMiner会将该设备错误地标记为机器人。出现的例外情况似乎是由于实际的机器人所产生的警报量而产生的。在这种情况下，如果机器人不使用IRC协议，如Mirai，或者在被分析的网络上没有实际的机器人，那么BotMiner可能容易产生误导性的结果，从而破坏了它本应保护的环境。总体看来，BotMiner可能是一个设计良好的工具，足以检测当代僵尸网络威胁，如基于物联网的威胁。然而，它似乎有一个缺点，那就是如果不进行适当的调整，在许多情况下，由于错误警报的数量，该技术可能会变得不够用。未来的工作将包括将BotMiner应用于外部产生的数据集，以进一步验证本文所产生的结果。然后，同样的实验方法将应用于其他传统僵尸网络检测技术，如BotProbe和BotHunter。最后，基于从BotMiner和其他技术中发现的优势和劣势，未来的工作将包括创建一个僵尸网络检测技术，该技术可以利用现有工作的能力，同时确保其检测基于物联网的僵尸网络的能力。

that would demonstrate aberrant behaviour that could trigger the alert detection component of BotMiner was planted in the network simulations for the experiment. In the majority of the results BotMiner would flag the device as a bot incorrectly. Exceptions that occurred appeared to have been produced due to the volume of alerts produced by the actuals bots. In such cases where bots do not use the IRC protocol, such as Mirai, or there are no actual bots on the network being analysed, then BotMiner may be prone to producing misleading results that could disrupt the environments it would have been intended to protect.

Overall it appears that BotMiner could be a tool potentially well designed enough to detect contemporary botnet threats, such as IoT-based threats. However it does appear to suffer from disadvantage that without proper tuning may render the technique inadequate in many circumstances due to the volume of false alerts.

Future work will include applying BotMiner to externally produced datasets in order to further validate the results produced by this paper. The same experimental approach will then be applied towards other traditional botnet detection techniques such as BotProbe and BotHunter. Finally, based on the strengths and weaknesses found from BotMiner and other technique, future work shall include initiatives toward creating a botnet detection technique that can leverage the abilities of existing work while ensuring their capabilities towards detecting IoT-based botnets.

References

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials **17**(4), 2347–2376 (2015)
2. Alieyan, K., Almomani, A., Abdullah, R., Almutairi, B., Alauthman, M.: Botnet and internet of things (iots): A definition, taxonomy, challenges, and future directions. In: Research Anthology on Combating Denial-of-Service Attacks, pp. 138–150. IGI Global (2021)
3. Angrishi, K.: Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets. arXiv preprint arXiv:1702.03681 (2017)
4. Dange, S., Chatterjee, M.: Iot botnet: the largest threat to the iot network. In: Data Communication and Networks, pp. 137–157. Springer (2020)
5. Elzen, I., Heugten, J.: Techniques for detecting compromised IoT devices. Master's thesis, University of Amsterdam (2017), http://work.delaat.net/rp/2016-2017/p59/report.pdf
6. Eslahi, M., Salleh, R., Anuar, N.B.: 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. IEEE (2012)
7. Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B., Dagon, D.: Peer-to-peer botnets: Overview and case study. HotBots **7**(2007) (2007)
8. Gu, G., Perdisci, R., Zhang, J., Lee, W., et al.: Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In: USENIX Security Symposium. vol. 5, pp. 139–154 (2008)

