Towards Automating Analysis of Large-Scale Honeynet Events

Paper ID 343

ABSTRACT

Inspired by the work of Yegneswaran and colleagues on "Internet situational awareness" [30], we investigate ways to analyze data captured by *honeynets*—unused address blocks on which we deploy honeypot responders in order to elicit information about incoming probes—to understand the significance of large-scale "events" seen by the honeynet. In such events, an entire collection of remote hosts together probes the address space monitored by the honeynet in some sort of coordinated fashion. Our goal is to develop methodologies by which sites receiving such probes can infer information about the probing activity: does it reflect deliberate scanning, or a misconfiguration? If deliberate scanning, was the site itself specifically targeted, or only incidentally probed as part of a larger, indiscriminant scan? What can our analysis tell us about the scanning strategies employed by the botnets that dominate today's attack landscape?

Our analysis draws upon extensive honeynet data to explore the prevalence of different types of scanning, including properties such as trend, uniformity, coordination, and use of pregenerated "hit lists." In addition, we work towards extrapolating a scanning event's global scope as inferred from the limited local view of a honeynet. Cross-validating our inferences with data from *DShield* shows that our inferences exhibit promising accuracy.

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 a new worm propagating? Is the site being deliberately targeted? Is the site simply receiving one small part of much broader probing activity? Or does the activity in fact reflect a benign error of some sort, with no malicious underpinnings?

The answers to these questions greatly influence the resources the site will choose to employ in responding to the activity. Yet, how can a site attempt to infer which of the many cases a given spate of activity reflects?

In this work we undertake to develop a set of analysis techniques that sites can employ on *honeynet* data they gather by instrumenting an unused subset of their address space with honeypot responders, such that when the honeynet is probed, the responders elicit enough further activity from the sources such that related probes can be grouped into larger-scale "events." Our techniques aim to characterize different facets of such events to enable sites to draw inferences regarding the underlying nature of the activity.

We orient much of our methodology with an assumption that most such events reflect probing from the coordinated *botnets* that dominate today's Internet attack landscape. To this end, a particular touchstone for our analysis is to enable sites to determine the global targeting scope of likely botnet scanning, so that the sites operators can assess whether their network is a specific target of botnet activity, or if the botnet scanning is targeting a large network scope and that the site simply happens to be part of it. Naturally, the operators will have different attitudes towards these two cases.

We base our analysis techniques solely on activity seen at what we presume to be a site's fairly modest honeynet (spanning say a few hundred or so unused addresses), with no external knowledge assumed such as relate activity seen by other sites, or information regarding botnet command & control mechanisms. We then seek to answer questions such as: Given a scanning event observed in the honeynet, how can we infer properties such as the scan's progression, uniformity, degree of coordination, and whether it reflects a pre-generated "hit list"? How can we extrapolate the global properties of such activity, in terms of its broader scope and how many hosts (bots) participated? What can we learn by applying our techniques to previously recorded activity in terms of how do attackers employ botnets today to conduct their scanning operations? To what degree might their opportunities be to do so more efficiently in the future?

In addressing these questions, our work makes a number of contributions. First, we examine the source code of five popular types of bots to understand the scanning techniques they provide. One particularly popular, and fairly efficient, such scheme is uniform-random, which we analyze in some detail. We also develop a more advanced permutation scan strategy that could perform considerably better, and design a detection method for it (Section 3).

Second, we develop a set of statistical approaches to assess the attributes of large-scale events seen in honeynets, with an emphasis on the characteristics of the scanning patterns. These techniques include checking for trends, uniformity, coordination, and hit-lists (Section 5).

Third, for some types of scans we develop further techniques to extrapolate the global properties of the scanning event based on honeynet's limited local view. To do so, we devised two algorithms with different underlying assumptions and accuracies. These attempt to infer the global scanning scope, the total number of bots including those unseen by the honeynet, and their average scanning speed. We discuss these issues in Section 4.

We drive the development and evaluation of our techniques using a twelve-month trace corpus of honeynet traffic gathered at a large research institute, totaling more than 220 GB of data as described in Section 6. From this data, we find that most scan events exhibit uniform random scanning, but some events reflect hit-list scanning. We are able to confirm that our statistical approaches for inferring other event properties are generally consistent with manual inspection/visualization.

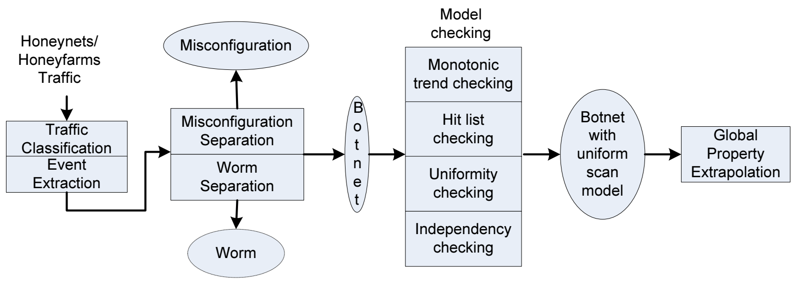

Figure 1: System architecture

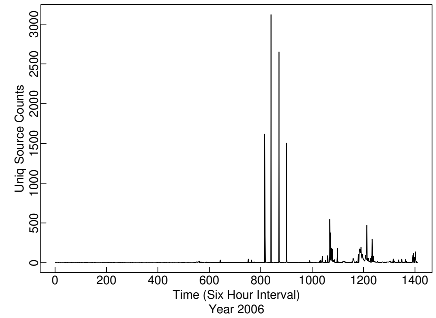

Figure 2: Temporal distribution of unique source count for VNC(5900)

To validate our estimates of the global properties of events, we compared our results with matching ones from the DShield project [24], the Internet's largest global scan alert repository. For most events, we find that our extrapolated scope is within a factor of two of the scan scope observed in DShield data. These results demonstrate that our approaches are accurate enough to enable site's to make reliable inferences regarding the degree to which an event reflects specific targeting of the site.

2. FRAMEWORK

While our major contributions focus on botnet scan model checking and global property extrapolation, to help understand the whole picture, in this section, we will introduce other modules in the system. The architecture of our design is shown in Figure 1. Except there is one step in the event extraction module which needs human intervene, all the other steps in our system are fully automated. We automatically classify the traffic seen on the honeynet sensors by different protocols or session semantics. A *session* is defined as a set of connections with a specific purpose between a pair of hosts, where the set of connections might involve multiple protocols. We extract spikes where there were a large number of unique source arrivals in the time window. We automatically classify the spikes into misconfigurations, botnet sweeps and worms. Our main focus is on botnets for automatically determining the scanning model employed and extrapolating the global properties of the botnet based on observations from our local sensors.

Internet background radiation can be interpreted as a multiplex of different signals. We primarily focus on the number of unique source arrivals in a given time window. According to the classification proposed by Yegneswaran et al. [30], large spikes in this signal usually correspond to large scale events, such as botnet sweeps, worm propagation and misconfigurations. We use a automated techniques for analyzing events rather than visualization technique used earlier. Automated approach is more accurate and can be done quickly on large data set as well as it is important for real-time event analysis and classification.

2.1 Honeynet and Data Collection

While traffic sent to unused Internet addresses ("darknets") can reflect a variety of activity, often we cannot determine the nature of the activity by simply watching passively as probes arrive, because the specifics of most forms of activity only manifest after the source establishes a connection (or, sometimes, a whole set of connections comprising a *session*) with the destination. As a general approach for doing so, we can take traffic sent to unresponsive darknets and channel it to a *honeypot* system that will respond in some fashion. Such a combination is often referred to as a *honeynet*.

Honeynet systems can employ low- or medium-interaction honeypots [21, 6], which provide fake responses of varying detail, and thus can elicit a range of possible activity from the sender. Going further, one can employ high-interaction honeypots (full, infectible systems, often running inside virtual machines), which when coupled with a honeynet is termed a *honeyfarm* [27, 12].

Our system is designed to analyze honeynet and honeyfarm traffic and works with most honeynet/honeyfarm techniques [21, 6, 27, 12]. We want some portion of the monitored IP space to be dark rather than coupled with responders, which we use for hit list detection.

