

Northumbria Research Link

Citation: Debashi, Mohamed and Vickers, Paul (2018) Sonification of Network Traffic for Detecting and Learning About Botnet Behavior. IEEE Access, 6. pp. 33826-33839. ISSN 2169-3536

Published by: IEEE

URL: <http://dx.doi.org/10.1109/ACCESS.2018.2847349>
<<http://dx.doi.org/10.1109/ACCESS.2018.2847349>>

This version was downloaded from Northumbria Research Link:
<http://nrl.northumbria.ac.uk/id/eprint/34521/>

Northumbria University has developed Northumbria Research Link (NRL) to enable users to access the University's research output. Copyright © and moral rights for items on NRL are retained by the individual author(s) and/or other copyright owners. Single copies of full items can be reproduced, displayed or performed, and given to third parties in any format or medium for personal research or study, educational, or not-for-profit purposes without prior permission or charge, provided the authors, title and full bibliographic details are given, as well as a hyperlink and/or URL to the original metadata page. The content must not be changed in any way. Full items must not be sold commercially in any format or medium without formal permission of the copyright holder. The full policy is available online: <http://nrl.northumbria.ac.uk/policies.html>

This document may differ from the final, published version of the research and has been made available online in accordance with publisher policies. To read and/or cite from the published version of the research, please visit the publisher's website (a subscription may be required.)

Received February 23, 2018, accepted May 19, 2018, date of publication June 14, 2018, date of current version July 6, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2847349

Sonification of Network Traffic for Detecting and Learning About Botnet Behavior

MOHAMED DEBASHI AND PAUL VICKERS[✉]

Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K.

Corresponding author: Paul Vickers (paul.vickers@northumbria.ac.uk)

This work was supported by the Ministry of Higher Education and Scientific Research of Libya under Grant (ST-10233-2014) (395/2012).

ABSTRACT Today's computer networks are under increasing threat from malicious activity. Botnets (networks of remotely controlled computers, or "bots") operate in such a way that their activity superficially resembles normal network traffic which makes their behavior hard to detect by current intrusion detection systems (IDS). Therefore, new monitoring techniques are needed to enable network operators to detect botnet activity quickly and in real time. Here, we show a sonification technique using the SoNSTAR system that maps characteristics of network traffic to a real-time soundscape enabling an operator to hear and detect botnet activity. A case study demonstrated how using traffic log files alongside the interactive SoNSTAR system enabled the identification of new traffic patterns characteristic of botnet behavior and subsequently the effective targeting and real-time detection of botnet activity by a human operator. An experiment using the 11.39 GiB ISOT botnet data set, containing labeled botnet traffic data, compared the SoNSTAR system with three leading machine learning-based traffic classifiers in a botnet activity detection test. SoNSTAR demonstrated greater accuracy (99.92%), precision (97.1%), and recall (99.5%) and much lower false positive rates (0.007%) than the other techniques. The knowledge generated about characteristic botnet behaviors could be used in the development of future IDSs.

INDEX TERMS Botnet detection, intrusion detection systems, network monitoring, situational awareness, sonification.

I. INTRODUCTION AND MOTIVATION

Network administrators commonly use a combination of Intrusion Detection System (IDS) software and sensors that inspect traffic on the network and wait for anomalous events to occur. Intrusions are defined as 'attempts to compromise confidentiality, integrity, or availability of data, or to bypass the security mechanisms of an IT system' [1, p. 147]. Network security monitoring is a crucial task in protecting organisational infrastructure from today's hidden and increasingly complex intrusions and attack patterns [2].

In 2015, IBM analysts reported a number of types of attempt to break into networks and organisational infrastructures, such as exploiting 'a vulnerability to inject command code into software, exploiting a backdoor, or bombarding a system with random passwords in hopes that one will work' [3, p. 3]. They declared that the majority of networks of all sizes are at risk and that 'CISOs and security leaders are now looking for fundamental ways to influence and improve both their own programs and established best practices' [3, p. 14].

This article looks at the problem of detecting botnet traffic. A botnet is a network of remotely controlled devices, or 'bots', such as personal computers and smartphones, whose security has been breached and control access given to a third party. The botnet controller directs the activities of the bots through messages sent via standard network protocols. Botnets are used for various malicious purposes such as conducting distributed denial of service (DDoS) attacks, spreading spam, spying, and stealing personal information [4]. They propagate over legitimate communication connections and, because an individual bot may send only a few packets to the host under attack, the volume of traffic looks normal.

Behind every attack is an underlying motive, and knowing what it is might allow administrators to anticipate attacks that might be deployed against their networks [5]. Axelsson and Sands suggested that in 'dealing with the more imaginative threats, a human operator needs to be in the loop and in order to be effective there should be tool support that enables her to quickly gain an understanding of the

situation' [6, p. 26]. SoNSTAR (Sonification of Networks for SiTuational AwaReness) [7], [8] is a monitoring tool developed to complement existing IDSs to provide another degree of flow granularity to operators, helping them to understand how a specific network operates and behaves. It leverages the human auditory system's ability to detect and recognize sonic patterns by mapping features of network traffic to sound, thus enabling an operator to listen to their network.

SoNSTAR reads every single packet entering and leaving a network and associates it with a flow (a sequence of packets between a source and destination host). It then conveys the state of each flow sonically while simultaneously updating a log file with information that helps the operator to understand the reasons for the sounds generated; for instance, when the operator wants to refer to certain behavior or pattern repetition or to find the IP addresses of the devices implicated in that behavior. This paper describes a sonification approach using SoNSTAR to target botnet behavior in TCP traffic.

A. RELATED WORK

Existing IDS technologies rely on a variety of techniques to detect botnets, including identifying repetitions of requests, statistical methods [9], [10] and entropy detection [11], and all such techniques tend to 'collect flow information from bots to depict their behavior' [12, p. 976]. A challenge facing all detection techniques is to validate them on real networks which vary from the test environments in which they were developed [12].

To-date, using visualization techniques to support botnet detection has received only modest attention. Seo *et al.* [13] proposed a security visualization tool called CCSvis to target botnet behavior based on Domain Name System (DNS) traffic. CCSvis presents visualizations of traffic using cylindrical coordinates to enable a human operator to identify botnet behaviors and patterns. Thus, detection is a cooperative activity involving both human and machine.

Kim *et al.* [14] also visualized DNS traffic with the aim of detecting botnets before they start carrying out their attacks. They defined four patterns of graphs as botnet signatures which can be identified by a human operator. Experimental results suggested that visualization could be used to detect both known and unknown botnet types.

Visual Threat Monitor [15] is a flow-based system which combines data mining and visualization to enhance botnet traffic detection. Its visualization method uses processed and selected data rather than raw data and the outputs consist of correlations, statistical analysis, clustering, aggregation, and visualization.

While some network sonification work has been reported [16]–[22] the technique has not yet been applied specifically to botnet detection. Below we show how the SoNSTAR system may be used to complement an existing IDS by sonifying network traffic in such a way as to enable botnet behavior to be detected and identified by a human

operator without the use of any botnet detection algorithms or machine learning classifiers.

B. MAIN RESULTS

This work deals with the extension of SoNSTAR by mapping TCP traffic flow features to sound such that it enables the human operator to recognize botnet activity and patterns without the need to manually inspect the traffic's content. The first contribution is the construction of four new traffic features that target parallel, horizontal, distributed and repetitive flow behaviors plus four new algorithms to process event feature conditions.

The second contribution is the discovery and definition of six patterns of botnet behavior based on IP flow. We created a new flow type called IP flow [7] which is identified within a certain time period by its source and destination IP addresses and protocol (as opposed to a traffic flow which is further differentiated by port number [23]). The significance of this discovery is that botnets exhibit unique repetitive IP flow patterns which can be used to detect them at the network layer instead of the growing demand to detect botnets at the application layer. The third contribution consists of using IP flow patterns for classification instead of using packet patterns, which opens up a new path of research to find better ways to develop IDSs in the future based on IP flows. Finally, as a fourth contribution, the proposed sonification tool (SoNSTAR) is an interactive, flexible, and scalable approach for botnet traffic detection that can be adjusted according to the understanding of the human operator and future security demands, and this is the first sonification solution to target botnet detection.

II. BOTNET SONIFICATION USING TCP

To consider how to apply sonification to botnet activity the characteristics of botnet traffic and behavior must first be understood.

