COMPUTERS & SECURITY 29 (2010) 45–58

available at www.sciencedirect.com

journal homepage: www.elsevier.com/locate/cose

Computers & Security

On the detection and identification of botnets

Alexander K. Seewald [a,*], *Wilfried N. Gansterer* [b]

[a] *Seewald Solutions, Leitermayergasse 33/24, Vienna, Austria*
[b] *University of Vienna, Research Lab Computational Technologies and Applications, Vienna, Austria*

ARTICLE INFO

Article history:
Received 13 March 2009
Received in revised form
7 July 2009
Accepted 31 July 2009

Keywords:
Botnet
E-mail spam
Traffic analysis
Machine learning
IT security

ABSTRACT

We develop and discuss automated and self-adaptive systems for detecting and classifying botnets based on machine learning techniques and integration of human expertise. The proposed concept is purely passive and is based on analyzing information collected at three levels: (i) the payload of single packets received, (ii) observed access patterns to a darknet at the level of network traffic, and (iii) observed contents of TCP/IP traffic at the protocol level.

We illustrate experiments based on real-life data collected with a darknet set up for this purpose to show the potential of the proposed concept for Levels (i) and (ii). As darknets cannot capture TCP/IP traffic data, we use a small spamtrap in our experiments at Level (iii). Strictly speaking, this approach for Level (iii) is not purely passive. However, traffic moving through a network could potentially be analyzed in a similar way to also obtain a purely passive system at this level.

1. Introduction

Undoubtedly, spam is a problem of global impact that keeps getting bigger. Already in 2001, according to a study undertaken for the European Commission (Gauthronet and Drouard, 2001), Internet subscribers worldwide wasted an estimated 10 billion Euro per year just in connection costs due to spam. Moreover, the quantifiable damage caused by successful phishing attempts was estimated to be millions of Euros in Germany in 2006.[1] The total damage caused is even higher, since these figures do *not* yet account for loss of reputation and reduced customer trust. The economic impact is only part of the problem – waste of time, resources and the gradual erosion of trust in e-mail communication must also be considered significant side-effects of the spam deluge.

Spam filters have become reasonably effective by now (see, for example, Seewald, 2007; Cormack, 2006). However, new challenges, such as new forms of image spam and audio-based spam may lead to decreasing performance of today's most widely used Bayesian spam filters. More generally, history has shown that spammers have adapted again and again to effective adversaries and countermeasures. Most of the widespread countermeasures against spam are *reactive* in nature and thus this ''arms race'' is very likely to continue in the future. In particular, it is a big challenge to reduce the overhead and waste of resources caused by spam (Janecek et al., 2008; Gansterer et al., 2007).

But spammers are already diversifying their operations, e.g. with *phishing* – faking entire online banking websites to harvest login and transaction data from a large userbase – which motivates the investigation of methods specifically targeted for this challenge (Gansterer and Pölz, 2009). Recently, a widespread vulnerability in today's DNS system has been detected which would have allowed spammers to fake DNS entries and

---

* *Corresponding author.*
E-mail addresses: alex@seewald.at (A.K. Seewald), wilfried.gansterer@univie.ac.at (W.N. Gansterer).

[1] http://www.at-mix.de/news/1226.html.

0167-4048/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2009.07.007

thus forward legitimate banking sites' URLs to the spammers' sites. This would have allowed them to bypass more costly harvesting efforts which depend on naivety of users, for example, based on phishing messages with embedded links to the faked sites. Although the vulnerability has been partially fixed in the meantime (US-CERT), sufficiently many vulnerable servers still exist for this to be a continuing threat.

Another alternative source of revenue comes from *Pump-and-dump* spam, which is an approach to link spamming directly to the market. Sending out millions of spam mails which aim to boost a specific company's share value leads to a small transient peak or following valley in share value (once people notice they have been fooled, they are desperate to sell). These fluctuations are used to make money directly via leveraging in form of call or put options.

Both approaches make the spammers' revenue independent of companies paying for their services. Recent news about systematically stolen user data of large companies, to be used, e.g., for identity theft and credit card fraud, is another indication for that. By diversifying their sources of revenue, spammers will in time become more independent and find new business models.

Most if not all of today's spam is sent via large networks of captured machines which are under direct control of spam operators. According to SpamHaus, about 300–400 spammers are responsible for 80% of world-wide spam (ROKSO Database, 2009). These large networks have been named *botnets*, from software *robot* networks. One of the largest spamfilter companies, SpamHaus,[2] has estimated that already in 2004, 70% of spam was sent out via such networks (NISCC, 2005). Computers without the latest security updates attached to the internet may be infected rapidly depending on the security of their operating system. Infection may happen via known security holes, by users clicking or only viewing infected mails or via surfing past an appropriately prepared webpage, using browser or browser plugin vulnerabilities. Even secure computers may be prone to *zero-day attacks*, which rely on the delay between the publication of a weakness and the availability of a patch to correct it. There are some indications that botnet operators invest in R&D to find specific zero-day vulnerabilities, aiming at exploiting them at leisure.

At least a million bots are known to exist according to the US FBI and public trackers like ShadowServer,[3] and the true number is likely to be much higher. Another worrying development is that although the security of newly deployed systems is clearly increasing, the number of bots is still growing at an exponential rate. This indicates that the potential to build new and larger botnets is not exhausted yet. A bot, being subverted, can harm the user in any number of ways: by capturing passwords, PIN and TAN numbers for telebanking, rerouting their online banking website to a page under the spammer's control, by sending out spam or malware, recruiting new bots, and so on.

By distributing their workload over several hundreds or thousands of machines, spammers are able to use combined bandwidth of gigabits per second to send out spam rapidly. But bots are not only used to send out spam. Different bots exist which attack large parts of the internet infrastructure via distributed denial-of-service attacks (DDoS), install backdoors and spyware to harvest password and login information and allowing direct access to all files accessible from the machine, defraud ad networks with pay-per-click systems like Google AdSense by faking millions of clicks from hundreds of thousands of machines, crack weak passwords in a brute-force manner by using the botnet as a computational grid to distribute workload among a large set of machines, or create specialized peer-to-peer networks for obfuscating the operator's point of origin.

The ease with which it is possible to create new malware variants for infecting machines with bots has human analysts at a great disadvantage. While botnet operators are able to automatically generate new malware variants in unprecedented numbers, out of necessity the anti-virus community has become focussed on just *detecting* malware. It clearly has become infeasible to analyze manually what each of the thousands of new variants of malware appearing each and every hour is capable of. Botnet software nowadays is capable of automatically downloading updates from the internet to stay ahead of anti-virus detection, so malware analysis would have to be done in near real-time as well. However, with only detection by antivirus software, it becomes almost impossible to fully understand the functionality of a given malware *before* it spreads on the internet. Recent approaches to characterize malware automatically such as (Seewald, 2008) seem promising, but currently do not scale sufficiently well and should still be considered open research problems.

To summarize, botnets constitute a major threat for the internet community and attack simultaneously from several directions:

- they disable or hamper internet infrastructure even during their *normal* operation,
- they enable or facilitate cybercrime such as phishing, pump-and-dump spams, click-fraud, harvesting private user data for identity theft and fraud, etc., and
- they reduce the chances of capturing spammers by obfuscating the point of origin.

1.1. *Objectives*

The objectives of this paper can be summarized as follows. As the botnet operators have resorted to extensive automation – in the generation of new malware bot variants, in the deployment, command and control of these variants, and in diverse profit-gaining activities – it is necessary to automate the detection, identification and tracking of botnets as far as technically feasible. As the botnets operate on a global level beyond national boundaries, any countermeasures also have to operate on a global level. As the botnet operators continually update and extend their systems, identification systems also have to be continually updated and extended. To be able to combat botnets proactively and reduce their effectiveness for any kind of usage – including spam – we must first learn more about them.

In this paper, we pursue a comprehensive approach by distinguishing three levels at which botnet analysis may take place and investigate various aspects and open issues on each

---

[2] www.spamhaus.org.

