This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



# CSC-Detector: A System to Infer Large-Scale Probing Campaigns

Elias Bou-Harb, Chadi Assi and Mourad Debbabi

**Abstract**—This paper uniquely leverages unsolicited real darknet data to propose a novel system, CSC-Detector, that aims at identifying Cyber Scanning Campaigns. The latter define a new phenomenon of probing events that are distinguished by their orchestration (i.e., coordination) patterns. To achieve its aim, CSC-Detector adopts three engines. Its fingerprinting engine exploits a unique observation to extract probing activities from darknet traffic. The system's inference engine employs a set of behavioral analytics to generate numerous significant insights related to the machinery of the probing sources while its analysis engine exploits the previously obtained inferences to automatically infer the campaigns. CSC-Detector is empirically evaluated and validated using 240 GB of real darknet data. The outcome discloses 3 recent, previously unreported large-scale probing campaigns targeting diverse Internet services. Further, one of those inferred campaigns revealed that the sipscan campaign that was initially analyzed by CAIDA is arguably still active, yet operating in a stealthy, very low rate mode. We envision that the proposed system that is tailored towards darknet data, which is frequently, abundantly and effectively used to generate cyber threat intelligence, could be used by network security analysts, emergency response teams and/or observers of cyber events to infer large-scale orchestrated probing campaigns. This would be utilized for early cyber attack warning and notification as well as for simplified analysis and tracking of such events.

**Index Terms**—Probing campaigns, probing inferences, data analytics, data correlation, cyber intelligence

## 1 Introduction

Probing, the task of scanning enterprise networks or Internet wide services, searching for vulnerabilities or ways to infiltrate IT assets, is a significant cyber security concern. It is basically a core mechanism and a facilitating factor of numerous cyber attacks. For instance, hackers have employed probing techniques to steal more than 4.5 million patient records from one of the largest hospital operators in North America [1]. Further, the United States Computer Emergency Readiness Team (US-CERT) revealed that attackers had performed coordinated probing activities to fingerprint WordPress sites and consequently launched their targeted attacks [2]. More alarming, a recent incident reported that attackers had escalated a series of "surveillance missions" against cyber-physical infrastructure operating various US energy firms that permitted the hackers to infiltrate the control-system software and subsequently manipulate oil and gas pipelines [3]. Thus, it is not surprising that Panjwani et al. and Treurniet [4, 5] concluded that a momentous 50% of attacks against cyber systems are preceded by some form of probing activity.

Lately, there has been a noteworthy shift towards a new phenomenon of probing events which, throughout this paper, is referred to as Cyber Scanning Campaign(s) (CSC(s)). These are distinguished from previous probing incidents as (1) the population of the participating bots is several orders of magnitude larger, (2) the target scope is generally the entire Internet Protocol (IP) address space, and (3) the bots adopt well-orchestrated, often botmaster-coordinated, stealth scan strategies that maximize targets' coverage while minimizing redundancy and overlap [6, 7]. Please note that although our previous work tackled the problem of inferring probing activities [8, 9], however, in this work, we focus on inferring and attributing such new phenomenon of probing activities dubbed as orchestrated CSCs by uniquely analyzing the dark space, which has never been attempted before. Indeed, very recently, Dainotti et al. [6] from the Cooperative Association for Internet Data Analysis (CAIDA) presented a pioneering measurement and analysis study of a 12-day Internet-wide CSC targeting VoIP (SIP) servers. In another work [7], the same authors admitted that they have detected the reported SIP CSC including the malware responsible for its actions (i.e., Sality malware) "serendipitously" (i.e., luckily and accidentally) while analyzing a totally unrelated phenomenon. They also stated that since *currently there exist no cyber security capability to discover such large-scale probing campaigns*, other similar events targeting diverse Internet and organizational infrastructure are going undetected. In another inquisitive, well executed work, an "anonymous" presented and published online [10] what they dubbed as the "Carna Botnet". The author exploited

• *The first author is affiliated with Florida Atlantic University and NCFTA Canada. ebouharb@fau.edu. The remaining two authors are affiliated with Concordia University and NCFTA Canada. (assi, debbabi)@ciise.concordia.ca*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



poorly protected Internet devices, developed and distributed a custom binary, to generate one of the largest and most comprehensive IPv4 census ever.
The aforementioned two CSC studies differ on various key observations. The work by Dainotti et al. disclosed that the bots were recruited into the probing botnet by means of a new-generation malware while the Carna Botnet was augmented using a custom code binary. Moreover, Dainotti et al. discovered that the bots were coordinated by a botmaster in a Command-and-Control (C&C) infrastructure where the bots used a reverse IP-sequential strategy to perform their probing, while the Carna Botnet was C&C-less and its bots used an interleaving permutation method to scan its targets. Further, the work by Dainotti et al. documented a horizontal scan that targeted world-wide SIP servers, while the Carna Botnet did not focus on one specific service but rather attempted to retrieve any available information that was associated with any host and/or service. Readers that are interested in more details related to the discussed CSCs are kindly referred to [6, 10]. We project that current undetected and unreported CSCs as well as future occurrences could be ominously leveraged to cause drastic Internet-wide and enterprise impacts as precursors of various amplified, debilitating and disrupting cyber attacks including, but not limited to, distributed and reflective denial of service attacks, advanced persistent threats and spamming campaigns.

Motivated by the requirement to build a prompt cyber security capability to generically detect and identify large-scale probing campaigns, we frame the paper's contribution as follows:

- Proposing a new method to fingerprint probing activities (i.e., detect activity and identify the scanning technique). Contrary to [11–15], the method does not rely on identifying the scanning source and is independent from the scanning strategy (remote to local, local to remote, local to local), the scanning aim (wide range or target specific) and the scanning method (single source or distributed). Further, the proposed method does not rely (for detection) on a certain predefined alert threshold, the transport protocol used (TCP or UDP) or the number of probed destinations and ports. CSC-Detector employs this method in its fingerprinting engine to infer (i.e., extract) probing activities embedded in darknet traffic.
- Presenting CSC-Detector, a system that employs a systematic approach in an attempt to detect and identify large-scale probing campaigns. Its inference engine employs a set of novel behavioral analytics to generate significant insights that aim at capturing the machinery of the probing sources while the system's analysis engine leverages those analytics in addition to several criteria and methods to automatically infer the campaigns, *without any prior knowledge about their specifications*.
- Developing CSC-Detector as a prototype, evaluating it using 240 GB of real darknet data and validating its inferences using other data sources [16]. Indeed, CSC-Detector operates by scrutinizing observed traffic as it is received by dark sensors. The latter is often dubbed as darknet traffic. In a nutshell, darknet traffic is Internet traffic destined to routable but unused Internet addresses. Since these addresses are unallocated, any traffic targeting them is deemed as suspicious. Darknet analysis has shown to be an effective method to generate cyber threat intelligence [17, 18]. We have been receiving, for the past three years, around 12 GB of daily darknet traffic from a trusted third party [19]. Such traffic originates from the Internet and is destined to numerous /13 network sensors ($\approx$ half a million dark IPs) in more than 12 countries. The data mostly consists of unsolicited TCP, UDP and ICMP traffic. It might contain as well some DNS traffic.

The remainder of this paper is organized as follows. Section 2 examines the related work and shows how the proposed work differs from the current literature. Section 3 elaborates on the proposed system. Specifically, it explains the rationale and employed methodologies that are adopted by each of CSC-Detector's three engines. In Section 4, we report on the system's implementation and present its empirical evaluation. The findings (1) disclose 3 recent, previously obscured and unreported large-scale probing campaigns targeting diverse Internet services (2) were validated using publically available data sources and (3) arguably proves that the CSC in [6], contrary to what the authors stated, is indeed still active yet operating in a stealthy, very low rate mode. Finally, Section 5 concludes the paper, pinpoints some lessons learned and paves the way for future work.

## 2 Related Work

In this section, we review the related work on various concerned topics.

**Extracting Probing Events:** Li et al. [20] considered large spikes of unique source counts as probing events. The authors extracted those events from darknet traffic using time series analysis; they first automatically identified and extracted the rough boundaries of events and then manually refined the event starting and ending times. At this point, they used manual analysis and visualization techniques to extract the event. In an alternate work, Jin et al. [21] considered any incoming flow that touches any temporary dark (grey) IP address as potentially suspicious.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



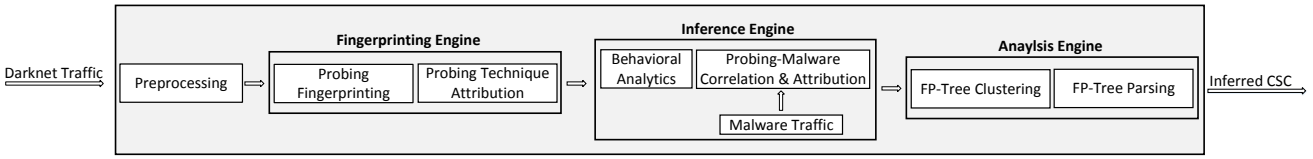



Fig. 1: CSC-Detector: System Architecture

The authors narrowed down the flows with sustained suspicious activities and investigated whether certain source or destination ports are repeatedly used in those activities. Using these ports, the authors separated the probing activities of an external host from other traffic that is generated from the same host. In contrast, in this work, we propose a method that exploits a unique observation related to the signal correlation status of probing events. By leveraging this, CSC-Detector's fingerprinting engine is able to differentiate between probing and other events and subsequently extract the former from incoming darknet traffic.

