DESIGN OF A DDOS ATTACK-RESISTANT DISTRIBUTED SPAM BLOCKLIST

Jem E. Berkes
Department of Electrical and Computer Engineering
University of Manitoba,
Winnipeg, MB., Canada R3T 5V6
Email: umberkes@cc.umanitoba.ca

Abstract

This paper introduces the high-level design for a novel distributed spam blocklist system based on Peer-to-Peer architecture. Deployed on the Internet, this blocklist would be resistant to Distributed Denial of Service (DDoS) attacks without requiring costly investments in server resources. Digital signatures make widespread network participation possible without compromising data integrity. This paper offers a much-needed solution for serving spam blocklists in hostile environments and outlines the constituent software and protocols required. The proposed system requires minimal modification to existing servers, as it can operate alongside current software.

Keywords: DDoS spam blocklist distributed P2P

I. Introduction

The primary defences against Unsolicited Bulk Email (UBE, or "spam") utilized by mail servers are blacklists or blocklists of IP addresses that send spam. Paul Vixie created a DNS-based blocklist (DNSBL) distribution technique that has since been adopted by most blocklist projects. For example, the SORBS project [1] maintains a list of IP addresses shown to be open relays or proxies through anonymous testing coordinated by volunteers. This list of IP addresses or domain names is loaded and served as a DNS zone by name servers [2].

Mail servers from around the Internet can query a blocklist by doing a simple DNS query to see if a particular IP address is listed. Using standard reverse-octet notation, a host could for example look up 23.16.179.130.*dnsbl.domain* to see if 130.179.16.23 is listed in the database. A mail server may wish to deny mail service to a connecting host that is listed in a certain blocklist; the specific policy is determined by the mail site administrator, not the blocklist.

Because these UDP-based queries are efficient and offer near real-time remote database lookups, they have become widely adopted for spam filtering in mail server software such as Postfix [3], Sendmail [4], and SpamAssassin [5]. Unfortunately, blocklist servers have also become popular targets for DDoS attacks from unknown parties; presumably, the attackers' intention is to disrupt blocklist services that are currently preventing the distribution of their spam.

The damage caused to both non-profit and commercial blocklists by DDoS attacks is non-trivial. One of the first casualties was Joe Jared's blocklists served from osirusoft.com; in late August 2003, global email was affected as Osirusoft listed all IP addresses in an effort to stop queries and shut down the service. George Herbert clarified on the NANOG forum, "Yes, this is due to a massive DDOS" [6].

The next month, DDoS attacks against Ron Guilmette's Monkeys.com blocklist prompted the termination of that service as well. Ron announced in a public forum:

> "I rode out the first massive DDoS against my site . . . but over the past three days I have been massively DDoS'd again, and I think that the handwriting is now on the wall. I will simply not be allowed to continue fighting spam."[7]

Since then, DDoS attacks have continued against even larger blocklists: Spamhaus [8], SPEWS [9], and SpamCop [10] have each endured attacks lasting several months. While these services have survived to date (due to greater resources), it's clear that malicious Internet attacks can easily cripple blocklists, or at least make running such blocklists costly and unattractive. This fact has motivated the design of a new, distributed blocklist system as described in this paper.

II. Motivation

One of my own projects is the Weighted Private Block List (WPBL), a collaborative effort among a handful of system administrators to detect and share real-time spam sources [11]. While the information contained in our blocklist database can be useful for the public, concern over insufficient server resources and DDoS attacks prevented us from opening up our list to public access.

Even if other system administrators donate name servers to help serve the public DNSBL, the fact remains that all resources allocated to the task become potential targets for attacks. Because the risks outweigh the benefits, fear of Internet-based attacks has kept the WPBL from becoming a useful public resource.

With the realization that this was a common theme among blocklist operators, an early design for a distributed blocklist system was drafted [12]. Positive feedback on the idea from colleagues prompted a more formal description of the system, as documented in this paper. One may hope that a distributed blocklist infrastructure like the one described in this paper can become a common method for serving public anti-spam resources using many individuals' resources (with several blocklists using the same infrastructure).

III. System Structure

While current DNSBLs are served from a relatively small number of static name servers, a distributed blocklist system calls for a large, dynamic network of relatively equal peers. The hosts in this network communicate in a Peer-to-Peer (P2P) fashion, sometimes acting as clients and other times acting as servers.

For the high-level description of the distributed system, Table 1 describes the entities and their purposes.