[3] www.shadowserver.org.

of these levels. A central focus is on how to replicate human expertise – in the identification of specific bot types, specific botnet types and specific SMTP traffic patterns – by appropriate machine learning systems, which enable natural update and extension of our systems by simple retraining. A widespread worldwide sensor network using all three levels identified here should be able to achieve all our objectives.

The final goal must be to characterize specific botnet commands to enable the identification of the exact perpetrators regardless of all obfuscation tactics, to facilitate and in some cases enable swift action by law enforcement, and to develop proactive countermeasures. This remains *the* challenge for the future, incorporating not only formidable technological but also major legal issues.

2. Background and related work

Two types of approaches can be distinguished in the area of botnet detection and botnet identification: *active approaches* and *passive approaches*. Active approaches entail defanging bots or writing specialized bots to simulate the behavior of a real bot. This enables the bot to log on to a control center and watch the activity there, and usually enables direct interference with the operation of the botnet. However, such activities can be detected by the botnet operator and thus will likely lead to countermeasures. Silent bots of this kind are currently ignored, but this may change in the future. Passive approaches are those that cannot be detected by any means, because there is no flow of information back to the botnet operator. They work with more subtle information sources than active approaches, and do not allow direct interference with the operation of the botnet. Rather, the passive approach intends to be used as a tool to study botnets and serve as an early warning system.

Active approaches comprise all kinds of techniques which make the botnet operator directly or indirectly aware of observation. Examples for active sensing include capturing bot malware and deactivating its malicious parts (i.e., defanging bots), and subsequent analysis of the commands sent and received over the network during prolonged execution of the now harmless bot. Defanging a bot is a complex task that at present can only be done through a painstakingly manual process. While defanging need not be done when bot communication is unencrypted, it is essential otherwise. Honeypots and honeynets also use active methods to lure malware, e.g., by simulating known vulnerabilities. While active approaches may seem promising at first sight, as mentioned before, they have the big disadvantage that they can easily be detected. Once this happens, botnet operators will inevitably adapt and circumvent any measures taken against botnets. If focussed, aimed and applied in a short time period, active approaches might be helpful for a once-in-a-lifetime shot at botnet operators, but this is not likely given the diversity of approaches currently followed and the lack of global supervision for deployed defense measures. The strong manual component also make analyzing the majority of malware infeasible, and therefore this approach is used only to track the most prevalent bots.

Passive approaches analyze traffic which the botnet generates and secondary effects of its usage (e.g. broken packets resulting from a distant DDoS attack). An example of a completely passive approach is the use of darknets in packet capture instead of using a machine which appears vulnerable (low interaction honeypot), or actually is vulnerable (high interaction honeypot) for attracting botnet attacks, malware, or spam. Both honeypot types can be detected by botnet operators: the first one, for example, by testing the functionality of the emulated service which is often very incomplete, and the second one, for example, by fingerprinting the operating system after it has been successfully compromised and then excluding known honeypot configurations. The analysis of existing traffic, e.g., from Tier 1 network as in Karasaridis et al. (2007), without introducing significant latency, also is a purely passive approach in this sense. Purely passive methods are more complex to set up, test and adapt, but have the big advantage that their detection is completely impossible (for example, in the case of darknet data capture without SYN packet reply, where the capturing machine is indistinguishable from an unused IP address).

The approach investigated in this paper is based on passively observing network traffic arriving in a *darknet*, i.e., an IP address range which is not used for any active machines. The term "darknet" is sometimes also used in a different meaning, namely for a closed private network of computers, for example, used for file sharing. At Level 3 of our approach (see Section 3.3), we needed to augment this data with received spam mails from our spamtrap. While the analysis presented there is thus not a passive system, the same data can be collected in a passive manner by analyzing SMTP traffic moving through a network.

Contrary to intrusion detection and prevention, which is concerned with analyzing attacks and cutting attackers off before their attack has succeeded, our approach is concerned with analyzing passively observed network traffic to unused IP ranges. The quantity and quality of our traffic data are significantly different from the one seen by intrusion detection and prevention systems.

In fact, in preliminary experiments we found that standard intrusion detection and prevention tools are not suitable for our traffic data as they are geared towards active attack patterns which simply do not appear in our data. We considered the following tools.

- **snort** is an open-source tool and the de-facto standard for intrusion detection and prevention. www.snort.org
- **p0f** is a passive operating system fingerprinting tool. http://lcamtuf.coredump.cx/p0f.shtml
- **fl0p** is a passive L7 flow fingerprinting tool. http://freshmeat.net/projects/fl0p
- **honeysnap** is a tool to analyze significant event in recorded packet files. https://projects.honeynet.org/honeysnap
- **nmap** (Network Mapper) is an active scanner for vulnerabilities. http://nmap.org
- **xprobe2** is an active operating system fingerprinting tool. http://xprobe.sourceforge.net

Reference data was provided by Marshal Ltd.[4] from their semi-automated TRACE system. We matched their IP and timestamp data to ours, taking into account that dynamic IPs

---

[4] http://marshal8e6.com.

are a reliable indicator of the underlying machine only for a short time period.

We will now present related research, grouped into active and passive analysis approaches as outlined above.

2.1. *Active analysis*

In Dagon et al. (2006), a model of botnet propagation using time zones is proposed. It is based on the assumption that potential bots – being machines of private internet users – will be switched off during the night. They used a DNS redirection technique to redirect known IRC Command & Control servers to IP addresses under their control. Over a six month period, they redirected 50 botnets. Their diurnal model of botnet prediction works reasonably well at predicting botnet population growth (better than basic Susceptible-Infectious-Removal (SIR) models from epidemiology (Daley and Gani, 1999)) and may be used to prioritize botnets by propagation potential. However, their type of data collection is a slow manual process and very disruptive to botnet operation and therefore easy to detect. Obviously, it would be possible to combine alternative data collection approaches (such the one discussed in this paper) with their approach.

Various graphs of botnet connectivity are analyzed in Dagon et al. (2005), and a taxonomy of botnets based on topological structure is proposed, with appropriate targeted responses in each case. While their test case is trivial, their work has some merit as a guideline to attack future botnets with more robust communication architecture than a central C&C server (as in IRC).

2.2. *Passive analysis*

Dhamankar and King (2007) propose a framework to guess protocol types based on classical traffic analysis, without reference to the content transferred. Such a system may be able to recognize even encrypted botnet communication traffic for peer-to-peer botnets by its characteristics. However, steganographic approaches would still remain undetected. This would be similar to our Level 2 approach discussed in Section 3.2. A set of relatively simple features easily computed from a traffic stream in real-time was able to distinguish normal UDP traffic, Skype traffic, NetBIOS and Real-time Transfer Protocol (RTP, used for Voice-over-IP). Similar to their results, we also found that simple features work reasonably well.

Collins et al. (2007) propose a network quality measure based on spatial and temporal *uncleanliness* (i.e. the proportion of bots among all IP addresses measured along time and space (subnet) axis) that tries to predict future botnet addresses based on previously known botnet addresses. While interesting in principle, the work is hampered by not properly addressing dynamic IP addresses, which influence both temporal and spatial uncleanliness.

Ramachandran et al. (2006) propose to use DNS blacklist queries to detect bots who are not yet on the blacklist. Their detection is a purely passive approach, and even their countermeasures are covert and may not be detected by botnet operators for some time. It should be noted that in the meantime most DNS blacklists have resorted to only return blacklist hits to query hosts with DNS MTA entries (i.e. verified mailhosts). As such, DNS blacklist lookups can no longer be used by most bots as they will always return NXDOMAIN. Queries may still be sent until botnet operators have adapted to this change. In any case, we found that more than 97% of our bots are not on blacklists, so explicit lookups are perhaps no longer needed. The high prevalence of dynamic IPs may also harm DNS blacklists' effectiveness (see Section 4).