**Analyzing Probing Events:** The authors of [21, 22] studied probing activities towards a large campus network using netflow data. Their goal was to infer the probing strategies of scanners and thereby assess the harmfulness of their actions. They introduced the notion of gray IP space, developed techniques to identify potential scanners, and subsequently studied their scanning behaviors. In another work, the authors of [20, 23, 24] presented an analysis that drew upon extensive honeynet data to explore the prevalence of different types of scanning. Additionally, they designed mathematical and observational schemes to extrapolate the global properties of scanning events including total population and target scope. Further, in our previous work, we have tackled the problem of analyzing probing activities targeting corporate networks [8] and fingerprinting independent probing activities [9]. In contrary, in this work, we aim at inferring large-scale probing campaigns rather than focusing on analyzing probing events. To achieve that, CSC-Detector's inference and analysis engines collaborate to scrutinize the behavior of probing events as perceived by a darknet and subsequently automatically build patterns that aim at correlating the probing sources into well-defined campaigns.

**Probing Measurement Studies:** In addition to [6, 10], Benoit et al. [25] presented the world's first Web census while Heidemann et al. [26] were among the first to survey edge hosts in the visible Internet. Further, Pryadkin et al. [27] offered an empirical evaluation of IP address space occupancy whereas Cui and Stolfo [28] presented a quantitative analysis of the insecurity of embedded network devices obtained from a wide-area scan. In a slightly different work, Leonary and Loguinov [29] demonstrated IRLscanner, a tool which aimed at maximizing politeness yet provided scanning rates that achieved coverage of the Internet in minutes. In this work, as previously mentioned, we strive to infer large-scale probing campaigns rather than solely providing measurements of particular probing events.

**Botnet Detection Frameworks:** A number of botnet detection systems have been proposed in the literature [30, 31]. Some investigates specific channels, others might require deep packet inspection or training periods, while the majority depends on malware infections and/or attack life-cycles. To the best of our knowledge, none of the proposals is dedicated to tackle the problem of inferring large-scale probing campaigns. Further, in this work, we aim to achieve that task by analyzing the dark IP space and by focusing on the machinery of the probing sources, without requiring content analysis or training periods. Indeed, as stated by CAIDA in [7], the capability to infer large-scale orchestrated probing campaigns does not exist, rendering the proposed system as a novel contribution.

## 3 Proposed System: CSC-Detector

This section introduces CSC-Detector as significantly simplified in Figure 1. In a nutshell, the system takes as input darknet traffic and outputs detected well-defined CSCs. We refer to the inferred CSCs as being "well-defined" since the probing sources of those CSCs will be identified and correlated through patterns that are automatically generated by the analysis engine in which each of those patterns is defined by specific, clearly identifiable, similar probing behavioral characteristics. This section elaborates on the goals, approaches, and methods that are adopted by each of CSC-Detector's three engines that permits the system to achieve its aim.

### 3.1 The Fingerprinting Engine

The primary goal of the fingerprinting engine is to detect probing activities while its secondary goal is to infer the specific probing techniques that were employed in the probing activities. Achieving those two goals will provide the engine with the capability to fingerprint and thus subsequently extract those probing activities from darknet traffic. Further, the secondary goal of the engine will be an added-value insight for the inference engine and will also be accessible by the analysis engine. Please note that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



in this section, we exclusively focus on clarifying the approach that is adopted by the fingerprinting engine to achieve the two mentioned goals. Any required darknet traffic pre-processing that permits CSC-Detector to achieve its ultimate goal (i.e., infer the CSC) will be discussed in Section 3.4.

The rationale of the approach that is employed by CSC-Detector's fingerprinting engine states, that regardless of the source, strategy and aim of the probing, such activity should have been generated using a certain probing technique (i.e., TCP SYN, UDP, ICMP, etc. [32]). We observe that a number of those techniques demonstrate a similar and a distinct temporal correlation and similarity when generating their corresponding probing traffic. In other words, the observation states that we can cluster the scanning techniques based on the correlation status derived from their generated traffic[1]. The approach exploits this unique observation to differentiate between probing and other malicious darknet traffic based on that status. Further, it attributes the probing traffic to a certain cluster of techniques sharing the same correlation status. To identify exactly which technique has been employed in the probing, the approach statistically estimates the relative closeness of the probing traffic in comparison with the techniques found in that cluster.

### 3.1.1 Detrended Fluctuation Analysis

To enable the capturing of the traffic correlation status, the fingerprinting engine leverages the Detrended Fluctuation Analysis (DFA) technique. DFA was first proposed in [33] and has since been used in many research areas to study signals correlation. Very limited work in the areas of cyber security and malicious traffic detection has utilized DFA [34], and to the best of our knowledge, no work has leveraged the DFA technique to tackle the problem of fingerprinting probing traffic. The DFA method of characterizing a non-stationary time series is based on the root mean square analysis of a random walk. DFA is advantageous in comparison with other methods such as spectral analysis and Hurst analysis since it permits the detection of long range correlations embedded in a seemingly non-stationary time series. It avoids as well the spurious detection of apparent long-range correlations that are an artifact of non-stationarity. Another advantage of DFA is that it produces results that are independent of the effect of the trend. Further, in terms of processing, the DFA approach requires $O(m)$ where $m$ is the size of the traffic time series. Given such time series, the following steps need to be applied to implement DFA:

- Integrate the time series; The time series of length $N$ is integrated by applying

$$y(k) = \sum_{i=1}^{k} [B(i) - B_{\text{ave}}] \quad (1)$$

where $B(i)$ is the $i^{\text{th}}$ interval and $B_{\text{ave}}$ is the average interval.
- Divide the time series into "boxes" (i.e., bin size) of length $n$.
- In each box, perform a least-squares polynomial fit of order $p$. The $y$ coordinate of the straight line segments is denoted by $y_n(k)$.
- In each box, detrend the integrated time series, $y(k)$, by subtracting the local trend, $y_n(k)$. The root-mean-square fluctuation of this integrated and detrended time series is calculated by

$$F(n) = \sqrt{\frac{1}{N} \sum_{k=1}^{N} [y(k) - y_n(k)]^2} \quad (2)$$

- Repeat this procedure for different box sizes (i.e., time scales) $n$

The output of the DFA procedure is a relationship $F(n)$, the average fluctuation as a function of box size, and the box size $n$. Typically, $F(n)$ will increase with box size $n$. A linear relationship on a log-log graph indicates the presence of scaling; statistical self-affinity expressed as $F(n) \sim n^{\alpha}$. Under such conditions, the fluctuations can be characterized by a scaling exponent $\alpha$, which is the slope of the line relating $logF(n)$ to $log(n)$. The scaling exponent $\alpha$ can take the following values, disclosing the *correlation status* of the traffic time series [33]: $\alpha < 0.5$: anti-correlated; $\alpha \approx 0.5$: uncorrelated or white noise; $\alpha > 0.5$: correlated; $\alpha \approx 1$: $1/f$-noise or pink noise; $\alpha > 1$: non-stationary, random walk like, unbounded; $\alpha \approx 1.5$: Brownian noise.

### 3.1.2 Engine Process

Figure 2 depicts the system process that is adopted by the fingerprinting engine to permit the inference of the probing activities as well as the employed probing techniques. One of the issues that will arise is where to apply DFA given a traffic time series that needs to be tested; this problem could be re-stated as follows: given a traffic time series $S_t$, find a starting position $X$ and an ending position $X + \sigma$ in $S_t$ where we can apply DFA on. Assuming a random or a predefined window location in the time series $S_t$ to apply DFA on will be erroneous as this might result in wrong inferences. For example, if the traffic time series that needs to be tested is of length 5 minutes, applying DFA on the entire distribution could indicate a result (suppose it was inferred that it is not probing) while the actual probing starts on the 3$^{\text{rd}}$ minute; the entire distribution's correlation status appears to be close to noise (i.e., $\alpha$ value $\approx$ 1 and hence not probing) while from the 3$^{\text{rd}}$

1. The validation of this observation can be found in http://public.eng.fau.edu/ebouharb/Observation_Validation.pdf

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



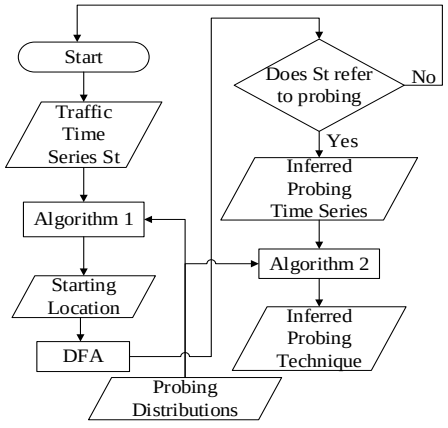



Fig. 2: Employed System Process

minute up to the 5th the signal is correlated (i.e., probing). To tackle this, we present Algorithm 1 which presents an approach to approximate when to apply DFA, to correctly infer whether or not the traffic refers to probing activities. As shown in Figure 2, Algorithm 1 takes as input the time series $S_t$ of the distribution under testing and all the distributions of the probing techniques $Sc_p$. The latter refers to the complete finite set of permuted distributions that abide with the boundaries of the correlation status as indicated in Section 3.1.1 which correspond to a specific probing technique. For instance, one set of distributions of $Sc_p$ for the FIN probing technique would be all the time series distributions that generate a scaling exponent $\alpha$ between 0 and 0.5. This will significantly aid in reducing the chances of the engine missing the probing because of a variation in the traffic distribution for a specific probing technique (i.e., might also aid in capturing zero-day probing activities). For every distribution related to the probing techniques, the algorithm calculates the Euclidean distance **E** between the points in $Sc_p$ and $S_t$. Subsequently, the probing technique distribution is moved one point in a sliding window fashion against $S_t$. For each sliding window, it records the distance between $Sc_p$ and $S_t$. After finishing all the sliding window procedures[2], the algorithm stores the minimum distance between both sliding windows in all the iterations. The algorithm finally selects $X$ as the minimum of all the distances in all sliding windows after all the techniques in $Sc_p$ have passed on $S_t$. This will approximate the starting position on where to apply DFA in $S_t$. Note that, $\sigma$ in the ending position $X$+$\sigma$, that was previously mentioned, is the length of the probing technique $Sc_p$ where $X$ was derived from. It is significant to note that we use the probing techniques of $Sc_p$ as a way to infer, in an apriori fashion, where to start applying DFA and hence where we estimate that the probing is initiated; we are not completely matching the techniques with the input time series $S_t$. Thus, any variation of the probing techniques' distributions is tolerated, as long as their correlation status, as we have observed and validated, are kept stable.

