Automating Analysis of Large-Scale Botnet Probing Events

Zhichun Li, Anup Goyal and Yan Chen
Northwestern University
2145 Sheridan Road
Evanston, IL, USA
{lizc,ago210,ychen}@cs.northwestern.edu

Vern Paxson
UC Berkeley & ICSI
1947 Center St., Suite 600
Berkeley, CA, USA
vern@cs.berkeley.edu

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.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*network monitoring*; C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Algorithms, Measurement, Security

Keywords

Botnet, Global property extrapolation, Honeynet, Scan strategy inference, Situational awareness, Statistical inference

1. 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 part of the attacker. Indeed, such targeted attacks have recently grown in prominence. Yet given the incessant level of probing all Internet addresses receive [21], 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 the coordinated *botnets* that dominate today's Internet attack landscape. Our approach is limited to analyzing 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 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 [11], or network-level analysis of command-and-control (C&C) activity [24], 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 characterization of botnet properties based on inferences from their probing behavior. In addition, our approach has the significant benefit of requiring only *local* information, rather than global information as required by collaborative efforts such as DShield [27]. We give more detailed comparisons in Section 6.

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 one specific form of "hit-list" (Section 3). The type of hit-list we focus on is *liveness-aware scan-*

ASIACCS'09 March 10-12, 2009, Sydney, NSW, Australia

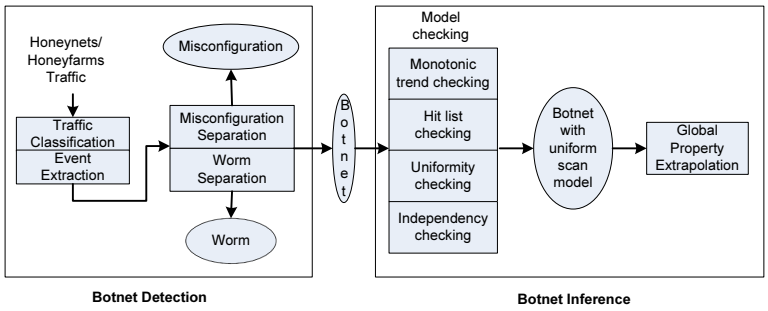

Figure 1: System architecture.

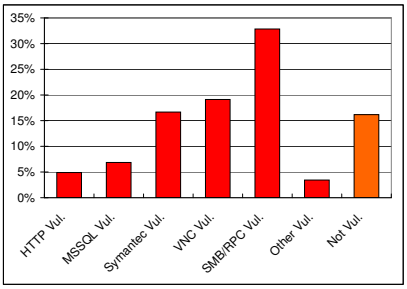

Figure 2: The distribution of the malicious payload discovered in the scan events.

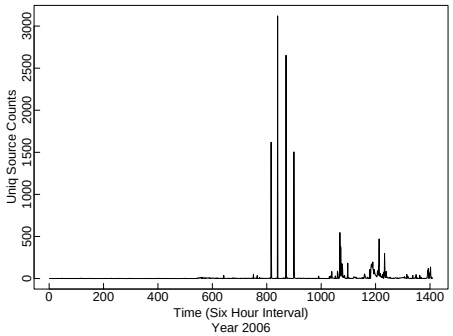

Figure 3: Temporal distribution of source count for VNC(5900).

ning, in which the attackers try to avoid darknets. For trend and uniformity checking, the statistical literature provides apt techniques, but assessing coordination and use of hit-lists requires developing 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 reveals some interesting and sophisticated botnet scan behaviors such as coordinated 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 differing 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 4). The global scanning scope enables the site's operators to assess whether their network is a specific target of botnet activity, or if instead the botnet's scanning targets a large network scope that simply happens to include the site. The estimated total botnet size can help us track trends in how botnets are used, with implications for their C&C capabilities.

The algorithms are rooted in the observation (confirmed by our checking of scanning properties) that the most frequent scanning patterns reflect uniform random scanning or uniform hit-list scanning. Indeed, nearly all of the probing events we observed follow one of these two scan patterns.[1] In Section 5, we evaluate our techniques using 24-month trace (293 GB total) of Honeynet traffic collected at a large research institution. Of the events classified as likely botnet activity (*i.e.*, not misconfigurations or worms), most reflected either uniform-random or uniform-hitlist scanning. Analyzing the data, we find that 66.5% of botnet events exhibit uniform random scanning and 16.3% of botnet events reflect hit-list scanning, 85% of which were also uniform.

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.*[2] We note that such botnet scans are one key technique employed for botnet recruitment [24]. Through event correlation study, we also find some interesting behaviors of how botmasters control their bots.

