Towards Situational Awareness of Large-scale Botnet Probing Events

Zhichun Li *Member, IEEE*, Anup Goyal, Yan Chen *Member, IEEE*, and Vern Paxson *Senior Member, IEEE*

***Abstract*—Botnets dominate today's attack landscape. In this work we investigate ways to analyze collections of malicious probing traffic in order to understand the significance of large-scale "botnet probes". In such events, an entire collection of remote hosts together probes the address space monitored by a sensor in some sort of coordinated fashion. Our goal is to develop methodologies by which sites receiving such probes can infer—using purely *local* observation—information about the probing activity: What scanning strategies does the probing employ? Is this an attack that specifically targets the site, or is the site only incidentally probed as part of a larger, indiscriminant attack?**

Our analysis draws upon extensive honeynet data to explore the prevalence of different types of scanning, including properties, such as trend, uniformity, coordination, and darknet avoidance. In addition, we design schemes to extrapolate the global properties of scanning events (*e.g.*, total population and target scope) as inferred from the limited local view of a honeynet. Cross-validating with data from *DShield* shows that our inferences exhibit promising accuracy.

***EDICS*—SEC-NETW Network security $<$ SECURITY & PRIVACY ANALYSIS**

***Index Terms*—Computer network security, Site security monitoring, Botnet, Global property extrapolation, Honeynet, Scan strategy inference, Situational awareness, Statistical inference**

I. Introduction

When a site receives probes from the Internet—whether basic attempts to connect to its services, or apparent attacks directed at those services, or simply peculiar spikes in seemingly benign activity—often what the site's security staff most wants to know is not "are we being attacked?" (since the answer to that is almost always "yes, all the time") but rather "what is the *significance* of this activity?" Is the site being deliberately targeted? Or is the site simply receiving one small part of much broader probing activity?

For example, suppose a site with a /16 network receives malicious probes from a botnet. If the site can determine that the botnet probed only their /16, then they can conclude that the attacker may well have a special interest in their enterprise. On the other hand, if the botnet probed a much larger range, *e.g.*, a /8, then very likely the attacker is not specifically targeting the enterprise.

The answers to these questions greatly influence the resources the site will choose to employ in responding to the activity. Obviously, the site will often care more about the probing if the attacker has specifically targeted the site, since such interest may reflect a worrisome level of determination on the attacker. Indeed, such targeted attacks have recently grown in prominence. For example, targeting *New York Times*, an attacker penetrated into the site through scanning and then stole more than 3,000 social security numbers [1]. Yet given the incessant level of probing all Internet addresses receive [2], how can a site assess the risk a given event reflects?

In this work we seek to contribute to the types of analysis that sites can apply to gauge such risks. We orient much of our methodology with an assumption that most probing events reflect activity from *botnets* (*i.e.*, coordinated bots) that dominate today's Internet attack landscape. Our approach aims to analyze fairly large-scale activity that involves multiple local addresses. As such, our techniques are suitable for use by sites that deploy *darknets* (unused subnets), *honeynets* (subnets for which some addresses are populated by some form of honeypot responder), or in general any monitored networks with unexpected access, for which we can detect the botnet probing events. The main contribution of this paper is the development of a set of techniques for analyzing botnet events, most of which do not require the use of responders. For simplicity, we will refer to the collection of sensors as the site's Sensors.

In contrast to previous work on botnets, which has focused on either host-level observations of single instances of a botnet activity, studies of particular captured botnet binaries [3], or network-level analysis of command-and-control (C&C) activity [4], our techniques aim to characterize facets of large-scale botnet probing events regardless of the nature of the botnet. Our analysis does not require assumptions about the internal organization and communication mechanisms employed by the botnets. We focus on the botnet inference and characterization through its probing behavior. In addition, our approach has the significant benefit of requiring only *local* information, although such inferences may possibly be also achievable by using a collaborative effort such as DShield [5], subject to with certain limitations. We give more detailed comparisons in Section VII.

We frame the contributions of our work as follows. First, we develop a set of statistical approaches to assess the attributes of large-scale probing events seen in Sensors, including checking for trends, uniformity, coordination, and hit-lists (liveness) (Section IV). Here we mainly focus on checking a special kind of hit-lists, liveness-aware scanning, in which the attackers try to avoid the darknets. For trend and uniformity checking, the statistical literature provides apt techniques, but for assessing coordination and use of hit-lists (liveness) we needed to develop new techniques. We confirmed the consistency of the statistical techniques for inferring event properties with manual inspection or visualization.

Applying such statistical testing on massive honeynet traffic

Manuscript received April 18, 2010; revised May 18, 2010; accepted September 11, 2010. Date of publication March 01, 2010; date of current version May 14, 2010. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. R. Sekar

Z. Li is with NEC Laboratories America, Inc., Princeton, NJ 08540 USA.

A. Goyal is with Yahoo! Search, Sunnyvale, CA 94089 USA.

Y. Chen is with the Electrical Engineering and Computer Science Department, Northwestern University, Evanston, IL 60208 USA .

V. Paxson is with the Electrical Engineering and Computer Sciences Department, UC Berkeley, Berkeley, CA 94704 USA, and also with the International Computer Science Institute (ICSI), Berkeley, CA 94704 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier XX.XXXX/TIFS.201X.XXXXXX

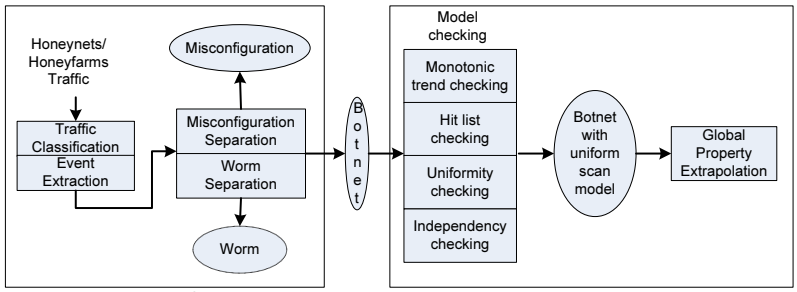

Fig. 1. System architecture.

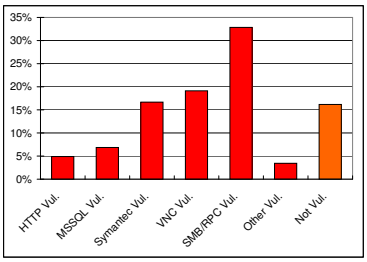

Fig. 2. The distribution of the malicious payload discovered in the scan events.

reveals some interesting and sophisticated botnet scan behaviors such as hit-list scans. We then used our suite of tests to frame the scanning strategies employed during different probe events, from which we can further extrapolate the global properties for particular strategies.

Second, we devise two algorithms to extrapolate the global properties of a scanning event based on a sensor's limited local view. These algorithms are based on different underlying assumptions and exhibit different accuracies. But both enable us to infer the global scanning scope of a probing event, as well as the total number of bots including those unseen by the Sensors, and the average scanning speed per bot (Section V). The global scanning scope enables the site's operators to assess whether their network is a specific target of botnet activity, or whether the botnet's scanning targets a large network scope that simply happens to include the site. The total size of botnet estimates can help us track trends in how botnets are used, with implications for their C&C capabilities.

Also, we find most of these probes include attacks. As shown in Figure 2, our honeynet measurements find that about 84% of scan events carry malicious payloads targeting vulnerabilities of different protocols, such as SMB/RPC, MSSQL, VNC, *etc.*[1] These attacks might be the prelude of more serious penetration; therefore they are dangerous. Moreover, botnet scans are one key technique employed for botnet recruitment [4]. Through event correlation study, we also find some interesting behaviors of how botmasters control their bots.

To validate our estimates of the global properties, we compare our results with those from DShield [5], the Internet's largest global alert repository. We find that in 75% of cases, our extrapolated scope is within a factor of 1.35 of the scan scope observed in DShield data. In all the cases it is within a factor of 1.5. The results demonstrate that our approaches are accurate enough to enable sites to make reliable inferences. Furthermore, we emulate targeted attacks and show our approach indeed can detect them.

II. System Framework

The architecture of our design is shown in Figure 1. The system has two subsystems: botnet detection and botnet inference. In this paper we focus on the latter (righthand half of Figure 1). All of the steps in our system are automated, most of them fully so. We mainly use the Honeynet sensor to drive the rest of the discussion, although other data collecting sensors can be used as well.

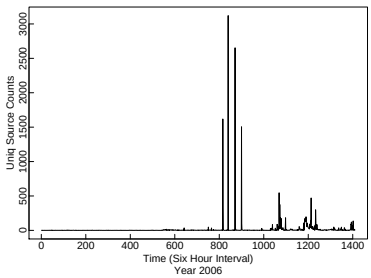

Fig. 3. Temporal distribution of source count for VNC.

[1]"Not Vul." consists of instances where the honeynet received little or no payload, or purely service testing probes.

