Fingerprinting Custom Botnet Protocol Stacks

Steve DiBenedetto*, Kaustubh Gadkari*, Nicholas Diel†, Andrea Steiner*, Dan Massey* and Christos Papadopoulos*
*Department of Computer Science
Colorado State University
Email: $\{dibenede, kaustubh, steiner, massey, christos\}$@cs.colostate.edu
†Engineeerity, LLC
Email: nick@engineerity.com

***Abstract*—This paper explores the use of TCP fingerprints for identifying and blocking spammers. Evidence has shown that some bots use custom protocol stacks for tasks such as sending spam. If a receiver could effectively identify the bot TCP fingerprint, connection requests from spam bots could be dropped immediately, thus reducing the amount of spam received and processed by a mail server. Starting from a list of known spammers flagged by a commercial reputation list, we fingerprinted each spammer and found the roughly 90% have only a single known fingerprint typically associated with well known operating system stacks. For the spammers with multiple fingerprints, a particular combination of native/custom protocol stack fingerprints becomes very prominent. This allows us to extract the fingerprint of the custom stack and then use it to detect more bots that were not flagged by the commercial service. We applied our methodology to a trace captured at our regional ISP, and clearly detected bots belonging to the Srizbi botnet.**

I. Introduction

Malware, after infecting a machine, sometimes installs a custom TCP/IP stack for its network communication. The reason for a custom stack is to bypass and thus avoid detection by security software such as firewalls, monitoring network traffic in and out of the machine [4]. Examples of such malware have been seen in the wild, with the most well-known example being the Reactor Mailer, which is used in the Srizbi botnet, one of the most prolific spam botnets seen so far. While detecting spambots is often easy, typically by monitoring their activity and using anti-spam software that analyze content, bots employing custom stacks but engaging in other, less prolific activities are hard to detect. Moreover, the presence of custom stacks is a fairly strong indication that a machine is infected, so detecting them independent of malware activity is important.

Custom stacks can often be detected by remote OS fingerprinting, which typically works by examining TCP SYN packets sent by a particular machine and looking for OS-specific fingerprints. These may include parameters such as initial window size, initial TTL, MSS, TCP options, etc [12]. While most operating systems have fixed, easily determined signatures, custom protocol stacks may differ significantly. Such differences can be used by OS fingerprinting tools such as [3] to detect the presence of an unknown stack. Note that OS fingerprinting can also detect native stacks that have been tweaked. For example, if an application changes the initial window size or the MSS, these changes will appear in the fingerprint.

Detecting the spread of malware with a custom or tweaked stack fingerprint in a robust way can be quite hard. To detect spambots for example, one might try to extract a dominant fingerprint among known spammers, with the goal of creating a system that blocks requests that match the fingerprint. This approach might work if the malware is widely spread so that the fingerprint stands out from other, legitimate stack fingerprints. If, however, the malware deployment is at its initial stages or not very wide, it will be very hard to detect the new signature and react early by simply looking at statistics. Moreover, this approach makes it hard to detect custom or tweaked stacks, and may be prone to many false positives.

In this paper we propose a more robust way to detect malware using custom or tweaked stacks, that does not require them to be widely deployed and thus is well suited at detecting the outbreak of new such malware. We achieve robustness by detecting the simultaneous presence of multiple fingerprints from the same IP address. Our approach relies on the fact that when a host is infected with malware that installs a custom stack, the host will exhibit at least two fingerprints: the machine's native OS fingerprint and the custom stack fingerprint. By restricting our detection to IP addresses that exhibit multiple fingerprints we drastically limit the pool of suspects, which enables us to robustly detect custom stacks using simple statistics. In turn, armed with the newly discovered fingerprint we can use it to detect the malware in hosts that do not exhibit multiple fingerprints.

We demonstrate our approach using spam as an example. Note, however, that while spam is our driving example we believe that our methodology is applicable to other protocols, too.