9. Gu, G., Porras, P.A., Yegneswaran, V., Fong, M.W., Lee, W.: Bothunter: Detecting malware infection through ids-driven dialog correlation. In: Usenix Security. vol. 7, pp. 1–16 (2007)
10. Gu, G., Yegneswaran, V., Porras, P., Stoll, J., Lee, W.: Active botnet probing to identify obscure command and control channels. In: Computer Security Applications Conference, 2009. ACSAC'09. Annual. pp. 241–253. IEEE (2009)
11. Jayasinghe, K., Poravi, G.: A survey of attack instances of cryptojacking targeting cloud infrastructure. In: Proceedings of the 2020 2nd Asia pacific information technology conference. pp. 100–107 (2020)
12. Kumar, A., Lim, T.J.: A secure contained testbed for analyzing iot botnets. In: International Conference on Testbeds and Research Infrastructures. pp. 124–137. Springer (2018)
13. Lester, T.: How does mirai's c&c communicate with its bots? stack. Stack Exchange (2017), https://security.stackexchange.com/questions/151507/how-does-mirais-cc-communicate-with-its-bots.
14. Liu, J., Xiao, Y., Ghaboosi, K., Deng, H., Zhang, J.: Botnet: classification, attacks, detection, tracing, and preventive measures. In: EURASIP journal on wireless communications and networking. vol. 2009, pp. 1184–1187. IEEE Computer Society (2009)
15. Paganini, P.: Http-botnets: The dark side of a standard protocol! Security Affairs (2013), https://securityaffairs.co/wordpress/13747/cyber-crime/http-botnets.html
16. Ramson, S.J., Vishnu, S., Shanmugam, M.: Applications of internet of things (iot)–an overview. In: 2020 5th international conference on devices, circuits and systems (ICDCS). pp. 92–95. IEEE (2020)
17. Shanthi, K., Seenivasan, D.: Detection of botnet by analyzing network traffic flow characteristics using open source tools. In: Intelligent Systems and Control (ISCO), 2015 IEEE 9th International Conference on. pp. 1–5. IEEE (2015)
18. Woodiss-Field, A., Johnstone, M.N.: Assessing the suitability of traditional botnet detection against contemporary threats. In: 2020 Workshop on Emerging Technologies for Security in IoT (ETSecIoT). pp. 18–21. IEEE (2020)
19. Zeidanloo, H.R., Shooshtari, M.J.Z., Amoli, P.V., Safari, M., Zamani, M.: A taxonomy of botnet detection techniques. In: Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on. vol. 2, pp. 158–162. IEEE (2010)

Table 1. Results for BotMiner applied to Traditional Networks with HTTP Botnet Activity (False Positive Scores are marked with *)

Set #	Total Devices	Infected	Abberant	TP	FP	TN	FN	BotMiner Scores
1	10	1	0	1	0	9	0	6.42
2	10	1	0	1	0	9	0	2.40
3	10	1	0	1	0	9	0	1.74
4	10	1	0	1	0	9	0	1.07
5	10	1	0	1	0	9	0	5.34
1	100	1	0	1	0	99	0	10.67
2	100	1	0	1	0	99	0	3.09
3	100	1	0	1	0	99	0	4.19
4	100	1	0	1	0	99	0	2.70
5	100	1	0	1	0	99	0	1.41
1	10	1	1	1	1	8	0	1.89,1.89*
2	10	1	1	1	1	8	0	2.46,2.79*
3	10	1	1	1	1	8	0	2.99,3.73*
4	10	1	1	1	1	8	0	1.62,1.95*
5	10	1	1	1	1	8	0	3.16, 3.56*
1	100	1	1	1	1	98	0	4.49,2.88*
2	100	1	1	1	1	98	0	4.17,3.51*
3	100	1	1	1	1	98	0	5.49,4.04*
4	100	1	1	1	1	98	0	6.78,6.24*
5	100	1	1	1	1	98	0	2.98,2.48*
1	10	3	0	3	0	7	0	6.8 (3)
2	10	3	0	3	0	7	0	4.44,3.44 (2)
3	10	3	0	3	0	7	0	4.43 (3)
4	10	3	0	3	0	7	0	2.98 (2),3.4
5	10	3	0	3	0	7	0	3.88,2.63(2)
1	10	3	1	3	1	6	0	4.01, 3.21,2.21,2.21*
2	10	3	1	3	1	6	0	4.04,2.67,3.24,4.04*
3	10	3	1	3	1	6	0	3.63,3.13,2.55,1.76*
4	10	3	1	3	1	6	0	3.9,4.73,5.15,3.24*
5	10	3	1	3	1	6	0	3.29,4.36(2),3.29*
1	100	3	0	3	0	97	0	13.99,13.66,13.23
2	100	3	0	3	0	97	0	10.77,10.27(2)
3	100	3	0	3	0	97	0	5.15,6.48(2)
4	100	3	0	3	0	97	0	6.69,7.54,7.27
5	100	3	0	3	0	97	0	4.47,5.22,5.72
1	100	3	1	3	1	96	0	7.74,6.33,6.9,4.93*
2	100	3	1	3	1	96	0	12.85,11.85,12.85,7.01*
3	100	3	1	3	1	96	0	8.67 (2),7.67,5.42*
4	100	3	1	3	1	96	0	6.81,8.99,7.75,5.88*
5	100	3	1	3	1	96	0	5.88(3),5.09*