A more promising approach has been discussed in Karasaridis et al. (2007), who proposed using traffic summaries of data flow collected on a large Tier-1 ISP network. In our terminology, their approach would also be classified as a Level 2 approach (see Section 3.2). They propose an anomaly-based passive analysis algorithm, which has been used to detect IRC botnet controllers with less than 2% false positive rate. This still needs to be combined with a secondary more detailed analysis to further reduce false positive rates, as otherwise legitimate services might be tagged as botnet controllers. However, very few institutions are likely to get direct access to a Tier-1 network, which strongly hinders the further development of this line of research. Non-centralized communication between bots was not considered.

2.3. *Approaches based on darknets*

Bailey et al. (2005) report results from a multi-year project, *Internet Motion Sensor* in darknet traffic analysis, for an almost passive distributed darknet. Their system is not completely passive, as they sent out ACK packets in response to TCP connection attempts via SYN. This increases available information significantly, but could potentially be detected by spammers. They demonstrate the usefulness of the darknet in tracking the spreading of the Blaster Worm, the Bagle backdoor scanning and the SCO denial of service attacks in 2003 and 2004. In each of these cases, specific analysis was done to detect these events while we focussed on learning such detection models. Their focus was not an early warning system nor botnet detection, but rather to track and analyze threats to internet infrastructure at a slower timeframe. This darknet has been discontinued since then.

Rajab et al. (2006) analyzed botnet behavior based on a distributed darknet which they constructed and used for measuring botnet activity. Whereas their emphasis was on measuring botnet traffic, in this paper we intend to go one step further towards identifying and classifying botnet activities based on machine learning techniques.

2.4. *Synopsis*

In this paper, we investigate a mostly passive botnet defense approach which proceeds at three hierarchical levels. Based on network traffic data observed in a darknet and on reference data, machine learning techniques are applied for automatically detecting and identifying existing botnets.

For clarity, we introduce our general framework in Section 3. Our approach is first summarized in general terms without too many details or concrete examples. In Section 4, we discuss concrete realizations of this framework for spam botnets, whose main activity is to send out spam e-mail. This includes diagrams of systems' architecture as well as pseudo-code for important algorithms. In Section 5 we provide an outlook on the large-scale technical realization of our framework before summarizing our findings in Section 6.

3. A three-level approach

There are several meaningful levels at which the identification of botnets can take place.

Level 1: At the level of a *single packet*, we can try to distinguish whether its payload is malicious or spam, whether it corresponds to a remote check for vulnerabilities, or whether it follows unusual conventions with respect to flags and TCP options.

Level 2: At the level of *network access*, we can try to distinguish, for example, the pattern of access to a given darknet (sequential or random access, time between accesses to a single or multiple darknets, distribution of access per type and time etc.), pattern of communication activity between bots (in case of peer-to-peer botnets) or with the command center (in case of centralized botnets) even without analyzing the content. *Traffic analysis* in the usual military meaning is exactly this.

Level 3: At the level of *TCP conversations*, we can try to distinguish legitimate TCP conversations from illegitimate TCP conversations – e.g., SMTP connections sending spam and malware vs. those sending regular e-mail, SMB transactions known to trigger vulnerabilities in the SMB protocol, etc. Contrary to the previous Level 2, here we also analyze the content of many packets and entire conversations and are thus limited to unencrypted or weakly encrypted communication channels.

While the activities at each of these levels will be different, it makes sense to view these levels as incremental. For example, feature descriptors for single packets which allow for identifying spambot types – as in our approach later on – might be reused to compute summary statistics on the level of network access over a large set of packets, and even on the third level of entire TCP conversations. Patterns in network access can be reused to speedily prioritize suspect TCP conversations, e.g., when one machine initiates connections to the same destination port of many other machines. The reuse of patterns and data from lower to higher levels makes a machine learning approach for all levels feasible. Combined with feedback on known good and bad examples on each level, a learning passive botnet detection and tracking system no longer hampered by restrictions of human processing speed becomes a real possibility.

As mentioned before, the big advantage of our approach is that it will remain undetected by botnet operators. Honeynets, the analysis of malware in sandboxes, and other active techniques (such as IRC detectors which play bot and record botnet commands, and can even be used to take down botnets) can potentially be detected, ignored and even subverted by botnet operators. Not only virtual sandboxes are vulnerable as no simulation is 100% accurate – even physical sandboxes are vulnerable, as the necessary reuse of operating system images allows to determine fingerprints for known sandbox environments or the special-purpose hardware to replicate harddisc images on startup, and to integrate these fingerprints into new bot versions. On the other hand, such a functionality could also be used to immunize certain operating system variants against the infection with bots. Additionally, the active approach needs deep knowledge of bot malware which is not available for most circulating variants.

3.1. *Level 1: analysis of single packets*

The challenge in single packet analysis is to get the most information out of a single packet sent out from a bot-infected machine. Combined with darknets, which can be converted into silent detectors by traffic forwarding, it is possible to construct completely passive detectors for bots.

Alternatively, instead of using a darknet, traffic moving through a network could be analyzed on packet level with similar methods. Robust models of botnet-specific packets may be built to detect any botnet activity within the system, based not only on traffic concerned with propagation (e.g., packets with malicious payload), but also on communication between bots and malicious bot activity addressed to machines within or without the network. Nevertheless, the available data which can be obtained without any risk of being detected by botnet operators is severely limited. Concrete examples of a Level 1 approach are given in Sections 4.1 and 4.2.

3.1.1. *Open issues*

One major concern is that botnet operators could potentially randomize packet payloads, and use open-source TCP stacks resp. the operating-system TCP stack. This would limit the applicability of fingerprinting and restrict the scope of our proposed passive darknet approach to UDP packets. It would then be necessary to switch to more active approaches, e.g. sending a SYN packet as response to a TCP SYN packet which asks for establishing a connection packet. This strategy has been reported by Bailey et al. (2005) to increase available data significantly, but also increases the chance of being detected. Clearly, receiving just a single SYN packet from every IP address within a large subnet, and never receiving anything else, makes this darknet approach highly suspicious to appropriate darknet detectors. This may be the reason why previous darknet approaches were not very successful.

Another option is to analyze packets *in transit*, i.e., when they are traveling to other vulnerable machines, at firewalls or at the network routing level. The same methods could be used, but far more care would be required to distinguish good packets from bad packets. Whereas in darknets almost every arriving packet can be assumed to come from a malicious source, this is obviously not true here. In fact, changes in machine behavior as evinced by sending unusual packets after receiving unusual packets may even be used to automatically train such a system, but this is almost a Level 2 approach (see Section 3.2). Karasaridis et al. (2007) proposed such an approach, but they only analyzed very simple summary statistics of traffic data rather than doing a full analysis on the packet level. This would be preferable to above active approach as it still retains the passive nature of the system.

A single packet only offers limited information. Still, analysis on this level works surprisingly well and should be more often implemented (see Section 4). The lack of data on correlations between spambot types and the corresponding packets sent out is an issue, though. Here, the availability of automatically generated data (e.g. via sandboxes) from known spambot types could create sufficient training data to test more complex approaches.

Another more long-term approach, especially in the view of the potential vulnerability of sandboxes, would be a larger darknet for data collection, or even analyzing a significant portion of packets in transit.

3.2. *Level 2: analysis of network traffic*

The challenge in network traffic analysis is to correlate patterns of connectivity (such as who sent a packet when and where) with botnet activity. As an example, the obvious fact that all bots within a single botnet can be expected to show the same behavior in very short time frames can be used to identify botnets quite reliably with minimal information about the actual data sent out. In military applications, even with strong encryption that is practically unbreakable, knowing who communicated with whom gives a strong strategic advantage. Botnets need communication and central control from a single source – the human operator, who needs to be at one definite physical location, but may obfuscate his position through a variety of techniques.

Bots are only efficient to use when they access many different machines. Thus, a sufficiently large sensor network which records source and destination IPs and ports of network traffic would be sufficient to detect probable bots with high accuracy, although a significant false positive rate would have to be expected. As mentioned in Section 2.2, Karasaridis et al. (2007) presented just such an approach, and reported a false positive rate of about 2%. This can clearly be done as a purely passive analysis without any tip-off to the botnet operators traced. A Level 1 model which tags packets corresponding to known spambot types could be used to reduce computational load and latency interference. A concrete example of a Level 2 approach is given in Section 4.3.