A. Honeynet and Data Collection

Our detection sensor consists of ten contiguous /24 subnets within one of Lawrence Berkeley National Laboratory's /16 networks. We deployed Honeyd responders [6] on five of the subnets and operated the other five completely "dark". (We use this latter for hit-list (liveness) detection.) The Honeyd configuration is similar to that used by Pang *et al.* in [2]: we simulate the HTTP, NetBIOS, SMB, WINRPC, MSSQL, MYSQL, SMTP, Telnet, DameWare protocols, with echo servers for all other port numbers. We evaluate our analysis techniques using 293 GB of trace data collected over two years (2006 and 2007).

B. Botnet Detection Subsystem

We define a *botnet event* as a group of coordinated bots probing the target network with the same goal, where "same goal" means that the probes use the same protocol(s) and, if visible, protocol/session semantics. We define a *session* as a set of connections between a pair of hosts with a specific purpose, perhaps involving multiple application protocols. Sessions occur when the botmaster commands the bots to probe in a similar fashion, reflecting the same underlying bot software. (Previous works [4], [7] suggest this is indeed the case.) Since the events of interest reflect *coordinated* bot activity, we presume that the botmaster commands the bots to probe in the same time frame.

This behavior manifests as a large number of unique sources arriving at the detection sensor in a short time window for a given protocol or protocol/session semantics. Worms or misconfigurations can also manifest such traffic spikes. Therefore, we need to further differentiate types of probing. For example, Figure 3 shows source arrival counts for VNC (TCP port 5900) for the year 2006, where each point represents the number of sources within a six-hour interval. Large spikes correspond to scanning from worms, botnets, or misconfigurations.

We identify the botnet events from the traces using three steps. First, through traffic classification we separate the traffic by different protocols or protocol/session semantics. Second, for each stream of traffic, we identify large spikes of unique source arrivals, which correspond to worm, botnet or misconfiguration events. Lastly, we separate worm and misconfiguration events from botnet events.

Traffic Classification: Attack traffic can have complex session structures involving multiple application protocols. For example, the attacker can send an exploit to TCP port 139 which, if successful, results in opening a shell and issuing an HTTP download command. In general, the application protocol contacted first is emblematic of the probing goal, so we label the session with the first protocol used. Doing so provides consistent labelling for those connection attempts where the

honeynet did not respond, for which we only observe the initial SYN packets. We aggregate the connections into sessions using an approach similar to the first step algorithm by Kannan *et al.* [8]. We consider all those connections within T_{aggreg} of each other as part of the same session for a given pair of hosts. We used the same threshold, $T_{aggreg} = 100$ seconds, and found that this appeared to correctly group the majority of connections between any given pair of hosts.

For application protocols not commonly used, the average background radiation noise is low and thus we can employ port numbers to extract event traffic. However, noise is usually quite high for more popular protocols, requiring further differentiation. Assuming that we observe at least one successful session from each sender, we can use the payload analysis of that session to separate it from other traffic. We use a similar approach for the *Radiation-analy* summaries proposed in [9].

Event Extraction: By detecting large spikes of unique source counts as *events*, we can gain insight of botnets, worms and misconfigurations. Formally, the problem is to recover the signal in a noisy time series. Potentially, many signal detection and reconstruction techniques can be used. Here, we use a simple semi-automated approach to discover the events.

We define the noise strength N as the typical unique source count in the absence of events. We calculate N as the median of unique source counts of T_N time intervals *before* the event. We define signal strength S as the peak unique source count arrival X minus the noise strength N, *i.e.*, $S = X - N$, and define the signal-to-noise ratio as *SNR*$= \frac{S}{N} = \frac{X-N}{N} = \frac{X}{N} - 1$. In this paper we use a six-hour time intervals. Since after we extract an event we refine it into smaller time intervals, the time interval select here does not influence the final results much. We use $T_N = 120$ (30 days) and $SNR \geq 50$ to identify the events.

We calculate the unique source count of every time interval, and perform event extraction using time series analysis. While many general statistical signal detection approaches might be applied here, we currently extract the events semi-automatically. We first automatically identify and extract the rough boundaries of events, and then manually refine the event starting and ending times.

We automatically extract potential events as follows: for a given time interval, we calculate the median of the previous T_N intervals and the *SNR*. For those spikes exceeding our *SNR* threshold, we extend the range until $S \leq \omega N$ where ω is a tunable parameter controlling the amount of the signal tail to include in the event. For multiple events within one time series, we extract the events iteratively, starting with the event with largest *SNR*.

After extracting an event, we further refine it by re-scaling it into smaller time intervals and recalculating the unique source counts. We use manual analysis and visualization techniques at this point to refine the event starting and ending times.

Misconfiguration and Worm Seperation: We separate misconfigurations from worms and botnets based on the presumption that botnet scans and worms will contact a significant range of the IP addresses in the sensor, whereas events with few hotspots repeatedly targeted are more likely due to misconfigurations. We use two metrics to separate misconfiguations from other events. The address hit ratio, N_E/N_D, where N_E is the number of destination addresses involved in the event and N_D is the number of destination addresses in the honeynet, should be much smaller for misconfigurations than for botnet sweeps or worms. Secondly, the average number of sources per destination address should be much larger for misconfigurations. If the first metric is below a given threshold while the second crosses a given threshold, we consider the event to be a misconfiguration; otherwise it is classified as a worm or botnet event. We found that almost all misconfiguration events are due to P2P traffic, as analyzed in [10].

In general, probing from worms (self-propagating processes) can look very similar to that from botnets (processes under a common C&C), and indeed the line between the two can blur depending on the nature of the commands that botmasters issue to their bots. For our purposes, we identify and remove as worms those events that exhibit an exponential growing trend (per the technique developed in [11]) and deem the remainder as botnet probing events.

C. Botnet Inference Subsystem

For botnet probing, there are numerous scanning strategies that attackers can potentially use. Identifying the particular approach can provide a basis to infer further properties of the events and perhaps of the botnets themselves. We refer to these strategies as *scan patterns*, and undertake to develop a set of scan-pattern checking techniques to understand different dimensions of such strategies: (i) monotonic trend checking, (ii) hit list checking, (iii) uniformity checking, and (iv) dependency checking. For details, see Section IV. Once we identify a probing event's scan pattern, we then use the scan pattern to extrapolate global properties of the event. We focus on two of the most common scan patterns: uniform random scanning, and uniform hit-list (liveness) scanning. We confirm their common use both from botnet source code analysis (Section III-A) and experimental observations (Section VI). We then extrapolate the global properties, such as the global scan scope and the global number of bots, using techniques developed in Section V.

III. Design Space Of Botnets Scan Patterns

In this section we analyze different facets of how bots—and thus, in aggregate, botnets—scan a target range of addresses. We refer to different scan strategies as different *scan patterns*, where each reflects a unique set of characteristics.

A. Bot Source Code Study

By analyzing the source code of five popular families of bots [7], [12], we studied different dimensions of scan strategies employed by botnets. Our findings confirm those in [12], but we studied scan patterns for each family in greater detail.

Overall, we find they employ simple scanning strategies. Each supports both *Global* scanning (a specified address block) and *Local* scanning (relative to each bot's address). None of the five directly automates hit-list (liveness) scanning, but an attacker can potentially achieve this via two steps: first, scanning to gather a list of live addresses/blocks; and then specifying these at the command line. By hit-list (liveness) scanning, we refer to an event for which the attacker appears to have previously acquired a specific list of targets. Such scans may heavily favor the use of "live" addresses (those that respond) to "dark" (non-responsive) addresses. In addition, most bot families support (uniformly) *Random* and *Sequential* scanning of the designated addresses or blocks.

Our dataset analysis accords with the above capabilities: most scanners we observe either use simple sequential scanning (IP address increments by one between scans) or independent uniform random scanning. We do observe more sophisticated monotonic trends (address incrementing by k), but very infrequently. We also observe botnets using hit-list (liveness) scanning quite frequently.

B. Features of Botnet Global Scan Patterns

There is a large design space for botmasters when developing scan strategies, but we expect the following features to usually manifest:

- **Cover the target scope fully.** Botmasters may want to scan every address within the target scope.
- **Distribute the load based on bots' capabilities.**
- **Low communication overhead for coordination.**
- **Scan detection evasion.** Botmasters may want bots to avoid aggressive scanning of a small address range, to avoid easy detection and blocking by IDS/IPS systems.
- **Redundancy.** Since the bots in a botnet can readily be lost due to detection or due to the host computer going offline, the botmaster will prefer instructing multiple bots to scan the same addresses.

Given these desired features, a simple and effective approach is to ask each bot to independently scan the specified range in a random uniform fashion. Doing so can achieve the scan detection evasion, low communication overhead, and load distribution, while also providing good coverage and redundancy. This approach is also simple to correctly implement. In the source code analysis we find the most popular such one implemented to date (four out of five bot families implemented this strategy). Most of the events we found in our datasets are close to uniform scanning. For the hit list cases we observed, we also found that it is likely to observe uniform scans of live IP blocks.

Advanced Scanning Strategies:

Independent uniform scanning, where each source independently scans the given range, is not optimal in either coverage or redundancy. For example, if d scans are sent out uniformly to d address, the coverage is only $(1 - 1/d)^d \approx 0.68$. We can address this shortcoming by the addition of coordination between the scanning sources. That is, make the senders have certain negative dependencies so that the senders can incur fewer scan collisions.