Table 2. Results for BotMiner applied to Traditional Networks with IRC Botnet Activity (False Positive Scores are marked with *)

Set #	Total Devices	Infected	Abberant	TP	FP	TN	FN	BotMiner Scores
1	10	1	0	1	0	9	0	2.40
2	10	1	0	1	0	9	0	3.64
3	10	1	0	1	0	9	0	3.47
4	10	1	0	1	0	9	0	2.48
5	10	1	0	1	0	9	0	3.48
1	100	1	0	1	0	99	0	3.68
2	100	1	0	1	0	99	0	4.04
3	100	1	0	1	0	99	0	2.85
4	100	1	0	1	0	99	0	2.13
5	100	1	0	1	0	99	0	4.04
1	10	1	1	1	1	8	0	2.21,1.14*
2	10	1	1	1	0	9	0	2.15,0.77*
3	10	1	1	1	1	8	0	4.88,1.16*
4	10	1	1	1	0	9	0	2.75,0.81*
5	10	1	1	1	0	9	0	2.36,0.94*
1	100	1	1	1	0	99	0	1.85,0.9*
2	100	1	1	1	1	98	0	3.37,1.18*
3	100	1	1	1	0	99	0	3.83,0.93*
4	100	1	1	1	0	99	0	2.23,0.48*
5	100	1	1	1	1	98	0	5.69,2.22*
1	10	3	0	3	0	7	0	3.62 (2),2.95
2	10	3	0	3	0	7	0	4.67 (2),6.67
3	10	3	0	3	0	7	0	4.43 (3)
4	10	3	0	3	0	7	0	3.47 (2), 4.22
5	10	3	0	3	0	7	0	3.68 (2),4.08
1	10	3	1	3	1	6	0	3.45 (2),3.9,1.42*
2	10	3	1	3	0	7	0	3.18 (3),0.71*
3	10	3	1	3	1	6	0	4.47 (3),1.63*
4	10	3	1	3	0	7	0	3.38 (2),2.93,0.68*
5	10	3	1	3	1	6	0	4.04,4.35 (2),1.71*
1	100	3	0	3	0	7	0	1.89,2.56(2)
2	100	3	0	3	0	7	0	7.01 (2), 6.26
3	100	3	0	3	0	7	0	6.56,7.22,8.49
4	100	3	0	3	0	7	0	4.04 (2),4.44
5	100	3	0	3	0	7	0	6.33 (2),4.79
1	100	3	1	3	0	7	0	2.47 (2),2.95,0.98*
2	100	3	1	3	1	6	0	4.58 (2),5.03,1.31*
3	100	3	1	3	1	6	0	6.39 (2),5.51,2.39*
4	100	3	1	3	1	6	0	3.55 (3),1.2*
5	100	3	1	3	1	6	0	4.81,5.99 (2),2.33*

Table 3. Results for BotMiner applied to IoT Networks with Bit-Stream Botnet Activity with Single Device Infections (False Positive Scores are marked with *)

