FEATURE

COVERT ZOMBIE OPS

John Aycock
University of Calgary, Canada

Time for a thought experiment. An evil botmaster wants to take over the world, as evil botmasters are wont to do, and a botnet of a million zombie computers has been amassed for this purpose. Can the botmaster send commands covertly to the zombies and control them in real time?

This is not necessarily a hypothetical question. A large-scale DDoS attack could be re-aimed dynamically at different targets, or a physical attack could be accompanied by an Internet attack that changes dynamically to cause confusion. It makes sense to consider in advance how the botmaster's command channel might look, so that it could be detected and disrupted if necessary.

Unless the botnet is meant to be obvious and short-lived, there is a set of severe design constraints on communication for the botmaster. Communication must be covert; it should reach a large majority of infected machines; it must be scalable; transmissions should be limited; it should be very hard to trace the communication source; it should be resistant to the insertion of false signals; it should be sustainable over a long period of time; it should be real-time (or close to it).

Some aspects can be handled easily with existing techniques. For example, resisting false signals can be achieved by applying public key cryptography to digitally signed commands [1]. If the botmaster encrypts commands with a private key, then the corresponding public key – distributed with the malware that created the zombies – can be used both to decrypt the command and to verify that the command came from the botmaster. Longer commands, following the usual wisdom for digital signatures, would encrypt/decrypt a digest of the botmaster's command for performance reasons.

Other communication aspects require more analysis. There would seem to be a direct relationship between the traceability of the botmaster and the degree of responsiveness/interactivity the botmaster experiences when controlling the botnet. Consider two extreme points:

1. The botmaster prerecords commands, and places them in some well-known location. The zombies could periodically poll for new commands. There is no interactivity for the botmaster, and low responsiveness due to the time lag between command recording and realization. However, the botmaster is very difficult to trace, and need not even be connected to the Internet when an attack occurs.

2. The botmaster broadcasts commands continuously to the botnet. Assuming sufficient bandwidth, this would yield the highest interactivity, with responsiveness

limited only by network latency. The disadvantage is that a single source pumping out continuous network traffic would be relatively easy to trace. There is legitimate work related to this in the area of music performance, where systems have been constructed to enable musicians to perform together across a network [2–5]. Not surprisingly, overcoming network latency is the major technical hurdle. The closest system to what a botmaster would need is the 'conductor architecture' [3], where one conductor sends a global signal to multiple musicians upon which they can synchronize. The similarity ends here, though: network music performance is not intended to be covert, nor designed to scale beyond a small, finite number of musicians.

Between these points lie many feasible methods of communication, depending on how much loss of responsiveness and interactivity the botmaster can tolerate.

There are two key goals for the botmaster to accomplish. First, the amount of communication from the botmaster must be reduced. This helps increase scalability and reduce traceability. By connecting the zombies into multiple small botnets, the botmaster needs to send commands to only a limited number of botnet command-and-control machines, instead of every single infected machine; commands propagate from botnet to botnet. In effect, the botmaster would have a network of botnets, a 'super-botnet', which can be constructed automatically to resist countermeasures, yet remain highly receptive to commands [6].

Second, the propagation of commands to individual infected machines must be hidden. This helps the zombies avoid detection for longer periods. HTTP is an excellent candidate through which infected machines can poll their botnet's command-and-control server (which would run an HTTP server on port 80) for new commands. This happens already [7], but it can be done much more covertly.

Using HTTP for intra-botnet communication has definite advantages: natural cover traffic generated by real users, carte blanche to pass through egress firewalls, and automatic leveraging of web caches. However, polling an HTTP server for commands at frequent, regular intervals is not typical user behaviour. What *is* typical behaviour has been studied extensively [8].

If zombies were designed to be covert and exhibit HTTP traffic characteristics typical of users, there would be implications for the botmaster sending commands, which include:

- Time. HTTP traffic has been shown by many studies to peak during the daytime [9–12], and zombies would need to shape their traffic accordingly. Interestingly, a diurnal pattern has also been noted in botnet communication [13], but it was rationalized by saying that 'many users turn their computers off at night'. With an increasing number of always-on computers, the possibility cannot be ignored that surreptitious malware may mimic the diurnal cycle deliberately to delay detection. The implications for a botmaster are that interactivity and responsiveness cannot be expected to be consistent across the globe; zombies exhibiting diurnal behaviour would respond faster during the day, and the physical location of zombies would become a factor.
- Transfer size. HTTP traffic can be broken down in various ways, but one potential concern for a botmaster is how large a command can be transferred to a zombie without raising suspicion, i.e. not looking like normal HTTP traffic. There have been many studies that gather statistics about the median file size transfer, the median size of HTML files, and the median HTTP transfer size [12,14–18]. In this data, it is *highly* unusual to see a reported median under 2K; most report 2K or higher. From this we conclude that a median size of 2K commands can be used reliably by botmasters, which is more than sufficient to contain a short command. Note that this doesn't preclude larger transfers, like executables, but just means that they may have to be broken into multiple pieces.
- Transfer frequency. Retrieving the complete contents of one web page for rendering may result in a burst of discrete HTTP transfers, such as the download of an HTML file followed by the fetching of inline images. We assume that such rapid-fire retrieval is not indicative of how often an infected machine can poll for new commands covertly; instead, the time that would normally elapse in between complete web page retrievals is of interest over the long term.