A. CHARACTERISTICS OF A BOTNET

TCP botnet traffic has certain characteristics that can be distinguished amongst legitimate traffic as follows [24], [25]:

- 1) Botnet lifecycles go through the same five stages. The first stage is infection and propagation where the botmaster infects new targets such as computers or servers to become bots. Propagation mechanisms refer to the method used to expand and search for new machines. They consist of horizontal scans, vertical scans, coordinated scans and other sophisticated propagation methods. A vertical scan is described as scanning a single host across a defined range of ports. Horizontal scans are where a single port is scanned across a defined range of hosts. The second is rallying, where a bot connects to the C&C (command-and-control) server or the bot receives updates such as a list of C&C IP servers. The third is command and report, where the bot connects to the C&C server to receive commands and to send its activity

reports. The fourth is ‘abandon’ where a bot becomes unusable. The fifth is securing, where the botmaster tries to conceal its bots from security detection systems.

- 2) The C&C mechanism has three architectures. The first is centralized, where the botmaster communicates with bots through a central C&C server. The second form is decentralized, where bots also act as C&C servers based on a peer-to-peer (P2P) network model. The third type is hybrid, where the botmaster can use any applicable protocols and architectures to implement its model.
- 3) Botnets perform several malicious activities depending on their size (large- or small-scale) such as DDoS and spamming.

However, botnets are a rapidly evolving phenomenon that is still not well understood. Moreover, bots mostly connect with C&C servers or other infected nodes which act as C&C servers using normal traffic communication patterns which are repeated again and again. Bots are relatively consistent in the repetitive communication mechanisms employed between them when they are using a P2P network model. Bots may scan the network to collect information that may help the botmaster to prepare for future attacks such as infecting other devices. Also, bots might be part of a botnet launching an attack such as a DDoS where they continuously communicate with an external host using similarly repetitive traffic patterns. Accordingly, SoNSTAR was extended to extract features and to enable users to create events and mappings to explore and identify bot behaviors.

B. EXPLORING TRAFFIC FOR BOTNET DETECTION

Exploring traffic aims to identify events useful in the recognition of botnet behavior. Since botnet traffic does not have a specific behavior, we aimed to define more features that allow the discovery of events which might be part of a botnet’s behavior. This method allows the user to assign sounds to different events based on his/her knowledge and experience and the literature on botnets.

Botnets exhibit stealthy behavior in several ways including distributed behavior, parallel behavior, repetitive behavior and stealth scanning. Stealth scanning is described as using vertical or horizontal scans with low frequency to avoid detection [26]. Parallel flow behavior occurs when a single host establishes flow connections with several network hosts on several ports. Distributed flow behavior is where a local host receives several flows from different external hosts on several ports. Repetitive flow behavior is where repeated flow patterns are observed and which are caused by bots repeatedly carrying out the same task (performed automatically or to a schedule) over the internet [27]. Although the previous SoNSTAR design is capable of addressing various traffic behaviors, horizontal, distributed, parallel and repetitive stealth behaviors were not addressed.

Since, on the face of it, botnet flows resemble normal traffic, additional features need to be considered to support the recognition of botnet activity. The method consists of using sonification to monitor events which are suspected as

evidence of botnet activity. The operator then reviews the log files corresponding to these suspicious events and creates a pattern based on IP flow features to match the selected events. This pattern can then be associated with a specific sound in SoNSTAR allowing any occurrences to be monitored. For example, in normal traffic behavior, it is virtually impossible to find identical IP flow patterns repeated within a single time window.

Botnet activity could consist of several different combinations of traffic events. It is left to the human operator to explore and decide which events might be part of botnet behavior. The correlation of events is useful for monitoring botnets by looking at botnet characteristics. This approach is very effective when performing real-time monitoring. The human mind correlates events to understand situations based on events’ sounds, sequence, occurrence, and other factors such as the nature of the network and the motivation of expected attacks. There is no specific rule to be used to recognize botnets but a human can look for various events and behaviors and tune thresholds according to his or her accumulated knowledge and understanding of botnet activity such as stealthy and repetitive behaviors, which indicate botnet activity.

C. SoNSTAR UPDATED DESIGN

Sonification has the potential to assist in the discovery of patterns of botnet network activity and relationships between seemingly disparate security events, though little has been done to leverage sonification technologies in current practice.

SoNSTAR [7] is a tool for supporting the situational awareness of network administrators that provides a real-time auditory representation of TCP/IP traffic. SoNSTAR works by inspecting the status flags of TCP/IP packet headers to identify traffic features; combinations of features then trigger the playback of the different elements of a soundscape. By noticing the timing, loudness, and sequence of the various sounds an administrator can monitor the network and become aware of possible malicious behavior without needing to continuously look at a visual display. Once an operator becomes used to the sound of a network’s normal behavior, the system allows small changes to be detected in much the same way that an experienced mechanic is able to troubleshoot problems with an engine through listening to the sounds it makes. Full details of the system and its effect on user workload are described in an earlier paper [7] and the system and example audio files can be accessed at the project repository [8].

SoNSTAR was designed to fit the work practices and operational environments of network security monitoring analysts in order to raise their situational awareness through reflecting human understanding in the monitoring process. The solution starts by extending SoNSTAR to enable the operator to explore a network’s botnet traffic patterns. Publicly-available labeled botnet datasets were used to demonstrate the technique. To address the problem of botnet detection SoNSTAR was extended as follows.

TABLE 1. Events-to-sound mappings.

No	Feature Conditions	Sound
1	SYN in IPs <30 and SYN ACK out IPs >0 and ACK in IPs >0 and RST out IPs <10	Forest bird
2	SYN in IPs >10 and SYN in IPs <30 and PSH ACK out IPs == 0 and RST out IPs >0	Rain on roof
3	SYN in IPs >8 and SYN ACK out IPs <4 and PSH ACK out IPs <50	Rain on roof
4	SYN in IPs >300 and SYN ACK out IPs <20 and PSH ACK out IPs <300	Thunder
5	SYN out IPs >10 and SYN ACK in IPs <3 and PSH ACK out IPs == 0 and RST in IPs >0	Rain
6	SYN out IPs <30 and SYN ACK in IPs >0 and PSH ACK out IPs >0	Forest bird
7	ACK in IPs >1 and RST out IPs >0 and the rest of IP flow feature == 0	Seagulls
8	ACK out IPs >1 and RST in IPs >0 and the rest of IP flow feature == 0	Loon
9	FIN in IPs >9 and FIN in IPs >SYN out IPs and FIN in IPs >SYN in IPs and F 4 >10 and PSH ACK out IPs <2 and PSH ACK in IPs <2	Cricket
10	FIN in IPs <50 and (FIN in IPs <= SYN out IPs or FIN in <= SYN in IPs)	Forest bird
11	FIN out IPs >9 and FIN out IPs >SYN out IPs and FIN out IPs >SYN in IPs and FC 3 >10 and PSH ACK out IPs <2 and PSH ACK in IPs <2	Sheep
12	FC 7 >9 and PSH ACK out IPs <5 and PSH ACK in IPs <5	Owl
13	NULL in IPs or NULL out IPs >0	Frog
14	URG PSH FIN in IPs or URG PSH FIN out IPs >0	Wolf
15	LAND in IPs or LAND out IPs >0	Beach
16	RST in IPs >30 and ACK in IPs <100 and PSH ACK out IPs or PSH ACK in IPs <2	Wind on grass
17	RST out IPs >30 and ACK out IPs <100 and PSH ACK out IPs or PSH ACK in IPs <2	Wind on grass
18	FC 1 >4 and PSH ACK out IPs or PSH ACK in IPs == 0	Fountain
19	FC 2 >4 and PSH ACK out IPs or PSH ACK in IPs == 0	Heavy rain
20	RST out IPs >30 and FC 5 <FC 14 and ACK out IPs <5 and PSH ACK out IPs <2	Wind
21	RST in IPs >5 and FC 6 <FC 15 and ACK in IPs <5 and PSH ACK in IPs <2	Wind
22	Flow Counter >1000	Fire
23	IPs Flow Counter >600	Fire
24	Src addr1 A count >200	Mosquito
25	Src addr1 A count >50 and Identical packet counts1 >250	Mouse squeaking
26	Dst addr2 A count >200	Bee Colony
27	Dst addr2 A count >50 and Identical packet counts2 >250	Rats (multiple) squeak
28	Src addr count3 >85 and Dst port count3 >95	Spring Peeper
29	Dst addr count4 >85 and Dst port count4 >95	Grass hopper

1) FEATURE EXTRACTOR – SELECTED TCP PARAMETERS

SoNSTAR works by selecting combinations of IP traffic features and mapping these feature combinations to discrete sounds in its soundscape. Table 1 shows 29 of these mappings (see the project repository [8] for further details). Four new arrays were implemented to collect more features that facilitate the targeting of repetitive parallel, horizontal and distributed flow behavior.