3.2.1. *Open issues*

A major problem is that most bots run from dynamic IP addresses, which change at relatively short intervals. A long-term record of activity based on IP addresses alone is thus useless for detecting longterm bot activity. Still, the observation that the same activity pattern is shown by two different IP addresses within a short period of time might help to tie the dynamic IP addresses together and thus generate a long-term partial model of dynamic IP address change for a certain subnet, thus making the identification of IP addresses much easier. It should be noted that internet service providers can of course trivially track the dynamic IP addresses of their own users, so a collaboration with internet service providers would solve this issue.

At this level, data privacy issues are a major concern. The techniques used to track dynamic IP addresses may, for example, be used to track specific machines and thus specific users over long time periods. Care must be taken to handle this data in accordance with legal restrictions. This is a major problem as there are no world-wide standards available. Again, for a proactive analysis, indications from Level 1 that packets exchanged can be traced to specific spambots may be helpful to convince law enforcement agencies to actively support such techniques, and to follow up on their results. As this is not a technical problem we will not discuss it further.

The target population of botnet attacks is likely to be quite large, thus getting a comprehensive overview (coverage) is likely to be a major problem, and the benefits are also widely distributed. An alternative to a target-oriented analysis of botnet attacks, especially in the light that identification of dynamic IP addresses is trivial in this case, would be to analyze outgoing traffic from internet service providers. This has not only the advantage that ISPs could arrange the legal details with their own users in the context of appropriate contracts, and thus keep their internal networks free of bots without legal problems, but should also be considered in the context of the EU Data Retention Directive 2006/24/EC (Stampfel et al., 2008a,b; Stark et al., 2008). The benefits of a botnet-free ISP service are clear and immediate, and such an approach would resolve all issues mentioned here, including legal concerns.

3.3. *Level 3: analysis of TCP/IP traffic*

In addition to just analyzing traffic patterns without respect to contents (which is already quite effective) it is also possible to analyze patterns in content, treating, e.g., TCP streams and timing of responses as features for a learning system. This provides a lot of data not available to simple traffic analysis.

While it is theoretically feasible to build a fully passive traffic content analysis system, the practical challenges are enormous. First, only very fast learning systems are able to cope with very large traffic volumes without slowing traffic down significantly, which is undesirable as it increases the chance for detection. Downsampling the data may be helpful here, but may reduce coverage. Second, tracking connections means a high memory load only for keeping connection data until definitive connection closure, timeout or a specific event which serves as checkpoint. This is feasible for small networks, as TCP connection firewalls have proved, but cannot be done at a scale similar to what would be feasible at Level 2. Selection mechanisms might be needed to reduce the traffic which needs to go through such a system. A concrete example of a Level 3 approach is given in Section 4.4.

3.3.1. *Open issues*

Far more so than in Level 2, data is very sensitive here. This is because content of TCP connections will include, among others, private e-mail data, social security-, bankcard- and credit card-numbers, and personal data from literally billions of innocent people. An automated system which analyzes just a small part of this data is a large operational risk if it can be subverted, controlled or even just inspected by outside parties. While the technical difficulties could be overcome, we believe that such a system cannot be run outside government control with appropriate checks and balances. Although it would prove the most effective weapon against botnet operators, the false positive rate and potential for misuse are something to consider very deeply in this case.

One possible use case might be in law enforcement, provided that a set of potential botnet operators has been identified with simpler systems. Automatically tracking the list of potential suspects for a few days until one of them can be linked to a known botnet operator might then be an acceptable use for such a system with manageable risks, provided data collected from innocent people is discarded as soon as possible.

4. Experiments

In this section, we discuss concrete realizations of the abstract three level approaches introduced before for a special type of botnets, for which most reference data could be obtained: spam botnets, whose main activity is to send out spam e-mail.

As proof-of-concept for Level 1 (single-packet analysis), we have built and deployed a purely passive darknet packet analysis system which identifies spambot type by header, contents and flags of a single incoming packet. Fig. 1 shows the systems' architecture. All incoming packets are recorded by tcpdump and analyzed once a minute by trackSpambots, which looks up the rDNS entry for each IP, and also queries the SpamHaus XBL blacklist. All results from these queries as well as the complete IP traffic by tcpdump are stored for later analysis. *trackSpambots* is also responsible for determining spambot type based on the pretrained model described in the next section.

As not even SYN packets are returned by our darknet, TCP conversations just yield one SYN packet, while ICMP and UDP packets yield a bit more information. Surprisingly, even these relatively small packets yield enough information to identify most spambot types quite reliably. We think this is because botnet operators have not yet identified these obvious vulnerabilities in their systems.

Our darknet currently contains 256 IPv4 addresses – equivalent to an /24 subnet – which are distributed among four different /26 subnets (A–D). No non-trivial traffic was obtained from the IP v6 /48 subnet during the whole duration of the project.

As feature set we use protocol type (ICMP, TCP, UDP), TCP option types and their order, ICMP and UDP payload data (represented as 2-gram word vector (16 bit per token)), TCP and UDP destination port, and the IP Don't Fragment field. Because of the passive nature of our darknet, we could not capture full SMTP conversations – only the connection attempt was registered via TCP SYN packet. Spam mail contents were taken from a separate small spamtrap built in a different project.

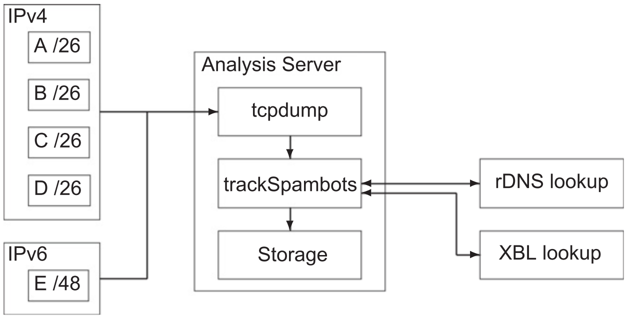

Fig. 1 – Systems' architecture for Level 1 and Level 2 (SpambotTracker).

In preliminary experiments we found that standard intrusion detection tools, such as snort, p0f, fl0p, honeysnap, ettercap, nmap and xprobe2 (cf. Section 2), were unsuitable for our analysis – partially because their operation mode is active (i.e., it involves sending packets which may be used to detect our presence), and partially because their patterns only fit known attacks, which the darknet effectively prevents (except for attacks purely via UDP) as no TCP conversations can be successfully set up. We therefore had to build our own feature descriptors.

Simple single packet analysis already allows a test of effectiveness of current DNS blacklist systems under reasonable assumptions. We assume that each packet sent to the darknet has been sent from a bot-infected machine. This slightly overcounts systematic port scanning activities by the administrators of our darknet, which however happen from a single IP, can be clearly identified, and can therefore be neglected.

We found an overlap of only 2.75% with the Spamhaus XBL, which specifically targets bot-infected machines, in online lookup (i.e., within 1 min of packet reception the IP is looked up). As typically dynamic IPs are reassigned on average once per hour, this is not very surprising.

4.1. *Analysis of single packets: identifying spambot type*

Here, we describe our approach to identify bots based on a single packet which is captured by our passive darknet detector. This is an example of a Level 1 approach.

Reference data was provided by Marshal Ltd.[5] from their semi-automated TRACE system. We received monthly alignments of our IP data with the entries of their database, and matched them via maximum time difference of ±1 h, which is a reasonable estimate of the time a dynamic IP remains the same on average, see Fig. 7(b) in Xie et al. (2007). The mentioned figure shows that 97% of IPs change once an hour or less often. On average only about 5% of our IPs could be assigned to an entry and thus spambot type from Marshal with these restrictions, yielding about 2000 packets which were highly likely to have been sent by a known spambot.

Another issue was the opening of a connection from a single bot to different destination IPs, reusing the same source port. The source port is therefore correlated with spambot type and must not be used. Assigning the right spambot type in this case is trivial based on rote learning the source port – spambot mapping. Therefore we only took one packet of each pair (source IP, source port) to prevent such a positive bias. This reduced our dataset further to 308 samples of eight spambot types.