Set #	Total Devices	Infected	Abberant	TP	FP	TN	FN	BotMiner Scores
1	10	1	0	1	0	9	0	6.02
2	10	1	0	1	0	9	0	14.19
3	10	1	0	1	0	9	0	13.20
4	10	1	0	1	0	9	0	23.97
5	10	1	0	1	0	9	0	17.57
1	100	1	0	1	0	99	0	2.81
2	100	1	0	1	0	99	0	9.32
3	100	1	0	1	0	99	0	16.31
4	100	1	0	1	0	99	0	9.77
5	100	1	0	1	0	99	0	10.67
1	500	1	0	1	0	499	0	3.16
2	500	1	0	1	0	499	0	4.74
3	500	1	0	1	0	499	0	11.85
4	500	1	0	1	0	499	0	8.71
5	500	1	0	1	0	499	0	6.25
1	10	1	1	1	1	8	0	8.4,1*
2	10	1	1	1	1	8	0	9.42,1.67*
3	10	1	1	1	1	8	0	13.97,1*
4	10	1	1	1	1	8	0	14.37,1*
5	10	1	1	1	1	8	0	12.12,1.67*
1	100	1	1	1	1	98	0	3.32,1.1*
2	100	1	1	1	1	98	0	4.12,1.29*
3	100	1	1	1	1	98	0	9.12,1.13*
4	100	1	1	1	1	98	0	6.18,1*
5	100	1	1	1	1	98	0	5.35,1*
1	500	1	1	1	1	498	0	3.32,1.1*
2	500	1	1	1	1	498	0	4.19,1.29*
3	500	1	1	1	1	498	0	9.12,1.13*
4	500	1	1	1	1	498	0	6.18,1.1*
5	500	1	1	1	1	498	0	5.35,1*

Table 4. Results for BotMiner applied to IoT Networks with Bit-Stream Botnet Activity with Mutliple Device Infections (False Positive Scores are marked with *)

Set #	Total Devices	Infected	Abberant	TP	FP	TN	FN	BotMiner Scores
1	10	3	0	3	0	7	0	19.65(2), 22.05
2	10	3	0	3	0	7	0	23.76 (2),26.26
3	10	3	0	3	0	7	0	34.87 (2),36.87
4	10	3	0	3	0	7	0	31.52,28.02,30.52
5	10	3	0	3	0	7	0	22.71 (2),26.71
1	100	3	0	3	0	97	0	7.3,7.64,7.42
2	100	3	0	3	0	97	0	10.06,12.37,11.87
3	100	3	0	3	0	97	0	21.11,22.43,21.68
4	100	3	0	3	0	97	0	18.36,21.17,21.86
5	100	3	0	3	0	97	0	17.44,14.88,16.77
1	500	3	0	3	0	497	0	10.15(3)
2	500	3	0	3	0	497	0	11.28,12.89(2)
3	500	3	0	3	0	497	0	22.05,23.14,22.81
4	500	3	0	3	0	497	0	20.21,20.61,22.77
5	500	3	0	3	0	497	0	15.81,14.99 (2)
1	10	3	1	3	1	6	0	16.53,17.53(2),2.2*
2	10	3	1	3	1	6	0	19.25,24.45,21.25,1*
3	10	3	1	3	1	6	0	25.61,29.21,29.61,1.6*
4	10	3	1	3	1	6	0	22.57,21.11,23.37,1*
5	10	3	1	3	1	6	0	24.26,25.46,21.86,1*
1	100	3	1	3	1	96	0	7 (2),7.29,1.2*
2	100	3	1	3	1	96	0	9.69,11.63,10.12,1*
3	100	3	1	3	1	96	0	20.24,20.57,20.87,1.4*
4	100	3	1	3	1	96	0	18.25,18.77,19.27,1*
5	100	3	1	3	1	96	0	15.44,13.42,15.29,1.3*
1	500	3	1	3	1	496	0	7 (2),7.29,1.2*
2	500	3	1	3	1	496	0	9.69,11.63,10.12,1*
3	500	3	1	3	1	496	0	20.24,20.57,20.87,1.4*
4	500	3	1	3	1	496	0	18.25,18.77,19.27,1*
5	500	3	1	3	1	496	0	15.44,13.42,15.29,1.3*

Table 5. Results for BotMiner applied to IoT Networks with IRC Botnet Activity with Single Device Infections (False Positive Scores are marked with *)