In our example we first assume that we have, or can collect, a trace in our network with the packet headers of all TCP SYNs initiating email transfers (port 25). This will enable us to fingerprint all IP addresses sending email to our servers. Then, we assume we have a spam reputation list, namely a list that contains IP addresses that are strongly suspected of sending spam. Note, however, that we do not assume that the list is entirely accurate, up to date, or complete. The list could, for example, be obtained by one of the many commercial providers or be freely available. We only assume that the list has reasonable coverage so that many of the spammers in the list contact our network. While the possession of a reputation list may seem at first to be sufficient to reject spam, we note that such lists are notorious for being incomplete [11], and are just one aspect of spam mitigation.

Armed with our trace and reputation list we first isolate spammers with multiple fingerprints. We then analyze the fingerprints from these spammers looking for potential prevalence of a particular fingerprint among the fingerprint combinations seen. If such a fingerprint is identified among spammers, we mark it as a possible custom stack fingerprint and use it to detect spammers among all the IP addresses that send email to our network. We validate spammers identified this way using the methodology proposed by Schatzmann *et. al.* in [10].

Our results show that using our methodology we are able to detect the custom protocol stack used by Srizbi and significantly enhance our existing reputation list. In total, we are able to detect 30% more spambots. While the number of spambots detected with our approach is a fraction of the total number of spammers, we believe that we detect most, if not all, the Srizbi bots contacting our network. This can be linked to the fact that when our trace was collected the Srizbi botnet was in decline due to the shutdown of the McColo co-location service provider and a Microsoft Malicious Software Removal Tool (MSRT) update. As a result, the percentage of spam observed from the botnet was small. However, our methodology was still able to pull out its signature, thus demonstrating that our approach does not depend on infection peak of the malware.

II. Related Work

Previous work has explored the use of passive TCP fingerprinting in identifying spammer/botnet signatures. Other work has used passive monitoring to detect spammers. Plugins for popular open-source spam detection systems also use passive fingerprinting to detect spam [6].

In [10], the authors explore the use of flow-based metrics to detect spammers in the network core. By analyzing traces taken at a major national ISP, they find that 81% of mail servers have a spam load in the range of 70-90%. The authors classify flows as *rejected, failed* and *accepted*, based on the size of the flow (in bytes). By detecting rejected or failed flows, the authors were able to identify most spammers, with a false detection rate of 4.5%.

In [8], the authors propose a router-level spam filtering system. The system detects spammer fingerprints by using a passive fingerprinting tool (p0f [3]). These fingerprints are pushed to the router, which drops any packets that have known-spammer signatures. However, unlike our work, the authors do not explore the use of multiple fingerprints to detect spammers or botnets.

Both [4] and [9] use fingerprinting to analyze the Srizbi botnet and the effects of McColo shutdown on the Srizbi botnet/Reactor Mailer. They provide the p0f signatures that can be used to detect the Srizbi botnet. Unlike our work, however, this work is limited to the Srizbi botnet. Also, this work does not look into the use of fingerprinting to look at custom stacks.

III. Data Description

We used two types of data in this work. The first is a trace collected at our ISP and the second is a commercial reputation list. The company has a world-wide customer base and provides spam and malware protection, including a reputation list to its customers.

A. Trace Data

Our trace data was collected at a 1 Gb/s link between our regional ISP and one of its tier-1 providers. We note that our ISP has other links to Tier-1 providers as well as links to Internet 2 and NLR. Therefore, we are not capturing all traffic at our ISP since we cannot monitor all its links. However, our results show that we capture a significant amount of spam arriving on the monitored link.

We captured a 24H packet trace on June 18, 2009 using a machine equipped with an Endace [1] card, which ensured there was no packet loss. Our trace consists of packet headers and further limited to TCP SYN packets of SMTP traffic only (traffic to port 25). During the 24H trace period, we counted approximately 16.5M SMTP flows.

B. Reputation List

The second set of data we used came from a commercial reputation-based blacklist. We receive and archive an updated copy of the reputation list every 30 minutes. The reputation list is constructed from feedback from the customers and is therefore more effective for customers. Since our university is not a customer (we just receive the list, we do not contribute to its composition), the reputation list is less effective for us than the actual customers. We utilized a time window of the reputation list archives that corresponds to the date our trace was taken. Past estimates have shown that the reputation list captures only about 25% of the spammers talking to machines in our ISP, and therefore cannot be used as the sole means to mitigate spam. The daily list contains on the order of 100K spammer addresses. We find that a significant subset of these spammers contact our ISP's network (the exact numbers are given in the Results section).