An advanced scanning strategy, called "worm scan permutation", was proposed in the context of worm propagation [13]. In that strategy, each worm uses the same predefined key to permute the IP scope, and then randomly chooses a starting point in the permutation sequence. The worm scans until it discovers a vulnerable host that has already been compromised, at which point it randomly chooses a new start point within the permutation sequence.

But the above strategy is optimized for worms and does not consider the usage of C&C channel of botnets. Using the botnet C&C for coordination, we frame new scan strategy, *advanced botnet permutation scan* (ABPS). Each bot permutes the whole IP address scope in the same way based on a key from the botmaster. Then drawing upon the bots' capabilities, the botmaster divides the ϕ replicates of the permuted IP scope across all of the bots. This can achieve much better coverage and redundancy. We simulate and evaluate this strategy in our evaluation.

In general, there can be many different ways to design collaborative scanning for botnets, even with relatively small communication overhead as ABPS achieves. Such strategies tend to have very good coverage, resilience to scan detection, and redundancy in the presence of failure. Although currently not yet prevalent, we still consider this issue and develop a dependency-checking scheme to detect them (Section IV-D). This can help us monitor whether botnets have adopted more advanced coordination approaches.

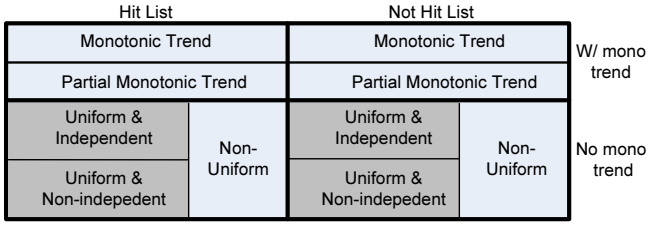

Fig. 4. Property Checking Design Space.

IV. Property Checking of Botnet Scan Patterns

In this section we develop a set of analysis algorithms. Each is designed to check a single dimension of characteristics in the scan pattern. Then we combine the characteristics of an event to construct the scan pattern in use, as shown in Figure 4.

We first classify the scan traffic pattern into monotonic, partially monotonic and non-monotonic trends. For non-monotonic trend, we assess the possible use of a hit-list or random-uniform scanning (even distribution of scans across the portion of the sensor space). Finally, for random-uniform pattern we test whether the senders can be modeled as independent.

A. Monotonic Trend Checking

Question: Do senders follow a monotonic trend in their scanning?

Monotonically scanning the destination IP addresses (*e.g.*, sequentially one after another) is a scan strategy widely used by network scanning tools. In our evaluation, we did find a few events that use the monotonic trend scanning. Furthermore, for random events, the monotonic trend checking can help filter out the noises caused by the non-bot scanners.

For each sender, we test for monotonicity in targeting by applying the Mann-Kendall trend test [14], a non-parametric hypothesis testing approach. In our study, we set the significance level to 0.5%, since a higher significance level will introduce more false positives and we need to check thousands of sources. In our evaluation, we manually check the statistical power and find it high enough to detect weak trends. The intuition behind this test is that if the data have a monotonic trend, the aggregated sign value($>\rightarrow 1$; $=\rightarrow 0$; $<\rightarrow -1$.) of all the consecutive value pairs would be out of the range the randomness can achieve.

We label an entire event as having a *monotonic trend* if more than 80% of senders exhibit a trend. We instead label the event as *non-monotonic* if more than 80% of senders do not exhibit a trend. We label the remainder as *partial monotonic*.

B. Hit-List (Liveness) Checking

Question: Do the bots use a target hit-list (list of live IP blocks) for scanning?

By hit-list (liveness) scanning, we refer to an event for which the attacker appears to have previously acquired a specific list of targets. Hit-list is often employed by sophisticated botmasters to achieve high scan efficiency. It is important for the network administrators to know whether they are in the hit-list, which indicate whether they will be scanned again and again. We detect the use of a hit-list based on the observation that such scans should heavily favor the use of "live" addresses (those that respond) to "dark" (non-responsive) addresses.

To this end, we operate half of our sensor region in a live fashion and half dark. If we observe an event only in the Honeynet portion, this provides strong evidence that the scan used a hit list. However, one consideration is event "pollution"

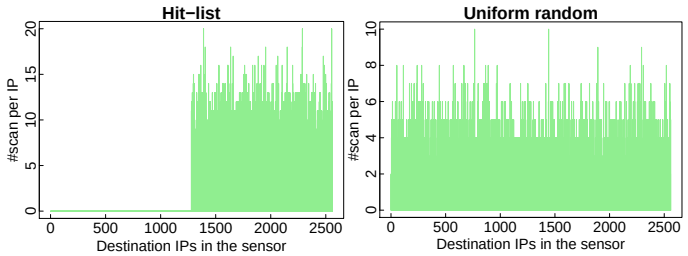

Fig. 5. Hit-list and uniform scanning on the sensor.

(sources that actually are background noise rather than part of the botnet). We do not require a *complete* absence of darknet scanning, instead test for the prevalence of honeynet scans over darknet scans significantly exceeding what we would expect.

Figure 5 compares an hit-list (liveness) event (WINRPC-070625) versus a random-uniform event (VNC-060729). To distinguish between two such cases, we define the ratio of the number of senders which target the darknet (m_d) over those of the honeynet (m_h) as $\theta = \frac{m_d}{m_h}$. Then we test whether θ crosses a given threshold. Our evaluation suggests the results are not sensitive to the threshold we choose.

Note that for the events that require application-level analysis to separate the activity from the background traffic (e.g., different types of HTTP probing), sources in the event will necessarily be restricted to the honeynet because application-level dialog requires responses that the darknet cannot provide. In this case we can still perform an approximate test, by testing the volume of traffic seen concurrently in the darknet using the same port number. Doing so, may miss some hit-list (liveness) events, however, because we tend to overestimate the amount of activity the botnet exhibits in the darknet.

Other factors hardly cause an significant imbalance between the darknet and the Honeynet (a small θ), except the one in which an attacker chooses a small scan range that happens to include only the Honeynet addresses. However, even if this occurs we would also (if it does not reflect previous scanning, *i.e.*, is not a hit-list) expect it to occur equally often the other way around, *i.e.*, including only darknet addresses, which have not been observed over two years.

In the 203 events we analyzed, we find 33 (16.3%) hit-list (liveness) events. To our knowledge, this is the first empirical confirmation of the extensive use of hit-list (liveness) scanning.

C. Uniformity Checking

Question: Does an event uniformly scan the target range?

A natural technical for bots is to employ uniform random scanning across the target range. Testing whether the scans are evenly distributed in the honeynet sensor can be described as a distribution checking problem. We employ a simple χ^2 test, which is well-suited for the discrete nature of address blocks. For χ^2 test, when choosing the number of bins, a key requirement is to ensure that the expected value E_i for any bin should exceed 5 [15]. Accordingly, given that our events have at least several hundred scans in them, we divide the 2,560 addresses in our Honeynet into 40 bins with 64 addresses per bin. We then use the χ^2 test with a significance level of 0.5%, which work well in our evaluation in Section VI-B.

D. Dependency Checking

Question: Do the sources scan independently or are they coordinated?

Sophisticated scanning strategies can introduce correlations between the sources in order to control the work that each contributes more efficiently. In Section III-B, we describe a more efficient coordinated scheme ABPS (Advanced Botnet Permutation Scanning) based on permutation scanning will induce negative correlations in the targeting among the sources (they try to "get out of each other's way").

Since traditional approaches only work in linear dependence or two-variable cases, we develop a new hypothesis testing approach. To test for such coordination, we use the following hypothesis test. The null hypothesis is that the senders act in a uniform, independent fashion (where we first test for uniformity as discussed above); while the alternative hypothesis is that the senders do not act in an independent fashion. If an event comprises n scans targeting d destinations in a uniform random manner, we can in principle calculate the distribution of the number of destinations that receive exactly k scans, Z_k. We then reject the null hypothesis if the observed value is too unlikely given this distribution (we again use a 0.5% significance level).

Theorem 1: If n scans target d addresses in a uniform independent manner, the number of addresses Z_0 ($k = 0$) which do not receive any scan follows the probability distribution function:

$$P(z_0) = \binom{d}{z_0} \times \text{Stirling2}(n, d - z_0) \times (d - z_0)!/d^n$$

Proof: There are d^n total ways to distribute the n scans into d addresses. Among them, suppose X_0 ways exhibit z_0 addresses receiving zero scans (*i.e.*, z_0 empty slots). We then have the overall probability of observing z_0 empty slots to be $P(z_0) = X_0/d^n$.