[1]Of course there is the usual "arms race" here between attackers and defenders. If our techniques become widely used, then attackers may modify their probing traffic to skew the defenders' analysis. But until the botmasters take steps to do so, these techniques have value. 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.

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

To validate our estimates of the global properties, we compare our results with those from DShield [27], 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 indicate that our approaches hold promise for sufficient accuracy to enable sites to make reliable inferences, with the caveat that we were unable to find any instances of events in our current dataset that reflected a global scope much different from /8.

2. 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 analysis system are automated, most of them fully so. We mainly use the Honeynet sensor to drive the rest of the discussion, although we can generally apply our analysis techniques (the botnet inference subsystem) to botnet probe events detected by other types of sensors. The system classifies traffic seen on the sensors by different protocols or by 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. The system extracts events based on the number of unique sources arriving in a window of time (*cf.* the spikes in Figure 3), classifying the activity into misconfigurations, worms, and botnet-like probing.

2.1 Honeynet and Data Collection

Our detection sensor consists of ten contiguous /24 subnets within one of a large research institution's /16 networks. We deployed Honeyd responders [23] on five of the subnets and operated the other five completely "dark". (We use this latter for hit-list detection.) The Honeyd configuration is similar to that used by Pang *et al.*in [21]: 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).

2.2 Botnet Detection Subsystem

In this paper we mainly focus on botnet inference. For the completeness we briefly introduce how to detect botnet events here. The details is available in our technique report [18].

Traffic Classification: Attack traffic can have complex session structure involving multiple application protocols. For example, an attacker can send an exploit to TCP port 139 which, if successful, results in opening a shell and issuing an HTTP download command. Often the application protocol contacted first is the protocol being exploited (an exception is an initial connection to a portmapper service), so we label sessions with the service associated with the first destination port appearing in them. Doing so also provides consistent labeling for connection attempts seen in darknets or other types of sensors. We aggregate connections into sessions using an approach similar to the first step algorithm by Kannan et al [14].

For application protocols not commonly used, the background radiation noise (including individual port scans) is typically low, and thus we use port numbers to separate event traffic. However, noise is usually strong for popular protocols, requiring further differentiation based on payload (when available). To do so, we implemented payload summary scripts for 20 commonly seen protocols, based on the Bro system's network analysis capabilities [22].

Event Extraction: Figure 3 shows source arrival counts for VNC (TCP port 5900) for the year 2006 on our sensor, where each point represents the number of sources within a six-hour interval. Large spikes in such plots generally correspond to scanning from worms or apparent botnets, or misconfigurations. We classify such spikes as *events*, as follows. We define the noise strength N as the per-interval count of unique sources seen in the absence of events. Suppose the time interval length is I. We calculate N as the median of unique source counts of K continuous time intervals *before* the event. We define signal strength $S = X - N$ as the peak unique source count arrival X minus the noise strength N, and define the signal-to-noise ratio as *SNR*$= \frac{S}{N} = \frac{X-N}{N} = \frac{X}{N} - 1$.

In our evaluation we use $I = 6$ hours and $K = 120$. The aggregated time window $I \times K$ is about 30 days. We only examine events with *SNR*≥ 50. We automatically extract potential events as follows: for any given time interval, we calculate the median of the previous normal K intervals and the *SNR*. For those spikes exceeding our *SNR* threshold, we extend the time range to both sides until $S \leq \omega N$ where ω is a tunable parameter controlling the amount of the signal tail to include in the event. (We use $\omega = 5$, though we find ranging it over $3 \ldots 8$ does not significantly alter the results.) For multiple events within one time series, we extract the events iteratively, starting with the event with largest *SNR*.

One problem we have to consider is that some events have complex session structures involving multiple protocols. After traffic classification by protocol information, a single event can be separated to multiple events. Therefore, after event classification, we need to merge them. We detect such cases by checking the connection correlation. If two connections are in one session, they will be both from host A to host B and the protocols of the two connections are fixed. For example, suppose the first connection is HTTP and the second one is WINRPC. If we find such events to be highly correlated, *i.e.*, for most connections in the HTTP event, each HTTP connection is followed by a WINRPC connection from the WINRPC event for the same source and destination pair, we merge them as one event.

Event Classification and Separation: We separate misconfigurations from worms or botnets based on the observation that botnet scans and worms should contact a significant range of the IP addresses, whereas misconfigurations exhibit a few hot-spot targets. We found that most misconfiguration events are due to P2P traffic. The detailed analysis of these misconfiguration is our technical report [17, 18].