As feature representation we have used the following set:

- flags to indicate ICMP, TCP or UDP traffic,
- whether or not the DF flag in the IP packet is set,
- the IP destination port (ICMP, one of (20; 25; 1433; 5900; 7557; 16,818; 23,790; 34,357; 35,999; 37,370; 37,713; 49,110) or UNKNOWN), and

---

[5] www.marshal.com.

- a 2-byte gram word vector for TCP options (TCP packet), or the ICMP payload (ICMP packet), or the UDP payload (UDP packet)

The set of IP destination ports was taken from training data. We have chosen to use all information within each packet, only removing those information which cannot be meaningfully interpreted on a single packet level (e.g. sequence numbers) or which would have tainted the test data (e.g. source port as mentioned above). Our complete training data is available upon request.

Because of the small amount of data, we have evaluated our system by leave-one-out crossvalidation. For simplicity, because of its robustness to irrelevant attributes and in face of little training data, we have chosen a linear Support Vector Machine for training, using the WEKA[6] implementation with the default cost parameter and no parameter optimization (weka.classifiers.functions.SMO, default parameters (-C 1)). A linear SVM learns a linear threshold function of the feature vector for each combination of two classes (in our case $\binom{8}{2} = 28$ linear threshold functions). Pseudocode follows.

```
for each captured packet
    compute feature vector
    normalize feature vector to training data ranges
    apply SVM model
    choose spambot type with highest estimated
    probability
end for
```

Detection turned out to be very good for the bot types Srizbi, Unknown 11, and Unknown 4 with high precision and high recall; Rustock and Unknown 9 have high precision and low recall, so at least some of circulating variants can be clearly identified, and the other types are performing rather badly. The latter might be due to the fact that no clear patterns exist (e.g., randomized payload, too many different variants, or too little training data to detect patterns – note that for Hacktool.Spammer (6), Mega-D (1) and Pushdo (2) we only have 16, 5 and 6 training examples, respectively, while Srizbi (3) has 215 examples. *Unknown* (4), (9) and (11) are distinct patterns observed by Marshal researchers which are thought to belong to distinct spambots not yet identified by antivirus researchers (Table 1).

Nevertheless, we can detect bot activity and in some cases reliably assign spambot types based on our model.

4.2. *Static vs. dynamic IP addresses*

Since most bots are hosted on dynamic IP addresses, a reliable assessment of bot infection must be coupled to an incoming bot packet within a short time period. To enable a long-term lookup for non-dynamic IPs, we have also looked into recognizing static and dynamic IP addresses from reverse DNS entries. Additionally, this allows for a long-term study of botnet activity for a small subset of the bots – those which are hosted on machines with static IP addresses. This is another example of a very simple Level 1 approach, which uses just the source IP address from each packet and rDNS lookup. Presumably, at the moment this cannot be detected easily by botnet operators.

Table 1 – Precision, recall and F-measure for spambot types as determined by Marshal.

Spambot name (number)	Precision	Recall	F-measure
Srizbi (3)	0.847	0.981	0.909
Rustock (7)	1.000	0.263	0.417
Pushdo (2)	0.000	0.000	0.000
Hacktool.Spammer (6)	0.500	0.313	0.385
Mega-D (1)	0.000	0.000	0.000
Unknown (4)	1.000	1.000	1.000
Unknown (9)	1.000	0.143	0.250
Unknown (11)	0.947	0.973	0.960

Personal communication by Phil Hayes, Marshal Ltd.

However, injecting IP addresses where the rDNS entry is only available on a DNS server under the botnet operators' control is not impossible and would allow detection of the application of such a classification system. This is clearly a worst case scenario which would also facilitate non-user-detectable phishing, so we do not take it into account for now.

For distinguishing dynamic and static IP addresses, we have created a training set of known dynamic and known static IP addresses. For static IP addresses, we have used all entries from www.dnswl.org referring to single machines (/32) with at least medium trust level (med and hi). For dynamic IP addresses, we have used an overlap of incoming IP addresses with the PBL from www.spamhaus.org, which has been initialized from the now obsolete dynalist which was intended to capture dynamic IP addresses without reference to their spamminess. While dynalist was explicitly a list of dynamic IP addresses, the PBL also includes non-MTA IPs and may thus be considered slightly tainted. However, a better resource for known dynamic IPs was not available (note that Xie et al. (2007) used the same resource). We thus obtained 8359 dynamic IPs with existing rDNS entry and 10 387 static IPs with existing rDNS entry. 33.96% of IPs did not have an rDNS entry and were discarded. An additional test of spam-sending IPs by our local spamtrap found that 36.33% of the IPs sending spam did not have an rDNS entry, which agrees well with our sample, so bot IP addresses and spam-sending IP addresses show a similar fraction of entries with existing rDNS entry – about two thirds.

We downsampled the list of static IP addresses to the same number of entries as the list of dynamic IP addresses and created a set with equal class distributions. We removed entries present in both sets (there were none) and used only unique entries. We randomly selected half of this set to create a same-sized training and a test set. Experiments were run as is, and also with training and test set swapped, reporting the average error rate (equivalent to a two-fold crossvalidation).

In a first experiment using word vectors with delimiter "." (dot), where word vectors were computed on the training set and applied unchanged to the test set, an error rate of 2.91% with a standard deviation $\sigma = 0.174\%$ was achieved using a multinomial version of Naive Bayes from WEKA (weka.classifiers.bayes.NaiveBayesMultinomial). In a second experiment using n-gram characters, we tested $n = 1,2,\ldots,9$. 4-grams achieved the smallest error rate of 0.848%

---

[6] www.cs.waikato.ac.nz/~ml/weka.

(with $\sigma = 0.083\%$) using the same learning algorithm. A linear SVM was competitive on the word vector data with error rate 2.18% ($\sigma = 0.16\%$), and on 4-grams with error rate 0.864% ($\sigma = 0.136\%$). Thus, we obtained very good results in differentiating dynamic and static IPs.

When applying this to our data, we found that approximately 4.38% of the IP addresses from our darknet detector could be predicted as static IP addresses, and thus could be tracked over longer time periods, giving a more comprehensive view of botnet activity. However, the overlap of this IP set with Marshal's reference data was too small for further analysis.

4.2.1. *Open issues*

While the current model performs very well, the fraction of clearly static IP addresses within bots is rather small at only 4.38%. There are types of dynamic IP addresses which change less often than once an hour – according to Xie et al. (2007) another 3% of IPs might be of this type. We could thus at least double the amount of usable IPs for medium- to long-term tracking at the cost of a more complex model which tries to predict the approximate length of dynamic IP turnover, and do a more complex analysis with a mix of dynamic IPs with different turnover distributions. This might be an interesting venue for further research, provided sufficiently high-quality reference data can be obtained.

4.3. *Analysis of network traffic*

Here, we describe an example of a Level 2 approach. We have analyzed the access pattern of bots to our darknet within a six month period. For simplicity we have focussed only on the order of accesses within a short time span, chosen to minimize the risk of source IP change. The same systems' architecture was used (see Fig. 1).