We now show that for a given z_0, the following holds:

$$X_0 = \binom{d}{z_0} \times \text{Stirling2}(n, d - z_0) \times (d - z_0)! \qquad (1)$$

For d addresses, there are $\binom{d}{z_0}$ configurations from which to choose which z_0 addresses receive zero scans. Each such configuration has z_0 addresses with zero scans and $d - z_0$ addresses receiving a non-zero number of scans. Stirling2(n, m) denotes the number of ways of partitioning a set of n element into m nonempty sets [16]. Consider after partitioning the n scans into $d - z_0$ sets, we have $(d - z_0)!$ ways to map the sets to the addresses. Therefore, for each configuration we have Stirling2$(n, d - z_0) \times (d - z_0)!$ ways to distribute the n scans into $d - z_0$ addresses. This then establishes Eqn 1 ■

Note, we also validated this formula using Monte Carlo simulations with and without introduced correlations.

V. Extrapolating Global Properties

We now turn to the problem of estimating a botnet event's global scope (target size, participating scanners) based only on local information. This task is challenging because the size of the local sensor may be very small compared to the whole range scanned by a botnet, giving only a very limited view of the scanning event. For our estimation, we considered eight global properties, as shown in Table I.

For both uniform-random and uniform-hit-list (uniform-liveness) scanning, the uniformity property enables us to consider the local view as a random sample of the global view. Thus, the operating system (OS), autonomous system (AS), and IP prefix distributions observed in local measurements provide an estimate of the corresponding global distributions (bottom three rows). However, we need to consider that if bots exhibit

Property name	uniform scanning	uniform hit list	estimation method
Global target scope	Yes	Yes	indirect
Total # of bots	Yes	Yes	indirect
Total # of scans	Yes	Yes	indirect
Average scan speed per bot	Yes	Yes	indirect
Coverage hit ratio	Yes	No	direct
Sender OS distribution	Yes	Yes	direct
Sender AS distribution	Yes	Yes	direct
Sender IP prefix distribution	Yes	Yes	direct

TABLE I
GLOBAL PROPERTIES INFERRED FROM LOCAL OBSERVATIONS.

Approach	Properties	Affected by botnet dynamics	Require IPID or port # continuity
Both	# of bots	No	No
Approach I	Global target scope	No	Yes
	Total # of scans	No	Yes
	Average scan speed per bot	Yes	Yes
Approach II	Global target scope	Yes	No
	Total # of scans	Yes	No
	Average scan speed per bot	Yes	No

TABLE II
ADDITIONAL ASSUMPTIONS AND REQUIREMENTS.

heterogeneity in their scanning rates, then the probability of observing a bot decreases for slower-scanning ones. The scanning rate heterogeneity mentioned above introduces a bias towards the faster bots for these distributional properties. By extrapolating the total number of bots, however, we can roughly estimate the prevalence of this effect. It turns out that, in all of our analyzed events, by extrapolating the global bot population, we find that more than 70% of the bots that are globally involved in the scanning during the event duration appear at the local sensor.[2] Thus, the bias is relatively small.

The "coverage hit ratio" gives the percentage of target IPs scanned by the botnet. As this metric is difficult to estimate for hit-list (liveness) probing, we mainly consider uniform scanning, for which certain destinations are not reached due to statistical variations. For uniform scanning, we can directly estimate this metric based on the coverage in our local sensor.

In the remainder of this section we focus on the four remaining properties, each of which requires indirect extrapolation.

A. Assumptions and Requirements

To proceed with indirect extrapolation, we must make two key assumptions:

First, *the attacker is oblivious to our sensors and thus sends probes to them without discrimination.* This assumption is fundamental to general honeynet-based traffic study, (cf. the probe-response attack developed in [17] and counter-defenses [18]). A general discussion of the problem is beyond the scope of this paper. However, since we assume our technique is mainly used by a single enterprise or a set of collaborating enterprises, we need not release sensing information to the public, which counters the basic attack in [17]. Moreover, we can employ counter-defense techniques such as random shuffling [18] without influencing the extrapolation. With this assumption, we can treat the local view as providing unbiased samples of the global view.

Second, *each sender has the same global scan scope.* This should be true if all the senders are controlled by the same botmaster and each sender scans uniformly using the same set of instructions.

We argue that these two fundamental assumption likely apply to any local-to-global extrapolation scheme. In addition, we check for one general requirement before applying extrapolation, namely consistency with the presumption that *each sender evenly distributes its scans across the global scan scope*. This requirement is valid for the dark regions shown in Figure 4 (Section IV above), *i.e.*, both uniform random scanning and random permutation scanning, regardless of whether employing a hit-list. Therefore, prior to applying the extrapolation approaches, we test for consistency with uniformity (via methodology discussed in Section IV), which many of the botnet scan events pass (80.3%). Of course there is the usual "arms race" here between attackers and defenders. If our techniques become widely used, then attackers will modify their probing traffic to skew the defenders' analysis. We adopt the view common in network security research that there is significant utility in "raising the bar" for attackers even if a technique is ultimately evadable.

There are some additional requirements specific to certain extrapolation approaches, as listed in Table II. Botnet dynamics, such as churn or growth, can influence certain extrapolation approaches. Accordingly these approaches work better for short-lived events. Approach I, as discussed in section V-C, requires continuity of the IP fragment identifier (IPID) or ephemeral port, which holds for botnets dominated by Windows or MacOS machines (in our datasets we found all the events are dominated by Windows machines). We use passive OS fingerprinting to check whether we can assume that this property holds.

B. Estimating Global Population

Table III shows the notation we use in our problem formulation and analysis, marking estimates with "hat"s. For example, $\hat{\rho}$ represents the estimated local over global ratio, *i.e.*, ratio of local sensor size comparing to the global target scope of the botnet event, and $\hat{G}$ represents the estimated global target scope. In addition, we define M, the total number of bots that participate in the scanning during the time window T. M includes all scanning bots regardless of whether they are active during the entire time window T.

If ρ is small, many senders may not arrive at the sensor at all. In this case, we cannot measure M directly. Instead, we extrapolate the total number of bots using:

$$\frac{m_1}{M} = \frac{m_{12}}{m_2} \tag{2}$$

based on the following reasoning. We can split the address range of the sensor into two parts. Since the senders observed in each part are independent samples from the total population M, Equation 2 follows from independence. For example, suppose there are total $M = 400$ bots. In the first half sensor, we see $m_1 = 100$ bots, which is $1/4$ of the total bot population. Consider the second half as another independent sensor, so the bots it observes form another random sample from the total population. Then we have a $1/4$ chance to see if there is a bot already seen in the first half. If the second half observes $m_2 = 100$ bots too, the shared bots will be close to $m_{12} = 100/4 = 25$. Since in Equation 2 we can directly measure m_1, m_2, and m_{12}, we can solve for M, the total number of bots in the population. This is a variation of a general approach used to estimate animal populations known as *Mark and Recapture*. Since the m_1,m_2 and m_{12} are measured at exactly the same time window[3], the estimated total population

[2]The high percentage of bots appearing at the local sensor arises due to the fact that probing events continue long enough to expose majority of the bots.

[3]Mark and Recapture requires the "close" system assumption since the two visits do not happen in the same time, which is different here.

T	Event duration observed in the local sensor
d	Size of the local sensor
G	Size of global target scope
ρ	Local over global ratio d/G
M	Total # of senders in the global view in T
m	Total # of senders in the local view in T
m_1	# of senders in the first half of the local view in T
m_2	# of senders in the second half of the local view in T
m_{12}	# of overlapped senders of m_1 and m_2 in T
R	Average scanning speed per bot
R_{Gi}	Global scanning speed of bot i
T_i	Time between first and last scan arrival time from bot i
n_i	Number of local scans observed from bot i in T
Δt_j	Inter-arrival time between the j and $j+1$ scans
Q	Local total # of scans in T

TABLE III
TABLE OF NOTATIONS.

M is the number of bots of the botnet in the time window.

C. Exploiting IPID/Port Continuity

We now turn to estimating the global scan scope. We investigated two basic strategies: first, inferring the number of scans sent by sources in between observations of their probes at the Honeynet (**Approach I**); second, estimating the average bot global scanning speed using the minimal inter-arrival time we observe for each source (**Approach II**, covered in Section V-D).

Approach I is based on measuring changes between a source's probes in the IPID or ephemeral port number. We predicate use of this test on first applying passive OS fingerprinting to identify whether the sender exhibits continuous IPID and/or ephemeral port selection. This property turns out (see below) to hold for modern Windows and Mac systems, as well as Linux systems for ephemeral ports.

IPID continuity. Windows and MacOS systems set the 16-bit IPID field in the IP header from a single, global packet counter, which is incremented by 1 per packet. During scanning, if the machine is mainly idle, and if the 16-bit counter does not overflow, we can use the difference in IPID between two observed probes to measure how many additional (unseen by us) scans the sender sent in an interval. (The algorithm becomes a bit more complex because of the need to identify and correct IPID overflow/wrap, as discussed below. We also need to take into account the endianness of the IPID counter.)