We expect that load reduction filtering techniques [19] will be employed by the honeynet, which influences the design of our analysis tools. We assume that the honeynet responds to at least one session for each sender in a given time period, thus providing payload information beyond the initial SYN packets. We further assume that the sender tries to perform the same activity on all other IP addresses it contacts. This assumption is claimed to hold in general as shown by Pang et al. [19].

Honeynet Details: Our detection sensor consisted of ten contiguous class C networks, where five of the networks used the Honeyd responder and the other five were dark. This configuration is similar to that used by Pang et al [19] and Yegneswaran et al [30], and we further adapt the source-destination filtering employed by Pang et al [19]. We evaluate our analysis techniques using trace data collected over an entire year (2006).

We supplimented our analysis using three weeks of data collected from a honeyfarm [12] consisting of four class B networks and some class C networks.

2.2 Traffic Classification

Some 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 a http download command. In general, the application protocol contacted first is the protocol being exploited, so we label the entire session with the first protocol used. This also provides consistent labelling for those connection attempts where the honeynet did not respond, where we observe only the initial SYN packet. We aggregate the connections to sessions using an approach similar to the first step algorithm by Kannan et al [13]. 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, as Kannan et al [13], and found that this grouped the majority of connections between any given pair of hosts.

For application protocols which are not commonly used, the average background radiation noise is low and thus port numbers are used to separate event traffic. However, noise is usually quite strong for more popular protocols, thus 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 [30], which further classify the traffic within one application protocol or one application protocol family by rich semantic analysis. We analyzed the semantics of 20 common and backdoor protocols based on Bro's application semantic analysis [20], and generated a session summary for each session (e.g., `445/tcp/[exploit] (NAMED_PIPE: "\\<dst-IP>\IPC$ \wkssvc"; RPC request (4280 bytes)))`. Based on the session summary we can further classify the traffic within one protocol family.

Given that we use the first connection protocol as the session protocol, we will treat botnets or worms with multiple infection vectors as independent cases if they choose different exploits for each IP address. The traffic classification process is fully automated.

2.3 Event Extraction

Figure 2 shows unique source arrival counts for VNC (TCP port 5900) for 2006 on our honeynet, where each point represents a six hour interval. By studying such large spikes, we can gain insight of botnets, worms, and misconfigurations, which in turn will help in improving the situational awareness of the target network. We classify such large spike as *events*. Signal strength S is defined as the peak of unique source count arrival, and the typical unique source count when there are no events is defined as noise strength N. Noise strength is calculated as the median of unique source counts of every time interval for T_N days before the event. If the event occurrence time is less than $T_N/2$, then noise strength is the median of the time window T_N. We define the signal to noise ratio as $SNR = \frac{S}{N}$, and examine only those events with large SNR. In this paper we use a six hour time interval, based on the observation that usually events do not last very long. Since we after an event is extracted we will refine the event into smaller time intervals, the time interval select here will not influence the final results much. We use $T_N = 120$ (30 days) since we never see an event last more than two days. The thresholds we use are $SNR \geq 50$ and the number of unique senders in an event ≥ 100.

We calculate the unique source count of every pre-defined time interval for a given protocol. Event extraction is done using time series analysis. While many general statistical signal detection approaches might be applied here, we currently extract the events semi-manually. We first automatically extract potential events using the following algorithm: for any given time interval, we calculate the median of the previous T_N intervals and the SNR. For those spikes which exceed 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.

After an event is extracted, we might refine the event by re-scaling it into smaller time intervals and recalculating the unique source counts. We use manual analysis and visualization techniques at this point, since re-scaling might make the shape of events more complex.

2.4 Misconfiguration and Worm Separation

Events with a large number of sources are usually misconfigurations, botnets and worms [30]. We separate misconfigurations from worms or botnets based on the observation that botnet scans and worms contact a significant range of the IP addresses in the sensor, whereas events with few hotspots target are caused by 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 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.

Worm behavior and botnet probes are quite similar: both scan and send exploits to the address range in a similar manner. However, usually the number of sources for worms grows much more quickly than botnets, and events also last longer for worms. But if botnets scans the entire Internet, with new infected bots continuing to join the scan activity, then there is no observable difference from worms. Hence it is difficult to define a strict distinction between botnet sweeps and worms. In this paper, we treat all events with an exponential growing trend in the number of sources as worms [31], and the other events as botnet sweeps. We use the Kalman filter based exponential trend detection proposed

Botnet name	Agobot	Phatbot	Spybot	SDBot	rxBot
Global	Y	Y	Y	Y	Y
Local	Y	Y	Y	Y	Y
Hitlist	N	N	N	N	N
Indep/ Uniform Random	Y	Y	N	Y	Y
Sequential	N	N	Y	Y	Y
# of lines	16855	21629	7371	3093	19021
Modularity	medium	high	low	low	high

Table 1: Botnet source code study

in [31] to differentiate botnet and worm events.

3. BOTNET SCANNING MODELLING

After filtering the misconfiguration and worm traffic, we focus on the botnet scans because currently the botnet is the major tool to launch Internet-scale attacks. In the 43 botnet scanning events we found, 32 (74%) of them after a successful scan follow a malicious payload. 13 of them related to five known exploits to Windows file sharing or PRC protocols; 9 of them related to attempt to existing backdoors; 6 of them related to attempt a VNC vulnerability and the other 4 related to MS SQL server vulnerabilities. Therefore most of the scan events are dangerous.

Several approaches can help to understand the botnet phenomena: source code study, botnet behavior study, and command and control (C&C) study. The last one is to examine the Internet Relay Chat (IRC) traffic or other C&C channels that botnet use for communication. However, currently, the trend has moved towards using private IRC servers or other communication protocols, such as Web or P2P. In addition, the botnets may encrypt their C&C channels.

Thus in this paper, we mainly focus on source code study and behavior study to investigate certain scan properties of a botnet. In the rest of this paper, we refer different scan strategies as different *scan models*. Each of the model include a set of unique characteristics.

3.1 Botnet Source Code Study

By analyzing five different popular botnets source code genres, we study different scanning strategies employed by botnets. Our study also confirms with the findings in [7]. But they does not provide any details on the scanning model.

Table 1 shows the scan strategies and complexity of botnets. We found most botnets have relatively complex design, but some of them are modularly well designed. Currently they use simple scan strategies. We found all of them supports global and local scanning, but do not support hit list scanning directly[1]. *Local scan* means each bot select the scanning range based on their local IP addresses. *Global scan* means the botmaster lists a set of IP prefixes and ask all the bots to scan these IP ranges, which are independent from the bots' local IPs. *Hit list* scan means the botmaster asks the bots to download a large IP blocks or IP lists, and scan only those IP blocks and IPs. Typically, the list could be very long and not suitable for a single command line. There are many different ways to generate the list, as mentioned in [25]. But the list should be a list of individual IP or IP blocks which mainly consist of either alive IP/IP blocks, or vulnerable ones. One typical practise is to avoid the dark spaces [2] identified by earlier scannings as observed in Section 6. In the following section, we show the certain observed hit list scan behaviors, which only target honeynet IP blocks, but not the darknet IP blocks.

We also confirmed the results of source code analysis with our datasets. Most scanners in our dataset either use simple *sequential scanning* or *independent uniform random scanning*. The simple sequential scanning is based on $IP_{n+1} = IP_n + 1$. More sophisticated monotonic trends are also observed although, very infrequently in the honeynet sensors. But we design mechanisms to detect them as well. Among all the random scanning studies, the most used scanning technique is independent uniform random scanning. It means each bot target the same scanning scope, scan uniformly, and independently choose the seed for the random number generator. So statistically they are independent. However, certain non uniform cases are also observed. Cases in which senders have positive correlations are manually checked by us. And we found out that some senders share same scan sequences, which can be explained by the fact that it might be a seeding problems.

Although currently we did not observe any complex scan strategies used, we modelled possible more efficient scan strategies and discussed whether we can detect their existence by using only a single sensor.

3.2 Modelling Botnet Global Scanning

As mentioned earlier, local scan can avoid the detection or filtering by most IDS/IPS and firewall systems which usually sit on the edge routers or gateways of enterprises, but they usually have limited scan scope. When the attackers have special interests to certain IP ranges, they have to use global scan. Therefore the global scan is very important which is the focus of this paper.

Rajab *et al.*study the botnet traffic through botnet binary analysis and executing it in a virtual machine environment. They analyzed the spread patterns and find that bots very seldom scan the whole Internet [22]. So it usually scans a pre-compiled hit list or a list of prefixes given by the botmaster. Next, we study each of the two cases.