IV. Fingerprint Methodology

It is often possible to remotely fingerprint an OS based on the signature of the system's TCP/IP stack. The stack is identifiable based on the specific values present in packet header fields, such as the initial TTL, window size, maximum segment size, and flags [3], [12].

There are two approaches to OS fingerprinting: (a) passive, and (b) active. In the passive approach, traffic is monitored and the values in TCP SYN packet headers are observed. A database mapping OS to specific SYN packet values is then used to identify the OS. Tools performing passive fingerprinting include p0f and satori [3], [5]. In the active approach tools, such as Nmap [2], send specially crafted packets to the host.

OS identification is then done based on the responses. On the other hand, passive fingerprinting is fairly easy and only requires access to a network trace with TCP SYN packets, or a passive network tap. While active fingerprinting can potentially be more accurate, it is, however, fairly heavy-weight requiring perhaps multiple packet exchanges between the fingerprinting tool and the target. It is also prone to generating abuse complaints.

In our work we use passive fingerprinting. Specifically, we use the p0f [3] tool for fingerprinting a trace we collected at our local ISP.

Given our goal of isolating the fingerprints of hosts running custom stacks, we first need to isolate spammers [1] with multiple fingerprints. We do this using our trace and the commercial reputation list as follows.

First, we begin by filtering our packet trace to isolate hosts sending mail through our network. We do this by extracting flows to port 25/TCP. We then intersect the set of IPs from the flows with the set of spamming IPs reported in our reputation list for the same time period our trace was collected. This produces a set of confirmed spammers contacting our network, but may leave out potential spammers that do not appear in the reputation list. Due to the way our reputation list is constructed, only spammers that contact the customers of the reputation service are included to the list. Thus, it is important to note that many of the IP addresses we see may still be spammers even though they do not appear in the reputation list (false negatives).

In the next step we fingerprint the port 25/TCP traffic coming through our network on a tier-1 provider link using p0f. The tool produces one or more signatures per IP address based on its fingerprint database. Multiple signatures typically occur due to custom stacks, which we are interested in detecting, or due to the presence of NATs, when more than one machine behind a NAT is fingerprinted, (thus more than two fingerprints are possible). Furthermore, hosts not identified by a fingerprint known by p0f produce a signature of UNKNOWN that explicitly states the values of the fingerprint. This allows us to produce statistics on similar, but unknown fingerprints.

Next, we classify addresses based on whether they appear in our reputation list or not, and whether they have one or more fingerprints. If an address appears in our reputation list we assume that this is a known spammer. Else, we cannot yet make a concrete determination, so we consider such addresses as potential spammers.

In the next step, we examine the multiple-fingerprint-known-spammer category looking for fingerprints that are prevalent. Since this category has multiple fingerprints we look for a fingerprint that appears prevalent in the ranking or is otherwise suspicious. The rationale is that machines running custom stacks maybe running different native operating systems, so the custom stack will be paired with different fingerprints. We also take into account other factors such as the likelihood that this is a genuine fingerprint based on what we know about the deployment of that particular OS. If such a fingerprint is identified, we look for other IP addresses in the potential spammer category (i.e., not in our reputation list) that have multiple fingerprints that include the suspicious fingerprint. We finally declare these IP addresses as identified spammers running custom stacks.

[1]For our purposes here, we define a spammer as an IP address that sends spam.

A. Limitations

Our current approach requires human intervention in terms of recognizing fingerprint combinations that look suspicious. We believe the process can be automated using advanced statistical or machine learning methods. For example, when a new custom stack is deployed, the prevalence of a new combination can be detected as an anomaly in the statistical distribution of signatures. This requires a system continuously monitoring OS fingerprints, which is not hard to do by a network operator.

Our system is subject to false positives due to DHCP, NAT or virtualized environments. We attempt to reduce these false positives by only examining traffic limited to one 24H period. While this is not sufficient to address cases such as wireless hotspots, these IP addresses are not likely to be flagged as spam anyway, due to the shortness of their lifetime. We are more interested in longer living addresses that are stable enough to be flagged as spammers. Thus, we feel that restricting our study to a 24H period is a reasonable compromise.