A potential problem that arises with this approach is retransmission of TCP SYN's, which may increment the IPID counter even though they do not reflect new scans. For a given sender whose global target scope is G, let x be the percentage of live addresses that return SYN/ACK packets, and thus will usually not involve retransmission. Let k be the retransmission count determined by the sender's OS (*i.e.*, total number of attempts made before giving up). Ideally, we need to reduce the estimated global scan rate by a factor of $k \times (1-x) + x = k - (k-1) \times x$. We can observe k directly from the sender. x, however, is hard to estimate. Assuming that the probability of hitting a live IP address (x) is very low, we can approximate $x = 0$ for a first order estimation; therefore, we divide the global scan rate by k.

Ephemeral port number continuity. We have inspected the source code for five popular families of botnets. All of them let the operating system allocate the ephemeral source port associated with scanning probes. Again, these are usually allocated by sequentially incrementing a single, global, counter. As with IPID, we then use observed gaps in this header field to estimate the number of additional scans we did not see. (In this case, the logic for dealing with overflow/wrapping is slightly more complex, since different OSes confine the range used for ephemeral ports to different ranges. If we know the range from the fingerprinted OS, we use it directly; otherwise, we estimate it using the range observed locally, *i.e.*, the maximum port number observed minus the minimum port number observed.)

IPID and ephemeral port number continuity validation. In a controlled experimental environment, we installed five versions of Windows, one of MacOS X, and two versions of Linux, each in a different virtual machine. We then ran Nmap on each to generate scans, confirming that all but Linux (2.4/2.6) exhibit continuity of IPID (with Win98 and NT4 incrementing it little-endian, but Win2000, WinXP, Win2003, and MacOS X using network order) and that all 8 systems allocated the ephemeral ports sequentially.

For all the botnet events in the two-year Honeynet dataset, OS fingerprinting (via the `p0f` tool) indicates the large majority of bots run Windows 2000/XP/2003/Vista (85%), enabling us to apply both IPID and ephemeral port number based estimation. We also know that the proportion of Windows 95/98/NT4 is very low (0.8%), and only for those cases we need to switch the byte order. (These percentages match install-based statistics [19].)

NAT effects on IPID and ephemeral port continuity. Since NATs can potentially alter IPID and ephemeral ports, we test three popular home routers in this regard—Linksys, Netgear and D-Link, which comprise more than 70% of the home router market [20]. We use Nmap to send the scans from hosts behind these NATs and examine whether their IPID or ephemeral ports changed. For all three, IPID remains unchanged, and for a single scanner behind the NAT, the ephemeral port also remains unchanged. For multiple scanners behind the NAT, the ephemeral port numbers of the first sender remain unchanged, though for the D-Link router the ports of additional scanners become arbitrary.

Even though IPID remains unchanged, the intermingling of multiple IPID sequences for a single apparent source address renders extrapolation of scanning speed impractical. Techniques exist for detecting the presence of multiple sources behind a NAT (also based on IPID), but these require observing a large portion of the traffic from the NAT [21], which is impractical in our case. However, given that we usually have a large number of distinct sources, we can restrict our analysis to those cases that exhibit strong linearity for either IPID or ephemeral port numbers, which avoids conflating patterns in these arising from multiple sources aliased to the same public IP address. In our evaluation, we find that on an average 463 senders maintain linearity in IPID and/or ephemeral port numbers for an event; thus, they can be used for extrapolation purpose.

Global scan speed estimation. As the IPID and ephemeral port number approaches work similarly, here we discuss only the former. We proceed by identifying the top sources originating in at least four sets of scanning. We test whether (after overflow recovery) the IPIDs increases linearly with respect to time, as follows. First, for two consecutive scans, if the IPID of the second is smaller than the first, we adjust it by 64K. We then try to fit the corrected IPID$_i$ and its corresponding arrival time t_i, along with previous points, to a line. If they fit with correlation coefficient $r > 0.99$, it reflects consistency with a near-constant scan speed, and the sender is a single host rather than multiple hosts behind a NAT. When this happens, we estimate the global speed from the slope.

It is possible that multiple overflows might occur, in which

case the simple overflow recovery approach will fail. However, in this case the chance that we can still fit the IPIDs to a line is very small, so in general we will discard such cases. This will create a bias when estimating very large global scopes, because they will more often exhibit multiple overflows.

Sources that happen to engage in activity in addition to scanning can lead to overestimation of their global scan speed, since they will consume IPID or possibly ephemeral port numbers more quickly than those that might be simply due to the scanning. To offset this bias, when we have both IPID and ephemeral port estimates, we use the lesser of the two. Furthermore, in our evaluation, for the cases where we can get both estimates, we check the consistency between them, and found that IPID estimates usually produce larger results, but more than 95% of the time within a factor of two of the ephemeral port estimate. (Clearly, IPID can sometimes advance more quickly if the scanner receives a SYN-ACK in response to a probe, and thus returns an ACK to complete the 3-way handshake.)

Global scan scope extrapolation. With the ability to estimate the global scan speed, we finally estimate the global scan scope. Since we know the local scope, the problem is equivalent to estimate the local over global ratio ρ. Suppose in a botnet event there are m senders seen by the sensor, for which we can estimate the global scan speeds R_{Gi} of a subset of size m'. For sender i ($i \in [m']$), we know T_i (duration during which we observe the sender in the Honeynet) and n_i (number of observed scans). We use the linear regression as we discussed before to estimate the R_{Gi} which is also quite accurate. The main estimation error comes from variation of the observed n_i from its expectation. Define $\hat{\rho_i} = \frac{n_i}{R_{Gi} \cdot T_i}$ for each sender. Sender i's global scan speed is R_{Gi}. Globally during T_i, it sends out $R_{Gi} \cdot T_i$ scans. n_i is the number of scans we see if we sample from $R_{Gi} \cdot T_i$ total scans with probability ρ. Therefore, $\hat{\rho_i}$ is an estimator of ρ. If we aggregate over all the m' senders, we get

$$\hat{\rho} = \frac{\sum_i^{m'} n_i}{\sum_i^{m'} R_{Gi} \cdot T_i} \tag{3}$$

In the following Theorem 2 and Theorem 3, we prove that $\hat{\rho}$ provides an unbiased estimator of ρ and exhibits greater accuracy than $\hat{\rho_i}$, which is based on only one sender. In our approach, we use $\hat{\rho}$ to estimate the global scan scope that a botnet targeted.

Theorem 2: $\hat{\rho}$ is an unbiased estimator for ρ.

Proof:

$$E(\hat{\rho}) = E(\frac{\sum_i^{m'} n_i}{\sum_i^{m'} R_{Gi} \cdot T_i}) = \frac{E(\sum_i^{m'} n_i)}{\sum_i^{m'} R_{Gi} \cdot T_i} = \frac{\sum_i^{m'} E(n_i)}{\sum_i^{m'} R_{Gi} \cdot T_i}$$

As we mentioned, n_i is the number of scans we see if we sample from $R_{Gi} \cdot T_i$ total scans with probability ρ, which follows a binomial distribution. Hence we have $E(n_i) = \rho \cdot R_{Gi} \cdot T_i$. Therefore,

$$E(\hat{\rho}) = \frac{\sum_i^{m'} \rho \cdot R_{Gi} \cdot T_i}{\sum_i^{m'} R_{Gi} \cdot T_i} = \rho \cdot \frac{\sum_i^{m'} R_{Gi} \cdot T_i}{\sum_i^{m'} R_{Gi} \cdot T_i} = \rho$$

■

Theorem 3: $VAR(\hat{\rho}) = \frac{\rho \cdot (1-\rho)}{\sum_i^{m'} R_{Gi} \cdot T_i} < VAR(\hat{\rho_i})$, *i.e.*, the accuracy of ρ estimator when aggregating over all m' senders is higher than that of each and every single sender.

Proof:

$$VAR(\hat{\rho}) = VAR(\frac{\sum_i^{m'} n_i}{\sum_i^{m'} R_{Gi} \cdot T_i}) = \frac{\sum_i^{m'} VAR(n_i)}{(\sum_i^{m'} R_{Gi} \cdot T_i)^2}$$

Similar as before since n_i follows a binomial distribution, we have $VAR(n_i) = \rho \cdot (1-\rho) \cdot R_{Gi} \cdot T_i$. Therefore,

$$VAR(\hat{\rho}) = \frac{\sum_i^{m'} \rho \cdot (1-\rho) \cdot R_{Gi} \cdot T_i}{\left(\sum_i^{m'} R_{Gi} \cdot T_i\right)^2} = \frac{\rho \cdot (1-\rho)}{\sum_i^{m'} R_{Gi} \cdot T_i}$$

On the other hand,

$$VAR(\hat{\rho_i}) = VAR(\frac{n_i}{R_{Gi} \cdot T_i}) = \frac{VAR(n_i)}{(R_{Gi} \cdot T_i)^2} = \frac{\rho \cdot (1-\rho)}{R_{Gi} \cdot T_i}$$

Therefore, $VAR(\hat{\rho}) < VAR(\hat{\rho_i})$ ■

Average Scan Speed Per Bot. After extrapolating ρ and M, we estimate the average scan speed per bot using:

$$\frac{Q}{R \cdot T \cdot M} = \rho \tag{4}$$

Here Q is the number of scans received by the sensor in time T, which should reflect a portion ρ of the total scans. We estimate the total scans by $R \cdot T \cdot M$, where R is the average scan speed per bot. This formulation assumes that each bot participates in the entire duration of the event, which is more likely to hold for short-lived events.