For worms, hit lists are mainly used to increase the infection speed especially at its early stage [25, 26]. Similarly, botnets can utilize hit list techniques to improve the scan or re-scan speed. For example, the botnet might only scan the routable address in one or two /8 networks [29]; or it might only re-scan the live IP blocks detected by previous

[1] Although the attacker can archive hit list scanning via two steps of scanning, get the alive IPs or IP blocks, then specified it at the command line, but no mechanism to automated the process observed in the source code.

[2] The dark spaces includes the IP blocks that do not respond to scanning, the unallocated IP blocks and the IP blocks with strong firewall policy, *etc.*.

scanning; or it might only target the vulnerable IPs detected previously, *etc.*. It is also possible that the botnet carry a two stage scanning. In the first stage it finds all the live IP blocks by moderate uniform scanning, which might not reach every IP address, however it is good enough to tell whether the IP blocks are alive or not, and then use the hit list scan to scan the alive IP blocks more carefully. Given a large hit list, the problem of effectively scanning it is similar to problem of effectively scanning a given single large scan scope. The following analysis for the scan scope is also applicable to the hit list.

There is a huge design space available for botmasters to develop scan strategies. But the following features are usually desired.

Coverage and scan distribution. Given a scan scope, the botmaster hopes to cover it fully. But a poor seeding of the random number generator may cause only a small number of IPs are really scanned by the bots.

Load distribution. A botmaster probably wants to distribute the scans based on the bots' capabilities.

Low communication overhead. The ideal strategy should be that the botmaster only gives one scan command, and the bots can automatically divide the scanning job optimally. Thus the extra communication between botmasters and different bots is minimized.

Scan detection evasion. Botmasters may not want the bots to scan aggressively a small IP range, which is easy to be detected and blocked by the IDS/IPS systems. At the same time, they want to maintain a good speed of scanning.

Redundancy. In a botnet, each bot can be turned off or taken down by people any time. So, for any given destination, the botmaster wants multiple bots to scan it which can be measured by the redundancy factor ϕ meaning that for any destination at least ϕ bots scan it. For d destination addresses, we need to send out at least $\phi \cdot d$ scans, and each destination gets exactly ϕ scans from ϕ different senders, which is optimal in this criterion.

Given all these desired features, a simple and effective approach is to ask each bot to uniformly scan the whole range. It can achieve the scan detection evasion, low communication overhead and load distribution very well and at the same time it has good performance on coverage and redundancy. It is also very simple and easy to implement correctly. In the source code analysis we do find that it is the most popular one implemented till now (four out of five bot genre 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 uniformly scans the alive IP blocks.

3.3 Advanced Scanning Strategies

Independent uniform scanningis not optimal in either coverage or redundancy. For example, if totally d scans are sent out to d address, the coverage only $(1-1/d)^d \approx 0.68$. This shortcoming can be addressed by the addition of coordination between the scanning sources. That is, make the senders have certain negative dependencies so that the senders can have less scan collisions. An advanced scanning strategy, called "worm scan permutation", was proposed in the context of worm propagation [25]. 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, we propose a new and better scan strategy called advanced botnet permutation scan (ABPS). This can achieve much better coverage and redundancy. Due to the interests of space we ignore the detailed design here, please refer to our technique report [**?**] for details. We simulate and evaluate this strategy in the our evaluation.

4. EXTRAPOLATING GLOBAL PROPERTIES

One of the major focus of this paper is extrapolating global properties given the limited local measurement scope. In particular, we are measuring Global target scope, Global Bots population, and the Global scanning speed of the bots. These measurements helps the network admin to determine that they are being targeted or simply a part of a bigger scanning event. Most of the botnet scanning events either follow the uniform random scanning model or uniform hit list model. These observations are confirmed both by theoretical botnet source code study and experimental observation of the LBL dataset that we present in the next section. In this paper, we focus on extrapolating global properties for scan events under these two models.

Next, we will introduce the assumptions and requirement for our extrapolation schemes. Then discuss the global properties to be extrapolated and finally discuss our two extrapolation approaches.

4.1 Assumptions and Requirements

There are certain assumptions that we have to make before any extrapolation can work. First, we assume the attacker is oblivious to detection sensors. Otherwise they can avoid sending scans to them and we will not see any scan traffic at all. This assumption is fundamental to general honeynet-based traffic study (cf. 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 our approach does not assume collaboration between sites, we need not release results to the public, which counters the basic attack in [9]. With this assumption, the local view of detection sensors serve as random samples of the global scan scope.

Second, we assume each sender has the same size of global scan scope. This will be true if all the senders are controlled by the same botmaster and each sender scans uniformly.

We believe that these two fundamental assumption applies to any extrapolation schemes. In addition, there are two general requirements for our approaches. We check for these requirements before applying extrapolation.

Property name	uniform scanning	uniform hit list	hardness
Global target scope	Y	Y	indirect
Total # of bots	Y	Y	indirect
Total # of scans	Y	Y	indirect
Average scan speed per bot	Y	Y	indirect
Coverage hit ratio	Y	N	direct
Sender OS distribution	Y	Y	direct
Sender AS distribution	Y	Y	direct
Sender IP prefix distribution	Y	Y	direct

Table 2: Property list

First, we presume that each sender evenly spreads out its scans in the global scan scope. Therefore, to apply the extrapolation approaches, we check the scan model as discussed in Section 5. The dark regions shown in Figure 4 are the scan models required. If the scans are uniformly distributed, then our extrapolation can be applied to both uniform random scanning and random permutation scanning. Uniform random scanning implies that each sender is independent and random permutation scanning implies that there is negative correlation between senders. Almost all of the botnet scan events detected pass the uniformity test, as mentioned earlier.

Second, we also require the aggregated local scan speed of the set of bots observed in the honeynet (and thus applied in the extrapolation) to be fairly constant. We can check this by the linearity of the scan arrivals of the set of senders with time.

If the aggregated local scan speed we see is close to constant, than we know the aggregated global scan speed of the bots will be close to constant too, under uniform scan model. This also will suggest that the botnet population is quite stable. If after the botmaster type the scan command most bots die quickly and cannot finish the scans, the total scan rate will decrease and will not be constant. By observing the constant scan rate, we know most bots are still alive in the given scan time period, and can ignore the bots dynamics. This is observed in all the botnet events we analyzed. Part of the reason is the scan event did not last long, usually are a few hours.

There are also some additional requirements which are specific to certain extrapolation approaches and need to be validated as well. We will explain them when introducing specific extrapolating techniques.

4.2 Properties Extrapolated and General Schemes

Table 2 shows the list of properties we currently extrapolate. When considering the local view as a random sample of the global view. The OS distribution, AS distribution and IP prefix distribution from local measurements should be the similar to the real global distributions. However, if a bot scans slower than others, which indicated less scans it sent out; then less chance we can see it, based on the uniform model. Therefore, the extrapolation of these three distribution properties will have some bias towards fast bots in the population. The higher the portion of the bots we see, the less the influence of the bias is. By extrapolating the total number of bots, we can roughly know which case is more accurate. The coverage hit ratio gives the percentage of target IP addresses indeed scanned by the botnet. This metric is not very meaningful for the hit list cases, thus we mainly consider this for uniform scanning for which certain destinations are not reached due to the chance variation. This metric can be directly measured with the local sensor and the local result is an unbiased estimation for the global result under uniform scanning. In this section, we mainly study how to estimate the remaining properties (the first four listed in the table 2).

d	The size of the local sensor
G	The size of global target scope
ρ	The local over global ratio d/G
M	The total # of senders in the global view
m	The total # of senders in the local view
m_1	The # of senders in the first half of the local view
m_2	The # of senders in the second half of the local view
m_{12}	The # of overlapped senders of m_1 and m_2
R	The average scanning speed per bot
R_{Gi}	The global scanning speed of bot i
T_i	The time between the first and last scan arrival time from bot i
n_i	The number of local scans observed from bot i
T	The event duration observed in the local sensor
Δt_j	The inter-arrival time between the j and $j+1$ scans
Q	The local total # of scans

Table 3: Table of notations

The notations we used in our problem formulation and analysis are shown in Table 3. We use $\hat{\rho}$ and $\hat{G}$ to represent the estimated probability and the estimated global range respectively. Note that since ρ is usually quite small, many senders may not arrive the senor at all. With the uniform scan speed assumption discussed above, we have:

$$\frac{Q}{R \cdot T \cdot M} = \rho \quad (1)$$

$$\frac{m_1}{M} = \frac{m_{12}}{m_2} \quad (2)$$

The idea of Equation 1 is that on average the number of the scans received by the sensor equals to ρ portion of total scans. If we split the IP range of the sensor to two parts, and the senders we observed in the two parts are independent samples from the total population M, we can have Equation 2 based on independency. For example, suppose there are total $M = 400$ bots. In first half sensor we see $m_1 = 100$ bots, which is $1/4$ of the total bot population. Consider the second half of the sensor is another indepedent sensor, so the bots it observed is another random sampling from the total population. Then, there are $1/4$ chance to see the bots which has already seen at the first half sensor. If the second half also observed $m_2 = 100$ bots, the share bots (observed by both first half and second half sensors) are close to $m_{12} = 100/4 = 25$.