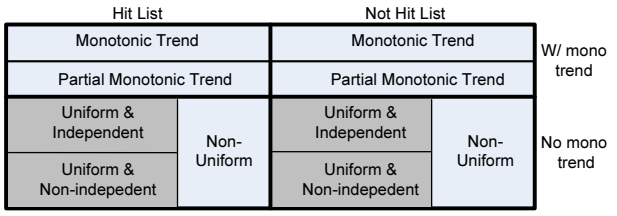

Figure 4: Model Checking Design Space.

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 [31]) and deem the remainder as botnet probing events.

2.3 Botnet Inference Subsystem

Scan Pattern Checking: For botnet probing events, 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:

- Monotonic trend checking
- Hit list checking
- Uniformity checking
- Dependency checking

For details, see Section 3.

Global Property Extrapolation: Once we identify a probing event's scan pattern, we then use the scan pattern to extrapolate a global view of the event. We focus on two of the most common scan patterns: uniform random scanning, and uniform hit-list scanning. We confirm their common use both from botnet source code analysis (Section A) and experimental observations (Section 5). We then extrapolate the global scan scope and the global number of bots based on these two scan patterns, using techniques developed in Section 4.

3. PROPERTY CHECKING OF BOTNET SCAN PATTERNS

The whole design space of the botnet probing strategies is very large. It is hard to consider all of them in our botnet inference framework. Through botnet source code analysis and reasoning what a rationale botnet master will do (the details is in Appendix A), We find the uniform random scanning, hit-list scanning, monotonic scanning and coordinated permutation scanning are the strategies more likely used by the botmasters, given they are simple and effective.

In this section we develop a set of analysis algorithms for detecting these scan strategies. 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.

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.

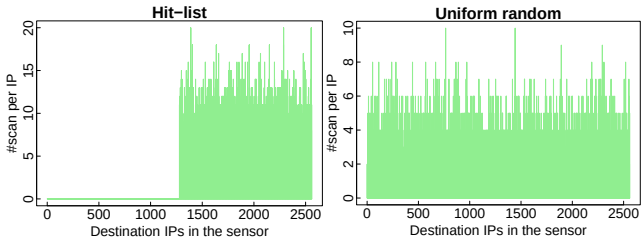

Figure 5: Hit-list and uniform scanning distribution on the sensor.

3.1 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 common scan strategy widely used by network scanning tools. In our evaluation, we did find a few events which use the monotonic trend scanning. Furthermore, for random events, the monotonic trend checking can help us 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 [15], 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. In our technical report [18], we describe the detailed approach and our enhancement to the original Mann-Kendall trend test.

We label an entire event as having a *monotonic trend* if more than 80% of senders exhibit a trend, and for further analysis remove those that do not reflect a trend as likely representing separate activity (and thus likely removing a source of potential noise). 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*.

3.2 Hit-List Checking

Question: Do the bots use a target hit-list for scanning?

By hit-list 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. When that is the case, most likely they will be re-scanned by the attacker 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 in the Honeynet portion, but not in the darknet portion, this provides strong evidence that the scan used a hit list. However, one consideration is event "pollution" (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 example hit-list 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. In our evaluation, we find 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 events, however, because we tend to overestimate the amount of activity the botnet exhibits in the darknet.

Even other factors could potentially cause an imbalance between the darknet and the Honeynet. However, most of these do not result in a significantly 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 but not Honeynet addresses, which have not been observed over two years.

In the 203 events we analyzed, we find 33 (16.3%) hit-list events.

3.3 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 for the test, a key requirement is to ensure that the expected value E_i for any bin should exceed 5 [26]. 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 is found to work well in our subsequent evaluation in Section 5.3.

3.4 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. For example, In Appendix A.2, 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 approach only an 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:*

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

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 1: Global properties estimated from local observations.

Approach	Property name	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 2: Additional assumptions and requirements.

The $Stirling2(n, y)$ denotes the Stirling number of the second kind [29], which is the number of ways to partition n elements to y non-empty sets. The proof is in Appendix B.

However, if $n \gg d$, then the sensor range will be sparsely populated, and this distribution does not give us much statistical power. Instead, we need to use a larger value of k. The more detailed analysis is in our technique report version [18].

We validate our tests using Monte Carlo simulations with and without introduced correlations. We also confirm that the test correctly detects the correlations introduced by our ABPS scheme. Finally, when applying our test to our two years' worth of data, we do not in fact find any cases exhibiting likely coordinated scanning.

4. EXTRAPOLATING GLOBAL PROPERTIES

We now turn to the problem of estimating a probing 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 1.