More specifically, we have extracted all accesses to our darknet from any source IP from the first occurrence until 1 h later. After this time, a *cooling-off period* of 6 h was chosen to prevent new accesses from the same IP in the next window. Beyond the cooling-off period, the same IP was again admitted as start for a new access sequence. The period of 1 h was chosen to minimize the risk that the (probably) dynamic source IP changes, and another machine would seamlessly take over the old source IP, mixing access patterns from two distinct machines. While this means that we lose some data for IP addresses which do not change as often as we assume in the worst case, we felt that such a precaution was necessary to get sufficiently clean data. Pseudocode follows.

```
clear global access sequence list
for each IP address
   clear candidate list
   for each access sequence to this IP address (sorted by time)
      if (within 1 h of first access sequence in cand.l)
        add access sequence to candidate list
      elsif (more than 6 h since last access sequence in cand.l)
         process_candidate_list
         clear candidate list
         add access sequence to candidate list
      end if
   end for
   process_candidate_list
end for
proc process_candidate_list (candidate_list)
   for each access sequence in candidate_list
      lookup spambot type in reference data by IP & timestamp
      if (lookup successful)
         if (same access sequence appears with >1 spambot type)
            remove access sequence from global access sequence list
         else
            add (access sequence, spambot type) to global acc.seq.l.
         end if
      end if
   end for
end proc
```

The access pattern sequences are astonishingly varied. On the whole six months data we received 25 588 access sequences with a length of at least three accesses on at least three unique target IP addresses. 98.14% of these access patterns were unique. There were 15 627 unique source IP addresses, so 38.93% of the access patterns came from IPs which appeared more than once. None of them appeared in consecutive time windows, so the window of 6 h was sufficient to isolate the accesses from one source IP address. The average length of an access pattern was 67.95 ± 107.53 accesses, the average number of unique target accesses was 47.44 ± 69.09. Systematic port scanning activities are quite common – the ten most common start sequences are accesses to consecutive IP addresses within one subnet.

A sensible choice to determine distance between access pattern sequences is the edit distance, where the character set contains all our target IP addresses. We chose to use the Levenshtein distance (Levenshtein, 1966) (cost of insertions, deletions and substitutions are uniformly 1.0), and visualized a random sample of 288 accesses (about 1% of the total number of accesses) via Sammon mapping (Sammon, 1969; Borg and Groenen, 1997). *Sammon mapping* is a non-linear multi-dimensional scaling technique which directly minimizes stress and thus tries to keep the same distance matrix between examples as far as possible, thus aiming for a *distance-preserving* visualization.

Fig. 2 shows the results. A cluster which contains a significant portion of access pattern sequences is clearly visible near (0,0) on the right side. The observed spreading-out pattern reflects most likely different access lengths, since a higher length will incur a higher edit distance.

For a more detailed analysis, we reused the reference data by Marshal and aimed to assign a definitive spambot type to each access pattern sequence. Again, the small overlap resulted in only 82 matches by source IP and timestamp (access within ±1 h of the reference) with 45 unique access pattern sequences. However, 13 of those had ambiguous spambot type (i.e., more than one type for the same access pattern) and had to be discarded, leaving us with 32 usable sequences. As we will

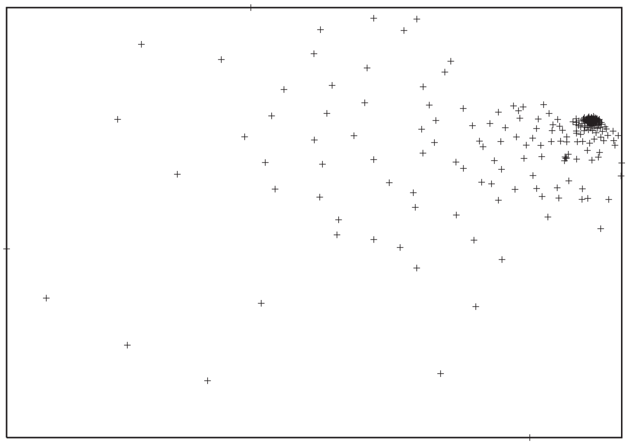

Fig. 2 – Sammon mapping of 288 access patterns.

see later, the unambiguous mapping of this information to spambot type may not be the most appropriate use of the access patterns. When using a Sammon mapping as in Fig. 2, the majority of sequences were clustered within a small area and significantly overlapped (data not shown), so we have chosen a different visualization technique for clarity.

Fig. 3 shows a distance matrix of all 32 sequences, where dark refers to small edit distance (i.e. similar access sequences) and white refers to high edit distance (i.e. dissimilar access sequences). For clarity, we have used logarithmic scale followed by histogram normalization. The columns and rows are sorted by edit distance values from small to high which facilitates the search for clusters. Columns and rows correspond to the access patterns and the diagonal corresponds to the distance of each pattern to itself, i.e. zero. The order of access patterns for rows and columns is the same. A cluster is therefore a submatrix containing one specific set of columns and the same set of rows, which is uniformly dark (e.g. rows 2,5 and 6 and columns 2,5 and 6 give a 3 × 3 submatrix which seems to be a cluster). Top left is column 1, row 1.

By visual inspection we can discern three clusters in Fig. 3: (A) second, fifth and sixth column/row; (B) first, eighth, and tenth column/row; (C) 13th, 16th, and 17th column/row. If the corresponding columns and rows from a cluster are selected, the resulting submatrix is almost completely dark (i.e. similar) relative to the background. Again, one effect of the edit distance's dependency on sequence length is that the longer sequences are to the right bottom corner of the figure and that the edit distance tends to get higher in that direction.

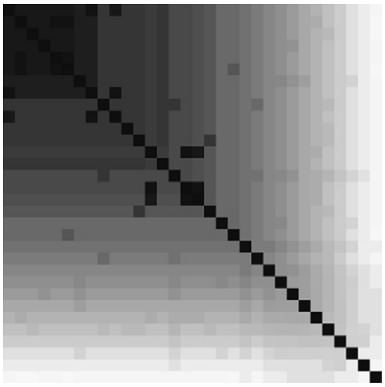

Fig. 3 – Complete edit distance matrix of 32 access pattern sequences with known spambot type (white = high distance, black = low distance).

A detailed analysis shows that cluster (A) contains three different sequences with a length of one which therefore all have an edit distance of one between them. This is of course rather uninteresting. However, (B) shows a variable number of accesses to a specific target IP – twice for spambot Srizbi, once for spambot Rustock. (C) also shows a variable number of accesses to a different specific target IP – twice for Srizbi, twice for Rustock. Two weaker clusters can also be discerned. (D = 9,15,22) shows again a variable number of accesses to a different specific target IP, all for Srizbi. In one case, a small set of other IPs were also accessed. (E = 12,18) shows again a variable number of accesses to a different specific target IP, once for Pushdo and once for Srizbi.

Immediately, we began to wonder why different spambots show extremely similar access patterns. Could this point towards a common control mechanism, e.g., a botnet consisting of multiple spambot types which were all asked to check out the same target IPs? It is highly unlikely that almost half of our sequences should be so similar if the process of choosing IP addresses to scan would be either random, or depend on spambot type. Neither hypothesis is supported by our data at present. However, we must leave a more detailed analysis for future work due to lack of data.

The six most dissimilar sequences (bottom-right of figure) cannot be fixed to a specific spambot type, either. These access sequences are long, seem to be random, and scan a significant portion of our address space (84.84% ± 18.17% on average, with 1.9 ± 0.10 accesses per target IP). These seem to be spambots set to a scanning mode, while the previous clusters may be reconnaissance probes intended to search for vulnerable subnets before an exhaustive scanning takes place. Note also that the access pattern sequence here is significantly longer than for an average pattern from the whole dataset.

All in all, the access pattern sequences may not tell us too much about the specific spambot type, but rather about the mode in which a botnet operates, and show tentative connections between source IPs and spambot types which may hint at a common operator, membership of the same botnet, or other factors which are not yet well understood.

4.4. *Analysis of TCP/IP traffic*

Here, we describe an example of a Level 3 approach. As our darknet does not allow TCP/IP connection set-up, no continuous TCP/IP traffic could be recorded. Therefore, we aligned data from a local spamtrap addresses with Marshal's spambot data in a similar way as described earlier. This allowed us to map a subset of SMTP conversations to a specific spambot type. We have visualized spam mail content vs. spambot type, and found some interesting patterns which may indicate that different spambots are used for sending out different kinds of spam. Fig. 4 shows the systems' architecture for this approach. We analyzed those IPs for which data from both Spamtrap and SpambotTracker was available.