Here, Q,T,m_1, m_2, and m_{12} are known, and there are three variables, R, ρ and M in the two equations above. We can solve $\rho \cdot R$ and M. Thus we need either to estimate ρ or R with some extra information independent from these two equations, we can estimate the other one. Then we can further use Q/ρ to estimate the total number of scans sent by the botnet.

In this paper, we developed two approaches. The first approach estimate ρ and the scope. it is is more accurate.The second approach uses minimal inter-arrival time of bots to estimate the average scan speed R.

4.3 IPID and Ephemeral Port Number Based Extrapolation

The basic idea of first approaches is to try to estimate the global scan speed for a given sender by measure its IPID or ephemeral port number changes. Then based on that to estimate the global range. Since the first approach requires to use IPID information. Before apply them, we needs to use p0f to fingerprint the OS versions. If majority are above Windows 2000 or MacOS then we should be able to use the IPID information. In practice for all the events, we observed are mainly Windows machines, more than 80% hosts in an event can be identified as Windows. And most of the other 20% hosts cannot be identified by p0f, and we conjecture they are also Windows, since the botnet usually homogeneous. And for the 80% of hosts identified as Windows, more than 95% of them are identified as Windows 2000 above so that they have big endian IPID.

IPID continuity For Windows or MacOS machines, the IPID field of the IP packets is a 16-bit global IP packet counter of the system. Therefore, for every consecutive packet pair, the IPID are consecutive too. If the machine is mainly idle when it does the scanning, in a give time interval when the IPID overflow does not happen, the IPID difference can be used to measure how many scans sent by this sender in the given time interval. Of cause we need more complex algorithm to consider the IPID overflow and its recovery in general. This metric is pretty reliable, since the only way to change IPID behavior is to implement the scan in raw IP socket, and simulate the TCP/IP stack themselves [3], which is quite hard for most of attackers.

When using IPID to estimate the scan speed, one potential problem is the TCP SYN resending. We estimate the global scan speed by the speed estimated by IPID divide by the average TCP SYN resending observed from the same sender.

Ephemeral port number continuity When analyzing the botnet source code we found all the botnets let the operating systems to allocate the ephemeral source port numbers, since it is easier to work with in non-block I/O or muitlthreading cases. Most operating systems sequential allocated the source port number. Therefore, the source port changes can be used to measure how many scans send out also. However, this metrics is less reliable since if the attacker allocate the source port number themselves, potentially they can use any allocation algorithms.

IPID and ephemeral port number continuity validation In a control experimental environment, we install five different Windows versions and one MacOS10 and two Linux versions on different virtual machines. Nmap scanning tool is used to generate scans. We tested the IPID behavior and ephemeral port number allocation behavior of eight OS versions as shown in Table 4.

OS version	IPID continuity	IPID byte order	ephemeral port number continuity
Win98	Y	little endian	Y
WinNT4	Y	little endian	Y
Win2000	Y	big endian	Y
WinXP	Y	big endian	Y
Win2003	Y	big endian	Y
MacOS10	Y	big endian	Y
Linux 2.4	N	big endian	Y
Linux 2.6	N	big endian	Y

Table 4: IPID and ephemeral port # continuity testing

On all the versions of Windows systems, the IPID is a global packet counter of the systems. Every packet sent out is counted. The only complexity is before Windows 2000 the IPID is in little endian, and Windows 2000 and the versions after Windows 2000 the IPID is in big endian, the network byte order. From the install based statistics [4], we found the Win98 and WinNT4 only occupied less 1.5% percent of install base. Moreover, in the evaluation we will show that based on the OS fingerprint the percentage of Win98 and WinNT4 is quite low. Therefore, we current ignore the byte order problem. The ephemeral port number is continuous to the scans send out, which is independent of operating systems.

NAT consideration Since the NAT boxes usually translate the ephemeral port number, we tested how the NAT boxes influence the ephemeral port number and the IPID behavior by using popular home routers, such as Linksys, Netgear and D-Link. These three brands shared more than 70% of the home router market [1]. We still use the Nmap to send the scans from hosts behind NAT and tested how the ephemeral port number or IPID change after the NAT box. We found in all the three brands, the IPID remain unchanged. If only a single scanner behind the NAT, the ephemeral port number also remain unchanged. If there are multiple senders behind NAT and they scan in same time, at least the ephemeral port number of the first sender remain unchanged. But for the D-Link router, the ephemeral port number of the second sender become totally random. Therefore, we can conclude in the NAT cases, if there is a single sender we should still be able to use IPID reliably. And in the multiple senders cases we might not be able to use IPID or ephemeral port to estimate global speeds. A technique of counting number of hosts behind NAT has been proposed in [8]. However, their approach assumed large portion of the traffic from the NAT box can be observed. In our case, only a few scans can be observed in the sensor. Therefore, we did not tried to differentiate different IPID sequences from a single NAT boxes. Since usually we can see large number of senders, even we ignore such cases, we still get enough senders for the extrapolation.

[3] From the source code study we found, all the scans when successful contain exploits.

Basic design Suppose there are m senders seen by the sensor, and a subset of the senders, m' senders, can estimate the global scan speed by the certain approach. The m' senders are with global speed R_{Gi} each. For sender i, we known in T_i, we observed n_i scans from it in our sensor. Since its global scan speed is R_{Gi}, globally in T_i it sends out $R_{Gi} \cdot T_i$ scans. Therefore we know $\frac{n_i}{R_{Gi} \cdot T_i} = \rho$. This is also true when we aggregate through all the m' senders. Therefore, $\frac{\sum_i^{m'} n_i}{\sum_i^{m'} R_{Gi} \cdot T_i}$. We use this formula to estimate $\hat{\rho}$, which will have less estimation error than only use a single sender. This technique can also be used to a single sender. But in that case, the reasonable large number of scans of the sender have to be seen on the sensor. Otherwise it will not be accurate. That is the reason we want to aggregate a set of senders.

Global scan speed estimation For the first approach, we find the top senders which send no less than four scan sessions. We test whether their first IPIDs or first ephemeral port numbers (after overflow recovery) of the sessions increase linearly with respect to time. IPID and ephemeral port number can be applied similarly for global scan speed estimation – they only differ on the overflow recovery. Next, we will mainly discuss IPID based method.

First, for two consecutive scans, if the IPID of the second one is smaller than the first one, we increase the IPID by 64K. Then we try to fit the corrected $IPID_i$ and corresponding arrival time t_i into a line. If they can be fit into a line with correlation coefficient $r > 0.99$, it means that the scan speed is close to a constant, and the sender is a single host but not multiple hosts behind a NAT. Then we can estimate the global speed using the slope. It is possible that multiple overflow happened for the IPID or ephemeral port number. In that case, the simple overflow recovery approach will fail, but then the chance that the IPIDs can still be fit into a line with arrival time is very small. We will discard such cases. But as long as we can have a sufficient number of senders which we can estimate the global scan speed (which is usually the case observed in our dataset), we can get an accurate $\hat{\rho}$ as discussed above.

For the overflow recovery of ephemeral port numbers, we leverage on the fact that most Windows machines the port number range is either 3976[4] or 64K. Therefore if the range of port numbers we observe is close to 3976 or 64K, we set its period to 3976 or 64K respectively. For other cases, we just use the actual range we obtain from the data, *i.e.*, the maximum port number minus the minimum port number.