The first array collects features which are used to identify any local hosts that attempt to perform parallel repetitive flows within the local network during a time window. The new feature events provide interesting knowledge about IP addresses and port number mechanisms within network traffic. Fig. 1 illustrates the features collected to target local source IP addresses which perform internal network parallel repetitive behaviors. The first column in the array contains a local source host, the next column contains a destination port number, and the next column contains the number of packets sent from the local source host to the destination port number. The features collected by the feature extractor into array 1 are listed in Table 2.

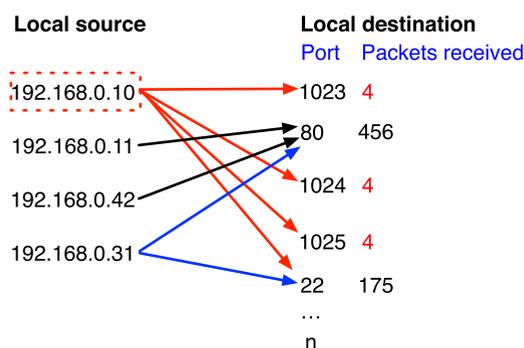


FIGURE 1. Features for targeting the behavior of local parallel repetitive flows. Local host 192.168.0.10 is repeatedly sending the same four packets to certain ports on multiple destination hosts on the network. The features collected are: list of local source hosts, destination port numbers, count of packets sent to each port.

The second array collects features that help to identify any local hosts that experience distributed repetitive flows behavior from external hosts during a time window. Fig. 2 illustrates the features collected to target local IP addresses

TABLE 2. The features collected in array 1.

Feature	Description
1 Src addr1 A	Local host A (Source IP) has sent packets to another local host B during current time window period.
2 Dst port1 B	The destination port number on local host B has received one or more packets local host A identified in element 1.
3 Packet count1	The number of packets sent from local host A to the local host B through the destination port number identified in element 2.

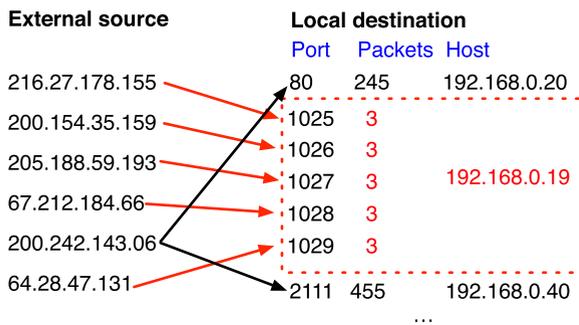


FIGURE 2. Features for targeting the behavior of incoming distributed repetitive flows. Multiple external hosts are targeting ports on local hosts with the same three packets. The features collected are: list of local destination addresses, destination port numbers, count of packets sent to each port.

which have experienced distributed repetitive flow behavior. The first column in the array contains a local destination host, the next column contains a destination port number, and the next column contains the number of packets received from the external source host by the local destination host through the destination port number. The features collected by the feature extractor into array 2 are listed in Table 3.

TABLE 3. The features collected by algorithm 2.

Feature	Description
1 Dst addr2 B	Local host B has received one or more packets or from external hosts during the current time window period.
2 Dst port2 B	The destination port number on local host B identified in element 1 which has received one or more packets from an external host.
3 Packet count2	The count of packets received through the destination port identified in element 2.

The third array collects features that help to identify any local hosts receiving distributed flows (such as a hidden scan behavior and malicious distributed flows as legitimate queries) from external hosts during a time window. Fig. 3 illustrates the features collected to identify local hosts receiving flows from external hosts such as stealth horizontal and vertical scans or other distributed activity. The features are

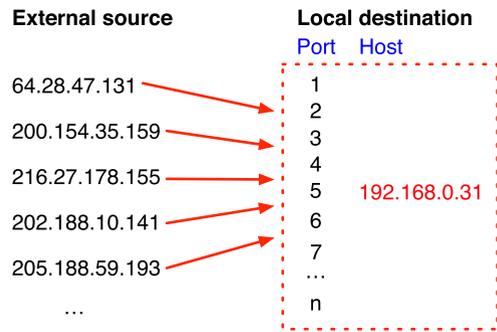


FIGURE 3. Features for targeting incoming vertical flow scans. Multiple external hosts are targeting ports on a single local host. The features collected are: list and count of local ports, list and count of source addresses.

collected for all local destination hosts in every time window. The features collected by the feature extractor into array 3 are listed in Table 4.

TABLE 4. The features collected in array 3.

Feature	Description
1 Dst addr3	Local host B has received one or more packets from a local and external hosts.
2 Src addr3 list	List of source addresses that have sent packets to local host B (element 1).
3 Src addr count3	The count of source addresses in the list from element 2.
4 Dst port list3	The list of destination ports on local host B which received one or more packets from sources in the hosts list.
5 Dst port count3	The count of destination ports in the list identified in element 4.

The fourth array collects features that help to identify any local hosts attempting to perform local horizontal and parallel activity within the local network during a time window period. Fig. 4 illustrates the features collected to target internal network horizontal scans and parallel flow activities, and the features are listed in Table 5.

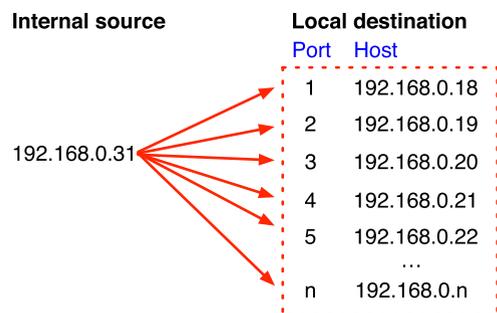


FIGURE 4. Features for targeting local horizontal and vertical scans and parallel flow activities. The features collected are: list and count of destination hosts, list and count of destination ports.

TABLE 5. The features collected in array 4.

Feature	Description
1 Src addr4	Local source host A has sent one or more packets to a local destination host.
2 Dst addr list4	The list of local destination hosts that have received one or more packets from local host A.
3 Dst addr count4	Count of destination hosts in the list from element 2.
4 Dst port list4	The list of destination ports of the local hosts from element 2 which received one or more packets from local host A.
5 Dst port count4	Count of destination ports in the list from element 4.

2) FEATURE COMBINER

At the end of each time window, the new features extracted using algorithms 1 to 4, together with the IP flow features and traffic flow features are passed to SoNSTAR’s feature combiner. The feature combiner uses the IP flow and traffic flow features to obtain newly discovered combinations to be used to create new events.

Since the existing SoNSTAR design already has eight combinations, we have updated the design by adding seven new features as shown in Table 6.

TABLE 6. Feature combinations.

Feature Combination	Description of Feature
FC 9	Result of (SYN out IPs) + (FIN in IPs)
FC 10	Result of (ACK in IPs) – (ACK out IPs)
FC 11	Result of (FIN in IPs) + (FIN out IPs)
FC 12	Result of (PSH ACK in IPs) – 1
FC 13	Result of (ACK in IPs) + (FC 9)
FC 14	Result of (RST out IPs) – 1
FC 15	Result of (RST in IPs) – 1

3) SONIFICATION

In this stage, SoNSTAR assigns events using the extracted features and then assigns recorded sounds to them. Here, event conditions can be modified, new events can be created, and threshold values can be changed. Most of the interaction of the operator with SoNSTAR to explore and construct new events and to explore malicious behaviors and botnet patterns occurs at this stage.

SoNSTAR’s real-time interaction capability allows the listener to choose to listen to some of the event sounds or to specifically targeted events representing certain behaviors and to ignore others or to change event conditions and assigned sounds.