Limitations. Note that the above techniques can fail if attackers either craft raw IP packets or explicitly bind the source port used for TCP probes. Thus, the schemes may lose power in the future. However, crafting raw IP packets and simulating a TCP stack is a somewhat time consuming process, especially given most bots (85+%) we observed run Windows, and in modern Windows systems the raw socket interface has been disabled. Empirically, in our datasets we did not find any case for which the techniques did not apply.

D. Extrapolating from Interarrival Times

For **Approach II**, we estimate global scanning speed (and hence global scope, via estimating ρ from an estimate of R using Equation 4) in a quite different fashion, as follows. Clearly, a sender's global scan speed s provides an upper bound on the local speed we might observe for the sender. Furthermore, if we happen to observe two consecutive scans from that sender, then they should arrive about $\Delta t = 1/s$ apart. Accordingly, the minimum observed Δt gives us a lower bound on s, but with two important considerations: *(i)* the lower bound might be too conservative, if the global scope is large, and we never observe two consecutive scans, and *(ii)* noise perturbing network timing will introduce potentially considerable inaccuracies in the assumption that the observed Δt matches the interarrival spacing present at the source.

We proceed by considering all m senders, other than those that sent only a single scan. We rank these by the estimated global scan rate they imply via $\hat{s} = 1/\hat{\Delta t}$, where $\hat{\Delta t}$ is the minimum observed interarrival time for the sender. Naturally, fast senders should tend to reflect larger estimated speeds, which we verified by comparing

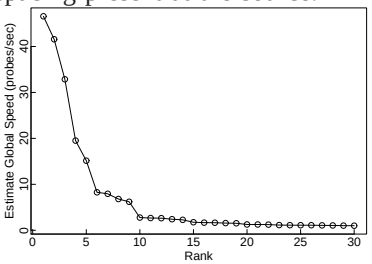

Fig. 6. Top 30 estimate speeds of Event VNC-060729.

Targeted Service	# of kinds of vul./probes	Events
NetBIOS/SMB/RPC	7	81
VNC	1	39
Symantec	1	34
MS SQL	1	14
HTTP	2	13
Telnet	1	12
MySQL	1	6
Others	4	4
total	18	203

TABLE IV
THE SUMMARY OF THE EVENTS.

$\hat{\Delta}t$ of each sender with how many scans we observed from it. We find that generally the correlation is clear though with considerable deviations.

Using the fast senders' speeds to form an estimate of the *average* scanning speed may of course overestimate the average speed. On the other hand, our technique aims at estimating a lower bound. Thus, it is crucial to find a balanced point among the possible estimates. We do so by presenting the different sorted estimates from which the analyst chooses the "knee" of the resulting curve, *i.e.*, the point with smallest rank k for which an increase in k yields little change in s. Figure 6 shows an example, plotting the top 30 maximum estimated speeds of Event VNC-060729. From the figure we would likely select $k = 6$ as the knee, giving an estimated speed 8.26.

VI. EVALUATION

We evaluate our techniques using the honeynet traffic described in Section II-A. The total data spans 24 months and 293 GB of packet traces. Since our extrapolation algorithms are linear algorithms, we find that our system takes less than one minute to analyze the scan properties and to perform the extrapolation analysis for a given event. We extract 203 botnet scan events and 504 misconfiguration events. There were a few moderate worm outbreaks observed during the period, such as the Allaple worm [22].

We first present characteristics of the botnet scanning events, followed by the botnet event correlation study. Next we discuss results for the four botnet scan pattern checking techniques and their validation. We finish with the presentation of global extrapolation results and their validation using DShield, a world-wide scan repository.

A. Basic Characteristics of Botnet Events

In Table IV, we break down 203 events according to their targeted services. We find that most of the events target popular services that have large install-base. We also find that 30 (14.8%) events are purely port reconnaissance without any payloads. Another three events check whether the HTTP service is open by requesting the homepage. The remaining (83.7%) events target certain vulnerabilities. Therefore, these botnet scans likely reflect attempted exploitations.

Figure 7 shows the CDF of event duration. A botnet event can last from a few minutes to a few days. There are 36 events that last very close to half an hour, leading to the spike in the Figure 7. Those events target a single SMB vulnerability and repeat daily. We find all those events share more than 35% of the same sources. We conjecture they stem from a single botnet, with the botmaster asking the botnet to repeatedly scan. In Figure 8, we show the CDF of unique number of ASes per event. Most of the bots (62.7%) come from more than 100 ASes. Only 3% of events reflect fewer than 20 ASes. This implies that cleaning the botnets from some part of the world (some of ASes) will not improve the situation. Also blocking them based on AS number is very hard due to large number of ASes involved.

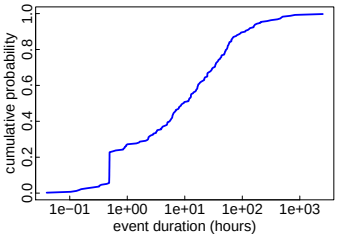

Fig. 7. Event Duration.

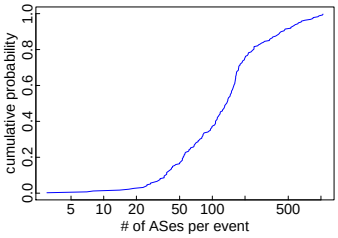

Fig. 8. # Source ASes.

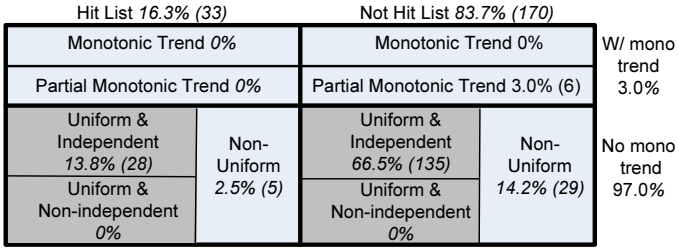

Fig. 9. Scan Pattern checking results.

B. Property-Checking Results

Figure 9 shows the breakdown of the events along different scanning dimensions. Six of the 203 events exhibit partial monotonic trends; 16.3% reflect hit-lists (liveness); 80.3% follow the random-uniform pattern, passing both uniformity and independence tests.

Through manual inspection of the partial monotonic events, we find that nearly half of the bots scan randomly and another half of bots scan sequentially. All of these bots start to scan at almost the same time. Perhaps they reflect two groups of bots controlled by the same botmaster, and the botmaster asking these two groups to use different scan strategies; but in general, this behavior is puzzling.

After that, we test the use of hit-list (liveness) scanning. As mentioned before, we use θ (the ratio of the number of senders in the darknet over to those of the honeynet) to classify the events. Out of the 106 events classified by port number, 33 reflect hit-list (liveness) scanning when using $\theta = 0.5$. In fact, all have empirical values for $\theta < 0.01$, and all of events with $\theta > 0.5$ have $\theta > 0.85$. The 97 other events use popular ports also seen in background radiation, and thus we have to classify them based on application-level behavior. For these, we conservatively assume that all the senders in the darknet using the same port number is possible members of the event, which tends to overestimate θ. For these 97 events, we did not find any with small θ and most of them have θ larger than one. We found in all the cases, the results are insensitive to the threshold of θ. In addition, none of the events only target the darknet.

date 2006	desc	ex. scope (I)(/8)	DShield scope (/8)	scope ratio (I)	ex. scope (II)(/8)
08-25	MSSQL	1.48	1	1.48	4.6
11-26	Symantec	0.59	0.75	0.79	0.1
11-27	Symantec	0.76	1	0.76	0.4
11-28	Symantec	0.92	1	0.92	4.0
07-23	VNC	0.63	0.9	0.7	0.9
07-29	VNC	0.63	0.87	0.72	0.9
10-31	VNC	0.80	0.80	1	0.6
08-24	NetBIOS	0.86	1	0.86	3.5
08-25	NetBIOS	1.13	1	1.13	2.5
08-29	NetBIOS	0.89	1	0.89	0.5
09-02	SMB	0.67	0.50	1.34	0.5
07-26	SMB	0.82	1	0.82	4.3

TABLE V
GLOBAL SCOPE EXTRAPOLATION RESULTS AND VALIDATION.

34 of the 197 random events fail the test for uniformity. We visually confirm that all of the remaining 163 events passing the test indeed appear uniform. Three of those that failed appear uniform visually, but have very large numbers of scans, for which the statistical testing becomes stringent in the presence of a minor amount of noise. In the remaining failed cases, we can see "hot-spot" addresses that clearly attract more activity than others; we do not know why.

Finally, we test the 163 uniform cases for coordination, not finding any instances at a 0.5% significance level. In addition, we simulate the advanced botnet permutation scan (ABPS) we proposed in Section III-B, and the dependency test accurately detects it.

C. Extrapolation Evaluation & Validation

We validate two forms of global extrapolation—global scan scope and total number of bots—using DShield [5], a very large repository of scanning and attack reports.