Since if there is some steady background traffic on the machine, it is possible that we overestimate the global scan speed. Currently, we use the minimum value between the IPID based estimation and ephemeral port number based estimation to reduce the chance of overestimation.

[4]Windows default port number range from 1025 to 5000

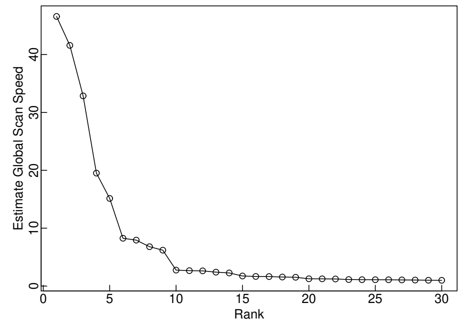

Figure 3: Top 30 estimate speeds of Event VNC-060729

4.4 Extrapolation through Minimal Inter-arrival Time Modelling

By solving Equations 1 and 2 in section 6.2, we get values of M and $\rho \cdot R$. We use the following method to estimate the average scan speed R. If we know a sender's global scan speed is s, then it is the upper bound of the possible local speed we can see in the sensor. If two consecutive scans from that given sender are observed by the sensor, then the inter arrival time $\Delta t = 1/s$. Basically, for a given sender, we can use inverse of the inter arrival time of the closest two scans seen from that sender to estimate its global scan rate. This approach may underestimate its global scan rate.

For all the m senders we observed, except the ones only observed one scan, for any given sender we estimate the minimum inter arrival time Δt. Then, we rank the senders by the their estimated global scan rate $s = 1/\Delta t$ in descend order. The first sender has the maximum estimate speed. Note that those fast senders tend to have larger estimate speeds. We verified this statement by correlate the minimum interarrival time of each sender with how many scan we observed from it. Usually the higher their sending speed the more scans we will observed. Due to the random scan nature, there are no deterministic relationship. But we do find on average the bots send more scan have smaller minimum interarrival time. Therefore, to use the top bots' speeds as the average speed of all the bots may overestimate the average speed. But for each sender we may underestimate their global scan rate. Thus is crucial to find a balanced point. Currently we choose the knee point of the estimate global speed distribution among the top senders where the knee point is the smallest rank k for which increase of k will have little change on s. An example of the top 30 maximum estimate speeds of Event VNC-060729 is shown in Figure 3. From the figure we derive that $k = 6$ is the knee point with the estimate speed 8.26. We use the similar methods to find the knee point for all the 37 events. This approach in general is less accurate than approach I but it does not need the IPID and ephemeral port number information, therefore more general.

5. BOTNET SCAN DETECTION

One of the major growing concern in security is botnet scanning of vulnerable application on host machine. Botnets exploits and install malicious software by finding a vulnerable host. Scanning is a predominant tool employed by botmaster to find such hosts. A little work has been dome in detecting and classifying the hideous act of scanning by botnets. In this section, we will classify and detect scan strategies used by botnets.

There are several dimensions across which a botnet can base a scan strategy, where each dimension represents a discrete characteristic of the scan model. The dimensions explored by us are presented in Figure 4. This scan model is modeled on the basis of botnet scan strategies analyzed in Section 3 and literature on worm scanning [25, 26]. We develop a set of algorithms, each designed to check a single characteristic of the scan model. We then combine the characteristics to detect the scan model used in any given event.

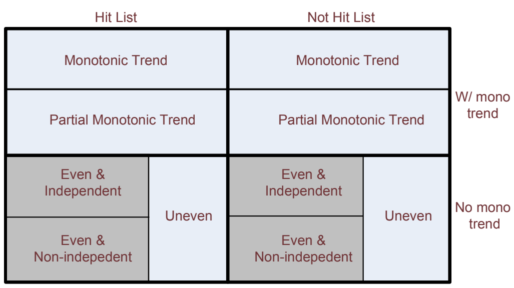

Figure 4: Model Checking Design Space

We apply a step-by-step process to detect the scan model used by bots, as shown in Figure 4. We first filter out non-monotonic trend from monotonic and partial monotonic trend. Then we detect the hit lists scanning strategy, followed by checking for uniformity. Lastly, we determine if the senders are independent. Uniformity is the measure of even distribution of scan sessions in the sensor space. As we describe below, monotonic trend checking is also used as a noise filter mechanism, which can help separate the monotonic trend senders and no trend senders.

5.1 Monotonic Trend Checking

If we know the event follows a monotonic trend, potentially we can infer how far the botnet scanned before reaching the sensor by study the arrive time difference between the fast bots and slow bots. This can help us infer whether the attacker specially target our enterprise network.

For each sender, we check for a monotonic statistical trend of destination arrivals. We label the entire event as having a *monotonic trend* if more than 80% of senders follow a trend, and ignore the remaining no trend senders. We label the event as *non-monotonic trend* if more than 80% of senders are without a trend. All other cases are labeled as a *partial monotonic trend*. By doing this step, we can also filter out the potential noise and purify the data.

Since any random sequence can be separated into multiple random monotonic sequences, we check whether the overall scan from a sender follows a monotonic trend. We apply Mann-Kendall trend test [14], a non-parametric statistical hypothesis testing approach. To make the test statistically sound, it is required that the number of scans should

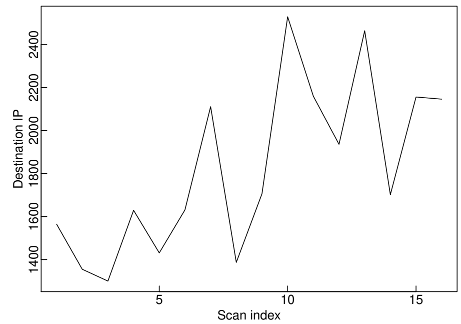

Figure 5: An Statistical Power Demonstration For Mann Kendall Test

be larger than ten [14]. In this paper we set the significance level for this test to 0.5%, which we find still has enough statistical power to detect weak statistical trends. Figure 5 shows an example of the boundary case with a p-value of 0.53%.

Given the limited view of a honeynet, we observe fewer than ten scans for many of the senders. For those senders with five to nine scans, we developed a strong monotonic check to test whether they follow a monotonic trend. For n scans ordered in time, if all consecutive scan pairs have the same sign, which means that $\mathrm{sgn}(IP_i - IP_{i+1})$ has the same value from $i = 1 \ldots n-1$, then we label the sender as having a monotonic trend, otherwise we label it as having a non-monotonic trend. The possibility for a random sequence to pass this test is $2/n!$, or 1.67% for $n = 5$. Suppose we have n i.i.d. variables from a probability distribution, $X_1, X_2, \cdots, X_n$. If the chances of $X_i = X_j$ for any $i, j \in [1, n]$ is neglectable, there are totally $n!$ possible orders of the n variables. Among them, two orders are considered to have strong monotonic trends. Since the all the $n!$ orders have equal probability to happen, the chance of false positive are $2/n!$. We further validate this via Monte Carlo method with different probability distributions such as uniform, binomial, norm and exponential distributions.

We label those cases where the number of scans from a sender are less than five as *no trend*.

5.2 Hit List Checking

Some of the sophisticated botmaster use hitlist for a targeted attack to a predefined IP space. Usually, they found the live and vulnerable host within the targeted IP space by doing a random prescanning. It is important for network admin to know that whether they are in the targeted hit list of the botnets or not. This usually imply the attackers have some interests on the enterprise so that they want to efficiently repeat the scan over and over again.

We use darknet space to detect the presence of hit list scan behavior, where the darknet space and honeynet are adjacent in IP space. If an attacker generates a hit list based on previous scanning, they will not rescan the darknet IP blocks, but will potentially rescan the honeynet IP blocks. Therefore, if we observe an event in the honeynet portion, but not the darknet portion, we know that the scan uses a hit list. This

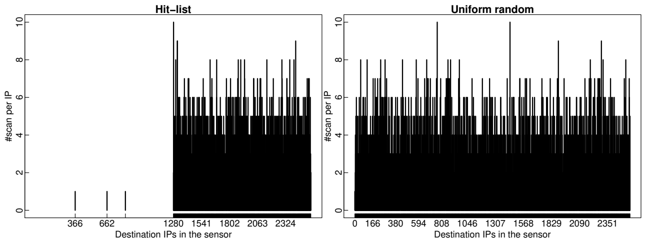

Figure 6: Hitlist and uniform scanning distribution on the sensor

detection should be effective in those cases where the attacker generates a list of IP blocks containing live addresses. For those cases where the attacker generates a hit list consisting only of vulnerable machines, the number of hosts on the list will depend on the configuration of the honeynet. Therefore, we might not detect this behavior.