4) SOUND MAPPING AND DESIGN

Sounds are mapped by assigning recorded sounds to events. Most of the events in the previous design have been mapped

Algorithm 1 Process the Features of Array 1

```

Time-window period ended
Call Function with Array1 and index 1
Open logs text file1 to write
for pointer1 <= array index do                                ▷ Get all row
information in the list
    Get local source address (Source IP) Column1 of
Array1
    Get Sent packet count (Packet count) Column3 of
Array1
    Rest (Src Addr1 Count1) = 0 and (Identical Packets
Count1) = 0
    State1 = False
    for pointer2 <= array index do                                ▷ Compare with all
sources list
        Get next source IP in the array (Src ip)
        Get next packet count in the array (P count)
        if Source IP == source ip then
            increase (Src Addr1 Count1) by 1
            if Src Addr1 Count1 >= 50 then
                State1 = True
            end if
            if Src Addr1 Count1 >= 200 then
                Write to logs file1, Anomaly
                Send message of Event 24 to Max/MSP
Patch
            end if
        end if
        if Packet count == P count, And Packet
count <= 15 then
            increase (Identical Packets Count1) by 1
            if State1 == True, And Identical Packets
Count1 >= 250 then
                Write to logs file1, Anomaly
                Send message of Event 25 to Max/MSP
Patch
            end if
        end if
    end for
end for
    
```

to the same original sounds with some modifications. A few events were dropped from the original design. In addition, new events are assigned to new recorded sounds. These recorded sounds are set to allow the user to recognize botnet events. SoNSTAR events are mapped to represent the occurrence of events derived from the SoNSTAR features. Table 1 lists the new feature-to-sound mappings.

At the end of each time window, Algorithm 1 checks whether any local source IP (Table 2, row 1) has created 200 or more flows, and this feature is called “Src addr1 A count”. It also checks whether any packet count (Table 2, row 3) is less than 15 packets and is repeated 250 times during the time window. It counts the number of identical

packet counts which has less than 15 packets passed through the destination ports. This feature is called “Identical packet counts1”.

Algorithm 2 Process the Features of Array 2

Time-window period ended

Call Function with Array2 and index 2

Open logs text file2 to write

for *pointer1* \leq *array index* **do** \triangleright Get all row information in the list

 Get destination address (Dest IP) Column1 of Array2

 Get total received packet count (Packet Count) Column3 of Array2

 Rest (*Dst addr2 count2*) = 0 and (*Identical Packets Count2*) = 0

for *pointer2* \leq *array index* **do** \triangleright compare with all dest IP's

 Get next destination IP in the array (dest ip)

 Get next packet count received in the array (*p count*)

if *Dest IP* == *dest ip* **then**

 increase (*Dst addr2 count2*) by 1

if *Dst addr2 count2* \geq 50 **then**

 State1 = True

end if

if *Dst addr2 count2* \geq 200 **then**

 Write to logs file2, Anomaly

 Send message of Event 26 to Max/MSP

Patch

end if

end if

if *Packet Count* == *p count*, And *packet count* < 15 **then**

 increase (*Identical Packets count2*) by 1

if State1 == True, And *Identical Packets*

Count2 \geq 250 **then**

 Write to logs file2, Anomaly

 Send message of Event 27 to Max/MSP

Patch

end if

end if

end for

end for

Algorithm 2 checks whether any local destination IP (Table 3, row 1) has created 200 or more flows and the generated feature is called “Src addr1 A count”. It also checks whether any packet count (Table 3, row 3) is less than 15 packets and is repeated 250 times during the time window. It counts the number of identical packet counts which have less than 15 packets passed through the destinations ports. This feature is called “Identical packet counts2”.

Algorithm 3 checks each local destination host (Table 4, row 1) in array 3, for when the count of source hosts (Table 4, row 3) is greater than 84 and the count of destination ports (Table 4, row 5) is greater than 95.

Algorithm 3 Process the Features of Array 3

Time-window period ended

Call Function with Array3 and index 3

Open logs text file3 to write

Rest array C to collect malicious IP address list

for *pointer* \leq *array index* **do** \triangleright Get all row information in the list

 Get destination address (dest IP)

 Get sources list and their count

 Get destination ports and their count

if *Src addr count3* \geq 85, and *Dst port count3* \geq 95

then

 Write to logs file3

 Write to logs file3, Anomaly

 Send message of Event 28 to Max/MSP Patch

end if

end for

Algorithm 4 checks each local source host (Table 5, row 1) in the array 4 for when the count of destination hosts (Table 5, row 3) is greater than 84 and the count of destination ports (Table 5, row 5) is greater than 95.

Algorithm 4 Process the Features of Array 4

Time-window period ended

Call Function with Array4 and index 4

Open logs text file4 to write

Rest array D to collect malicious IP address list

for *pointer* \leq *array index* **do** \triangleright Get all row information in the list

 Get sources address

 Get destination list and their count

 Get destination ports and their count

if *Dst addr count4* \geq 85, and *Dst port count4* \geq 95 **then**

 Write to logs file4, array extracted information

 Write to logs file4, Anomaly

 Send message of Event 29 to Max/MSP Patch

end if

end for

The thresholds for the above features were determined heuristically according to the nature, purpose, and expected traffic volumes for the specific network in question. Obviously, these would need to be adjusted for each separate network environment, though the above would serve as useful defaults for a network with modest numbers of visitors.

SoNSTAR's main algorithm is shown in Algorithm 5.

III. EXPERIMENTAL WORK

A. DATASET

A number of traffic datasets are available to researchers. The 1999 DARPA Intrusion Detection Evaluation dataset

Algorithm 5 SoNSTAR's Main Algorithm

```

Set Time-window period
Sniff packet and Get start time
if Packet == arrived then
  Unpack ethernet header
  Extract protocol
  if protocol == 8 then                                ▷ IP packet
    Unpack IP header
    Extract source and destination addresses
    Extract transmission protocol
  else
    Get next packet from the sniffer
  end if
  if protocol == 6 then                                ▷ TCP packet
    Unpack TCP header
    Collect and Extract (array Traffic flow, array IP
flow) features
    Collect and Extract (array1, array2, array3 and
array4) features
    Count IP flows and Traffic flows
    if Time – windowperiod == finished then
      Extract new features from Features Combiner
      Process Events of (array1, array2, array3 and
array4) features
      Process Events of (array Traffic flow, array IP
flow) features
      Write logs files while processing event
conditions
      Send Event messages of to Max/MSP for
sonification
    end if
    Get next packet from the sniffer a new
Time-window started
  else
    Get next packet from the sniffer
  end if
else
  Get next packet from the sniffer
end if
Max/MSP Patch
if messages == arrived then
  Play sound of similar messages once
end if

```

contains three weeks-worth of traffic data.¹ The first and third weeks are attack-free, with the second week containing a variety of labeled attack traffic. However, it is not labeled with sufficient detail to support tool evaluation and does not indicate which are the malicious packets [28]. Furthermore, because some traffic has been inserted post hoc into the original data the dataset does not maintain trace consistency. The KD-99 dataset is based on the DARPA set and inherits its limitations. The CAIDA [29], PREDICT [30], and

¹<https://ll.mit.edu/ideval/data/1999data.html>.

DEFCON [31] datasets all contain anomalous traffic but are not labeled. The University of New Brunswick provides several datasets of network traffic [32]. Their ISCX 2014 dataset is comprehensively labeled. However, the ISOT dataset from the University of Victoria [33] is labeled packet-by-packet and distinguishes normal from malicious traffic and so is well suited to the purposes of this paper and was the most recent such dataset available at the time of conducting this work.

1) EVALUATION DATASET

The ISOT evaluation dataset [34] was used for the experiment. The dataset is an 11.39 GiB PCAP-format file and contains traffic conforming to the TCP, UDP, DNS and ICMP protocols. The dataset contains Strom, Waledac, and Zeus botnet command and control traffic. The dataset is labeled and contains a number of malicious and non-malicious flows, as shown in Table 7. The dataset was gathered by the French chapter of the HoneyNet Project.

TABLE 7. Breakdown of ISOT malicious and non-malicious flows.

State	No of flows
Malicious	55,904 (3.33%)
Non-malicious	1,619,520 (96.66%)
Total	1,675,424 (100%)

B. EXPERIMENT 1: EXPLORING TRAFFIC FOR BOTNET ACTIVITY

Botnets mostly use distributed normal behavior and stealth horizontal and vertical activity to prevent detection, but repetitive traffic patterns still occur as bots are programmed to repeat scheduled tasks. For example, the botmaster sends an order through the C&C server to make its army of bots perform attacks against a specific web server. Since bots are usually high in number and use identical software, parts of their communication patterns will be identical or quite similar. Therefore, the victim web server is going to receive similar communication patterns from different bots, which is extremely unlikely in normal traffic.