**input** : A time series $S_t$ of the distribution under testing; the set of time series $Sc_p$ of the distributions of the probing techniques.
**output**: $X$, reflecting the starting location on where to apply DFA in $S_t$

$m$=length($S_t$);
**for every** $Sc_p$ **do**
$n$=length($Sc_p$);
**for** *i=1* $\rightarrow (m-n)$ **do**
  s[i]=compare[$S_t(1+i, \cdots, n+i)$, $Sc_p(1, \cdots, n)$];
**end**
S[p]=min(s[]);
**end**
$X$=min(S[]);
return ($X$);

**compare**(A, B)
**for** *i=1* $\rightarrow n$ **do**
  $K[i]$= $\|\mathbf{E}\| = \mathrm{d}(A(i), B(i))$;
  $sum$+=$K[i]$;
  return ($sum$);
**end**

**Algorithm 1:** Approximating the starting location on where to apply DFA in $S_t$

After applying DFA, given the output information of Algorithm 1, we end up with a certain correlation status. We expect that the correlation status indicates a probing activity. However, we may encounter the case where the correlation status does not indicate probing (i.e., uncorrelated, $1/f$-noise or Brownian noise). If the activity refers to probing, then the output correlation status will lead us to a certain cluster of probing techniques that this probing activity has been generated from. To exactly identify which technique has been used to probe, we present Algorithm 2, which discloses an approach to achieve that. As depicted in Figure 2, Algorithm 2 takes as input a time series $S_b$ of the probing distribution that DFA was previously applied on; $S_b$ refers to the time series extracted from $X$ to $X + \sigma$. For each of the probing technique signals $Sc_{bi}$ in the cluster $Sc_b$ that is related to the previous output correlation status, we statistically measure the relative closeness of $S_b$ to each probing technique in that cluster using Bhattacharyya distance [35], **Bha**. The latter statistical test is an established and an effective metric to determine the overlap of two sample distributions [35]. Intuitively, the algorithm selects the technique's distribution that is closer to $S_b$, $Sc_{bi}$. $Sc_{bi}$ will be

2. The maximum number of iterations is $m/n$ of Algorithm 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



**input** : A time series $S_b$ of the probing distribution that DFA was previously applied on; a cluster of time series $Sc_b$ of the distributions of the probing techniques related to the correlation status.
**output**: $Sc_{bi}$, reflecting one probing technique that is estimated to be generating the probing activity found in $S_b$.

**for every** $Sc_{bi}$ **do**
$Bha_{bi}$ = $\|\mathbf{Bha}\| = \mathrm{d}(\mathbf{Sc_{bi}}, \mathbf{S_b})$;
**end**
$d_i = Min(Bha_{bi})$;
return ($Sc_{bi}|i$ of $d_i$);

**Algorithm 2:** Identifying the employed probing technique

identified as the probing technique that $S_b$ was employing.

It might be worthy to mention that an attacker will not be able to avoid the proposed approach by performing any significant distribution modifications. This is factual as although he might have knowledge about the existence of the dark space and the deployed approach, however, it is reasonable to state and claim that he has no capability to neither fingerprint the darkspace nor bypass the approach. In the rare event where this would occur, we postulate that literature approaches rooted in change point detection could aid in evasion prevention. See footnote 3 for proof-of-concept.

### 3.2 The Inference Engine

The goal of CSC-Detector's inference engine is to generate inferences and insights related to the machinery of the probing sources. The engine takes as input the previously extracted probing sessions and outputs a series of behavioral characteristics related to the probing sources. In what follows, we pinpoint the concerned questions and subsequently present our undertaken approach (i.e., the behavioral analytics) in an attempt to answer those.

**Is the probing traffic random or does it follow a certain pattern?** When sources generate their probing traffic, it is significant to capture the fashion in which they accomplish that. To achieve this task, we proceed as follows. For each distinct pair of hosts retrieved from the probing sessions (probing source to target), we test for randomness in the generated traffic using the non-parametric Wald-Wolfowitz statistical test. If the result is positive, we record it for that specific probing source and apply the test for the remaining probing sessions. If the outcome is negative, we infer that the generated traffic follows a certain pattern. To capture the specific employed pattern, we model the probing traffic as a Poisson process and retrieve the maximum likelihood estimate intervals (at a 95% confidence level) for the Poisson parameter $\lambda$ that corresponds to that traffic. The choice to model the traffic as a Poisson distribution is motivated by [36], where the authors observed that probe arrivals is coherent with that distribution. After the test has executed for all the probing sources, we apply the CLUstEring based on local Shrinking (CLUES) algorithm [37] on the generated patterns. CLUES allows non-parametric clustering without having to select an initial number of clusters. The outcome of that operation is a set of specific $\lambda$ intervals. The aim of this is to map each probing source that was shown to employ a pattern to a certain $\lambda$ interval by removing overlapping values that could have existed within the initially generated $\lambda$ intervals.

**How are the targets being probed?** As revealed in [6, 10], coordinated probing sources employ various strategies when probing their targets. These strategies could include IP-sequential [38], reverse IP-sequential [26], uniform permutation [39] or other types of permutations. In an attempt to capture the probing strategies, we execute the following: For each probing source, we extract its corresponding distribution of target IPs. To differentiate between sequential and permutation probing, we apply the Mann-Kendall statistical test, a non-parametric hypothesis testing approach, to check for monotonicity in those distributions. The rational behind the monotonicity test is that sequential probing will indeed induce a monotonic signal in the distribution of target IPs while permutation probing will not. Further, in this work, we set the significance level to 0.5% since a higher value could introduce false positives. To differentiate between (forward) IP-sequential and reverse IP-sequential, for those distributions that tested positive for monotonicity, we also record the slope of the distribution; a positive slope defines a forward IP-sequential strategy while a negative one renders a reverse IP-sequential strategy. For those distributions that tested negative for monotonicity (i.e., not a sequential strategy), we leverage the chi-square goodness-of-fit statistical test. The latter insight will inform us whether or not the employed strategy is a uniform permutation; if the test fails, then the employed strategy will be deemed as a permutation; uniform permutation otherwise.

**What is the nature of the probing source?** It is significant as well to infer the nature of the probing source; whether it is a probing tool or a probing bot. From the two previous questions, we can infer those probing events that are random and monotonic. It is known that monotonic probing is a behavior of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



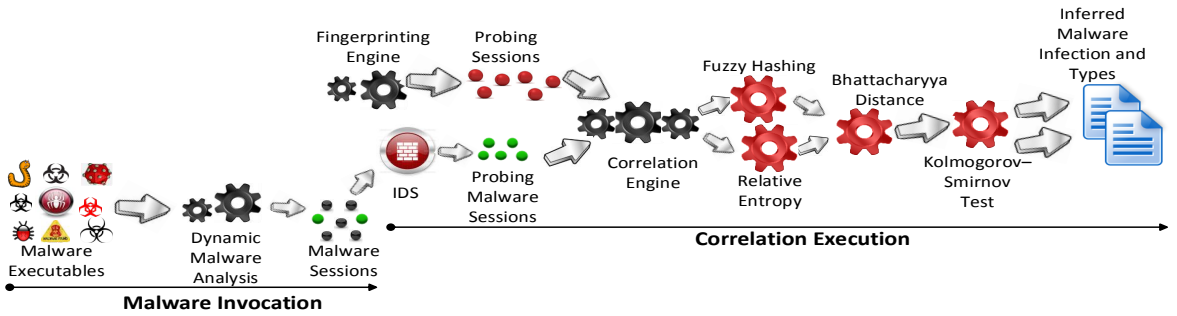



Fig. 3: Inferring the maliciousness of the probing sources

probing tools in which the latter sequentially scan their targets (IPs and ports) [40]. Furthermore, for random events, the monotonic trend checking can help filter out traffic caused by the non-bot scanners [20]. Thus, we deem a probing source as leveraging a probing tool if their traffic is randomly generated and if they adopt a sequential probing strategy (i.e., including reverse IP-sequential). Otherwise, we deem the source as a bot.