For both uniform-random and uniform-hit-list 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 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 in the population 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, we find that more than 70% of the bots appear at the local sensor[3] by comparing the number of bots seen at the local sensors with the extrapolated global bot population as shown in Table 6. Thus, the bias is relatively small.

The "coverage hit ratio" gives the percentage of target IP addresses scanned by the botnet. As this metric is difficult to estimate for hit-list 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 seen in our local sensor.

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

4.1 Assumptions and Requirements

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

1 *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 [9] and counter-defenses [10]). 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 [9]. With this assumption, we can treat the local view as providing unbiased samples of the global view.

2 *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 3 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 3), which many of the botnet scan events pass (80.3%).

There are some additional requirements specific to certain extrapolation approaches, as listed in Table 2. 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 4.3, 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.

4.2 Estimating Global Population

Table 3 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.

If ρ is small, many senders may not arrive at the sensor at all. In this case, we cannot measure the total bot population directly. Instead, we extrapolate the total number of bots as follows. With the uniform scan assumption discussed above, we have:

$$\frac{m_1}{M} = \frac{m_{12}}{m_2} \quad (1)$$

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 1 follows from independence. For example, suppose there are total

[3]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.

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 3: Table of notations.

$M = 400$ 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 1 we can directly measure m_1, m_2, and m_{12}, we can therefore solve for M, the total number of bots in the population. This is a simple 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[4], the estimated total population M is the number of bots of the botnet in the time window.

4.3 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 the next section).

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 counter as present in the IP header.)

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. Thus, when estimating global scan speed we divide by the average TCP SYN retransmission rate we observe for the sender.

Ephemeral port number continuity. All of the botnets for which we could inspect source code 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 operating systems 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.)

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

Operating System	Clients	
Windows	159,152	(85.2%)
Windows 2000/XP	155,869	(97.9%)
Windows 2003/Vista	231	(<.1%)
Windows NT4	1708	(1.07%)
Windows 98	1237	(0.7%)
Windows 95	68	(<.01%)
Windows other	39	(<.01%)
BSD	458	(0.2%)
Linux	126	(<.1%)
Novell	20	(<.01%)
Unidentified	27,047	(14.4%)
Total	186,725	

Table 4: Aggregate operating system distribution, from passive OS fingerprinting of probing events.

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.

As shown in Table 4, for all the probing events in the two-year Honeynet dataset, OS fingerprinting (via the `p0f` tool) indicates that the large majority of bots run Windows 2000/XP/2003/Vista (85%), enabling us to apply both IPID and ephemeral port number based estimation. From this analysis, we also know that the proportion of Windows 95/98/NT4 is very low (0.8%), and only for those cases do we need switch the byte order. (These percentages match install-based statistics [5] indicating that Win98 and NT4 comprise less than 1.5% of systems overall.)

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 [1]. 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 simple 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 coming out of the NAT [8], 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 for-

mer. 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 probing 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 with correlation coefficient $r > 0.99$ (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

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

As show in Appendix C, we formally prove that $\hat{\rho}$ is an unbiased estimator of ρ, and it is more accurate than $\hat{\rho}_i$, which only reflects a single sender. We then can use $\hat{\rho}$ to estimate the global scope a probe targeted.

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

$$\frac{Q}{R \cdot T \cdot M} = \rho \tag{3}$$

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 both of 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 appear to apply.

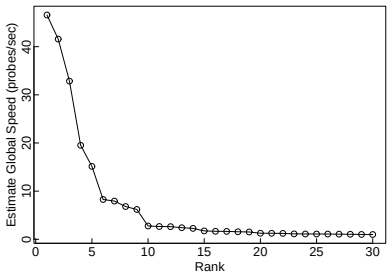

Figure 6: Top 30 estimate speeds of Event VNC-060729.

4.4 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 3) 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 we observe, 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 $\hat{\Delta}t$ 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.

5. EVALUATION

We evaluate our techniques using the honeynet traffic described in Section 2.1. The total data spans 24 months and 293 GB of packet traces. Since the extrapolation algorithms we use are linear in the number of scans in the events, we find that our system takes less than one minute to analyze the scan properties and perform the extrapolation analysis for a given event. We use *SNR*$= 50$ and a tail parameter $\omega = 5$ for event extraction (ranging ω from 3 to 8 yields identical results). 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 [4].

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 5: The summary of the events

The misconfiguration events are mainly caused by P2P traffic. In this paper, we focus on the botnet scan events.

We first present characteristics of the botnet scanning events. Then we present 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.