SoNSTAR does not use any machine learning processes; instead operator-defined events allow the operator to target different network behaviors. The experienced user chooses what events should be sonified and then decides whether the behavior being heard is suspicious or not. A single botnet attack will typically comprise several specific events. The operator recognizes several event sounds in a different sequence. The sound type and sequence allow the operator to understand what is going on in their network. For example, in performing a SYN scan an adversary sends SYN packets to several ports on the target machine. SoNSTAR will play the sound of rain telling the operator that there are many SYN packets incoming to a specific host. Every open port then replies with a SYN-ACK packet, but closed ports reply with RST packets. Therefore, the target machine will send an RST packet against each closed port causing SoNSTAR to play

a wind sound. So, when SoNSTAR generates rain followed by wind sounds, the operator would know that a SYN packet scan is happening.

To explore this traffic with SoNSTAR, the operator carries out the following steps.

- 1) Set appropriate time window period.
- 2) Run SoNSTAR to read from the ISOT dataset.
- 3) Listen to the soundscape for any sounds indicating malicious behavior or suspicious activity.
- 4) When a candidate sound is heard, open the log file corresponding to the event which triggered the sound.
- 5) Search the log file for a message indicating the local IP address and the time window number which caused the sound in order to confirm recognized behavior.
- 6) Open the IP flow log file and look for the same time window number and then locate the local IP address obtained in the previous step.
- 7) Assign a recorded sound to the suspected botnet pattern and set an event condition for when the pattern is repeated twice in a time window to confirm it is a botnet performing repetitive tasks using identically programmed bots.

An initial time window was set at 35 seconds and then SoNSTAR was run on the dataset and its soundscape listened to. Different time windows can be chosen, but the longer the window the longer the user has to wait to hear changes in network state (see section III-C.2 below for comparisons at different time window settings). The repeated sequence of a bee colony sound (Table 1, row 26) followed by multiple rat squeak sounds (row 27) were observed during a single time window. The bee colony sound indicates that a local host has received connections from more than 200 different external hosts during this time window. This indicates a high possibility of distributed flow behavior. The multiple rat squeak sound indicates that a local host has received the same number of packets from 250 different external hosts. This indicates a very high possibility of repetitive flow behavior. Since this behavior is suspicious and also strange to be occurring in a single time window, it is suspected that this behavior belongs to a botnet. Therefore, the log files are inspected to confirm the situation. Table 8 shows part of the log file at time window 4 which contains these events.

It is observed that the local host 172.16.0.12 has received the same number of packets (13) from different external hosts into different ports. The log file also shows these port numbers are sequential. Also, the local host has connections with 497 different external hosts in this time window. All of this information leads to the suspicion that this is botnet behavior. Furthermore, the spring peeper sound (Table 1, row 28) indicates that a local host has received connections from more than 85 different external hosts through more than 95 different ports. This is suspected to be a horizontal scan but this depends on the purpose of the local host and its expected traffic. Also, it indicates that the local host

TABLE 8. Sample of vertical activity to local destination IP log file.

Time window	Flow i.d.	Local Dest.	Local port	No. packets
4	483	172.16.0.11	2490	13
4	484	172.16.0.11	2491	13
4	485	172.16.0.11	2492	13
4	486	172.16.0.11	2507	13
4	487	172.16.0.11	2508	13
4	488	172.16.0.11	2509	13
4	489	172.16.0.11	2512	13
4	490	172.16.0.11	2513	13
4	491	172.16.0.11	2520	13
4	492	172.16.0.11	2521	13
4	493	172.16.0.11	2526	13
4	494	172.16.0.11	2527	13
4	495	172.16.0.11	2529	13
4	496	172.16.0.11	2531	13
4	497	172.16.0.11	2532	13

might be part of a botnet communication network. Therefore, the log file was inspected. For example, at time window 8, the local destination host 172.16.0.11 has received connections from 461 different external hosts through 615 different ports, as shown in Fig. 5. Fig. 5 also shows part of the external host list. Fig. 6 shows part of the local port list at the local destination host. The complete log files can be found in the 'examples/logs' folder in the SoNSTAR repository [8].

```
8 Dest IP: 172.16.0.11 Sources count: 461
Ports count: 615
8 Sources list: ['74.205.83.92' '208.56.131.207'
'128.227.74.56' '203.84.221.51' '66.221.154.185'
'70.84.121.39' '68.249.145.10' '209.191.88.239'
'66.218.67.35' '67.63.20.218' '207.213.175.225'
'72.249.26.99' '170.94.248.237' '216.84.45.242'
'165.190.1.131' '128.115.249.90' '67.69.240.22'
'12.3.33.11' '207.115.20.21' '216.39.53.1'
'83.175.213.162' '213.81.152.19' '65.109.124.21'
'66.218.66.70' '72.14.215.27' '64.79.170.114'
'65.54.244.40' '70.158.51.118' '209.86.93.229'
'65.119.39.207' '24.75.46.254' '207.28.212.5'
'195.50.106.7' '65.54.245.8']
```

FIGURE 5. Sample of external horizontal scan log file, part 1.

```
'66.170.2.43' '213.199.154.22' '212.115.192.194']
ports list: ['10004'
'10005' '10006' '10007' '10012' '10018' '10019' '10020'
'10021' '10029' '10030' '10031' '10034' '10039' '10044'
'10048' '10049' '10050' '10052' '10053' '10060' '10066'
'10067' '10070' '10073' '10077' '10080' '10083' '10084'
'10085' '10091' '10095' '10096' '10101' '10102' '10105'
'10110' '10112' '10113' '10120' '10123' '10126' '10127'
'10129' '10131' '10133' '10136' '10138' '10142' '10147'
'10150' '10153' '10156' '10159' '10161' '10162' '10165'
'10171' '10178' '10181' '10182' '10183' '10184']
```

FIGURE 6. Sample of external horizontal scan log file, part 2.

1) RESULTS AND BOTNET PATTERNS

Two local IP addresses were identified (172.16.0.11 and 172.16.0.12) which exhibit suspicious behavior over several time widows. Based on our knowledge of

botnet characteristics, this confirms that botnet behavior is detected.

One goal of the experiment was to determine whether or not SoNSTAR features can create events capable of targeting specific behavior. Clearly, SoNSTAR features and events can be used to draw a base line for normal traffic behavior.

One of the purposes of SoNSTAR is to support existing IDSs and to contribute to their development by helping human operators to discover features, events, and patterns that can be used by IDSs to detect malicious behavior. Therefore, to use SoNSTAR to detect and confirm botnet patterns, the operator carries out the following steps.

- 1) Open the IP flow log file and look for the same time window number and then locate the local IP address obtained in the previous steps.
- 2) Assign a new recorded sound to the suspected botnet pattern and set an event condition to play that sound when the pattern occurs twice in a time window. The repetition of this event indicates that bots are performing repetitive specific flow patterns.

The IP flow log file was opened and the IP flow patterns associated with the suspicious traffic obtained. Feature conditions to match the traffic were constructed and mapped to sounds as shown in Table 10. When SoNSTAR was run, the sounds of a squirrel running quickly and a rat moving in dry leaves were heard which confirmed the presence of botnet behavior. Therefore, IP flow features demonstrated very advanced capabilities for constructing patterns which can be used to detect botnet traffic. SoNSTAR could help the user to discover botnet behavior by only mapping for repetitive normal patterns within the TCP traffic without the need to consider the application layer. The IP flow log files can be found in the ‘examples/logs’ folder in the SoNSTAR repository [8].

Fig. 7 shows part of the IP flow log file indicating how some botnet patterns use normal behaviors that are repeated several times within a single time window. This represents

```

14&19&172.16.0.11&194.176.201.22&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&20&172.16.0.11&64.250.228.130&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&21&172.16.0.11&222.255.37.15&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&22&172.16.0.11&216.200.145.235&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&23&172.16.0.11&64.233.183.27&3&3&3&0&0&3&0&0&6&12&21&21&0&0&0\\
Anomaly. rebot activity A 111
14&24&172.16.0.11&207.115.21.22&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&25&172.16.0.11&217.72.192.149&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&26&172.16.0.11&13.8.138.217&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&27&172.16.0.11&60.229.18.61&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&28&172.16.0.11&12.168.122.203&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&29&172.16.0.11&64.5.42.8&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111
14&30&172.16.0.11&144.140.80.13&1&1&1&0&0&1&0&0&2&4&7&7&0&0&0\\
Anomaly. rebot activity A 111

```

FIGURE 7. Part of the IP Flow log file (‘rebot’ = repeated botnet).

part of time window number 14 from the IP flow numbers 19 to 30. The IP flow log file can be found in the repository [8].