**Is the probing targeted or dispersed?** When sources probe their targets, it would be interesting to infer whether their probing traffic is targeted towards a small set of IPs or dispersed. To answer this, for each probing source $b$, we denote GF($b$) as the collection of flows generated by that specific source that target the dark space. The destination target IPs used by the flows in GF($b$) induce an empirical distribution. Subsequently, we borrow the concept of relative uncertainty and apply it on those distributions. The latter index is a decisive metric of variety, randomness or uniformity in a distribution, regardless of the sample size. An outcome that is close to 0 defines that the probing source is using a targeted approach while an outcome value close to 1 means that its corresponding probing traffic is dispersed.

**Are the probing sources infected?** On one hand, the authors of [6] pinpointed that the SIP CSC probing bots were infected by the Sality malware and were coordinated in a C&C infrastructure. On the other hand, the author of [10] did not employ any specific malware to recruit the probing bots into the CSC. Thus, it is of momentous importance to infer whether or not the probing sources are infected by a malware and if they are, which exact malware type/family/variant is causing the probing. The latter insight would be an added-value inference, for this work, for the purpose of correlating the sources into well-defined CSCs, as well as, for future work, for the analysis of the pinpointed malware binary for the sake of understanding its inner workings and perhaps inferring its C&C servers. In an attempt to answer this question, we employ the approach that is summarized in Figure 3.

We do receive, on a daily basis, malware samples from ThreatTrack Security [41]. We operate a dynamic malware analysis module that is based on a sandbox environment (i.e., controlled environment). After receiving the malware samples from ThreatTrack feeds, they are interactively sent to the sandbox, where they are executed by client machines. The clients could be virtual or real and possess the capability to run Windows or Unix, depending on the malware type under execution. The behavior of each malware is monitored and all its corresponding activities (i.e., created files, processes, network traffic, etc.) are recorded. For the sake of this work, we extract the network traffic generated by approximately 100 thousand unique and recent (June 2012 to March 2014) malware samples as packet captures (pcaps). The pcaps contain one-way communication traffic generated from the malware to other internal or external hosts. Those malware samples belong to diverse malware types including, Trojan, Virus, Worm, Backdoor, and AdWare coupled with their corresponding families and variants. We rely on Kaspersky for a uniform malware naming convention. To investigate whether the probing sessions that were initially received from the fingerprinting engine demonstrate any signs of malware infection, we perform the following. We leverage Snort's probing engine [42], the sfPortscan pre-processor, to detect which malware pcaps possess any signs of probing activity. We omit those malware pcaps that demonstrate a negative output. To attribute a specific malware to a probing session, we adopt a two-step procedure. First, we apply the notion of fuzzy hashing [43] between the probing session and the remaining malware pcaps. Fuzzy hashing is advantageous in comparison with typical hashing as it can provide a percentage of similarity between two samples rather than producing a null value if the samples are different. This popular technique is derived from the digital forensics research field and is typically applied on files or images [43]; to the best of our knowledge, our approach is among the first to explore the capabilities of this technique

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



on cyber security data. We further apply relative entropy, as proposed by Lee and Xiang [44], between the given probing session and the malware pcaps. If the relative entropy is $= 0$, this indicates that the two datasets have the same regularity. At this point, we (1) omit the probing sessions that demonstrate less than 5% similarity using both tests and (2) select the top 10% malware pcaps that were found to minimize the entropy and maximize the fuzzy hashing percentage. The rational behind the latter approach stems from the need to filter out the malware pcaps that do not possess probing signs similar to the probing session. Second, using the remaining 10% malware pcaps, we extract their probing sessions as pinpointed by sfPortscan. For each of the malware probing sessions, we apply the Bhattacharyya distance between those and the given probing session. By selecting 1% of malware pcaps that were shown to reduce the Bhattacharyya distance, we further significantly reduce the possible malware pcaps that the given probing session could be similar to. Finally, to exactly attribute the given probing session to a specific malware, we employ the two sample Kolmogorov-Smirnov statistical test [45] between the remaining malware probing sessions and the given probing session. The test will output On one hand, the authors of [6] pinpointed that the SIP CSC probing bots were infected by the Sality malware and were coordinated in a C&C infrastructure. On the other hand, the author of [10] did not employ any specific malware to recruit the probing bots into the CSC. Thus, it is of momentous importance to infer whether or not the probing sources are infected by a malware and if they are, which exact malware type/family/variant is causing the probing. The latter insight would be an added-value inference, for this work, for the purpose of correlating the sources into well-defined CSCs, as well as, for future work, for the analysis of the pinpointed malware binary for the sake of understanding its inner workings and perhaps inferring its C&C servers.

In an attempt to answer this question, we employ the approach that is summarized in Figure 3. We do receive, on a daily basis, malware samples from ThreatTrack Security [41]. We operate a dynamic malware analysis module that is based on a sandbox environment (i.e., controlled environment). After receiving the malware samples from ThreatTrack feeds, they are interactively sent to the sandbox, where they are executed by client machines. The clients could be virtual or real and possess the capability to run Windows or Unix, depending on the malware type under execution. The behavior of each malware is monitored and all its corresponding activities (i.e., created files, processes, network traffic, etc.) are recorded. For the sake of this work, we extract the network traffic generated by approximately 100 thousand unique and recent (June 2012 to March 2014) malware samples as packet captures (pcaps). The pcaps contain one-way communication traffic generated from the malware to other internal or external hosts. Those malware samples belong to diverse malware types including, Trojan, Virus, Worm, Backdoor, and AdWare coupled with their corresponding families and variants. We rely on Kaspersky for a uniform malware naming convention.

To investigate whether the probing sessions that were initially received from the fingerprinting engine demonstrate any signs of malware infection, we perform the following. We leverage Snort's probing engine, the sfPortscan pre-processor, to detect which malware pcaps possess any signs of probing activity. We omit those malware pcaps that demonstrate a negative output. To attribute a specific malware to a probing session, we adopt a two-step procedure. First, we apply the notion of fuzzy hashing [43] between the probing session and the remaining malware pcaps. Fuzzy hashing is advantageous in comparison with typical hashing as it can provide a percentage of similarity between two samples rather than producing a null value if the samples are different. This popular technique is derived from the digital forensics research field and is typically applied on files or images [43]; to the best of our knowledge, our approach is among the first to explore the capabilities of this technique on cyber security data. We further apply relative entropy, as proposed by Lee and Xiang [44], between the given probing session and the malware pcaps. If the relative entropy is $= 0$, this indicates that the two datasets have the same regularity. At this point, we (1) omit the probing sessions that demonstrate less than 5% similarity using both tests and (2) select the top 10% malware pcaps that were found to minimize the entropy and maximize the fuzzy hashing percentage. The rational behind the latter approach stems from the need to filter out the malware pcaps that do not possess probing signs similar to the probing session.

Second, using the remaining 10% malware pcaps, we extract their probing sessions as pinpointed by sfPortscan. For each of the malware probing sessions, we apply the Bhattacharyya distance between those and the given probing session. By selecting 1% of malware pcaps that were shown to reduce the Bhattacharyya distance, we further significantly reduce the possible malware pcaps that the given probing session could be similar to. Finally, to exactly attribute the given probing session to a specific malware, we employ the two sample Kolmogorov-Smirnov statistical test between the remaining malware probing sessions and the given probing session. The test will output 0 if a positive match

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



occur; 1 otherwise. If a positive match occurs, this indicates that the probing session has been generated from the inferred exact malware. In summary, the outcome of the aforementioned approach is whether or not the probing sources demonstrate signs of malware infection and if they do, which exact (or probable) malware type/family/variant is responsible for their probing.

**Miscellaneous Inferences:** For each probing source, we also record its rate (packets/second), its ratio of destination overlaps defined as $r = \frac{nc}{nt}$ where $nc$ defines the number of common sessions between all the sources and $nt$ is total number of all probing sessions, and its target ports.

It is evident that CSC-Detector's inference engine significantly depends on numerous statistical tests and methods to capture the behavior of the probing sources. We assert that such behavioral analytics are arguably more sound than heuristics or randomly set thresholds. Further, to prevent diversion of the scope of the paper, we chose not to elaborate on the theory of every single leveraged statistical method. It is also worthy to mention that all the employed statistical tests assume that the data is drawn from the same distribution. Since CSC-Detector operates on one type of data, namely, darknet data, we can safely presume that the values follow and are in fact drawn from the same distribution.