Table 1. System entities and purposes

Entity	*Purpose*
Publisher (role)	• Blocklist creator and authority • Injects data into network
Package (data)	• Basic data unit, reasonable size • Created by Publisher only • Subset of entire blocklist • Partitioned to facilitate search
Node (peer)	• Stores blocklist data (Packages) • Serves Packages to clients/Nodes • Gets Packages from other Nodes • Stores information about Nodes

The Publisher is the source of all blocklist data. Unlike existing DNSBLs, where zone transfers move entire lists, this distributed system involves the transfer of smaller data Packages. Various techniques can be used for effective partitioning of the entire list; examples are indexing based on the first M octets of an IP address, or the first N bits of the hash of an entry. Either way, the goal is to determine the unique Package name that must contain (or not contain) the desired entry.

Figure 1 shows how the Publisher injects new Packages into the network, and how these Packages are distributed among the Nodes. Any Node may act as a "client", i.e. the user doing a lookup.

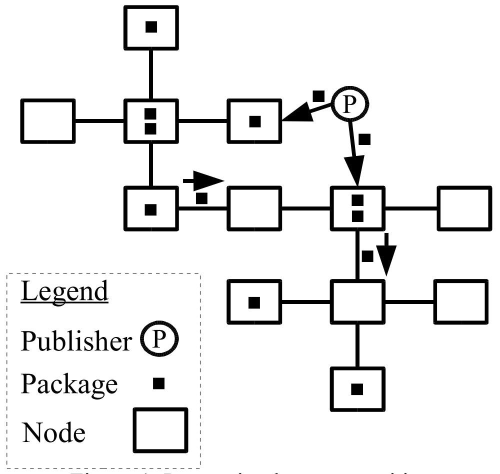

Figure 1. Interaction between entities

IV. Trust / Integrity

Since the Publisher's data Packages are passed through a large number of Nodes, this raises the concern of malicious data tampering or accidental corruption. This paper proposes a solution based on public/private-key cryptography; specifically, digital signatures provided by PGP technology [13].

The Publisher widely distributes her public key to all Nodes, through any means. PGP allows anybody to see public keys – even malicious parties. Any Node that wishes to participate in the distributed network stores a copy of the Publisher's public key, for the purposes of verifying digital signatures on *all Packages* received through the network.

The Publisher digitally signs every Package using her private key. This digital signature consists of a hash code or message digest which protects the integrity of the data, along with a time stamp [14]. When a signature is generated from these fields, the Publisher's public key will verify the signature only if the signed data remains unchanged. Authenticity is ensured.

Since no Node can trust another Node (nor any "Publisher" attempting to inject data), central to the distributed system's operation is the practice of verifying digital signatures on all Packages received, and discarding invalid Packages. This scheme ensures that only certified Packages propagate throughout the network of Nodes. The mandatory PGP signature checking also introduces some interesting characteristics of the network:

- Allows "anybody" to run a network Node, without having to establish their trust
- Provides reliable detection of rogue Nodes
- Allows the Publisher to inject data from any point of the network
- Allows an end-point client to pull equally reliable data from any network Node

V. Node Behaviour

The behaviour of each Node must facilitate the propagation and retrieval of valid blocklist Packages throughout the network. While the specifics of Node algorithms are beyond the scope of this paper, certain basic behaviours are vital to running a successful network:

1. Package signature checking: As described earlier, every Node must use the Publisher's public key to verify digital signatures on all Packages received, dropping any invalid Packages and noting rogue peers.
2. Caching: Packages moving through Nodes should be cached in local storage to some degree. This provides ample duplication of the blocklist data, allowing several Nodes to answer calls for data.
3. Tracking neighbours: Nodes must be aware of URLs for other Nodes, perhaps through human collaboration. This knowledge may be shared with other Nodes, provided the neighbours are returning authentic Packages.
4. Package updating: Packages with newer timestamps (also protected by digital signatures [14]) must invalidate older Packages, and Nodes must make an effort to acquire newer data once a Package has become stale. This should allow fresh data injected by the Publisher to propagate.
5. Content advertising: Nodes should tell their peers what they have cached locally, in order to help spread the most recent data and facilitate rapid Package lookups in the future.

VI. Protocols

Two main protocols are ideal for the proposed distributed blocklist due their widespread use and extensibility.

Hypertext Transfer Protocol v. 1.1 [15]

HTTP is proposed for the transfer of data between Nodes in the network, using standard URLs. GET and POST requests can be used to both download and upload arbitrary amounts of data over TCP/IP connections.