2) FEATURE CONSTRUCTION: EXAMPLE 1

Event 2 in Table 1 has the following event condition based on the three way handshake mechanism, and the function of the RST packet in the TCP protocol: ‘SYN in IPs >10 and SYN in IPs <30 and PSH ACK out IPs == 0 and RST out IPs >0’.

This means that if a host received 10–30 SYN packets requesting a connection but returned no data (zero PSH-ACK packets and one or more RST packets) then the rain-on-roof sound should be played. This sound will tell the operator that the network is receiving requests for connection on multiple ports but no data was returned. This is analogous to customers repeatedly coming into a shop, asking the price of an item, and then leaving without making a purchase. As any closed port will sent an RST packet on receipt of any incoming packet type, this behavior is indicative of a port scan.

3) FEATURE CONSTRUCTION: EXAMPLE 2

For another example, consider time window 8 where the IP address 172.16.0.11 was recognized. The IP flow log file is searched for time window 8 and the IP address 172.16.0.11. An extract of this time window log is shown in Table 9.

Using this information a pattern can be built. The first IP flow seen is flow 412 between hosts 172.16.0.11 and 195.188.53.99. The numbers of FIN out and FIN in, SYN out, and SYN-ACK in packets are greater than 0 and are all equal. Therefore, the first part of the pattern is the condition: ‘FIN out IPs == FIN in IPs == SYN out IPs == SYN ACK in IPs > 0’.

We know from the TCP protocol that following a FIN in and SYN out pair, an ACK out packet is a confirmation of the communication. We see that in IP flow 412, ACK out = FIN in + SYN out, so another condition is added to the pattern: ‘ACK out IPs == FC9 where FC9 = FIN in IPs + SYN out IPs’ (see Table 6).

We also see that the number of ACK-in packets is greater than ACK-out by 2, so the pattern is extended by the condition ‘FC10 >= 2 where FC10 = ACK in IPs – ACK out IPs’.

Next we observe that PSH-ACK-out and PSH-ACK-in packet counts are equal (7), so the condition ‘PSH ACK out IPs == PSH ACK in IPs >= 1’ is added.

The condition ‘ACK in IPs < PSH ACK in IPs’ is added to reflect the relationship between the number of ACK-in and PSH-ACK packets.

Therefore, the complete pattern is ‘FINoutIPs == FINinIPs == SYNoutIPs == SYNACKinIPs >0 and ACK-outIPs == FC9 and FC10 >1 and PSH ACK out IPs == PSH ACK in IPs >0 and ACK in IPs <PSH ACK in IPs’, which is event condition 30 in Table 10. If this pattern is repeated twice

TABLE 9. Sample of vertical activity to local destination IP log file.

Time window	Flow	Host A	Host B	Packet counts														
				FIN		SYN		SYN-ACK		RST		ACK		PSH-ACK		URG-PSH-FIN	NULL	LAND
				Out	In	Out	In	Out	In	Out	In	Out	In	Out	In			
8	412	172.16.0.11	195.188.53.99	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	413	172.16.0.11	82.185.226.116	2	2	2	0	0	2	0	0	4	8	14	14	0	0	0
8	414	172.16.0.11	66.193.69.2	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	415	172.16.0.11	63.166.215.100	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	416	172.16.0.11	217.116.0.152	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	417	172.16.0.11	80.240.225.37	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	418	172.16.0.11	195.242.120.10	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	419	172.16.0.11	209.59.136.109	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	420	172.16.0.11	80.12.242.15	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	421	172.16.0.11	66.218.66.215	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	422	172.16.0.11	205.152.58.32	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	423	172.16.0.11	143.100.37.72	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	424	172.16.0.11	213.161.248.130	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	425	172.16.0.11	212.88.148.234	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	426	172.16.0.11	64.251.84.10	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	427	172.16.0.11	80.207.150.20	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0
8	428	172.16.0.11	66.216.121.101	1	1	1	0	0	1	0	0	2	4	7	7	0	0	0

TABLE 10. Feature-to-sound mappings of botnet patterns. The squirrel sound is used when multiple sources repeatedly target a single host. The rat sound denotes a single host receiving the same number of packets across multiple ports.

No	Event Conditions	Sound
30	FIN out IPs == FIN in IPs == SYN out IPs == SYN ACK in IPs >0 and ACK out IPs == FC 9 and FC 10 >1 and PSH ACK out IPs == PSH ACK in IPs >0 and ACK in IPs <PSH ACK in IPs (if repeated twice in the same time window (if R2T))	Squirrel running quickly
31	FIN out IPs == 0 and FIN in IPs == SYN out IPs == SYN ACK in IPs >0 and ((ACK out IPs == FC 9 and RST out IPs == 0) or (ACK out IPs == FC 9 +1 and RST out IPs == 0) or (ACK out IPs == FC 13 and RST out IPs == 1)) and ACK in IPs <ACK out IPs and PSH ACK out IPs >PSH ACK in IPs >0 and RST in IPs == 0 (if R2T)	Rat moving in dried leaves
32	FIN out IPs == FIN in IPs == SYN out IPs == SYN ACK in IPs == ACK in IPs >0 and RST in IPs == FC 9 and ACK out IPs == PSH ACK out IPs == PSH ACK in IPs >0 and ACK in IPs <ACK out IPs and SYN in IPs == SYN ACK out IPs == RST out IPs == 0 (if R2T)	Squirrel running quickly
33	FIN out IPs == FIN in IPs >0 and SYN out IPs == SYN ACK in IPs == ACK in IPs >0 and RST in IPs == FC 11 and ((ACK out IPs == PSH ACK in IPs >0 and (PSH ACK out IPs == FC 12 or PSH ACK out IPs == PSH ACK in IPs)) or (PSH ACK in IPs == PSH ACK out IPs >0 and ACK out IPs == FC 12 or ACK out IPs == PSH ACK in IPs)) and ACK in IPs <ACK out IPs and SYN in IPs == SYN ACK out IPs == RST out IPs == 0 (if R2T)	Squirrel running quickly
34	FIN out IPs == 0 and FIN in IPs == 0 and SYN out IPs == 1 and SYN in IPs == 0 and SYN ACK out IPs == 0 and SYN ACK in IPs == 1 and RST in IPs == RST out IPs == 0 and ACK out IPs == 1 and ACK in IPs == 0 and PSH ACK out IPs == 0 and PSH ACK in IPs == 0 (if R2T)	Rat moving in dried leaves
35	FIN out IPs == FIN in IPs >0 and SYN out IPs == SYN in IPs == SYN ACK out IPs == SYN ACK in IPs == RST out IPs == 0 and RST in IPs == FC 11 and ACK out IPs == PSH ACK out IPs == PSH ACK in IPs >FC 11 and ACK in IPs <ACK out IPs (if R2T)	Squirrel running quickly

in a time window then this is indicative of botnet activity. Because normal traffic takes the form of a random pattern it cannot be repeated as quickly as happens in the case of botnets especially if it repeats in a single time period several times.

C. EXPERIMENT 2: USING SoNSTAR AS A PASSIVE IDS TO VALIDATE DISCOVERED PATTERNS

Above, by acting in the role of network operator, we used SoNSTAR to identify the behavior of botnets inside the ISOT dataset traffic. As a result, through sound and log files, we were able to discover six IP flow patterns which indicate bot behavior. Therefore, we considered using SoNSTAR as a passive IDS based on the patterns discovered. Consequently,

an algorithm was added to detect botnets in the dataset based on the IP flow patterns shown in Table 10.

The results need to be evaluated based on a labeled dataset and compared against other methods using the same dataset. Therefore, the detection algorithm was configured for different durations of time window to identify and label each detected flow as normal or malicious before storing the results in a log file. Table 11 shows how the resulting log file is structured.

The log file columns are, from left to right, the time window, the flow number in the time window, the flow number in the dataset, host A, host B, SoNSTAR classification result, and the label derived from the labeled dataset.

TABLE 11. SoNSTAR classification log file excerpt. The SoNSTAR classifications can be compared against the labels in the ISOT dataset.