5.1 Basic Characteristics of the Botnet Events

In Table 5, 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. As we will discuss in Section 5.2, it is a cluster of events which scan the same vulnerability every half hour over and over again, for days on end. Most likely these botnet events are driven by a single botmaster. From Figure 8, we also find that the number of sources involved in a botnet event is quite heterogeneous. In Figure 9, 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. We also find that the number of destinations a bot scans differs significantly for different events, as show in Figure 10.

We further study the OS, AS and IP distribution of the events. Table 4 in Section 4 shows the aggregated OS distribution. We see that Microsoft Windows is the most popular OS, with more than 83% of bots using Windows 2000/XP. (We see similar results when analyzing individual events.) For AS and IP address distribution, we find that the aggregated results (203 events together) are close to those seen in previous work [25]. However, we find very large variation across individual events; thus, address blacklists derived from one event might not be effective when defending against other events.

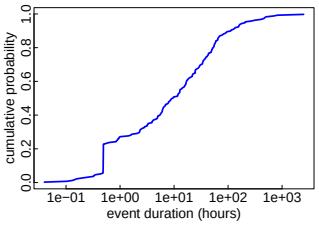

Figure 7: Event Duration.

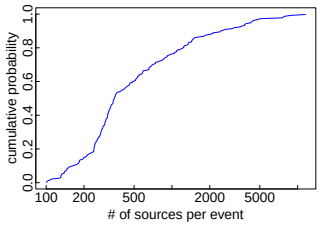

Figure 8: # of Sources.

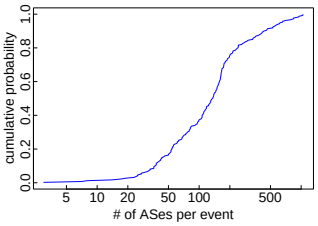

Figure 9: # Source ASes.

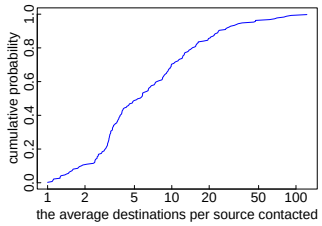

Figure 10: Avg. # Destinations / Source.

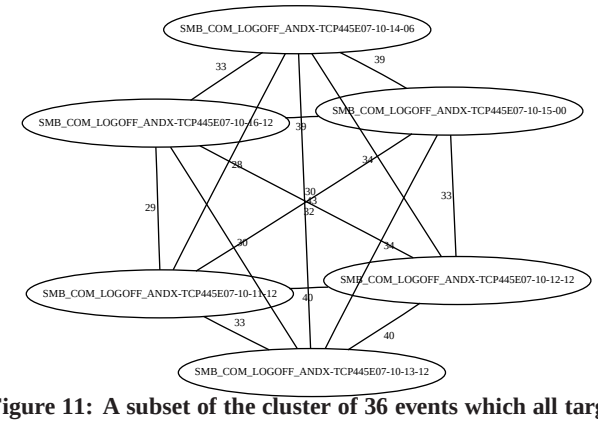

Figure 11: A subset of the cluster of 36 events which all target a same vulnerablity in SMB. The number on an edge labels the percentage of bots sharing.

5.2 Event Correlation

We study the temporal and source (bot IP address) correlation of different events. In this context, if we find two events that have more than 20% source addresses in common, we consider them as correlated. We calculate the percentage of sharing as the maximum of the shared addresses over total addresses of two events. We observe two types of interesting behavior:

Behavior 1: The botmasters ask the same botnet to scan the same vulnerability repeatedly. In our two years of data, we find several event clusters that exhibit this behavior. For example, there is a cluster of 36 events that occur every day, always scanning the same SMB vulnerability. These events form a nearly complete clique, *i.e.*, each event shares $\geq$ 20% of the same source addresses in common with most of the other events. In Figure 11, we show a subset of this commonality graph. These events on average share about 35% of the same sources. Each event occurs on a different day. We speculate this activity reflects the botmaster commanding the same botnet to re-scan the same address range repeatedly.

Behavior 2: The botmasters appear to ask most of the bots in a botnet to focus on one vulnerability, while choosing a small subset of the bots to test another vulnerability. Apart from these big clusters, we find there are some cases in which two events has very high correlation (more than 80% of source address commonality), and occur very close in time, usually the same day. We find that often the first event is much larger in terms of the number of bots than the second; the second is just a small subset of the bots from the first. This behavior illustrates that the difficulty of fingerprinting botnet activity, given that botmasters may select a subset of bots to assign to different tasks.