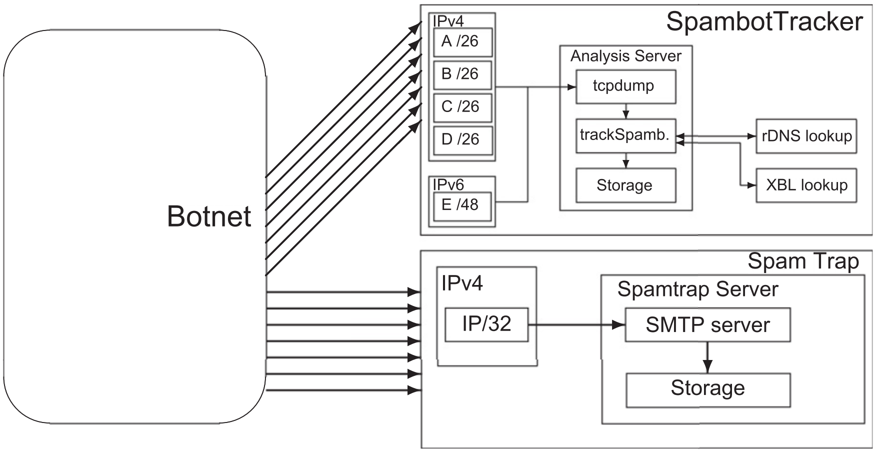

Fig. 4 – Systems' architecture for Level 3 (combined SpambotTracker and Spamtrap).

This specific experiment does not result in a completely passive system, as the receipt of SMTP mail needs a TCP conversation which has to be simulated by our spamtrap, and would thus be amenable for detection attempts. However, by analyzing SMTP traffic moving through a network the same analysis could be done in a completely passive way as well. This could be done e.g. with an external firewall. Strictly speaking, the analysis of full TCP/IP traffic is not possible with a purely passive darknet.

By analyzing about 200 000 spam mails from our spamtrap, we found that most spam data is indeed sent out in a single packet. However, not all packets sent to port #25 are spam content. In fact, we found e.g. DATA commands on top of the spam mail and QUIT afterwards, multiple spam mails in a single packet with appropriate SMTP commands between them and so on. Only an analysis of the whole TCP conversation gives enough information to distinguish the content of the mail which was sent from status and informational data transferred. In fact, we found a few messages containing malware executables where TCP fragments were used to distribute the content of the message into multiple packets, presumably to confuse packet-level firewall virus scanners.

Reference data was again provided by Marshal from their automated TRACE system. We restricted the data and obtained two selective alignments, each for a period of 24 h. Again, we matched them via maximum time difference of 1 h. We obtained 192 pairs with (received spam mail, spambot type). Because of earlier experiments reported in Seewald and Kleedorfer (2007) we chose a 6-character-gram representation for each mail instead of the more widely used word vector representation. This yielded 7008 unique 6-gram tokens. For visualization we again chose Sammon mapping. We reduced the 7008-dimensional binary vector to two dimensions, and applied the known spambot types for tagging afterwards. Pseudocode follows.

```
for each received spam mail
    compute 6-gram feature vector from mail contents
    lookup spambot type in reference data by IP &
    timestamp
    if (lookup successful)
        output feature vector and spambot type
    end if
end for
```

Fig. 5 shows the results. As it can be seen, Rustock (7) and Srizbi (3) are responsible for the majority of incoming spam. The clusters of spam mail content are clearly distinguishable. Note that one spambot type which could not be assigned to any darknet IP (i.e. no reference data for this spambot type could be matched to any of our darknet packets – this means that no verified activity of this spambot was observed by our darknet) was nevertheless found sending out spam, namely Grum (8). A few less active spambots can also be discerned.

Although spambots can potentially be used to send out any kind of spam, there seem to be clear patterns in the actual spam sent out. These might reflect usage patterns of spammers (such as preferring simpler spambots to send out cheaper spam), technical constraints (e.g., specific interfaces between some text generators for spam and some spambots, which facilitate combined usage) or some other factors, and thus could give valuable information about the source of the spam, as well as of the underlying botnets and their operators.

5. Outlook

Given our prototype system, and reasonable assumptions about the number of bots, it is possible to roughly compute the necessary size of our darknet to catch each bot at least once per day.

We have taken the estimate by ShadowServer[7] of around 600 000 bots, multiplied it by three and rounded up to 2 000 000 bots. Our present system sees 350 unique IPs each day from analyzing 30 000 packets, so we need on the order of 1.5 million unused IP addresses to see each bot on average once per day, and need to analyze 170 million packets per day (2000 packets per second). This might be difficult to obtain because of IPv4 shortage, but we remain optimistic that a sufficient number of

---

[7] www.shadowserver.org.

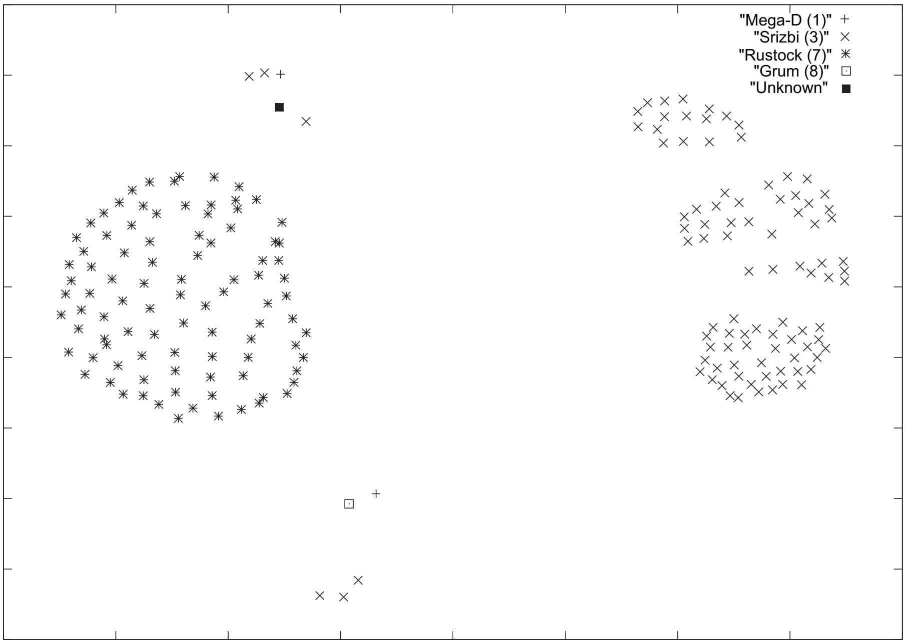

Fig. 5 – Sammon mapping of 192 spam mails based on content with attached spambot types.

unused IPs will be donated via IP-over-UDP forwards. Please contact the authors if you are interested to contribute.

Concerning processing speed, our current system is reasonably efficient: 5000 packets could be analyzed each second, so theoretically a single server would be sufficient to analyze this data at Level 1. The technical limit for packet capture seems to be around 170 000 packets per second, but this does not include analysis and storage of this data is only feasible for short time periods.

Higher level analysis will need more computational power, but will also enable more complex detection and identification of botnets. The present system is not yet able to identify whole botnets. Although the spambot type is one weak indicator of botnet membership, a reconstruction of botnets is not feasible with only this information. Reference data on known botnets would be very helpful for creating such a system.

Some of the patterns we mentioned could be removed quite easily. Even a complete passive analysis does not prevent the awareness of such patterns by papers such as this, and countermeasures are usually available. However, all such tasks will make the botherders' job harder and more complex, thus increasing costs and making botnets less attractive. Also, through our focus on machine learning approaches, regular validation with data obtained from other systems enables us to update our systems and note when some features are no longer working.

While technically feasible, there is a legal problem to automatically deactivate bots. Also, for obsolete operating systems which can no longer be made safe, removing one bot will yield another infection quite soon. We therefore believe that ISPs are best placed to solve this problem for their own customers. Forcing ISPs to keep their nets bot-free, e.g. by disconnecting known long time offenders from the internet, would be a way to speed up this process.