Time window	Time window flow id	Dataset flow i.d.	Host A	Host B	Classification	
					SoNSTAR	ISOT
75	42	11174	172.16.0.12	65.54.244.40	Malicious	Malicious
75	43	11175	172.16.0.12	65.55.88.22	Malicious	Malicious
75	44	11176	172.16.0.12	216.39.53.3	Malicious	Malicious
75	45	11177	172.16.0.12	128.192.1.108	Malicious	Malicious
75	46	11178	172.16.0.12	65.54.245.72	Malicious	Malicious
75	47	11179	172.16.0.12	65.54.245.8	Malicious	Malicious
75	48	11180	172.16.2.13	203.69.42.35	Normal	Normal
75	49	11181	172.16.2.2	69.147.121.161	Normal	Normal
75	50	11182	172.16.2.12	203.84.202.164	Normal	Normal
75	51	11183	172.16.2.13	87.248.113.14	Normal	Normal
75	52	11184	172.16.2.2	209.85.135.103	Normal	Normal
75	53	11185	172.16.2.2	209.85.135.147	Normal	Normal

1) EVALUATION METRICS

To evaluate SoNSTAR as an anomaly detector based on the IP flow patterns just constructed, performance was assessed using the following metrics:

- (i) The number of true positives (TP) where SoNSTAR correctly identifies a malicious flow.
- (ii) The number of true negatives (TN) where SoNSTAR correctly identifies a normal (non-malicious) flow.
- (iii) The number of false positives (FP) where SoNSTAR mistakenly identifies a normal flow as malicious (botnet) activity.
- (iv) The number of false negatives (FN) where SoNSTAR mistakenly identifies a malicious flow as a normal flow.

These metrics were, in turn, used to calculate the precision, recall (true positive rate), false positive rate, accuracy, and F-measure [35]–[37] as follows:

$$FPR = \frac{FP}{FP + TN} \tag{1}$$

$$TPR/recall = \frac{TP}{TP + FN} \tag{2}$$

$$accuracy = \frac{TP + TN}{N} \tag{3}$$

$$precision (p) = \frac{TP}{TP + FP} \tag{4}$$

$$F\text{-measure} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5}$$

2) RESULTS

The SoNSTAR detection algorithm was run three times to read the whole ISOT dataset with three different time windows (20, 30, and 60 s) to evaluate the reliability of the discovered IP flow patterns. Tables 12 and 13 show the results achieved.

3) COMPARISON

The experiment used the same ISOT evaluation dataset and metrics as Kirubavathi and Anitha [38]. They evaluated three different machine learning classifiers: the Boosted

TABLE 12. FP, FN, TP and TN results.

Time window	FP	FN	TP	TN
20 s	152	144	12,543	510,720
40 s	136	75	11,105	463,000
60 s	312	49	10,632	443,819

TABLE 13. Comparison measures with time window of 20 s, 40 s and 60 s.

Window	Precision	Recall	F-measure	Accuracy	FPR
20 s	0.988	0.989	0.988	0.999	0.000297
40 s	0.987	0.993	0.991	0.9995	0.000293
60 s	0.971	0.9954	0.983	0.999	0.0007

decision tree ensemble classifier (AdaBoostM1+J48), Naive Bayesian (NB) statistical classifier and the Support Vector Machine (SVM) discriminative classifier. Their results are based on three time windows of 60 s, 120 s, and 180 s. Table 14 compares the results achieved by SoNSTAR with the three systems measured by Kirubavathi and Anitha with a 60 s time window. The comparison shows that the six patterns discovered by listening to SoNSTAR achieved a better detection accuracy, recall, precision and very low false positive rate. Note, that SoNSTAR achieves even better results with smaller time windows, with the best results being achieved with a 20 s window (Table 13). We conclude that involving human understanding in the detection of botnets increases the chances of detecting them, especially so in the case of new botnets.

4) DISCUSSION

SoNSTAR enables the user to explore distributed, parallel and horizontal behaviors that are similar to normal behaviors

TABLE 14. Comparison with existing methods at time window of 60 s.

Classifier	Precision	Recall	F-measure	Accuracy	FPR
AdaBoostM1+J48	0.958	0.933	0.954	0.9413	0.12
NB	0.979	0.976	0.968	0.9586	0.04
SVM	0.936	0.943	0.939	0.9137	0.14
SoNSTAR	0.971	0.995	0.983	0.9992	0.0007

but which can be linked to DDoS or botnet characteristics. Although every single flow may be normal, the overall behavior is suspicious. Moreover, the user recognizes identical IP flow patterns repeated several times within the same time window and probably also within several subsequent time windows. It is not expected to see such behavior within normal traffic and what might be treated as normal behavior by a machine, looks suspicious to a human operator. Therefore, normal behavior from several different external hosts cannot use a specific communication mechanism unless it was programmed to perform such action.

Acting as the network operator, we have discovered some repeated identical behaviors within traffic flows. In our study of these flow patterns, we found that it would be impossible for this normal behavior to be repeated several times in a single time period, which indicates that it is botnet behavior. To facilitate the detection of the discovered patterns in the future, we have mapped them into recorded sounds.

The advantage of using sonification is that the user is informed immediately about any traffic activity and the sounds are easier to follow and comprehend. Visualization is capable of representing traffic behavior, but it is difficult for the user to follow and recognize the frequency and sequence of occurrence of events in the way that sonification can provide, not least because it would require constant attention to a visual display. Therefore, sonification and visualization have to be integrated to raise security situational awareness.

Any features, events or patterns discovered to be symptoms of malicious behavior could then be passed to any IDS and tested and used in a machine learning process afterwards. SoNSTAR improves situational awareness levels and allows users to learn more about their own network environment rather than studying network behavior in general. As soon as the SoNSTAR user starts monitoring the network, they will recognize various thresholds and normal behaviors that pertain to their network environment and after some time will easily be able to distinguish different normal behaviors. Any unusual 'normal' behavior will become a suspicious behavior which will improve the user's awareness level.

The operator can use SoNSTAR to study the vulnerability of a network. For example, the operator could perform a penetration test against the network while monitoring it with SoNSTAR. The user can perform expected attacks based on the network's purpose and any adversary's motivations. SoNSTAR will be able to help the user to create events which reveal those attacks even if they take the form of normal behavior.

IV. CONCLUSION

IDS technologies do not include the protocol flow granularity required to understand network events inside an environment. Our proposed solution is to use SoNSTAR to learn about those environments and then IDS technology could be developed specifically for specific environments and can be deployed with confidence in detecting malicious activity.

This paper described a novel and innovative method to tackle botnet issues. This represents the first mechanism for sonifying botnet behavior. Its objective is to target botnet events in order to enable the operator to recognize them. To successfully achieve this target, we have introduced new extracted features that create events which can target botnet behavior. SoNSTAR does not use any botnet detection algorithm, but enables a human operator to recognize botnets by linking sounds of events to the structure of botnet behavior and then to extract botnet IP flow patterns, and then to confirm the presence of botnet activity by finding those patterns repeated within a time window.

We defined six patterns of botnet behavior representing normal flow patterns used by botnets. From the ISOT dataset we found evidence of botnets having a unique repetitive IP flow patterns. This shows that our sonification approach and IP flow structures can be used to detect known botnets as well as novel ones. And we have demonstrated that our sonification mechanism is effective in revealing important aspects of botnet patterns. The pattern validation experiment shows how patterns discovered by SoNSTAR could be used by IDSs to mitigate the effects of a zero-day attack.

In its current form, the system provides the network administrator with a way of hearing various traffic behaviors thereby allowing them to draw inferences about what is happening in much the way that we monitor our own everyday surroundings. Because SoNSTAR maps data to pre-recorded sounds there is scope for further parameterized control whereby additional data features could modulate those sounds in meaningful ways (a technique called Parameter Mapping Sonification).

Further research and development can be conducted to represent log file information in a visual manner, which would enable more real-time integration between sonification and visualization. Furthermore, four new algorithms were added to the proof-of-concept SoNSTAR systems and no detrimental impact on the system performance was observed. However, it would be instructive to run performance tests to determine the scalability of adding successive algorithms for dealing with new traffic features. SoNSTAR can be installed on a network gateway or, if the network is very large with lots of traffic, then multiple instances could be installed on subnet gateways. Further work is needed to determine the thresholds for making such decisions.