Finding: 75% of our estimates of global scanning scope using only local data lie within a factor of 1.35 of estimates from DShield's global data, and all within a factor of 1.5.

Finding: 64% of bot population estimates are within 8% of relative errors from DShield's global data, and all within 27% of relative errors

For 163 uniform events, 135 reflect independent uniform scanning and 28 reflect hit-list (liveness) scanning. For each type we estimate either the total scanning ranges or the total size of the hit lists, respectively. It is difficult to verify hit-list (liveness) extrapolations because of the difficulty of assessing how the hit-list will align with sources that report to DShield. However, we can validate extrapolations from the first class of events since we find they usually target a large address range. Due to limited data access to DShield, we have only been able to verify 12 cases as of today, as shown in Table V.

1) Global Scope Extrapolation and Validation: We present results from extrapolation, discuss our validation methodology, and apply the methodology to analyze our extrapolation accuracy.

Global scope extrapolation results: In Table V, we show the extrapolated scan scope we estimate from the local honeynet comparing with the estimation we make with the DShield data. Column *ex. scope (I)* shows the honeynet extrapolated scan scope by Approach I. Column *DShield scope* shows the DShield based estimation. Column *scope ratio* gives the ratio of the extrapolated scan scope by Approach I over the DShield scope. Column *ex. scope (II)* shows the extrapolated scan scope by Approach II. From the results, we see that our findings are consistent with those derived from DShield. Next, we introduce how the DShield validation works, and then we will analyze the accuracy of our results.

Validation Methodology: We find that most DShield sensors have synchronized clocks (*i.e.*, we often find significant temporal overlap between our honeynet events and corresponding DShield reports). For a given extraplation, we take two steps for validation.

Because our extrapolation results (Column *ex. scope (I)* in Table V) suggest that most global scanning scopes of botnet events are close to a /8 in extent, in the DShield validation we first analyze which /8's are involved, and then further infer actual scanning scopes within each related /8. We then sum up the scanning scopes in all of the /8's to produce the final validation result.

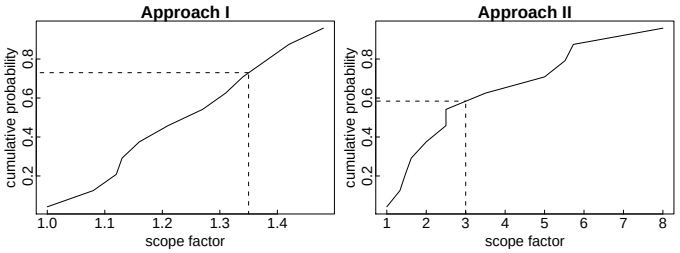

Fig. 10. Scope factors of the 12 events validated.

Step 1. Let X denote the /8 IP prefix of our sensor. We first calculate the number of shared senders $N(X)$ between our event data and scan logs for X from DShield. We consider additional /8 prefixes Y_i if their numbers of senders shared with the honeynet $N(Y_i)$ are larger than $N(X)/3$, reflecting an assumption that if a botnet uniformly scans multiple /8 prefixes, each should see quite a few sources in common. For X and each Y_i, we select the full width at half maximum (FWHM) of the unique source arrival process as a (conservative) way to delineate the global interval of the event. We then calculate the time range overlap with X for each Y_i; if the overlap of Y_i exceeds 50% of X's interval, we consider that the botnet scanned X and Y_i at the same time.

Step 2. After finding the scanned /8 networks, we estimate the scan scope within each. Alternatively, we compute the ratio of sensors in each network reporting the scans. There are several limitations of DShield data. First, it does not contain complete scan information (only a subset of scans within a prefix are reported). Second, different sensors might use different reporting thresholds and might not see all activity (*e.g.*, due to firewall filtering). Thus all these limitations makes calibration of data a challenging job.

To assess the limitations, we check a one-week interval around our events to find which DShield sensors *ever* report a given type of activity. We treat all the reporting sensors in one /24 network as a single unique sensor. We count the number of sensors from different /24 networks, denoted by C_{total}. Similarly, we count the number of unique sensors from different /24 networks that reported scans from shared senders of the given event, denoted C_{est}. We reduce the noise from the DShield data by removing sensors that only report a single address within a /24 sensor. We then use C_{est}/C_{total} to estimate the fraction of a /8 networks scanned by the botnet, which gives us a conservative estimate of the event's total range. We add up such fractions if there are multiple related /8 networks discovered in the first step, indicating the results in Column *DShield scope* of Table V.

Accuracy Analysis: We define the *scope factor* as $\max(D/H, H/D)$, where the D is the Dshield scope and H is the Honeynet scope. The scope factor indicates the absolute relative error in the log scale. The DShield data shows that our local estimates of global scope exhibit a promising level of accuracy. As shown in Figure 10, for Approach I, the scope factors of 75% events are less than 1.35, and all of them are less than 1.5. Approach II (column *ex. scope II*) works less well (58% of events are within a factor of three and 92% within a factor of six), but it may still exhibit enough power to enable sites to differentiate scans that specifically target them versus broader sweeps. In our two-year dataset, we did not find any scan events specifically targeting LBL, where the sensor resides. Moreover, it is less likely for a research institution such as LBL to be a target. We would presume that targeted attacks are more likely to occur at a site with high business interest,

such as financial corporations and well-known companies.

2) Targeted Attack Emulation and Detection: Since we did not discover any actual targeted attacks in the LBL dataset, we created synthetic attacks to evaluate the power of our approach. For each event shown in Table V, we emulated a synthetic targeted attack using the same characteristics of that real event (X). This includes the same number of bots as observed in X, including the same scan duration a observed in X. We randomly chose the scan speed for each bot and fixed the average scan speed of all bots to be similar to the extrapolated average scan speed of X. During this emulation, each bot scans uniformly randomly to the same targeted scope. We emulate those bots scanning one of LBL's /16 networks, along with the honeynet sensor collecting the data. The results show that our approach indeed correctly extrapolates the global scan scope to be one /16 network with less than 2% of error. This demonstrates that our approach can accurately detect targeted attacks. We also generate synthetic events targeting a /8 network, finding the global extrapolated accuracy similar in accuracy to the occurrence in real events, demonstrating the feasibility of such emulation. The smaller targeted scope, the more scans that our sensor will observe, and as a consequence the higher the resulting detection accuracy. This is the reason that the accuracy for /16 networks is much higher than for /8 networks.

3) Total Population Estimates and Validation: We assume that our honeynet event data and the corresponding DShield data give us two independent samples of the bot population, which is another chance to use the Mark and Recapture principle. We count the sources observed by DShield sensors of IP prefix X on the same port number in the same time window as the sources of DShield sensors. We term the number of sources in common between our honeynet and DShield as the *shared sources*. Based on the similar idea of Equation 2, we know the fraction of the shared sources to the sources of DShield should be equal to the ratio between bots observed in the honeynet and total population. Since DShield sensors will see other scanners (constituting noise) as well, we will likely underestimate the first fraction, and consequently overestimate the bot population. Per the results shown below, we find the estimates very close to those we estimate locally by splitting the sensor into two halves.

Table VI. shows the extrapolation and DShield validation results. Column *ex. #bots* shows our bot population extrapolation constructed by splitting the sensor into two halves. Column *#bot DShield* shows the results using DShield's global data. Column *#bots ratio* gives the ratio between the two of these. Note, we only validate the seven port number based events (MSSQL, Symantec and VNC). The NetBIOS/SMB events require payload analysis, which cannot validate through DShield since it does not provide any payloads. We find our approach is quite accurate given 64% of cases are within 8% of relative error ($|(our - DShield)|/DShield$).

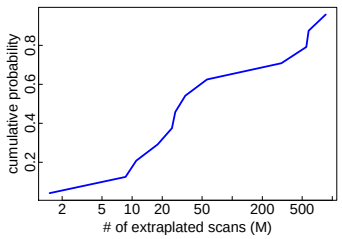

Fig. 11. Extrapolated # of scans.

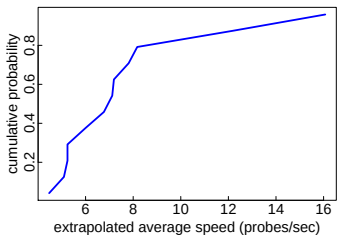

Fig. 12. Extrapolated the average scan speed.

date 2006	desc	ex. #bots	#bots DShield	#bots ratio
08-25	MSSQL	3100	3139	0.99
11-26	Symantec	228	215	1.06
11-27	Symantec	276	373	0.73
11-28	Symantec	305	331	0.92
07-23	VNC	2752	2712	1.01
07-29	VNC	3628	3696	0.98
10-31	VNC	526	622	0.84

TABLE VI
EXTRAPOLATED BOT POPULATION RESULTS AND VALIDATION.