Set #	Total Devices	Infected	Abberant	TP	FP	TN	FN	BotMiner Scores
1	10	1	0	1	0	9	0	3.75
2	10	1	0	1	0	9	0	3.00
3	10	1	0	1	0	9	0	5.25
4	10	1	0	1	0	9	0	5.50
5	10	1	0	1	0	9	0	6.00
1	100	1	0	1	0	99	0	3.38
2	100	1	0	1	0	99	0	4.68
3	100	1	0	1	0	99	0	4.06
4	100	1	0	1	0	99	0	3.88
5	100	1	0	1	0	99	0	4.92
1	500	1	0	1	0	499	0	3.15
2	500	1	0	1	0	499	0	3.97
3	500	1	0	1	0	499	0	4.38
4	500	1	0	1	0	499	0	3.50
5	500	1	0	1	0	499	0	3.26
1	10	1	1	1	0	9	0	2,0.5*
2	10	1	1	1	1	8	0	4.8,2*
3	10	1	1	1	0	9	0	2,0*
4	10	1	1	1	0	9	0	4.98,0*
5	10	1	1	1	0	9	0	3.92,0*
1	100	1	1	1	0	99	0	2.57,0.71*
2	100	1	1	1	0	99	0	2.32,0.5*
3	100	1	1	1	0	99	0	2.57,0*
4	100	1	1	1	0	99	0	2.68,0*
5	100	1	1	1	0	99	0	4.13,0.11*
1	500	1	1	1	0	499	0	2.28,0.5*
2	500	1	1	1	0	499	0	2.76,0.5*
3	500	1	1	1	0	499	0	3.57,0.11*
4	500	1	1	1	0	499	0	2.99,0*
5	500	1	1	1	0	499	0	2.3,0.01*

Table 6. Results for BotMiner applied to IoT Networks with IRC Botnet Activity with Mutliple Device Infections (False Positive Scores are marked with *)

Set #	Total Devices	Infected	Abberant	TP	FP	TN	FN	BotMiner Scores
1	10	3	0	3	0	7	0	7.2,6 (2)
2	10	3	0	3	0	7	0	4.8 (3)
3	10	3	0	3	0	7	0	6 (2),4.5
4	10	3	0	3	0	7	0	5 (2),3
5	10	3	0	3	0	7	0	3.75,4.95 (2)
1	100	3	0	3	0	97	0	3.9 (3)
2	100	3	0	3	0	97	0	6.3 (3)
3	100	3	0	3	0	97	0	6.36 (2),3.6
4	100	3	0	3	0	97	0	5.27 (2),3.92
5	100	3	0	3	0	97	0	5.88 (2),6.3
1	500	3	0	3	0	497	0	3.33 (3)
2	500	3	0	3	0	497	0	3.4 (2),3.69
3	500	3	0	3	0	497	0	6.4 (2),3.49
4	500	3	0	3	0	497	0	4.15,3.48,5.44
5	500	3	0	3	0	497	0	4.36 (2),4.04
1	10	3	1	3	0	7	0	5 (3), 0.75*
2	10	3	1	3	0	7	0	4.45 (3),0.75*
3	10	3	1	3	0	7	0	7 (2),5.6,0*
4	10	3	1	3	0	7	0	4.59(2),2.5,0*
5	10	3	1	3	0	7	0	4.6 (2),5.77,0*
1	100	3	1	3	1	96	0	3.29 (3), 1.29*
2	100	3	1	3	0	97	0	7.51 (3),0.75*
3	100	3	1	3	0	97	0	6.6 (2),4.38,0.09*
4	100	3	1	3	0	97	0	4.84 (2), 3.46, 0.25*
5	100	3	1	3	0	97	0	5.7,4.74 (2),0.3*
1	500	3	1	3	1	496	0	3.09 (3), 1.03*
2	500	3	1	3	0	497	0	4.39 (3),0.83*
3	500	3	1	3	0	497	0	5.18 (2),3.12,0.03*
4	500	3	1	3	0	497	0	4.32 (2),2.95,0*
5	500	3	1	3	0	497	0	3.98 (3),0.06*