### 3.3 The Analysis Engine

Previous works [46] suggested that coordinated bots within a botnet campaign probe their targets in a similar fashion. CSC-Detector's analysis engine exploits this idea by automatically building patterns that consist of similar probing behavioral characteristics. The latter aims at identifying and correlating the probing sources into well-defined campaigns. The engine considers the criteria (i.e., features) that are summarized in Table 1. Inferred from the fingerprinting engine, the employed probing technique is a significant behavior; [6] demonstrated that all the CSC bots used UDP scanning. Further, the analysis engine consumes all the previously inferred probing machinery that is derived from the inference engine. It considers the fashion of the generated probing traffic; whether random or not, and which specific pattern has been adopted if not random; which probing strategy has been employed; whether the probing source is a probing tool or a bot; whether the probing is targeted or dispersed; whether or not the probing sources demonstrate any signs of malware infection and which specific malware type/family/variant if they do. Additionally, bots/sources in a CSC are postulated to possess similar probing rates and ratios of destination overlaps; we consider a 90% confidence interval as being similar. Finally, the analysis engine considers the target port as a significant criterion; [6] disclosed that all the CSC bots used port 5060.

| Employed probing technique |
|---|
| Probing traffic (Random Vs Patterns) |
| Employed pattern |
| Adopted probing strategy |
| Nature of probing source |
| Type of probing (Targeted Vs Dispersed) |
| Signs of malware infection |
| Exact malware type/variant |
| Probing rate |
| Ratio of destination overlaps |
| Target port |

TABLE 1: Criteria adopted by the Analysis Engine

To automatically infer orchestrated probing campaigns, the analysis engine leverages all the previously extracted inferences and insights related to the probing sessions/sources to build and parse a Frequent Pattern (FP) tree [47]. In such a tree, each node after the root represents a feature extracted from the probing sessions, which is shared by the sub-trees beneath. Each path in the tree represents sets of features that co-occur in the sessions, in non-increasing order of frequency of occurrences. Thus, two sessions that have several frequent features in common and are different just on infrequent features will share a common path on the tree. The analysis engine also employs the FP tree based mining method, FP-growth [47], for mining the complete set of generated frequent patterns. As an outcome, the generated patterns represent frequent and similar probing behavioral characteristics that correlate the probing sources into well-defined campaigns.

We should emphasize that such an approach that is employed by CSC-Detector's analysis engine possess the following advantages. First, the generated patterns are *not defined apriori;* they are naturally and automatically detected. This permits CSC-Detector to infer novel probing campaigns, without requiring previous knowledge about their specifications. The latter is a crucial feature, especially with the continuous evolution of probing campaigns and their employed strategies. Second, the FP-Tree not only provides the capability for the system to correlate the probing sources into campaigns, but also semantically describes how the probing sessions have been constructed. Third, by engineering parsing algorithms that traverse the FP-Tree in various ways, the system can infer horizontal probing campaigns, similar to the CSC in [6], as well as campaigns that do not focus on one port but rather probe multiple targets on various ports, similar to the CSC in [10]. Fourth, from a performance perceptive, the employed approach is scalable since probing sessions are not compared pairwise, which could lead to a quadratic complexity [47]. In fact, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



cost of the algorithm is the cost of inserting probing session features in the FP-Tree, which is linear.

### 3.4 System Operation

As mentioned in Section 1, CSC-Detector operates by scrutinizing darknet traffic. Indeed, such traffic is typically composed of three types of traffic, namely, probing, backscattered and misconfiguration [48]. Probing arises from bots, worms and tools (or binaries) while backscattered traffic commonly refers to unsolicited traffic that is the result of responses to DoS attacks with spoofed source IP addresses. On the other hand, misconfiguration traffic is due to network/routing or hardware/software faults causing such traffic to be sent to the darknet sensors.

To prepare the darknet traffic for input to CSC-Detector, we augment the system by including a pre-processing module. First, the module aggregates the connections into sessions using an approach similar to the first step algorithm by Kannan et al. [49]. Further, we consider all those connections within $T_{aggreg}$ of each other as part of the same session for a given pair of hosts. We use the same proposed threshold, $T_{aggreg}$ = 100 seconds, and found that this seems to correctly group the majority of connections between any given pair of hosts. Second, the pre-processing module attempts to filter out misconfiguration traffic. To accomplish this, the module adopts a simple metric that records the average number of sources per destination darknet address. This metric should be significantly larger for misconfiguration than probing traffic. However, although it differentiates misconfiguration from probing, it could include as well backscattered traffic as it also can possess a large average number of sources per destination (i.e, in case of a DoS). To cope with this issue, we observe, per the technique in [48], flags in packet headers, such as TCP SYN+ACK, RST, RST+ACK, ACK, etc., that resemble backscattered traffic [48]. Subsequently, the module filters out flows that lack that observation, deeming them as misconfiguration. Consequently, the output of the pre-processing module is probing and backscattered sessions that are fed as input to CSC-Detector's fingerprinting engine. From this point onwards, the system operates as previously elaborated in Sections 3.1, 3.2 & 3.3. We envision that the proposed system that is tailored towards darknet data, which is frequently, abundantly and effectively used to generate cyber threat intelligence, could be easily leveraged by any security operator, emergency response teams and/or observers of cyber events to infer large-scale orchestrated probing campaigns. Indeed, this could be utilized for early cyber attack warning and notification as well as for simplified analysis and tracking of such events.

## 4 System Evaluation

### 4.1 Implementation

We implemented the proposed CSC-Detector as a prototype in Java[3]. We utilized the jNetPcap SDK for packets/sessions processing. For all the statistical distances and tests as well as entropy and Poisson fitting, we employed their MATLAB implementations and included them in Java using the MATLAB Builder JA. We also used ssdeep, a fuzzy hashing implementation written in C, and wrapped it unto java using SWIG. We as well leveraged a python implementation of the FP-growth algorithm. We adopted a MongoDB database to save the instances of the probing sessions coupled with their generated inferences.

### 4.2 Empirical Evaluation

We evaluate CSC-Detector using one month of darknet data (240 GB) that was collected during the recent duration of April 1$^{st}$ to April 30$^{th}$, 2014. For each day in the dataset, we extracted 1,000 sessions for a total of 30,000 sessions to experiment with. Note that, the 30,000 sessions are extracted from the output of the pre-processing module (recall Section 3.4) and are selected to be generated from unique sources.

#### *4.2.1 Evaluating the fingerprinting engine*

The 30,000 sessions were fed as input to CSC-Detector. The aim is to evaluate the capability of the fingerprinting engine to differentiate between probing and backscattered traffic (i.e., DoS related activity). It was shown that probing activity corresponds to 81% (24,300) of all the sessions. This probing to backscattered traffic ratio is somehow coherent (i.e., within 4%) with other darknet studies [48]. Note that DoS related activity was fingerprinted as such since it was shown from its DFA results that 19% (5,700) of the sessions possessed a correlation status corresponding to either uncorrelated, $1/f$-noise or Brownian noise. The fact that DoS related traffic demonstrated noise or Brownian noise is compliant with what was found in [50] when the authors performed DFA analysis on DoS traffic. To further validate such inference, we implemented the DoS detection algorithm by Moore et al. [51] and applied it on the 5,700 sessions. 5,641 sessions out of the 5,700 were detected as DoS related. Thus, the fingerprinting engine seemed to have erroneously identified 59 sessions as DoS related. To understand why that occurred, we inspected the 59 sessions. 44 sessions out of the 59 possessed a DFA scaling exponent $\alpha$ ranging from 1.51 to 1.67, and accordingly were fingerprinted as Brownian noise (i.e., DoS related). However, after inspecting their traffic packets, they were shown to belong to RPC

3. For implementation challenges, performance evaluation and current status, we kindly refer the reader to http://public.eng.fau.edu/ebouharb/Supplementary.pdf

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



or Window probing. This suggests that one should not consider large $\alpha$ values as valid results or at least keep those incidents in a 'quarantine' for further automated post-processing. The remaining 15 sessions that were also erroneously fingerprinted seem to be misconfiguration that apparently, were not correctly filtered by the pre-processing module as expected.

To evaluate the accuracy of the probing fingerprinting capabilities of the engine, we initially experimented with Snort's sfPortscan pre-processor [52] using the same 24,300 sessions that were previously fingerprinted as probing. The sfPortscan preprocessor detects scans by counting RST packets from each perceived target during a predetermined timeout interval. Before declaring a scan, 5 events (i.e., RST packets) are required from a given target within a window. The sliding timeout window varies from 60 to 600 seconds by sensitivity level; at the highest level, an alert will be generated if the 5 events are observed within 600 seconds. We relied on sfPortscan's output as a baseline for our comparison. Snort's sfPortscan detected 24,234 scans. After a semi-automated analysis and comparison that was based on the logged scanning traffic flows (i.e., source and destination IP and port, protocol, and timestamp), we identified that all the 24,300 scans that CSC-Detector's fingerprinting engine had detected as probing activities included sfPortscan's 24,234 scans. Therefore, relative to this technique, we confirm that the fingerprinting engine yielded no false negative. Moreover, according to the results, the engine generated 66 sessions that could be considered as false positive. It is worthy to pinpoint that the engine can detect certain types of probes that were not included at the time of the experiment, and by default, in Snort's sfPortscan definitions. These include probes from a single host to a single port on a single host, slow probes and a specific host probing multiple ports on multiple hosts. To further assess the accuracy of CSC-Detector's fingerprinting engine, we also compared it against the Bro IDS [53], which yielded 0 false negative rate and around 4.5% false positive rate. By investigating those false positive cases, we inferred that 2% of the latter where darknet misconfiguration traces and the remaining where indeed scans that probed less than 25 hosts, which did not trigger Bro's default threshold for number of probed destinations.

We proceed by investigating the specific probing techniques that were employed in the 24,300 probing sessions. Figure 4 illustrates the output. According to the fingerprinting engine, TCP SYN probing leads with 27% of all the sessions, closely followed by ICMP Echo, TCP Connect() and UDP probing. FIN, Xmas Tree, and Null probing are typically considered as members of the 'stealth' scans because they send a single frame to a TCP port without any TCP handshaking or any additional packet transfers. They are relatively effective in evading firewall detection and they are often employed [32].