  This time has been measured in various ways. Reported medians are 11s [17] and 15s [16], with heavy-tailed distributions that result in larger mean times of 47s [17] and 81s [19]. A definite conclusion about covert zombie behaviour is harder to draw from this data. An average polling rate of one minute appears likely, so long as plenty of time variations are introduced artificially. However, infected machines will not be polling in lock-step with one another. This means that the average polling rate should be taken as a worst-case indication of overall response time to the botmaster's commands.
- Domain names. Phishing URLs and *Google*'s cache notwithstanding, it is fair to say that a large majority of URLs specify the HTTP server using a domain name, which must be mapped into an IP address; typically this mapping is done via DNS queries. Although not part of the HTTP protocol *per se*, DNS lookups are thus a

characteristic of normal HTTP traffic, and covert zombie communication would have to exhibit this too.

Defensively, it is tempting to try and block or corrupt the zombie's HTTP traffic, but a false positive when detecting zombie communication would affect user HTTP, a high-visibility error. And, even if good and bad HTTP could be separated, it would have to be separated on a very large scale to be effective, because there would be numerous distributed HTTP servers rather than a small number of centralized ones.

Are DNS lookups the Achilles' heel of covert zombies? Security work has been done correlating DNS lookups with subsequent connections [20]; this could be applied to flag computers with consistently anomalous DNS behaviour. Zombies that get DNS lookups correct – and could avoid anomaly detection – would be relying on the DNS infrastructure. But this may not afford effective detection either; DNS caches, for example, may prevent suspicious queries from reaching a detection system.

While not giving complete responsiveness and interactivity, it seems to be within the technical reach of botmasters to have some degree of dynamic, covert control of large numbers of zombies. Users are yet again a key element, not as an infection vector, not as victims, but as the model of behaviour to which covert zombies must conform.

REFERENCES AND NOTES

My research is funded in part by a grant from the Natural Sciences and Engineering Research Council of Canada. Helpful suggestions were made by Andrew Warfield, Carey Williamson, and a reviewer.

[1] Schneier, B. (1996). *Applied Cryptography* (2nd ed.). Wiley.

[2] Chafe, C., Wilson, S., Leistikow, R., Chisholm, D., & Scavone, G. (2000). A simplified approach to high quality music and sound over IP. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*.

[3] Bouillot, N. (2004). The auditory consistency in distributed music performance: a conductor based synchronization. *Information Science for Decision Making*, *13*.

[4] Gu, X., Dick, M., Kurtisi, Z., Noyer, U., & Wolf, L. (2005, June). Network-centric music performance: practice and experiments. *IEEE Communications Magazine*.

[5] Kurtisi, Z., Gu, X., & Wolf, L. (2006). Enabling network-centric music performance in wide-area networks. *Communications of the ACM*, *49*(11).

[6] Vogt, R., Aycock, J., & Jacobson, M., Jr. (2007). Army of Botnets. In *Network and Distributed System Security Symposium 2007*.

[7] Ianelli, N., & Hackworth, A. (2005). Botnets as a vehicle for online crime. CERT Coordination Center.

[8] Pitkow, J. E. (1999). Summary of WWW characterizations. *World Wide Web*, *2*(1–2).

[9] Crovella, M. E., & Bestavros, A. (1996). Self-similarity in World Wide Web traffic: Evidence and possible causes. In *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*.

[10] Gribble, S. D., & Brewer, E. A. (1997). System design issues for Internet middleware services: deductions from a large client trace. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*.

[11] Abdulla, G. (1998). *Analysis and modeling of World Wide Web traffic*. Ph.D. thesis, Virginia Polytechnic Institute.

[12] Williams, A., Arlitt, M., Williamson, C., & Barker, K. (2005). Web workload characterization: ten years later. In X. Tang, J. Xu, & S. T. Chanson (Eds.), *Web content delivery*. Springer.

[13] Dagon, D., Zou, C., & Lee, W. (2006). Modeling botnet propagation using time zones. In *13th Annual Network & Distributed Security Symposium*.

[14] Bray, T. (1996). Measuring the Web. *Computer Networks and ISDN Systems*, *28*(7–11).

[15] Woodruff, A., Aoki, P. M., Brewer, E., Gauthier, P., & Rowe, L. A. (1996). An investigation of documents from the World Wide Web. *Computer Networks and ISDN Systems*, *28*(7–11).

[16] Mah, B. A. (1997). An empirical model of HTTP network traffic. In *Proceedings of INFOCOM '97*, *2*.

[17] Abrahamsson, H. (1999). *Traffic measurement and analysis*. Technical report T99:05, Swedish Institute of Computer Science.

[18] Barford, P., Bestavros, A., Bradley, A., & Crovella, M. (1999). Changes in Web client access patterns: characteristics and caching implications. *World Wide Web*, *2*(1–2).

[19] Vicari, N. (1997). *Measurement and modeling of WWW-sessions*. Technical report 184, Institute of Computer Science, University of Würzburg.

[20] Whyte, D., Kranakis, E., & van Oorschot, P. C. (2005). DNS-based detection of scanning worms in an enterprise network. In *12th Annual Network & Distributed Security Symposium*.