5.3 Property-Checking Results

Figure 12 shows the breakdown of the events along different scanning dimensions. Six of the 203 events exhibit partial monotonic trends; 16.3% reflect hit-lists; 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

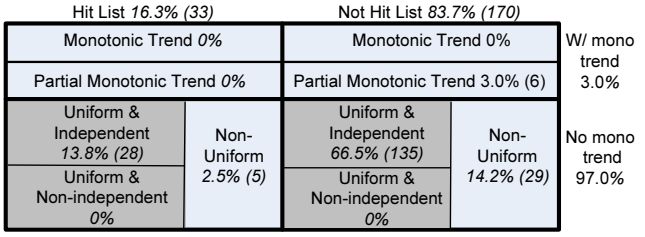

Figure 12: Scan Pattern checking results.

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 liveness-aware scanning (which we term "hit-lists"). As mentioned above, we use θ (the ratio of the number of senders in the darknet over to those of the live honeynet) as the metric to classify the events. Out of the 106 events classified by port number, 34 reflect hit-list 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 6: Global scope extrapolation results and validation (ex. denotes extrapolated; DShield denotes the validation results using DShield data.).

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), and find the dependency test can accurately detect it even with $0\% \sim 20\%$ packet loss. Thus, none of the scanning we observe appears to reflect any significant degree of coordination.

5.4 Extrapolation Evaluation and Validation

We validate two forms of global extrapolation—global scan scope and total number of bots—using data from DShield [27], 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 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 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 6.

5.4.1 Global Scope Extrapolation and Validation.

Global scope extrapolation results: In Table 6, 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 honeynet 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 appear to 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. First, since the extrapolation results we got are all of /8 size or quite close, we try to find all the /8 networks (except those with private IP prefixes) with sufficient source overlap with the honeynet events. Secondly, for these /8 networks, we infer the scan scopes and compare them with our results.

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

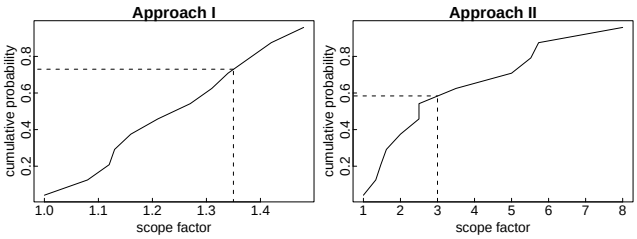

Figure 13: The CDFs of the scope factors of the 12 events we validate.

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 6.

Accuracy Analysis: We define the *scope factor* as

$$\text{scope factor} = \max\left(\frac{\text{DShield scope}}{\text{Honeynet scope}}, \frac{\text{Honeynet scope}}{\text{DShield scope}}\right)$$

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 13, we can clearly know that, 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 the research institution where the sensor resides; this fits with the institute's threat model, which is mainly framed in terms of indiscriminant attacks.

5.4.2 Total Population Estimate and Validation

We assume that our honeynet event data and the corresponding DShield scan 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 1, 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 7. 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 ($|(our - DShield)|/DShield$).

5.4.3 Other Extrapolation Results

Based on Approach I, we can also infer the total number of scans and extrapolated average scan speed of the bots in each event. In

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 7: extrapolated bot population results and validation.

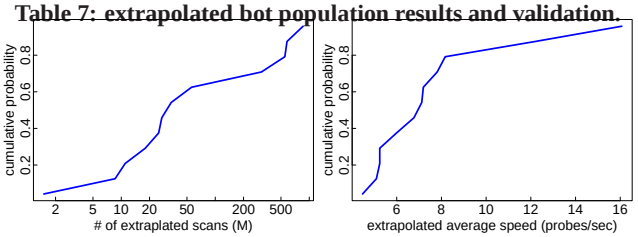

Figure 14: Extrapolated # of scans.

Figure 15: Extrapolated the average scan speed.

Figure 14, 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 in each event differ. In Figure 15, we show the extrapolated average scan speed of the bots.

6. RELATED WORK

The work that most heavily influences us is the vision paper of Yegneswaran and colleagues on "Internet situational awareness" [30]. 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. In this work, 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.

DShield is the Internet's largest global alert repository [27]. The advantages of our approach comparing with DShield are as follow: (i) In our experience, DShield data is quite noisy, and the sensor density quite non-uniform. These lead to cases where it is difficult to develop sound inferences from the data. (ii) DShield is subject to pollution and avoidance [9]. Depending solely on DShield might not be reliable for operational security. (iii) When the target scope is small, it is hard to find other sensors in DShield which share the same behavior; thus DShield will fail to work in such cases.

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 [2]. At the network level, Honeysnap [3] 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.*, [24]). 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 [24]) localized in time.