6. Conclusions

We have presented a framework for passive tracking and identification of botnets based on analysis on three distinct but related levels:

- Level 1: single packet analysis
- Level 2: network traffic analysis
- Level 3: TCP/IP traffic content analysis

We have also presented and experimented with working systems at each of the three levels. For Levels 1 and 2 these systems are already purely passive and thus cannot be detected by botnet operators directly. Although the system presented for Level 3 is not purely passive yet, it can also be made so with minimal adaptation.

For Level 1, we have built a system to detect botnet traffic and recognize specific spambot types. For three of the eight considered spambot types, we obtained an F-measure of at least 0.9 and for an additional two we obtained perfect precision but low recall (<0.26) on unseen data. We also built a system to distinguish static from dynamic IP addresses based on rDNS data with an estimated accuracy of 99.15% and noted that – as expected – most bot IPs are dynamic.

For Level 2, we have analyzed access pattern sequences to our darknet from specific IP addresses. While we could find no correlation of access patterns to spambot type, we noted that different spambots showed surprisingly similar access patterns. This may indicate that these different spambots are controlled by the same operator and forced to visit the same IP addresses in the same order, and may be used to reconstruct control flow beyond specific botnets.

For Level 3, we correlated output from our system concerning spambot type with additional data on incoming spam mails from a spamtrap and were able to show that different spambots send out radically different types of spam mail. We also noted that the amount of spam sent out by these bots is very different – two bots, Srizbi and Rustock, sent out almost all of the spam received during the 24 h period which we had considered.

We are interested in further development of our system towards more comprehensive tracking of bots, and seek collaboration with providers of reference data on botnet activity, owners of unused IPv4 addresses, law enforcement and other groups who take an active interest in addressing the botnet problem.

Acknowledgements

This work was partly supported by Internet Privatstiftung Austria (IPA). We also deeply acknowledge the support of Marshal Ltd., New Zealand, and especially our contact Phil Hay, who provided the reference data which was essential to our experiments.

Appendix. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.cose.2009.07.007.

REFERENCES

Bailey M, Cooke E, Jahanian F, Nazario J, Watson D. The internet motion sensor: a distributed blackhole monitoring system. In: Proceedings of the 12th annual symposium on network and distributed system security (NDSS 2005), San Diego, California.

Borg I, Groenen P. Modern multidimensional scaling. New York: Springer-Verlag; 1997.

Collins MP, Shimeall TJ, Faber S, Janies J, Weaver R, De Shon M, et al. Using uncleanliness to predict future botnet addresses. IMC '07. In: Proceedings of the 7th ACM SIGCOMM conference on internet measurement. October 24–26, 2007, San Diego, California, USA. New York, NY: ACM. p. 93–104, http://doi.acm.org/10.1145/1298306.1298319; 2007.

Cormack GV. Email spam filtering: a systematic review. Foundations and Trends in Information Retrieval 2006;1(4):335–55.

Dagon D, Zou C, Lee W. Modeling botnet propagation using time zones. In 13th annual symposium on network and distributed system security (NDSS 2006), San Diego, California, USA; 2006.

Dagon D, Gu G, Zou C, Grizzard J, Dwivedi S, Lee W, et al. A taxonomy of botnets. Unpublished report; 2005. http://www.math.tulane.edu/~tcsem/botnets/ndss_botax.pdf.

Daley DJ, Gani J. Epidemic modeling: an introduction. Cambridge University Press; 1999.

Dhamankar R, King R. Protocol identification via statistical analysis (PISA). Black Hat, Las Vegas, USA; 2007.

Gansterer WN, Pölz D. E-mail classification for phishing defense. In Proceedings of the 31st European conference on information retrieval (5478); 2009. p. 449–60.

Gansterer WN, Janecek AGK, Lechner P. A reliable component-based architecture for e-mail filtering. In: Proceedings of the ARES international conference on availability, reliability and security 2007. Washington, D.C.: IEEE Computer Society; 2007. p. 43–52.

Gauthronet S, Drouard E. Unsolicited commercial communications and data protection. Technical report, Commission of the European Communities; 2001.

Janecek AGK, Gansterer WN, Kumar KA. Multi-level reputation-based greylisting. In: Proceedings of the ARES international conference on availability, reliability and security. Washington, D.C.: IEEE Computer Society; 2008. p. 10–7.

Karasaridis A, Rexroad B, Hoeflin D. Wide-scale botnet detection and characterization. In First workshop on hot topics in understanding botnets (HotBots '07), Cambridge, MA, USA; 2007.

Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics – Doklady 1966;10: 707–10.

National Infrastructure Security Co-ordination Centre (NISCC). Botnets – the threat to the critical national infrastructure. NISCC Monthly Bulletin, London, October 17, 2005.

Rajab MA, Zarfoss J, Monrose F, Terzis A. A multifaceted approach to understanding the botnet phenomenon. In Proceedings of the 6th ACM SIGCOMM conference on internet measurement (IMC 2006), Rio De Janeiro, Brazil; 2006. p. 41–52.

Ramachandran A, Feamster N, Dagon D. Revealing botnet membership using DNSBL counter-intelligence. In Proceedings of the 2nd workshop on steps to reducing unwanted traffic on the internet (SRUTI 2006), San Jose, California, USA; 2006.

Register of Known Spam Operations (ROKSO) Database. Operated by the Spamhaus Project, http://www.spamhaus.org/rokso/index.lasso [accessed 10.03.09].

Sammon JW. A nonlinear mapping for data structure analysis. IEEE Transactions on Computers 1969;C-18:401–9.

Seewald AK, Kleedorfer F. An approximation of the string subsequence kernel for practical SVM classification and redundancy clustering. Advances in Data Analysis and Classification December 2007;1(3):221–39. doi:10.1007/s11634-007-0012-1.

Seewald AK. An evaluation of Naive Bayes variants in content-based learning for spam filtering. Intelligent Data Analysis 2007;11(5):497–524.

Seewald AK. Towards automating malware classification and characterization. In Konferenzband der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik [German-language proceedings], Saabrücken; April 2008. p. 291–302.

Stampfel G, Gansterer WN, Ilger M. Data retention – the EU Directive 2006/24/EC from a technological perspective. Medien und Recht, ISBN 978-3-900741-53-2; 2008a.

Stampfel G, Gansterer WN, Ilger M. Implications of the EU Data Retention Directive 2006/24/EC. In Proceedings of Sicherheit P-128. Springer lecture notes in informatics; 2008b. p. 45–58.

Stark K, Stampfel G, Gansterer WN. A distributed data warehouse for e-mail data retention. In: Proceedings of the 4th international conference on e-government (ICEG08). Academic Publishing Limited; 2008. p. 413–22. ISBN 978-1-906638-20-7.

US-CERT Vulnerability Note VU#800113, http://www.kb.cert.org/vuls/id/800113 [accessed 10.03.09].

Xie Y, Yu F, Achan K, Gillum E, Goldszmidt M, Wobber T. How dynamic are IP addresses. In Proceedings of SIGCOMM '07, August 27–31, 2007, Kyoto, Japan.

Alexander K. Seewald graduated with a masters degree (Dipl.-Ing.) in Computer Science from Vienna University of Technology and received a PhD (Dr.techn.) in Machine Learning from Vienna University of Technology. Subsequently, he worked for several years at the Austrian Research Institute for Artificial Intelligence, Dept. for Machine Learning. He joined the GE Money Bank Austria for a year and afterwards founded his own company, Seewald Solutions, in 2007.

His research interests include intelligent data analysis, image, video and audio mining, practical machine learning, proactive spamfiltering, and ensemble learning.

Wilfried N. Gansterer graduated with a masters degree (Dipl.-Ing.) in Technische Mathematik from Vienna University of Technology, with an MSc in Scientific Computing/Computational Mathematics from Stanford University, and received a PhD (Dr.techn.) in Scientific Computing from Vienna University of Technology. Subsequently, he worked for several years at the Department of Computer Science at the University of Tennessee at Knoxville. He joined the Department of Distributed and Multimedia Systems as assistant professor at the end of 2002. Since 2006, he is the head of the Research Lab Computational Technologies and Applications.

His research interests include scientific computing and computational science, parallel and distributed computing, numerical and high performance computing, internet security and data mining.