Finally, it is possible that our system may flag an IP address where a spamming fingerprint was detected, but the spammer is now long gone, and a legitimate server now occupies the address. This, however, is likely to be a dynamic IP address, which is typically filtered by spam mitigation systems as an address that should not send email. Thus, our system does not interfere with current policies. We do recommend that hysteresis is used to minimize any damage from this policy.

B. Validation

To validate our results we use the approach described in Schatzmann *et al* [10]. In this work, the authors explore the use of flow-based metrics to detect spammers in the network core. By analyzing traces taken a major national ISP, the authors classify SMTP flows as *rejected, failed* and *accepted*. They find that 97% of failed flows have less than 322 bytes and most rejected flows have between 400 and 800 bytes, while the size of the accepted flows depends on the email size distribution. By detecting the rejected and failed flows, the authors find that they can correctly detect most spammers, with a false detection rate of less that 4.5%.

We note that this approach is applicable to our trace because it was collected at our ISP at a tier-1 link. Traces collected at the edge of an organization may be less appropriate due to the combined local mail server knowledge this technique leverages.

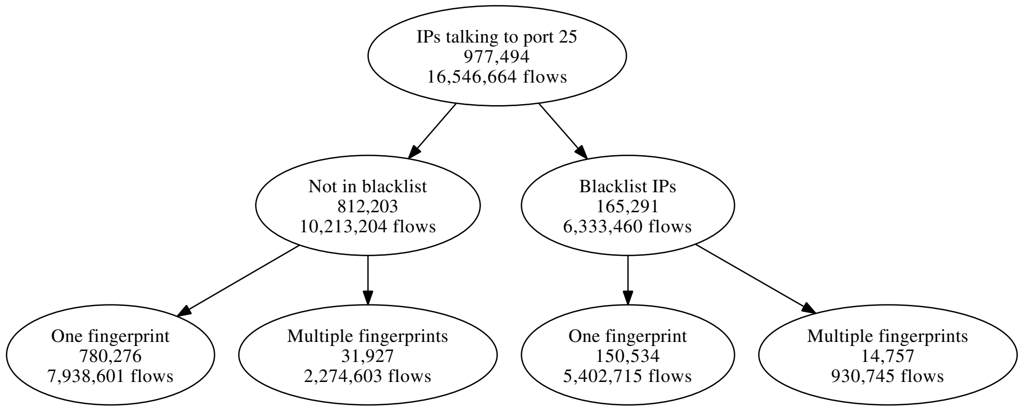

Fig. 1. Fingerprint hierarchy from our trace. On the right we see data from identified spammers in the reputation list. We also include flow counts for illustration purposes only. We do not draw any conclusions based on flow counts.

V. Results

In this section we present the results of our study. We begin by breaking down the trace into spammers and unknowns, according to our reputation list, and then further subdivide to those with single v.s. multiple fingerprints. The address and fingerprint breakdown is shown in Figure 1. The figure also shows the total number of flows the hosts at the different levels were responsible for.

We begin by noting that there were 977,494 unique IP addresses sending email to our network during the 24H period covered in our trace. This is shown at the top of the hierarchy. Out of these addresses we determined that about 165K are present in our reputation list marked as spammers (level two in the hierarchy, right side). This leaves about 812K addresses that are unknown. Note, however, the flow distribution between the two sets (6M v.s. 10M flows). Since many studies have shown that most email is spam [7], we conjecture that many of the unknown flows are spam, and thus many of the hosts on the left side are spammers. This leads us to conclude that our reputation list misses many spammers.

Looking down further in the hierarchy on the spammer side on the right, we see the breakdown between addresses with single v.s. multiple fingerprints. Note that the vast majority of addresses show a single fingerprint (150K or 91% of the spammer addresses), with only about 14.8K (9% of the spammers or 1.5% of the total) of the addresses showing multiple fingerprints.

We now focus on the 14,757 unique spamming IP addresses for our analysis. Recall that these are spammers according to our reputation list, and have multiple fingerprints. These spamming hosts produce a total of 2,598 unique fingerprint combinations. While this may sound like a lot, there is a heavy tail. Figure 2 shows the CDF of the fingerprint combinations, and Table I shows the ranking of the top 10 fingerprint sets when ranked by population. We see that the top-10 covers more than 55% of all the addresses in this category. Furthermore, the top ranked fingerprint set, (Solaris 2.5, Windows 2000 SP4, XP SP1+), accounts for 23% of the total.