In our current honeynet configuration, the first half of IP space (five /24s), are used as the darknet, which did not respond any requests, while the second half are honeynet, which has the Honeyd responder for most popular protocols and SYN/ACK responses for unknown TCP protocols.

In theory, given a hit list scan, we should see the scan activity only in honeynet and not on the darknet. However, due to noise, we might see some traffic on the darknet as well. If the signal over noise ratio SNR is high, there should still be a large difference between the number of scans observed in the honeynet versus the darknet. An example of comparisons of a hit-list case (WINRPC-061126) and a uniform random case (VNC-060729)is shown in Figure 6.

If the event is extracted based on port number (and so likely has a low level of noise), we can perform an unbiased comparison. We define the ratio of the number of senders in the darknet over the honeynet as $\theta = \frac{m_d}{m_h}$. If θ is less than a given threshold we label the event as performing hit list scanning. In our evaluation, we use 0.1 as the threshold and show that in most of our events θ is exactly zero, while in the remaining cases θ is close to 1. Therefore, the detection does not appear to be sensitive to the threshold we choose.

If the event happens on a popular scanning port and we use protocol semantic analysis to extract the events. We cannot find an exact counterpart in the darknet since we can only separate the traffic in the darknet by port numbers. In these cases, we can still perform an approximate check if the event signal is strong enough, extracting the traffic on the darknet only using the same port number and the event time window. We use θ as the detection metric with the same threshold. However, this results in overestimating the true ratio, so that the false negative rate will be higher than for the port number based event extraction.

Although other factors could potentially cause the unbalance between darkent and honeynet events, it is unlikely they will result in a small θ. One possibility is that someone chooses a small scan range that happens to include only the honeynet IP space but not the darknet. However, since the attacker does not know which part is the darknet and which part is the honeynet, the likelihood that he only scans in the honeynet will not be significantly different than the likelihood that he only scans in the darknet. For the 43 events we analyzed, we did not observe any cases in which the attack only scanned the darknet but not the honeynet. Another possibility is that the attacker performs unevenly distributed scanning. However, we found for all the events we analyzed that they scanned evenly or nearly evenly in the honeynet IP space. Therefore we believe all the cases we identify should be due to hit list behavior.

5.3 Uniformity Checking

Most of the botnets wants to evenly distribute the scan in the targeting scope both for hiding detection and covering full IP space. We used strong checks for detecting such uniform distribution cases of scan in runtime. This detection is valuable for predicting the global scan scope.

After we test the monotonic trend cases, we use the hit list test to check whether its targeting range is continuous or is from a hit list. We then check the distribution of targets in the IP space, which is used for determining how to extrapolate the global scan scope. In Section 3.2, we presented a list of criteria for scanning and concluded that a good strategy is to evenly distribute the scan in the targeting scope.

To test whether the scan is evenly spread out in our honeynet sensor can be described as a distribution checking problem. Essentially, we want to know whether the scan distribution on the sensor IP space is uniform. For this problem, the Goodness-of-Fit distribution checking approaches, such as Anderson-Darling test, Kolmogorov-Smirnov test and Chi-square(χ^2) test, are well suited. However, only the χ^2 test can be used for discrete cases allowing ties. The sensor IP space belongs to discrete cases, and multiple scans can arrive at a given destination IP address. Thus we use the χ^2 test to determine if the destination IP distribution matches a uniform random distribution.

When using the χ^2 test, an issue arises regarding how to choose the number of bins. A key requirement is that the expected value E_i in any bin must be larger than five [23]. Accordingly, for the ten class C networks in our honeynet we used 40 bins and 64 addresses per bin, with a significance level of 0.5%. Our evaluation demonstrates that these values work well.

In some cases an attacker wants the scan to be uniformly distributed but, due to a seeding problem or its poor implementation, the scan can be very non-uniform and so be rejected by the uniformity test.

5.4 Dependency Checking

One of the novel approach that we develop in the scan detection technique is dependency checking. This technique can detect sophisticated scanning strategies employed by smart professionals. This can help gain inside to how smart the adversaries are.

More advanced scan strategies, such as the permutation scan proposed in Section 3.3, can be used by an attacker to achieve better performance over the random uniform scanning strategy, in which each sender independently performs uniform scanning. The more advanced strategies require correlation among the senders, which we should be able to detect.

The null hypothesis H_o is that the senders are uniform and independent, and the alternative hypothesis H_a is that the senders are not independent[5]. If we assume the senders are independent and each sender scans in a uniform random manner, then the number of scans a given destination receives will follow a binomial distribution. Suppose the size of the sensor is d. Given a scan, the probability that the scan will hit a particular destination IP is $p = 1/d$. Therefore, if there are a total of n scans, for a particular destination the distribution of number of scans it receives will be a binomial distribution centered at $n \cdot p$.

Given we have d address in total and d is sufficiently large, we can calculate the confidence interval for how many addresses will receive k scans. For example, if we observe n scans in d addresses, how many address will not receive any scans? The more advanced scan strategies may have very different outcomes than the uniform random case. Therefore, if the difference between the distribution we observed and the theoretical distribution from uniform independent cases is statistically significant, we should be able to detect the more advanced scan strategies. The larger the value of n and d, the more sensitive the test will be, *i.e.*, the higher the statistical power.

The vacancy test checks if the number of addresses that receive zero scans is significantly different from the value we get in the uniform independent case. The vacancy test is a special case as $k = 0$. We derived the theoretical distribution and confidence interval of the uniform independent case. Due to the interests of space, we ignored the theoretical deduction here.

If the total number of scans we observed in our sensor is $n < d$ or $n \sim d$, then we can use the vacancy test since we have the exact distribution in theory. However, if $n \gg d$, we cannot use the vacancy test since number of vacancies is always close to zero. Instead, we use $k = \lfloor n/d \rfloor$ test. The distribution of $k = \lfloor n/d \rfloor$ can be theoretically derived, however in practice it is difficult to calculate the exact distribution. Instead, we use the Monte-Carlo method to simulate the experiment, which we run 10000 independent runs in order to obtain the distribution and confidence interval.

In our evaluation, we use a 0.5% significance level. We did not observe any cases where H_o was rejected in our data. However, we simulated the permutation scan strategy proposed in Section 3.3 and observed that H_o was rejected with a very small p-value. Therefore, we believe that it can be used to detect the advanced scan strategies which are likely to be observed in the future.

6. EVALUATION

We evaluated our system with the honeynet traffic described in Section 2.1. The total data is about 220GB.

As mentioned in Section 2, after we classify the traffic by different port numbers or session semantics, we extract the events. We use $SNR = 50$ and tail parameter $\omega = 5$ to extract all the events for the evaluation. We tried different ω value from 3 to 8, and found we get exactly the same results for model checking and extrapolations. For the manual time window selection step, the rule of thumbs are to select the full width at half maximum (FWHM) [28] and the range where the scan arrival speed is close to a constant. Eventually, we will automate this step as well which is part of our future work.

In total we found 303 events. There are 260 misconfiguration cases and the rest 43 are botnet scans. There is no worm breakout during the time period we studied. Among the 260 misconfiguration cases, 182 (70%) of them we observed payload in the packets. Through payload analysis, we identified 169 out of 182 (93%) are due to P2P softwares. Most of them are attributed to famous P2P softwares, such as eMule, Bittorrent, and Gnutella. Causes for the prevalence of such anomaly require further investigation which is mainly our future work.

6.1 Model Checking Results

Out of 43 botnet events, none of them follow monotonic trend, 18 of them are hitlist, 37 of them follow the uniform distribution model passing both uniformity and dependency tests

For the 43 botnet events we applied the model checking techniques described in Section 5. First, none of the events are identified to have monotonic trend or partial monotonic trend because all the events have less than 2% of monotonic trend senders. As shown in Figure 5, the statistical power of Mann Kendall trend test is quite high and can pick up some weak trends. There are totally 48,339 senders in the 43 events and 112 senders identified as the monotonic senders. We manually checked these 112 senders and found 51 of them have strong monotonic trends. The other 61 senders (0.1% of the total senders) have weak monotonic trends. Some of them can be the false positives. But even considering all of them are false positive, the false positive rate is still much lower than the expected false positive rate, the significant level 0.5%.