Furthermore, the literature includes a number of forensic case studies analyzing specific large-scale events, particularly worms [16, 20]. 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.

Finally, Gu *et al.*propose a series botnet detection techniques based on behavior correlation [12, 13]. In contrast, we focus on inferring botnet properties in the wake of detection, rather than detection itself.

7. CONCLUSIONS

In this paper we present several algorithms that can automatically analyze and determine the features of large-scale events that give insight into their underlying nature observed at a honeynet. In particular, we develop techniques for recognizing botnet scanning strategies and inferring a distributed scan's global properties. 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.

8. ACKNOWLEDGMENT

We would like to thank Vinod Yegneswaran and Ruoming Pang for helping collect the data and implementing the Bro payload summary scripts, the operations staff of the Lawrence Berkeley National Laboratory for facilitating the LBNL honeypot setup, and anonymous reviewers for their valuable comments. This work was supported by DOE CAREER award DE-FG02-05ER25692//A001, DOD (Air Force of Scientific Research) Young Investigator Award FA9550-07-1-0074, and NSF grants NSF-0433702 and CNS-0627320. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding sources.

9. REFERENCES

[1] AP Market Sharing. `http://news.com.com/Microsofts+Wi-Fi+ups+and+downs/2100-1039_3-994518`.

[2] HoneyBow Sensor. `http://honeybow.mwcollect.org`.

[3] Honeysnap. `http://www.honeynet.org/tools/honeysnap/index.html`.

[4] Net-Worm.Win32.Allaple.a. `http://www.viruslist.com/en/viruses/encyclopedia?virusid=145521`.

[5] OS Platform Statistics by W3school. `http://www.w3schools.com/browsers/browsers_stats.asp`.

[6] BACHER, P., HOLZ, T., KOTTER, M., AND WICHERSKI, G. Know your Enemy: Tracking Botnets. `http://www.honeynet.org/papers/bots`.

[7] BARFORD, P., ET AL. An inside look at botnets. In Series: Advances in Information Security. Springer, 2006.

[8] BELLOVIN, S., ET AL. A technique for counting NATted hosts. In *Proc. of USENIX/ACM IMW* (2002).

[9] BETHENCOURT, J., ET AL. Mapping internet sensors with probe response attacks. In *Proc. of the USENIX Security* (2005).

[10] CAI, J., ET AL. Honeynets and honeygames: A game theoretic approach to defending network monitors. Tech. Rep. TR1577, University of Wiscconsin, 2006.

[11] CHIANG, K., AND LLOYD, L. A case study of the rustock rootkit and spam bot. In *Proc. of USENIX HotBots* (2007).

[12] GU, G., PORRAS, P., YEGNESWARAN, V., FONG, M., AND LEE, W. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Proc. of USENIX Security* (2007).

[13] GU, G., ZHANG, J., AND LEE, W. Botsniffer: Detecting botnet command and control channels in network traffic. In *Proc. of NDSS* (2008).

[14] KANNAN, J., JUNG, J., PAXSON, V., AND KOKSAL, C. Semi-automated discovery of application session structure. In *Proc. of ACM IMC* (2006).

[15] KENDALL, M. G. *Rank Correlation Methods*. Griffin., 1976.

[16] KUMAR, A., PAXSON, V., AND WEAVER, N. Exploiting underlying structure for detailed reconstruction of an internet scale event. In *Proc. of ACM IMC* (2005).

[17] LI, Z., GOYAL, A., CHEN, Y., AND KUZMANOVIC, A. P2p doctor: Measurement and diagnosis of misconfigured peer-to-peer traffic. Tech. Rep. NWU-EECS-07-06, Northwestern University, 2007.

[18] LI, Z., GOYAL, A., CHEN, Y., AND PAXSON, V. Towards situational awareness of large-scale botnet events using honeynets. Tech. Rep. NWU-EECS-08-08, Northwestern University, 2008.

[19] MANNA, P., CHEN, S., AND RANKA, S. Exact modeling of propagation for permutation-scanning worms. In *IEEE INFOCOM* (2008).

[20] MOORE, D., PAXSON, V., SAVAGE, S., SHANNON, C., STANFORD, S., AND WEAVER, N. Inside the slammer worm. *IEEE Security and Privacy* (2003).

[21] PANG, R., YEGNESWARAN, V., BARFORD, P., PAXSON, V., AND PETERSON, L. Characteristics of Internet background radiation. In *Proc. of ACM IMC* (2004).

[22] PAXSON, V. Bro: A system for detecting network intruders in real-time. *Computer Networks 31* (1999).