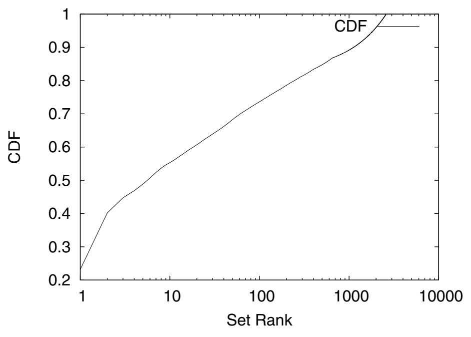

Fig. 2. CDF of spammer population size and fingerprint set rankings.

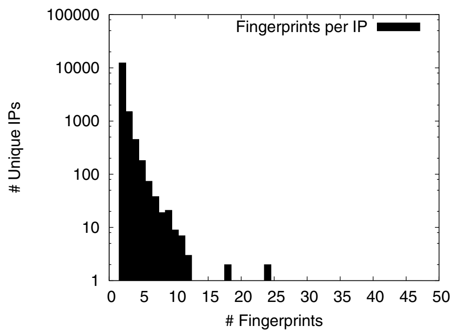

Fig. 4. Distribution of fingerprint set size for known spammers with multiple fingerprints.

The representative nature of this top-10 set is conspicuous when we examine the distribution of fingerprint set sizes (figure 4). Like most of the observed spammers with multiple fingerprints, our top-10 sets tend to contain two fingerprints. Note, however, that this cannot easily be generalized to every spammer in the Internet as we are limited by where spammers send their messages.

We are now ready to identify any suspicious fingerprints in Table I. Not surprisingly, Windows fingerprints dominate, with a Solaris 2.5 fingerprint being the sole exception. Upon closer examination, many of the Windows fingerprints appear

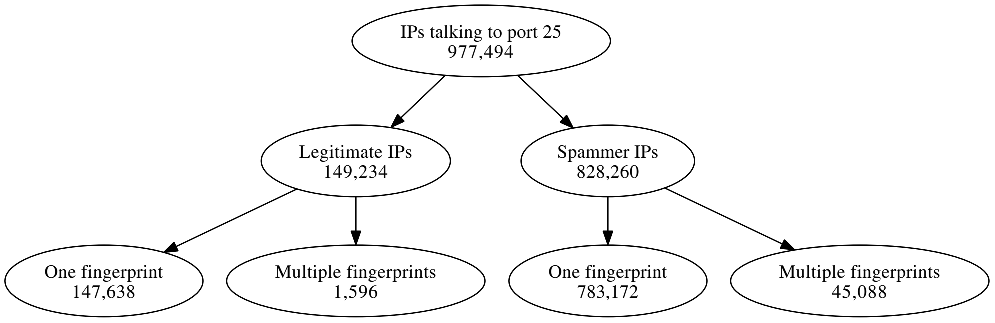

Fig. 3. Fingerprint hierarchy based on IP addresses identified by the methodology in [10].

# Unique IPs	Fingerprint Set
3,426	**Solaris 2.5** Windows 2000 SP4, XP SP1+
2,503	Windows 2000 SP4, XP SP1+ Windows 2000 SP2+, XP SP1+ (seldom 98)
675	**Solaris 2.5** Windows 2000 SP2+, XP SP1+ (seldom 98)
311	Windows 2000 SP4, XP SP1+ Windows XP/2000 (RFC1323+, w+, tstamp)
295	**Solaris 2.5** Windows 2000 SP4, XP SP1+ Windows 2000 SP2+, XP SP1+ (seldom 98)
273	Windows XP/2000 (RFC1323+, w+, tstamp) Windows 2000 SP2+, XP SP1+ (seldom 98)
245	Windows 2000 SP4, XP SP1+ Windows 2000 SP4, XP SP1+ [priority1]
192	Windows XP SP1+, 2000 SP3 Windows 2000 SP4, XP SP1+
142	Windows 2000 SP4, XP SP1+ Windows XP SP1+, 2000 SP3 (NAT!)
101	Windows 2000 SP4, XP SP1+ Windows XP/2000 (RFC1323+, w, tstamp)