These monotonic senders are often noises to the non-monotonic events. After filtering them out for each event, we check the hit list property. Among the 43 events, 28 events are classified by port number while the rest 15 events are on popular scan port numbers and thus have to be classified by the protocol semantics. In the 28 events classified by port numbers, 14 of them are hit lists. Twelve out of these fourteen cases have no sender scan the darknet while the remaining two only have one sender scan the darknet each. We found θ is a good differentiator for hit list: for all the 14 hit list events, $\theta < 0.007$. On the other hand, θ of the 14

[5] we perform a uniformity test before a dependency test.

non-hit-list cases are all close to one.

For the 15 events classified by the protocol semantics, we cannot get the accurate number of senders in the darknet because the darknet does not provide any payload. So we count the senders in the darknet only by the port number, which will overestimate θ. We find that four cases can be identified as hist list events. The maximum θ among the four is 0.03, well below the pre-determined threshold 0.1. For most of the remaining cases, $\theta > 1$. Therefore totally we find 18 hit list cases. There are no events which only target the darknet.

For the 43 events, we further test their uniformity. For the hit list cases (18 of them) and the non-hit-list on popular port numbers (15-4=11 in total), we only test the uniformity on the honeynet IP space. For the rest of botnet scans, we use the traffic on the aggregated IP space of the honynet plus the darknet. Altogether, six cases fail the tests. We checked all the events through visualization. The 37 cases pass the uniformity test are indeed uniform. One out of the six failed cases has a p-value close to the threshold, and it looks close to uniform through visualization. Two out of the six cases also look close to uniform distribution. The reason they fail the test might be because they have very large number of scans, *i.e.*, the number of scans received is more than 10 times larger than the number of IP addresses in the sensor, so the statistical power of the uniformity test is very high. Any slight noise will make them fail the test. Another case might due the the seeding problems. Some senders share similar scan sequences. The remaining two cases are from a single botnet[6]. There are some hot spot destinations which received large number of scans, and make the cases deviate from uniform.

For the 37 uniform cases, finally we test the dependency. Since the uniformity test will pick up the ones with positive dependency (thus several senders hit same IP addresses), the dependency checking mainly tries to find whether any case has negative correlation which will happen if it is a permutation scan case. For the 37 cases we did not find any case with negative correlation at 0.5% significance level. In addition, we simulate the advanced botnet permutation scan (ABPS) proposed in Section 3.3, and find the dependency test can accurately detect it even with $0\% \sim 20\%$ packet loss.

6.2 Extrapolation Evaluation and Validation

As shown in the Table 2, There are eight global properties that can be extrapolated. The first four of them need inference while last four can be directly observed. We mainly extrapolate first four properties using our approach. We validate the two properties, global scan scope and total number of bots by using Dshield data, the largest scan data repository in the world.

Out of 20 non hit list uniform random scanning events, 12 are validated against Dshield data. Extrapolate scope is in factor of 2 from Dshield and only 25% estimation error in bot population estimation

We extrapolated all global properties including scan scopes for all 37 uniform events. Among them 20 are independent uniform scanning cases and 17 are hit list cases. For independent uniform scanning cases the results are total scanning ranges. And for the hit list cases, the results are total sizes of the hit lists if they uniformly choose IP from hit lists. It is hard to verify the hit list cases since we do not know who will be on the list. However we can validate the independent uniform scanning cases, since usually they target a few quite large IP blocks. We validate our results against the Dshield data, which has the scan detection sensors around the world. Due to limited access to Dshield data, we are only able to verify 12 out of 20 cases. In Table 6, the *139-NBSS* denotes the negotiation protocol semantic of the NETBIOS session protocol. The *445-negsetup* denotes the negotiation protocol semantic of the microsoft-ds protocol. And the *445-wkssvc* denotes the RPC call to wkssvc.

Validation methodology for global scope extrapolation Dshield converts time of different sensors from different time zones to UTC. Although clocks may shift in each sensor, we found that for most sensors clocks are quite synchronized. For a given event, We take two steps for validation. First, since the extrapolation results we got are mainly of /8 sizes, we try to find all the /8 networks (except those with private IP prefixes) with sufficient source overlap with the honeynet event. Secondly, for these /8 networks, we infer the scan scopes and compare that with our results.

Step 1. Assuming the /8 IP prefix of our sensor as X, we first calculate the number of shared senders $N(X)$ between the event we seen and the scan logs of X from Dshield. For remaining /8 IP prefixes, we list all /8 prefixes Y_i whose number of shared senders with the honeynet $N(Y_i)$ is larger than $N(X)/3$. This is based on the assumption that if a botnet uniformly scans multiple /8 prefixes, each of them should see quite a few shared senders. For each Y_i and X we extract the time interval of maximum unique source arrival. The methodology we used to extract the time interval is exactly same as the event extraction we used before. We select the full width at half maximum (FWHM) of the unique source arrivals as the time interval. Then we calculate the time range overlap with X for each Y_i, if the overlap of Y_i is larger than 50% of the time interval of X, we consider that botnet scans X and Y_i at same time. Based on that observation, we can find those /8 prefixes that the botnet scanned. We found that for all events the time window of prefix X is almost same as the time window of the event we observed in honeynets.

Step 2. After finding all the scanned /8 networks, we need to find the scan scope within each network scanned by the same botnet. Alternatively, we try to find the ratio of sensors in each network which report the scans. There are several limitations of Dshield data. First, it does not contain complete scan information. Secondly, different reporting sensors might be of different sensitivity and some sensor might not report certain port numbers (*e.g.*, due to firewall filtering). Thus all these limitations makes calibration of data a chal-

[6]More than 90% of sources are shared between these two events.

lenging job. To overcome these problems, we adopt the following methodology. First we pull the Dshield data for our event date and port number.

Then we try to find the Dshield sensors which can detect scans on these port numbers by checking one-week scan log starting from the event date. For such sensors, we get an aggregated IP address coverage denoted as C_{total}. Similarly, we get the aggregated IP address coverage for the sensors which report scans from the shared senders of the honeynet event, denoted as C_{est}, which is a conservative estimation on the total scan range of this botnet. Then we use C_{est}/C_{total} to estimate the fraction of a /8 network scanned by the botnet. We add up such fractions if there are multiple related /8 networks discovered in the first step and put the results in Column *Dshield scope* of Table 6. Column *extrapolated scope (I)* shows the honeynet extrapolated scan scope by approach I. Column *scope ratio* shows the ratio of the honeynet extrapolated scan scope by approach I over the Dshield scope.

Validation methodology for total number of bots estimation We assume that honeynet event and the corresponding Dshield scan data of the same /8 IP prefix are two independent samples of the bot population. Now we apply Equation 2 to calculate the total bot population[7]. To validate the total number of bots we extrapolate, we calculate the fraction of shared sources to the total sources in the time window for IP prefix X. Now, based on Equation 2, we found out that this fraction should be close to the ratio between bots observed in the honeynet event and total bots population. Since Dshield sensors will see other scanners in that period, thus the ratio will be underestimated; thus overestimate the number of bots. Based on this assumption, we can calculate the size of the bots population. We found that results are very close to what we estimated locally by splitting the sensor into two halves. For popular port number events, we cannot do payload based classification from Dshield scan logs because of unavailability of payload, thus Dshield cannot be used to verify those kind of events.

In table 6, the column *extrapolated #bots* shows the extrapolated total number of bots by splitting the sensor into two halves. The column *#bot Dshield* shows the extrapolated total number of bots using both Dshield data and the honeynet sensor. The column *#bots ratio* shows the ratio of the extrapolated total number of bots over the total number of bots estimated by using Dshield data.

Validation results After validating against Dshield data, we found out that global scope extrapolation and the total number of bots estimation are promisingly accurate. For the scope extrapolation approach I, for most cases, the global scope extrapolation is in a factor of two from the results of Dshield. For approach II (column *extrapolated scope II*), except one case, the extrapolate scope is in a factor of five from the results of Dshield. And the estimation error for total number of bots is around 25%. In Table 6 we also show the extrapolated total number of scans and the extrapolated average scan speeds.