4) Other Extrapolation Results: Based on Approach I, we can also infer the total number of scans and extrapolated average scan speed in each event. In Figure 11, we show the extrapolated total number of scans, using a log-scaled X axis. We can see the number of scans sent by the events could differ significantly given the duration and the number of bots differ. Figure 12 shows the extrapolated average scan speed of the bots, which we find to be quite low. We confirmed that scanning tools such as Nmap generate comparable TCP scanning rates. One underlying reasoning is as follows. On Windows platforms, the scan rate is limited by (i) the ephemeral port number range, and (ii) the waiting time before a closed connection tuple can be reused. Unless these OS parameters are changed, they necessarily bound the scanning rate. We analyzed botnet source codes (Section III) and did not find any botnet changes to these parameters. We might conjecture that—given the ease by which botmasters recruit new bots—improving the efficiency of single bots is of secondary concern. In addition, slow scanning rates are less likely to be detected.

VII. RELATED WORK

The work that most heavily influences us is the vision paper of Yegneswaran and colleagues on "Internet situational awareness" [9]. Their work outlines the general problem of analyzing honeynet traffic to assess its significance for the site observing it. The authors present the potential promise of such analysis using techniques that rely considerably on visualization. Along with [23], we aim to go substantially further, developing a "toolkit" for analyzing particular features of large-scale honeynet events, and devising techniques and a general framework to automatically or semi-automatically derive conclusions based on honeynet data.

In [24], Katti *et al.* propose novel approaches for evaluating the importance of collaboration among IDSes and show that indeed collaboration can improve detection speed and accuracy. Their paper studied collaboration regarding targets (different IDSes), while our study mainly focuses on the coordination of the sources (bots).

DShield is the Internet's largest global alert repository [5], a collaborative effort for detecting attackers. Our appproach does not rely on collaboration—an individual enterprise can by itself adopt our method to understand the significance of botnet probes. In the absence of collaboration, enterprises can keep their detection sensor information private, lessening concerns of pollution and detection avoidance [17]. Moreover, in our experience, DShield data is quite noisy due to non-uniform sensor density, which can hamper its use for inference.

While the state of the art in terms of building honeynet systems has advanced considerably, the analysis of large-scale events captured by such systems remains in its early stages. The Honeynet project has developed a set of tools for host-level honeypot analysis [25]. At the network level, Honeysnap [26] analyzes the contents of individual connections, particularly

for investigating IRC traffic used for botnet command-and-control. These approaches all either focus on single instances of activity, or on study of particular botnets over time (*e.g.*, [4]). In contrast, in this paper, we aim instead to understand the significance of single, large-scale events as seen by honeynets. Such activity by definition entails analysis integrated across a large number of instances of the activity, but also (unlike [4]) localized in time.

Furthermore, the literature includes a number of forensic case studies analyzing specific large-scale events, particularly worms [27], [28]. Such case studies have often benefited from *a priori* knowledge of the underlying mechanisms generating the traffic of interest. For our purposes, however, our goal is to infer the mechanisms themselves from a starting point of more limited knowledge.

VIII. Discussion

To fully use our approach, an enterprise needs to allocate an IP address block divided into a darknet and a honeynet. Enterprises that can only deploy darknets still gain most benefits, though without a honeynet they cannot detect hit-list (liveness) scanning, nor employ payload analysis to further classify the traffic using protocol/session semantics rather than simply port numbers.

Enterprises that lack unused address blocks can still partially take the advantage of our approach if they have blocks with known limited access. For example, if a block does not provide any web service, then the enterprise can use it to detect botnet events that scan port 80. (Indeed, the enterprise could even set up a partial honeynet operating on just that port.)

From our experiences, ten /24 networks worked well, and we would expect that fewer will too. This requirement should be well within the capability of a large enterprise.

IX. Conclusions

In this paper, we develop techniques for recognizing botnet scanning strategies and inferring the global properties of botnet events. An evaluation of our tools using extensive honeynet and DShield data demonstrates the promise our approach holds for contributing to a site's "situational awareness"—including the crucial question of whether a large probing event detected by the site simply reflects broader, indiscriminate activity; or instead reflects an attacker who has explicitly targeted the site.

X. Acknowledgments

Our thanks to Ruoming Pang and Vinod Yegneswaran for the development of the responders used by our honeynet, and the Lawrence Berkeley National Laboratory for support of the darknet and honeynet operations. This work was supported in part by NSF Awards 0433702 and 0905631, DoD Young Investigator Award FA9550-07-1-0074, and DoE Career award DE-FG02-05ER25692/A001. Opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the funding sources.

References

[1] "Adrian Lamo charged with computer crimes," http://www.securityfocus.com/news/6888.
[2] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of Internet background radiation," in *Proc. of ACM IMC*, 2004.
[3] K. Chiang and L. Lloyd, "A case study of the rustock rootkit and spam bot," in *Proc. of USENIX HotBots*, 2007.
[4] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proc. of ACM IMC*, 2006.
[5] SANS Institute, "Dshield.org: Distributed intrusion detection system," http://www.dshield.org/.
[6] N. Provos, "A virtual honeypot framework," in *Proc. of USENIX Security*, 2004.
[7] P. Bacher, T. Holz, M. Kotter, and G. Wicherski, "Know your Enemy: Tracking Botnets," http://www.honeynet.org/papers/bots.
[8] J. Kannan *et al.*, "Semi-automated discovery of application session structure," in *Proc. of ACM IMC*, 2006.
[9] V. Yegneswaran, P. Barford, and V. Paxson, "Using honeynets for internet situational awareness," in *In Proc. of ACM Hotnets IV*, 2005.
[10] Z. Li, A. Goyal, Y. Chen, and A. Kuzmanovic, "Measurement and diagnosis of address-misconfigured P2P traffic," in *Proc. of IEEE INFOCOM*, 2010.
[11] C. C. Zou *et al.*, "Monitoring and early warning for internet worms," in *Prof. of ACM CCS*, 2003.
[12] P. Barford and V. Yegneswaran, "An inside look at botnets," ser. In Series: Advances in Information Security. Springer, 2006.
[13] S. Staniford *et al.*, "How to 0wn the Internet in your spare time," in *Proc. of USENIX Security*, 2002.
[14] M. G. Kendall, *Rank Correlation Methods*. Griffin., 1976.
[15] J. A. Rice, *Mathematical Statistics and Data Analysis*. Duxbury Press, 1994.
[16] W. E. Weisstein, "Stirling Number of the Second Kind," http://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html.
[17] J. Bethencourt, J. Franklin, and M. Vernon, "Mapping internet sensors with probe response attacks," in *Proc. of the USENIX Security*, 2005.
[18] J. Cai, V. Yesneswaran, C. Alfeld, and P. Barfard, "Honeynets and honeygames: A game theoretic approach to defending network monitors," University of Wiscconsin, Tech. Rep. TR1577, 2006.
[19] "OS Platform Statistics by W3school," http://www.w3schools.com/browsers/browsers_stats.asp.
[20] "AP Market Sharing," http://news.com.com/Microsofts+Wi-Fi+ups+and+downs/2100-1039_3-994518.
[21] S. Bellovin *et al.*, "A technique for counting NATted hosts," in *Proc. of USENIX/ACM IMW*, 2002.
[22] "Net-Worm.Win32.Allaple.a," http://www.viruslist.com/en/viruses/encyclopedia?virusid=145521.
[23] Z. Li, A. Goyal, Y. Chen, and V. Paxson, "Automating analysis of large-scale botnet probing events," in *Proc. of ACM AsiaCCS*, 2009.
[24] S. Katti, B. Krishnamurhy, and D. Katabi, "Collaborating against common enemies." in *Proc. of ACM IMC*, 2005.
[25] "HoneyBow Sensor," http://honeybow.mwcollect.org.
[26] "Honeysnap," http://www.honeynet.org/tools/honeysnap/index.html.
[27] D. Moore *et al.*, "Inside the slammer worm," *IEEE Security and Privacy*, 2003.
[28] A. Kumar, V. Paxson, and N. Weaver, "Exploiting underlying structure for detailed reconstruction of an internet scale event," in *Proc. of ACM IMC*, 2005.

Zhichun Li is a Research Staff Member in NEC Laboratories America, Inc. He received his Ph.D. in Computer Science from Northwestern University in December 2009. His research focuses on network security, system security, network measurement and monitoring, and distributed system diagnosis.

Anup Goyal Anup Goyal is an engineer in Yahoo Search since October 2008. He has received his M.S. degree in 2008 from Computer Science and Engineering Department at Northwestern University and his B.Tech degree from IIT, Kharagur.

Yan Chen is an Associate Professor in the Department of Electrical Engineering and Computer Science at Northwestern University, Evanston, IL. He got his Ph.D. in Computer Science at the University of California at Berkeley in 2003. His research interests include network measurement, monitoring and security, and P2P systems. He won the DOE Early CAREER award in 2005 and the Microsoft Trustworthy Computing Awards in 2004 and 2005.

Vern Paxson is a Professor in the Electrical Engineering and Computer Sciences Department of the University of California, Berkeley, and a Senior Scientist at the International Computer Science Institute, also in Berkeley. His research focuses on real-time monitoring of network attacks, network measurement, botnet infiltration, and cybercrime profitability. He is an ACM Fellow and recipient of the ACM Grace Murray Hopper Award for his work on Internet measurement.