The use of TCP itself, as opposed to UDP for current DNSBL lookups, introduces several low-level advantages. TCP is a connection-oriented protocol [16] that uses sequence number handshaking in connection establishment, which makes forging IP addresses nearly impossible (an important consideration for a system designed to withstand attacks). Because it's connection-oriented, TCP also ensures that unresponsive hosts are not bombarded with packets. UDP packets, on the other hand, often bombard unresponsive DNS servers (automatic retries) causing unintentional DoS attacks.

HTTP servers are already widely deployed on the Internet, running on small and large sites alike. This provides a stepping stone to easily deploying the distributed blocklist system, perhaps as a CGI application or HTTP server module. An important aspect of the distributed blocklist is having as many Nodes as possible participating, and HTTP provides a suitable generic infrastructure.

Finally, HTTP offers many extensions and features that can be very useful for inter-Nodal communications: virtual hosts, gzip stream compression, flexible headers, and even Transport Layer Security [17].

OpenPGP [14]

The OpenPGP specification, which is based on Philip Zimmermann's PGP [13], provides a standard message format used by interoperating PGP software such as the GNU Privacy Guard [18]. The specification describes the format for public/private keys as well as digital signatures, a vital component of the distributed blocklist.

An established message format standard is required for Nodes that may be running diverse software, since all must recognize and correctly interpret the Publisher's public key as well as digital signatures on Packages.

VII. Typical Use

In order to establish the blocklist system, a number of sites would install the appropriate software on their HTTP servers. Nodes would be configured according to available resources: a home broadband user, for instance, may wish to cache little data and provide mostly referrals, while a large ISP may cache all Packages and answer all public queries directly.

A Publisher would announce their blocklist name and public key, and people wishing to participate in this particular blocklist network would install the Publisher's public key on their own Nodes. Node operators would swap URLs with colleagues; Nodes with plentiful resources may publicly advertise their URLs to provide smaller Nodes with an entry point into the network. The Publisher can now send signed Packages to large Nodes and these Packages should become accessible across the network.

Someone wanting to *query* the blocklist would establish their own Node and run a local DNS server to provide DNSBL service. This way, the mail server software need not be modified.

Looking up an IP address in the blocklist involves determining the appropriate Package, checking local storage for the Package or going out to the network if the Package does not exist or has expired due to age (see Figure 2). Since each Package covers a significant portion of IP address space, few blocklist queries should result in actual P2P network traffic.

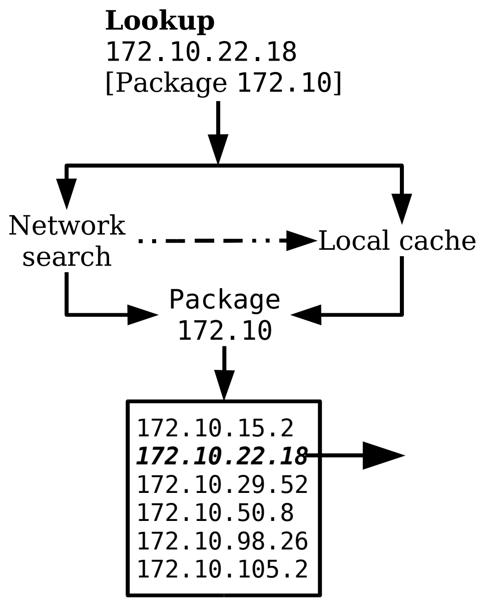

Figure 2. Looking up an IP address

VIII. Attack Scenarios

Scenario 1: Standard Denial-of-Service

If an attacker wants to perform a DDoS attack, they would target the higher-profile Nodes in the network. However, current network Nodes already know of plenty other Nodes by this time, meaning the attack would only cause difficulties for *new* Nodes trying to join the network. Packages are heavily duplicated across the network, meaning that all blocklist information should still be accessible. Announcing URLs for new Nodes publicly would quickly introduce new network entry points. A dynamic host cache could also be used to provide a large list of potential network entry points.

Scenario 2: Rogue Nodes

If a set of attackers want to disturb the network or tamper with blocklist data, they may attempt to establish a number of rogue Nodes. If these peers modify blocklist Packages, other Nodes will reject the Packages due to invalid digital signatures (which can not be forged without compromising the Publisher's private key). Nodes providing invalid data would be excluded from inter-Nodal referrals, effectively dropping the rogue nodes from the network.

Scenario 3: Attacking the Publisher

The Publisher's home site might become an attractive target for attackers. However, due to digital signatures on all Packages, any Node can be used to inject new data into the network. This further *provides anonymity* for the Publisher behind a fleet of Nodes.