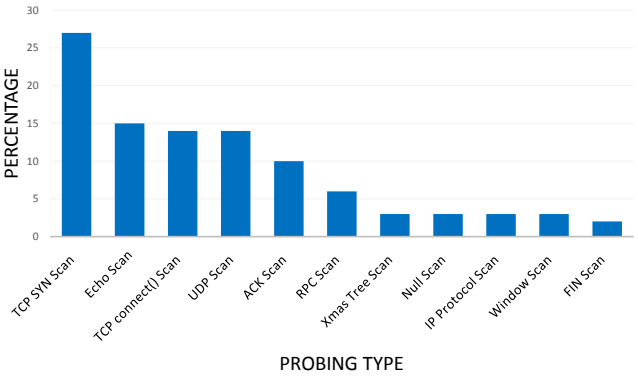



Fig. 4: Distribution of the Probing Types

#### 4.2.2 *Evaluating the inference engine*

We proceed by invoking the behavioral analytics (recall Section 3.2) that are employed by CSC-Detector's inference engine on the 24,300 probing sessions. The outcome is summarized in Figure 5. It is revealed that 62% of the probing sources used certain patterns when generating their probing traffic. Applying the CLUES clustering algorithm on the generated patterns resulted in 12 specific non-overlapping Poisson $\lambda$ intervals. We associate the probing sources to those intervals. Concerning the employed probing strategy, it is shown that 57% of the probing sources leveraged a permutation while the remaining adopted a sequential strategy when probing their targets. Of those that employed a permutation, 76% used a uniform permutation while 24% adopted other types of permutations. The majority ($\approx$ 98%) of those that employed a sequential strategy were found to adopt a forward IP-sequential strategy while only 2% adopted a reverse IP-sequential strategy. It is noteworthy to mention that Dainotti et al. [6] reported that the SIP CSC indeed used a reverse IP-sequential strategy; the authors deemed that as rare and novel. Other studies such as [29] dismissed the possible use of this strategy since, as they noted, the strategy is difficult to be used to extrapolate certain metrics from especially when dealing with partial probes. Further, the inference engine disclosed that $\approx$ 75% of the sources were probes from bots while only 15% were generated from probing tools. Moreover, it was inferred that 90% of the sources were generating probing that is dispersed while only 10% of the sources were generating probing targeted towards a small set of IPs; the average relative uncertainty of 21,870 probing sources was shown to be $> 0.72$. Last but not least, the inference engine employed the approach of Section 3.2 to investigate any signs of malware infection in the probing sessions. The output demonstrated

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



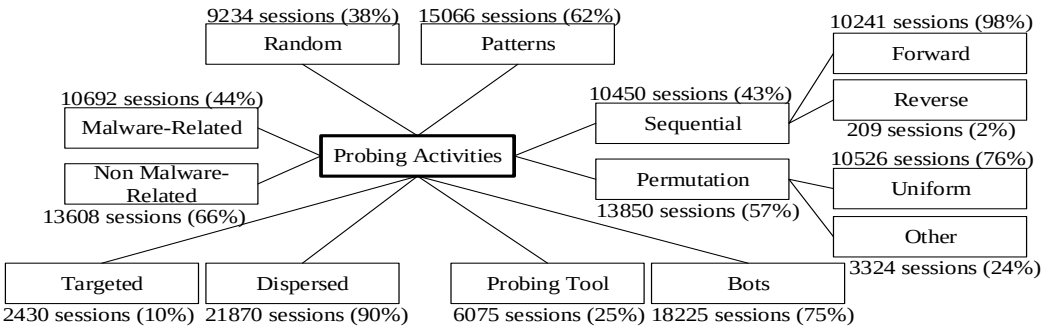



Fig. 5: The outcome of the probing behavioral analytics

that 44% of the probing sessions exhibited such signs. The top 5 malware contamination that caused the probing were attributed to the following: Trojan-FakeAV.Win32.Agent.cwa, Virus-Win32.Sality.s, Trojan-Win32.Jorik.Shakblades.foc, Trojan-Downloader.Win32.KiayksayRen.b and Worm-Win32.VBNA.b. It is noteworthy to pinpoint that the latter inference that aims at coupling probing and malware by only analyzing their generated traffic has never been attempted before. The obtained results also expressed that the average rate of all the probing sources was around 80 packets/sec, the Domain Name System (DNS) on port 53 and the Session Initiation Protocol (SIP) on port 5060 were the most probed UDP services, while TCP ports 80 and 1433 that respectively represent the HTTP and Microsoft (MS)-SQL services were the most targeted TCP services. To the best of our knowledge, the above generated inferences represent the first comprehensive empirical results of probing behaviors and characteristics. Although we are unable to validate every single inference due to the lack of its ground truth, the 3 CSCs that will be uncovered by the analysis engine (that consumes these generated inferences) will be validated using a publically available data source, namely, DShield data [16], advocating the relative accuracy of such insights.

#### 4.2.3 Evaluating the analysis engine

We proceed by invoking the analysis engine. Recall that this engine consumes the generated insights from the fingerprinting and inference engines to automatically build patterns that consist of similar probing sources behavioral characteristics. For the sake of this work, we have devised a simplistic parsing algorithm which automatically builds patterns from the FP-tree that aims at capturing orchestrated probing events that probe horizontally; probe all IPs by focusing on specific ports.

**Inferred Campaigns:** CSC-Detector automatically inferred the patterns that are summarized in Table 2, which respectively captured three different CSCs. The first pattern permitted the detection, identification and correlation of 846 unique probing bots into a well-defined orchestrated probing event that targeted the VoIP (SIP) service. It is shown that this event, that was initiated on the 17$^{th}$ of April, 2014, adopted UDP scanning, probed around 65% of the monitored dark space (i.e., 300,000 dark IPs) where all its bots did not follow a certain pattern when generating their probing traffic. Further, the results demonstrate that the bots employed a reverse IP-sequential probing strategy when probing their targets. Moreover, the malware responsible for this event was shown to be attributed to the Sality malware[4]. The second pattern successfully captured and correlated 817 unique probing bots to form a campaign that targeted the HTTP (Web) service. This campaign, that first occurred on April 3$^{rd}$, leveraged the TCP SYN scanning technique and did not adopt a pattern when generating its probing traffic. Moreover, all its bots were found to uniformly permute the dark IP space. Additionally, the malware responsible for this event was shown to be attributed to the Jorik trojan. The third campaign that was identified by the third pattern on April 16$^{th}$ was unique by targeting the MS-SQL service, leveraging ACK scanning, employing a clearly identifiable pattern, focused on a small set of targeted probing IPs, where its 438 probing sources leveraged a scanning tool and a forward IP-sequential strategy. Note that we also generate supplementary material related to the above inferred CSCs including geo-location information per real sources, organizations, Internet Service Providers (ISPs), cities, regions and countries. However, we refrain from publishing those due to sensitivity/legal issues.

**Validation:** Since currently there exist no cyber security capability to discover such large-scale orchestrated probing campaigns [7], we are unable

4. The reverse engineering of a sample of the Sality malware that was performed in [6] showed the same probing characteristics as CSC1. Further, by analyzing auxiliary darknet data pertaining to the first 6 months of 2014, it was shown that CSC1 is unique by its employed strategy and probing rate, compared with regular probing that occurs recurrently towards port 5060.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing



| | CSC1 | CSC2 | CSC3 |
|---|---|---|---|
| Employed probing technique: | UDP | TCP SYN | ACK |
| Probing traffic: | Random | Random | Pattern |
| Employed pattern: | Null | Null | [19.17-21.23] |
| Adopted probing strategy: | Reverse IP-sequential | Uniform Permutation | Forward IP-Sequential |
| Nature of probing source: | Bot | Bot | Tool |
| Type of probing: | Dispersed | Dispersed | Targeted |
| Signs of malware infection: | Yes | Yes | No |
| Exact malware type/variant: | Virus.Win32.Sality.bh | Trojan.Win32.Jorik | Null |
| Probing rate (in pps): | 12 | 118 | 77 |
| Target port: | 5060 | 80 | 1433 |
| Number of Probing Bots/Sources: | 846 | 817 | 438 |

TABLE 2: The automatically inferred patterns capturing three large-scale orchestrated probing campaigns

to directly compare the inferred CSCs with other systems or approaches. However, in an attempt to validate our results, we resort to publicly accessible data that is provided by DShield/Internet Storm Center (ISC). ISC data comprises of millions of intrusion detection log entries gathered daily from sensors covering more than 500,000 IP addresses in over 50 countries. From such data, we extract Figures 6a, 6b and 6c which respectively depict probe counts generating probing activities towards UDP port 5060 and TCP ports 80 and 1433 during the month of April, 2014. Figure 6a indeed reveals a significant peak on the $17^{th}$ of April consisting of an increasing number of probe counts targeting the SIP service; this is the same day where the first orchestrated probing campaign, that was previously inferred by CSC-Detector, was detected to be targeting the SIP service. Further, Figure 6b, according to DShield's data, shows a significant number of probes targeting the Web service on April $3^{rd}$; this is as well coherent with the second CSC that was inferred by CSC-Detector, that targeted the same service on the same day. Similarly, Figure 6c demonstrates a peak of probing packets targeting the MS-SQL service on the same day (i.e., April $16^{th}$) when the third CSC was detected to be launching its probing activities towards the MS-SQL service. Additionally, the target scope of those CSCs, also retrieved from DShield's data on those particular days, recorded an unprecedented numbers reaching millions of targets (compared with hundreds on other days). The latter information strongly corroborate that the proposed system was indeed accurately successful in inferring those unusual large-scale events. Note that, those inferred probing campaigns went undetected and unreported in the cyber security community until now.