[7] Assume m_1 is the scan observed in our sensor and m_2 is the scan observed in the same /8 prefix of Dshield.

7. RELATED WORK

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 [5] has developed a set of tools for host-level honeypot analysis [2], and one can also operate general exploit analysis tools [11] on high-interaction honeypot systems. At the network level, Honeysnap [3] analyzes the contents of individual connections, particularly for investigating IRC traffic used for botnet command-and-control.

However, all of these approaches focus on single instances of activity. In this work, we aim instead to understand large-scale events as seen by honeynets. Such activity by definition entails analysis integrated across a large number of instances of the activity. The work that most heavily influences us in this regard 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.

Finally, the literature includes a number of forensic case studies analyzing specific large-scale events, particularly worms [17, 18, 15] and backscatter traffic [16]. In particular, the work of Kumar et al. in [15] shares a similar design philosophy to our efforts, *i.e.*, an emphasis on uncovering underlying structure present in large-scale events. However, their work relied heavily upon details regarding the use by the "Witty" worm of a specific pseudo-random number generator, and on the relative ease of deconstructing the worm itself. Regarding this latter point, the Witty worm was only a few hundred bytes of executable code total, whereas some of the bots we deal with have more than 10,000 lines of code (cf. Table 1). Thus, we see their work as inspirational in terms of the structural style of analysis, but not as a source of specific techniques for us to draw upon.

More generally, 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.

8. CONCLUSIONS

In this paper we present several algorithms which can automatically analyze and determine the nature of large-scale events observed at a honeynet. In particular, for botnet activities, we contribute techniques for recognizing scanning

date 2006	port #/ desc	# of uniq. src.	duration (T)	#AS	major OS: Windows percentage	coverage hit ratio	ex. scope (I)(/8)	Dshield scope (/8)	scope ratio (I)	ex. scope (II)(/8)	ex. #bots	#bots Dshield	#bots ratio	ex. total scans
08-25	1433	3080	9h	549	93.0%	1	2.2	1.6	1.4	4.6	3100	3139	0.99	5.8×10^8
11-26	2967	171	10m	31	99.4%	0.127	0.7	0.75	0.9	0.1	228	215	1.06	1.5×10^6
11-27	2967	258	3h:21m	46	99.2%	0.541	1.9	1	1.9	0.4	276	373	0.73	2.5×10^7
11-28	2967	286	5h	55	99.3%	0.610	1.7	1	1.7	4.0	305	331	0.92	2.7×10^7
07-23	5900	1344	8m	383	93.4%	0.503	0.7	0.8	0.9	0.9	2752	2712	1.01	8.6×10^6
07-29	5900	3037	23m	634	94.0%	0.928	0.8	0.8	1	0.9	3628	3696	0.98	3.4×10^7
10-31	5900	382	45m	92	71.2%	0.245	2.3	0.75	3.1	0.6	526	622	0.84	1.1×10^7
08-24	139-NBSS	2933	16h:15m	453	94.0%	1	1.4	1	1.4	3.5	2972	N/A	N/A	8.6×10^8
08-25	139-NBSS	1370	20h	361	94.9%	1	1.9	1	1.9	2.5	1383	N/A	N/A	5.5×10^8
08-29	139-NBSS	325	1h	32	89.5%	0.678	0.9	1.7	0.5	0.5	331	N/A	N/A	1.8×10^7
09-02	445 negsetup	2053	1h	138	88.6%	0.989	0.7	0.66	1.1	0.5	2184	N/A	N/A	5.6×10^7
07-26	445-wkssvc	1197	15h	269	97.4%	1	1.1	1	1.1	4.3	1221	N/A	N/A	3.1×10^8

Table 5: Extrapolation Results and Validation

date 2006	port #/ desc	# of uniq. src.	duration (T)	#AS	ex. scope (I)(/8)	Dshield scope (/8)	scope ratio (I)	ex. scope (II)(/8)	ex. #bots	#bots Dshield	#bots ratio	ex. total scans	ex. avg speed
08-25	1433	3080	9h	549	2.2	1.6	1.4	4.6	3100	3139	0.99	5.8×10^8	5.99
11-26	2967	171	10m	31	0.7	0.75	0.9	0.1	228	215	1.06	1.5×10^6	12.2
11-27	2967	258	3h:21m	46	1.9	1	1.9	0.4	276	373	0.73	2.5×10^7	7.81
11-28	2967	286	5h	55	1.7	1	1.7	4.0	305	331	0.92	2.7×10^7	5.24
07-23	5900	1344	8m	383	0.7	0.8	0.9	0.9	2752	2712	1.01	8.6×10^6	6.77
07-29	5900	3037	23m	634	0.8	0.8	1	0.9	3628	3696	0.98	3.4×10^7	7.19
10-31	5900	382	45m	92	2.3	0.75	3.1	0.6	526	622	0.84	1.1×10^7	8.17
08-24	139-NBSS	2933	16h:15m	453	1.4	1	1.4	3.5	2972	N/A	N/A	8.6×10^8	5.24
08-25	139-NBSS	1370	20h	361	1.9	1	1.9	2.5	1383	N/A	N/A	5.5×10^8	5.09
08-29	139-NBSS	325	1h	32	0.9	1.7	0.5	0.5	331	N/A	N/A	1.8×10^7	16.07
09-02	445 negsetup	2053	1h	138	0.7	0.66	1.1	0.5	2184	N/A	N/A	5.6×10^7	7.12
07-26	445-wkssvc	1197	15h	269	1.1	1	1.1	4.3	1221	N/A	N/A	3.1×10^8	4.47

Table 6: Extrapolation Results and Validation

strategies and inferring its global properties. Evaluation and validation with extensive honeynet and Dshield data demonstrate our techniques are promising to achieve the situational awareness which is crucial for current Internet security.

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

[5] The Honeynet Project. `http://www.honeynet.org`.

[6] BAILEY, M., ET AL. The internet motion sensor: A distributed blackhole monitoring system. In *Proc. of NDSS* (2005).

[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] CRANDALL, J. R., SU, Z., AND WU, S. F. On deriving unknown vulnerabilities from zeroday polymorphic and metamorphic worm exploits. In *Proc. of ACM CCS* (2005).

[12] CUI, W., ET AL. GQ: Realizing a system to catch worms in a quarter million places. Tech. Rep. TR-06-004, ICSI, 2006.

[13] KANNAN, J., ET AL. Semi-automated discovery of application session structure. In *Proc. of ACM IMC* (2006).

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

[15] KUMAR, A., ET AL. Exploiting underlying structure for detailed reconstruction of an internet scale event. In *Proc. of ACM IMC* (2005).

[16] MOORE, D., ET AL. Inferring Internet denial-of-service activity. In *USENIX Security* (2001).

[17] MOORE, D., ET AL. Code-red: a case study on the spread and victims of an internet worm. In *Proc. of ACM IMW* (2002).

[18] MOORE, D., ET AL. Inside the slammer worm. *IEEE Security and Privacy* (2003).

[19] PANG, R., ET AL. Characteristics of internet background radiation. In *Proc. of ACM IMC* (2004).

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

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

[22] RAJAB, M., ET AL. A multifaceted approach to understanding the botnet phenomenon. In *Proc. of ACM IMC* (2006).

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

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

[25] STANIFORD, S., ET AL. How to 0wn the Internet in your spare time. In *Proc. of USENIX Security* (2002).

[26] STANIFORD, S., ET AL. The top speed of flash worms. In *Proc. of ACM CCS WORM* (2004).

[27] VRABLE, M., ET AL. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. In *Proc. of SOSP* (2005).

[28] WEISSTEIN, W. E. Full Width at Half Maximum. `http://mathworld.wolfram.com/FullWidthatHalfMaximum.html`.

[29] WU, J., ET AL. An effective architecture and algorithm for detecting worms with various scan techniques. In *Proc. of NDSS* (2004).

[30] YEGNESWARAN, V., ET AL. Using honeynets for internet situational awareness. In *In Proc. of ACM Hotnets IV* (2005).

[31] ZOU, C. C., ET AL. Monitoring and early warning for internet worms. In *Prof. of ACM CCS* (2003).