TABLE I
TOP 10 FINGERPRINT SETS FOR SPAMMERS WITH MULTIPLE FINGERPRINTS.

to be fairly similar. For example the two fingerprints in the second entry in the table appear to differ by one byte in the initial window size (65535 v.s. 65536). We conjecture that this is the same fingerprint with minor tweaks in the protocol stack. Following our hypothesis, the third ranked entry appears equivalent to the first, and similar conclusions can be drawn for many of the remaining entries. Moreover, we find it surprising that a Solaris fingerprint would be as popular as Windows fingerprints. We therefore reach two conclusions: (a) the fingerprint which we suspect belongs to a bot with a custom stack is the Solaris 2.5 fingerprint, and (b) the OS that accompanies the Solaris fingerprint is very vulnerable to the bot with the custom stack.

Despite what we feel is strong evidence pointing to the Solaris 2.5 fingerprint as a custom spambot stack, we were still intrigued how this could be so. Solaris machines are not a very popular target for spammers due to their low numbers. Upon closer examination of the fingerprint and after consulting prior work including [4] and [8], [9], we realized that the Solaris 2.5 fingerprint was virtually identical to the fingerprint of the custom stack in the Srizbi botnet. The two works above define p0f fingerprints for the Srizbi botnet. After adding these into our database and re-running the fingerprinting process, we got a virtually 100% hit on the Solaris 2.5 addresses. We believe that this is conclusive evidence that our methodology did indeed identify a custom protocol stack, namely the stack in the Srizbi botnet, whose fingerprint is very close to the Solaris 2.5 fingerprint. Moreover, since we were able to drill down into the data (looking at spammers with multiple fingerprints), the process of identifying the fingerprint was fairly straightforward. Finally, the process gave us not only the custom stack fingerprint, but also identified an operating system that appears very prone to infection by this bot.

Armed with the custom stack fingerprint we now set off to measure the prevalence of this fingerprint and enhance our reputation list with more addresses of spammers. In order to validate the additional spammers we detect, we apply the methodology in [10] identify a more complete set addresses. Figure 3 shows the breakdown of these IP addresses. Most importantly, this approach flags nearly 97% of the addresses as spammers, which is consistent with other work [7].

We begin by examining the fingerprints of spammers already in our reputation list. Looking at the absolute count of spammers with multiple fingerprints where at least one signature matched the bot we find that 6,308 out of 14,757 (43%) do match. Looking at the single fingerprint spammers (third level on the right in Figure 1), 3,845 (2.6%) of the 150,534 spammers matched our fingerprint, which increases our original custom stack bot count (6,308) by more than 50%.

We next look at fingerprints of addresses not in our reputation list, but still exhibiting multiple fingerprints. Recall we argued earlier that we suspect many of these addresses to be spammers. From the set of 31,927 IP addresses we found 8,653 which *exactly* match the bot fingerprint, thus almost doubling again our current bot list. Since these are suspected spammers, we again resort to the spam detection methodology in [10] for validation. This test flagged 8,244 (95%) of these addresses as spammers.

Staying with the same set (addresses with multiple fingerprints not in our reputation list) we now relax our requirement that all fingerprints should match exactly and we look for fingerprints that simply contain the bot fingerprint. We find that 14,960 (47%) of the 31,927 IP addresses contain our bot fingerprint. Of these, 14,264 (95%) are validated by [10], which gives us confidence that these are indeed spammers. Thus, using our relaxed requirement in place of the strict fingerprint set matching increases our bot list by almost 2.5 times to 25,113.

Finally, we searched for the bot fingerprint among addresses with a single fingerprint but not in our reputation list. Of the 780,276 addresses, 33,921 (4%) matched our bot fingerprint and 26,176 of these (77%) validate as spammers. This seems consistent with the single fingerprint results obtained for addresses in our reputation list. Our bot list now stands at 59,034, a nine-fold increase from the original 6,308.

In addition to obtaining statistics about the bot, our results above show that our methodology can be used to improve our reputation list. Suppose for example, that our organization wanted to use the reputation list. The list alone flags 165,291 spammer addresses. Using our methodology we were able to flag an additional 48,881 or 30% more IP addresses, significantly improving the reputation list's effectiveness. Moreover, we showed that while the original list may be incomplete, this affects our final results very little.