IX. Implementation Alternatives

While a decentralized distribution system using HTTP can be built specifically for this blocklist application, alternative implementations that make use of existing P2P infrastructure may be more practical. One notable P2P network that already exists and which offers the required facilities is Gnutella, described by the open Gnutella protocols v0.4 and v0.6 [19]. Clients adhering to this protocol and communicating with neighbouring Nodes can share the (digitally signed) data Packages as described earlier. Gnutella further offers mechanisms to search for data and organize groups of Nodes.

Another implementation alternative relates to the direction of information flow. The system outlined in this paper describes unidirectional data flow from a Publisher to many clients. To better satisfy the needs of an anonymous public service, a distributed blocklist system may provide clients with a mechanism to send feedback data to the Publisher. This feedback is essential for refining the blocklist contents and correcting errors, without having to resort to a standard web site (which becomes another DDoS target). This bidirectional data flow between the Publisher and the clients can again be accomplished over a P2P network like Gnutella, protecting the Publisher from directed attacks.

X. Conclusion

The distributed spam blocklist system described in this paper provides a new way to publish blocklists that is more resistant to DDoS attacks than conventional DNSBLs. The proposed system further allows a number of cooperating hosts to pool resources, rather than placing large resource burdens on few servers. Both are significant advantages, given the challenges facing public anti-spam services.

The primary disadvantage of the proposed system is decreased speed and increased overhead of queries, compared to today's DNSBLs. Further work is required to establish whether the proposed distributed system can operate with acceptable performance. Nevertheless, the proposed structure and protocols should provide a useful basis for further development.

Acknowledgements

I would like to acknowledge the networking and attack-scenario advice, as well as general help offered by Dr. Robert McLeod of the University of Manitoba. I would also like to thank members of the mail administrator community for their valuable feedback.

References

[1] SORBS, Spam and Open Relay Blocking System, http://www.sorbs.net/, May 2004.

[2] P. Mockapetris, RFC 1034: Domain Names – Concepts and Facilities, ftp://ftp.rfc-editor.org/in-notes/rfc1034.txt, Nov 1987.

[3] W. Venema, Postfix, http://www.postfix.org/, May 2004.

[4] Sendmail Consortium, Sendmail Home Page, http://www.sendmail.org/, Dec 2002.

[5] SpamAssassin, SpamAssassin – Tests Performed, http://www.spamassassin.org/tests.html, May 2004.

[6] G. W. Herbert, Re: Re[2]: relays.osirusoft.com, http://www.merit.edu/mail.archives/nanog/2003-08/msg01145.html, Aug 2003.

[7] R. F. Guilmette, ANNOUNCE: MONKEYS.COM: Now retired from spam fighting, <vn1lufn8h6r38@corp.supernews.com> on USENET news.admin.net-abuse.email, Sept 2003.

[8] Spamhaus Project, The Spamhaus Project – dDoS Attacks, http://www.spamhaus.org/cyberattacks/, May 2004.

[9] SPEWS.ORG, Spam Prevention Early Warning System, http://www.spews.org/, May 2004.

[10] SpamCop.net, Inc., SpamCop.net, http://www.spamcop.net/, May 2004.

[11] J. Berkes, WPBL - Weighted Private Block List, http://wpbl.pc9.org/, Feb 2004.

[12] J. Berkes, Distributed HTTP server blocklist system, http://www.sysdesign.ca/archive/dhttp-bl.txt, Sept 2003.

[13] P. Zimmermann, *The official PGP user's guide* (Cambridge, MA: The MIT Press, 1995).

[14] J. Callas, L. Donnerhacke, H. Finney, R. Thayer, RFC 2440: OpenPGP Message Format, http://www.ietf.org/rfc/rfc2440.txt, Nov 1998.

[15] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, RFC 2616: Hypertext Transfer Protocol – HTTP/1.1, ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt, June 1999.

[16] J. Postel, RFC 793: Transmission Control Protocol, ftp://ftp.rfc-editor.org/in-notes/rfc793.txt, Sept 1981.

[17] R. Khare, S. Lawrence, RFC 2817: Upgrading to TLS Within HTTP/1.1, ftp://ftp.rfc-editor.org/in-notes/rfc2817.txt, May 2000.

[18] Free Software Foundation, Inc., GnuPG, http://www.gnupg.org/, May 2004.

[19] Gnutella Developer Forum, The GDF, http://groups.yahoo.com/group/the_gdf, July 2004.