Moreover, in an attempt to further validate our results, we have performed two additional experiments.

**First experiment:** We extracted all the 2101 source IP addresses that were pinpointed to be part of the three inferred large-scale probing campaigns. We fed those IP addresses to third party publically available data sources provided by the online services, ThreatStop, MxLookup, brightcloud and ReputationAuthority[5]. The latter cyber security data repositories provide information on Internet-scale incidents (i.e., scanning, malware, spamming, etc.) per IP address. In particular, ThreatStop indexes full, present and historical dshield data. Since the three probing campaigns were inferred in April 2014, we verified the existence of the inferred source IP addresses that belong to the campaigns against those reported in the online services in that specific time frame. The outcome of such experiment, from all the four sources, demonstrated that (1) 98% of the IP addresses of the campaigns were indeed found and flagged as malicious in those repositories and (2) around 91% of those IP addresses were specifically flagged as being involved in scanning activities.

**Second experiment:** In an attempt to further validate the phenomenon by correlating activities of source IPs, we extracted the packet features[6] generated by the IP addresses for each of those three campaigns. Subsequently, the k-means data clustering technique was applied on such features. The outcome is illustrated in Figure 7. The figure corroborates the creation of three unique clusters, representing the three inferred campaigns.

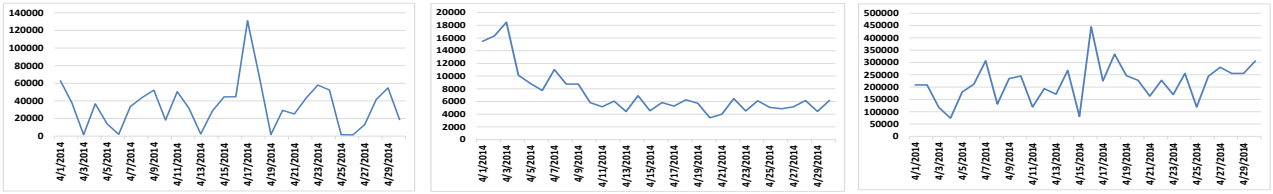
Thus, on one hand, the first experiment validates that the source IP addresses of the inferred campaigns are indeed malicious where the majority of them were found to be involved in probing activities in April 2014. On the other hand, the second experiment demonstrates that the inferred campaigns are indeed well constructed where their IP addresses coupled with their generated probing traffic are strongly correlated as revealed by the formed independent clusters.

5. {ThreatStop,MxLookup,brightcloud,borderware}.com

6. Adopted from [54], where they have been shown to produce distinguishing characteristics when applied on network data

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2593441, IEEE Transactions on Dependable and Secure Computing





(a) Probes Targeting the SIP Service (b) Probes Targeting the Web Service (c) Probes Targeting the MS-SQL Service

Fig. 6: Probe counts extracted from DShield/ISC data (April 2014)

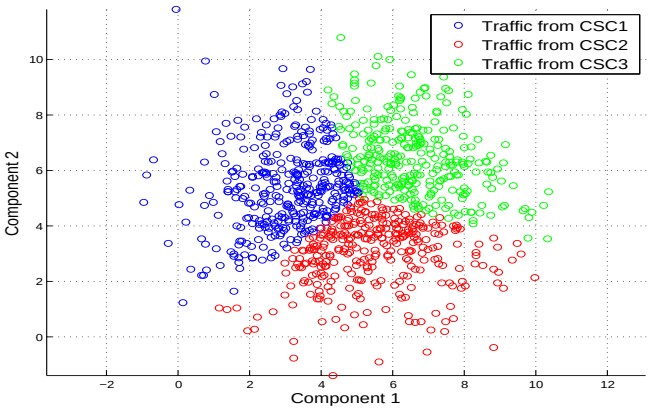



Fig. 7: K-means output

**Discussion:** The aforementioned inferred CSC1 is indeed interesting; as elaborated at the beginning of this paper, in [6], CAIDA presented an analysis study of an orchestrated probing campaign that targeted VoIP (SIP) servers. According to CAIDA, the event occurred from January 31$^{st}$, 2011 till February 12$^{th}$, 2012. CSC1 that the proposed system was able to *automatically* infer possesses the exact characteristics of that CAIDA event. They both were attributed to the Sality malware, generated UDP scanning, targeted SIP servers on port 5060, and used a reverse IP-sequential probing strategy. The latter behavior is predominantly stimulating because, as previously mentioned, that strategy is known to be extremely under-employed [29]. However, one distinguishing feature between those two events is that the probing bots of the campaign that the proposed system was able to infer has considerably lower probing rate than those of CAIDA's event; on average, the bots of the inferred campaign recorded 12 packets per second (pps) while those of CAIDA measured around 60 pps. Such information (1) arguably proves that CAIDA's event is indeed still active, yet operating in a stealthy, very low rate mode in an attempt to achieve its reconnaissance task without being detected or (2) a new instance of the same orchestrated event commanded by the same C&C took place in April, 2014 without any cyber security body reporting it. In either cases, we find the latter information motivating and to a certain degree bewildering. Thus, we aim in the near future work to track that event to verify and elaborate on its inner details. It is worthy to mention that in the month of April 2014, DShield, the media or other sources did not announce a campaign that CSC-Detector was not able to infer.

## 5 Concluding Remarks

This paper presented CSC-Detector, a systematic effort towards the challenging goal of detecting and identifying large-scale orchestrated probing campaigns specifically tailored towards the dark IP space. The system was primarily motivated by the prompt need for such a cyber security capability as stated by CAIDA [6, 7]. The system's fingerprinting engine exploits a unique observation to extract probing activities from darknet traffic. Consequently, CSC-Detector's inference engine employs a set of behavioral analytics to generate insights that capture numerous characteristics of the probing sources. Last but not least, the system's analysis engine leverages several criteria, methods and approaches to automatically identify and correlate the probing sources into well-defined campaigns. CSC-Detector was empirically evaluated using significant amount of real darknet data. The recently inferred CSCs, which were validated using DShield data, indeed demonstrates that the system exhibits promising accuracy and can generate substantial evidence to infer diverse large-scale probing campaigns.

### 5.1 Lessons Learned & Future Work

Although we did not have the opportunity to operate and experiment with CSC-Detector for a long period of time, however, we can extract some points that we recently have had the chance to observe:

- We are often faced with the case in which the probing bot sends two probing packets to the same destination but with two different destination port numbers. For example, if the intention of the scanner/campaign is to probe for the SIP service, the scanner might send two probing packets towards ports 80 and 5060. This was also observed and concurred in [6]. CSC-Detector is frequently identifying those bots as belonging to

two different CSCs. One solution to compensate this issue would be to permit multiple port number assignments in the target IP criteria through the analysis engine.

- Most of the probing bots that the system is identifying as belonging to the same CSC are often very geographically dispersed. This suggests that it is very hard if not impossible to mitigate their probes or their possible subsequent cyber attacks by blocking them based on their location, ISP or Autonomous System Number (ASN).
- Typically, we would expect that probing bots within the same correlated CSC to possess similar probing rates, as they are coordinated by the same botmaster in a C&C infrastructure that characteristically adopt a unified probing strategy. For the three CSCs that the system was able to infer, the latter was found to be **false** (using a 90% confidence interval); perhaps this implies that the same botmaster is requiring different bot groups within the same campaign to adopt different probing strategies in an attempt to avoid being attributed to the same campaign. In general, we deem this behavior as stimulating, uncommon and somehow confusing to understand and interpret.

As for future work, we plan to perform CSC intent analysis; the ability to infer what the probing CSC bots will eventually execute after finalizing their probing activities. We aim to achieve the latter by correlating the generated inferences from this work with other data sources, including but not limited to, passive dns, and public intrusion and firewall logs.

## REFERENCES


[1] Hackers Exploited Heartbleed Bug to Steal 4.5 Million Patient Records, 2014. http://tinyurl.com/jleuqtq.

[2] WordPress sites targeted by mass brute-force attack, 2013. http://tinyurl.com/z9479h5.

[3] Iranian Cyber-Attackers Target US Energy Companies, 2013. http://tinyurl.com/o9r8u3r.

[4] S. Panjwani, S. Tan, K.M. Jarrin, and Michel Cukier. An experimental evaluation to determine if port scans are precursors to an attack. In *Proceedings of the International Conference on Dependable Systems and Networks. DSN 2005*, pages 602–611, 2005.

[5] J. Treurniet. A network activity classification schema and its application to scan detection. *Networking, IEEE/ACM Transactions on*, 19(5):1396–1404, Oct 2011.

[6] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescap. Analysis of a "/0" Stealth Scan from a Botnet. *IEEE/ACM Transactions on Networking*, 2014.

[7] Alberto Dainotti, Alistair King, and Kimberly Claffy. Analysis of internet-wide probing using darknets. In *Proceedings of the 2012 ACM Workshop on Building analysis datasets and gathering experience returns for security*, BADGERS '12, pages 13–14, New York, NY, USA, 2012. ACM.

[8] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. A systematic approach for detecting and clustering distributed cyber scanning. *Computer Networks*, 57(18):3826–3839, 2013.

[9] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. A statistical approach for fingerprinting probing activities. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 21–30. IEEE, 2013.