Finally, we look at the effect of our fingerprinting on flows. Looking at Figure 1 again, we see that we would have blocked 912,114 spam messages. While this only represents a small portion of all observed flows (6%) during a period of 24H, having the spammer fingerprint means that we can continue to discover and block spam from these and more IP addresses in the future - the effect is cumulative. We also note the value of a fingerprint: while the IP addresses of spam bots may be prone to change due to dynamic addresses, a fingerprint remains virtually constant, and is thus more valuable than IP addresses when shared.

VI. Conclusions and Future Work

In this paper, we present an approach to detect IP addresses that use a custom TCP stack. While our driving example is spambots, we believe that the approach generalizes to other areas.

We focus on custom protocol stacks because this is a particularly dangerous mode of infection of a machine by a bot. Such a stack allows malware to bypass security software installed on the machine, such as firewalls. Such modes of infection, while hard to do since the malware requires raw access to the network, have been documented in the wild, and are readily detected by our approach. Windows machines, if updated, have closed raw access to the network, but there are still a huge number of machines running older or unpatched software, or other operating systems such as many flavors of Unix and Linux, that still allow raw access to the network.

While it is not clear that this will be a popular mode of attack in the future, our work raises the bar for malware that may chose to go that route. Malware has two choices: (a) install a custom stack ignoring the fingerprint and thus running the danger of being detected as we describe in this paper, or (b) try to emulate the fingerprint of the native OS of the infected machine. The latter, however, is hard because malware needs to be aware of the native fingerprint of the infected machine and emulate it, or restrict infection to a select set of machines. Either way, this makes life harder for malware writers and raises the bar.

Our approach is based on the observation that malware using a custom stack will exhibit at least two different fingerprints - the host's native OS fingerprint and that belonging to the custom stack. This allows us to drill down to the data focusing only on addresses exhibiting multiple fingerprints, which allows us to identify the custom stack fingerprint more readily.

Utilizing our method, a network trace and an incomplete for our site commercial reputation list, we were able to identify 48,881 spamming IP addresses that were not included in the commercial reputation list. By adding these IP addresses to the blacklist, our organization would be able to block 30% more spamming IP addresses.

In future work, we want examine the OS fingerprints of IP addresses across different protocols to automate the detection of custom stacks. This will eliminate the requirement that we see email from both the native and custom stacks of an infected machine. We could, for example, observe web or other popular traffic that uses TCP, and detect custom stacks in other applications besides mail.

References

[1] Endace. http://www.endace.com.

[2] Nmap. http://nmap.org.

[3] p0f. http://lcamtuf.coredump.cx/p0f/p0f.shtml.

[4] The rise and fall of reactor mailer. http://projects.csail.mit.edu/spamconf/SC2009/Henry_Stern/.

[5] Satori. http://myweb.cableone.net/xnih/.

[6] Spamassassin p0f plugin catches bot spam. http://advosys.ca/viewpoints/2007/07/spamassassin-p0f-plugin-catches-bot-spam/.

[7] Z. Duan, K. Gopalan, and X. Yuan. Behavioral characteristics of spammers and their network reachability properties. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 164 –171, 24-28 2007.

[8] H. Esquivel, T. Mori, and A. Akella. Router-level spam filtering using tcp fingerprints: Architecture and measurement-based evaluation.

[9] T. Mori, H. Esquivel, A. Akella, and A. Shimoda. Understanding the world's worst spamming botnet. Technical Report 1660, University of Wisconsin - Madison Computer Sciences Department, July 2009.

[10] D. Schatzmann, M. Burkhart, and T. Spyropoulos. Inferring spammers in the network core. In *PAM '09: Proceedings of the 10th International Conference on Passive and Active Network Measurement*, pages 229–238, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] S. Sinha, M. Bailey, and F. Jahanian. Shades of grey: On the effectiveness of reputation-based "blacklists". In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 57 –64, 7-8 2008.

[12] M. Zalewski. *Silence on the Wire: A Field Guide to Passive Reconnaissance and Indirect Attacks.* No Starch Press, 2005.