REFERENCES

- [1] C. C. Aggarwal, *Data Classification: Algorithms and Applications*, 1st ed. London, U.K.: Chapman & Hall, 2014.
- [2] M. D. Cavelty and V. Mauer, *Power and Security in the Information Age: Investigating the Role of the State in Cyberspace*. Evanston, IL, USA: Routledge, 2016.
- [3] N. Bradley, M. Alvarez, D. McMillen, and S. Craig, "Reviewing a year of serious data breaches, major attacks and new vulnerabilities," IBM Secur., Somers, NY, USA, Tech. Rep. SEW03133-USEN-01, 2016.
- [4] D. Zhao *et al.*, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013.
- [5] J. L. Sutherland. (Mar. 2016). *Know Your Enemy: Understanding the Motivation Behind Cyberattacks*. [Online]. Available: <https://security-intelligence.com/know-your-enemy-understanding-the-motivation-behind-cyberattacks/>

- [6] S. Axelsson and D. Sands, *Understanding Intrusion Detection through Visualization*. New York, NY, USA: Springer, 2006.
- [7] M. Debashi and P. Vickers, "Sonification of network traffic flow for monitoring and situational awareness," *PLoS ONE*, vol. 13, no. 4, p. e0195948, 2018.
- [8] M. Debashi and P. Vickers. (2017). *Nuson-SoNSTAR: Sonification of Networks for SiTuational AwaReness*. Accessed: Jun. 12, 2018. [Online]. Available: <https://github.com/nuson/SoNSTAR>, doi: 10.5281/zenodo.1072535.
- [9] E. Stalmans and B. Irwin, "A framework for DNS based detection and mitigation of malware infections on a network," in *Proc. IEEE Inf. Secur. South Africa (ISSA)*, Aug. 2011, pp. 1–8.
- [10] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, and X. Luo, "Detecting stealthy P2P botnets using statistical traffic fingerprints," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2011, pp. 121–132.
- [11] J. Kang and J.-Y. Zhang, "Application entropy theory to detect new peer-to-peer botnet with multi-chart CUSUM," in *Proc. 2nd Int. Symp. Electron. Commerce Secur. (ISECS)*, May 2009, pp. 470–474.
- [12] A. Karim, R. B. Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, "Botnet detection techniques: Review, future trends, and issues," *J. Zhejiang Univ. Sci. C*, vol. 15, no. 11, pp. 943–983, 2014.
- [13] I. Seo, H. Lee, and S. C. Han, "Cylindrical coordinates security visualization for multiple domain command and control botnet detection," *Comput. Secur.*, vol. 46, pp. 141–153, Oct. 2014.
- [14] I. Kim, H. Choi, and H. Lee, "Botnet visualization using DNS traffic," in *Proc. WISA*, 2008, pp. 1–13.
- [15] A. Shahrestani, M. Feily, R. Ahmad, and S. Ramadass, "Architecture for applying data mining and visualization on network flow for botnet traffic detection," in *Proc. Int. Conf. Comput. Technol. Develop. (ICCTD)*, Nov. 2009, pp. 33–37.
- [16] M. Ballora, R. J. Cole, H. Kruesi, H. Greene, G. Monahan, and D. L. Hall, "Use of sonification in the detection of anomalous events," *Proc. SPIE*, vol. 8407, p. 84070S, May 2012.
- [17] M. A. Garcia-Ruiz, A. Edwards, M. V. Martin, and S. El Seoud, "Auditory display as a tool for teaching network intrusion detection," *iJET*, vol. 3, no. 2, pp. 59–62, 2008.
- [18] M. Gilfix and A. L. Couch, "Peep (the network auralizer): Monitoring your network with sound," in *Proc. 14th System Admin. Conf. (LISA)*. New Orleans, LA, USA: The USENIX Association, 2000, pp. 109–117. [Online]. Available: <https://www.usenix.org/publications/library/proceedings/lisa2000/gilfix.html>
- [19] M. Kimoto and H. Ohno, "Design and implementation of stetho: Network sonification system," in *Proc. Int. Comput. Music Conf.*, 2002, pp. 273–279.
- [20] P. Vickers, C. Laing, and T. Fairfax, "Sonification of a network's self-organized criticality for real-time situational awareness," *Displays*, vol. 47, pp. 12–24, Apr. 2017.
- [21] K. E. Wolf and R. Fiebrink, "Sonnet: A code interface for sonifying computer network data," in *Proc. 13th Int. Conf. New Inter. Musical Expression (NIME)*, 2013, pp. 1–4.
- [22] D. Worrall, "Realtime sonification and visualisation of network metadata," in *Proc. 21st Int. Conf. Auditory Display (ICAD)*, K. Vogt, A. Andreopoulou, and V. Goudarzi, Eds. Graz, Austria: Inst. Electron. Music Acoust., Univ. Music Performing Arts Graz, 2015, pp. 337–339.
- [23] N. Brownlee, C. Mills, and G. Ruth. (1999). *Traffic Flow Measurement: Architecture*. [Online]. Available: <https://tools.ietf.org/html/rfc2722>
- [24] M. Eslahi, R. Salleh, and N. B. Anuar, "Bots and botnets: An overview of characteristics, detection and challenges," in *Proc. IEEE Int. Conf. Control Syst., Comput. Eng. (ICCSCE)*, Nov. 2012, pp. 349–354.
- [25] P. Barford and V. Yegneswaran, "An inside look at botnets," in *Malware Detection*, M. Christodorescu, S. Jha, D. Maughan, D. Song, and C. Wang, Eds. Boston, MA, USA: Springer, 2007, pp. 171–191.
- [26] V. Yegneswaran, P. Barford, and J. Ullrich, "Internet intrusions: Global characteristics and prevalence," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 138–147, 2003.
- [27] K. G. Geerthidevi, T. S. Prakash, and S. Tharani, "Social network based security schema for botnet detection and prevention," *Int. J. Eng. Comput. Sci.*, vol. 4, no. 6, pp. 12687–12691, 2015.
- [28] C. Brown, A. Cowperthwaite, A. Hijazi, and A. Somayaji, "Analysis of the 1999 DARPA/lincoln laboratory IDS evaluation data with NetAD-HICT," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl. (CISDA)*, Jul. 2009, pp. 1–7.
- [29] CAIDA Center for Applied Internet Data Analysis. (2007). *The CAIDA DDoS Attack Dataset*. [Online]. Available: <http://www.caida.org/data/overview/>
- [30] RTI International. (2011). *Predict: Protected Repository for the Defense of Infrastructure Against Cyber Threats*. [Online]. Available: <http://www.predict.org>
- [31] The Shmoo Group. (2011). *Defcon Dataset*. [Online]. Available: <http://cctf.shmoo.com>
- [32] University of New Brunswick. (2012). *ISCX Datasets—The Canadian Institute for Cybersecurity*. [Online]. Available: <http://www.unb.ca/cic/research/datasets/>
- [33] University of Victoria. (2010). *Isot Botnet Dataset*. Accessed: Jan. 16, 2017. [Online]. Available: <http://www.uvic.ca/engineering/ece/isot/datasets/>
- [34] S. Saad *et al.*, "Detecting P2P botnets through network behavior analysis and machine learning," in *Proc. 9th Annu. Int. Conf. Privacy, Secur. Trust (PST)*, Jul. 2011, pp. 174–180.
- [35] D. L. Olson and D. Delen, *Advanced Data Mining Techniques*. Berlin, Germany: Springer, 2008.
- [36] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.
- [37] L. Liu and M. T. Özsu, Eds., *Encyclopedia of Database Systems*. Berlin, Germany: Springer, 2009.
- [38] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics," *Comput. Electr. Eng.*, vol. 50, pp. 91–101, Feb. 2016.



MOHAMED DEBASHI received the B.Sc. degree in electronics and communications engineering from Al Zawiya University and the M.Sc. degree in information and communication technology for engineers from Coventry University. He is currently pursuing the Ph.D. degree with the Department of Computing and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K. His research is in the domain of cyber and computer network security.



PAUL VICKERS received the B.Sc. degree in computer studies and the Ph.D. degree in software engineering and HCI. He is currently an Associate Professor in computer science and computational perceptualization with Northumbria University, Newcastle upon Tyne, U.K. He is also a U.K. Chartered Engineer. His research focuses on auditory display and sonification. He was on the board of the International Community for Auditory Display (2004–2012 and 2015–2018).

...