[10] Internet Census 2012-Port scanning /0 using insecure embedded devices. http://tinyurl.com/c8af8lt.

[11] J. Treurniet. A network activity classification schema and its application to scan detection. *IEEE/ACM Transactions on Networking,*, 19(5):1396–1404, 2011.

[12] S. Staniford, J.A. Hoagland, and J.M. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1/2):105–136, 2002.

[13] W. Zhang, S. Teng, and X. Fu. Scan attack detection based on distributed cooperative model. In *Proceedings of the International Conference on Computer Supported Cooperative Work in Design, CSCWD 2008*, pages 743–748. IEEE, 2008.

[14] R. Baldoni, G. Di Luna, and L. Querzoni. Collaborative Detection of Coordinated Port Scans. Technical report, 2012. http://www.dis.uniroma1.it/~midlab.

[15] G. Conti and K. Abdullah. Passive visual fingerprinting of network attack tools. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 45–54. ACM, 2004.

[16] Internet Storm Center & DShield. https://isc.sans.edu/threatfeed.html.

[17] Michael Bailey, Evan Cooke, Farnam Jahanian, Jose Nazario, David Watson, et al. The internet motion sensor-a distributed blackhole monitoring system. In *NDSS*, 2005.

[18] Vinod Yegneswaran et al. On the design and use of internet sinks for network abuse monitoring. In *RAID*, volume 3224, pages 146–165. Springer Berlin Heidelberg, 2004.

[19] Security Information Exchange (SIE). Farsight Security Inc. https://www.farsightsecurity.com.

[20] Zhichun Li, A. Goyal, Yan Chen, and V. Paxson. Towards situational awareness of large-scale botnet probing events. *IEEE Transactions on Information Forensics and Security,*, 6(1):175–188, 2011.

[21] Yu Jin, György Simon, Kuai Xu, Zhi-Li Zhang, and Vipin Kumar. Grays anatomy: Dissecting scanning activities using ip gray space analysis. *Usenix SysML07*, 2007.

[22] Yu Jin, Zhi-Li Zhang, Kuai Xu, Feng Cao, and Sambit Sahu. Identifying and tracking suspicious activities through ip gray space analysis. In *Proceedings of the 3rd annual ACM workshop on Mining network data*, MineNet '07, pages 7–12, New York, NY, USA, 2007. ACM.

[23] Zhichun Li et al. Automating analysis of large-scale botnet probing events. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 11–22, New York, NY, USA, 2009. ACM.

[24] Vinod Yegneswaran, Paul Barford, and Vern Paxson. Using honeynets for internet situational awareness. *In Proc. of ACM Hotnets IV*, 2005.

[25] Darcy Benoit and André Trudel. World's first web census. *International Journal of Web Information Systems*, 3(4):378, 2007.

[26] John Heidemann et al. Census and survey of the visible internet. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 169–182, New York, NY, USA, 2008. ACM.

[27] Y Pryadkin, R Lindell, J Bannister, and R Govindan. An empirical evaluation of ip address space occupancy. *USC/ISI Technical Report ISI-TR*, 598, 2004.

[28] Ang Cui and Salvatore J. Stolfo. A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 97–106, New York, NY, USA, 2010. ACM.

[29] Derek Leonard and Dmitri Loguinov. Demystifying service discovery: implementing an internet-wide scanner. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pages 109–122, New York, NY, USA, 2010. ACM.

[30] Guofei Gu et al. Bothunter: detecting malware infection through ids-driven dialog correlation. In *Proceedings of 16th USENIX Security Symposium*, SS'07, pages 12:1–12:16, Berkeley, CA, USA, 2007. USENIX Association.

[31] Jan Goebel and Thorsten Holz. Rishi: Identify bot contaminated hosts by irc nickname evaluation. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets (USENIX HotBots)*, pages 8–8. Cambridge, MA, 2007.

[32] E. Bou-Harb, M. Debbabi, and C. Assi. Cyber scanning: A comprehensive survey. *Communications Surveys & Tutorials, IEEE*, PP(99):1–24, 2013.

[33] C-K Peng, Sergey V Buldyrev, Shlomo Havlin, M Simons, H Eugene Stanley, and Ary L Goldberger. Mosaic organization of dna nucleotides. *Physical Review E*, 49(2):1685, 1994.

[34] K. Fukuda, T. Hirotsu, O. Akashi, and T. Sugawara. Correlation among piecewise unwanted traffic time series. In *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM 2008*, pages 1–5, 2008.

[35] T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology,*, 15(1):52–60, 1967.

[36] Zhichun Li, Anup Goyal, and Yan Chen. Honeynet-based botnet scan traffic analysis. In *Botnet Detection*, pages 25–44. Springer, 2008.

[37] Xiaogang Wang et al. Clues: A non-parametric clustering method based on local shrinking. *Computational Statistics & Data Analysis*, 52(1):286–298, 2007.

[38] Genevieve Bartlett, John Heidemann, and Christos Papadopoulos. Understanding passive and active service discovery. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 57–70, New York, NY, USA, 2007. ACM.

[39] Stuart Staniford, Vern Paxson, Nicholas Weaver, et al. How to own the internet in your spare time. In *USENIX Security Symposium*, pages 149–167, 2002.

[40] Sushil Jajodia, Peng Liu, Vipin Swarup, and Cliff Wang. *Cyber Situational Awareness: Issues and Research*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[41] ThreatAnalyzer: Dynamic Malware Analysis. http://www.threattracksecurity.com/enterprise-security/malware-analysis-sandbox-tools.aspxl.

[42] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.

[43] Jesse Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, 3, Supplement(0):91 – 97, 2006.

[44] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. *SIGCOMM Comput. Commun. Rev.*, 35(4):169–180, August 2005.

[45] Hubert W Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318):399–402, 1967.

[46] Moheeb Abu Rajab et al. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 41–52, New York, NY, USA, 2006. ACM.

[47] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages 1–12, New York, NY, USA, 2000. ACM.

[48] Eric Wustrow et al. Internet background radiation revisited. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pages 62–74, New York, NY, USA, 2010. ACM.

[49] Jayanthkumar Kannan et al. Semi-automated discovery of application session structure. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 119–132, New York, NY, USA, 2006. ACM.

[50] Uli Harder et al. Observing internet worm and virus attacks with a small network telescope. *Electronic Notes in Theoretical Computer Science*, 151(3):47 – 59, 2006.

[51] David Moore et al. Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24(2):115–139, May 2006.

[52] Daniel Roelker, Marc Norton and Jeremy Hewlett. sfportscan, 2004. http://projects.cs.luc.edu/comp412/dredd/docs/software/readmes/sfportscan.

[53] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[54] Riyad Alshammari and A Nur Zincir-Heywood. Can encrypted traffic be identified without port numbers, ip addresses and payload inspection? *Computer networks*, 55(6):1326–1350, 2011.


**Elias Bou-Harb** is currently an Assistant Professor at the computer science department at Florida Atlantic University. Previously, he was a visiting research scientist at Carnegie Mellon University. Elias is also a research scientist at the National Cyber Forensic and Training Alliance (NCFTA) of Canada. Elias holds a Ph.D. degree in computer science from Concordia University, Montreal, Canada. His research and development activities and interests focus on the broad area of operational cyber security, including, attacks detection and characterization, Internet measurement, cyber security for critical infrastructure and mobile network security.

**Chadi Assi** received his B.Eng. degree from the Lebanese University, Beirut, Lebanon, in 1997 and his Ph.D. degree from the City University of New York (CUNY) in April 2003. He is currently a full professor with the Concordia Institute for Information Systems Engineering, Concordia University. Before joining Concordia University in August 2003 as an assistant professor, he was a visiting researcher with Nokia Research Center, Boston, Massachusetts, where he worked on quality of service in passive optical access networks. His research interests are in the areas of networks and network design and optimization. He received the prestigious Mina Rees Dissertation Award from CUNY in August 2002 for his research on wavelength-division multiplexing optical networks. He is on the Editorial Board of IEEE Communications Surveys & Tutorials, IEEE Transactions on Communications, and IEEE Transactions on Vehicular Technologies.

**Mourad Debbabi** is a Full Professor at the Concordia Institute for Information Systems Engineering. He holds the Concordia Research Chair Tier I in Information Systems Security. He is also the President of the National Cyber Forensics Training Alliance (NCFTA Canada). He is the founder and one of the leaders of the Computer Security Laboratory (CSL) at Concordia University. In the past, he was the Specification Lead of four Standard JAIN (Java Intelligent Networks) Java Specification Requests (JSRs) dedicated to the elaboration of standard specifications for presence and instant messaging. Dr. Debbabi holds Ph.D. and M.Sc. degrees in computer science from Paris-XI Orsay, University, France. He published 2 books and more than 230 research papers in journals and conferences on computer security, cyber forensics, privacy, cryptographic protocols, threat intelligence generation, malware analysis, reverse engineering, specification and verification of safety-critical systems, formal methods, Java security and acceleration, programming languages and type theory. He supervised to successful completion 20 Ph.D. students and more than 60 Master students. He served as a Senior Scientist at the Panasonic Information and Network Technologies Laboratory, Princeton, New Jersey, USA; Associate Professor at the Computer Science Department of Laval University, Quebec, Canada; Senior Scientist at General Electric Research Center, New York, USA; Research Associate at the Computer Science Department of Stanford University, California, USA; and Permanent Researcher at the Bull Corporate Research Center, Paris, France.