[23] PROVOS, N. A virtual honeypot framework. In *Proc. of USENIX Security* (2004).

[24] RAJAB, M., ZARFOSS, J., MONROSE, F., AND TERZIS, A. A multifaceted approach to understanding the botnet phenomenon. In *Proc. of ACM IMC* (2006).

[25] RAMACHANDRAN, A., AND FEAMSTER, N. Understanding the network-level behavior of spammers. In *Proceedings of ACM SIGCOMM '06* (September 2006).

[26] RICE, J. A. *Mathematical Statistics and Data Analysis*. Duxbury Press, 1994.

[27] SANS INSTITUTE. Dshield.org: Distributed intrusion detection system. `http://www.dshield.org/`.

[28] STANIFORD, S., PAXSON, V., AND WEAVER, N. How to 0wn the Internet in your spare time. In *Proc. of USENIX Security* (2002).

[29] WEISSTEIN, W. E. Stirling Number of the Second Kind. `http://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html`.

[30] YEGNESWARAN, V., BARFORD, P., AND PAXSON, V. Using honeynets for internet situational awareness. In *In Proc. of ACM Hotnets IV* (2005).

[31] ZOU, C., GAO, L., GONG, W., AND TOWSLEY, D. Monitoring and early warning for internet worms. In *Prof. of ACM CCS* (2003).

APPENDIX

A. MODELING HOW BOTS SCAN

A.1 Bot Source Code Study

By analyzing the source code of five popular families of bots, we study different dimensions of scan strategies employed by botnets. The popularity of these five bot families is confirmed in [6, 7]. Our findings confirm those in [7], but we more focus on scan pattern study.

Botnet name	Agobot	Phatbot	Spybot	SDBot	rxBot
Global	Yes	Yes	Yes	Yes	Yes
Local	Yes	Yes	Yes	Yes	Yes
Hit-list	Possible	Possible	Possible	Possible	Possible
Independent & Uniform	Yes	Yes	No	Yes	Yes
Sequential	No	No	Yes	Yes	Yes
# of lines	16855	21629	7371	3093	19021
Modularity	Medium	High	Low	Low	High

Table 8: Botnet source code study.

Table 8 shows the scan strategies and complexity of the bot families. Some of them are modularly well designed. Currently, these bot families mainly use simple scanning strategies. Each supports both *Global* scanning (a specified address block) and *Local* scanning (relative to each bot's address). By hit-list 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. The five bot families we analyzed do not directly automate hit-list scanning, but an attacker can *possibly* achieve this via two steps, first scanning to gather a list of live addresses/blocks, and then specifying these at the command line. 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 use either 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 scanning quite frequently.

A.2 Modeling Botnet Global Scanning

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

- **Cover the target scope fully.**
- **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 simply the host computer going offline, the botmaster will prefer instructing multiple bots to scan the same addresses.

A similar analysis is proposed in [19] for worms. 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. Most of the events we found in our datasets are close to uniform scanning.

Advanced Scanning Strategies.

In fact, by introducing some simple coordination between bots one can do better than random uniform for both coverage and redundancy. An advanced scanning strategy, called "worm scan permutation", was proposed in the context of worm propagation [28]. But the above strategy is optimized for worms and does not consider the usage of C & C channels of botnets. Potentially, with C & C channels botnets can achieve even better coordination. Using the botnet C & C, we propose a better scan strategy called Advanced Botnet Permutation scan (ABPS). Each bot permutes the whole scanning scope in the same way with a key from botmaster. Then based on bots' capabilities, the botmaster divides the replicates of the permuted IP scope to all the bots. This can achieve much better coverage and redundancy. We simulate and evaluate this strategy in our evaluation.

B. PROOF OF THEOREM 1

PROOF. There are totally d^n ways to distribute the n scans into d addresses. Among them if there are X_0 ways which have z_0 addresses receiving zero scan (*i.e.*, z_0 empty slots). Then, we know $P(z_0) = X_0/d^n$. We will show that for a given z_0 the X_0 is

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

In d addresses, there are $\binom{d}{z_0}$ configurations to select which z_0 addresses got zero scan. Each configuration has z_0 addresses which got zero scan and $d - z_0$ addresses got non-zero scans. Stirling2(n, m) denotes the number of ways of partitioning a set of n element into m nonempty sets [29]. 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. Hence we proved

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

□

C. PROOF OF THEOREM 2 AND 3

Proof of Theorem 2:

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,

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

□

Proof of Theorem 3:

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, $VAR(\hat{\rho}) < VAR(\hat{\rho}_i)$ □